Some Submodular Data-Poisoning Attacks on Machine Learners

Shike Mei and Xiaojin Zhu
Department of Computer Sciences
University of Wisconsin–Madison
jerryzhu@cs.wisc.edu

March 1, 2015

Abstract

We study data-poisoning attacks using a machine teaching framework. For a family of NP-hard attack problems we pose them as submodular function maximization, thereby inheriting efficient greedy algorithms with theoretical guarantees. We demonstrate some attacks with experiments.

1 Introduction

The security community has long recognized the threats of *data-poisoning attacks* (a.k.a. causative attacks) on machine learning systems [1–6, 9, 10, 12, 16], where an attacker modifies the training data, so that the learning algorithm arrives at a "wrong" model that is useful to the attacker. To quantify the capacity and limits of such attacks, we need to know first how the attacker may modify the training data to maximally manipulate the learned model while incurring minimal effort. Recently, we proposed a unified attack framework [14, 15] to compute the optimal data-poisoning attacks on machine learners. The framework accommodates a wide range of attack effectiveness measures, attacker effort measures, and victim machine learning algorithms. It is based on machine teaching [17], which creates an optimal training data to make the learner learn the model in the teacher's mind.

The solutions for optimal attacks can be difficult (e.g. NP-hard in Section 3) depending on the attack setting. Our contribution here is to pose a large family of NP-hard attack problems into submodular function maximization, thereby inheriting efficient greedy algorithms with theoretical guarantees. We show the solution and its application with examples.

2 The Machine Teaching Framework Characterizes Data-Poisoning Attacks

We first review the machine teaching framework. In general, let $\hat{\theta}_D$ denote the model that the machine learning algorithm learns from training data $\mathcal{D}$. For example, if we denote the original training data $\mathcal{D}_0$, then without attacks the machine learner would have learned a model $\hat{\theta}_{D_0}$. Note that, during an attack $\mathcal{D}$ is not necessarily an *iid* sample drawn from some underlying $p(x, y)$ distribution, but this does not prevent the learner from applying the learning algorithm to $\mathcal{D}$ and obtain some model $\hat{\theta}_D$.

The attacker has full knowledge of the learning algorithm. The attacker carries out an attack by manipulating the training data from $\mathcal{D}_0$ to some set $\mathcal{D}$. For instance, in the context of classification the attacker may add a number of foreign items (x', y') to $\mathcal{D}_0$, remove some existing items (x_i, y_i) from $\mathcal{D}_0$, or change the value of x_i's and y_i's. In general, we use $\mathbb{D}$ to denote the (often infinite) set of allowable training sets that the attacker can use.

The attacker wants to change the model that the learner learns. In "attractive attacks," the attacker has a specific target model θ^* in mind. For example, an attacker may want $\theta^*_k \approx -\theta_{D_0,k}$ for certain feature dimension k (e.g., whether a vaccine was administrated or not) to make the k-th feature ostensibly having the opposite correlation with the outcome variable, while keeping other dimensions unchanged: $\theta^*_j \approx \theta_{D_0,j}$ for $j \neq k$. Hence the attacker may mislead a domain expert who attempts to interpret the learned model. For such attractive attacks we may define an *attack loss function* $\text{loss}(\mathcal{D}) = \|\hat{\theta}_D - \theta^*\|$ with an appropriate norm. In "repulsive attacks," the attacker does not want a specific target model but simply wants $\hat{\theta}_D$ to be as far away from $\hat{\theta}_{D_0}$ as possible. This can be captured by a different (non-convex) loss function $\text{loss}(\mathcal{D}) = -\|\hat{\theta}_D - \hat{\theta}_{D_0}\|$ on parameter values, or by $\text{loss}(\mathcal{D}) = \int \mathbb{1}_{f(x|\hat{\theta}_D)=f(x|\hat{\theta}_{D_0})} p(x)dx$ on future classification agreement, where $f(x \mid \theta)$ is the predicted label on test item x under model θ, and $p(x)$ is the marginal distribution of test items. In general, the definition of $\text{loss}(\mathcal{D})$ is problem-specific and must be defined carefully by researchers who study such attacks.

