Parametric Kernels for Sequence Data Analysis

Young-In Shin and Donald Fussell
The University of Texas at Austin
Department of Computer Sciences
{codeguru,fussell}@cs.utexas.edu

Abstract

A key challenge in applying kernel-based methods for discriminative learning is to identify a suitable kernel given a problem domain. Many methods instead transform the input data into a set of vectors in a feature space and classify the transformed data using a generic kernel. However, finding an effective transformation scheme for sequence (e.g. time series) data is a difficult task. In this paper, we introduce a scheme for directly designing kernels for the classification of sequence data such as that in handwritten character recognition and object recognition from sensor readings. Ordering information is represented by values of a parameter associated with each input data element. A similarity metric based on the parametric distance between corresponding elements is combined with their problem-specific similarity metric to produce a Mercer kernel suitable for use in methods such as support vector machine (SVM). This scheme directly embeds extraction of features from sequences of varying cardinalities into the kernel without needing to transform all input data into a common feature space before classification. We apply our method to object and handwritten character recognition tasks and compare against current approaches. The results show that we can obtain at least comparable accuracy to state of the art problem-specific methods using a systematic approach to kernel design. Our contribution is the introduction of a general technique for designing SVM kernels tailored for the classification of sequence data.

1 Introduction

A common technique in mechanical classification and regression is to define a feature space for the domain of interest, transform the input data into vectors in that space, and then apply an appropriate classification or regression technique in the feature space. However, some data are more naturally represented in a much more irregular form than feature vectors.

In this paper, we propose a new approach based on direct matching using *parametric kernels*, where inputs are variable length sequences of elements. Each element is associated with a point in a parameter space, and similarity metrics are defined independently for the elements and their parameters. The parameter space is decomposed into possibly overlapping ranges, and the overall similarity of the sequences is computed by taking the summation kernel over the elements and their associated parameters for each range. The intuition behind the use of parameters is to encode information about the ordering of input data in the kernel in a flexible way. Such information is frequently lost in feature-space based approaches. Since we can choose to apply a wide range of order-preserving transformations on the distances between elements in our parameterizations, we obtain additional control over the similarity metrics we can define as well. Note that our use of the term "parametric" here does not imply that we assume any sort of a priori parametric model of the data, as for example in the case of generative approaches [Jaakkola and Haussler, 1998]. Note also that sequences of any fixed dimensional feature vectors can be used instead of input data elements in our method as needed since all we assume about elements is that they are fixed-dimensional.

Consider an input $\mathbf{x}$ represented as a sequence of n elements $x_i \in X$, i.e. $\mathbf{x} = [x_1, \cdots, x_n]$. We could choose to associate each element x_i with parameter $\boldsymbol{\tau}(x_i)$ in parameter space T as follows

$$\boldsymbol{\tau}(x_i) = \begin{cases} \sum_{k=2}^{i} \|x_k - x_{k-1}\| & \text{if } i > 1, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

The associated parameters form a non-decreasing sequence of non-negative real numbers, starting from zero. For any two such sequences, a parametric kernel at each iteration picks one element from each of the sequences and computes the similarities between the elements and their associated parameters separately. These are multiplied to return an overall similarity between the elements. This product of the similarities of the elements and their parameters implies that two similar elements may contribute significantly to the overall sequence similarity only when their associated parameters are similar as well. This step is repeated for all pairs of elements from the two sequences, and the results are summed to return the overall sequence similarity.

