Efficiency versus Convergence of Boolean Kernels for On-Line Learning Algorithms

Roni Khardon
Tufts University
Medford, MA 02155
roni@eecs.tufts.edu

Dan Roth
University of Illinois
Urbana, IL 61801
danr@cs.uiuc.edu

Rocco Servedio
Harvard University
Cambridge, MA 02138
rocco@deas.harvard.edu

Abstract

We study online learning in Boolean domains using kernels which capture feature expansions equivalent to using conjunctions over basic features. We demonstrate a tradeoff between the computational efficiency with which these kernels can be computed and the generalization ability of the resulting classifier. We first describe several kernel functions which capture either limited forms of conjunctions or all conjunctions. We show that these kernels can be used to efficiently run the Perceptron algorithm over an exponential number of conjunctions; however we also prove that using such kernels the Perceptron algorithm can make an exponential number of mistakes even when learning simple functions. We also consider an analogous use of kernel functions to run the multiplicative-update Winnow algorithm over an expanded feature space of exponentially many conjunctions. While known upper bounds imply that Winnow can learn DNF formulae with a polynomial mistake bound in this setting, we prove that it is computationally hard to simulate Winnow's behavior for learning DNF over such a feature set, and thus that such kernel functions for Winnow are not efficiently computable.

1 Introduction

The Perceptron and Winnow algorithms are well known learning algorithms that make predictions using a linear function in their feature space. Despite their limited expressiveness, they have been applied successfully in recent years to several large scale real world classification problems. The SNoW system [7, 2], for example, has successfully applied variations of Perceptron [6] and Winnow [4] to problems in natural language processing. The system first extracts Boolean features from examples (given as text) and then runs learning algorithms over restricted conjunctions of these basic features.

There are several ways to enhance the set of features after the initial extraction. One idea is to expand the set of basic features $x_1, \ldots, x_n$ using conjunctions such as $x_1\overline{x}_3x_4$ and use these expanded higher-dimensional examples, in which each conjunction plays the role of a basic feature, for learning. This approach clearly leads to an increase in expressiveness and thus may improve performance. However, it also dramatically increases the number of features (from n to 3^n if all conjunctions are used) and thus may adversely affect both the computation time and convergence rate of learning.

This paper studies the computational efficiency and convergence of the Perceptron and Winnow algorithms over such expanded feature spaces of conjunctions. Specifically, we study the use of kernel functions to expand the feature space and thus enhance the learning abilities of Perceptron and Winnow; we refer to these enhanced algorithms as *kernel Perceptron* and *kernel Winnow*.

1.1 Background: Perceptron and Winnow

Throughout its execution Perceptron maintains a weight vector $w \in \Re^N$ which is initially $(0, \ldots, 0)$. Upon receiving an example $x \in \Re^N$ the algorithm predicts according to the linear threshold function $w \cdot x \geq 0$. If the prediction is 1 and the label is -1 (false positive prediction) then the vector w is set to $w - x$, while if the prediction is -1 and the label is 1 (false negative) then w is set to $w + x$. No change is made if the prediction is correct.

The famous Perceptron Convergence Theorem [6] bounds the number of mistakes which the Perceptron algorithm can make:

Theorem 1 *Let $\langle x^1, y_1 \rangle, \ldots, \langle x^t, y_t \rangle$ be a sequence of labeled examples with $x^i \in \Re^N$, $\|x^i\| \leq R$ and $y_i \in \{-1, 1\}$ for all i. Let $u \in \Re^N, \xi > 0$ be such that $y_i u \cdot x^i \geq \xi$ for all i. Then Perceptron makes at most $\frac{R^2 \|u\|^2}{\xi^2}$ mistakes on this example sequence.*

The Winnow algorithm [4] has a very similar structure. Winnow maintains a hypothesis vector $w \in \Re^N$ which is initially $w = (1, \ldots, 1)$. Winnow is parameterized by a promotion factor $\alpha \geq 1$ and a threshold $\theta > 0$; upon receiving an example $x \in \{0, 1\}^N$ Winnow predicts according to the threshold function $w \cdot x \geq \theta$. If the prediction is 1 and the label is -1 then for all i such that $x_i = 1$ the value of w_i is set to w_i/α; this is a *demotion* step. If the prediction is -1 and the label is 1 then for all i such that $x_i = 1$ the value of w_i is set to αw_i; this is a *promotion* step. No change is made if the prediction is correct.

