CS562 Presentation

Game Theoretic Analysis for Adversarial Learning

Zijian Huang (zijianh4@illinois.edu)

Adversarial Learning

Daniel Lowd, Christopher Meek

Overview

- *adversarial classifier reverse engineering* (ACRE) *learning problem*
- *minimal adversarial cost* (MAC) & *ACRE learnable*
- k-IMAC & *ACRE k-learnable*
- Find IMAC for continuous features & boolean features

Problem Definition – Notation

- X: *instance space*
- X_i: features (may be real, integer, Boolean, etc.)
- $\boldsymbol{x} \in X$: *intances*
- x_i: value of the *i*th feature in instance $\boldsymbol{x}$
- c: *classifier*, a function from instances $\boldsymbol{x} \in X$ to $\{0,1\}$
- *positive instances*: instances x for which $c(\boldsymbol{x}) = 0$
- *negative instances*: instances x for which $c(\boldsymbol{x}) = 1$

Problem Definition – Assumption

- The adversary can issue *membership queries* to the classfier for arbitrary instances;
- The adversary has access to an *adversarial cost function* $a(\boldsymbol{x})$ that maps instances to nonnegative real numbers;
- The adversary is provided with one positive instance, $\boldsymbol{x}^+$, and one negative instance, $\boldsymbol{x}^-$.

Problem Definition – IMAC

- *minimal adversarial cost* (MAC) of a classifier c and cost function a: the minimum cost $a(\boldsymbol{x})$ over all instances $\boldsymbol{x}$ classified negatively by c

$$MAC(c, a) = \min_{\boldsymbol{x}: c(\boldsymbol{x})=0} a(\boldsymbol{x})$$

- *instances of minimal adversarial cost* (IMAC): the set of all instances $\boldsymbol{x}$ classified negatively by c and with minimal cost

$$IMAC(c, a) = \{\boldsymbol{x} \in X | a(\boldsymbol{x}) = MAC(c, a) \; and \; c(\boldsymbol{x}) = 0\}$$

- *adversarial classifier reverse engineering* (ACRE) *learning problem* for classifier c and adversarial cost function a: find an instance $\boldsymbol{x} \in IMAC(c, a)$

Problem Definition – ACRE learnable

- a set of classifiers C is *ACRE learnable* under a set of cost functions A: if an algorithm exists that, for any $c \in C$ and any $a \in A$, finds some $\boldsymbol{x} \in IMAC(c, a)$ using only polynomially many membership in:

(1) n: the number of features

(2) **size**(c): the encoded size of c

(3) **size**(x^+, x^-): the encoded size of the positive and negative instances (assume to be encoded as strings of digits in a known, fixed base)

Problem Definition – k-IMAC & *ACRE k-learnable*

- *k*-IMAC: the set of all negative instances whose costs are within a constant factor *k* of the MAC:

$$k - \mathrm{IMAC}(c, a) = \{\mathbf{x} \in X | a(\mathbf{x}) \leq k \cdot MAC(c, a) \text{ and } c(\mathbf{x}) = 0\}$$

- *ACRE k-learnable*

Adversarial Cost Functions

- *linear cost function*:

$$a(\mathbf{x}) = \sum_i a_i |x_i - x_i^a|$$

- *uniform linear cost functions*: $a_i = 1$

Boolean Formulae

- In general, this set of classifiers is not ACRE learnable under most interesting adversarial cost functions;
- Certain classes of Boolean formulae are ACRE learnable. For example,

(1) Monotone k-DNF problems: $O(n^k)$

(2) Boolean Formulae only with $\bigwedge$ and one positive instance: n queries

Linear Classifiers

- weight vector: $\mathbf{w} \in R^n$, threshold: T
- Positive instance: $\mathbf{w} \cdot \mathbf{x} > T$, $\mathbf{gap}(\mathbf{x}) \coloneqq |\mathbf{w} \cdot \mathbf{x} - T|$
- *sign witness* to a feature f: a pair of instances, $\mathbf{s}^+$ and $\mathbf{s}^-$ such that $c(\mathbf{s}^+) = 1, c(\mathbf{s}^-) = 0$, and $\forall i \neq f, s_i^+ = s_i^-$