Naïve application of this approach has the potential to be swamped by the computational expense of performing many comparisons between elements with widely divergent param-

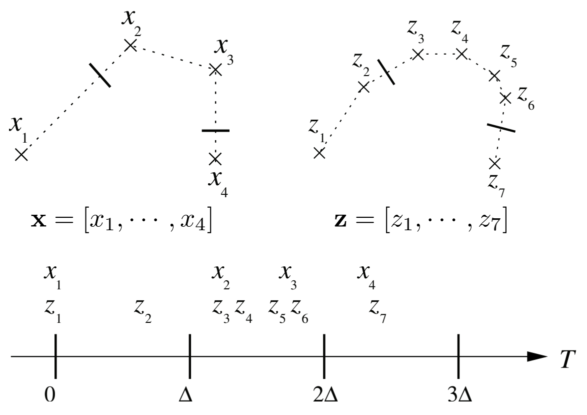

Figure 1: Mapping Sequence to Parameter Space

eters which contribute little or nothing to the final result. Intuitively, we could handle this by limiting the comparisons only to subsets of elements that are *close* in parameter space. This closeness in parameter space is easily specified by the decomposition of parameter space into ranges, so that close elements are defined to be those whose parameters fall into the same range. For instance, we could decompose T into non-overlapping intervals of equal length Δ, and elements from the two sequences are close if their associated parameters fall into the same interval. See Figure 1, where elements in $\mathbf{x}$ and $\mathbf{z}$ are grouped into three ranges based on the aforementioned decomposition scheme. x_1, for instance, will be compared against only z_1 and z_2, both in X and T. Any *sequence* data types fall into this category, e.g. handwritten characters, laser sensor readings, digital signals, and so on.

Ideally, we wish to find a method that meets all of the following requirements : a) computationally efficient to handle large inputs, b) no need to assume any probabilistic models, c) kernels are positive semi-definite, d) no requirement of fixed form inputs, and e) the ability to flexibly embed structural information from the input. While previous approaches fail to satisfy some or all of these, all of the requirements are satisfied by our approach. With an appropriate decomposition scheme, parametric kernels may be computed in time linear to the number of elements in inputs. Parametric kernels are positive semi-definite, thus they are admissible by kernel methods such as SVMs that require Mercer kernels for optimal solution.

We have applied our method to handwritten character recognition and object recognition from sensor readings. Our results compare favorably to the best previously reported schemes using only very simple similarity metrics.

2 Related Work

A great deal of work has been done on the application of kernel-based methods to data represented as vectors in a feature space [Shawe-Taylor and Christianini, 2004; Schölkopf and Smola, 2002]. Discriminative models can find complex and flexible decision boundaries and their performance is often superior to that of alternative methods, but they often rely on a heuristic preprocessing step in which the input data is transformed into a set of uniform size feature vectors. The similarity of two input data is then assessed by evaluating generic kernel functions on the feature vectors. To handle sequence data in this traditional kernel framework, features based on distance metrics are often extracted, e.g. the mean coordinates, second order statistics such as median, variance, minimum and maximum distances, area, etc. However, such features do not generalize well because they often impose restrictions that input patterns must be of equal lengths or sampled at equal rates. Alternatively, histograms can be constructed from locations, speed, size, aspect ratio, etc. Histograms are an effective scheme to map varying length sequences into uniform dimensional feature vectors [Porikli, 2004; Grauman and Darrell, 2005]. Unfortunately, structural information between elements in input patterns is inevitably lost in histogramming.

Methods based on direct matching do not suffer from such drawbacks as they retain the structural information. They map input sequences into sequences of equal dimensional *local* features, which are further compared using well known direct sequence matching techniques. For instance, a number of methods based on Dynamic Time Warping (DTW) have recently been proposed for classifying sequence data such as speech or handwritten characters [Bahlmann *et al.*, 2002; Shimodaira *et al.*, 2002]. Indeed, an extensive survey has shown that DTW methods are among the most effective techniques for classifying sequence data [Lei and Govindaraju, 2005; Keogh and Kasetty, 2002]. In [Bahlmann *et al.*, 2002], normalized coordinates and the tangent slope angle is computed at each of the points in a stroke sequence to form a feature vector. DTW computes the optimal distance between two sequences along a Viterbi path, which is then used as the exponent of a radial basis function (RBF). In [Tapia and Rojas, 2003], a fixed size feature vector is computed from strokes and an SVM classifier is used. They achieved a high accuracy over 98% but since they allowed only fixed feature vectors for each stroke, the method's application is quite limited. In [Lei and Govindaraju, 2005], Extended R-squared (ER^2) is proposed as the similarity measure for sequences. It uses coordinates of points directly as features, but can only operate on fixed-length windows of such points.