For our purposes the following mistake bound, implicit in [4], is of interest:

Theorem 2 *Let the target function be a k-literal monotone disjunction $f(x_1, \ldots, x_N) = x_{i_1} \vee \cdots \vee x_{i_k}$. For any sequence of examples in $\{0, 1\}^N$ labeled according to f the number of prediction mistakes made by Winnow(α, θ) is at most $\frac{\alpha}{\alpha - 1} \cdot \frac{N}{\theta} + k(\alpha + 1)(1 + \log_\alpha \theta)$.*

1.2 Our Results

Our first result in Section 2 shows that it is possible to efficiently run the kernel Perceptron algorithm over an exponential number of conjunctive features:

Theorem 3 *There is an algorithm that simulates Perceptron over the 3^n-dimensional feature space of all conjunctions of n basic features. Given a sequence of t labeled examples in $\{0, 1\}^n$ the prediction and update for each example take poly(n, t) time steps.*

This result is closely related to one of the main open problems in learning theory: efficient learnability of disjunctions of conjunctions, or DNF (Disjunctive Normal Form) expressions.[1] Since linear threshold elements can represent disjunctions (e.g. $x_1 \vee x_2 \vee x_3$ is true iff $x_1 + x_2 + x_3 \geq 1$), Theorems 1 and 3 imply that kernel Perceptron can be used to learn DNF. However, in this framework the values of N and R in Theorem 1 can be exponentially large, and hence the mistake bound given by Theorem 1 is exponential rather than polynomial in n. The question thus arises whether, for kernel Perceptron, the exponential

[1] Angluin [1] proved that DNF expressions cannot be learned efficiently using hypotheses which are themselves DNF expressions from equivalence queries and thus also in the mistake bound model which we are considering here. However this result does not preclude the efficient learnability of DNF using a different class of hypotheses such as those generated by the kernel Perceptron algorithm.

upper bound implied by Theorem 1 is essentially tight. We give an affirmative answer, thus showing that kernel Perceptron cannot efficiently learn DNF:

Theorem 4 *There is a monotone DNF f over $x_1, \ldots, x_n$ and a sequence of examples labeled according to f which causes the kernel Perceptron algorithm to make $2^{\Omega(n)}$ mistakes.*

Turning to Winnow, an attractive feature of Theorem 2 is that for suitable α, θ the bound is logarithmic in the total number of features N (e.g. $\alpha = 2$ and $\theta = N$). Therefore, as noted by several researchers [5], if a Winnow analogue of Theorem 3 could be obtained this would imply efficient learnability of DNF. We show that no such analogue can exist:

Theorem 5 *There is no polynomial time algorithm which simulates Winnow over exponentially many monotone conjunctive features for learning monotone DNF, unless every problem in #P can be solved in polynomial time.*

We observe that, in contrast to Theorem 5, Maass and Warmuth have shown that the Winnow algorithm can be simulated efficiently over exponentially many conjunctive features for learning some simple geometric concept classes [5].

While several of our results are negative, in practice one can achieve good performance by using kernel Perceptron (if n is small) or the limited-conjunction kernel described in Section 2 (if n is large). This is similar to common practice with polynomial kernels[2] where typically a small degree is used to aid convergence. These observations are supported by our preliminary experiments in an NLP domain which are not reported here.

2 Theorem 3: Kernel Perceptron with Exponentially Many Features

It is easily observed, and well known, that the hypothesis w of the Perceptron algorithm is a $\pm$ sum of the previous examples on which prediction mistakes were made. If we let $L(x) \in \{-1, 1\}$ denote the label of example x, then $w = \sum_{v \in M} L(v)v$ where M is the set of examples on which the algorithm made a mistake. Thus the prediction of Perceptron on x is 1 iff $w \cdot x = (\sum_{v \in M} L(v)v) \cdot x = \sum_{v \in M} L(v)(v \cdot x) \geq 0$.

For an example $x \in \{0,1\}^n$ let $\phi(x)$ denote its transformation into an enhanced feature space such as the space of all conjunctions. To run the Perceptron algorithm over the enhanced space we must predict 1 iff $w^\phi \cdot \phi(x) \geq 0$ where w^ϕ is the weight vector in the enhanced space; from the above discussion this holds iff $\sum_{v \in M} L(v)(\phi(v) \cdot \phi(x)) \geq 0$. Denoting $K(v, x) = \phi(v) \cdot \phi(x)$ this holds iff $\sum_{v \in M} L(v)K(v, x) \geq 0$.

Thus we never need to construct the enhanced feature space explicitly; we need only be able to compute the kernel function $K(v, x)$ efficiently. This is the idea behind all so-called kernel methods, which can be applied to any algorithm (such as support vector machines) whose prediction is a function of inner products of examples; see e.g. [3] for a discussion.

The result in Theorem 3 is simply obtained by presenting a kernel function capturing all conjunctions. We also describe kernels for all monotone conjunctions which allow no negative literals, and kernels capturing all (monotone) conjunctions of up to k literals.