Linear Classifiers – Continuous Features

Algorithm 1 FINDCONTINUOUSWEIGHTS(x^+, x^-, ϵ, δ)

$(\mathbf{s}^+, \mathbf{s}^-, f) \leftarrow$ FINDWITNESS($\mathbf{x}^+, \mathbf{x}^-$)
$w_f \leftarrow 1.0 \cdot (s_f^+ - s_f^-)/|s_f^+ - s_f^-|$
Use $(\mathbf{s}^+, \mathbf{s}^-)$ to find negative instance $\mathbf{x}$ with $\mathsf{gap}(\mathbf{x}) < \epsilon/4$
$x_f \leftarrow x_f - w_f$
for each feature $i \neq f$ **do**
  Let $\hat{\mathbf{i}}$ be the unit vector along the ith dimension
  if $c(\mathbf{x} + \hat{\mathbf{i}}/\delta) = c(\mathbf{x} - \hat{\mathbf{i}}/\delta)$ **then**
    $w_i \leftarrow 0$
  else
    $w_i \leftarrow$ LINESEARCH($\mathbf{x}, i, \epsilon/4$)
  end if
end for

THEOREM 5.1. *Let c be a continuous linear classifier with vector of weights $\mathbf{w}$, such that the magnitude of the ratio between two non-zero weights is never less than δ. Given positive and negative instances $\mathbf{x}^+$ and $\mathbf{x}^-$, we can find each weight within a factor of $1+\epsilon$ using a polynomial number of queries.*

Linear Classifiers – Continuous Features

- The gap btween original sign witness

$$\begin{aligned} \text{gap}(\mathbf{s}^+) + \text{gap}(\mathbf{s}^-) &= |\mathbf{w}\cdot\mathbf{s}^+ - T| + |\mathbf{w}\cdot\mathbf{s}^- - T| \\ &= \mathbf{w}\cdot\mathbf{s}^+ - \mathbf{w}\cdot\mathbf{s}^- \\ &= \mathbf{w}\cdot(\mathbf{s}^+ - \mathbf{s}^-) \\ &= |\mathbf{s}_\mathbf{f}^+ - \mathbf{s}_\mathbf{f}^-| \end{aligned}$$

- Refine the gap using a binary search on the value of feature f to find a negative instance $\mathbf{x}$ with gap less than $\epsilon/4$: $O\left(\log\left(\frac{1}{\epsilon}\right) + \text{size}(\mathbf{s}^+, \mathbf{s}^-)\right)$
- Total error: at most $(1+\frac{\epsilon}{4})^2 < 1+\epsilon$, for $\epsilon < 8$
- The number of queries per feature is logarithmic in $1/\epsilon$ and the ratio w_f/w_i, where $\log(w_f/w_i)$ is $O(\text{size}(c))$

Linear Classifiers – Continuous Features

- Total error: at most $(1+\frac{\epsilon}{4})^2 < 1+\epsilon$, for $\epsilon < 8$
- The number of queries: $O\left(\log\left(\frac{1}{\epsilon}\right) + \log(\mathbf{gap}(\mathbf{x^a}))\right)$

Algorithm 2 FINDCONTINUOUSIMAC(x^+, x^-, ϵ)

$\delta \leftarrow \min_i a_i$
Run FINDCONTINUOUSWEIGHTS($x^+, x^-, \epsilon/4, \delta$)
$f \leftarrow \mathsf{argmax}_i |w_i|/a_i$
$t \leftarrow$ LINESEARCH($\mathbf{x^a}, f, \epsilon/4$)
Let $\hat{\mathbf{f}}$ be the unit vector along dimension f
return $\mathbf{x^a} + t\hat{\mathbf{f}}$

THEOREM 5.2. *Linear classifiers with continuous features are ACRE $(1+\epsilon)$-learnable under linear cost functions.*

Linear Classifiers – Boolean Features

Algorithm 3 FINDBOOLEANIMAC($\mathbf{x}^{\mathbf{a}}, \mathbf{x}^{-}$)