DTW methods define kernels which can be shown to be symmetric and satisfy the Cauchy-Schwartz inequality [Shimodaira *et al.*, 2002]. SVM classifiers may employ such kernels to classify sequential data. However, kernels based on DTW are not metrics. The triangle inequality is violated in many cases and the resulting kernel matrices are not positive semi-definite. Therefore, they are not admissible for kernel methods such as SVM in that they cannot guarantee the existence of a corresponding feature space and any notion of optimality with respect to such a space. In contrast, our parametric kernels are positive and semi-definite and are thus well-suited for use in kernel-based classifiers.

3 Approach

Kernel-based discriminative learning algorithms can find complex non-linear decision boundaries by mapping input examples into a feature space F with an inner product that can be evaluated by a kernel function $\kappa : X \times X \rightarrow \mathbb{R}$ on examples in the input space X. A linear decision function in F is found, which corresponds to a non-linear function in X. This "kernel trick" lets us find the decision boundary without explicit evaluation of the feature mapping function $\phi : X \rightarrow F$ and taking the inner product, which is computationally very expensive and may even be intractable. For this to guarantee a unique optimal solution, the kernel functions must satisfy Mercer's condition, i.e. the Gram matrix obtained from the kernel functions must be positive and semi-definite.

3.1 Parameters

The underlying intuition in our work is to associate a *parameter* with each of the elements so that enforcing parametric similarity is equivalent to the similarity in the structure of elements in the input patterns. Our work assumes that input patterns are varying length sequences of elements from some common input or feature space. For example, handwritten characters are sequences of 2D points, while images are 2D grids of fixed dimensional color values. The *structure* of a sequence is a 1D manifold of 2D vectors and that of an image is a 2D manifold of 3D vectors, assuming colors are represented as RGB for instance. A parameter of an element is then a point in this manifold that corresponds to the element. Therefore, if two parameters of any two elements are close, then they are structurally close. This parametric association must be defined as part of making the choice of kernel functions.

3.2 Parametric Kernel

Our input pattern $\mathbf{x}$ is a sequence of $|\mathbf{x}|$ elements, where each element x_i is a d-dimensional vector, i.e. $\mathbf{x} = [x_1, \cdots, x_{|\mathbf{x}|}]$ and $x_i \in \mathbb{R}^d$. The number of elements may vary across sequences. We associate each element with a parameter in parameter space T via a function $\boldsymbol{\tau} : \mathbb{R}^d \rightarrow T$. Consider a decomposition of T into N non-overlapping ranges

$$T = \bigcup_{t=0}^{N-1} T_t. \tag{2}$$

For instance, recall the earlier sequence example, where T was the set of non-negative real numbers and $\boldsymbol{\tau}$ is defined as (1). T was decomposed as shown in Figure 2.

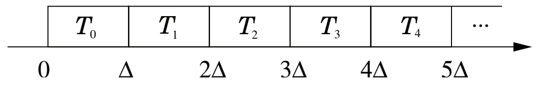

Figure 2: Parameter space is decomposed into non-overlapping ranges of length Δ.

For the derivation of parameter kernel functions, we first define *decomposed element set* for T_t as $\mathcal{I}_t(\mathbf{x}) = \{x_i | \boldsymbol{\tau}(x_i) \in T_t\}$, which is the set of elements of $\mathbf{x}$ whose associated parameters are in T_t. In our previous example shown in Figure 1, for instance, $\mathcal{I}_1(\mathbf{x}) = \{x_2, x_3\}$ and $\mathcal{I}_1(\mathbf{z}) = \{z_3, z_4, z_5, z_6\}$. We then compute a similarity for each range by taking a weighted sum of the similarities of every pair of elements of the sets being compared whose parameters fall within the range. For T_1, we will compare x_2 with z_3, x_2 with $z_4, \cdots$, and x_3 with z_6. The similarity for a given pair of elements is obtained by taking the product of the similarity $\kappa_{\boldsymbol{\tau}} : T \times T \rightarrow \mathbb{R}$ between those elements' parameters and a similarity $\kappa_x : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ defined directly on the elements themselves, each of which is a Mercer kernel function. Then, the feature extraction function $\boldsymbol{\phi}$ of a parametric kernel is defined as

