Leaving the Span

Manfred K. Warmuth[1],* and S.V.N. Vishwanathan[2]

[1] Computer Science Department, University of California, Santa Cruz,
CA 95064, U.S.A.
manfred@cse.ucsc.edu
[2] Machine Learning Program, National ICT Australia**,
Canberra, ACT 0200, Australia
SVN.Vishwanathan@nicta.com.au

Abstract. We discuss a simple sparse linear problem that is hard to learn with any algorithm that uses a linear combination of the training instances as its weight vector. The hardness holds even if we allow the learner to embed the instances into any higher dimensional feature space (and use a kernel function to define the dot product between the embedded instances). These algorithms are inherently limited by the fact that after seeing k instances only a weight space of dimension k can be spanned.

Our hardness result is surprising because the same problem can be efficiently learned using the exponentiated gradient (EG) algorithm: Now the component-wise logarithms of the weights are essentially a linear combination of the training instances and after seeing k instances. This algorithm enforces *additional constraints* on the weights (all must be non-negative and sum to one) and in some cases these constraints alone force the rank of the weight space to grow as fast as 2^k.

1 Introduction

Linear methods are inadequate for many learning problems. However, if linear methods are enhanced by the *kernel trick*, then they can lead to powerful learning methods. For this purpose, the instance domain $\mathcal{X}$ is mapped to a Reproducing Kernel Hilbert Space (RKHS) F via a possibly non-linear embedding map Φ. Now linear models in the feature space F can describe highly non-linear models in the original instance space $\mathcal{X}$ and linear learning algorithms (in feature space) can become powerful learning methods. The caveat is that this method requires a restriction to learning algorithms whose weight vectors are linear combinations of

* Supported by NSF grant CCR 9821087. Part of this work was done while the first author visited NICTA.

** mm National ICT Australia is funded by the Australian Government's Department of Communications, Information Technology and the Arts and the Australian Research Council through Backing Australia's Ability and the ICT Center of Excellence program.

P. Auer and R. Meir (Eds.): COLT 2005, LNAI 3559, pp. 366–381, 2005.

the embedded training instances.[1] In this case, computing dot products between the weight vector and a new embedded instance reduces to efficiently computing the dot product between two embedded instances, i.e., $\langle \phi(\mathbf{x}), \phi(\tilde{\mathbf{x}}) \rangle$. This is done via the so-called *kernel function* $k(\mathbf{x}, \tilde{\mathbf{x}}) = \langle \phi(\mathbf{x}), \phi(\tilde{\mathbf{x}}) \rangle$.

In this paper the following set of conditions is called **kernel paradigm**: The weight vector (in feature space) is a linear combination of embedded training instances, the dot product of this weight vector with new instances is computed via a kernel function and the individual features (components of the embedded instances $\phi(\mathbf{x})$) are not accessed by the algorithm.

Now, consider the following *sparse linear* learning problem first discussed in Kivinen and Warmuth (1997), Kivinen et al. (1997): The instances $\mathbf{x}_t$ are the rows of an n-dimensional Hadamard matrix and the possible targets are one of the n columns of the matrix. In other words, if the target is the i-th column, then the instances are labeled by the i-th feature and this target corresponds to the standard basis vector $\mathbf{e}_i$ (This vector has a one in the i-th position and zeros elsewhere). Hadamard matrices have ±1 entries and orthogonal rows. Therefore, as argued before in Kivinen and Warmuth (1997), Kivinen et al. (1997), this problem is hard to learn when the weight vector is required to be a linear combinations of instances: Any linear combination of past instances predicts zero on any new instance (labeled ±1) and thus incurs constant loss.

In this paper, we show that even if the learner is allowed to embed the instances into any Euclidean space (via the use of a kernel), the above sparse linear problem is still hard to learn[2]. Any algorithm that predicts with a linear combination of the embedded instances seen so far has the property that after k instances it can only span a weight space of rank k. However, the n standard basis vectors vectors (our possible target weight vectors) form a matrix of rank n. We show that for any k training instances and any embedding into some Euclidean space (of any dimension), there always is one of the targets for which the average square loss is at least $1 - \frac{k}{n}$. Thus, after seeing half of all the instances, the average square loss is still a half.

The first question is, what is the family of algorithms that always predicts with a linear combination of the instances. The Representer Theorem and various extensions (Kimeldorf and Wahba, 1971, Schölkopf et al., 2001) provide minimization problems whose solutions are always linear combinations of instances. A more general geometric condition on the learning algorithm is given in Kivinen et al. (1997): *Any linear algorithm whose predictions are invariant with respect to a rotation of the embedded instances must predict with a weight vector that is a linear combination of the embedded instances.* However, it is important to note that our lower bounds hold for any algorithm that predicts

[1] The only specialized exceptions are the algorithms of Takimoto and Warmuth (2003).

[2] Our hardness result does not hold for embeddings in arbitrary dot product spaces. However, we believe that this is only a technical restriction.

with a linear combination of instances.[3] This includes algorithms that choose the coefficients of the linear combination by accessing the individual components of the embedded instances - which breaks the kernel paradigm.

Our lower bound currently only holds for linear regression with respect to the square loss. We conjecture that changing the loss function for the linear prediction algorithm does not alleviate this problem, i.e., we conjecture that for any non-negative convex loss function L, s.t. $L(y, \hat{y}) \geq 1$ whenever $|y - \hat{y}| \geq 1$, there always is one of the targets with average loss $1 - \frac{k}{n}$. Along these lines, we prove in the full paper that the lower bounds hold for the following generalization of the square loss: $L_p(y, \hat{y}) = |y - \hat{y}|^p$, for $1 < p \leq \infty$. The proof requires considerably more tools from linear algebra.

The lower bounds may be surprising because there are simple linear learning algorithms that can easily learn the above sparse linear problem. One such algorithm belongs to the Exponentiated Gradient (EG) family of algorithms which essentially have the property that the component-wise logarithms of the linear weights are a linear combination of the (embedded) training instances. By varying the coefficient of a single instance, the set of possible weight vectors reachable is already as high as the number of distinct components in the instance. Also the weight space based on k instances can contain up to 2^k standard basis vectors.

