Nested Barycentric Coordinate System as an Explicit Feature Map

Lee-Ad Gottlieb Ariel University
Eran Kaufman Ariel University
Aryeh Kontorovich Ben-Gurion University
Gabriel Nivasch Ariel University
Ofir Pele SanDisk

Abstract

We introduce a new embedding technique based on a barycentric coordinate system. We show that our embedding can be used to transform the problem of polytope approximation into one of finding a *linear* classifier in a higher dimensional (but nevertheless quite sparse) representation. In effect, this embedding maps a piecewise linear function into an everywhere-linear function, and allows us to invoke well-known algorithms for the latter problem to solve the former.

We demonstrate that our embedding has applications to the problems of approximating separating polytopes — in fact, it can approximate any convex body and unions of convex bodies — as well as to classification by separating polytopes and piecewise linear regression.

1 Introduction

In this paper we introduce a new embedding technique which uses a barycentric coordinate system to embed input points into a higher-dimensional representation, where the target space is nevertheless quite sparse. The embedding, which we term the Nested Barycentric Coordinate System (NBCS), has the useful property that a piece-wise linear function in the origin space can be represented as a single linear function in the target space. We will demonstrate that the NBCS has application to multiple problems, including the following:

Approximating and learning polytopes. The problem of finding a simple polytope consistent with a labelled set is known to be computationally intractable (Khot and Saket, 2011), although better bounds are known for polytopes with margin (Gottlieb et al., 2018). Still, state-of-the-art algorithms for this problem feature a steep dependence on the inverse margin and half-space number defining the polytope. We will show that NBCS can be used to model this problem as the well-studied maximum-margin hyperplane problem. In fact, our technique can even solve the case where the labelled space is defined by a union of multiple disjoint polytopes.

The related problem of *learning* a separating polytope is known to require a large sample size (Goyal and Rademacher, 2009), although this too can be mitigated when the polytope has a large margin (Gottlieb et al., 2018). We show that NBCS can be used for this learning problem as well.

Function approximation and regression. The problem of computing a piece-wise linear approximation to an input function is of significant mathematical and numerical interest. The learning counterpart of this problem is to model a finite number of function observations using a piece-wise linear regressor, while avoiding overfitting. However, algorithms for these problems often scale poorly with increased dimension or observation size (Hannah and Dunson, 2013). As with polytope approximation, we can show that NBCS can be used to transform the problem of finding a piece-wise linear approximator or regressor to that of finding a single linear approximator or regressor.

Techniques. Our proposed embedding technique partitions the input space into a nested hierarchy of simplices, and then embeds each data point into features corresponding to the barycentric coordinates of its containing simplex. This yields an explicit feature map, and allows us to fit a linear separator (or train a linear classifier) in the rich feature space obtained from the simplices.

For sample size n in d-dimensional space, our method has runtime $O(d^2 n)$ regardless of the dimension of the embedding space (when the approximation parameter is taken to be fixed, see Sections 5 and 7 for exact bounds). In contrast, standard polynomial embedding techniques for p-degree polynomials typically run

Proceedings of the 24[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2021, San Diego, California, USA. PMLR: Volume 130.

in time $O(d^p n)$. Likewise, typical kernel embedding have runtime $O(dn^2)$, and so scale poorly to big data. Our embedding technique allows for highly non-linear decision boundaries, although these are linear within each simplex (and hence piecewise-linear overall), as explained in Section 3. At the same time, our approach is sufficiently robust to closely approximate realizable convex bodies or functions for any given error in only linear time for a fixed dimension (Section 4). We also give generalization bounds based on empirical margin (Theorem 5.1) and a novel hybrid sample compression technique (Theorem 5.2). Finally, we perform an extensive empirical evaluation in which our method compares favorably with a wide range of other popular kernel and embedding methods, for classification and regression (Section 7).

2 Related Work

Approximating convex polytopes. Learning arbitrary convex bodies requires a very large sample size (Goyal and Rademacher, 2009), and so we focus instead on convex polytopes defined by a small number of halfspaces. However, the problem of finding consistent polytopes is known to be NP-complete even when the polytope is simply the intersection of two hyperplanes (Megiddo, 1988). In fact, Khot and Saket (2011) showed that "unless NP = RP, it is hard to (even) weakly PAC-learn intersection of two halfspaces", even when allowed the richer class of $O(1)$ intersecting halfspaces. Klivans and Sherstov (2009) showed that learning an intersection of n^ε halfspaces is intractable regardless of hypothesis representation (under certain cryptographic assumptions). These negative results have motivated researchers to consider the problem of discovering consistent polytopes which have some separating margin. Several approximation and learning algorithms have been suggested for this problem, featuring bounds with steep dependence on the inverse margin and number of halfspaces forming the polytope (Arriaga and Vempala, 2006; Klivans and Servedio, 2008; Gottlieb et al., 2018; Goel and Klivans, 2018).

In contrast, we show in Section 4 that our method is capable of approximating any convex polytope in time independent of the halfspace number: We can achieve linear runtime (in fixed dimension) with mild dependence on the inverse margin, or quadratic runtime with only polylogarithmic dependence on the inverse margin. We accomplish this by finding a linear separator in the higher-dimensional embedded space, and projecting the solution back into the origin space. However, our approach is not strictly comparable to those above, as they are concerned with minimizing the disagreement between the computed polytope (or object) and the true underlying polytope with respect to the point space, while we minimize the volume of the space between the polytopes.