The general case: When $\phi(\cdot)$ includes all 3^n conjunctions (with positive and negative literals) $K(x, y)$ must compute the number of conjunctions which are true in both x and y. Clearly, any literal in such a conjunction must satisfy both x and y and thus the corresponding bit in x, y must have the same value. Counting all such conjunctions gives $K(x, y) = 2^{\mathrm{same}(x,y)}$ where $\mathrm{same}(x, y)$ is the number of original features that have the same value in x and y. This kernel has been obtained independently by [8].

[2] Our Boolean kernels are different than standard polynomial kernels in that all the conjunctions are weighted equally. While expressive power does not change, convergence and behavior, do.

Monotone Monomials: In some applications the total number n of basic features may be very large but in any one example only a small number of features take value 1. In other applications the number of features n may not be known in advance (e.g. due to unseen words in text domains). In these cases it may be useful to consider only monotone monomials. To express all monotone monomials we take $K(x,y) = 2^{\mathrm{samepos}(x,y)}$ where $\mathrm{samepos}(x,y)$ is the number of active features common to both x and y.

A parameterized kernel: In general, one may want to trade off expressivity against number of examples and convergence time. Thus we consider a parameterized kernel which captures all conjunctions of size at most k for some $k < n$. The number of such conjunctions that satisfy both x and y is $K(x,y) = \sum_{l=0}^{k} \binom{\mathrm{same}(x,y)}{l}$. This kernel is reported also in [10]. For monotone conjunctions of size at most k we have $K(x,y) = \sum_{l=0}^{k} \binom{\mathrm{samepos}(x,y)}{l}$.

3 Theorem 4: Kernel Perceptron with Exponentially Many Mistakes

We describe a monotone DNF target function and a sequence of labeled examples which cause the monotone kernel Perceptron algorithm to make exponentially many mistakes.

For $x, y \in \{0,1\}^n$ we write $|x|$ to denote the number of 1's in x and $|x \cap y|$ to denote $\mathrm{samepos}(x,y)$. We use the following lemma (constants have not been optimized):

Lemma 6 *There is a set S of n-bit strings $S = \{x^1, \ldots, x^t\} \subset \{0,1\}^n$ with $t = e^{n/9600}$ such that $|x^i| = n/20$ for $1 \le i \le t$ and $|x^i \cap x^j| \le n/80$ for $1 \le i < j \le t$.*

Proof: The proof uses the probabilistic method. For each $i = 1, \ldots, t$ let $x^i \in \{0,1\}^n$ be chosen by independently setting each bit to 1 with probability 1/10. For any i it is clear that $E[|x^i|] = n/10$; a Chernoff bound implies that $\Pr[|x^i| < n/20] \le e^{-n/80}$, and thus the probability that any x^i satisfies $|x^i| < n/20$ is at most $te^{-n/80}$. Similarly, for any $i \neq j$ we have $E[|x^i \cap x^j|] = n/100$; a Chernoff bound implies that $\Pr[|x^i \cap x^j| > n/80] \le e^{-n/4800}$, and thus the probability that any x^i, x^j with $i \neq j$ satisfies $|x^i \cap x^j| > n/80$ is at most $\binom{t}{2} e^{-n/4800}$. For $t = e^{n/9600}$ the value of $\binom{t}{2} e^{-n/4800} + te^{-n/80}$ is less than 1. Thus for some choice of $x^1, \ldots, x^t$ we have each $|x^i| \ge n/20$ and $|x^i \cap x^j| \le n/80$. For any x^i which has $|x^i| > n/20$ we can set $|x^i| - n/20$ of the 1s to 0s, and the lemma is proved. ■

The target DNF is very simple: it is the single conjunction $x_1 x_2 \ldots x_n$. While the original Perceptron algorithm over the n features $x_1, \ldots, x_n$ makes at most $\mathrm{poly}(n)$ mistakes for this target function, we now show that the monotone kernel Perceptron algorithm which runs over all 2^n monotone monomials can make $2 + e^{n/9600}$ mistakes.

Recall that at the beginning of the Perceptron algorithm's execution all 2^n coordinates of w^ϕ are 0. The first example is the negative example 0^n; since $w^\phi \cdot \phi(x) = 0$ Perceptron incorrectly predicts 1 on this example. The resulting update causes the coefficient $w^\phi_\emptyset$ corresponding to the empty monomial (satisfied by any example x) to become -1 but all $2^n - 1$ other coordinates of w^ϕ remain 0. The next example is the positive example 1^n. For this example we have $w^\phi \cdot \phi(x) = -1$ so Perceptron incorrectly predicts -1. Since all 2^n monotone conjunctions are satisfied by this example the resulting update causes $w^\phi_\emptyset$ to become 0 and all $2^n - 1$ other coordinates of w^ϕ to become 1. The next $e^{n/9600}$ examples are the vectors $x^1, \ldots, x^t$ described in Lemma 6. Since each such example has $|x^i| = n/20$ each example is negative; however as we now show the Perceptron algorithm will predict 1 on each of these examples.