$$\boldsymbol{\phi}(\mathbf{x}) = [\phi_0(\mathbf{x}), \phi_1(\mathbf{x}), \cdots \phi_{N-1}(\mathbf{x})], \tag{3}$$

where

$$\phi_t(\mathbf{x}) = \sum_{x_i \in \mathcal{I}_t(\mathbf{x})} w_{x_i} \kappa_x(x_i, \cdot) \kappa_{\boldsymbol{\tau}}(\boldsymbol{\tau}(x_i), \boldsymbol{\tau}(\cdot)), \tag{4}$$

and w_{x_i} is a non-negative weighting factor. Given two sequences $\mathbf{x} = [x_1, \cdots, x_{|\mathbf{x}|}]$ and $\mathbf{z} = [z_1, \cdots, z_{|\mathbf{z}|}]$, the parametric kernel function *before normalization* is then defined as the sum of similarity for all ranges :

$$\kappa(\mathbf{x}, \mathbf{z}) = \langle \boldsymbol{\phi}(\mathbf{x}) \cdot \boldsymbol{\phi}(\mathbf{z}) \rangle = \sum_{t=0}^{N-1} \langle \phi_t(\mathbf{x}) \cdot \phi_t(\mathbf{z}) \rangle, \tag{5}$$

where

$$\langle \phi_t(\mathbf{x}) \cdot \phi_t(\mathbf{z}) \rangle \tag{6}$$

$$= \sum_{\substack{x_i \in \mathcal{I}_t(\mathbf{x}) \\ z_j \in \mathcal{I}_t(\mathbf{z})}} w_{x_i} w_{z_j} \kappa_x(x_i, z_j) \kappa_{\boldsymbol{\tau}}(\boldsymbol{\tau}(x_i), \boldsymbol{\tau}(z_j)). \tag{7}$$

Note that since the product of κ_x and $\kappa_{\boldsymbol{\tau}}$ is taken in (4), both must score high to have significance in (5). Also note that no comparison is made between elements that are not from a common range. If, instead, we have to compare all elements from one input with all from the other input, we will face a number of undesirable consequences. For instance, in Figure 1, we will compare x_1 with $\{z_1, \cdots, z_7\}$, rather than $\{z_1, z_2\}$. This will result in a higher similarity value, which may be helpful for certain cases. But, at the same time, we are more likely to be confused by inputs where there are too many *far* elements, in which case we are swamped by bad comparisons. Of course, we may need dramatically more time to compute (5) as the number of kernel evaluations is significantly increased.

Parameter space decomposition solves such problems. However, such a decomposition scheme can introduce quantization errors. To overcome this problem, we allow the ranges to overlap. For instance, ranges in Figure 2 may overlap by $\Delta/2$, as shown in Figure 3. To suppress over-contribution of elements that fall into the intersections of ranges, we introduce weighting factors in (4).

The simplest weighting scheme is to take the average of the similarity at the overlapped regions. In this scheme, the default value for w_{x_i} is $1/|\mathcal{T}_{x_i}|$, where $\mathcal{T}_{x_i} = \{T_t | \boldsymbol{\tau}(x_i) \in T_t\}$. When no ranges overlap, we have $|\mathcal{T}_{x_i}| = 1$ and therefore, $w_{x_i} = 1$. Otherwise, overlapped ranges may yield

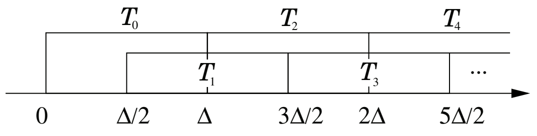

Figure 3: Ranges overlap by $\Delta/2$.