Embeddings and Kernel maps. Finding piecewise linear functions, as of today, have no embedding or kernel trick. Kernel methods provide two principal benefits over embedding techniques: (1) They implicitly induce a non-linear feature map, which allows for a richer space of classifiers and (2) when the kernel trick is available, they effectively replace the dimension d of the feature space with the sample size n as the computational complexity parameter. As such, these are well-suited for the 'high dimension, moderate data size' regime. For very large datasets, however, naive use of kernel methods becomes prohibitive. The cost is incurred both at the training stage, where an optimal classifier is searched for over an n-dimensional space, and at the hypothesis evaluation stage, where a sum of n kernel evaluations must be computed. For these reasons, for large data sets, *explicit feature maps* are preferred. Various approximations have been proposed to mitigate the computational challenges associated with explicit feature maps, including Chang et al. (2010); Maji et al. (2012); Perronnin et al. (2010); Rahimi and Recht (2007); Vedaldi and Zisserman (2012); Li et al. (2010); Shahrampour and Tarokh (2018); Chum (2015); Zafeiriou and Kotsia (2013). Kernel approximations for explicit feature maps come in two basic varieties:

Data-dependent kernel approximations. This category includes Nystrom's approximation (Williams and Seeger, 2000), which projects the data onto a suitably selected subspace. If $K(x, z_i)$ is the projection of example x onto the basis element z_i, the points $\{z_1, \ldots, z_n\}$ are chosen to maximally capture the data variability. Some methods select z_i from the sample. The selection can be random (Williams and Seeger, 2001), greedy (Smola and Schökopf, 2000), or involve an incomplete Cholesky decomposition (Fine and Scheinberg, 2001). Perronnin et al. (2010) applied Nystrom's approximation to each dimension of the data independently, greatly increasing the efficiency of the method.

Data-independent kernel approximations. This category includes sampling the Fourier domain to compute explicit maps for translation invariant kernels. Rahimi and Recht (2007, 2009) do this for the radial basis function kernel, also known as Random Kitchen Sinks. Li et al. (2010); Vedaldi and Zisserman (2012) applied this technique to certain group-invariant kernels, and proposed an adaptive approximation to the χ^2 kernel. Porikli and Ozkan (2011) map the input data onto a low-dimensional spectral (Fourier) feature space via a cosine transform. Vempati et al. (2010) proposed a skewed chi squared kernel, which allows for a simple Monte Carlo approximation of the feature map. Maji

et al. (2012) approximated the intersection kernel and the χ^2 kernel by a sparse closed-form feature map. Pele et al. (2013) suggested using not only piecewise linear function in each feature separately but also to add all pairs of features. Chang et al. (2010) conducted an extensive study on the usage of the second-order polynomial explicit feature map. Bernal et al. (2012) approximated second order features relationships via a Conditional Random Field model.

SVM Decompositions. Simplex decompositions have been used to produce proximity-based classifiers (Belkin et al., 2018; Davies, 1996), but to the best of our knowledge, ours is the first work to utilize either nested simplex decompositions or barycentric centers in conjunction with SVM. Simplex decompositions are related to the quadtree, and the quadtree has been used together with SVM for various learning tasks (Saavedra et al., 2004; Beltrami and da Silva, 2015), but not for the creation of kernel embeddings. Simplex decompositions are more efficient than quadtrees, since a simplex naturally decomposes into only $d+1$ sub-simplices (Section 3), while a quadtree cell naturally decomposes into 2^d sub-cells.

As mentioned, our emphasis in this paper is specifically on explicit feature maps, but there are numerous approaches to reducing kernel SVM runtime (for example the CoreSVM of Tsang et al. (2005, 2007)). Another related paradigm is that of *Local SVM* (Hao Zhang et al., 2006; Gu and Han, 2013), which assumes continuity of the labels with respect to spacial proximity; similarly labeled points tend to cluster together. This differs from the underlying assumption motivating kernel SVM, which assumes that the data is approximately linearly separable, but not necessarily clusterable. These approaches find success in distinct settings, and are incomparable.

Regression and function approximation. Univariate piecewise linear regression is defined as a union of lines corresponding to distinct segments of the x axis. Given a partition of the axis into K segments (defined by endpoints $a_1, \ldots, a_k$) and associated weights $w_{k=1}^K$ and values $b_{k=1}^K$, define a class of functions function $g_k(x)$ as $g_k(x) = x$ when $x \in [a_k, a_{k+1}]$ and zero otherwise. The hypothesis takes the form:

$$\tilde{f}(x) = \sum_{k \in [K]} w_k g_k(x) + b_k \tag{1}$$

Typically the objective is to reduce the mean square error between the target function and its piecewise linear approximation. The extension to higher dimensions is far from trivial, as the choice of high-dimensional partition is not immediate, and potentially quite complex. But if we add assumptions on $f(\cdot)$ such as monotonicity and convexity, then the problem can take the form of a convex program, yielding a convex regression problem:

$$\tilde{f}(x) = \max_{k \in [K]} w_k^T \cdot x + b_k. \tag{2}$$

Here $w_k \in \mathbb{R}^d$ and $\tilde{f}(x)$ is a piecewise linear convex function whose knots (split points) are parametrically estimated.

This convex formulation has a computational complexity of $O((d+1)^4 n^5)$ (Monteiro and Adler (1989)), which quickly becomes impractical. Some methods include an objective function constrained to a set of convex functions only for each pair of observations (Hildreth (1954); Kuosmanen (2008); Seijo and Sen (2011); Allon et al. (2007); Lim and Glynn (2012)) or semidefinite constraints over all observations (Aguilera and Morin (2008); Wang and Ni (2012)). Aguilera et al. (2011) proposed a two step smoothing and fitting process. First, the data is smoothed and functional estimates are generated over an ϵ-net over the domain. Then the convex hull of the smoothed estimate is used as a convex estimator. Although this algorithm does scale to larger data sets, it does not scale gracefully to large dimension due to the ϵ-net dividing each dimension separately into K partitions, thereby resulting in a new embedding space of order $O(d^K)$. Hannah and Dunson (2011) proposed a Bayesian model that placed a prior over the set of all piecewise linear models. They were able to show adaptive rates of convergence, but the inference algorithm did not scale to more than a few thousand observations. Koushanfar et al. (2010) transformed the ordering problem associated with shape constrained inference into a combinatorial optimization problem which was solved with dynamic programming; this scales to a few hundred observations. Magnani and Boyd (2009) proposed to divide the data into K random subsets and a linear model was fit within each subset; a convex function was generated by taking the maximum over these hyperplanes. Hannah and Dunson (2013) introduced convex adaptive partitioning , which creates a globally convex regression model from locally linear estimates fit on adaptively selected covariate partitions. Our work suggests a different approach by embedding the problem directly into a Hilbert space via an explicit feature map. In the embedded space, the regression problem is a non-constrained linear regression problem instead of a piecewise regression problem. However, after solving the problem in the embedded space, the projection back to the original space creates a smooth, convex, continuous and differentiable piecewise linear manifold.