Fix any value $1 \le i \le e^{n/9600}$ and consider the hypothesis vector w^ϕ just before example x^i is received. Since $|x^i| = n/20$ the value of $w^\phi \cdot \phi(x^i)$ is a sum of the $2^{n/20}$ different

coordinates w_T^ϕ which correspond to the monomials satisfied by x^i. More precisely we have $w^\phi \cdot \phi(x^i) = \sum_{T \in A_i} w_T^\phi + \sum_{T \in B_i} w_T^\phi$ where A_i contains the monomials which are satisfied by x^i and x^j for some $j \neq i$ and B_i contains the monomials which are satisfied by x^i but no x^j with $j \neq i$. We lower bound the two sums separately.

Let T be any monomial in A_i. By Lemma 6 any $T \in A_i$ contains at most $n/80$ variables and thus there can be at most $\sum_{r=0}^{n/80} \binom{n/20}{r}$ monomials in A_i. Using the well known bound $\sum_{j=0}^{\alpha\ell} \binom{\ell}{j} \leq 2^{H(\alpha)\ell}$ where $\alpha \leq 1/2$ and $H(\alpha)$ is the binary entropy function there can be at most $2^{0.041n}$ terms in A_i. Moreover the value of each w_T^ϕ must be at least $-e^{n/9600}$ since w_T^ϕ decreases by at most 1 for each example, and hence $\sum_{T \in A_i} w_T^\phi \geq -e^{n/9600} 2^{0.041n} > -2^{0.042n}$. On the other hand, for any $T \in B_i$ we clearly have $w_T^\phi = 1$. By Lemma 6 for any $r > n/80$ every r-variable monomial satisfied by x_i must belong to B_i, and hence $\sum_{T \in B_i} w_T^\phi \geq \sum_{r=n/80+1}^{n/20} \binom{n/20}{r} > 2^{0.049n}$. Combining these inequalities we have $w \cdot x^i \geq -2^{0.042n} + 2^{0.049n} > 0$ and hence the Perceptron prediction on x^i is 1.

4 Theorem 5: Learning DNF with Kernel Winnow is Hard

In this section, for $x \in \{0,1\}^n$ $\phi(x)$ denotes the $(2^n - 1)$-element vector whose coordinates are all nonempty monomials (monotone conjunctions) over $x_1, \ldots, x_n$. A sequence of labeled examples $\langle x^1, b_1 \rangle, \ldots, \langle x^t, b_t \rangle$ is *monotone consistent* if it is consistent with some monotone function, i.e. $x_k^i \leq x_k^j$ for all $k = 1, \ldots, n$ implies $b_i \leq b_j$. If S is monotone consistent and has t labeled examples then clearly there is a monotone DNF formula consistent with S which contains at most t conjunctions. We consider the following problem:

KERNEL WINNOW PREDICTION(α, θ) (KWP)
Instance: Monotone consistent sequence $S = \langle x^1, b_1 \rangle, \ldots, \langle x^t, b_t \rangle$ of labeled examples with each $x^i \in \{0,1\}^m$ and each $b_i \in \{-1, 1\}$; unlabeled example $z \in \{0,1\}^m$.
Question: Is $w^\phi \cdot \phi(z) \geq \theta$, where w^ϕ is the $N = (2^m - 1)$-dimensional hypothesis vector generated by running Winnow(α, θ) on the example sequence $\langle \phi(x^1), b_1 \rangle, \ldots \langle \phi(x^t), b_t \rangle$?

In order to run Winnow over all $2^m - 1$ nonempty monomials to learn monotone DNF, one must be able to solve KWP efficiently. The main result of this section is proved by showing that KWP is computationally hard for any parameter settings which yield a polynomial mistake bound for Winnow via Theorem 2.

Theorem 7 *Let $N = 2^m - 1$ and $\alpha > 1$, $\theta \geq 1$ be such that $\max(\frac{\alpha}{\alpha - 1} \cdot \frac{N}{\theta}, (\alpha + 1)(1 + \log_\alpha \theta)) =$ poly(m). Then KWP(α, θ) is #P-hard.*

Proof of Theorem 7: For N, α and θ as described above it can easily be verified that $1 + \frac{1}{\text{poly}(m)} < \alpha < \text{poly}(m)$ and $\frac{2^m}{\text{poly}(m)} < \theta < 2^{\text{poly}(m)}$. The proof of the theorem is a reduction from the following #P-hard problem [9]: (See [9] also for details on #P.)