The crucial feature of the EG algorithm seems to be that it maintains constraints on the weights: the weights must be non-negative and sum to one. We can show that in some special cases these constraints alone let us reach a weight space of rank 2^k after seeing k examples. Not surprisingly, the EG algorithm, as well as any algorithm that enforces the constraints explicitly, require access to the weights of the individual features, and this breaks the kernel paradigm[4].

Following Kivinen and Warmuth (1997), the goal of this type of research is to characterize which type of linear learning algorithm is suitable for a given class of linear problems. The focus of this paper is to explore which linear prediction algorithms are suitable for sparse linear problems.

Key Open Problem: Can similar lower bounds be proven for linear thresholded predictors, i.e., now $\hat{y} = \sigma(\langle \mathbf{w}, \phi(\mathbf{x}) \rangle)$, where σ is the threshold function and $\mathbf{w}$ a linear combination of the embedded instances.

Related Work: There has been an on-going discussion of the advantages and disadvantages of kernel algorithms versus the multiplicative algorithms such as the EG algorithm and the Winnow algorithm (Littlestone, 1988). In short, multiplicative algorithms often generalize well after seeing only few examples in the case when the target is sparse. However, for harder learning problems, exponentially many weights need to be manipulated explicitly and this is too expensive

[3] This includes the work of Cristianini et al. (1999) that uses the EG algorithm to determine the coefficients of the linear combination.

[4] The only exceptions to this that we know of are the updates discussed in Takimoto and Warmuth (2003), where, in polynomial time exponentially many weights are updated with the EG algorithm. Curiously enough special kernels are used for this update.

((e.g., Section 9.6 of Kivinen and Warmuth, 1997) and Khardon et al. (2001)). In contrast, the kernel algorithm may converge slower for the same problems, but the kernel trick allows us to implicitly manipulate exponentially many feature weights at a low computational cost.

The embedding map can be seen as a special case of a reduction between prediction problems. For a more general notion of reduction and non-reducibility results that take the complexity of the learning problem into account see Pitt and Warmuth (1993), Warmuth (1989). Many worst-case loss bounds (e.g., Gentile and Warmuth, 1999) and generalization bounds (Schapire et al., 1998) are known to improve with the size of the margin. The goal is therefore to choose embeddings with large margins. To obtain a scaling-invariant notion of margin we must normalize the margin by the product of a pair of dual norms: the maximum p-norm of the instances and the q-norm of the target weight vector, where $\frac{1}{p} + \frac{1}{q} = 1$. In Ben-David et al. (2002) it was shown that with high probability a large fraction of the concept classes with a fixed VC dimension cannot be embedded with a 2-2-margin other than the *trivial* margin of $\frac{1}{\sqrt{n}}$, where n is the number of points. On the other hand (Forster et al., 2001) showed upper bounds on the 2-2 margin of a concept class in terms of its operator norm (the largest singular value of a matrix).

The family of algorithms whose weight vector is a linear combination of the instances seems to relate to the 2-2-margin. However, the performance of the EG algorithm seems to relate to the 1-∞-margin. For our simple Hadamard problem there is no embedding with a 2-2-margin better than $\frac{1}{\sqrt{n}}$ (the trivial 2-2-margin) (Forster et al., 2001). However, the 1-∞-margin is 1 for this problem and this seems to be the key reason why the EG algorithm does well in this case.

In this paper we prove *lower bounds* for the family of algorithms that predicts with a linear combination of the instances. However, we don't use pairs of dual norms (as was done in Kivinen and Warmuth (1997)) and we also completely bypass the concept of margins. Linear classifiers with large margins have good generalization error, but we know of no lower bounds in terms of margins (see Herbrich et al. (2005) for related experiments). Instead, we prove our lower bounds using the Singular Value Decomposition[5] (SVD) of the matrix defining the linear problem and a simple averaging argument.

2 Hadamard Matrices and SVD

We make use of properties of *Hadamard matrices* in various proofs. A Hadamard matrix is an orthogonal matrix with $\{\pm 1\}$ elements. The following definition allows us to recursively define *Hadamard matrices*: When $n = 2^d$ for some d, the $n \times n$ Hadamard matrix $\mathbf{H}^n$ is given by the following recurrence:

[5] In Forster et al. (2001) it was shown that the 2-2 margin of a concept class is *upper bounded* by the largest singular value over n.

$$\mathbf{H}^1 = (+1) \quad \mathbf{H}^2 = \begin{pmatrix} +1 & +1 \\ +1 & -1 \end{pmatrix} \quad \mathbf{H}^4 = \begin{pmatrix} +1 & +1 & +1 & +1 \\ +1 & -1 & +1 & -1 \\ +1 & +1 & -1 & -1 \\ +1 & -1 & -1 & +1 \end{pmatrix} \quad \mathbf{H}^{2n} = \begin{pmatrix} \mathbf{H}^n & \mathbf{H}^n \\ \mathbf{H}^n & -\mathbf{H}^n \end{pmatrix}$$

We use the shorthand $\mathbf{H} = \mathbf{H}^n$, where the dimension n is understood from the context. Note that all rows of the Hadamard matrix $\mathbf{H}$ are orthogonal and of length $\sqrt{n}$.

In the *Hadamard Learning Problem*, the examples are the rows of the Hadamard matrix labeled by one of the columns. So there are n instances and n possible targets.

A matrix $\mathbf{M} \in \mathbb{R}^{n\times m}$ can be decomposed as $\mathbf{M} = \mathbf{U}\,\mathbf{S}\,\mathbf{V}^\top$ where $\mathbf{U} \in \mathbb{R}^{n\times n}$, $\mathbf{V} \in \mathbb{R}^{m\times m}$ are orthogonal and $\mathbf{S} \in \mathbb{R}^{n\times m}$ is a diagonal matrix, i.e., $S_{ij} = 0$ for $i \neq j$. Furthermore, the diagonal elements of $\mathbf{S}$ are sorted and non-negative, i.e., $S_{11} \geq S_{22} \geq \ldots \geq S_{qq} \geq 0$, where $q = \min\{m, n\}$. Henceforth, we will use s_i to denote $S_{i,i}$. If the rank r of $\mathbf{M}$ is less than q, then exactly the last $q - r$ diagonal entries are zero. Furthermore, the numbers s_i (singular values) are uniquely determined by the square roots of the eigenvalues of $\mathbf{M}\,\mathbf{M}^\top$. The columns of $\mathbf{U}$ are eigenvectors of $\mathbf{M}\,\mathbf{M}^\top$ and the columns of $\mathbf{V}$ are eigenvectors of $\mathbf{M}^\top\,\mathbf{M}$ (arranged in the same order as the corresponding eigenvalues s_i^2). Such a decomposition is called the Singular Value Decomposition (SVD) (e.g., Theorem 7.3.5 of Horn and Johnson, 1985). Under some mild technical assumptions, the SVD can also be extended to bounded linear operators.

