A Theory for Memory-Based Learning

Jyh-Han Lin and Jeffrey Scott Vitter

Department of Computer Science
Brown University
Providence, Rhode Island 02912

CS-92-53
November 1992

A Theory for Memory-Based Learning

Jyh-Han Lin [1]
Motorola Inc.
Communications Systems Lab.
Applied Research
Paging and Wireless Data Group
1500 N.W. 22nd Avenue
Boynton Beach, FL 33426–8292
jyh-han_lin@pts.mot.com

Jeffrey Scott Vitter
Department of Computer Science
Brown University
Providence, R. I. 02912–1910
jsv@cs.brown.edu

[1]The research was conducted while the author was at the Department of Computer Science, Brown University.

Abstract

A memory-based learning system is an extended memory management system that decomposes the input space either statically or dynamically into subregions for the purpose of storing and retrieving functional information. The main generalization techniques employed by memory-based learning systems are the nearest-neighbor search, space decomposition techniques, and clustering. Research on memory-based learning is still in its early stage. In particular, there are very few rigorous theoretical results regarding memory requirement, sample size, expected performance, and computational complexity. In this paper, we propose a model for memory-based learning and use it to analyze several methods— ϵ-covering, hashing, clustering, tree-structured clustering, and receptive-fields—for learning smooth functions. The sample size and system complexity are derived for each method. Our model is built upon the generalized PAC learning model of Haussler [16] and is closely related to the method of vector quantization in data compression. Our main result is that we can build memory-based learning systems using new clustering algorithms (Lin and Vitter [24]) to PAC-learn in polynomial time using only polynomial storage in typical situations.

Keywords: Memory-based learning, PAC learning, clustering, approximation, linear programming, relaxation, covering, hashing.

1 MOTIVATION

In this paper, we introduce a model for *memory-based learning* and consider the problem of learning smooth functions by memory-based learning systems. A *memory-based learning system* is an extended memory management system that decomposes the input space either statically or dynamically into subregions for the purpose of storing and retrieving functional information for some smooth function. The main generalization techniques employed by memory-based learning system are the nearest-neighbor search, space decomposition techniques, and clustering. Although memory-based learning systems are not as powerful as neural net models in general, the training problem for memory-based learning systems may be computationally more tractable. An example memory-based learning system is shown in Figure 1. The "encoder" γ maps an input from the input space X into a set of addresses and the "decoder" β maps the set of activated memory locations into an output in the output space Y. The look-up table for memory-based learning systems can be organized as hash tables, trees, or full-search tables. The formal definitions of memory-based learning systems will be given in Section 2.

The motivation for our model is as follows: In the human motor system, most of the computations done are entirely subconscious. The detailed computations of what each muscle must do in order to coordinate with other muscles so as to produce the desired movement are left to low-level, subconscious computing centers. Considering the complexity of the type of manipulation tasks routinely performed by biological organisms, it seems that the approach of controlling robotic manipulator systems by a mathematical formalism such as trigonometric equations is inadequate to produce truly sophisticated motor behavior. To remedy this situation, Albus [1,2, 3] proposed a memory-driven, table-reference motor control system called *Cerebellar Model Articulation Controller* (*CMAC*). The fact that for n input variables with R distinguishable levels there are R^n possible inputs may be sufficient to discourage this line of research. However, Albus observed that for any physical manipulator system, the number of different inputs that are likely to be encountered (and thus the size of memory that is actually needed) is much smaller than R^n. He also noticed for similar motor behaviors (for example, swinging a bat or a golf club) that the required muscle movements are similar. Albus outlined a memory management technique to take

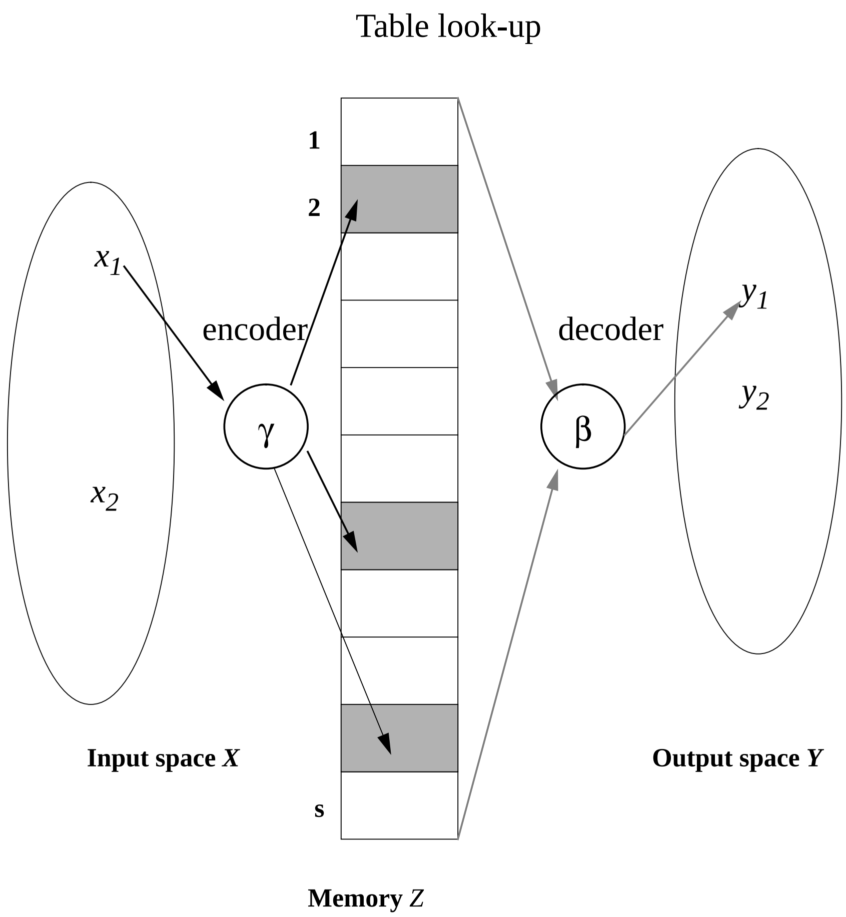

Figure 1: An example memory-based learning system. The encoder γ maps an input from the input space X into a set of addresses and the decoder β maps the set of activated memory locations into an output in the output space Y.

advantage of these two properties and make the memory-based approach to learning control functions more practical.

In the CMAC system, each input x from an input space X is assigned by a mapping γ to a set $\gamma(x)$ of locations in a memory V. Each location contains a vector in an output space Y. The output $f(x)$ is computed by summing the values (weights) at all of the memory locations assigned to x:

$$f(x) = \sum_{i \in \gamma(x)} V[i].$$

The mapping has the characteristic that similar inputs in the input space X map to overlapping sets of locations in the memory V, while dissimilar inputs map to distinct sets of locations in the memory V. The amount of overlap between two sets of locations in the memory V is related to the generalized Hamming distance between two corresponding inputs in X. This mapping is supposed to give automatic generalization (interpolation) between inputs in X: that is, similar inputs produce similar outputs.

Clearly, this scheme may require the size of memory V to be on the same order of magnitude as the total number of possible input vectors in X. In practice, this is hardly feasible. For this reason, the memory V is considered to be only a hypothetical memory; each location in V is mapped using a hash function h to a physical memory $\mathcal{Z}$ of practical size. The output $f(x)$ is then computed by summing the values in the memory $\mathcal{Z}$ that are mapped to by the input x:

$$
\begin{aligned}
f(x) &= \sum_{i \in \gamma(x)} \mathcal{Z}[h(i)] \\
&= \sum_{i \in \gamma'(x)} \mathcal{Z}[i],
\end{aligned}
$$

where $\gamma' = h \circ \gamma$. As a result of the random hashing from the hypothetical memory V to the physical memory $\mathcal{Z}$, the sets of memory locations mapped to by dissimilar inputs in input space X have a low, but nonzero, probability of overlapping; this can create an undesirable generalization between dissimilar inputs.

The resulting system will produce an output $f(x) \in Y$ for any input x in the input space X. Since the number of locations in the real memory $\mathcal{Z}$ will typically be much smaller than the total number of possible inputs, it is unlikely that the weights in $\mathcal{Z}$ can be found such that the outputs of CMAC system are correct over the entire input space. On the other hand, it is unlikely that all possible inputs will be encountered in solving a particular control or classification problem.

The standard CMAC model has been applied to the real-time control of robots with encouraging success (Miller [27]; Miller, Glanz, and Kraft [28]). A comprehensive coverage of the CMAC models and learning algorithms is given by Dean [7].

Research on the CMAC model and its variants is still in its early stage. In particular, there are very few rigorous theoretical results available. Many problems remained unanswered, among them the following:

1. In the current experimental study, learning parameters are chosen on an *ad hoc* basis. The effects of the scale of resolution, the size of physical memory, and

the size of the training database (examples) on system performance are largely unknown.