MONOTONE 2-SAT (M2SAT)
Instance: Monotone 2-CNF Boolean formula $F = c_1 \wedge c_2 \wedge \ldots \wedge c_r$ with $c_i = (y_{i_1} \vee y_{i_2})$ and each $y_{i_j} \in \{y_1, \ldots, y_n\}$; integer K such that $1 \leq K \leq 2^n$.
Question: Is $|F^{-1}(1)| \geq K$, i.e. does F have at least K satisfying assignments in $\{0,1\}^n$?

4.1 High-Level Idea of the Proof

The high level idea of the proof is simple: let (F, K) be an instance of M2SAT where F is defined over variables $y_1, \ldots, y_n$. The Winnow algorithm maintains a weight w_T^ϕ for each monomial T over variables $x_1, \ldots, x_n$. We define a 1-1 correspondence between these monomials T and truth assignments $y^T \in \{0,1\}^n$ for F, and we give a sequence of examples for Winnow which causes $w_T^\phi \approx 0$ if $F(y^T) = 0$ and $w_T^\phi = 1$ if $F(y^T) = 1$.

The value of $w^\phi \cdot \phi(z)$ is thus related to $|F^{-1}(1)|$; some additional work ensures that $w^\phi \cdot \phi(z) \geq \theta$ if and only if $|F^{-1}(1)| \geq K$.

In more detail, let $U = n + 1 + \lceil (\lceil \log_\alpha 4 \rceil + 1) \log \alpha \rceil$, $V = \lceil \frac{n+1}{\log \alpha} \rceil + 1$, $W = \lceil \frac{U+2}{\log \alpha} \rceil + 1$ and $m = n + U + 6Vn^2 + 6UW + 3$. We describe a polynomial time transformation which maps an n-variable instance (F, K) of M2SAT to an m-variable instance (S, z) of KWP(α, θ) where $S = \langle x^1, b_1 \rangle, \ldots, \langle x^t, b_t \rangle$ is monotone consistent, each x^i and z belong to $\{0,1\}^m$ and $w^\phi \cdot \phi(z) \geq \theta$ if and only if $|F^{-1}(1)| \geq K$.

The Winnow variables $x_1, \ldots, x_m$ are divided into three sets A, B and C where $A = \{x_1, \ldots, x_n\}$, $B = \{x_{n+1}, \ldots, x_{n+U}\}$ and $C = \{x_{n+U+1}, \ldots, x_m\}$. The unlabeled example z is $1^{n+U}0^{m-n-U}$, i.e. all variables in A and B are set to 1 and all variables in C are set to 0. We thus have $w^\phi \cdot \phi(z) = M_A + M_B + M_{AB}$ where $M_A = \sum_{\emptyset \neq T \subseteq A} w_T^\phi$, $M_B = \sum_{\emptyset \neq T \subseteq B} w_T^\phi$ and $M_{AB} = \sum_{T \subseteq A \cup B, T \cap A \neq \emptyset, T \cap B \neq \emptyset} w_T^\phi$. We refer to monomials $\emptyset \neq T \subseteq A$ as *type-A* monomials, monomials $\emptyset \neq T \subseteq B$ as *type-B* monomials, and monomials $T \subseteq A \cup B, T \cap A \neq \emptyset, T \cap B \neq \emptyset$ as *type-AB* monomials.

The example sequence S is divided into four stages. Stage 1 results in $M_A \approx |F^{-1}(1)|$; as described below the n variables in A correspond to the n variables in the CNF formula F. Stage 2 results in $M_A \approx \alpha^q |F^{-1}(1)|$ for some positive integer q. Stages 3 and 4 together result in $M_B + M_{AB} \approx \theta - \alpha^q K$. Thus the final value of $w^\phi \cdot \phi(z)$ is approximately $\theta + \alpha^q(|F^{-1}(1)| - K)$, so we have $w^\phi \cdot \phi(z) \geq \theta$ if and only if $|F^{-1}(1)| \geq K$.

Since all variables in C are 0 in z, if T includes a variable in C then the value of w_T^ϕ does not affect $w^\phi \cdot \phi(z)$. The variables in C are "slack variables" which (i) make Winnow perform the correct promotions/demotions and (ii) ensure that S is monotone consistent.