The Frobenius norm of a matrix $\mathbf{M} \in \mathbb{R}^{n\times m}$ is defined as $||\,\mathbf{M}\,||_F = \sqrt{\sum_{i=1}^n \sum_{j=1}^m |M_{i,j}|^2}$. It is invariant under orthogonal transformations and

$$||\,\mathbf{M}\,||_F^2 = s_1^2 + \ldots + s_q^2 \qquad q = \min\{m, n\}. \tag{1}$$

The following theorem allows us to write the best rank-k approximation to a given matrix $\mathbf{M}$ in terms of its SVD (Page 450 (Problem 1) and Example 7.4.5, Horn and Johnson, 1985).

Theorem 1. *Let* $\mathbf{M} = \mathbf{U}\,\mathbf{S}\,\mathbf{V}^\top$ *denote the SVD of* $\mathbf{M} \in \mathbb{R}^{n\times m}$. *For* $k < r = \operatorname{rank}(\mathbf{M})$ *define* $\mathbf{M}_k = \mathbf{U}\,\widehat{\mathbf{S}}\,\mathbf{V}^\top$ *where* $\widehat{\mathbf{S}} \in \mathbb{R}^{n\times m}$ *with* $\hat{s}_i = s_i$ *for* $i = 1, \ldots, k$ *and* $\hat{s}_j = 0$ *for* $k < j \leq m$. *Then*

$$\min_{\operatorname{rank}(\widehat{\mathbf{M}})=k} ||\,\mathbf{M} - \widehat{\mathbf{M}}\,||_F^2 = ||\,\mathbf{M} - \mathbf{M}_k\,||_F^2 = \sum_{j=k+1}^{q} s_j^2, \qquad q = \min\{m, n\}.$$

For a Hadamard matrix $\mathbf{H}$ of dimension $n\times n$, it is easy to see that $\operatorname{rank}(\mathbf{H}) = n$ and $\mathbf{H}\,\mathbf{H}^\top = n\,\mathbf{I}$. Thus all eigenvalues of $\mathbf{H}\,\mathbf{H}^\top$ are equal to n and the n singular values s_i are equal to $\sqrt{n}$. The flat spectrum of the Hadamard matrix will be used later to prove our lower bounds.

3 No Embedding Leads to Small Loss

As discussed before, each of the n possible linear targets in the Hadamard Learning Problem corresponds to a standard basis vector $\mathbf{e}_i$. In our first theorem, we show that any linear combination of the instances (rows of the Hadamard matrix) in which not all instances are used is far away from all targets. So any algorithm that predicts with a linear combination of the instances cannot express any of the targets unless all examples have been observed.

Theorem 2. *Any linear combination of $k < n$ instances/rows of the n-dimensional Hadamard matrix has distance at least $\sqrt{1-\frac{k}{n}}$ from any of the n-dimensional standard basis vectors.*

We now show that linear combinations of $k < n$ rows are not just far away from any standard basis vector, but they also have large average square loss w.r.t. any such target. In the theorem below we give lower bounds for the noise-free case, i.e., when the labels are consistent with a target (which is one of the components of the instances).

Theorem 3. *For any linear combination of k rows of the n-dimensional Hadamard matrix and any n dimensional standard basis vector, the average square loss over all n examples is at least $1-\frac{k}{n}$.*

Next we show that a similar lower bound can be proven even if the instances can be embedded into any higher dimensional space (for instance by using a kernel function). So this lower bound applies to all algorithms that predict with a linear combination of the expanded instances. Our proof exploits the flat SVD spectrum of Hadamard matrices.

Our theorem applies to any learning algorithm that follows the following protocol: It first chooses an embedding of all n instances. It then receives a set of k embedded training instances labeled by one of the targets (i.e., we are in the noise-free case)[6]. The algorithm then produces a linear combination of the embedded training instances as its linear weight vector in feature space and is then evaluated w.r.t. the same target on all n instances.

Theorem 4. *For any embedding of the rows of the n-dimensional Hadamard matrix $\mathbf{H}$, any subset of k rows and any n linear combinations $\mathbf{w}_i$ of the embedded k rows (one per target), the following holds: If ℓ_i is the average square loss of $\mathbf{w}_i$ on the i-th target (where the average is taken over* **all** *n examples), then $\frac{\sum_i \ell_i}{n} \geq 1-\frac{k}{n}$.*

The generality of the theorem might be confusing: The theorem holds for any weight vectors that are linear combinations of the k embedded *training instances*, where the coefficients of the linear combination can depend arbitrarily on the target and the training instances. In particular, the lower bound holds for the following learning model: The k training instances are drawn at random w.r.t. any distribution. If the k training instances are drawn with replacement, then the learner might end up with $k' < k$ unique instances and the lower bound

[6] Our results hold even if the learner embeds the instances *after* seeing the k training instances. The only restriction is that the embedding must be the same for all targets.

for those draws is then $1-\frac{k'}{n}$ instead of $1-\frac{k}{n}$. More discussion of probabilistic models is given in Section 5.

Note that this lower bound is weaker than the previous one in that we now average over instances *and* targets. However, since we have a lower bound on the average over targets, that same lower bound must hold for at least one of the targets. The average loss is measured w.r.t. the uniform distribution *on all* n instances. This is crucial because it disallows the case in which the algorithm predicts badly on the k training instances and well on the $n-k$ remaining test instances. Averaging over targets is also necessary because as we shall see later, for some embeddings there are linear combinations of k expanded instances that predict perfectly on k of the n targets.