3 The Nested barycentric coordinate system

Here we describe the nested barycentric coordinate system. We explain its construction and description, how to embed a point from the origin space into the new coordinate system, and how a point in the embedded system can be projected back into the origin space (Section 3.1). We then show that if we associate a weight with each simplex point, then the embedding and weights together imply some (not necessarily convex) polytope on the origin space (Section 3.2). Later in Section 4 we will show that this system is sufficiently robust that it can be used to approximate any convex body.

3.1 Nested barycentric embedding

Let $S \subset \mathbb{R}^d$ be a regular simplex of unit side-length, and let $(q_0, \ldots, q_d)$ be its vertices. Each point x inside the simplex can be written using the barycentric coefficients:

$$\begin{aligned} x &= \sum_{i=0}^{d} \alpha_i q_i \\ \sum_{i=0}^{d} \alpha_i &= 1 \quad 0 \le \alpha_i \le 1 \end{aligned} \tag{3}$$

Here α_i denotes the coefficient of point x corresponding to vertex q_i. Denote the vector of α's corresponding to x by $\phi_{d+1}(x) = (\alpha_0, \ldots, \alpha_d)$. This vector can be computed by letting

$$\phi_{d+1}(x) = Q^{-1} \cdot (x, 1)^T, \tag{4}$$

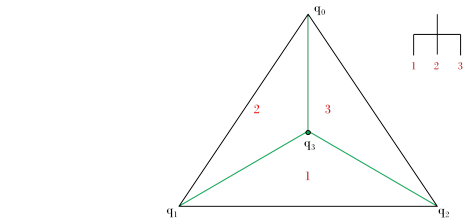
Where Q is the $(d+1) \times (d+1)$ matrix whose ith column is $(q_i, 1)^T$.

We can further refine the system by introducing a new point q_{d+1} inside the simplex, thereby inducing a partition of the simplex into $d+1$ new sub-simplices, $S_1, \ldots, S_{d+1}$. We order the coordinates of our system as $(q_0, \ldots, q_{d+1})$. A point x inside the system is embedded by first identifying the sub-simplex S_i containing x, and then utilizing the $d+1$ vertices of S_i to compute the barycentric coefficients (the α's) of equation (3). Then x is assigned a vector wherein each coordinate corresponding to each vertex of S_i is set to the coefficient of that vertex, and the remaining coordinates is set equal to 0. This defines the embedding $\phi_{d+2}(x) : \mathbb{R}^d \to \mathbb{R}^{d+2}$.

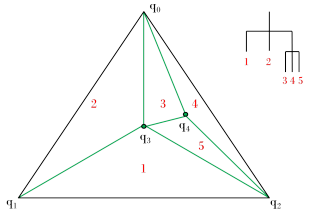
The refinement process can be continued by choosing points inside simplices to further split these simplices. In general, after a sequence $Q_t = (q_0, ..., q_t)$ of $t+1$ points have been chosen, we have a nested architecture represented by a $(d+1)$-ary tree B_t. Each node v of B_t is labeled by a $(d+1)$-tuple of points of Q_t, which are the vertices of a simplex $s(v) \subseteq S$. The root of the tree is labeled by the vertices $q_0, ..., q_d$ of the original simplex S. The simplices corresponding to the leaves of B_t form a partition of S. Given a new point q_{t+1}, we form B_{t+1} from B_t by first traversing the tree from the root down, choosing the nodes who's simplices contain q_{t+1} until we find the leaf v whose simplex contains q_{t+1}. Then we add $d+1$ children to v, each one labeled by a different $(d+1)$-tuple in which q_{t+1} replaces a vertex of v.

The tree B_t defines an embedding $\phi_t(x) : \mathbb{R}^d \to \mathbb{R}^t$ as follows: Given x, let v be the leaf of B_t whose simplex $s(v)$ contains x. Let $q_{i_0}, ..., q_{i_d}$ be the vertices of $s(v)$. Write $x = \Sigma \alpha_{i_j} q_{i_j}$ as a convex combination of these vertices, and let $\alpha_k = 0$ for all other points q_k. Then $\phi_t(x) = (\alpha_0, ..., \alpha_t)$. See Figure 1. (If x lies on the boundary of several leaf simplices, then we can choose any of them, since the resulting embedding $\phi_t(x)$ will be the same in all cases.)

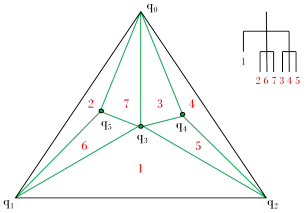
We note that the embedding — the *nested barycentric coordinate system* — is sparse, as at most $d+1$ coefficients are non-zero, and also that the embedded points lie on the L_1 sphere ($\sum \alpha = 1$).



(a) step 1



(b) step 2.1



(c) step 2.2

Figure 1: Creation of the nested system

A point in the embedded space can be projected back into the original space by

$$x = \sum_{i=0}^{t} \alpha_i q_i. \tag{5}$$

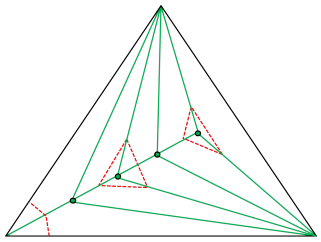


Figure 2: An example of a border polytope and two other disjoint polytopes corresponding to $w \cdot \phi_t(x) = 0$.

3.2 Weights, hyperplanes and polytopes

Given an embedding, we will assign a sequence of real-valued weights $w = (w_0, \ldots, w_t)$ to the vertices $(q_0, \ldots, q_t)$. Then the set of points R such that

$$R = \{x \in S : w \cdot \phi_t(x) \geq 0\} \tag{6}$$

is a union of interior-disjoint convex regions, each of which equals the intersection of one of the simplices with a halfspace (see Figure 2 for an illustration).