$w_{x_i} < 1$. For instance, with the decomposition scheme in Figure 3, at intersection $[\Delta/2, \Delta)$, we will set $w_{x_i} = 1/2$, since $|\mathcal{T}_{x_i}| = 2$. Note that this scheme will not result in $|\mathcal{T}_{x_i}| = 0$, i.e. $w_{x_i} \to \infty$, since (4) will be evaluated only when $\mathcal{I}_t(\mathbf{x}) \neq \varnothing$. If $\mathcal{I}_t(\mathbf{x}) = \varnothing$, then $\phi_t(\mathbf{x}) \equiv 0$ and the term is just ignored. Further discussion of different decomposition schemes is given in 3.3.

Finally, to avoid favoring large inputs, we normalize (5) by dividing it with the product of their norms,

$$\boldsymbol{\kappa}(\mathbf{x}, \mathbf{z}) = \frac{\kappa(\mathbf{x}, \mathbf{z})}{\sqrt{\kappa(\mathbf{x}, \mathbf{x})\kappa(\mathbf{z}, \mathbf{z})}}. \quad (8)$$

This is equivalent to computing the cosine of the angle between two feature vectors in a vector space model.

3.3 Parameter Space Decomposition Scheme

In this section, our decomposition scheme is discussed in general in terms of pros and cons with respect to additional cost of computation and changes in classification performance due to range overlapping and similarity weighting. Then, a number of different decomposition schemes are presented.

As mentioned above, decomposition lets us avoid swamping by bad comparisons and dramatically reduces the computational cost of kernel evaluation but introduces quantization error. This is alleviated by allowing for range overlapping and similarity weighting. However, overlapping must be allowed with care. Increasing the size of range overlaps will require additional computation since it is likely to involve more kernel evaluations as more elements are found in each intersection. The gain of decreasing quantization error may provide little improvement in classification performance if we are swamped by bad comparisons. Thus there is a tradeoff between quantization error and classification performance.

One issue left is the time to compute the decomposed element sets. The more complicated a decomposition scheme gets, the more difficult the implementation becomes and the more time it takes to run. Fortunately, our experimentation has revealed that classification performance is relatively insensitive to minor changes in the decomposition scheme. Therefore, we can often favor simpler decomposition schemes for ease of implementation and efficient kernel evaluation with the expectation of only minimal losses in performance.

We now present a number of example parameter space decomposition schemes.

Regular Decomposition Parameter ranges all have a common length Δ as in Figure 2 or 3. Implementation is simple, and it shows good classification performance in general. But we have limited freedom to fit the data.

Irregular Decomposition Parameter ranges are of varying lengths. Implementation is complicated and often decomposed element sets take longer to compute. But we can freely decompose the parameter space to better fit the data.

Multi-scale Decomposition Parameter ranges form a hierarchical structure at different resolutions. For instance, we may consider a decomposition where ranges form a pyramid as shown in Figure 4. Elements from non-adjacent ranges could be compared at coarser resolutions. Along with a proper weighting scheme, this may improve the performance. But implementation is more complex and kernel evaluation may take longer.

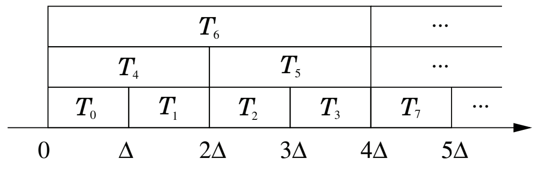

Figure 4: Pyramidal Parameter Space Decomposition

3.4 Mercer Condition

According to Mercer's theorem, a kernel function corresponds to an inner product in some feature space if and only if it is positive and semi-definite. A unique optimal solution is guaranteed by kernel methods only when kernels are positive and semi-definite. To see that our proposed method produces positive and semi-definite kernels, we first note that (4) is positive and semi-definite. In [Haussler, 1999], such kernels are called *summation kernels* and proven to be positive and semi-definite. It is not difficult to see that (5) is just a sum of summation kernels. Since we can synthesize a new Mercer kernel by adding Mercer kernels, (5) is a Mercer kernel.