$\mathbf{y} \leftarrow \mathbf{x}^{-}$
repeat
  $\mathbf{y}^{\mathbf{prev}} \leftarrow \mathbf{y}$
  for all $f \in C_y$ **do**
    toggle f in $\mathbf{y}$
    if $c(\mathbf{y}) = 1$ **then**
      toggle f in $\mathbf{y}$
    end if
  end for
  for all $f_1 \in C_y; f_2 \in C_y; f_3 \notin C_y$ **do**
    toggle f_1, f_2, and f_3 in $\mathbf{y}$
    if $c(\mathbf{y}) = 1$ **then**
      toggle f_1, f_2, and f_3 in $\mathbf{y}$
    end if
  end for
until $\mathbf{y}^{\mathbf{prev}} = \mathbf{y}$
return $\mathbf{y}$

Linear Classifiers – Boolean Features

THEOREM 5.3. *In a linear classifier with Boolean features, determining if a sign witness exists for a given feature is NP-complete.*

- The problem is NP-hard because it is a reduction from the subset sum problem

THEOREM 5.4. *Boolean linear classifiers are ACRE 2-learnable under uniform linear cost functions.*

- Proof omitted (refer to the paper)

LEMMA 5.3.1. *For two sequences of non-positive real numbers* $(s_1, \ldots, s_m)$ *and* $(t_1, \ldots, t_n)$*, if the following conditions hold*

$$\sum_i s_i \leq u \tag{1}$$

$$n > 2m \geq 2 \tag{2}$$

$$\text{for all } j, \sum_i t_i - t_j > u \tag{3}$$

then there exists j, k, l *such that* $l \neq k$ *and* $s_j - t_k - t_l < 0$.

Empirical Study: Spam Filtering

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

Future Work

- Other forms of adversarial cost functions, types of classifiers, more complex learning scenarios;
- Proving the learnability of less restricted Boolean formulas under different adversarial cost functions;
- What conditions ACRE learning robust to noisy classifiers?
- When queries are expensive?

Discussion

Adversarial Classification

Nilesh Dalvi, Pedro Domingos, Mausam, Sumit Sanghai, Deepak Verma

Overview

- Formulate adversarial learning problems from game theory perspectives
- Classifier strategy & Adversary strategy in one round game

Problem Definition

Symbol	Meaning
$X = (X_1, X_2, \ldots, X_n)$	Instance.
$P(x)$	Probability distribution of untainted data.
X_i	i^{th} feature (attribute).
$\mathcal{X}, \mathcal{X}_i$	Domain of X and X_i, respectively.
x , x_i	An instance and the i^{th} attribute of that instance.
$\mathcal{S}, \mathcal{T}$	Training and test set.
$y_C = C(x)$	The CLASSIFIER function.
$x_A = \mathcal{A}(x)$	The ADVERSARY transformation.
V_i	Cost of measuring X_i.
$U_C(y_C, y)$	Utility for CLASSIFIER of classifying as y_C an instance of class y.
$W_i(x_i, x_i'), W(x, x')$	Cost of changing the i^{th} feature from x_i to x_i' and instance x to x', respectively.
$U_A(y_C, y)$	Utility accrued by ADVERSARY when CLASSIFIER classifies as y_C an instance of class y.
$\mathcal{X}_C(x)$	Set of features measured by C.
$LO_C(x_i)$	Log-odds or "contribution" of i^{th} attribute to naive Bayes classifier $\left(\ln \frac{P(X_i=x_i\|+)}{P(X_i=X_i\|-)}\right)$.
$gap(x)$	$gap(x) > 0 \iff$ NB classifies x as positive $\left(LO_C(x) - \frac{U_C(-,-)-U_C(+,-)}{U_C(+,+)-U_C(-,+)}\right)$.
ΔU_A	ADVERSARY's utility gain from successfully camouflaging a positive instance $(U_A(-,+)-U_A(+,+))$.
$\Delta LO_{i,x_i'}$	"Gain" towards making x negative by changing i^{th} feature to x_i' $(LO_C(x_i) - LO_C(x_i'))$.
$MCC(x)$	Nearest instance (costwise) to x which Naive Bayes classifies as negative.
$x_{[i=x_i']}$	An instance identical to x except that i^{th} attribute is changed to $x_i' \in \mathcal{X}_i$.
$P_A(x)$	Probability distribution after ADVERSARY has modified the data.