Lemma 3.1. *Any hyperplane that crosses a single simplex can be defined by a sequence of weights $w = (w_0, \ldots, w_d)$, such that all points x that lie on the hyperplane satisfy the equation $w \cdot \phi_{d+1}(x) = 0$.*

Proof. Choose d affinely independent points on the given hyperplane. These points have a unique set of barycentric coefficients of the containing simplex, and thus a unique representation. Finding the desired weights is equivalent to solving $A \cdot w = 0$, where A is a matrix of dimension $d \times (d+1)$ whose rows are the embeddings $\phi_{d+1}(x)$ of the points. Since this homogeneous linear system has more unknowns than equations, it has a nontrivial solution. □

In Section 4 we will show that a simple nested barycentric system, together with a prudent choice of weights, can be used to closely approximate any given convex body. To this end, we will require a useful property of these systems — essentially, that splitting a simplex cannot decrease the expressiveness of the system. For this end, it is enough to show the following:

Theorem 3.2. *Let $B_{d+1} = (q_0, \ldots, q_d)$ consist of a single simplex P, let $w = (w_0, \ldots, w_d)$ be a sequence of weights, and let $H = \{x \in P : w \cdot \phi_{d+1}(x) = 0\}$. Let $q_{d+1} \in P$ be a split point, and let $B_{d+2} = (q_0, \ldots, q_{d+1})$. Then there exists a weight w_{d+1} for q_{d+1} such that, letting $w' = (w_0, \ldots, w_{d+1})$, we have*

$$\{x \in P : w' \cdot \phi_{d+2}(x) = 0\} = H \qquad \text{for all } x \in P.$$

Proof. We will show that the desired weight is $w_{d+1} = \phi_{d+1}(q_{d+1}) \cdot w$. Indeed, let

$$\begin{aligned} \phi_{d+1}(x) &= (\alpha_0, \ldots, \alpha_d), \\ \phi_{d+2}(x) &= (\beta_0, \ldots, \beta_{d+1}), \\ \phi_{d+1}(q_{d+1}) &= (\gamma_0, \ldots, \gamma_d). \end{aligned}$$

Then

$$x = \sum_{i=0}^{d+1} \beta_i q_i = \sum_{i=0}^{d} (\beta_i + \beta_{d+1}\gamma_i) q_i.$$

Since the barycentric representation is unique, this implies that $\alpha_i = \beta_i + \beta_{d+1}\gamma_i$ for all $i \leq d$. Hence,

$$\begin{aligned} w' \cdot \phi_{d+2}(x) &= \sum_{i=0}^{d} \beta_i w_i + \beta_{d+1} \sum_{i=0}^{d} \gamma_i w_i \\ &= \sum_{i=0}^{d} \alpha_i w_i \\ &= w \cdot \phi_{d+1}(x), \end{aligned}$$

as claimed. □

4 Convex body approximation

In this section prove that a convex manifold can be approximated in the Hausdorff distance sense using NBCS. Given a Hausdorff distance of ε an approximate polytope $\tilde{P}$ can be constructed in $O(ln(\frac{1}{\varepsilon}))$. Also given a convex body with margin ε a consistent polytope $\tilde{P}^{(-\varepsilon)}$ can be constructed to fall within the ε margin using even the simplest method of equal volume subdivisions. We have also proven that a convex body or a function can be approximated in the MSE sense using a more elaborate Archimedean style subdivision. The first approximation is more useful for classification tasks and the second for regression and function approximation using piecewise linear functions.

Details of these proofs are presented in Section A,B of the Supplementary Material.

5 Learning algorithms

In Section 4, we demonstrated that the uniform subdivision embedding, coupled with an appropriate choice of weights, can represent an approximation to any given convex body. This motivates an embedding technique for a linear classifier.

For some parameter q (determined by cross validation), our classification algorithm produces a q-stage uniform subdivision: Beginning with a single simplex covering the entire space, at each stage we add to the system the barycentric center of each simplex, thereby splitting all simplices into $d+1$ sub-simplices. We call a set

of $d+1$ simplices formed by a split *siblings*. The procedure stops after q stages, having produced $(d+1)^q$ simplices. We note that there is nothing to be gained by splitting an empty simplex, so the algorithm may ignore these (and this may also help us avoid creating thin simplices). Then an empty simplex must have a sibling that contains points, and since a simplex has d siblings, we have that the total number of simplices is not greater than $\min\{(d+1)^q, dnq\}$.

Parameter q is analogous to depth parameter s of Lemma A.4; however, we have consistently observed by empirical cross-validation that it suffices to take q as a very small constant (at most 5), and so we stipulate in our algorithm that q be bound by a small universal constant.

Having computed the nested coordinate system, we use it to embed all points into high-dimensional space. To find an appropriate weight assignment w for the simplex points, we compute a linear classifier on the embedded space to separate the data. A linear classifier takes the form $h(x) = \text{sign}(w \cdot x)$, and this w serves as our weight vector for the embedding – we note that as w is computed globally over points (and not locally as was done in Section 4), we expect it to produce a smoother classifier with no rigid spikes. We use soft SVM as our linear classifier, and note that the training phase can be executed in time $O(dn)$ on $(d+1)$-sparse vectors (Joachims, 2006). The total runtime of the algorithm is bounded by the cost of executing the sparse SVM plus the total number of simplex points, that is

$$O(\min\{d(d+1)^q+dn, d^2qn\}) = \min\{d^{O(1)}+dn, O(d^2n)\}.$$

To classify a new point, we can simply search top-bottom for its lowest containing simplex: We begin at the initial simplex, investigate which of its d sub-simplices contains the query point, and iterate on that simplex. This can all be done in time $O(qd^2) = O(d^2)$. After bounding the run time, we want to bound the out of sample error:

Theorem 5.1. *If our classifier achieves sample error $\hat{R}$ with margin γ (i.e., $\hat{R}$ is the fraction of the points whose margin is less than γ) on a sample of size n after stopping at stage q, its generalization error R is bounded by*

$$\hat{R} + O(1/(\gamma\sqrt{n}) + \sqrt{\log(q/\delta)/n}) \qquad (7)$$

with probability at least $1-\delta$.