4.2 Details of the Proof

Stage 1: Setting $M_A \approx |F^{-1}(1)|$. We define the following correspondence between truth assignments $y^T \in \{0,1\}^n$ and monomials $T \subseteq A$: $y_i^T = 0$ if and only if x_i is not present in T. For each clause $y_{i_1} \vee y_{i_2}$ in F, Stage 1 contains V negative examples such that $x_{i_1} = x_{i_2} = 0$ and $x_i = 1$ for all other $x_i \in A$. Assuming that (1) Winnow makes a false positive prediction on each of these examples and (2) in Stage 1 Winnow never does a promotion on any example which has any variable in A set to 1, then after Stage 1 we will have that $w_T^\phi = 1$ if $F(y^T) = 1$ and $0 < w_T^\phi \leq \alpha^{-V}$ if $F(y^T) = 0$. Thus we will have $M_A = |F^{-1}(1)| + \gamma_1$ for some $0 < \gamma_1 < 2^n \alpha^{-V} < \frac{1}{2}$.

We now show how the Stage 1 examples cause Winnow to make a false positive prediction on negative examples which have $x_{i_1} = x_{i_2} = 0$ and $x_i = 1$ for all other i in A as described above. For each such negative example in Stage 1 six new slack variables $x_{\beta+1}, \ldots, x_{\beta+6} \in C$ are used as follows: Stage 1 has $\lceil \log_\alpha(\theta/3) \rceil$ repeated instances of the positive example which has $x_{\beta+1} = x_{\beta+2} = 1$ and all other bits 0. These examples cause promotions which result in $\theta \leq w_{x_{\beta+1}}^\phi + w_{x_{\beta+2}}^\phi + w_{x_{\beta+1}x_{\beta+2}}^\phi < \alpha\theta$ and hence $w_{x_{\beta+1}}^\phi \geq \theta/3$. Two other groups of similar examples (the first with $x_{\beta+3} = x_{\beta+4} = 1$, the second with $x_{\beta+5} = x_{\beta+6} = 1$) cause $w_{x_{\beta+3}}^\phi \geq \theta/3$ and $w_{x_{\beta+5}}^\phi \geq \theta/3$. The next example in S is the negative example which has $x_{i_1} = x_{i_2} = 0$, $x_i = 1$ for all other x_i in A, $x_{\beta+1} = x_{\beta+3} = x_{\beta+5} = 1$ and all other bits 0. For this example $w^\phi \cdot \phi(x) > w_{x_{\beta+1}}^\phi + w_{x_{\beta+3}}^\phi + w_{x_{\beta+5}}^\phi \geq \theta$ so Winnow makes a false positive prediction.

Since F has at most n^2 clauses and there are V negative examples per clause, this construction can be carried out using $6Vn^2$ slack variables $x_{n+U+1}, \ldots, x_{n+U+6Vn^2}$.

Stage 2: Setting $M_A \approx \alpha^q |F^{-1}(1)|$. The first Stage 2 example is a positive example with $x_i = 1$ for all $x_i \in A$, $x_{n+U+6Vn^2+1} = 1$ and all other bits 0. Since each of the 2^n

monomials which contain $x_{n+U+6Vn^2+1}$ and are satisfied by this example have $w_T^\phi = 1$, we have $w^\phi \cdot \phi(x) = 2^n + |F^{-1}(1)| + \gamma_1 < 2^{n+1}$. Since $\theta > 2^m/\text{poly}(m) > 2^{n+1}$ after the resulting promotion we have $w^\phi \cdot \phi(x) = \alpha(2^n + |F^{-1}(1)| + \gamma_1) < \alpha 2^{n+1}$.

Let $q = \lceil \log_\alpha(\theta/2^{n+1}) \rceil - 1$, so $\alpha^q 2^{n+1} < \theta \leq \alpha^{q+1} 2^{n+1}$. Stage 2 consists of q repeated instances of the positive example described above. After these promotions we have $w^\phi \cdot \phi(x) = \alpha^q(2^n + |F^{-1}(1)| + \gamma_1) < \alpha^q 2^{n+1} < \theta$. Since $1 < |F^{-1}(1)| + \gamma_1 < 2^n$ we also have $\alpha^q < M_A = \alpha^q(|F^{-1}(1)| + \gamma_1) < \alpha^q 2^n < \theta/2$.