Table 1: Summary of the notation used in this paper.

Problem Definition

$$U_C = \sum_{(x,y)\in\mathcal{XY}} P(x,y)\left[U_C(\mathcal{C}(\mathcal{A}(x)),y) - \sum_{X_i\in\mathcal{X}_\mathcal{C}(x)} V_i\right] \quad (1)$$

$$U_\mathcal{A} = \sum_{(x,y)\in\mathcal{XY}} P(x,y)\left[U_A(\mathcal{C}(\mathcal{A}(x)),y) - W(x,\mathcal{A}(x))\right] \quad (2)$$

THEOREM 2.1. *Consider a classification game with a binary cost model for* ADVERSARY, *i.e., given a pair of instances x and x',* ADVERSARY *can either change x to x' (incurring a unit cost) or it cannot (the cost is infinite). This game always has a Nash equilibrium, which can be found in time polynomial in the number of instances.*

Cost-Sensitive Learning

- Naive Bayes: $$P(y|x) = \frac{P(y)}{P(x)}P(x|y) = \frac{P(y)}{P(x)}\prod_{i=1}^{n} P(x_i|y) \quad (3)$$

- conditional utility: $$U(y_C|x) = \sum_{y \in \mathcal{Y}} P(y|x)U_C(y_C, y) \quad (4)$$

- Use greedy forward selection to select features

Adversary Strategy

ASSUMPTION 1. ***Complete Information:*** *Both* CLASSIFIER *and* ADVERSARY *know all the relevant parameters:* V_i, U_C, W_i, U_A *and the naive Bayes model learned by* CLASSIFIER *on* $\mathcal{S}$ *(including* $\mathcal{X}_C$*,* $P(y)$*, and* $P(x_i|y)$ *for each feature and class).*

ASSUMPTION 2. ADVERSARY *assumes that* CLASSIFIER *is unaware of its presence (i.e.,* ADVERSARY *assumes that* $\mathcal{C}(x)$ *is the naive Bayes model described in the previous section).*

Adversary Strategy

$$\log \frac{P(+|x)}{P(-|x)} = \log \frac{P(+)}{P(-)} + \sum_{x_i \in \mathcal{X}_C} \log \frac{P(x_i|+)}{P(x_i|-)} \qquad (5)$$

$$LO_C(x) = \log \frac{P(+|x)}{P(-|x)} \qquad LO_C(x_i) = \log \frac{P(x_i|+)}{P(x_i|-)}$$

- Naive Bayes classifies an instance as positive if

$$\frac{P(+|x)}{P(-|x)} > \frac{U_C(-,-) - U_C(+,-)}{U_C(+,+) - U_C(-,+)} \qquad (6)$$

$$LT(U_C) = \log \frac{U_C(-,-) - U_C(+,-)}{U_C(+,+) - U_C(-,+)} \quad gap(x) = LO_C(x) - LT(U_C)$$

- Adversary gain a utility of $\Delta U_A = U_A(-,+) - U_A(+,+)$

Adversary Strategy

- $\delta_{i,x_i'}$: 1 if the feature X_i is changed from x_i to x_i', and 0 otherwise
- *gain* in Adversary's objective of making the instance negative: $\Delta LO_{i,x_i'} = LO_C(x_i) - LO_C(x_i')$
- integer (binary) linear program (NP-hard, can be reduced from 0-1 knapsack problem):

$$\min \left\{ \sum_{X_i \in \mathcal{X}_C} \sum_{x_i' \in \mathcal{X}_i} W_i(x_i, x_i') \delta_{i,x_i'} \right\} \quad \text{s.t.}$$

$$\sum_{X_i \in \mathcal{X}_C} \sum_{x_i' \in \mathcal{X}_i} \Delta LO_{i,x_i'} \delta_{i,x_i'} \geq gap(x)$$

$$\delta_{i,x_i'} \in \{0,1\}, \quad \sum_{x_i' \in \mathcal{X}_i} \delta_{i,x_i'} \leq 1$$