The attacker also wants to minimize the risk of being detected. More manipulation on the training data implies a higher risk of detection. The amount of manipulation and the associated risk can be characterized by an *attack effort function* $\text{effort}(\mathcal{D})$. For example, $\text{effort}(\mathcal{D}) = |\mathcal{D}\Delta\mathcal{D}_0|$ the cardinality of the symmetric difference to the original training set. Again, the definition of $\text{effort}(\mathcal{D})$ is problem-specific. We will see several examples in later sections.

In summary, the attacker would like to maximize its attack effect and minimize effort while being constrained to search for an attacking training set within $\mathbb{D}$. We use the following optimization-based framework for optimal attacks on machine learning:

$$\min_{\mathcal{D}\in\mathbb{D}} \quad \text{loss}(\mathcal{D}) + \text{effort}(\mathcal{D}). \tag{1}$$

Obviously, our framework needs to be instantiated for a specific learner (implicit

in $\hat{\theta}_D$), attacking manipulations $\mathbb{D}$, loss() and effort() functions. Not surprisingly, the difficulty in solving (1) depends critically on these attack settings. The rest of the paper is devoted to one attack setting corresponding to certain NP-hard combinatorial optimization problems. We will show how to convert the optimization problem (1) into submodular function maximization and hence greedy algorithms with guarantee.

3 Submodular Attacks

We consider attack settings that result in a submodular function maximization problem. Specifically, the minimization of loss() is equivalent to maximizing some submodular function, while effort() contains a submodular cost function. The main attraction of submodularity is efficient greedy procedures with theoretical guarantees.

First, we briefly review submodular function maximization [8]. Denote a submodular function as $f : 2^V \to \mathbb{R}$ where V is a finite set. For every $A, B \subseteq V$

$$f(A \cap B) + f(A \cup B) \leq f(A) + f(B).$$

In addition, f is monotone if for every $A \subseteq B \subseteq V$, $f(A) \leq f(B)$. Let c_i be the cost associated with selecting the i-th element in V, and L the total budget. Then the problem of submodular function maximization subject to budget constraint is:

$$\max_{S \subseteq V} \quad f(S) \quad \text{s.t.} \sum_{i \in S} c_i \leq L. \tag{2}$$

This problem is in general NP-hard [7]. However, there exists efficient greedy algorithms with $(1 - \frac{1}{\sqrt{e}})$ approximation guarantee [11]: Let the greedy solution be $S' \subseteq V$, then $f(S') \geq \left(1 - \frac{1}{\sqrt{e}}\right) \max_{S:\sum_{i \in S} c_i \leq L} f(S)$. The greedy algorithm is as follows: we start with $S_0 = \emptyset$ and add the best element

$$\text{argmax}_{x \in V} \frac{f(\{x\} \cup S_{i-1}) - f(S_{i-1})}{c_x} \tag{3}$$

to S_{i-1} in iteration i.

The key, then, is to identify attack settings that can be formulated as submodular functions. Below, we exhibit two such attack settings with distinct attack manipulations.

3.1 Flipping-Label Attacks

Flipping-label attack is a family of attacks applicable to most binary classification learners. The attack flips the label of selected training items, hence the name. In standard binary classification, the original training data $\mathcal{D}_0$ contains n feature $x \in \mathcal{X}$ and label $y \in \mathcal{Y} = \{-1, 1\}$ pairs, that is $\mathcal{D}_0 = \{(x_{0i}, y_{0i})\}_{i=1}^n$.

We focus on the case where the learned model is a discriminative function $\hat{h}_{\mathcal{D}_0} : \mathcal{X} \to \mathbb{R}$ and $\text{sign}(\hat{h}_{\mathcal{D}_0}(x))$ predicts the label of x.

We consider attractive attacks where the attacker manipulates data into $\mathcal{D}$ in order to guide the learned model $\hat{h}_{\mathcal{D}}$ to some target model h^*. The attacker can only "flip" some training labels. Therefore, the manipulated training data $\mathcal{D}$ will be in the search space $\mathbb{D} = \{\{(x_{0i}, y_i)\}_{i=1}^n | y_i \in \mathcal{Y}\}$. Further assume that the i-th training item has a cost c_i representing the effort the attacker must take to flip the i-th label (or, alternatively, the risk of detection). We assume the costs are additive, though it can be generalized to a submodular function, too.

