Daniel Lowd
University of Washington
lowd@cs.washington.edu

Adversarial Learning

Christopher Meek
Microsoft Research
meek@microsoft.com

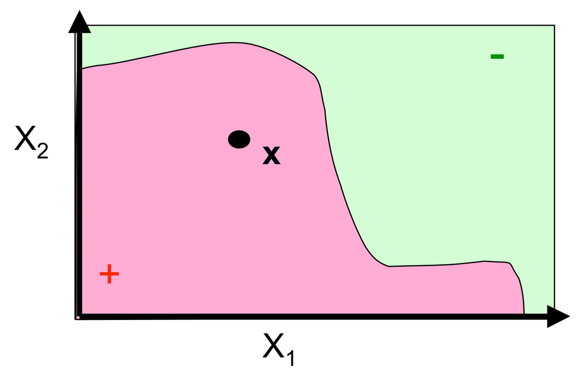
> "If you know the enemy and know yourself, you need not fear the result of a hundred battles."
> -- Sun Tzu, *The Art of War*

Abstract

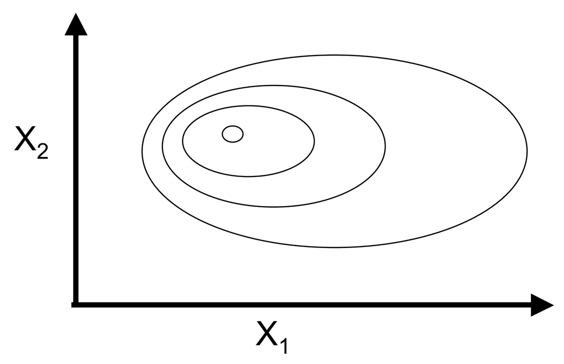
Many classification tasks, such as spam filtering, intrusion detection, and terrorism detection, are complicated by an adversary who wishes to avoid detection. Previous work on adversarial classification has made the unrealistic assumption that the attacker has perfect knowledge of the classifier [Dalvi et al., 2004]. In this paper, we introduce the adversarial classifier reverse engineering (ACRE) learning problem, the task of learning sufficient information about a classifier to construct adversarial attacks. We present efficient algorithms for reverse engineering linear classifiers with either continuous or Boolean features and demonstrate their effectiveness using real data from the domain of spam filtering.

ACRE Learning

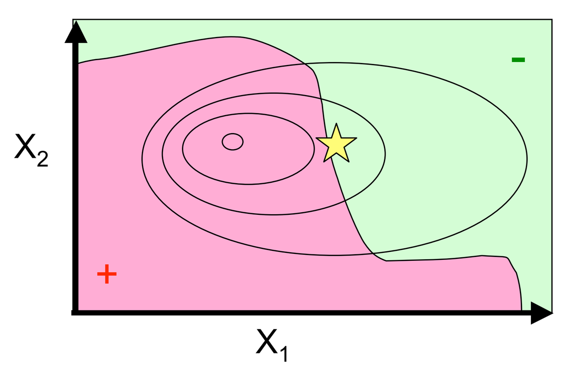
A **classifier**, $c(\mathbf{x})$, maps n-dimensional feature vectors (instances) to classes + and - :



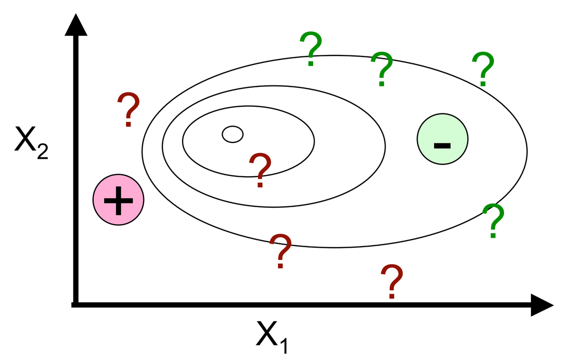
An **adversarial cost function**, $a(\mathbf{x})$, represents the adversary's preference for some instances over others. We visualize this as constant-cost contours in the instance space:



In an **adversarial classifier reverse engineering (ACRE) learning problem**, the adversary tries to minimize $a(\mathbf{x})$ subject to the constraint $c(\mathbf{x}) = -$.



The adversary is given one positive instance, one negative instance, and may issue membership queries to the classifier. In each query, the adversary learns the class of a single chosen instance.

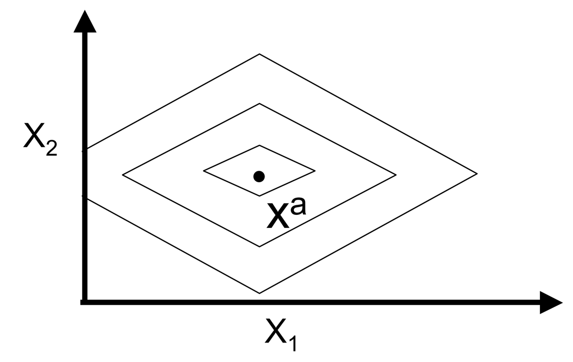


A set of classifiers and cost functions is ***k*-ACRE learnable** if the adversary can always find a negative instance within a factor k of the minimum cost using a polynomial number of queries.