2. Given a class $\mathcal{F}$ of functions and a tolerable relative error bound, what are the sample size and memory size required to approximate functions in $\mathcal{F}$?

3. Given a sample, what are the computational complexities of training? That is, how much time does it require to determine system parameters from the sample?

In Section 2 we outline a theoretical framework for answering these problems. Our memory-based learning model is built upon the generalized PAC learning model of Haussler [16] and is closely related to the method of vector quantization in data compression (Gersho [13]; Gray [15]; Riskin [38]; Gersho and Gray [14]). Section 3 introduces the notion of *quantization number*, which is intended to capture the optimal memory requirement of memory-based learning systems for a given error bound. The quantization number can be significantly smaller than the *covering number* in practice. In Section 4 we use our model to analyze several methods for learning smooth functions by nearest-neighbor systems. Our main result is that we can build memory-based learning systems using the new clustering algorithms of Lin and Vitter [24] to PAC-learn in polynomial time using only polynomial storage in typical situations. We extend our analysis to *tree-structured* and *higher-order* memory-based learning system in Section 5 and 6, respectively. We conclude with some possible extensions to our model in Section 7.

2 A MEMORY-BASED LEARNING MODEL

Let T be a complete and separable metric space with distance metric d_T. We denote the metric space by (T, d_T). Let $\mathcal{H}(T)$ denote the space whose points are the compact subset of T. The *diameter* of a set $A \in \mathcal{H}(T)$, denoted as $diam(A)$, is $\sup_{t_1,t_2 \in T} d_T(t_1, t_2)$. The distance $d_T(t, A)$ from a point t to a set $A \in \mathcal{H}(T)$, is defined as $\inf_{x \in A} d_T(t, x)$. For any $\epsilon > 0$, an ϵ-cover for A is a finite set $\mathcal{U} \subseteq T$ such that for all $t \in A$ there is a $u \in \mathcal{U}$ such that $d_T(t, u) \leq \epsilon$. If A has a finite ϵ-cover for every $\epsilon > 0$ then A is *totally bounded.* Let $\mathcal{N}(A, \epsilon, d_T)$ denote the size of the smallest ϵ-cover for A. We refer to $\mathcal{N}(A, \epsilon, d_T)$ as the *covering number*.

In this paper, we let $X \subset \Re^k$ be the input space and $Y \subset \Re^\ell$ be the output space and let d_X and d_Y be the Euclidean metrics. In typical applications, X and Y are usually hypercubes or hyperrectangles. Let $M_X = diam(X)$ and $M_Y = diam(Y)$. For a positive integer s, let $\mathbf{N}_s$ denote the set $\{1, \ldots, s\}$. Let $\mathbf{N}_s^r$ be the collection of all r-element subsets (r-subsets) of $\mathbf{N}_s$. Let $\mathcal{U} = \{u_1, \ldots, u_s\}$ and B be a subset of $\mathcal{U}$, then $index(B)$ denotes the set of indices of elements in B.

2.1 MEMORY-BASED LEARNING SYSTEMS

Definition 1 A generic *memory based learning system* $\mathcal{G}$ realizes a class of functions from the input space (X, d_X) to the output space (Y, d_Y). Each function g realizable by $\mathcal{G}$ can be specified by a sequence of memory contents $\mathcal{Z} = \langle z_1, \ldots, z_s \rangle$, where s is a positive integer, and a pair of functions $\langle \gamma, \beta \rangle$; γ is the encoder, which is a mapping from X to $2^{\mathbf{N}_s}$ and β is the decoder, which is a mapping from $2^{\mathbf{N}_s} \times \Re$ to Y. We can write g as the composition $\beta \circ \gamma$. We denote $\mathcal{Z}(i) = z_i$.

We may regard $\mathbf{N}_s$ as the address (or neuronal) space and $2^{\mathbf{N}_s}$ as the collection of sets of activated addresses (or neurons).

We will often study *parameterized* classes of memory-based learning systems. Let $\mathcal{C} : \mathcal{G} \to \Re^+$ be a complexity function of memory-based learning systems, which maps a system $g \in \mathcal{G}$ to a positive real number. The most straightforward complexity measure is the size of memory, which we will use in this paper. However, for some applications, other complexity measures may be more appropriate. For example, in real-time applications, we may be more concerned with the speed of encoding and decoding. In remote-control applications, the sensor/encoder and effector/decoder may not be at the same location, and the sensor has to send control signals (addresses) to the effector via communication channels. In such a scenario, communication complexity may be a more important issue. We let G_s denote the class of memory-based learning systems of complexity at most s, that is $G_s = \{g \mid \mathcal{C}(g) \leq s\}$.

We are interested in the following two types of memory-based learning systems: *full-search systems* and *tree-structured systems*. In a full-search system, each memory location corresponds to a region in the input space and contains a representative vector (key) and a functional value; the encoder maps an input to the memory locations corresponding to regions that include the input point. Examples of full-search systems include Voronoi systems and receptive-field systems.

Definition 2 The class $\mathcal{G} = \cup_{s \geq r} G_s^r$ of *(generalized) Voronoi systems of order* r is defined as follows: Let $\mathcal{U} = \{u_1, \ldots, u_s\}$y and B be an r-subset of $\mathcal{U}$, then $Vor(B; r)$ denotes the Voronoi region of order r for B. The encoder γ of a Voronoi system of order r and size s is a mapping from X to $\mathbf{N}_s^r$ and maps $x \in X$ to $index(B)$ if and only if $x \in Vor(B; r)$. The decoder β is a mapping from $X \times \mathbf{N}_s^r$ and is defined as

$$g(x) = \frac{1}{r} \sum_{i \in \gamma(x)} \mathcal{Z}(i).$$

We shall refer to the first-order Voronoi systems simply as *Voronoi systems*.

Definition 3 The class $\mathcal{G} = \cup_{s \geq 1} G_s$ of *receptive-field systems* is defined as follows: Let $\mathcal{R} = \{R_1, \ldots, R_s\}$ be a collection of polyhedral sets (regions) such that $\bigcup_{\mathcal{R}} R_i = X$. The encoder γ maps an input x to the set $\gamma(x)$ of indices of regions that contain x. Note that the regions are allowed to be overlapped. The maximum degree of overlap is the *order* of the system. The decoder β is a mapping from $X \times \mathbf{N}_s^r$ and is defined as

$$g(x) = \sum_{i \in \gamma(x)} \mathcal{Z}(i).$$

Notable examples of receptive-field systems include the CMAC model and Moody's multi-resolution hierarchies (Moody [29]).

In a tree-structured system, the encoder partitions the input space into a hierarchy of regions. An input is mapped to the memory location corresponding to the region represented by a leaf. The computational advantage of tree-structured systems over full-search systems in sequential models of computation is that the mapping from an input to a memory location can be done quickly by tree traversal.

Definition 4 The class $\mathcal{G} = \cup_{s \geq 1} G_s$ of *tree-structured systems* is defined as follows: The encoder γ of a tree-structured systems of size s partitions the input space into a hierarchy of regions specified by a tree with s nodes. Each internal node has a number of branches, each of which is associated with a key. Given an input, starting at the root node, the encoder compares the input with each key and follows the branch associated with the key nearest to the input; the search proceeds this way until a leaf is reached. The search path is output by the encoder as the address for that input. The decoder β takes a search path and outputs the value in the leaf.

Examples of tree-structured systems include learning systems based upon quadtrees and k-d trees such as SAB-trees of Moore [31].

2.2 THE MEMORY-BASED LEARNING PROBLEM

Informally, given a probability measure P over $X \times Y$, the goal of learning in this model is to approximate P by a memory-based learning system $g \in \mathcal{G}$ of reasonable complexity. The expected error of the hypothesis g with respect to P is denoted by

$$\mathbf{er}_P(g) = \mathbf{E}\left[d_Y(g(\mathbf{x}), \mathbf{y})\right] = \int_{X \times Y} d_Y(g(x), y)\ dP(x, y),$$

where $\langle \mathbf{x}, \mathbf{y} \rangle$ is the random vector corresponding to P. The formal PAC memory-based learning model is defined below:

Definition 5 A *memory-based learning problem* $\mathcal{B}$ is specified by a class $\mathcal{G}$ of memory-based learning systems and a class $\mathcal{P}$ of probability measures over $X \times Y$, where $X \subseteq \Re^k$ and $Y \subseteq \Re^\ell$. We say that $\mathcal{B}$ is *learnable* if for any $0 < \delta < 1/2$ and $0 < \epsilon < 1/2$ the following holds: There exists a (possibly randomized) algorithm L such that if L is given as input a random sample sequence $\overline{\xi} = \langle (x_i, y_i) \rangle$ of polynomial size $m(\frac{1}{\delta}, \frac{1}{\epsilon}, k, \ell)$, then with probability at least $1 - \delta$, L will output a memory-based learning system $L(\overline{\xi}) \in \mathcal{G}$ that satisfies

$$\mathbf{er}_P(L(\overline{\xi})) \leq \epsilon M_Y.$$

If L runs in polynomial time, then we say that $\mathcal{B}$ is *polynomial-time learnable.*

2.3 SMOOTH FUNCTIONS

Without any restriction on the class $\mathcal{P}$ of probability measures over $X \times Y$, learning is not likely to be feasible in terms of memory requirement, sample size, and computational complexity. In this paper, we restrict $\mathcal{P}$ to be generated by some smooth function f and some probability measure P_X over X, that is, the sample point is of the form $(x, f(x))$. Further justification for the smoothness assumption is given by Poggio and Girosi [33,34].