Proof. Let $\phi : \mathbb{R}^n \to \mathbb{R}^m$ denote an arbitrary function that is used to map the rows of the $n \times n$ Hadamard matrix $\mathbf{H}$ to a matrix $\mathbf{Z} \in \mathbb{R}^{n\times m}$. We use $\widehat{\mathbf{H}}$ and $\widehat{\mathbf{Z}}$ to denote the sub-matrices of $\mathbf{H}$ and $\mathbf{Z}$ which contain the k rows corresponding to the training instances. The weight vector must be a linear combination of the embedded instances, i.e., the k rows of $\widehat{\mathbf{Z}}$. Let $\mathbf{w}_i = \widehat{\mathbf{Z}}^\top \mathbf{a}_i$ denote the weight vector when the instances are labeled with the i-th column of $\mathbf{H}$. We use matrix notation to combine all n learning problems into one. Let $\mathbf{A} \in \mathbb{R}^{k\times n}$ be the matrix whose columns are the coefficient vectors $\mathbf{a}_i$. All n weight vectors form the matrix $[\mathbf{w}_1, \ldots, \mathbf{w}_n] = \widehat{\mathbf{Z}}^\top \mathbf{A}$. Observe that the prediction of the algorithm on the n rows of the Hadamard matrix is given by $\mathbf{Z}\,\widehat{\mathbf{Z}}^\top \mathbf{A}$, while the target predictions are $\mathbf{H}\,\mathbf{I} = \mathbf{H}$. For the square loss we can write the total loss of the n linear classifiers as $||\,\mathbf{Z}\,\widehat{\mathbf{Z}}^\top \mathbf{A} - \mathbf{H}\,||_F^2$. The Hadamard matrix $\mathbf{H}$ has rank n while $\widehat{\mathbf{Z}}$ has rank at most k and hence $\mathbf{Z}\,\widehat{\mathbf{Z}}^\top \mathbf{A}$ has rank at most k. From Theorem 1 it is clear that the loss is minimized when $\mathbf{Z}\,\widehat{\mathbf{Z}}^\top \mathbf{A} = \mathbf{U}\,\widehat{\mathbf{S}}\,\mathbf{V}^\top$ where $\mathbf{H} = \mathbf{U}\,\mathbf{S}\,\mathbf{V}^\top$ is the SVD of $\mathbf{H}$ and $\hat{s_i} = s_i = \sqrt{n}$ for $i = 1, \ldots, k$ while $\hat{s}_{k+1} = \ldots = \hat{s}_n = 0$. The squared Frobenius norm of the residual or the total loss incurred by the algorithm is therefore $(n-k)s_n^2 = n(n-k)$. By uniformly averaging the loss over the n targets and n instances, the expected value of the loss is $1-\frac{k}{n}$. ■

If the hypotheses are allowed to be a bias plus a linear combination of the k chosen training instances, then the total loss of the n classifiers becomes $||\,\mathbf{Z}\,\widehat{\mathbf{Z}}^\top \mathbf{A} + \mathbf{B} - \mathbf{H}\,||_F^2$, where the bias matrix $\mathbf{B}$ is any $n \times n$ matrix with identical entries in each column. Since $\mathbf{B}$ has rank one, $\mathbf{Z}\,\widehat{\mathbf{Z}}^\top \mathbf{A} + \mathbf{B}$ has rank at most $k+1$. Thus the lower bound in the above theorem changes to $1-\frac{k+1}{n}$ instead of $1-\frac{k}{n}$. So the lower bounds discussed in this section are not majorly affected by allowing a bias, and for the sake of simplicity we only state our results for the homogeneous case.

There is a trivial kernel that shows that the above theorem can be tight: We let the ith row of $\mathbf{H}$ map to the n-dimensional standard basis vector $\mathbf{e}_i$ (i.e., $\mathbf{Z} = \mathbf{I}$). After seeing a subset of k training instances (labeled by one of the targets), we build a hypothesis weight vector $\mathbf{w}$ as follows: w_i is set to y_i if the ith row was a training instance labeled with y_i and zero otherwise. This weight vector $\mathbf{w} \in \mathbb{R}^n$ is a linear combination of the training instances (which are all standard basis vectors $\mathbf{e}_j$ s.t. row j of $\mathbf{Z}$ is one of the k training instances). The predictions of $\mathbf{w}$ are as follows: They agree with the labels of the target on the k training instances and are zero on the remaining $n-k$ instances. We call any weight vector with the above predictions a *memorizing* weight vector. Whenever the target labels are ±1, any memorizing weight vector has average loss $1-\frac{k}{n}$ on the n instances. So for any of the n standard basis vectors the average loss on the n instances is exactly $1-\frac{k}{n}$.

Note that the Hadamard matrix itself is a rotated and scaled unit matrix. So the embedding $\mathbf{Z} = \mathbf{H}$ (i.e., the identity embedding) could also be used to realize memorizing weight vectors for each target. In other words an optimal embedding for the Hadamard problem is the identity embedding (used in theorems 2 and 3). Theorem 4 shows that no kernel can lead to an improvement (when averaged over targets).

The following embedding shows that averaging over targets is necessary in the above theorem and the lower bound does not necessarily hold for the average loss w.r.t. *each* target (as was the case for the identity kernel (Theorem 3)). In this embedding $\mathbf{Z}$ consists of the first k columns of $\mathbf{H}$ (i.e., the dimension of the instances is shrunk from n down to k). Furthermore let $\widehat{\mathbf{Z}}$ be the first k rows of $\mathbf{Z}$ (in other words $\widehat{\mathbf{Z}} = \mathbf{H}(1:k, 1:k)$ and is nonsingular). We first define a weight vector $\mathbf{w}_i$ for each target $\mathbf{e}_i$ and then show that these weight vectors are realizable as linear combinations of the k training instances: Let $\mathbf{w}_i$ be zero if $i > k$ and $\mathbf{w}_i = \mathbf{e}_i$ otherwise. To realize these weight vectors as linear combinations of the k training instances set the coefficient vector $\mathbf{a}_i$ to zero if $i > k$ and to the ith column of $(\widehat{\mathbf{Z}}^T)^{-1}$ otherwise. Now the prediction vector $\mathbf{Z}\,\mathbf{w}_i$ is the ith column of $\mathbf{H}$, if $i \leq k$, and zero otherwise. In other words, using this embedding, k of the targets can be predicted perfectly (average loss 0), and the remaining $n-k$ targets have average loss 1. When we average over all n instances *and* targets, then this average is still $1-\frac{k}{n}$ (So again the above theorem is tight). However now the average loss for each target is not lower bounded by $1-\frac{k}{n}$, and therefore Theorem 4 cannot be strengthened as discussed above.