Given training data $\mathcal{D}$, a large family of machine learning algorithms learn by regularized risk minimization:

$$\hat{h}_{\mathcal{D}} = \text{argmin}_{h \in \mathcal{H}} \frac{1}{n} \sum_i \phi(y_i, h(x_{0i})) + \Omega(h) \tag{4}$$

where $\mathcal{H}$ is the hypothesis space considered by the learner, and $\Omega()$ a regularizer. $\phi(y, h(x)) : \mathcal{Y} \times \mathbb{R} \to \mathbb{R}_{\geq 0}$ is the surrogate loss. For example, in support vector machines (SVM) it is the hinge loss $\phi(y, h(x)) = \max(0, 1 - yh(x))$; in logistic regression (LR) it is the logistic loss $\phi(y, h(x)) = \log(1 + \exp(-yh(x)))$.

Accordingly, given the attack target h^* an attacker may define the loss function as:[1]

$$\text{loss}(D) \quad \triangleq \quad \sum_i \phi(y_i, h^*(x_{0i})), \tag{5}$$

where we ignored the regularizer since it is not a function of D. To build the connection to submodular function minimization, we denote the set of flipped items by $S = \{i \mid y_i \neq y_{0i}\} \subseteq \{1, 2, ..., n\}$ and rewrite the loss as a function of S:

$$\begin{aligned} \text{loss}(S) \quad &= \quad \sum_{i \in S} [\phi(y_i, h^*(x_{0i})) - \phi(y_{0i}, h^*(x_{0i}))] \\ &\quad + \sum_{i=1}^n \phi(y_{0i}, h^*(x_{0i})). \end{aligned} \tag{6}$$

The last term is a constant with respect to S and can be ignored. We define the loss decrease $[\phi(y_{0i}, h^*(x_{0i})) - \phi(-y_{0i}, h^*(x_{0i}))]$ as the gain of flipping a label (denoted as g_i),

$$g_i \quad \triangleq \quad \phi(y_{0i}, h^*(x_{0i})) - \phi(-y_{0i}, h^*(x_{0i})) \tag{7}$$

We define the attack effort to be the (infinite) indicator function that the sum of costs on flipped items cannot exceed the total budget L:

$$\text{effort}(S) = \mathcal{I}(\sum_{i \in S} c_i \leq L) \tag{8}$$

[1]This loss function does not guarantee that the victim learner will arrive at h^*, but is merely an encouragement toward h^*.

Putting things together, the optimal attack problem (1) for flipping-label attack can be written equivalently as

$$\max_S \sum_{i \in S} g_i \quad \text{s.t. } \sum_{i \in S} c_i \leq L. \tag{9}$$

It is precisely the budgeted submodular maximization problem in (2). After discarding the items with negative gain g_i (these items can never be in the optimal solution), the objective function is also monotone. In fact, it is the Knapsack problem. By applying the results for submodular maximization, we can solve (9) with the greedy algorithm with approximation guarantee $(1 - \frac{1}{\sqrt{e}})$ [11]. We point out that the flipping-label attack enjoys the greedy algorithm and the guarantee even when the surrogate loss $\phi()$ is not convex, thus the flipping label attack can be inflicted on a large number of binary classifiers.

3.2 Pool-Based Adding-Data Attack on 1NN

We now present another attack setting where submodularity is useful. In a pool-based adding-data attack, the learner is also a binary classification algorithm. Unlike in flipping-label attacks, the attacker has a separate pool of *candidate attack items* $V = \{z_1, \ldots, z_m\}$ (e.g. a collection of pre-selected normal-looking documents) in addition to the training data. In reality, the attacker may only aim to increase either false positive or false negative. Therefore, without loss of generality, all items in V are assumed to be labeled as positive. The attacker can choose a subset $S \subseteq V$, and make $D = (S, 1) \bigcup \mathcal{D}_0$ the manipulated training set. Therefore, $\mathbb{D}$ consists of 2^m such augmented training data sets.