Definition 6 A function f from X into Y is called a *Lipschitz function* if and only if for some $K < \infty$ we have

$$d_Y(f(x), f(x')) \leq K d_X(x, x'),$$

for all $x, x' \in X$. Let $\|f\|_L$ denote the smallest such K. A class of functions $\mathcal{F}$ from X into Y is called *Lipschitz functions* if and only if for some $K < \infty$ we have

$$\sup_{f \in \mathcal{F}} \|f\|_L \leq K.$$

Let $\|\mathcal{F}\|_L$ denote the smallest such K. We call K the *Lipschitz bound.*

The Lipschitz bound does not have to hold everywhere; it suffices for our purpose if it holds with probability one. For example, the class of piece-wise Lipschitz functions satisfies this relaxed condition. Haussler [16] has relaxed the Lipschitz condition further:

Definition 7 For each $f \in \mathcal{F}$ and real $\epsilon > 0$, $\lambda(f, \epsilon, \cdot)$ is the real-valued function on X defined by

$$\lambda(f, \epsilon, x) = \sup\{d_Y(f(x), f(x'))\},$$

where the supremum is taken over all $x' \in X$ for which $d_X(x, x') \leq \epsilon$. Let P_X be a probability measure over X. We say that the $\mathcal{F}$ is *uniformly Lipschitz on the average* with respect to P_X if for all $\epsilon > 0$ and all $f \in \mathcal{F}$ there exists some $0 < K < \infty$ such that

$$\mathbf{E}\left[\lambda(f, \epsilon/K, \mathbf{x})\right] \leq \epsilon.$$

Let $\|\mathcal{F}\|_L^{P_X}$ be the smallest such K. For a class $\mathcal{P}_X$ of probability measures over X, we define $\|\mathcal{F}\|_L^{\mathcal{P}_X} = \sup_{P_X \in \mathcal{P}_X} \|\mathcal{F}\|_L^{P_X}$.

3 VORONOI ENCODERS AND QUANTIZATION NUMBERS

The class $\mathcal{G} = \cup_{s \geq 1} G_s$ of *Voronoi systems* (*nearest-neighbor systems*) is defined as follows: We can specify each $g \in G_s$ by a set $\mathcal{U} = \{u_1, \ldots, u_s\}$ of size s. Let $Vor(u_j)$ denote the Voronoi region for the point u_j. The encoder γ of g is a mapping from X to $\mathbf{N}_s$ and maps $x \in X$ to j if and only if $x \in Vor(u_j)$. Let $\mathcal{Z} = \{z_1, \ldots, z_s\} \subset Y$. The decoder β of g is a mapping from $\mathbf{N}_s$ to Y defined by $\beta(j) = z_j$. In other words, the system maps an input x to its nearest neighbor in $\mathcal{U}$, and then outputs the value stored in the memory location corresponding to that point.

We call the encoders of Voronoi systems the *Voronoi encoders*. In the following, we introduce the notion of *quantization number*, which characterizes the optimal size of Voronoi encoders for a given error bound. The quantization number can be substantially smaller than the covering number.

Definition 8 Let P_X be a probability measure over X and let $\mathbf{x}$ be the random vector corresponding to P_X. For any $\epsilon > 0$, the *quantization number* $\mathcal{Q}_{P_X}(X, \epsilon, d_X)$ of P_X is defined as the smallest integer s such that there exists a Voronoi encoder γ of size s that satisfies

$$\mathbf{E}\left[d_X(\mathbf{x}, u_{\gamma(\mathbf{x})})\right] \le \epsilon.$$

For a class $\mathcal{P}_X$ of probability measures over X, we define $\mathcal{Q}_{\mathcal{P}_X}(X, \epsilon, d_X) = \sup_{P_X \in \mathcal{P}_X} \mathcal{Q}_{P_X}(X, \epsilon, d_X)$.

3.1 THE PSEUDO-DIMENSION OF VORONOI ENCODERS

Building on the work of Vapnik and Chervonenkis [41,42], Pollard [35,36], Dudley [10], and Devroye [8], Haussler [16] introduced the notion of *pseudo-dimension*, which is a generalization of VC dimension. He first defined the notion of *fullness of sets*:

Definition 9 For $x \in \Re$, let $sign(x) = 1$ if $x > 0$; else $sign(x) = 0$. For $\overline{x} = (x_1, \ldots, x_k) \in \Re^m$, let $sign(\overline{x}) = (sign(x_1), \ldots, sign(x_m))$ and for $A \subset \Re^m$ let $sign(A) = \{sign(\overline{y}) \mid \overline{y} \in A\}$. For any $A \subset \Re^m$ and $\overline{x} \in \Re^m$, let $A + \overline{x} = \{\overline{y} + \overline{x} \mid \overline{y} \in A\}$, that is, the translation of A obtained by adding the vector $\overline{x}$. We say that A is *full* if there exists $\overline{x} \in \Re^m$ such that $sign(A + \overline{x}) = \{0, 1\}^m$, that is, if there exists some translation of A that intersect all 2^m orthants of $\Re^m$.

For example, hyperplanes in $\Re^m$ are not full, since no hyperplanes in $\Re^m$ can intersect all orthants of $\Re^m$. The pseudo-dimension is defined as follows:

Definition 10 Let $\mathcal{F}$ be a class of functions from a set X into $\Re$. For any sequence $\overline{\xi}_X = (x_1, \ldots, x_m)$ of points in X, let $\mathcal{F}(\overline{\xi}_X) = \{(f(x_1), \ldots, f(x_m)) : f \in \mathcal{F}\}$. If $\mathcal{F}(\overline{\xi}_X)$ is full then we say that $\overline{\xi}_X$ is shattered by $\mathcal{F}$. The pseudo-dimension of $\mathcal{F}$, denoted by $\mathbf{dim_P}(\mathcal{F})$, is the largest m such that there exists a sequence of m points in X that is shattered by $\mathcal{F}$. If arbitrarily long sequences are shattered, then $\mathbf{dim_P}(\mathcal{F})$ is infinite.

It is clear when $\mathcal{F}$ is a class of $\{0,1\}$-valued functions that the definition of the pseudo-dimension is the same as that of the VC dimension. Dudley and Haussler have shown the following useful property of pseudo-dimension:

Theorem 1 (Dudley [9]) *Let $\mathcal{F}$ be a k-dimensional vector space of functions from a set X to $\Re$. Then $\mathbf{dim_P}(\mathcal{F}) = k$.*

Theorem 2 (Haussler [16]) *Let $\mathcal{F}$ be a class of function from a set X into $\Re$. Fix any nondecreasing (or nonincreasing) function $h : \Re \to \Re$ and let $\mathcal{H} = \{h \circ f : f \in \mathcal{F}\}$. Then we have $\mathbf{dim_P}(\mathcal{H}) \leq \mathbf{dim_P}(\mathcal{F})$.*

To derive the pseudo-dimension of Voronoi encoders, we use the following lemma attributed to Sauer [39]:

Lemma 1 [Sauer's Lemma] *Let $\mathcal{F}$ be a class of functions from $S = \{1, 2, \ldots, m\}$ into $\{0,1\}$ with $|\mathcal{F}| > 1$ and let d be the length of the longest sequence of points $\overline{\xi}_S$ from S such that $\mathcal{F}(\overline{\xi}_S) = \{0,1\}^d$. Then we have*

$$|\mathcal{F}| \leq (em/d)^d,$$

where e is the base of the natural logarithm.

We now are ready to bound the pseudo-dimension of Voronoi encoders:

Lemma 2 *Let G_s be the Voronoi system of size at most s and let d_X be the Euclidean metric. For each possible encoder γ of G_s, we define $f_\gamma(x) = d_X(x, u_{\gamma(x)})$ and let $\Gamma_s : X \to [0, M_X]$ be the class of all such functions $f_\gamma(x)$. Then we have*

$$\mathbf{dim_P}(\Gamma_s) \leq 2(k+1)s\log(3s) = O(ks\log s),$$

where k is the dimension of the input space.