Note that the only fact about the Hadamard matrix that enters into the previous theorem is that its SVD spectrum is flat and it is straightforward to generalize the above theorems to arbitrary matrices. For the sake of simplicity we go back to averaging over all instances.

Corollary 1. *As in Theorem 4, but now* $\mathbf{H}$ *is any* $n \times n$ *dimensional matrix with SVD spectrum* s_i*. Then the lower bound on the expected square loss over all* n *instances changes to* $\frac{1}{n^2}\sum_{i=k+1}^{n} s_i^2$*. When all singular values are equal to* s*, then the lower bound becomes* $(1-\frac{k}{n})\frac{s^2}{n}$*.*

4 Random Matrices Are Hard

Hadamard matrices seem rather special. However we will show that if the Hadamard matrix is replaced by a random ±1 matrix, then (with high probability) the lower bound of Theorem 4 holds with slightly smaller constants. The reason for this is that the SVD spectrum of random ± 1 matrices has a heavy tail and hence when learning the columns of a random matrix, the expected loss is large for at least one column (as in Corollary 1). In Figure 1 we plot the spectrum s_i (as a function of i) of the Hadamard matrix which is a flat line at level $\sqrt{n}$, where $n = 1024$. We also plot the spectra of 500 random 1024×1024 matrices with ± 1 entries. Each such spectrum s_i is a line that is closely approximated by the linear curve $2\sqrt{n} - \frac{2i}{\sqrt{n}}$. Notice the heavy tail and low variance of the spectra: In the plot, the 500 lines become one thick line.

Recall that after seeing half of the examples, the expected square loss for the Hadamard Learning Problem is at least $\frac{1}{2}$ for at least one of the target columns (Theorem 4). When the matrix is chosen at random, then Corollary 1 implies that this loss is still about $\frac{1}{4}$.

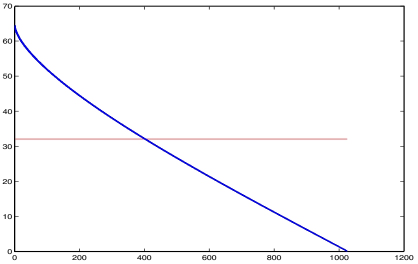

Fig. 1. The horizontal line represents the spectrum s_i (as a function of i) of the 1024 dimensional Hadamard matrix (at level $\sqrt{1024}$). The plot also contains the spectra of 500 random 1024×1024 matrices with $\{\pm 1\}$ entries. The variance of these spectra lines is small and therefore the 500 lines form one thick line

Before we detail what is provable for random matrices we would like to discuss some related work. In Ben-David et al. (2002) it was shown that most concept classes of VC dimension d can be embedded only with a trivial 2-2 margin. This is seen as evidence that most concept classes of VC dimension d "may" be hard to learn by kernel based algorithms. (Note that random concept classes of VC dimension d might not even be *efficiently* learnable by *any* algorithm.) Furthermore, we are not aware of any formal lower bound in terms of the 2-2 margin.

In contrast, we completely bypass the notion of the margin and give a stronger result. We define a class of *easy to learn linear problems* characterized by a random $n \times n$ matrix with ± 1 entries[7]. The instances are the rows of this matrix and the target concepts are the columns of this matrix.

In the full paper we show analytically that, with high probability, any algorithm that predicts with a linear combination of the instances cannot learn random problems of this type. By Corollary 1 it suffices to investigate the properties of the random variable $Q = \frac{1}{n^2}\sum_{i=k+1}^{n} s_i^2$. Using techniques from Davidson and Szarek (2003), Meckes (2004), we show that Q is sharply concentrated around $1 - c \cdot \frac{k}{n}$ where c is a scalar constant.

[7] The VC dimension of our learning problem is at most $\lg n$.

5 Probabilistic Models

In this section we prove lower bounds for the case when the training examples are chosen based on some probabilistic model.

Theorem 5. *Assume we have a uniform distribution on the n rows of the Hadamard matrix. Assume the algorithm first embeds*[8] *the n rows and then draws k random training examples without replacement that are all labeled by one of the n targets. It then forms its hypothesis by choosing a linear combination of the embedded k training instances.*

If ℓ_i is the expected average square loss when the target is the i-th target and the loss is averaged over **all** *n examples, then $\frac{\sum_i \ell_i}{n} \geq 1 - \frac{k}{n}$.*

Proof. Follows from Theorem 4. ■

The lower bound on $\frac{\sum_i \ell_i}{n}$ is tight for the identity kernel and the memorizing weight vector.

Note that we always average the loss over all n instances. We believe that this is the only reasonable model for the case when the instance domain is finite. In the full paper we also develop lower bounds for the case when the loss is averaged over the $n - k$ test instances. There are no surprises but the bounds are slightly more complicated to state.

We now prove a similar theorem for the case when the training examples are drawn with replacement.

Theorem 6. *Assume we have a uniform distribution on the n rows of the Hadamard matrix. Assume the algorithm first embeds the n rows and then draws t training examples independently at random with replacement that are labeled by one of the n targets. It then forms its hypothesis by choosing the linear combination of the t embedded training instances.*

If ℓ_i is the expected average square loss when the target is the i-th target and the loss is averaged over **all** *n examples, then $\frac{\sum_i \ell_i}{n} \geq \left(1 - \frac{1}{n}\right)^t$.*

Proof. By Theorem 4 the average square loss (over all instances and targets) conditioned on the fact that k distinct training examples were drawn is at least $1 - \frac{k}{n} = \frac{n-k}{n}$. Note that $n - k$ is the number of examples *missed* in the training set. Let M be a random variable denoting the number of missed examples. By the above argument the lower bound is $\frac{E(M)}{n}$.