Attacking Linear Classifiers

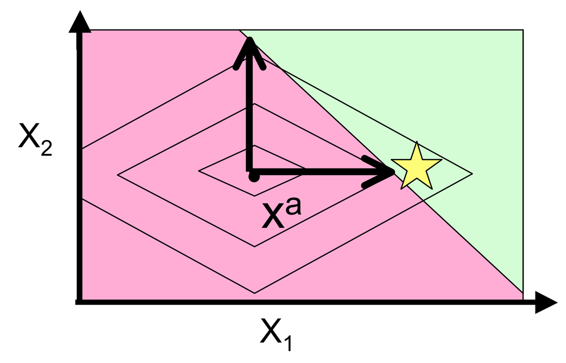
…With Continuous Attributes

A **linear cost function** is of the form: $a(\mathbf{x}) = \sum_i a_i |x_i - x_i^a|$. Informally, a_i represents the cost of changing the ith feature by one unit, with respect to the base instance, $\mathbf{x^a}$.



Theorem: *Linear classifiers with continuous features are ACRE $(1 + \varepsilon)$-learnable under linear cost functions.*

Proof Sketch: We only need to change one feature, the one with highest weight-to-cost ratio. We can efficiently find this feature by doing line searches in each dimension.

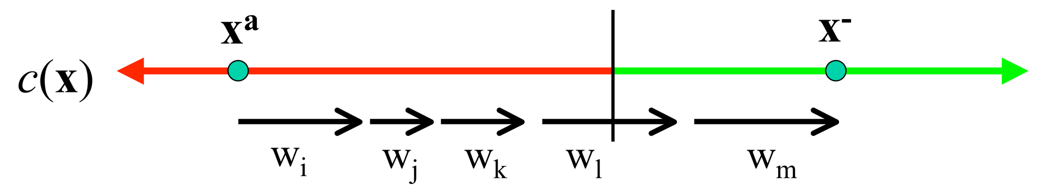


…With Boolean Attributes

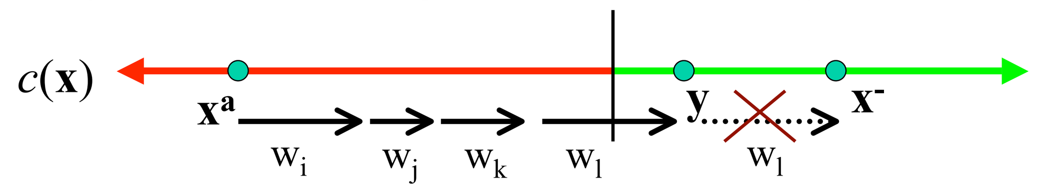
A **uniform linear cost function** assigns unit cost to each changed feature, with respect to the base instance $\mathbf{x^a}$.

Theorem: *Linear classifiers with Boolean features are ACRE 2-learnable under uniform linear cost functions.*

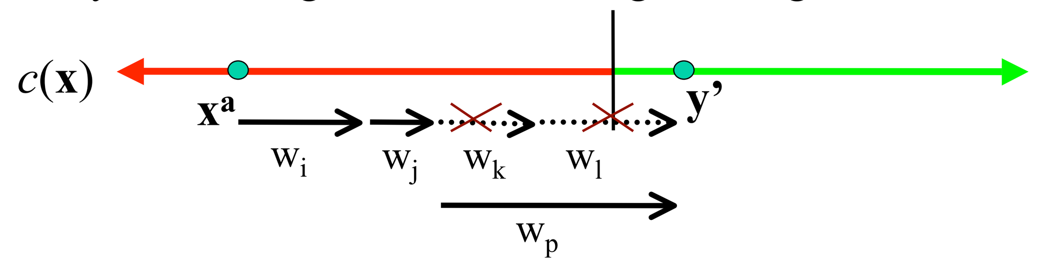
Proof Sketch: Start with the negative example, $\mathbf{x^-}$. Initially, each changed feature f contributes some weight, w_f, to making $\mathbf{x^-}$ negative:



We iteratively reduce the cost in two ways.
1. Remove any individual change that is unnecessary:



2. Replace any two changes with new, single change:



When neither modification can be made without yielding a positive example, the cost must be less than twice optimal.

Empirical Evaluation: Spam

Spammer goal: minimally modify a spam message to achieve a spam that gets past a spam filter.

Corresponding ACRE problem:
spam filter ⟶ linear classifier with Boolean features
"minimally modify" ⟶ uniform linear cost function

Filter configuration:
- Naïve Bayes (NB) and maxent (ME) filters
- 500,000 Hotmail messages for training
- > 250,000 features

Adversary feature sets:
- 23,000 English words (Dict)
- 1,000 most frequent English words (Freq)
- 1,000 random English words (Rand)

Results (see table):
- Reduced feature set almost as good
- Cost ratio is excellent
- Number of queries is reasonable (parallelize)

Conclusion: ACRE algorithms are practical!

Table 1: Empirical Results in Spam Domain

	med. cost	max cost	med. ratio	max ratio	med. queries	max queries
Dict NB	23	723	1.136	1.5	261k	6,472k
Dict ME	10	49	1.167	1.5	119k	646k
Freq NB	34	761	1.105	1.5	25k	656k
Freq ME	12	72	1.108	1.5	10k	95k
Rand NB	31	759	1.120	1.5	23k	755k
Rand ME	12	64	1.158	1.5	9k	78k

For each scenario, we list the median and maximum number of changes required, ratio (relative to optimal), and number of queries required.