Proof: By the definition of the Euclidean metric, we can write $(f_\gamma(x))^2$ as a polynomial in k variables with $2k+1$ coefficients, where k is the dimension of the input space. By Theorems 1 and 2, we have $\mathbf{dim_P}(\Gamma_1) \leq 2k+1$. Let $\overline{\xi}_X$ be a sequence of m points in X and let $\overline{r}$ be an arbitrary m-vector. Since $f_\gamma(x) = \min_{u \in \mathcal{U}} d_X(x, u)$, we have

$$\begin{aligned} |sign(\Gamma_s(\overline{\xi}_X) + \overline{r})| &\leq |sign(\Gamma_1(\overline{\xi}_X) + \overline{r})|^s \\ &\leq \left(\frac{em}{2k+1}\right)^{(2k+1)s}. \end{aligned}$$

The last inequality follows from Sauer's Lemma. If $m = 2(k+1)s\log(3s)$, then $(em/(2k+1))^{(2k+1)s} < 2^m$. Therefore, we have $\mathbf{dim_P}(\Gamma_s) \leq 2(k+1)s\log(3s) = O(ks\log s)$. □

3.2 SAMPLE SIZE FOR THE UNIFORM CONVERGENCE OF VORONOI ENCODERS

In this section, we bound the sample size for estimating the error of Voronoi encoders. In the following, let $\hat{\mathbf{E}}_{\overline{\xi}_X}(f) = \frac{1}{m}\sum_{i=1}^{m} f(x_i)$ be the empirical mean of the function f, and let $d_\nu(r,t) = |r-t|/(\nu+r+t)$. We need the following corollary from Haussler and Long [18]:

Corollary 1 *Let $\mathcal{F}$ be a family of functions from a set X into $[0, M_X]$, where $\mathbf{dim_P}(\mathcal{F}) = d$ for some $1 \leq d < \infty$. Let P_X be a probability measure on X. Assume $\nu > 0$ and $0 < \alpha < 1$. Let $\overline{\xi}_X$ be generated by m independent draws from X according to P_X. If the sample size is*

$$m \geq \frac{9M_X}{\alpha^2\nu}\left(2d\ln\frac{24M_X}{(\alpha\sqrt{\alpha})\nu} + \ln\frac{4}{\delta}\right),$$

then we have

$$\mathbf{Pr}\{\exists f \in \mathcal{F} \mid d_\nu(\hat{\mathbf{E}}_{\overline{\xi}_X}(f), \mathbf{E}(f)) > \alpha\} \leq \delta.$$

Lemma 2 and Corollary 1 imply the following theorem:

Theorem 3 *Let Γ_s be defined as in Lemma 2. Assume $\nu > 0$ and $0 < \alpha < 1$. Let P_X be a probability measure on X and $\overline{\xi}_X$ be generated by m independent draws from X according to P_X. If the sample size is*

$$m \geq \frac{9M_X}{\alpha^2\nu}\left(2(2k+1)s\log(3s)\ln\frac{24M_X}{(\alpha\sqrt{\alpha})\nu} + \ln\frac{4}{\delta}\right),$$

then we have

$$\mathbf{Pr}\{\exists f \in \Gamma_s \mid d_\nu(\hat{\mathbf{E}}_{\overline{\xi}_X}(f), \mathbf{E}(f)) > \alpha\} \leq \delta.$$

4 MEMORY-EFFICIENT LEARNING OF SMOOTH FUNCTIONS

In this section, we investigate in detail three methods of learning smooth functions by Voronoi systems: ϵ-covering, hashing, and clustering. Our results are summarized in Table 1.

First, we introduce some notation: Let $\overline{\xi} = \langle (x_1, y_1), \ldots, (x_m, y_m) \rangle$ be a random sample sequence of length m. We denote the sequence $\langle x_1, \ldots, x_m \rangle$ by $\overline{\xi}_X$. We denote the random vector corresponding to a probability measure $P \in \mathcal{P}$ by $(\mathbf{x}, \mathbf{y})$. We denote the average empirical distance from the x-components of the examples to $\mathcal{U}$ by

$$\hat{d}_{\overline{\xi}_X}(\mathcal{U}) = \frac{1}{m} \sum_{i=1}^{m} d_X(x_i, \mathcal{U}).$$

The discrete version of the above problem is to restrict $\mathcal{U}$ to be a subset of $\overline{\xi}_X$.

The learning problem is specified as follows: We are given a class $\mathcal{G}$ of Voronoi systems and a class $\mathcal{P}$ of probability measures generated by a class $\mathcal{P}_X$ of probability measures over X and a class $\mathcal{F}$ of smooth functions from X to Y with $\|\mathcal{F}\|_L^{\mathcal{P}_X} = K$. Each sample point is of the form $(x, f(x))$ for some $f \in \mathcal{F}$. Give $0 < \delta, \epsilon < 1$ and sample sequence $\overline{\xi} = \langle (x_1, y_1), \ldots, (x_m, y_m) \rangle$, the goal of learning is to construct a Voronoi system $g \in \mathcal{G}$ such that the size of g is as small as possible and the expected error rate satisfies

$$\mathbf{er}_P(g) \le \epsilon M_Y,$$

with probability at least $1 - \delta$.

4.1 LEARNING BY ϵ-COVERING

The main idea of ϵ-covering is to cover the input space with small cells of radius ϵ and assign each cell a constant value. The smoothness condition assures a small expected error for the resulting system. The algorithm essentially learns by brute force:

Algorithm LE (learning by ϵ-covering):

1. Let $\mathcal{U}$ be an $\frac{\epsilon M_Y}{4K}$-cover of size N, where $N = \mathcal{N}(X, \frac{\epsilon M_Y}{4K}, d_X)$. Let $m = \frac{2N}{\epsilon} \ln \frac{N}{\delta}$ be the sample size.

2. For each $u_i \in \mathcal{U}$, if $Vor(u_i) \cap \overline{\xi}_X \neq \emptyset$ then we choose an arbitrary y_j such that $x_j \in Vor(u_i) \cap \overline{\xi}_X$ and set $\mathcal{Z}(i) = y_j$; otherwise, we set $\mathcal{Z}(i)$ arbitrarily.

Theorem 4 *With probability at least* $1 - \delta$*, the expected error for Algorithm LE satisfies* $\mathbf{er}_P(LE(\overline{\xi})) \le \epsilon M_Y$.

Algorithm	System size	Sample size
ϵ-covering (*LE*)	N	$O\left(\frac{N}{\epsilon}\log\frac{N}{\delta}\right)$
perfect hashing (*LH1*)	$O\left(\frac{1}{\delta}(pN)^2\right)$	$O\left(\frac{pN}{\epsilon}\log\frac{pN}{\delta}\right)$
universal hashing (*LH2*)	$O\left(\frac{1}{\epsilon}pN\right)$	$O\left(\frac{pN}{\epsilon}\log\frac{pN}{\delta}\right)$
coalesced hashing	$O(pN)$	$O\left(\frac{pN}{\epsilon}\log\frac{pN}{\delta}\right)$
optimal clustering (*LC1*)	s	$O\left(\frac{ks}{\epsilon}\log s\log\frac{1}{\epsilon}+\frac{1}{\epsilon}\log\frac{1}{\delta}\right)$
approx. clustering (*LC2*)	$O\left(s\left(\log\frac{ks}{\epsilon}+\log\log\frac{1}{\delta}\right)\right)$	$O\left(\frac{ks}{\epsilon}\log s\left(\log\frac{ks}{\epsilon}\right)^2+\frac{1}{\epsilon}\log\frac{1}{\delta}\right)$

Table 1: Upper bounds on system size and sample size for six algorithms for learning smooth functions by Voronoi systems. The goal of learning for each learning algorithm L is to achieve with probability at least $1-\delta$ an error bound of $\mathbf{er}_P(L(\overline{\xi})) \leq \epsilon M_Y$. In the table, k is the dimension of the input space, N is the covering number $\mathcal{N}(X, \frac{\epsilon M_Y}{4K}, d_X)$, $p \ll 1$ is the fraction of nonempty Voronoi cells, and s is the quantization number $\mathcal{Q}_{P_X}(X, \frac{\epsilon M_Y}{4K}, d_X)$.

Proof: For each Voronoi cell $Vor(u_i)$ satisfying $P_X(Vor(u_i)) \geq \frac{\epsilon}{2N}$, we have

$$\begin{aligned}\mathbf{Pr}(Vor(u_i)\cap\overline{\xi}_X=\emptyset) &\leq \left(1-\frac{\epsilon}{2N}\right)^{\frac{2N}{\epsilon}\ln\frac{N}{\delta}} \\ &\leq \frac{\delta}{N}.\end{aligned}$$

Therefore, with probability at least $1-\delta$, all Voronoi cells with probability over $\frac{\epsilon}{2N}$ will be hit by some sample point.