- minimum cost camouflage (MCC) of x
- Adversary startegy:

$$\mathcal{A}(x) = \begin{cases} MCC(x) & \text{if } NB(x) = +,\ W(x, MCC(x)) < \Delta U_A \\ x & \text{otherwise} \end{cases}$$

Adversary Strategy

- A pseudo-linear time ($O(W \sum_i ||\mathcal{X}_i)$) algorithm can be obtained by discretizing LO_C

Algorithm 1 FINDMCC(i,w)

if $w \leq 0$ **then**
  return $(0, \{\})$
end if
if $i = 0$ **then**
  return $(\infty, Undefined)$
end if
$MinCost \leftarrow \infty$
$MinList \leftarrow Undefined$
for $x'_i \in \mathcal{X}_i$ **do**
  if $\Delta LO_{i,x'_i} \geq 0$ **then**
    $(CurCost, CurList) \leftarrow$FINDMCC$(i-1, w - \Delta LO_{i,x'_i})$
    $CurCost \leftarrow CurCost + W_i(x_i, x'_i)$.
    $CurList \leftarrow CurList + (i, x'_i)$.
    if $CurCost < MinCost$ **then**
      $MinCost \leftarrow CurCost$
      $MinList \leftarrow CurList$
    end if
  end if
end for
return $(MinCost, MinList)$

Algorithm 2 $\mathcal{A}(x)$

$W \leftarrow gap(x)$ (discretized).
$(MinCost, MinList) \leftarrow$FINDMCC$(n, W)$
if $NB(x) =$ **+ and** $MinCost < \Delta U_A$ **then**
  $newx \leftarrow x$
  for all $(i, x'_i) \in MinList$ **do**
    $newx_i \leftarrow x'_i$
  end for
  return $newx$
else
  return x
end if

Adversary Strategy

- 1st pruning rule:

LEMMA 4.1. *If*

$$\max_{i,x_i'} \left(\frac{\Delta LO_{i,x_i'}}{W_i(x_i, x_i')} \right) < \frac{gap(x)}{\Delta U_A}$$

then $\mathcal{A}(x) = x$.

- 2nd pruning rule: sort all the (i, x_i') tuples in increasing order of $W_i(x_i, x_i') \geq 0$. For identical values of $W_i(x_i, x_i')$, use decreasing order of $\Delta LO_{i,x_i'}$ as the secondary key and use the first entry in the list.

Classifier Strategy

- Assumptions:

ASSUMPTION 3. CLASSIFIER *assumes that* ADVERSARY *uses its optimal strategy to modify test instances (Algorithm 2).*

ASSUMPTION 4. *The training set* $\mathcal{S}$ *used for learning the initial naive Bayes classifier is not tampered with by* ADVERSARY *(i.e.,* $\mathcal{S}$ *is drawn from the real distribution of adversarial and non-adversarial data).*

ASSUMPTION 5. $\forall X_i \in \mathcal{X}$, $W_i(x_i, x_i')$ *is a semi-metric, i.e., it has the following properties:*

1. $W_i(x_i, x_i') \geq 0$ *and the equality holds iff* $x_i = x_i'$
2. $W_i(x_i, x_i'') \leq W_i(x_i, x_i') + W_i(x_i', x_i'')$

- (Implies $W(x, x'') \leq W(x, x') + W(x', x'')$)

Classifier Strategy

- The probability of observing an instance x':

$$P_A(x'|+) = \sum_{x \in \mathcal{X}} P(x|+) P_A(x'|x,+) \qquad (8)$$

$$P_A(x'|+) = \sum_{x \in \mathcal{X}_A(x')} P(x|+) \qquad (9)$$

where $\mathcal{X}_A(x') = \{x : x' = \mathcal{A}(x)\}$

$$P_A(x'|+) = \left(\sum_{x \in \mathcal{X}'_A(x')} P(x|+) \right) + I(x') P(x'|+) \qquad (10)$$

where $\mathcal{X}'_A(x') = \{x : x' = \mathcal{A}(x)\} \backslash \{x'\}, I(x') = 1$ if $NB(x') = -$ or $W\left(x', MCC(x')\right) \geq \Delta U_A$, and $I(x') = 0$ otherwise

