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

Roni Khardon

Tufts University

Dan Roth

University of Illinois, Urbana-Champaign

Rocco Servedio

Harvard University

Abstract (I)

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 classifier.

Details (I)

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.

Details (II)

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.

Search Terms

- Learning Boolean functions with Perceptron and Winnow

- Kernels for the Boolean Domain

- Computational Efficiency vs. Generalization Ability over expended feature spaces

- Perceptron: +++++++   −−−−−−−
- Winnow: −−−−−−−   +++++++

Story

- Linear learning algorithms have been applied successfully to several large scale real world classification problems.

- The SNoW system [Roth et.al] has successfully applied variations of Perceptron and Winnow to problems in natural language processing.

- First, extract Boolean features from examples (e.g., text)

- Generate structurally restricted conjunctions of these basic features.

- Use linear learning algorithms over these expanded higher-dimensional examples, in which each conjunction plays the role of a basic feature

Expressiveness, Efficiency, Generalization

Expansion leads to an increase in expressiveness
⇒<u>may improve performance.</u>

Expansion also dramatically increases the number of features
⇒ may adversely affect both the <u>computation time</u> and
<u>convergence rate of learning</u>.

Goal: study the use of kernel functions to expand the feature
space and enhance the learning abilities of Winnow and
Perceptron

Specifically: <u>computational efficiency and convergence</u> over
expanded feature spaces of conjunctions.

Background: Perceptron

- Maintains a weight vector $w \in \mathcal{R}^N$, $\quad w^0 = (0,\ldots,0)$.

- Upon receiving an example $x \in \mathcal{R}^N$

- Predicts according to the linear threshold function $w \bullet x \geq 0$.

  – If $w \bullet x \geq 0$, and the label is -1, $\quad$ w = w-x $\quad$ (**demotion**)

  – If $w \bullet x < 0$, and the label is 1, $\quad$ w = w+x $\quad$ (**promotion**)

  – No weight update if there is no mistake.

Theorem 1 [Novikoff] *Let* $(x^1; y_1),\ldots,: (x^t; y_t)$, *be a sequence of labeled examples with* $x^i \in \mathcal{R}^N$, $\|x^i\| \leq R$ *and* $y_i \in \{-1,1\}$ *for all i. Let* $u \in \mathcal{R}^N$, $\xi > 0$ *be such that* $y\_i \, u \bullet x^i \geq \xi$ *for all i.* *Then Perceptron makes at most* $\|u\|^2 R^2 / \xi^2$ *mistakes on this example sequence.*

Background: Winnow

- Maintains a weight vector $w \in \mathcal{R}^N$, $w^0 = (1, \ldots, 1)$, $\alpha > 1$, $\theta > 1$
- Upon receiving an example $x \in \{0,1\}^N$,
- Predicts according to the linear threshold function $w \cdot x \geq \theta$.
  - If $w \cdot x \geq \theta$, but label is -1, $w_i = w_i / \alpha$ for i s.t. $x_i = 1$ **(demotion)**
  - If $w \cdot x < \theta$, but label is 1, $w_i = w_i \alpha$ for i s.t. $x_i = 1$ **(promotion)**
  - No weight update if there is no mistake.

Theorem 2 [Littlestone] *Let the target function be a* k-*literal monotone disjunction* $f(x_1, \ldots, x_N) = x_{i1} \vee \ldots \vee x_{ik}$. *For any sequence of examples in* $\{0,1\}^N$ *labeled according to* f, *the number of prediction mistakes made by Winnow*$(\alpha; \theta)$ *is at most*

$\alpha/(\alpha - 1) \cdot N / \theta + k(\alpha + 1)(1 + \log_\alpha \theta)$ (Use, e.g. $\alpha = 2$ and $\theta = N$).

Result I:
Kernel Perceptron with Exponentially Many 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$(n; t)$ steps.

Comment Closely related to the problem of efficient learnability of DNF expressions. However, 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.

Question Is the exponential upper bound implied by Theorem 1 tight for kernel Perceptron?.

Answer Yes. Thus kernel Perceptron cannot efficiently learn DNF (next)

Result II:
Kernel Perceptron with Exponentially Many Mistakes

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

Corollary Kernel Perceptron cannot efficiently learn DNF

Result III:
Learning DNF with Kernel Winnow is Hard

Theorem 5 If P≠ #P then there is no polynomial time algorithm which simulates Winnow over exponentially many monotone conjunctive features for learning monotone DNF.

Comment An attractive feature of Winnow (Theorem 2) is that for suitable values of α, θ the bound is logarithmic in the total number of features N (e.g. α = 2 and θ = N). Therefore, if a Winnow analogue of Theorem 3 existed, this would imply efficient learnability of DNF. Theorem 5 shows that no such analogue can exist.