This bound is a consequence of the SVM margin bound (Mohri et al., 2012, Theorem 4.5) and the stratification technique (Shawe-Taylor et al., 1998), where the q-th stage receives weight $1/2^q$.

Adaptive splitting strategies. The above algorithm is data-independent in its selection of split points. It is reasonable to suggest that a data-dependent choice of split points can improve the performance of the learning algorithm, and this too may be less prone to creating thin simplices. Several greedy strategies suggest themselves, but after empirical trials we suggest the following split heuristic: At every stage, a linear classifier of the embedding space is computed. For each simplex, we identify the points in the simplex have been misclassified so far, and choose a data point which is closest to the barycentric center *of the misclassified points*. As before, we limit the heuristic to a constant number of stages, and it is also not necessary to subdivide an empty simplex, or one that contains not many misclassified points. (See Section 7 for empirical results.) The following bounds follow from Corollary C.1.2:

Theorem 5.2. *If our adaptive classifier achieves sample error $\hat{R}$ with margin γ (i.e., $\hat{R}$ is the fraction of the points whose margin is less than γ) on a sample of size n after stopping at stage q and retaining k split points, its generalization error R is bounded by*

$$\hat{R} + O(1/(\gamma\sqrt{n-k}) + \sqrt{\log(q/\delta)/(n-k)}) \qquad (8)$$

with probability at least $1-\delta$.

We note that in the above algorithms, computations done on individual simplices are easily parallelizable.

NBCS Regression. Having shown in Section 4 how to use NBCS to approximate convex functions, we can now apply this tool to regression. Consider a regression problem with $y_i = f(x_i) + \epsilon_i$, where $y_i \in \mathbb{R}$, $x_i \in \mathbb{R}^d$ and $\epsilon_i \sim N(0, \sigma^2)$ ($\sigma \leq 1$) is an error term. Theorem B.2 implies that the NBSC embedding reduces the mean square error at every step, and that the number of steps can be bounded by $O(\log \frac{1}{\sigma})$. In order to find an appropriate weight assignment w for the simplex points, we compute a linear regressor on the embedded space, where the regressor is of the form $h(x) = w \cdot x + b$. Following the algorithm for concave function approximation, we can suggest after each step calculating $f'(x) = f(x) - \sum w_i\phi(x)$ and taking the data point within each new simplex which maximizes the value of $f'(x)$. If $f'(x)$ evaluated at the knot is less than the predefined constant parameter ϵ, then there is nothing to be gained by further splitting this simplex. This causes the algorithm to make more splits in the 'curved' parts of the manifold than on the nearly linear ones.

6 Experiments

Our embedding technique is *motivated* by provable bounds for convex polytopes, but we find that it is sufficiently robust to yield impressive empirical results for general piecewise linear functions with maximal margin. All experiments utilized the python scikit-learn library (Pedregosa et al., 2011) [1]. The SVM regularization parameter C was 5-fold cross-validated over the set $\{2^{-5}, 2^{-3}, \ldots, 2^{15}\}$, and for the RBF kernel, the γ parameter was five-fold cross-validated over the set $\{2^{-15}, 2^{-3}, \ldots, 2^{3}\}$. For polynomial kernel we cross-validated the polynomial exponent d over the set $\{2, \ldots, 10\}$.

For our methods, the maximum iteration parameter q was cross validated over the set $\{2, \ldots, \log_d(n)\}$. Our algorithms usually converged even before reaching the maximum number of allowed iterations.

Polytope approximation. Before presenting the experiments, we give a simple example that illustrates the power of our approach in approximating non-convex polytopes. We created a random data-set wherein all positive examples were taken from within a 5-gon in $2D$ and the negative points from outside it. This data was randomly generated within the unit circle: For each halfspace, we sampled a random direction vector w_j uniformly from the unit sphere, and then sampled a random offset value $b_j \in [.05, .95]$ to produce the halfspace (w_j, b_j). The intersection of these halfspaces is the target polytope. All data points inside the polytope with margin 0.05 were labeled as positive, all data points outside the polytope with margin 0.05 were labeled as negative, and the rest were discarded.

Figure 3 shows the iterative boundary formation, where the bold black line is the decision boundary and the dotted lines are the margin $(w \cdot \phi_t(x) = \pm 1)$. For each iteration, the nested barycentric system is illustrated by the red lines. A consistent approximation of the underlying polytope for multiple runs was achieved after only 3 iterations. Notice how the margins become smaller at each iteration until reaching their predetermined size, and also that the constructed polytope is non-convex, although there exists a convex polytope consistent with the points.

For our regression simulation we chose a piecewise linear function consisting of four lines with noise. As in Birke and Dette (2007), we ran simulations with uniformly distributed design points for the explanatory variables and added normal noise with standard deviation $\sigma = 0.05$ to the response variable. Figure 4 shows the result of $\{1, 2, 4\}$ steps along with $\log_{10}$-normalized MSE for

[1] code can be found at https://github.com/erankfmn/NBCS-embedding

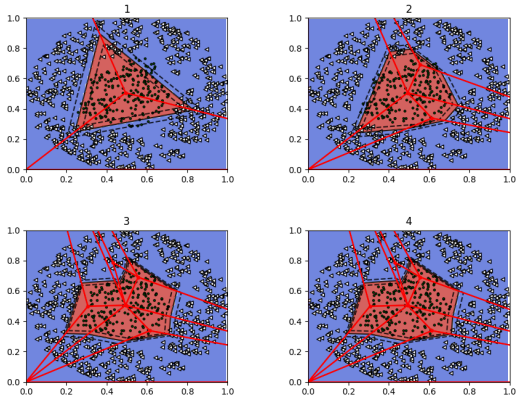


Figure 3: Learning a polytope separating the red and blue points. Note that by the third step the polytope is consistent with data, so that no change in the embedding is effected by the fourth step.

all different steps from $1-6$. Notice how the algorithm chooses the knots on the vertices of the polytope and does not add more knots within the linear parts of the edges. Stipulating a minimum inter-point distance serves to prevent the creation of simplices with very thin volume (spikes) and also acts as a regulator to prevent overfitting.