We demonstrate repulsive attacks here, where the attacker wants the model trained with $\mathcal{D}$ to be maximally different from the model trained with $\mathcal{D}_0$. We assume that the attacker knows the marginal distribution $p(x)$ from which future test items will be drawn. This assumption can be easily replaced with a separate validation set drawn from $p(x)$, and the changes to the following algorithm is straightforward. Let $R(S)$ be the set of points labeled differently by $\hat{h}_{\mathcal{D}}(x)$ and $\hat{h}_{\mathcal{D}_0}(x)$:

$$R(S) \quad = \quad \{x \in \mathcal{X} : \hat{h}_{\mathcal{D}_0 \cup S}(x) \neq \hat{h}_{\mathcal{D}_0}(x)\}. \tag{10}$$

Then the attacker may define the loss as

$$\text{loss}(S) = 1 - p(R(S)) = 1 - \int_{x \in R(S)} p(x)dx. \tag{11}$$

The attacker may use the same effort function as in flipping-label attack, namely $\text{effort}(S) = \mathcal{I}(\sum_{z_i \in S} c_i \leq L)$. The optimal attack problem (1) can be equivalently written an

$$\max_{S \subseteq V} \quad \int_{x \in R(S)} p(x)dx \quad \text{s.t. } \mathcal{I}(\sum_{z_i \in S} c_i \leq L).$$

We now prove that when the learner is an 1-Nearest-Neighbor (1NN) classifier, (12) is a budget constrained maximum coverage problem and a special case of budgeted submodular maximization [7]. To see this, let $\mathcal{D}_0^-$ and $\mathcal{D}_0^+$ be the subsets of $\mathcal{D}_0$ with negative and positive labels, respectively. Define the distance between a point x and a set Z as $d(x, Z) = \min_{z \in Z} \|x - z\|$. We show that $R(S) = \bigcup_{z_i \in S} R(\{z_i\})$.

$$\begin{aligned} R(S) &= \{x \mid d(x,S) < d(x,D_0^-) < d(x,D_0^+)\} \\ &= \{x \mid \min_{z_i \in S} d(x,\{z_i\}) < d(x,D_0^-) < d(x,D_0^+)\} \\ &= \left\{ x \mid \bigvee_{z_i \in S} \left(d(x,\{z_i\}) < d(x,D_0^-) < d(x,D_0^+) \right) \right\} \\ &= \bigcup_{z_i \in S} \{x \mid d(x,\{z_i\}) < d(x,D_0^-) < d(x,D_0^+)\} \\ &= \bigcup_{z_i \in S} R(\{z_i\}). \end{aligned}$$

We formulate the objective in (12) as a coverage function:

$$\int_{x \in R(S)} p(x)dx = \int_{x \in \bigcup_{z_i \in S} R(\{z_i\})} p(x)dx. \tag{12}$$

Therefore, the objective is the coverage of the set $R(S)$ and is submodular and monotone [7]. Following results in submodular maximization [11], the same algorithm used in flipping-label attack can constructs the attack set S' with a $1 - \frac{1}{\sqrt{e}}$ guarantee.

4 Experiments

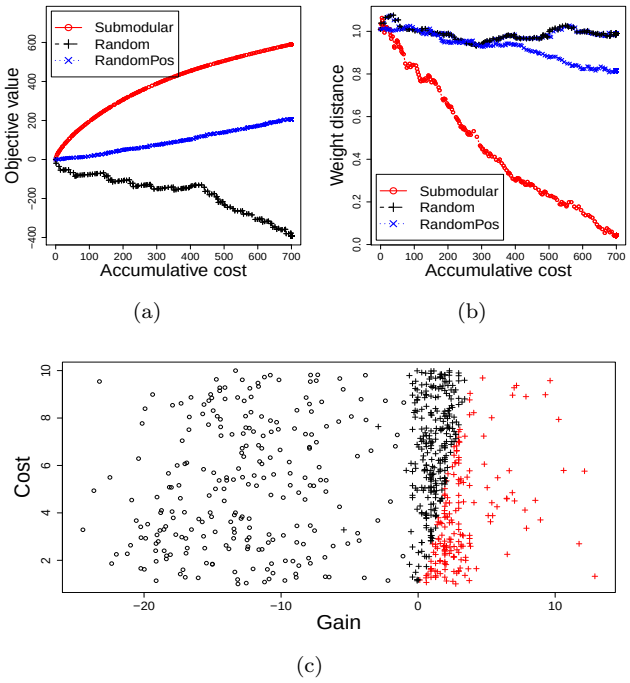
4.1 Flipping-Label Attacks

(Attack Setting) We use the Breast Cancer Wisconsin data set. [2] It contains 699 data points. Each data point consists of 9 numeric features and a binary label (benign vs. malignant).