3.5 Efficiency

The time complexity to compute parametric kernels depends greatly on the particular decomposition scheme used. Here we provide a brief analysis only for regular decomposition schemes.

Assume that constant time is needed to evaluate κ_x and $\kappa_{\boldsymbol{\tau}}$ and that we are using a regular decomposition scheme as in Figure 2 or 3. Then the time complexity of evaluating (5) for sequences $\mathbf{x}$ and $\mathbf{z}$ composed of $|\mathbf{x}|$ and $|\mathbf{z}|$ elements, respectively, is, on average, $O(|\mathbf{x}||\mathbf{z}|\Delta/L)$, where $L = \max(\boldsymbol{\tau}(x_{|\mathbf{x}|}), \boldsymbol{\tau}(z_{|\mathbf{z}|}))$. In the worst case without decomposition, $\Delta = L$, so the complexity is $O(|\mathbf{x}||\mathbf{z}|)$. In general, we expect decomposition to produce $\Delta \ll L$ since we would like to have only a reasonably small subset of elements in each range.

The storage complexity is $O(1)$ since we only keep the sum of kernel evaluations in memory. The time complexity to decompose the sequences into their respective element sets is $O(|\mathbf{x}| + |\mathbf{z}|)$, if constant time is taken for each element.

4 Results

For our handwritten character and object recognition experiments, an implementation based on Matlab SVM Toolbox in [Gunn, 1997] was used. For handwritten character recognition, we limited the scope of our testing to the recognition of isolated characters. We built our own training and test sets, similar to those in UNIPEN data[1].

4.1 Handwritten Character Recognition

Handwritten characters are represented as sequences of 2D (x, y) coordinates of pixels (points) on the screen. Our objective is to learn from a training set a function that correctly classifies an unseen handwritten character. We normalize by scaling inputs to fit a 300×300 pixel bounding box and aligning them at the center. The input data points in each sequence are chosen as the elements. Our training set is composed of 200 labeled examples created by two writers, each of whom wrote numeric characters from '0' to '9' ten times. Our test data is composed of 500 labeled examples created by other authors, 50 for each character. Figure 5 shows some training examples for characters '1' and '5', with the number of points in each of them shown below.

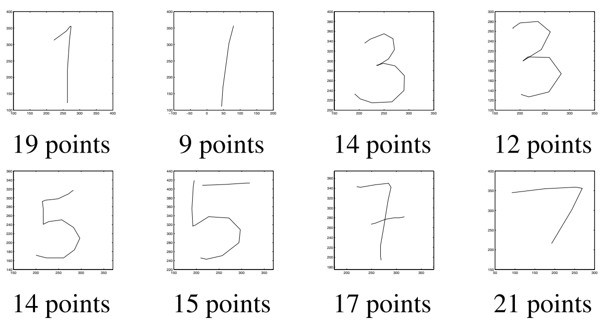

Figure 5: Sample Raw Inputs for Handwritten Numbers

We trained one-versus-all multi-class support vector novelty detectors (SVNDs) with the parametric kernel. See [Shawe-Taylor and Christianini, 2004; Schölkopf and Smola, 2002] for an introduction to SVND. Parameter space was regularly decomposed with overlapping allowed as shown in Figure 3. Window and hop sizes were set to 60 and 30, respectively. We chose RBF for κ_x and $\kappa_{\boldsymbol{\tau}}$ with $\sigma = 30$ and $\nu = 0.8$, where ν is the lower bound on the ratio of support vectors and at the same time the upper bound on the ratio of outliers. SNVD predicts how novel a test sample is in terms of novelty. Each of the test examples is classified as the class that has the minimum novelty value, and the result is compared against its label. The result is shown in Figure 6. The classification error was measured as the percentage of misclassified examples in the test set. We obtained an error rate less than 1% in most cases.