Clearly, $M = \sum_i M_i$, where M_i is a binary random variable indicating whether the ith example was missed, and

$$E(M) = E(\sum_i M_i) = \sum_i E(M_i) = nE(M_1) = n(1 - \frac{1}{n})^t. \quad ■$$

[8] Theorem 4 guarantees that the lower bound also holds for the following protocol: The algorithm first draws k rows without replacement. It then chooses its embedding (that may depend on the chosen rows). Finally the chosen rows are labeled by one of the targets and a linear combination of the embedded k training instances is chosen as the hypothesis.

6 Rotation Invariance

Kernel algorithms are commonly motivated by weight updates derived from a "Representer Theorem" (Kimeldorf and Wahba, 1971, Schölkopf et al., 2001). This type of theorem states that if the weight vector is produced by a certain minimization problem, then it must be a linear combination of the expanded instances. In the simplest form

$$\mathbf{w} = \operatorname*{arginf}_{\mathbf{w}'} \left(\Omega(\|\, \mathbf{w}' \,\|^2) + L(\mathbf{w}') \right), \tag{2}$$

where Ω is a monotonic non-decreasing function and L is a convex real-valued loss function that only depends on the dot products between the weight vector $\mathbf{w}'$ (in feature space) and the expanded instances $\phi(\mathbf{x}_i)$.

Here, we follow Kivinen and Warmuth (1997) and first point out that there is a simple *geometric* property of an algorithm that guarantees that the algorithm produces a linear combination of the expanded instances. This property is the notion of rotation invariance of an algorithm. Representer theorems are a special case of this characterization because the objective functions (eg. (2)) used in these theorems are rotation invariant.

Representer theorems (and more generally rotation invariance) guarantee that the weight vector of an algorithm is a linear combination of the expanded instances. However, the lower bounds of our paper hold for *any* algorithm whose hypotheses are such linear combinations. This *includes* (see example at the end of section) algorithms that are not rotation invariant and algorithms that break the kernel paradigm.

We denote the examples as $(\mathbf{x}, y)$ and assume that instances $\mathbf{x}$ already lie in some expanded feature space $\mathcal{X}$ and the labels y in some label domain $\mathcal{Y} \subseteq \mathbb{R}$. For the sake of simplicity the instance domain $\mathcal{X} = \mathbb{R}^n$ for some n. An algorithm maps arbitrary sequences of examples $\langle\, \mathcal{S} \,\rangle = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_T, y_T)\}$ to a weight vector $\mathbf{w}(\langle\, \mathcal{S} \,\rangle)$ in the instance domain. We study the behavior of algorithms when they receive rotated sequences. If $\mathbf{U}$ is an orthonormal matrix in $\mathbb{R}^{n\times n}$, then $\langle\, \mathbf{U}\, \mathcal{S} \,\rangle$ denotes the sequence $\{(\mathbf{U}\, \mathbf{x}_1, y_1), (\mathbf{U}\, \mathbf{x}_2, y_2), \ldots, (\mathbf{U}\, \mathbf{x}_T, y_T)\}$. (Note that the rotation only affects the instances.)

Theorem 7. *Let* $\langle\, \mathcal{S} \,\rangle = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_T, y_T)\} \subseteq \mathbb{R}^n \times \mathbb{R}$ *be any sequence of examples and let* $\mathbf{w}$ *be the input-output mapping of an algorithm from sequences of examples in* $\mathbb{R}^n \times \mathbb{R}$ *to* vectors *in* $\mathbb{R}^n$. *Consider the following three statements:*

1. $\mathbf{w}$ *is* rotation invariant *in the sense that*

$$\mathbf{w}(\langle\, \mathcal{S} \,\rangle)^\top \mathbf{x} = \mathbf{w}(\langle\, \mathbf{U}\, \mathcal{S} \,\rangle)^\top \mathbf{U}\, \mathbf{x},$$

for all sequences $\langle\, \mathcal{S} \,\rangle$, *orthonormal matrices* $\mathbf{U} \in \mathbb{R}^{n\times n}$, *and* $\mathbf{x} \in \mathbb{R}^n$.

2. *For all* $\langle\, \mathcal{S} \,\rangle, \mathbf{w}(\langle\, \mathcal{S} \,\rangle)$ *must be a linear combination of the instances of* $\langle\, \mathcal{S} \,\rangle$.
3. *For all* $\langle\, \mathcal{S} \,\rangle$ *and rotation matrices* $\mathbf{U}$, $\mathbf{w}(\langle\, \mathbf{U}\, \mathcal{S} \,\rangle) = \mathbf{U}\, \mathbf{w}(\langle\, \mathcal{S} \,\rangle)$.

Now, $1 \implies 2 \wedge 3$ *and* $3 \implies 1$.

The following example shows that the implications $2 \Longrightarrow 3$ and $2 \Longrightarrow 1$ do not hold in general:

$$\mathbf{w}(\langle\, \mathcal{S}\,\rangle) = \begin{cases} \mathbf{0} & \text{if } x_{1,1} > 0 \\ \mathbf{x}_1 & \text{otherwise.} \end{cases}$$

Clearly, $\mathbf{w}(\langle\, \mathcal{S}\,\rangle)$ is a linear combination of the instances in $\langle\, \mathcal{S}\,\rangle$ and therefore Statement 2 holds for this algorithm. Choose $\mathbf{U}$ as $-\mathbf{I}$, i.e., minus the identity matrix and the first instance in $\langle\, \mathcal{S}\,\rangle$ as $\mathbf{x}_1 = (1, 0, \ldots, 0)^\top$. Now, $\mathbf{w}(\langle\, \mathcal{S}\,\rangle) = \mathbf{0}$ and $\mathbf{w}(\langle\, \mathbf{U}\,\mathcal{S}\,\rangle) = \mathbf{x}_1$ and therefore statements 1 and 3 are both false. Note that in this example the individual components of the instances are accessed and thus the kernel paradigm is violated. However, as long as the hypotheses produced are linear combinations, our lower bounds apply.

Even though we only defined rotation invariance in $\mathbb{R}^n$, it should be apparent that this notion can easily be generalized to arbitrary RKHS.