Kernel Perceptron with Exponentially Many Features

Examples $\mathbf{x} \in \{0,1\}^N$, Hypothesis $w \in \mathcal{R}^N$, $f(\mathbf{x}) = \mathbf{Th}_\theta(\sum_{i=1}^{n} \mathbf{w}_i \mathbf{x}_i(\mathbf{x}))$

Let I be the set $\mathbf{t}_1, \mathbf{t}_2, \mathbf{t}_3, \dots$ of monomials over $\mathbf{x}_1, \mathbf{x}_2, \dots \mathbf{x}_n$.

Then we can write a linear function over this new feature space.

$$f(\mathbf{x}) = \mathbf{Th}_\theta\left(\sum_{i \in I} \mathbf{w}_i \mathbf{t}_i(\mathbf{x})\right)$$

Example : $\mathbf{x}_1 \mathbf{x}_2 \mathbf{x}_4(11010) = 1$ $\mathbf{x}_3 \mathbf{x}_4(11010) = 0$

Details (II):
Kernel Perceptron with Exponentially Many Features

$$f(x) = Th_\theta\left(\sum_{i \in I} w_i t_i(x)\right)$$

- Assume running Perceptron over the new feature space

P:set of promoted examples; D:set of demoted examples M= P$\cup$D

$$f(x) = Th_\theta\left(\sum_{i \in I}\left[\sum_{z \in P, t_i(z)=1} 1 - \sum_{z \in D, t_i(z)=1} 1\right] t_i(x)\right) = Th_\theta\left(\sum_{i \in I}\left[\sum_{z \in M} S(z) t_i(z) t_i(x)\right]\right)$$

Where S(z)=1 if z $\in$P and S(z) = -1 if z $\in$D. Reordering:

$$f(x) = Th_\theta\left(\sum_{z \in M} S(z) \sum_{i \in I} t_i(z) t_i(x)\right)$$

With the standard notation: $K(x, z) = \sum_{i \in I} t_i(z) t_i(x)$

$$f(x) = Th_\theta\left(\sum_{z \in M} S(z) K(x, z)\right)$$

- To run Perceptron over the enhanced feature space we must predict 1 iff $w^t \cdot t(x) \geq \theta$,

  where w^t is the weight vector in the enhanced space.

- The previous discussion shows that this holds iff

$$\sum_{z \in M} S(z)K(x, z) \geq \theta$$

Where M - the examples on which the algorithm made mistakes,

$S(Z) \in \{-1,1\}$, the label of the example, and $K(x, z) = \sum_{i \in I} t_i(z)t_i(x)$

Kernel Perceptron with Exponentially Many Features
Boolean Kernels

The general case: To express all 3^n conjunctions (with positive and negative literals) we take $K(x,y) = 2^{same(x,y)}$ where 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 Sadohara'01.

Monotone Monomials: To express all monotone monomials we take $K(x,y) = 2^{samepos(x,y)}$ where samepos(x,y) is the number of active features common to both x and y.

Useful where the total number n of basic features is large (or unknown in advance) but any one example only has a few 1 features.

A parameterized kernel: Captures all conjunctions of size at most k for some k<n: The number of such conjunctions that satisfy both x and y is $K(x, y) = \sum_{l=0}^{k} C(same(x, y),l)$ (also:Watkins'99)

Allows to trade off expressivity against #(examples) and convergence.

Details
Kernel Perceptron with Exponentially Many Mistakes

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.

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.

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 makes at most poly(n) mistakes for this target function, we show that the monotone kernel Perceptron algorithm which runs over all 2^n monotone monomials can make $2 + e^{n/9600}$ mistakes.

Details (II)
Kernel Perceptron with Exponentially Many Mistakes

The first example is the negative examples 0^n . The Perceptron predicts 1. Empty coefficient is demoted. Other remain 0.

The second examples is the positive example 1^n . The Perceptron predicts -1. All coefficient are promoted $(0, 1, 1,\ldots,1)$.

The next $e^{n/9600}$ examples are given by the lemma below. All are negative, but the Perceptron predicts 1 on all.

Lemma 6 *There is a set 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* $1 \leq i \leq t$ *and* $|x^i \cap x^i| = n/80,$ *for* $1 \leq i < j \leq t$

Details
Learning DNF with Kernel Winnow is Hard

Theorem 5 If P≠ #P then there is no polynomial time algorithm which simulates Winnow over exponentially many monotone conjunctive features for learning monotone DNF.

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. Theorem 5 is proved by showing that KWP is computationally hard for any parameter settings which yield a polynomial mistake bound for Winnow via Theorem 2.

Learning DNF with Kernel Winnow is Hard

KWP(α, θ) is reduced to the problem of counting the number of satisfying assignment of a monotone 2-SAT.

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 $\{0, 1\}^n$?

This is used to prove the hardness results:

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($\alpha, \theta$) is #P-hard.*

See paper for details.

Conclusions

⬦ 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.