Stage 3: Setting $M_B \approx p$. At the start of Stage 3 each type-B and type-AB monomial T has $w_T^\phi = 1$. There are n variables in A and U variables in B so at the start of Stage 2 we have $M_B = 2^U - 1$ and $M_{AB} = (2^n - 1)(2^U - 1)$. Since no example in Stages 3 or 4 satisfies any x_i in A, at the end of Stage 4 M_A will still be $\alpha^q(|F^{-1}(1)| + \gamma_1)$ and M_{AB} will still be $(2^n - 1)(2^U - 1)$. Thus ideally at the end of Stage 4 the value of M_B would be $\theta - (2^n - 1)(2^U - 1) - \alpha^q(K + \gamma_1)$, since this would imply that $w^\phi \cdot \phi(z) = \theta + \alpha^q(|F^{-1}(1)| - K)$ which is at least θ if and only if $|F^{-1}(1)| \geq K$. However it is not necessary for M_B to assume this exact value; since $|F^{-1}(1)|$ must be an integer and $0 < \gamma_1 < \frac{1}{2}$, as long as $\theta - (2^n - 1)(2^U - 1) - \alpha^q K \leq M_B < \theta - (2^n - 1)(2^U - 1) - \alpha^q(K - \frac{1}{2})$ we will have that $M_A + M_B + M_{AB} \geq \theta$ if and only if $|F^{-1}(1)| \geq K$.

For ease of notation let D denote $\theta - (2^n - 1)(2^U - 1) - \alpha^q K$. We now describe the examples in Stages 3 and 4 and show that they will cause M_B to satisfy $D \leq M_B < D + \frac{1}{2}\alpha^q$.

Let $c = \lceil \log_\alpha 4 \rceil$, so $\alpha^{q-c} \leq \frac{1}{4}\alpha^q$ and hence there is a unique smallest integer p such that $D \leq p\alpha^{q-c} < D + \frac{1}{4}\alpha^q$. The Stage 3 examples will result in $p < M_B < p + \frac{1}{4}$. Using the definition of D and the fact that $1 \leq K < 2^n$ it can be verified that $\alpha^{q-c} < D \leq p\alpha^{q-c} < D + \frac{1}{4}\alpha^q \leq \theta - \frac{3}{4}\alpha^q \leq \alpha^{q+1}2^{n+1} - 3\alpha^{q-c} = \alpha^{q-c} \cdot (\alpha^{c+1}2^{n+1} - 3)$. Hence we have $1 < p \leq \alpha^{c+1}2^{n+1} - 3 \leq 2^{n+1+\lceil (c+1)\log\alpha \rceil} - 3 = 2^U - 3$. We use the following lemma:

Lemma 8 *For all $\ell \geq 1$, for all $1 \leq p \leq 2^\ell - 1$, there is a monotone CNF $F_{\ell,p}$ over ℓ Boolean variables which has at most ℓ clauses, has exactly p satisfying assignments in $\{0,1\}^\ell$, and can be constructed from ℓ and p in poly(ℓ) time.*

Proof: The proof is by induction on ℓ. For the base case $\ell = 1$ we have $p = 1$ and $F_{\ell,p} = x_1$. Assuming the lemma is true for $\ell = 1, \ldots, k$ we now prove it for $\ell = k + 1$:

If $1 \leq p \leq 2^k - 1$ then the desired CNF is $F_{k+1,p} = x_{k+1} \wedge F_{k,p}$. Since $F_{k,p}$ has at most k clauses $F_{k+1,p}$ has at most $k + 1$ clauses. If $2^k + 1 \leq p \leq 2^{k+1} - 1$ then the desired CNF is $F_{k+1,p} = x_{k+1} \vee F_{k,p-2^k}$. By distributing x_k over each clause of $F_{k,p-2^k}$ we can write $F_{k+1,p}$ as a CNF with at most k clauses. If $p = 2^k$ then $F_{k,p} = x_1$. ■

Let $F_{U,p}$ be an r-clause monotone CNF formula over the U variables in B which has p satisfying assignments. Similar to Stage 1, for each clause of $F_{U,p}$, Stage 3 has W negative examples corresponding to that clause, and as in Stage 1 slack variables in C are used to ensure that Winnow makes a false positive prediction on each such negative example. Thus the examples in Stage 3 cause $M_B = p + \gamma_2$ where $0 < \gamma_2 < 2^U\alpha^{-W} < \frac{1}{4}$. Since six slack variables in C are used for each negative example and there are $rW \leq UW$ negative examples, the slack variables $x_{n+U+6Vn^2+2}, \ldots, x_{m-2}$ are sufficient for Stage 3.

Stage 4: Setting $M_B + M_{AB} \approx \theta - \alpha^q K$. All that remains is to perform $q - c$ promotions on examples which have each x_i in B set to 1. This will cause $D \leq p\alpha^{q-c} < (p + \gamma_2)\alpha^{q-c} = M_B < D + \frac{1}{4}\alpha^q + \gamma_2\alpha^{q-c} < D + \frac{1}{2}\alpha^q$ which is as desired.