By the Lipschitz bound, for each Voronoi cell hit, we have

$$\mathbf{E}\left[d_Y(z_{\gamma(\mathbf{x})},\mathbf{y}) \mid Vor(u_i)\right] \leq \frac{\epsilon M_Y}{2P_X(Vor(u_i))}.$$

Furthermore, the total probability measure of Voronoi cells with less than $\frac{\epsilon}{2N}$ probability is at most $\epsilon/2$. Therefore, we have

$$\begin{aligned}\mathbf{er}_P(LE(\overline{\xi})) &\leq \left(1-\frac{\epsilon}{2}\right)\frac{\epsilon M_Y}{2}+\frac{\epsilon M_Y}{2}\\ &\leq \epsilon M_Y.\end{aligned}$$

□

4.2 LEARNING BY HASHING

Algorithm LE in the previous section covers the whole input space X with points. However, most of the Voronoi cells formed by points in the ϵ-cover $\mathcal{U}$ are likely to be empty. In this section we use hashing techniques to take advantage of this property. Below we outline three hashing-based algorithms: perfect hashing, universal hashing, and hashing with collision-resolution.

Let h be a hash function from $\mathbf{N}_N$ to $\mathbf{N}_{N'}$, where $N = |\mathcal{U}|$ and N' is a positive integer. For each address $1 \leq i \leq N'$ we define $h^{-1}(i)$ to be the subset of points in $\overline{\xi}_X$ that hash to memory location i, namely, $\{x_j \mid h(\gamma(x_j)) = i \text{ and } x_j \in \overline{\xi}_X\}$. We let $\mathcal{H}_{N,N'}$ be a class of universal hash functions (Carter and Wegman [4]) from $\mathbf{N}_N$ to $\mathbf{N}_{N'}$.

For the ease of exposition, we assume in the following that the portion p of nonempty Voronoi cells is known. This assumption can be removed using the techniques of Haussler, Kearns, Littlestone, and Warmuth [17].

4.2.1 Perfect Hashing

The first algorithm uses uniform hash functions and resorts to large physical memory to assure perfect hashing with high probability.[1]

Algorithm *LH1* (learning by perfect hashing):

1. Let $\mathcal{U}$ be an $\frac{\epsilon M_Y}{4K}$-cover of size N, where $N = \mathcal{N}(X, \frac{\epsilon M_Y}{4K}, d_X)$, and let $0 < p \ll 1$ be the fraction of non-empty Voronoi cells. Let $m = \frac{2pN}{\epsilon}\ln\frac{2pN}{\delta}$ be the sample

[1]The perfect hashing techniques as surveyed by Ramakrishna and Awasthi [37] assume a static set of keys, so we are not able to use these techniques for learning, which is dynamic in nature. However, when the learning is complete (the set of keys (addresses) is fixed), we can use perfect hashing techniques to reduce the size of physical memory.

size.

2. Let $N' = \frac{2}{\delta}(pN)^2$ be the size of physical memory $\mathcal{Z}$ and choose a uniform hash function h.

3. For each address i, if $h^{-1}(i)$ is not empty then we choose an arbitrary $1 \leq j \leq m$ such that $x_j \in h^{-1}(i)$ and set $\mathcal{Z}(i) = y_j$; otherwise we set $\mathcal{Z}(i)$ arbitrarily.

Theorem 5 *With probability at least $1 - \delta$, the expected error for Algorithm LH1 satisfies $\mathbf{er}_P(LH1(\overline{\xi})) \leq \epsilon M_Y$.*

Proof: Without any collision, by similar analysis as in the proof of Theorem 4, with probability at least $1 - \delta/2$, we have $\mathbf{er}_P(LH1(\overline{\xi})) \leq \epsilon M_Y$.

By choosing physical memory size as $N' = \frac{2}{\delta}(pN)^2$, we bound the probability that at least one hashing collision occurs by

$$(pN)^2 \frac{1}{\frac{2}{\delta}(pN)^2} \leq \frac{\delta}{2}.$$

Therefore, with probability at least $1 - \delta$, we have no collisions and $\mathbf{er}_P(LH1(\overline{\xi})) \leq \epsilon M_Y$. □

4.2.2 Universal Hashing

It is not necessary to avoid collisions completely. What we really need is a "good" hash function that incurs not too many collisions. The following algorithm uses universal hashing for finding a good hash function with high probability. We denote the Lipschitz bound by K.

Algorithm *LH2* (learning by universal hashing):

1. Let $\mathcal{U}$ be an $\frac{\epsilon M_Y}{4K}$-cover of size N, where $N = \mathcal{N}(X, \frac{\epsilon M_Y}{8K}, d_X)$, and let $0 < p \ll 1$ be the fraction of non-empty cells. Let $m = \frac{8pN}{\epsilon} \ln \frac{2pN}{\delta}$ be the sample size and let $N' = \frac{8}{\epsilon} pN$ be the size of physical memory $\mathcal{Z}$.

2. Repeat the following procedure $\log_{4/3}(2/\delta)$ times and choose the system with minimum empirical error: We choose a hash function h randomly from the class $\mathcal{H}_{N,N'}$ of universal hash functions and then call the subroutine $H(\overline{\xi}, h)$, which is given immediately below:

Subroutine H: Given a sample sequence $\overline{\xi}$ and a hash function h, for each address i, if $h^{-1}(i)$ is not empty then we choose an arbitrary $1 \leq j \leq m$ such that $x_j \in h^{-1}(i)$ and set $\mathcal{Z}(i) = y_j$; otherwise we set $\mathcal{Z}(i)$ arbitrarily.

Theorem 6 *With probability at least $1-\delta$, the expected error for Algorithm LH2 satisfies* $\mathbf{er}_P(LH2(\overline{\xi})) \leq \epsilon M_Y$.

Proof: For each Voronoi cell $Vor(u_i)$ with $P_X(Vor(u_i)) \geq \frac{\epsilon}{8pN}$, we have

$$\begin{aligned}\mathbf{Pr}(Vor(u_i) \cap \overline{\xi}_X = \emptyset) &\leq \left(1 - \frac{\epsilon}{8pN}\right)^{\frac{8pN}{\epsilon}\ln\frac{2pN}{\delta}} \\ &\leq \frac{\delta}{2pN}.\end{aligned}$$

Therefore, using sample size $m = \frac{8pN}{\epsilon}\ln\frac{2pN}{\delta}$, with probability at least $1-\delta/2$, all Voronoi cells with probability over $\epsilon/(8pN)$ will be hit by some sample point. By the property of universal hashing (Carter and Wegman [4]), for each Voronoi cell hit, the probability that the cell is involved in some hash collision is at most $pN/N' = \epsilon/8$. Therefore, by the Lipschitz bound, we have

$$\begin{aligned}&\mathbf{E}\left[d_Y(z_{h(\gamma(\mathbf{x}))}, \mathbf{y}) \mid Vor(u_i)\right] \\ &\quad\leq \frac{1}{P_X(Vor(u_i))}\left(\left(1 - \frac{\epsilon}{8}\right)\frac{\epsilon M_Y}{2} + \frac{\epsilon}{8}M_Y\right) \\ &\quad\leq \frac{5\epsilon M_Y}{8P_X(Vor(u_i))},\end{aligned}$$

where h is the random universal hash function. Thus, we have

$$\begin{aligned}\mathbf{E}\left[\mathbf{er}_P(A(\overline{\xi}, h))\right] &\leq \left(1 - \frac{\epsilon}{8}\right)\frac{5\epsilon M_Y}{8} + \frac{\epsilon M_Y}{8} \\ &\leq \frac{3\epsilon M_Y}{4},\end{aligned}$$

where the expectation is taken over $\mathcal{H}_{N,N'}$.

We say that a hash function h is "good" if the following inequality holds:

$$\mathbf{er}_P(H(\overline{\xi}, h)) \leq \epsilon M_Y.$$

By Markov's inequality, at least one fourth of hash functions in $\mathcal{H}_{N,N'}$ are good. Therefore, by calling subroutine H at least $\log_{4/3}(2/\delta)$ times, the probability that we do not get a good hash function is at most $\delta/2$. Thus, with probability at least $1-\delta$, we have $\mathbf{er}_P(LH2(\overline{\xi})) \leq \epsilon M_Y$. □

4.2.3 Hashing with Collision-Resolution

The physical memory size can be reduced to $O(pN)$ while maintaining an $O(1)$ worst-case access time by using collision-resolution techniques. This can be achieved, for example, by using coalesced hashing, which was analyzed in detail by Vitter and Chen [43] and Siegel [40].

4.3 LEARNING BY CLUSTERING

Although hashing techniques take advantage of the sparseness of distributions, they do not take advantage of the skewness of distributions. We can exploit the skewness of distributions by using clustering (or median) algorithms. Given a positive integer $s \leq m$, the (*continuous*) *s-median* (or *clustering*) *problem* is to find a median set $\mathcal{U} \subseteq X$ such that $|\mathcal{U}| = s$ and the average empirical distortion $\hat{d}_{\overline{\xi}_X}(\mathcal{U})$ is minimized. The *discrete s-median problem* is to restrict $\mathcal{U}$ to be a subset of $\overline{\xi}_X$.