We simulate two separate attacks, one on logistic regression (LR), the other on support vector machine (SVM). We choose LR and SVM because their learned weights $\hat{\mathbf{w}}_D$ are often interpreted by domain experts for data exploration and decision making, and hence an attack that changes the weights could have adversarial consequences. For this particular data set, a positive weight indicates a positive correlation between the feature and a benign outcome.

For the sake of demonstration, we arbitrarily selected the feature "Normal Nucleoli" and assume that the attacker wants to changes its weight in the learned model. The weight of this feature, denoted as $\hat{w}_{\mathcal{D}_0,NN}$ (obtained by LR or

[2] `http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Original)`



(a) (b)

(c)

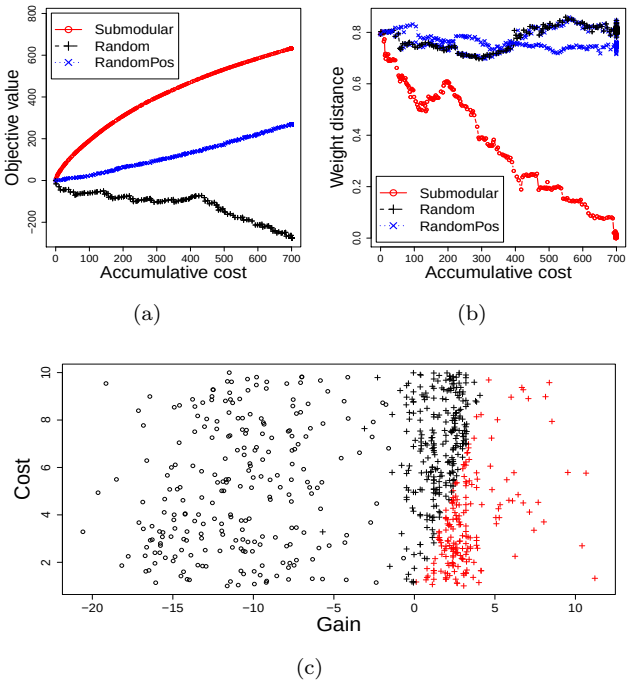
Figure 1: Flipping-label attack on logistic regression. (a) Objective value as attack progresses; (b) Distance to attack target $|\hat{w}_{D,NN} - w^*_{NN}|$; (c) Flipping of training points, +: original label y_{0i} positive, o: original label y_{0i} negative, red: selected to flip label, black: not selected.

SVM, respectively), is originally positive. The attacker wants the target weight on Normal Nucleoli to have the opposite sign: $w^*_{NN} = -\hat{w}_{\mathcal{D}_0,NN}$, while the weights on the other dimensions unchanged: $w^*_j = \hat{w}_{\mathcal{D}_0,j}$ for $j \neq NN$.

Since this is a proof-of-concept simulation, we generate a random cost for flipping each item's label. Specifically, the costs c_i are sampled from a uniform distribution on $[1, 10]$. The total budget L in Eq (9) is set generously at 700.

We are not aware of prior work on flipping-label attacks. As such, we only compare our optimal attack framework (called *submodular attack*) against two naive baselines. The first baseline *random attack* randomly selects training items to flip the label subject to the budget constraint. The second baseline *randomPos attack* differs from random attack in that it only selects positive-gain items.

(Results) First, we show the objective value defined in Eq (9) of attacking LR and SVM in Figures 1(a) and 2(a). The x-axis is the accumulative cost over the items selected to flip labels as each algorithm progresses. Our submodular attack increased the objective value rapidly. In contrast, random attack even decreased the objective value, and randomPos attack increased it only slowly.



(a) (b)

(c)

Figure 2: Flipping-label attack on SVM

Therefore, submodular attack maximizes the objective much more effectively than the two baselines.