This represents a classification accuracy that is comparable to the recognizer in [Tapia and Rojas, 2003] and superior to those in [Lei and Govindaraju, 2005; Keogh and Kasetty, 2002; Bahlmann *et al.*, 2002]. However, unlike those approaches, we achieved this without any sophisticated feature extraction or restrictions on the input structures.

[1]http://unipen.nici.kun.nl/

Class	# SVs	Error	Class	# SVs	Error
'1'	17	0.98 %	'6'	18	1.15 %
'2'	18	1.12 %	'7'	17	0.41 %
'3'	15	0.10 %	'8'	19	0.23 %
'4'	19	0.89 %	'9'	14	0.09 %
'5'	17	1.01 %	'0'	18	0.01 %

Figure 6: Handwritten Number Recognition

4.2 Object Recognition from Sensor Data

We also analyzed sensor data captured at 0.1 second intervals by a Hokuyo URG-04LX laser range finder[2] mounted on the front side of a Segway RMP[3]. The Segway RMP navigates in its environment under the control of an attached tablet PC communicating with it via USB commands. Our tablet PC program reads the sensor data and detects nearby objects and obstacles during the navigation. The specific objective here is to locate a subregion in the sensor data that corresponds to a soccer ball. Each frame of input data is represented as an array of 768 regular directional samples in a range of $-120°$ to $120°$. Each of these is the distance to the nearest object in millimeters, ranging from 20 to 4095 with maximum 1% error. See Figure 7 (a) for a snapshot.

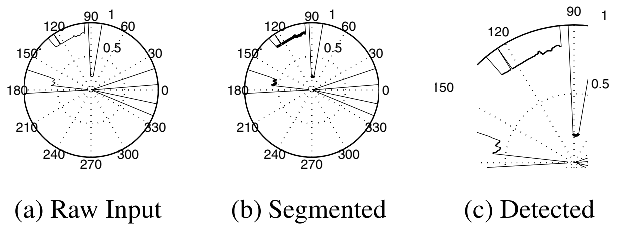

Figure 7: Raw sensor input is shown in (a) and thick curves in (b) are the blobs after segmentation. In (c), the thick round blob at angle about $90°$ and distance 0.2 is detected as the soccer ball.

After normalizing the sensor data by dividing it by the maximum distance, it is segmented into several subregions called blobs. A blob is a subregion of a frame where all distance values are in $(\theta, 1)$ and every consecutive values are at most δ apart. In this experiment, we used $\theta = 0.05$ and $\delta = 0.02$. Blobs are further normalized by scaling and translating so that the min and max distance values in each blob are 0 and 1.

Blobs are sequences of scalar values. Ball blobs resemble semi-circles distorted by some small sensor noise because the laser range finder scans from only one side of the ball. However, since this is also the case for all round objects such as human legs or beacons, they will confuse our classifier. For

[2]http://www.hokuyo-aut.jp/
[3]http://www.segway.com/products/rmp/

demonstration purposes, it is assumed that no such confusing objects exist in the scene. Some of the ball and non-ball examples are shown in Figure 8.

Our data is composed of examples for 442 ball blobs and 2199 non-ball blobs. Ball and non-ball blobs are labeled as +1 and -1, respectively. We trained a soft-margin support vector classifier (SVC) with a quadratic loss function and the parametric kernel. See [Shawe-Taylor and Christianini, 2004; Schölkopf and Smola, 2002] for an introduction to SVCs. We used a similar parameter space decomposition scheme as in the handwritten character recognition task. Window and hop sizes were set to 0.1 and 0.05, respectively. We chose RBF for κ_x and κ_τ with $\sigma = 0.1$ and $C = 1000$, where C is a positive number that defines the relative trade-off between norm and loss in regularized risk minimization. We measured the classification error from cross-validation. We randomly sampled 20 positive and 22 negative examples from our data for training, and the classification error was measured by the percentage of examples misclassified by the trained classifier among the rest of the data. On average, we obtained error rates of 0.71% and 0.92% for positive and negative test data, respectively, and 78.6% of the training data were support vectors. The error rate for non-ball blobs was a bit higher because of over-fitting when we treated all non-ball blobs as negative examples. Each scene on average contained about 10 to 20 blobs, among which only one corresponds to a soccer ball. Our C++ implementation running on a tablet PC with Intel Pentium M 1.2 GHz processor on average required less than 0.1 seconds to classify blobs in each scene.