For simplicity, we assume in the following that the quantization number $s = \mathcal{Q}_{P_X}(X, \frac{\epsilon M_Y}{4K}, d_X)$ is known. This assumption can be removed using the techniques Haussler, Kearns, Littlestone, and Warmuth [17]. In the following, we also assume that the Lipschitz bound holds with probability one.

4.3.1 Optimal Clustering

Ideally, we would like to use an algorithm for finding optimal clustering for learning:

Algorithm *LC1* (learning by optimal clustering):

1. Let $m = \Omega(\frac{ks}{\epsilon}\log s \log\frac{1}{\epsilon} + \frac{1}{\epsilon}\log\frac{1}{\delta})$ be the sample size, where s is the quantization number $\mathcal{Q}_{P_X}(X, \frac{\epsilon M_Y}{4K}, d_X)$.

2. Find the optimal $\mathcal{U}$ such that $\hat{d}_{\overline{\xi}_X}(\mathcal{U})$ is minimized.

3. For each $u_i \in \mathcal{U}$, we choose $1 \le j \le m$ such that x_j is the closest point to u_i and set $\mathcal{Z}(i) = y_j$.

Theorem 7 *With probability at least $1-\delta$, the expected error for Algorithm LC1 satisfies* $\mathbf{er}_P(LC1(\overline{\xi})) \le \epsilon M_Y$.

Proof: In Theorem 3, we choose $\alpha = 1/11$ and let $\nu = \epsilon M_Y/(2K)$. Thus, by choosing sample size as $\Omega(\frac{ks}{\epsilon}\log s\log\frac{1}{\epsilon} + \frac{1}{\epsilon}\log\frac{1}{\delta})$, with probability at least $1-\delta$, for all $V \subset X$ of size s, we have

$$\hat{d}_{\overline{\xi}_X}(V) \le \frac{6\mathbf{E}\left[d_X(\mathbf{x}, V)\right]}{5} + \frac{\epsilon M_Y}{20K},$$

and

$$\mathbf{E}\left[d_X(\mathbf{x}, V)\right] \le \frac{6\hat{d}_{\overline{\xi}_X}(V)}{5} + \frac{\epsilon M_Y}{20K}.$$

Let $\mathcal{U}^*$ be the optimal median set of size s with respect to P_X, then we have

$$\begin{aligned}
\mathbf{E}\left[d_X(\mathbf{x}, \mathcal{U})\right] &\le \frac{6\hat{d}_{\overline{\xi}_X}(\mathcal{U})}{5} + \frac{\epsilon M_Y}{20K} \\
&\le \frac{6\hat{d}_{\overline{\xi}_X}(\mathcal{U}^*)}{5} + \frac{\epsilon M_Y}{20K} \\
&\le \frac{6}{5}\left(\frac{6\mathbf{E}\left[d_X(\mathbf{x}, \mathcal{U}^*)\right]}{5} + \frac{\epsilon M_Y}{20K}\right) + \frac{\epsilon M_Y}{20K}.
\end{aligned}$$

Since $\mathcal{U}^*$ is optimal, we have $\mathbf{E}\left[d_X(\mathbf{x}, \mathcal{U}^*)\right] \le \frac{\epsilon M_Y}{4K}$. Therefore,

$$\begin{aligned}
\mathbf{E}\left[d_X(\mathbf{x}, \mathcal{U})\right] &\le \frac{6}{5}\left(\frac{6}{5}\frac{\epsilon M_Y}{4K} + \frac{\epsilon M_Y}{20K}\right) + \frac{\epsilon M_Y}{20K} \\
&\le \frac{\epsilon M_Y}{2K}.
\end{aligned}$$

We lose another factor of 2 by the way we set $\mathcal{Z}(i)$. The rest of the proof follows from the Lipschitz bound. $\square$

4.3.2 Approximate Clustering

Unfortunately, finding optimal clusters is $\mathcal{NP}$-hard even in Euclidean space (Kariv and Hakimi [20]; Garey and Johnson [12]; Papadimitriou [32]; Megiddo [26]). However, as shown by Lin and Vitter [24], we have approximate clustering algorithms with provably good performance guarantees. We may use these approximate clustering algorithms for learning:

Algorithm *LC2* (learning by approximate clustering):

1. Let $m = \Omega(\frac{ks}{\epsilon}\log s(\log\frac{ks}{\epsilon})^2 + \frac{1}{\epsilon}\log\frac{1}{\delta})$ be the sample size, where s is the quantization number $\mathcal{Q}_{P_X}(X, \frac{\epsilon M_Y}{4K}, d_X)$.

2. Apply the greedy (discrete) s-median algorithm of Lin and Vitter [24] with relative error bound on distortion as $1/8$. (For convenience, the greedy s-median algorithm is given in the appendix.) Let $\mathcal{U}$ be the median set returned by the greedy s-median algorithm.

3. For each $x_j = u_i \in U$ we set $\mathcal{Z}(i) = y_j$.

By Corollary 3 in the Appendix and the triangle inequality, we have the following corollary:

Corollary 2 *Let $\mathcal{U}$ be the median set returned by the greedy s-median algorithm and let $\mathcal{U}^*$ be the set of optimal s-medians. Then we have*

$$\hat{d}_{\overline{\xi}_X}(\mathcal{U}) \leq 2(1+\frac{1}{8})\hat{d}_{\overline{\xi}_X}(\mathcal{U}^*) \leq \frac{9}{4}\hat{d}_{\overline{\xi}_X}(\mathcal{U}^*).$$

and

$$|\mathcal{U}| = O(s\log m).$$

Proof: Let $\mathcal{U}^* = \{u_1, \ldots, u_s\}$ be the set of optimal s-medians. Then by replacing each point $u_i \in \mathcal{U}^*$ by its nearest neighbor v_i in $\overline{\xi}_X$, we obtain a median set $\mathcal{V} \subseteq \overline{\xi}_X$ such that

$$\begin{aligned}
\hat{d}_{\overline{\xi}_X}(\mathcal{V}) &= \frac{1}{m}\sum_{i=1}^{m} d_X(x_i, \mathcal{V}) \\
&= \frac{1}{m}\sum_{i=1}^{s}\sum_{x\in Vor(u_i)\cap\overline{\xi}_X} d_X(x, \mathcal{V}) \\
&\leq \frac{1}{m}\sum_{i=1}^{s}\sum_{x\in Vor(u_i)\cap\overline{\xi}_X} d_X(x, v_i).
\end{aligned}$$

By the triangle inequality, we have

$$\begin{aligned}\hat{d}_{\overline{\xi}_X}(\mathcal{V}) &\leq \frac{1}{m}\sum_{i=1}^{s}\sum_{x\in Vor(u_i)\cap\overline{\xi}_X}(d_X(x,u_i)+d_X(u_i,v_i))\\ &\leq \frac{1}{m}\sum_{i=1}^{s}\sum_{x\in Vor(u_i)\cap\overline{\xi}_X}2d_X(x,u_i)\\ &= 2\hat{d}_{\overline{\xi}_X}(\mathcal{U}^*).\end{aligned}$$

The rest of the proof follows from Corollary 3 in the Appendix. □

Theorem 8 *With probability at least $1-\delta$, the expected error for Algorithm LC2 satisfies* $\mathbf{er}_P(LC2(\overline{\xi}))\leq \epsilon M_Y$.

Proof: We apply Theorem 3 with $\alpha = 1/11$ and $\nu = \epsilon M_Y/(2K)$. By using $m = \Omega(\frac{ks}{\epsilon}\log s(\log\frac{ks}{\epsilon})^2+\frac{1}{\epsilon}\log\frac{1}{\delta})$ sample points, with probability at least $1-\delta$, for all $V\subset X$ of size at most $|\mathcal{U}|$, we have

$$\hat{d}_{\overline{\xi}_X}(V)\leq\frac{6\mathbf{E}\left[d_X(\mathbf{x},V)\right]}{5}+\frac{\epsilon M_Y}{20K},$$

and

$$\mathbf{E}\left[d_X(\mathbf{x},V)\right]\leq\frac{6\hat{d}_{\overline{\xi}_X}(V)}{5}+\frac{\epsilon M_Y}{20K}.$$

Let $\mathcal{U}^*$ be the set of optimal s-medians. By Corollary 2 and by algebraic manipulations similar to the proof of Theorem 7, we have

$$\mathbf{E}\left[d_X(\mathbf{x},\mathcal{U})\right]\leq\frac{\epsilon M_Y}{K}.$$

The rest of the proof follows from the Lipschitz bound. □

4.4 SUMMARY

We summarize the results of this section in Table 1. We remark that, in $\Re^k$, the covering number is exponential in the dimensionality of the input space. That is, we have $N=\mathcal{N}(X,\frac{\epsilon M_Y}{4K},d_X)=\Theta((\frac{1}{\epsilon})^k)$. On the other hand, in practice, it is reasonable to assume that the quantization number $s=\mathcal{Q}_{P_X}(X,\frac{\epsilon M_Y}{4K},d_X)$ is a low-degree polynomial in $\frac{1}{\epsilon}$, as explained in Section 1. In such typical cases, clustering algorithms reduce the dependency of memory size on dimensionality by an exponential factor.