We also show the distance between the learned model weight (over attacked training data) and the target weight for the feature "Normal Nucleoli" $|\hat{w}_{D,NN} - w^*_{NN}|$ for LR and SVM in Figures 1(b) and 2(b), respectively. Submodular attack decreased this distance much more rapidly than random and randomPos attacks. For other weight dimensions, we report that the signs of all weights remain the same and the change is not larger than 0.2. Therefore, submodular attack successfully mislead the LR and SVM learners to the target model by flipping the label of well-selected items.

We show the selection of flipping items in Figures 1(c) and 2(c) for attacking LR and SVM, respectively. We draw each training item as a point with coordinates (gain g_i,cost c_i). The gain was defined in Eq (7). Items originally labeled as positive/negative are marked as crosses/circles. Selected items are colored as red and the unselected items are colored as black. Submodular attack effectively utilizes the budget to maximize the total gain.

4.2 Pool-Based Adding-Data Attack on 1NN

(Attack Setting) We demonstrate pool-based adding-data attack on 1NN using two data sets: MNIST handwritten digits and hotel review documents.

We use the MNIST data[3] which contains gray-scale images with handwritten digits. We define a binary classification task on "1" (positive class) vs. "7" (negative class). MNIST already contains a training vs. validation split. We use the about 2000 validation points as our validation set. We further randomly split out 25% of the about 11000 training points as our attack candidate pool. We use Euclidean distance over gray-scale pixels for the distance between images.

We also use the HotelReview data set [13] which consists of hotel review articles and an overall rating (an integer from 1 to 5) associated with each review. The data set contains 5,000 reviews with 1,000 reviews for each rating. We merged the 2,000 reviews with ratings 1 and 2 as negatively labeled documents, and the 2,000 reviews with ratings 4 and 5 as positively labeled documents. The 1,000 reviews with rating 3 were discarded. We randomly sampled 25% of the resulting data set as the attack candidate pool. We further randomly split the remained documents as training data and validation data with a ratio of $7:3$. The distance between two documents x, x' is defined as $1 - \cos(x, x')$.

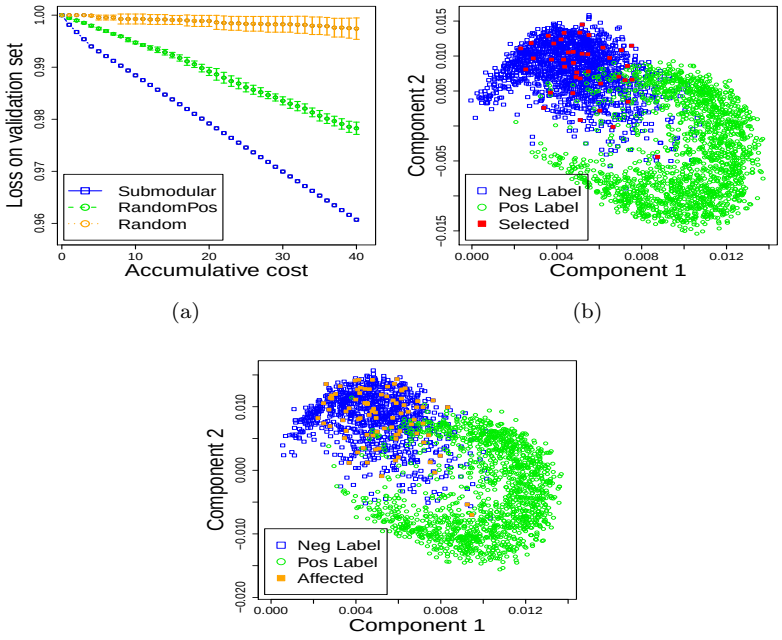
For simplicity, we set the cost for selecting any item in the candidate attack pool at a constant $c_i = 1$. We choose the total budget at $L = 40$. Given the constant cost, this budget means that we can at most choose 40 items from the attack pool. We refer to our pool-based adding-data attack algorithm as submodular attack again. For comparison, we implement two random baselines: *random attack* randomly selects an item from the attack pool, and *randomPos attack* randomly selects items which strictly decrease the loss (recall the loss defined in Eq (11) can be zero).