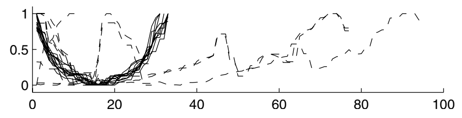

Figure 8: Solid (dashed) curves are ball (non ball) examples, where vertical and horizontal axis correspond to the normalized distance and the number of points in blobs. Clearly, ball blobs are semi-circular, while others are irregular.

5 Conclusions and Future Work

Our parametric kernel provides an efficient and very flexible framework for extending discriminative learning to sequential input data. Our approach allows for direct use of input data without computing sophisticated feature vectors, which makes the problem simpler to solve and easier and more efficient to implement. Thus we believe that this scheme often provides a more systematic, flexible, and intuitive way to build effective kernel functions for discriminative classifiers than previous methods while providing similar or superior performance.

While we have designed our techniques to apply to sequential input data, we believe that the advantages of a systematic approach that is free of the need for uniform input data can be applied to other types of data with natural input orderings such as grids, graphs, sets, etc. We believe that this is due to the intuitive binding of an application-dependent feature comparison scheme with kernel functions based on the natural ordering of the input data.

References

[Bahlmann *et al.*, 2002] C. Bahlmann, B. Haasdonk, and H. Burkhardt. On-line handwriting recognition with support vector machines - a kernel approach. In *Proceedings of the 8th International Workshop on Frontiers in Handwriting Recognition*, pages 49–54, 2002.

[Grauman and Darrell, 2005] K. Grauman and T. Darrell. The pyramid match kernel : Discriminative classification with sets of image features. In *IEEE International Conference on Computer Vision*, October 2005.

[Gunn, 1997] S. R. Gunn. Support vector machines for classification and regression. In *Technical Report*, University of Southampton, 1997.

[Haussler, 1999] D. Haussler. Convolution kernels on discrete structures. In *UC Santa Cruz Technical Report UCS-CRL-99-10*, 1999.

[Jaakkola and Haussler, 1998] T. S. Jaakkola and David Haussler. Exploiting generative models in discriminative classifiers. In *Advances in Neural Information Processing Systems, Vol. 11*, 1998.

[Keogh and Kasetty, 2002] E. Keogh and S. Kasetty. On the need for time series data mining benchmarks: A survey and empirical demonstration. In *SIGKDD*, pages 102–111, 2002.

[Lei and Govindaraju, 2005] H. Lei and V. Govindaraju. Similarity-driven sequence classification based on support vector machines. In *ICDAR*, pages 252–261, 2005.

[Porikli, 2004] F. M. Porikli. Trajectory distance metric using hidden markov model based representation. In *European Conference on Computer Vision (ECCV)*, May 2004.

[Schölkopf and Smola, 2002] B. Schölkopf and A. J. Smola. *Learning with Kernels - Support Vector Machines, Regularization, Optimization, and Beyond.* The MIT Press, Cambridge, MA, 2002.

[Shawe-Taylor and Christianini, 2004] J. Shawe-Taylor and N. Christianini. *Kernel Methods for Pattern Analysis.* Cambridge University Press, 2004.

[Shimodaira *et al.*, 2002] H. Shimodaira, K. Ichi, N. M. Nakai, and S. Sagayama. Dynamic time-alignment kernel in support vector machines. In *Advances in Neural Information Processing Systems, Vol. 14*, 2002.

[Tapia and Rojas, 2003] E. Tapia and Raúl Rojas. Recognition of on-line handwritten mathematical formulas in the e-chalk system. In *IDCAR*, 2003.