7 Benchmarks

We compared our method on real-world datasets with varying sizes and dimensions (see Table 1). We first confirmed that our runtime on large datasets is competitive with other feature map methods and Core-SVM (Table 2). We then compared our results to other explicit feature map methods discussed in Section 2. Here we focused only on accuracy, since all these methods have similar runtime complexity. Figure 5 demonstrates a comparison of the average accuracy between our embedding technique (adapt-NBCS), Kitchen Sink (KS) (Rahimi and Recht, 2007), Nystrom's approximation (Williams and Seeger, 2000) and the adaptive χ^2 (Vedaldi and Zisserman, 2012), all of which have open source implementations within the scikit-learn library. (We note that Nystrom, KS and CoreSVM are all RBF kernel approximations using subsampling techniques.) We included the accuracy achieved by RBF and polynomial SVM in kernel mode. We also included the accuracy of random forests (RF), a technique which tied for first place in comparisons made by Delgado et al. (2014) over the entire UCI dataset. Random forest is a good comparison to our method since much like our method it will create piecewise linear boundaries, but without maximal margin. Our algorithm's

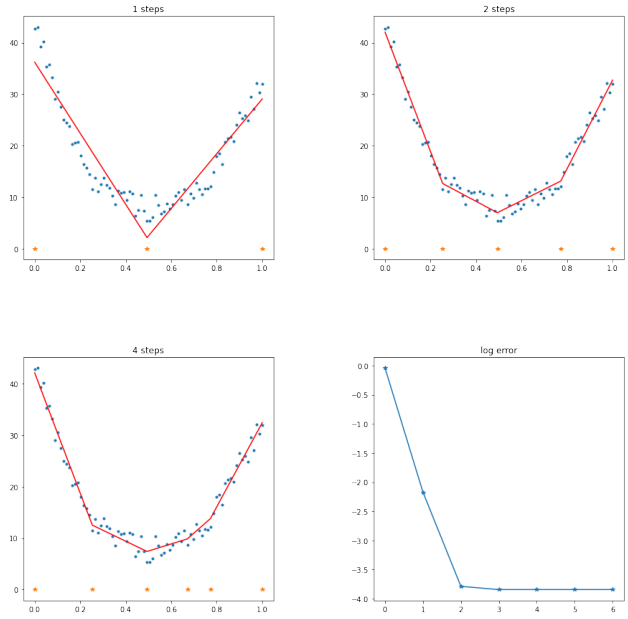


Figure 4: Function approximation over a partial linear function with noise and the overall log error. Note that the adaptive splitting did not add any splits to the linear parts once they were found.

accuracy compared favorably with the others.

For regression purposes, we compared our method to other embedding and convex regression methods on real-world datasets. Table 3 demonstrates a comparison of the the squared correlation coefficient (R^2) of our embedding technique (NBCS) against othe embedding and kernel techniques as mentioned above

over a large variety of datasets, varying in size and dimension. The results show that NBCS compared favorably to other methods for all datasets.

8 Discussion and future work

In this paper, we introduced the barycentric coordinate system embedding system, and showed its applicability to problems such as finding consistent polytopes, classification, function approximation and regression. We derived a statistical foundation for this approach, and presented experiments on datasets which show promising empirical results. Future work includes analytical and empirical investigations of other natural splitting strategies.

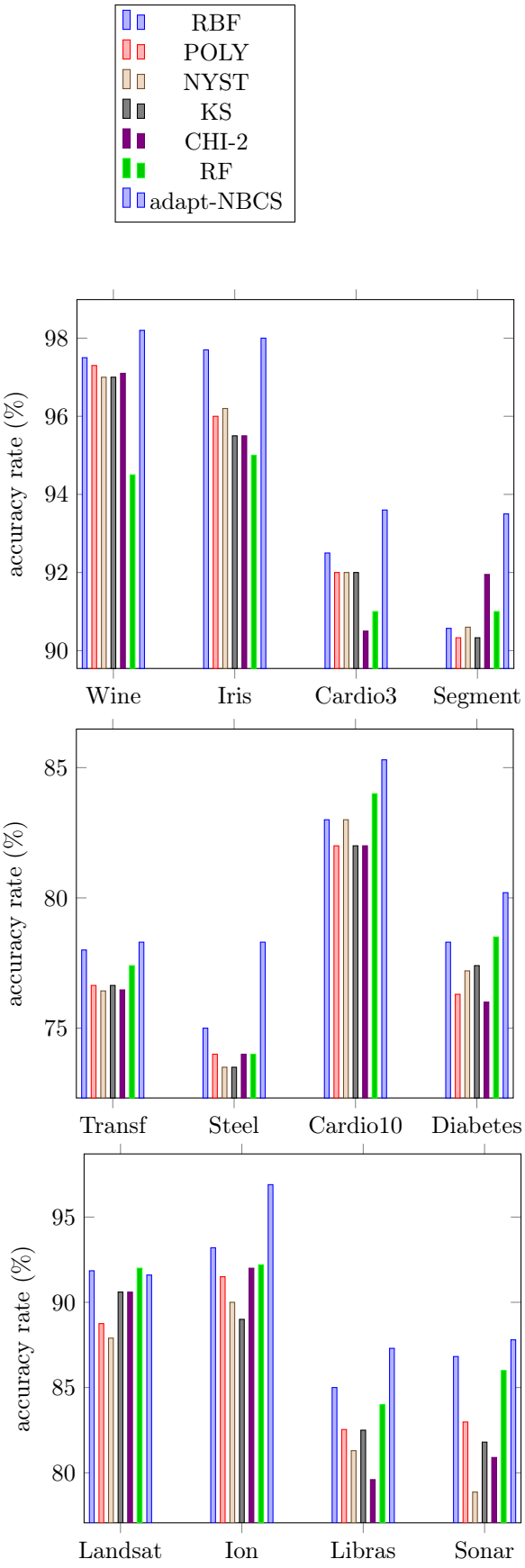


Figure 5: classification results for different embedding techniques

Table 1: Datasets

	Iris	Wine	Ion	Libras	Sonar	Transf
#dims	4	13	34	90	60	4
#examples	150	178	351	360	208	748
#classes	3	3	2	15	2	2
	Steel	Cardio3	Cardio10	Segment	Landsat	Diabetes
#dims	27	21	21	19	36	13
#examples	1941	2126	2126	2310	6435	768
#classes	7	3	10	7	6	2

Table 2: Classification results for different embedding techniques.