Classifier Strategy

Algorithm 3 $\mathcal{C}(x')$

$P^{-}_{x'} \leftarrow \hat{P}(-) \prod_i \hat{P}(X_i = x'_i|-)$
$P^{+}_{x'} \leftarrow \hat{P}(+)\hat{P}_A(x'|+)$
$U(+|x') \leftarrow P^{+}_{x'}\, U_C(+,+) + P^{-}_{x'}\, U_C(-,+)$
$U(-|x') \leftarrow P^{+}_{x'}\, U_C(-,+) + P^{-}_{x'}\, U_C(-,-)$
if $U(+|x') > U(-|x')$ **then**
  return +
else
  return -
end if

- Solution to compute $\sum_{x \in X'_A(x')} P(x|+)$: iterate through all possible positive examples and check if x' is their minimum cost camouflage (need pruning!)

Classifier Strategy – pruning rules

LEMMA 5.1. *Let x_A be any positive instance and let $x' = MCC(x_A)$. Then, $\forall i$,*

$$(x_A)_i \neq x'_i \Rightarrow gap(x') + LO_C((x_A)_i) - LO_C(x'_i) > 0$$

THEOREM 5.2. *Let x_A be a positive instance such that $x' = MCC(x_A)$. Let $\mathcal{D}$ be the set of features that are changed in x_A to obtain x'. Let $\mathcal{E}$ be a non-trivial subset of $\mathcal{D}$, and let x'_A be an instance that matches x' for all features in $\mathcal{E}$ and x_A for all others, i.e., $(x'_A)_i = x'_i$ if $X_i \in \mathcal{E}$, $(x'_A)_i = (x_A)_i$ otherwise. Then $x' = MCC(x'_A)$.*

COROLLARY 5.3. *Let FV be the set of feature-value pairs that satisfy Lemma 5.1. Let $GV \subset FV$ be such that $(i, x_i) \in GV$ if $x'_{[i=x_i]} \in \mathcal{X}'_A(x')$. Then for every $x_A \in \mathcal{X}'_A(x')$, the set of feature-value pairs where x_A and x' differ form a subset of GV.*

THEOREM 5.4. *Let x' be any instance and let GV be the set defined in Corollary 5.3. Let $G = \{i | \exists x_i (i, x_i) \in GV\}$ and let $\mathcal{X}_i^G = \{x_i \in \mathcal{X}_i | (i, x_i) \in GV\}$. Then*

$$\sum_{(i,x_i)\in GV} P(x'_{[i=x_i]}|+) \leq \sum_{x\in\mathcal{X}'_A(x')} P(x|+) \leq$$

$$\left\{ \prod_{i\in G} \left[1 + \sum_{x_i \in \mathcal{X}_i^G} \frac{P(x'_{[i=x_i]}|+)}{P(x'|+)} \right] - 1 \right\}$$

Experiments – one round

	(+, +)	(+, −)	(−, +)	(−, −)
U_A	0	0	20	0
U_C	1	−10/−100/−1000	−1	1

Table 2: Utility matrices for Adversary and Classifier used in the experiments.

$U_C(+,-)$	10		100		1000	
Classifier	FN	FP	FN	FP	FN	FP
NB-PLAIN	94	2	124	1	165	1
NB-AW	481	2	481	1	481	1
AC-AW	93	0	123	0	164	0
NB-AL	477	2	477	1	477	1
AC-AL	94	0	124	0	165	0
NB-SYN	408	2	413	1	414	1
AC-SYN	164	1	196	0	229	0

Table 3: False positives and false negatives for naive Bayes and the adversary-aware classifier on the Ling-Spam dataset. The total number of positives in this dataset is 481, and the total number of negatives is 2412.