5 TREE-STRUCTURED SYSTEMS

In a tree-structured system, the encoder partitions the input space into a hierarchy of regions. An input is mapped to the memory location corresponding to the region represented by a leaf. The full definition of tree-structured systems is given in Section 2.1. We call the encoders of tree-structured systems the *tree-structured encoders.*

Lemma 3 *Let G_s be the tree-structured systems of size s and let d_X be the Euclidean metric. For each possible encoder γ of G_s, we define $f_\gamma(x) = d_X(x, u_{\gamma(x)})$ and let $\Gamma_s : X \to [0, M_X]$ be the class of all such functions. Then we have $\mathbf{dim_P}(\Gamma_s) \leq 2(k+1)(s-1)\log(3(s-1)) = O(ks\log s)$.*

Proof: There are $s-1$ branches in a tree of size s, in which each branch corresponds to a comparison. By derivation similar to the proof of Lemma 2, we have $\mathbf{dim_P}(\Gamma_s) \leq 2(k+1)(s-1)\log(3(s-1)) = O(ks\log s)$. □

Lemma 3 and Corollary 1 imply the following result:

Theorem 9 *Let Γ_s be defined as in Lemma 3. Assume $\nu > 0$ and $0 < \alpha < 1$. Let P_X be a probability measure on X and $\overline{\xi}_X$ be generated by m independent draws from X according to P_X. If the sample size is*

$$m = \Omega\left(\frac{M_X}{\alpha^2\nu}\left(ks\log s\log\frac{M_X}{(\alpha\sqrt{\alpha})\nu} + \log\frac{1}{\delta}\right)\right),$$

then we have

$$\mathbf{Pr}\{\exists f \in \Gamma_s \mid d_\nu(\hat{\mathbf{E}}_{\overline{\xi}_X}(f), \mathbf{E}(f)) > \alpha\} \leq \delta.$$

In the following we outline an algorithm for building tree-structured systems:

1. Construct a tree-structured encoder for the input space from the x-components of the sample.

2. Estimate a functional value for each node of the tree by averaging the y-components of examples covered by the region represented by that node.

The smoothness of the function to be learned assures that the resulting system has small expected error. The algorithm for building a tree-structured encoder is given by Lin and Vitter [23,24]. In addition to memory-based learning, the algorithm also has applications to regression, computer graphics, and lossy image compression (Lin and Vitter [23]).

6 HIGHER-ORDER SYSTEMS

In a higher-order memory-based learning system, an input can activate more than one memory location. Higher-order learning systems have the advantages of fault tolerance and possibly better generalization ability given a limited number of examples. By fault tolerance, we mean the capability to deal with memory failures or misclassification of sample points.

In this section, we look at the r-nearest-neighbor systems and receptive-field systems based upon the combinations of first-order systems:

6.1 VORONOI SYSTEMS OF HIGHER ORDER

The definition for the Voronoi systems of order r (r-nearest-neighbor systems) is given in Section 2.1. In this section we extend our analysis in Section 3 to the rth-order Voronoi Systems. We call the encoders of Voronoi systems of order r the *Voronoi encoders of order* r.

Lemma 4 *Let G_s^r be the Voronoi systems of order r and size s and let d_X be the Euclidean distance. For each possible encoder γ of G_s^r, we define*

$$f_\gamma(x) = \frac{1}{r}\sum_{i=1}^{r} d_X(x, u_{\gamma^{(i)}(x)}),$$

where $\gamma^{(i)}(x)$ maps an input x to its ith nearest neighbor in $\mathcal{U}$ and let $\Gamma_s^r : X \to [0, M_X]$ be the class of all such functions. Then we have $\mathbf{dim_P}(\Gamma_s^r) = O(krs\log r \log s)$.

Proof: By the definition of $f_\gamma(x)$, it is clear that the pseudo-dimension of Γ_s^r is bounded by the pseudo-dimension of sums of r functions from Γ_s, which is defined as in Lemma 2. By derivation similar to the proof of Lemma 2, we have $\mathbf{dim_P}(\Gamma_s^r) = O(krs\log r \log s)$. □

Lemma 4 and Corollary 1 imply the following:

Theorem 10 *Let Γ_s^r be defined as in Lemma 4. Assume $\nu > 0$ and $0 < \alpha < 1$. Let P_X be a probability measure on X and $\overline{\xi}_X$ be generated by m independent draws from X according to P_X. If the sample size is*

$$m = \Omega\left(\frac{M_X}{\alpha^2 \nu}\left(krs \log r \log s \log \frac{M_X}{(\alpha\sqrt{\alpha})\nu} + \log \frac{1}{\delta}\right)\right),$$

then we have

$$\mathbf{Pr}\{\exists f \in \Gamma_s^r \mid d_\nu(\hat{\mathbf{E}}_{\overline{\xi}_X}(f), \mathbf{E}(f)) > \alpha\} \leq \delta.$$

6.2 RECEPTIVE-FIELD SYSTEMS

In a receptive-field system, the regions may overlap. In this section, we propose to model the receptive-field systems as weighted sums of first-order Voronoi systems.

Definition 11 Let G_s be the class of (first-order) Voronoi systems as defined in Section 3. The r-combinations G_s^r of Voronoi systems are defined as the weighted sums of r Voronoi systems. That is, $G_s^r = \{\sum_{i=1}^r w_i g_i \mid g_i \in G_s \text{ and } 0 \leq w_i \leq M_Y\}$.

A receptive-field system as defined above can be arranged in a "multi-resolution" manner (Moody [29]), that is, as a sum of r Voronoi systems of different sizes. The learning algorithm for such systems can start by approximating the function to be learned by the smallest (lowest-resolution) component system, and then approximating the errors by the second smallest component system, and so forth, until the largest (highest-resolution) component system is trained.

7 CONCLUSIONS

In this paper, we propose a model for memory-based learning and use it to analyze several methods for learning smooth functions by memory-based learning systems. Our model is closely related to the generalized PAC learning model of Haussler [16] and the methods of vector quantization in data compression. Our main result is that we can build memory-based learning systems using new clustering algorithms (Lin and Vitter [24]) to PAC-learn in polynomial time using only polynomial storage in typical situations. We also extend our analysis to tree-structured and higher-order memory-based learning systems.

The memory-based learning systems that we have examined in this paper approximate the functional value in each region by a constant. In practice, we might get better approximations by using more complicated basis functions. However, this usually makes the training problem harder; most work along this line has been mostly experimental in terms of computational complexity. Interested readers are referred to the work of Friedman [11], Moody and Darken [30], and Poggio and Girosi [33,34].

Our memory-based learning algorithms mainly take advantage of the skewness of distributions over the input space and assume the smoothness of functions over the input space. However, the degree of smoothness may vary widely from one region to the other (Dean [7]). In practice, after the initial clustering, we may estimate the degree of smoothness of each region and then merge or split regions according to their degrees of smoothness. Alternatively, we could incorporate either *a priori* or observed local estimates of smoothness directly into the distance matrix in such a way so that minimizing the distance corresponds to minimizing the error; for example, we could define $d_X(x, x')$ to be $d_Y(f(x), f(x'))$. From a theoretical viewpoint, we must develop models that adequately capture this property and are computationally tractable.

Appendix: Approximate Clustering

In this appendix, we adapt the greedy (discrete) s-median algorithm of Lin and Vitter [24] to do the clustering needed for Algorithm *LC2* in Section 4.3.2. The discrete s-median problem is defined as follows: Let $\overline{\xi}_X = \langle x_1, \ldots, x_m \rangle$ be a sequence of points in X and let s be a positive integer. The goal is to select a subset $\mathcal{U} \subseteq \overline{\xi}_X$ of s points such that the average distance (distortion)

$$\hat{d}_{\overline{\xi}_X}(\mathcal{U}) = \frac{1}{m} \sum_{i=1}^{m} d_X(x_i, \mathcal{U}).$$

is minimized.

The discrete s-median problem can be formulated as a 0-1 integer program of minimizing

$$\frac{1}{m} \sum_{i=1}^{m} \sum_{j=1}^{m} d_X(x_i, x_j) p_{ij} \qquad (1)$$

subject to

$$\begin{aligned}
\sum_{j=1}^{m} p_{ij} &= 1, & i &= 1,\ldots,m, & (2)\\
\sum_{j=1}^{m} q_j &\le s, & & & (3)\\
p_{ij} &\le q_j, & i,j &= 1,\ldots,m, & (4)\\
p_{ij}, q_j &\in \{0,1\}, & i,j &= 1,\ldots,m, & (5)
\end{aligned}$$

where $q_j = 1$ if and only if x_j is chosen as a cluster center, and $p_{ij} = 1$ if and only if $q_j = 1$ and x_i is "assigned" to x_j.