(Results on MNIST data) For the MNIST data set we show the loss (11), which the attacker attempts to minimize, in Figure 3(a). Submodular attack greatly misled the 1NN learner as the chosen attack items are added to the training set. The 40 attack items represents less than 1% of the MNIST training data. Submodular attack decreases the loss much more than the two baselines.

To shed light on the attack, we visualize the 40 selected attack items in Figure 4. Recall that in this particular attack, the attacker always claims any inserted items as in the positive class. Note that the selected attack items are all in fact digits "7" (negative class). Therefore, the attacker's strategy can be understood as selecting representative "7" images in the candidate pool (which contains a mixture of "1" and "7"), falsely claiming that these "7" images have a positive label, and adding them to the training set. As a result, the 1NN learner will be misled to incorrectly classify many neighbors of these "7" images as class "1."

To further visualize the attack, we run principal component analysis (PCA) on the whole 1 vs. 7 data set (the training data, the validation data and the attack candidate pool). We then project the data onto the first two principal directions. Figure 3(b) shows the attack candidate pool with their true labels. The selected attack items are marked in red. There is an interesting tradeoff: It is clear that the attacker should not select a "7" image in a low density region to falsely call class 1, because it may not have many "7" neighbors to

[3] `http://yann.lecun.com/exdb/mnist/`



(a) (b) (c)

Figure 3: Pool-Based Adding-Data Attack on 1NN with the MNIST data. (a) Loss as the attack progresses; (b) PCA projection of the attack candidate pool; (c) PCA projection of the validation set.

Figure 4: The 40 selected attack items in that order. All of them will be labeled by the attacker as in class "1" and be added to the training set to attack 1NN.

mislead 1NN much. But the attacker should not select a "7" image in a very high density region either, because the training set will likely already contain many nearby 7's with the correct label – again the "7" will not have a large effect on misleading 1NN. As a result, the actually selected items in Figure 3(b) seem to form a loose cover of the 7's. Figure 3(c) shows the validation set with their true labels. An "affected" point there is defined to be one who becomes misclassified after the attack. The affected points are also evenly distributed in the image space, which supports our analysis.

(Results on HotelReview data) The loss defined in Eq (11) is shown in Figure 5(a) at the attack progresses. Similar to MNIST, submodular attack greatly misled the 1NN learner with only small manipulations on the training data (40 items are about 2% of the hotelreview training data) and decreased the loss much more than the two baselines. We also visualize the documents. Unlike

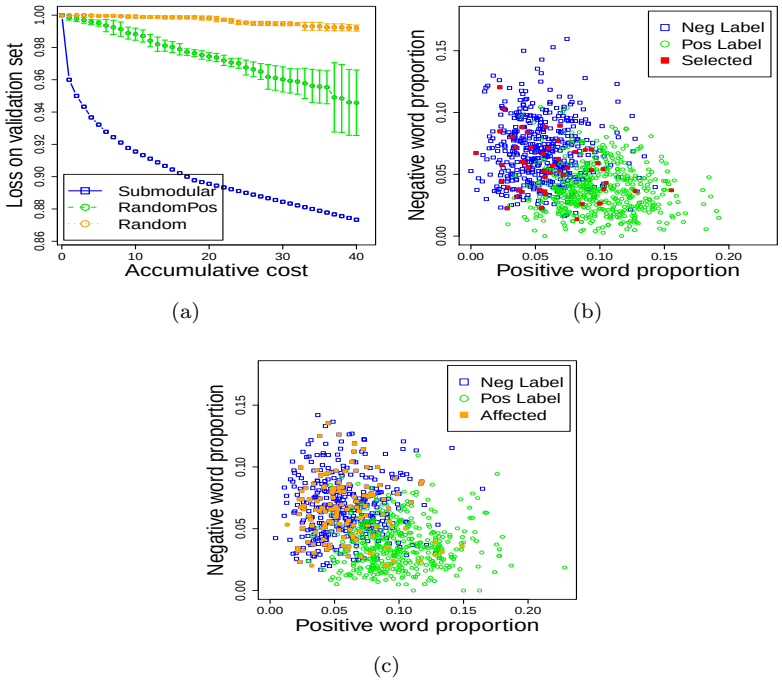


Figure 5: Pool-Based Adding-Data Attack on 1NN with the hotelreview data. (a) Loss as the attack progresses; (b) Sentiment projection of the attack candidate pool; (c) Sentiment projection of the validation set.

in MNIST, PCA does not provide a clear picture here. Instead, our intuition is that a review article's positive and negative sentiment word counts can reflect its opinion. We count both of them for each document. [4].