Dataset	n	d	3rd degree SVM	CoreSVM-RBF	**uni-NBCS**	**adapt-NBCS**
SkinNonSkin	245,057	4	99.4%, 20 sec	98.9%, 570.4 sec	**97.6%, 4 sec**	**98.8%, 4.2 sec**
cod-rna	59,535	8	95.2%, 18.6 sec	94.3%, 23 sec	**93.6%, 8.7 sec**	**94.5%, 9 sec**
shuttle	58,000	9	98%, 25.8 sec	93.2%, 5.3 sec	**95.4%, 8 sec**	**97.8%, 6.3 sec**
forest cover type	522,911	54	81.5%, 8028 sec	94.5%, 2028 sec	**92.3%, 2040 sec**	**96%, 2140 sec**

Table 3: Regression results for different embedding techniques.

Dataset	n	d	Poly	KS	Nystroem	χ^2	**NBCS**
Abalone	4177	8	0.746 ± 0.01	0.798 ± 0.01	0.815 ± 0.02	0.805 ± 0.02	**0.812 ± 0.01**
Bodyfat	252	14	0.973 ± 0.01	0.957 ± 0.02	0.923 ± 0.02	0.852 ± 0.02	**0.966 ± 0.01**
Cadata	20,640	8	0.949 ± 0.01	0.952 ± 0.01	0.943 ± 0.01	0.936 ± 0.01	**0.945 ± 0.01**
Cpusmall	8192	12	0.961 ± 0.01	0.968 ± 0.01	0.932 ± 0.01	0.973 ± 0.01	**0.971± 0.01**
Housing	506	13	0.672 ± 0.03	0.769 ± 0.01	0.854 ± 0.01	0.844 ± 0.01	**0.825± 0.01**
Prim	74	27	0.791 ± 0.01	0.815 ± 0.01	0.780 ± 0.01	0.823 ± 0.02	**0.870± 0.02**
Spacega	3107	6	0.682 ± 0.03	0.711 ± 0.03	0.781 ± 0.01	0.823 ± 0.01	**0.875 ± 0.02**
Triazines	186	60	0.729 ± 0.01	0.720 ± 0.01	0.711 ± 0.01	0.720 ± 0.01	**0.723± 0.01**
Eunite2001	367	16	0.825 ± 0.01	0.681 ± 0.03	0.725 ± 0.02	0.683 ± 0.03	**0.812 ± 0.01**

References

Aguilera and Morin, P. (2008). Approximating optimization problems over convex functions. *Numerische Mathematik*, 111(1):1–34.

Aguilera, N., Forzani, L., and Morin, P. (2011). On uniform consistent estimators for convex regression. *Journal of Nonparametric Statistics*, 23(4):897–908.

Allon, G., Beenstock, M., Hackman, S., Passy, U., and Shapiro, A. (2007). Nonparametric estimation of concave production technologies by entropic methods. *Journal of Applied Econometrics*, 22(4):795–816.

Anthony, M. and Bartlett, P. L. (1999). *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, Cambridge.

Arriaga, R. I. and Vempala, S. (2006). An algorithmic theory of learning: Robust concepts and random projection. *Machine Learning*, 63(2):161–182.

Belkin, M., Hsu, D. J., and Mitra, P. (2018). Overfitting or perfect fitting? risk bounds for classification and regression rules that interpolate. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 31*, pages 2300–2311. Curran Associates, Inc.

Beltrami, M. and da Silva, A. C. L. (2015). Grid-quadtree algorithm for support vector classification parameters selection. *Appl. Math. Sci*, 9:75–82.

Bernal, A., Crammer, K., and Pereira, F. (2012). Automated gene-model curation using global discriminative learning. *Bioinformatics*.

Birke, M. and Dette, H. (2007). Estimating a convex function in nonparametric regression. *Scandinavian Journal of Statistics*, 34(2):384–404.

Chang, Y., Hsieh, C., Chang, K., Ringgaard, M., and Lin, C. (2010). Training and testing low-degree polynomial data mappings via linear SVM. *JMLR*.

Chum, O. (2015). Low dimensional explicit feature maps. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 4077–4085.

Davies, S. (1996). Multidimensional triangulation and interpolation for reinforcement learning. In *Advances in Neural Information Processing Systems 9, NIPS, Denver, CO, USA, December 2-5, 1996*, pages 1005–1011.

Delgado, M. F., Cernadas, E., Barro, S., and Amorim, D. G. (2014). Do we need hundreds of classifiers to solve real world classification problems? *J. Mach. Learn. Res.*, 15(1):3133–3181.

Eisenstat, D. (2014). Algorithm intersecting polytope and half line. Stack Overflow. https://stackoverflow.com/questions/25399417/algorithm-intersecting-polytope-and-half-line.

Fine, S. and Scheinberg, K. (2001). Efficient SVM training using low-rank kernel representations. *Journal of Machine Learning Research*, 2:243–264.

Gärtner, B. (1995). A subexponential algorithm for abstract optimization problems. *SIAM Journal on Computing*, 24(5):1018–1035.

Goel, S. and Klivans, A. (2018). Learning neural networks with two nonlinear layers in polynomial time (arxiv:1709.06010v4).

Gottlieb, L., Kaufman, E., Kontorovich, A., and Nivasch, G. (2018). Learning convex polytopes with margin. In *NeurIPS*, pages 5711–5721.