Experiments – one round

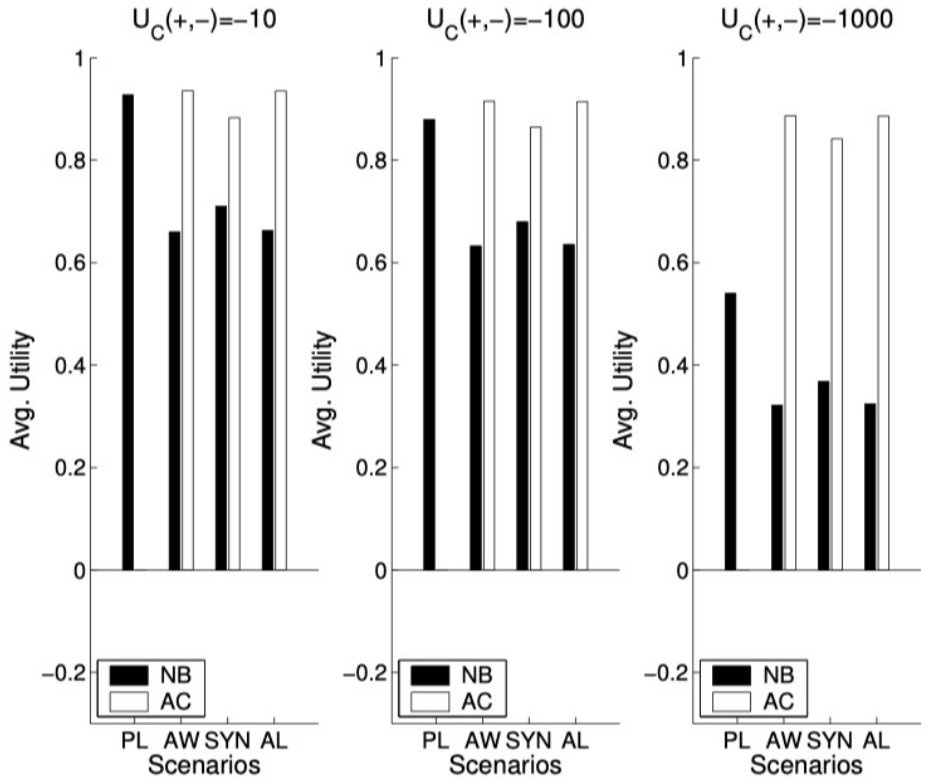

Figure 1: Utility results on the Ling-Spam dataset for different values of $U_C(+,-)$.