We then define the positive word proportion of a document as $\frac{\#positive\ words\ in\ document}{\#total\ words\ in\ document}$ and negative word proportion similarly. We project each document as a point to this 2D "sentiment space" with the proportions as coordinates in Figure 5(b) and Figure 5(c). Similar to the MNIST data, the attacker selects the truly negative labeled items, pretends them as positive label data, and adds them into the training data. The distribution of selection is also near evenly distributed.

We conclude that submodular attack effectively misleads the 1NN learner with very small amount of manipulation.

5 Discussions

We showed several examples of submodular attacks. These attacks can be readily generalized to other learners with a submodular objective. They can po-

[4]The positive and negative sentiment word lists are downloaded from `http://www.cs.uic.edu/~liub/FBS/opinion-lexicon-English.rar`

tentially be further generalized. For example, for attack settings that are not submodular, one can sometimes relax the problem into a submodular maximization problem, then our submodular attack applies on the relaxation. These extensions are left for future work.

Acknowledgments Support for this research was provided in part by NSF grant IIS 0953219 and by the University of Wisconsin–Madison Graduate School with funding from the Wisconsin Alumni Research Foundation.

References

[1] M Barreno, P.L. Bartlett, F.J. Chi, A.D. Joseph, B. Nelson, B.I.P. Rubinstein, U. Saini, and J.D. Tygar. Open problems in the security of learning. In *The First ACM Workshop on AISec*, 2008.

[2] M. Barreno, B. Nelson, A. D. Joseph, and J.D. Tygar. The security of machine learning. *Machine Learning Journal*, 81(2):121–148, 2010.

[3] M. Barreno, B. Nelson, R. Sears, A.D. Joseph, and J.D. Tygar. Can machine learning be secure? In *CCS*, 2006.

[4] B. Biggio, Giorgio Fumera, and Fabio Roli. Security evaluation of pattern classifiers under attack. *IEEE TKDE*, 2013.

[5] B. Biggio, B. Nelson, and P. Laskov. Poisoning attacks against support vector machines. In *ICML*, 2012.

[6] M.J. Kearns and M. Li. Learning in the presence of malicious errors. *SIAM J. Comput.*, 22(4):807–837, 1993.

[7] S. Khuller, A. Moss, and J. Naor. The budgeted maximum coverage problem. *Inf. Process. Lett.*, 70(1):39–45, 1999.

[8] Andreas Krause and Daniel Golovin. Submodular function maximization. In *Tractability: Practical Approaches to Hard Problems (to appear)*. Cambridge University Press, February 2014.

[9] P. Laskov and M. Kloft. A framework for quantitative security analysis of machine learning. In *The 2nd ACM Workshop on AISec*, 2009.

[10] P. Laskov and R. Lippmann. Machine learning in adversarial environments. *Machine Learning*, 81(2):115–119, 2010.

[11] Hui Lin and Jeff Bilmes. Multi-document summarization via budgeted maximization of submodular functions. In *NAACL HLT*, 2010.

[12] W. Liu and S. Chawla. A game theoretical model for adversarial learning. In *ICDM Workshops*, 2009.

[13] Shike Mei, Jun Zhu, and Xiaojin Zhu. Robust RegBayes: Selectively incorporating First-Order Logic domain knowledge into Bayesian models. In *ICML*, 2014.

[14] Shike Mei and Xiaojin Zhu. The security of latent dirichlet allocation. 2015.

[15] Shike Mei and Xiaojin Zhu. Using machine teaching to identify optimal training-set attacks on machine learners. 2015.

[16] Han Xiao, Huang Xiao, and Claudia Eckert. Adversarial label flips attack on support vector machines. In *ECAI*, 2012.

[17] Xiaojin Zhu. Machine teaching: an inverse problem to machine learning and an approach toward optimal education. In *The Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI "Blue Sky" Senior Member Presentation Track)*, 2015.