7 Leaving the Span with Constraints

Consider a set of k instances (rows) of the following form (Figure 2): All components are ± 1 and all 2^k bit patterns exactly appear once as a column. Assume the instances are labeled by one of the $n = 2^k$ columns (i.e., the target is one of the n standard basis vectors). Consider two algorithms: The first is any algorithm that predicts with a linear combination of the instances and the second any algorithm that produces a weight vector consistent with the examples *and* the *additional constraints* that the weights are non-negative and sum to one. For each of the algorithms, form a matrix whose columns are the n weight vectors produced by the respective algorithm as the target is varied. Clearly, the rank of the first algorithm's weight matrix is at most k (even if the instances are allowed to be embedded). However, we will show now that the rank of the second algorithms weight matrix is at least 2^k.

$$\begin{array}{cccccccc} -1 & +1 & -1 & +1 & -1 & +1 & -1 & +1 \\ -1 & -1 & +1 & +1 & -1 & -1 & +1 & +1 \\ -1 & -1 & -1 & -1 & +1 & +1 & +1 & +1 \end{array}$$

Fig. 2. Each bit pattern appears once as a column of the $\lg n$ instances ($n = 8$).

Lemma 1. *Assume the examples are the rows of a $k \times n$ dimensional matrix with entries ± 1 which are labeled by one of the columns of the matrix. Then any weight vector $\mathbf{w}$ that is consistent with the examples* and *satisfies the above additional constraints has the following property: If $w_i > 0$ then the i-th column coincides with the labels. If all columns are unique, then the $\mathbf{w}$ is always the standard basis vector that identifies the target column.*

Proof. W.l.o.g. the labels are all ones. Otherwise, multiply the corresponding instance and label by -1 and keep the weights unchanged. Now, because of the additional constraints on the weights, it is easy to see that all non-zero weights must be on columns that contain only ones. ■

This means that the weight matrix of the second algorithm is the n-dimensional unit matrix (rank $n = 2^k$). So adding the constraint forced the rank of the weight matrix to grow exponentially with the number of examples instead of linearly.

See Figure 3 for a simple problem where imposing these additional constraints makes the weight of the consistent column grow exponentially. Maintaining constraints can be expensive. In particular, the non-negativity constraints access the individual features and this breaks the kernel paradigm.

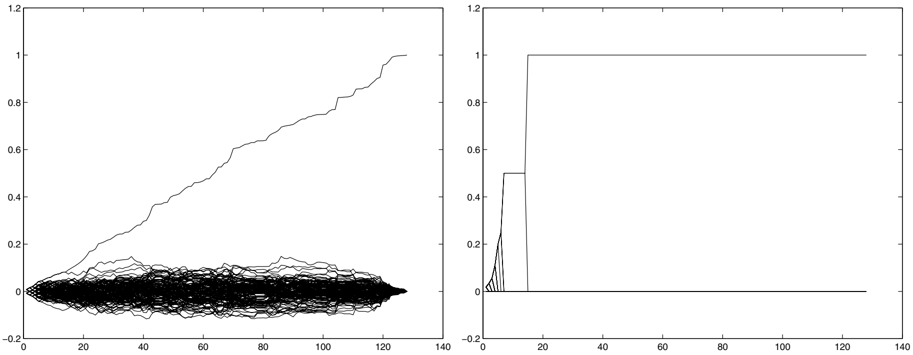

Fig. 3. The examples are the rows of a random 128 × 128 dimensional matrix with ±1 entries labeled by column one. Let $\mathbf{w}^t$ be the shortest weight vector consistent with the first t examples. For $1 \leq i \leq 128$, we plot (left) the w_i^t as a function of t ($x-$axis). The line representing the first weight w_1^t grows roughly linearly and only after seeing all 128 examples the target weight vector $\mathbf{e}_1$ is found. On the right we have the same plot, but we enforce the additional constraints that $w_i^t \geq 0$ and $\sum_{i=1}^n w_i^t = 1$. Now the first weight grows much faster and the target weight vector is found much sooner

One simple algorithm that maintains the additional constraints is the Exponentiated Gradient (EG) Algorithm whose weight vector has the following *exponential form*:

$$w_i = \frac{w_i^1 \ \exp\left(\sum_{t=1}^k a_t x_i^t\right)}{Z}, \text{ where } Z = \sum_{j=1}^n w_j^1 \ \exp\left(\sum_{t=1}^k a_t x_j^t\right). \tag{3}$$

Here $\mathbf{w}^1$ is an initial weight vector satisfying the additional constraints. Note that $\ln w_i = \sum_{t=1}^k a_t x_i^t + \ln w_i^1 - \ln Z$. So except for the additional terms introduced by the initial weights and the normalization, the logarithms of the weights are a linear combination of examples [9].

If the EG algorithm is used as our second algorithm then it realizes the n standard basis vectors as follows: Set the coefficients a_t of the k examples to

[9] Set the nth weight to $w_n = 1 - \sum_{j=1}^{n-1} w_j$. If the initial weights are uniform then $\ln \frac{w_j}{1-\sum_{j'=1}^{n-1} w_{j'}} = \sum_{t=1}^k (a_t x_j^t - a_n x_n^t)$.

$\pm\eta$ and let the learning rate η go to infinity. Each such weight vector converges to a standard basis vector, one for each of the $2^k = n$ sign patterns of the coefficients. The cost of the EG algorithm is $O(1)$ per example and feature (Kivinen and Warmuth, 1997) and again the kernel paradigm is broken because individual features are accessed.

In general, weight updates can be defined in terms of a link function[10] $\mathbf{f}$ or its associated Bregman divergence (Azoury and Warmuth, 2001). The updated weights minimize a Bregman divergence plus η times the loss on the last instance and the Bregman divergence serves as a barrier function for maintaining the constraints (e.g., Kivinen and Warmuth, 2001, Helmbold et al., 1999).

Bounds: We only sketch the bounds provable for a slightly more general version of the EG algorithm called EG$^{\pm}$ (Kivinen and Warmuth, 1997). This algorithm maintains the constraint $||\,\mathbf{w}\,||_1 \leq U_1$. Assume the instances have infinity norm at most X_∞ and there is a consistent weight vector $\mathbf{u}$ s.t. $||\,\mathbf{u}\,||_1 \leq U_1$. (Both U_1 and X_∞ are parameters to the algorithm). One can show that after receiving k training examples drawn from any fixed distribution, the expected loss[11] of this algorithm (w.r.t. the same distributions) is $\frac{X_\infty^2 U_1^2 \ln n}{k}$ (e.g., Kivinen and Warmuth, 1997). If the learning rate is properly tuned, then these algorithms also have good bounds in the noisy case.