It can be verified from the definitions of U and c that $\frac{U-n}{\log\alpha} \geq c$. The first $q - \lceil \frac{U-n}{\log\alpha} \rceil$ examples in S are all the same positive example which has each x_i in B set to 1 and $x_{m-1} = 1$. The first time this example is received $w^\phi \cdot \phi(x) = 2^U + p + \gamma_2 < 2^{U+1}$. It can

be verified that $2^{U+1} < \theta$, so Winnow performs a promotion; after $q - \lceil \frac{U-n}{\log \alpha} \rceil$ occurrences of this example $w^\phi \cdot \phi(x) = \alpha^{q-\lceil \frac{U-n}{\log \alpha} \rceil}(2^U + p + \gamma_2) < \alpha^{q-\lceil \frac{U-n}{\log \alpha} \rceil} 2^{U+1} \leq \alpha^q 2^{n+1} < \theta$ and $M_B = \alpha^{q-\lceil \frac{U-n}{\log \alpha} \rceil}(p + \gamma_2)$.

The remaining examples in Stage 4 are $\lceil \frac{U-n}{\log \alpha} \rceil - c$ repetitions of the positive example x which has each x_i in B set to 1 and $x_m = 1$. If promotions occurred on each repetition of this example then we would have $w^\phi \cdot \phi(x) = \alpha^{\lceil \frac{U-n}{\log \alpha} \rceil - c}(2^U + \alpha^{q-\lceil \frac{U-n}{\log \alpha} \rceil}(p + \gamma_2))$, so we need only show that this is less than θ. We reexpress this quantity as $\alpha^{\lceil \frac{U-n}{\log \alpha} \rceil - c} 2^U + \alpha^{q-c}(p+\gamma_2)$. We have $\alpha^{q-c}(p+\gamma_2) < p\alpha^{q-c} + \frac{1}{4}\alpha^{q-c} \leq \theta - \frac{3}{4}\alpha^q + \frac{1}{16}\alpha^q < \theta - \frac{1}{2}\alpha^q$. Some easy manipulations show that $\alpha^{\lceil \frac{U-n}{\log \alpha} \rceil - c} 2^U \leq \frac{\alpha}{2} 2^{2U-n} < \frac{1}{2}\alpha^q$, so indeed $w^\phi \cdot \phi(x) < \theta$.

Finally, we observe that by construction the example sequence S is monotone consistent. Since $m = \text{poly}(n)$ and S contains $\text{poly}(n)$ examples the transformation from M2SAT to KWP(α, θ) is polynomial-time computable and the theorem is proved. (Theorem 7) ■

5 Conclusion

It is necessary to expand the feature space if linear learning algorithms are to learn expressive functions. This work explores the tradeoff between computational efficiency and convergence (i.e. generalization ability) when using expanded feature spaces. We have shown that additive and multiplicative update algorithms differ significantly in this respect; we believe that this fact could have significant practical implications. Future directions include the utilization of the kernels developed here and studying convergence issues of Boolean-kernel Perceptron and Support Vector Machines in the PAC model.

Acknowledgements: R. Khardon was supported by NSF grant IIS-0099446. D. Roth was supported by NSF grants ITR-IIS-00-85836 and IIS-9984168 and by EPSRC grant GR/N03167 while visiting University of Edinburgh. R. Servedio was supported by NSF grant CCR-98-77049 and by a NSF Mathematical Sciences Postdoctoral Fellowship.

References

[1] D. Angluin. Negative results for equivalence queries. *Machine Learning*, 2:121–150, 1990.

[2] A. Carlson, C. Cumby, J. Rosen, and D. Roth. The SNoW learning architecture. Technical Report UIUCDCS-R-99-2101, UIUC Computer Science Department, May 1999.

[3] N. Cristianini and J. Shaw-Taylor. *An Introduction to Support Vector Machines*. Cambridge Press, 2000.

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

[5] W. Maass and M. K. Warmuth. Efficient learning with virtual threshold gates. *Information and Computation*, 141(1):378–386, 1998.

[6] A. Novikoff. On convergence proofs for perceptrons. In *Proceeding of the Symposium on the Mathematical Theory of Automata*, volume 12, pages 615–622, 1963.

[7] D. Roth. Learning to resolve natural language ambiguities: A unified approach. In *Proc. of the American Association of Artificial Intelligence*, pages 806–813, 1998.

[8] K. Sadohara. Learning of boolean functions using support vector machines. In *Proc. of the Conference on Algorithmic Learning Theory*, pages 106–118. Springer, 2001. LNAI 2225.

[9] L. G. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal of Computing*, 8:410–421, 1979.

[10] C. Watkins. Kernels from matching operations. Technical Report CSD-TR-98-07, Computer Science Department, Royal Holloway, University of London, 1999.