The linear program relaxation of the above program is to allow q_j and p_{ij} to take rational values between 0 and 1. Clearly, the optimal fractional solution (linear program solution) is a lower bound on the solutions of the discrete s-median problem.

Our greedy algorithm for the s-median problem works as follows:

1. Solve the linear program relaxation of the discrete s-median problem by linear programming techniques; denote the fractional solution by $\widehat{q}, \widehat{p}$.
2. For each i, compute $\widehat{D_i} = \sum_{j=1}^{m} d_X(x_i, x_j)\widehat{p}_{ij}$.
3. Given a relative error bound $c > 0$, for each j such that $\widehat{q}_j > 0$, construct a set S_j: A point x_i is in S_j if and only if $d_X(x_i, x_j) \le (1+c)\widehat{D_i}$.
4. Apply the greedy set cover algorithm (Johnson [19]; Lovász [25]; Chvátal [5]): Choose the set which covers the most uncovered points. Repeat this process until all points are covered. Let U be the set of indices of sets chosen by the greedy heuristic. Output $\mathcal{U} = \{x_j\}_{j \in U}$ as the median set.

The linear programming problem can be solved in provably polynomial time by the ellipsoid algorithm (Khachian [22]) or by the interior point method (Karmarkar [21]). The simplex method (Dantzig [6]) works very efficiently in practice, although in the worst case its performance is not polynomial-time.

The results of Lin and Vitter [24]yield the following application:

Corollary 3 *Given any $c > 0$, the greedy algorithm outputs a median set $\mathcal{U}$ of size less than*

$$(1 + 1/c)s(\ln n + 1)$$

such that

$$\hat{d}_{\bar{\xi}_X}(\mathcal{U}) \leq (1+c)\widehat{D},$$

where $\widehat{D}$ *is the average distance of the optimal fractional solution for the discrete s-median problem.*

Acknowledgements

Support was provided in part by a National Science Foundation Presidential Young Investigator Award CCR–9047466 with matching funds from IBM, by NSF research grant CCR–9007851, and by Army Research Office grant DAAL03–91–G–0035.

References

[1] J. S. Albus, *Brains, Behaviour, and Robotics*, Byte Books, Peterborough, NH, 1981.

[2] J. S. Albus, "Data Storage in the Cerebellar Model Articulation Controller (CMAC)," *Journal of Dynamic Systems, Measurement, and Control* (September 1975), 228–233.

[3] J. S. Albus, "A New Approach to Manipulator Control: The Cerebellar Model Articulation Controller (CMAC)," *Journal of Dynamic Systems, Measurement, and Control* (September 1975), 220–227.

[4] J. L. Carter and M. N. Wegman, "Universal Classes of Hash Functions," *Journal of Computer System and Science* 18 (April 1979), 143–154.

[5] V. Chvátal, "A Greedy Heuristic for the Set-Covering Problem," *Mathematics of Operations Research* 4 (1979), 233–235.

[6] G. Dantzig, "Programming of Interdependent Activities, II, Mathematical Models," in *Activity Analysis of Production and Allocation*, John Wiley & Sons, Inc, New York, 1951, 19–32.

[7] T. L. Dean and M. P. Wellman, *Planning and Control*, Morgan Kaufmann Publishers, 1991.

[8] L. Devroye, "Automatic Pattern Recognition: A Study of the Probability of Error," *IEEE Transactions on Pattern Recognition and Machine Intelligence* 10 (1988), 530–543.

[9] R. M. Dudley, "Central Limit Theorems for Empirical Measures," *Annals of Probability* 6 (1978), 899–929.

[10] R. M. Dudley, "A Course on Empirical Processes," in *Lecture Note in Mathematics 1097*, 1984.

[11] J. H. Friedman, *Multivariate Adaptive Regression Splines*, Technical Report 102, Standford University, Lab for Computational Statistics, 1988.

[12] M. R. Garey and D. S. Johnson, *Computers and intractability: A Guide to the Theory of $\mathcal{NP}$-completeness*, W. H. Freeman and Co., San Francisco, CA, 1979.

[13] A. Gersho, "On the Structure of Vector Quantizers," *IEEE Transactions on Information Theory* 28 (March 1982), 157–166.

[14] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*, Kluwer Academic Press, Massachusetts, 1991.

[15] R. M. Gray, "Vector Quantization," *IEEE ASSP Magazine* (April 1984), 4–29.

[16] D. Haussler, "Generalizing the PAC Model: Sample Size Bounds from Metric Dimension-Based Uniform Convergence Results," in *Proceedings of the 30th Annual IEEE Symposium on Foundations of Computer Science*, 1989, 40–45.

[17] D. Haussler, M. Kearns, N. Littlestone, and M. K. Warmuth, "Equivalence of Models for Polynomial Learnability," *Information and Computation* 95 (1991), 129–161.

[18] D. Haussler and P. Long, "A Generalization of Sauer's Lemma," UCSC-CRL-90-15, April 1990.

[19] D. S. Johnson, "Approximation Algorithms for Combinatorial Problems," *Journal of Computer and System Sciences* 9 (1974), 256–278.

[20] O. Kariv and S. L. Hakimi, "An Algorithmic Approach to Network Location Problems. II: The p-Medians," *SIAM Journal on Applied Mathematics* (1979), 539–560.

[21] N. Karmarkar, "A New Polynomial-Time Algorithm for Linear Programming," *Combinatorica* 4 (1984), 373–395.

[22] L. G. Khachiyan, "A Polynomial Algorithm in Linear Programming," *Soviet Math. Doklady* 20 (1979), 191–194.

[23] J.-H. Lin and J. S. Vitter, "Nearly Optimal Vector Quantization via Linear Programming," in *Proceedings of the IEEE Data Compression Conference*, Snowbird, Utah, March 1992, 22–31.

[24] J.-H. Lin and J. S. Vitter, "ϵ-Approximations with Minimum Packing Constraint Violation," in *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*, Victoria, BC, Canada , May 1992, 771–782.

[25] L. Lovász, "On the Ratio of Optimal Integral and Fractional Covers," *Discrete Mathematics* 13 (1975), 383–390.

[26] N. Megiddo and K. J. Supowit, "On the Complexity of Some Common Geometric Location Problems," *SIAM Journal on Computing* 13 (1984), 182–196.

[27] W. T. Miller, "Sensor-Based Control of Robotic Manipulators Using a General Learning Algorithms," *IEEE Journal of Robotics and Automation* 3 (April 1987), 157–165.

[28] W. T. Miller, F. H. Glanz, and L. G. Kraft, "Application of a General Learning Algorithm to the Control of Robotic Manipulators," *International Journal of Robotics Research* 6 (Summer 1987), 84–98.

[29] J. Moody, "Fast Learning in Multi-Resolution Hierarchies," in *Advances in Neural Information Processing Systems I*, Morgan Kaufmann Publisher , 1989, 29–39.

[30] J. Moody and C. Darken, "Learning with Localized Receptive Fields," in *Proceedings of the 1988 Connectionist Models Summer School*, Morgan Kaufmann Publisher , 1988, 133–143.

[31] A. W. Moore, *Acquisition of Dynamic Control Knowledge for Robotic Manipulator*, Manuscript, 1989.

[32] C. H. Papadimitriou, "Worst-case and Probabilistic Analysis of a Geometric Location Problem," *SIAM Journal on Computing* 10 (1981), 542–557.

[33] T. Poggio and F. Girosi, "A Theory of Networks for Approximation and Learning," MIT. Artificial Intelligence Laboratory, A. I. Memo No. 1140, Boston, MA, 1989.

[34] T. Poggio and F. Girosi, "Extensions of a Theory of Networks for Approximation and Learning: Dimensionality Reduction and Clustering," MIT. Artificial

Intelligence Laboratory, A. I. Memo No. 1167, Boston, MA, 1990.

[35] D. Pollard, *Convergence of Stochastic Processes*, Springer-Verlag New York Inc, 1984.

[36] D. Pollard, *Empirical Processes: Theory and Applications*, NSF-CBMS Regional Conference Series in Probability and Statistics Volume 2, 1990.

[37] M. V. Ramakrishna and V. Awasthi, Manuscript, December 1991.

[38] E. A. Riskin, *Variable Rate Vector Quantization of Images*, Ph. D. Dissertation, Stanford University, 1990.

[39] N. Sauer, "On the Density of Families of Sets," *Journal of Combinatorial Theory (A)* 13 (1972), 145–147.

[40] A. Siegel, "Coalesced Hashing is Computably Good," Manuscript, 1991.

[41] V. N. Vapnik, *Estimation of Dependences Based on Empirical Data*, Springer Verlag, New York, 1982.

[42] V. N. Vapnik and A. Ya. Chervonenkis, "On the Uniform Convergence of Relative Frequencies of Events to their Probabilities," *Theory of Probability and its Applications* (1971), 264–280.

[43] J. S. Vitter and W.-C. Chen, *Design and Analysis of Coalesced Hashing*, Oxford University Press, 1987.