Goyal, N. and Rademacher, L. (2009). Learning convex bodies is hard, arxiv:0904.1227.

Gu, Q. and Han, J. (2013). Clustered support vector machines. In *Artificial Intelligence and Statistics*, pages 307–315.

Hannah, L. and Dunson, D. B. (2011). Approximate dynamic programming for storage problems. In Getoor, L. and Scheffer, T., editors, *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pages 337–344. Omnipress.

Hannah, L. A. and Dunson, D. B. (2013). Multivariate convex regression with adaptive partitioning. *Journal of Machine Learning Research*, 14:3261–3294.

Hanneke, S. and Kontorovich, A. (2019). A sharp lower bound for agnostic learning with sample compression schemes. In *ALT*.

Hao Zhang, Berg, A. C., Maire, M., and Malik, J. (2006). Svm-knn: Discriminative nearest neighbor classification for visual category recognition. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 2126–2136.

Hildreth, C. (1954). Point estimates of ordinates of concave functions. *Journal of the American Statistical Association*, 49:598.

Joachims, T. (2006). Training linear SVMs in linear time. In *KDD*.

Khot, S. and Saket, R. (2011). On the hardness of learning intersections of two halfspaces. *J. Comput. Syst. Sci.*, 77(1):129–141.

Klivans, A. R. and Servedio, R. A. (2008). Learning intersections of halfspaces with a margin. *J. Comput. Syst. Sci.*, 74(1):35–48.

Klivans, A. R. and Sherstov, A. A. (2009). Cryptographic hardness for learning intersections of halfspaces. *J. Comput. Syst. Sci.*, 75(1):2–12.

Koushanfar, F., Majzoobi, M., and Potkonjak, M. (2010). Nonparametric combinatorial regression for shape constrained modeling. *IEEE Transactions on Signal Processing*, 58(2):626–637.

Kuosmanen, T. (2008). Representation theorem for convex nonparametric least squares. *The Econometrics Journal*, 11(2):308–325.

Li, F., Ionescu, C., and Sminchisescu, C. (2010). *Random Fourier Approximations for Skewed Multiplicative Histogram Kernels*, pages 262–271. Springer Berlin Heidelberg, Berlin, Heidelberg.

Lim, E. and Glynn, P. W. (2012). Consistency of multidimensional convex regression. *Operations Research*, 60(1):196–208.

Magnani, A. and Boyd, S. P. (2009). Convex piecewise-linear fitting. *Optimization and Engineering*, 10(1):1–17.

Maji, S., Berg, A., and J., M. (2012). Efficient classification for additive kernel SVMs. *PAMI*.

Megiddo, N. (1988). On the complexity of polyhedral separability. *Discrete & Computational Geometry*, 3(4):325–337.

Mohri, M., Rostamizadeh, A., and Talwalkar, A. (2012). *Foundations Of Machine Learning*. The MIT Press.

Monteiro, R. D. C. and Adler, I. (1989). Interior path following primal-dual algorithms. part ii: Convex quadratic programming. *Mathematical Programming*, 44(1):43–66.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Pele, O., Taskar, B., Globerson, A., and Werman, M. (2013). The pairwise piecewise-linear embedding for efficient non-linear classification. In *ICML*.

Perronnin, F., Senchez, J., et al. (2010). Large-scale image categorization with explicit data embedding. In *CVPR*.

Porikli, F. and Ozkan, H. (2011). Data driven frequency mapping for computationally scalable object detection. In *2011 8th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 30–35.

Rahimi, A. and Recht, B. (2007). Random features for large-scale kernel machines. *NIPS*.

Rahimi, A. and Recht, B. (2009). Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. In *NIPS*, pages 1313–1320. Curran Associates, Inc.

Saavedra, E., Grauel, A., and Morton, D. (2004). Support vector machines and quad-trees applied to image compression. In *Proceedings of the 6th Nordic Signal Processing Symposium-NORSIG*, volume 2004. Citeseer.

Seijo, E. and Sen, B. (2011). Nonparametric least squares estimation of a multivariate convex regression function. *Ann. Statist.*, 39(3):1633–1657.

Shahrampour, S. and Tarokh, V. (2018). Learning bounds for greedy approximation with explicit feature maps from multiple kernels. NIPS'18, pages 4695–4706.

Shawe-Taylor, J., Bartlett, P. L., Williamson, R. C., and Anthony, M. (1998). Structural risk minimization over data-dependent hierarchies. *IEEE Transactions on Information Theory*, 44(5):1926–1940.

Smola, A. J. and Schökopf, B. (2000). Sparse greedy matrix approximation for machine learning. ICML '00, pages 911–918.

Tsang, I. W., Kocsor, A., and Kwok, J. T. (2007). Simpler core vector machines with enclosing balls. In *Proceedings of the 24th international conference on Machine learning*, pages 911–918. ACM.

Tsang, I. W., Kwok, J. T., and Cheung, P.-M. (2005). Core vector machines: Fast svm training on very large data sets. *Journal of Machine Learning Research*, 6(Apr):363–392.

Vedaldi, A. and Zisserman, A. (2012). Efficient additive kernels via explicit feature maps. *PAMI*.

Vempati, S., Vedaldi, A., Zisserman, A., and Jawahar, C. V. (2010). Generalized RBF feature maps for efficient detection. In *BMVC*, pages 1–11. British Machine Vision Association.

Wang, Y. and Ni, H. (2012). Multivariate convex support vector regression with semidefinite programming. *Knowl.-Based Syst.*, 30:87–94.

Williams, C. and Seeger, M. (2000). The effect of the input density distribution on kernel-based classifiers. In *ICML*, pages 1159–1166. Morgan Kaufmann.

Williams, C. K. I. and Seeger, M. (2001). Using the nyström method to speed up kernel machines. In *NIPS '01*, pages 682–688.

Zafeiriou, S. and Kotsia, I. (2013). On one-shot similarity kernels: Explicit feature maps and properties. In *ICCV '13*, pages 2392–2399.