It is important to note that even though the bounds provable for these algorithms are messy to state, the essentials are quite simple. The weight vectors are defined using a relative entropy regularization term (in general any Bregman divergence) instead of the squared Euclidean distance used for the kernel based algorithms (which predict with a linear combination of the instances).

8 Conclusion

In Kivinen and Warmuth (1997) and Gentile and Littlestone (1999) a pair of dual norms was used to characterize the generalization performance of different families of learning algorithms. For each algorithm there are certain settings in which its bound beats the bounds of the other algorithm. In this paper we showed how the lower bounds for one important family (the one predicting with a linear combination of the instances) still hold even if the instances can be embedded into any Euclidean space.

Kernel methods are often described as "non-linear" methods because they allow the use of non-linear features. However, no matter what embedding is used, kernel methods build linear models in feature space and for some problems this is highly restrictive.

[10] Now $\mathbf{f}(\langle\,\mathcal{S}\,\rangle)$ is a linear combination of the (embedded) instances and $\mathbf{f}(\mathbf{w}(\langle\,\mathcal{S}\,\rangle))^\top\,\mathbf{x} = \mathbf{f}(\mathbf{w}(\langle\,\mathbf{U}\,\mathcal{S}\,\rangle))^\top\,\mathbf{U}\,\mathbf{x}$ for any orthonormal matrix $\mathbf{U}$.

[11] The bound only holds for the average weight vector: Do one pass over all k examples; starting from the initial weight vector, update the weights after each example is processed and average the resulting $k+1$ weight vectors (e.g., Section 8 Kivinen and Warmuth, 1997).

Acknowledgments. We thank Claudio Gentile, Adam Kowalczyk, Alan Pajor and Stéphane Canu for insightful discussions.

References

K. Azoury and M. K. Warmuth. Relative loss bounds for on-line density estimation with the exponential family of distributions. *Machine Learning*, 43(3):211 – 246, 2001. Special issue on Theoretical Advances in On-line Learning, Game Theory and Boosting.

S. Ben-David, N. Eiron, and H. U. Simon. Limitations of learning via embeddings in Euclidean half-spaces. *Journal of Machine Learning Research*, 3:441 – 461, Nov. 2002.

Nello Cristianini, Colin Campbell, and John Shawe-Taylor. Multiplicative updatings for support vector learning. In *Proc. of European Symposium on Artificial Neural Networks*, pages 189 – 194, 1999.

K. R. Davidson and S. J. Szarek. Banach space theory and local operator theory. In J. Lindenstrauss and W. Johnson, editors, *Handbook of the Geometry of Banach Spaces*. North-Holland, 2003.

J. Forster, N. Schmitt, and H. U. Simon. Estimating the optimal margins of embeddings in Euclidean half spaces. In *Proc. of the 14th Annual Conference on Computational Learning Theory*, pages 402 – 415. Springer, 2001.

C. Gentile and N. Littlestone. The robustness of the *p*-norm algorithms. In *Proc. 12th Annu. Conf. on Comput. Learning Theory*, pages 1–11. ACM Press, New York, 1999.

Claudio Gentile and M. K. Warmuth. Linear hinge loss and average margin. In M. S. Kearns, S. A. Solla, and D. A. Cohn, editors, *Advances in Neural Information Processing Systems 11*, pages 225 – 231, Cambridge, MA, 1999. MIT Press.

D. P. Helmbold, J. Kivinen, and M. K. Warmuth. Relative loss bounds for single neurons. *IEEE Transactions on Neural Networks*, 10(6):1291 – 1304, Nov. 1999.

R. Herbrich, T. Graepel, and R. C. Williamson. *Innovations in Machine Learning*, chapter The Structure of Version Space. Springer, January 2005. D. Holmes and L. C. Jain Editors.

R. A. Horn and C. R. Johnson. *Matrix Analysis.* Cambridge University Press, Cambridge, 1985.

R. Khardon, D. Roth, and R. Servedio. Efficiency versus convergence of Boolean kernels for on-line learning algorithms. In *Advances in Neural Information Processing Systems 14*, pages 423–430, 2001.

G. S. Kimeldorf and G. Wahba. Some results on Tchebycheffian spline functions. *J. Math. Anal. Applic.*, 33:82 – 95, 1971.

J. Kivinen and M. K. Warmuth. Relative loss bounds for multidimensional regression problems. *Machine Learning*, 45(3):301 – 329, 2001.

J. Kivinen and M. K. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132(1):1 – 64, January 1997.

J. Kivinen, M. K. Warmuth, and P. Auer. The perceptron learning algorithm vs. Winnow: Linear vs. logarithmic mistake bounds when few input variables are relevant. *Artificial Intelligence*, 97(1 - 2):325 – 343, 1997.

Nick Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2:285 – 318, 1988.

M. W. Meckes. Concentration of norms and eigenvalues of random matrices. *Journal of Functional Analysis*, 211(2):508–524, June 2004.

L. Pitt and M. K. Warmuth. The minimum consistent DFA problem cannot be approximated within any polynomial. *Journal of the ACM*, 40(1):95 – 142, 1993.

R. Schapire, Y. Freund, P. L. Bartlett, and W. S. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *Annals of Statistics*, 26:1651 – 1686, 1998.

B. Schölkopf, R. Herbrich, and A. J. Smola. A generalized representer theorem. In *Proc. of the Annual Conference on Computational Learning Theory*, pages 416 – 426, 2001.

E. Takimoto and M. K. Warmuth. Path kernels and multiplicative updates. *Journal of Machine Learning Research*, 4:773 – 818, October 2003.

M. K. Warmuth. Towards representation independence in PAC-learning. In J. P. Jantke, editor, *Proc. of AII-89 Workshop on Analogical and Inductive Inference*, volume 397 of *Lecture Notes in Artificial Intelligence 397*, pages 78 – 103. Springer-Verlag, October 1989.