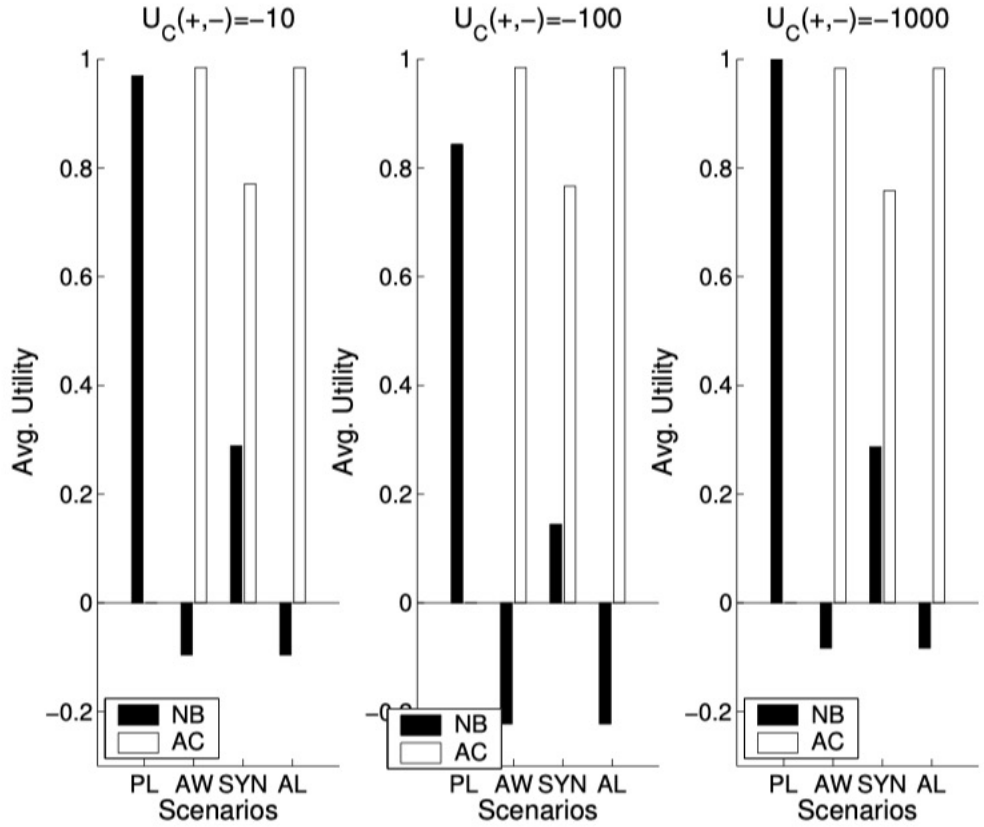

Figure 2: Utility results on the Email-Data set for different values of $U_C(+,-)$.

Experiments– repeated game

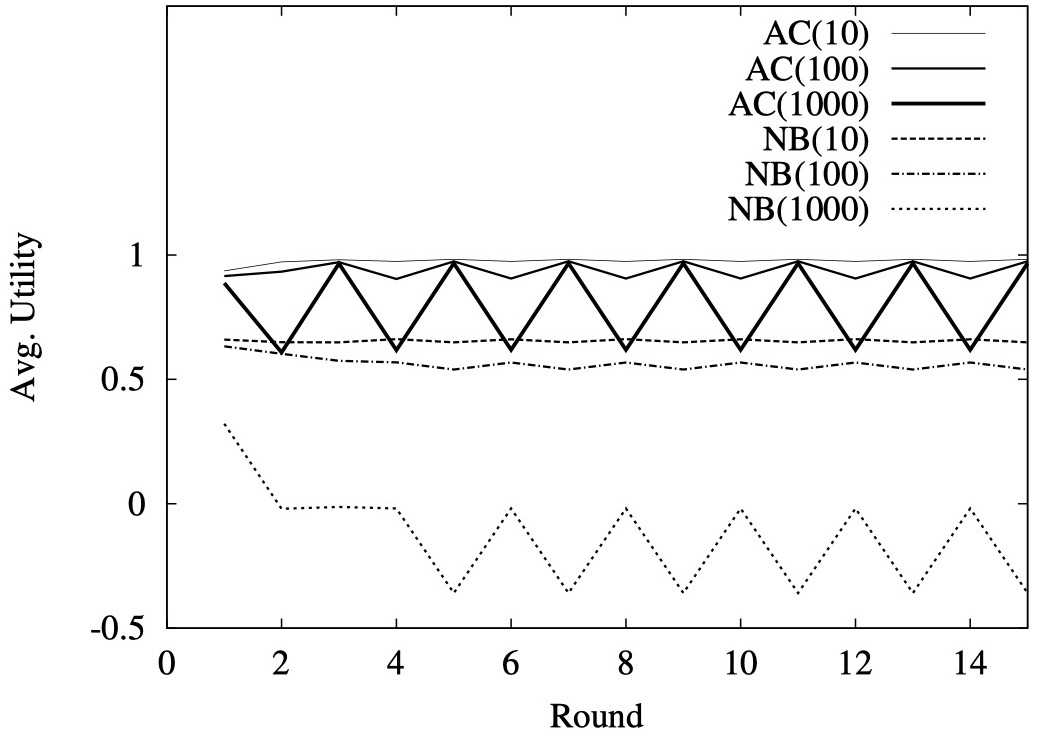

Figure 3: Utility of naive Bayes and the adversarial classifier for a repeated game in the AW scenario and Ling-Spam dataset. The number in parentheses is U_C(+,−).

Future Work

- The general existence and form of Nash or other equilibria in adversarial classification;
- Limitation: single-shot version of the adversarial classification game: one move by each of the players;
- Experiments on spam testbed lack feature measurement costs

Discussion