CS 763: Security and Privacy in Data Science **Fall 2019**

Presentation 3: Data Poisoning

Presenters: Somya Arora & Zi Wang *Scribes: Pierre Petrella & Miru Park*

3.1

How secure are our classifiers?

3.1.1 Introduction

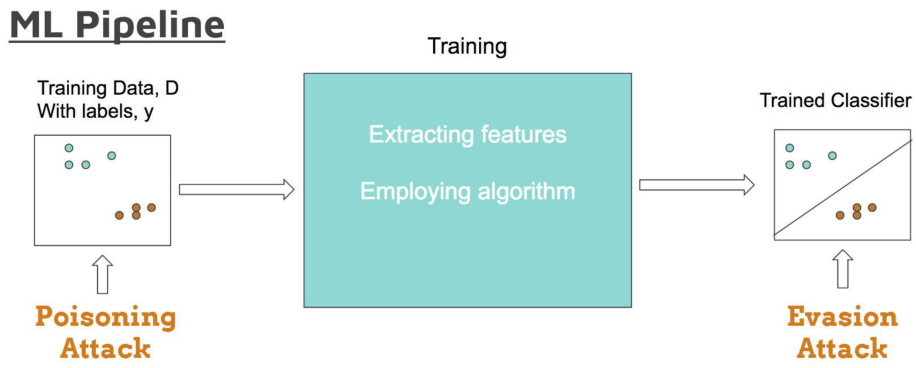
This is a summary based on the presentation given by Zi and Somya. They introduced a method with which supervised machine learning algorithms could be exploited by an "intelligent adversary". The method presented is known as poisoning attack. The name is derived from the fact that the adversary need only craft and inject one or a few malicious training examples to maximize the loss function of the machine learning algorithms in question, leading to higher classification error. The presentation was based on two reasonably recent papers: paper1 and paper2

3.1.2 potential threats of ML pipeline poisoning

Machine learning systems have made itself necessary in this new data era requiring data analysis that can only be managed by automated processes. It is more precisely used for image recognition from distinguishing cats from dogs, red lights from green lights to facial recognition.

Unfortunately these machine learning systems can be compromised. They are now becoming the weakest part of the security chain and consequently of the whole system. If no precautions are taken, this weakness can be used as a weapon by the attackers.

3.1.3 General attack on a ML Pipeline



3.2 Targeted clean labor poisoning attacks on Neutral Networks

3.2.1 Assumptions

The attacker has:

1. No knowledge of the training data
2. No control over the target instance during test time
3. No control over labelling of data for training
4. Knowledge of the model and it's parameters

These restriction are quite strict on the attacker. This implies that the data poisoning requires minimal intrusion which makes it difficult to detect for the ML model.

3.2.2 Properties

Clean labels

There are various types of poisoning attacks. In this paper, we are focusing on clean labels opposed to poisoning that involve tampering with the labels. clean labels allows to poison a training set with minimal intrusion as the poisoned image can simply be uploaded online and wait to be used by a ML model.

Targeted

This type of attack is built to affect one image specifically and not tamper with the other ones. This allows the poisoning to happen without the Users noticing that the model was tampered with. the Degradation of the model should be unnoticeable.

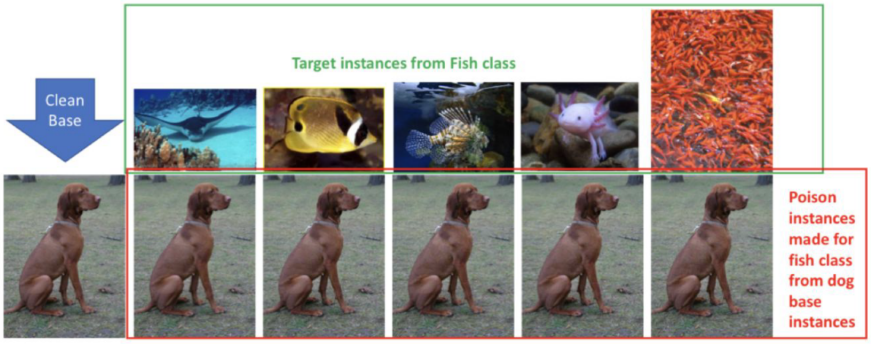
good success rate

If the poisoning is affecting the last more specific nodes of the ML model, we can assume that the poisoning will have a success rate of 100% most of the time.

3.2.3 poisoning Attacks example

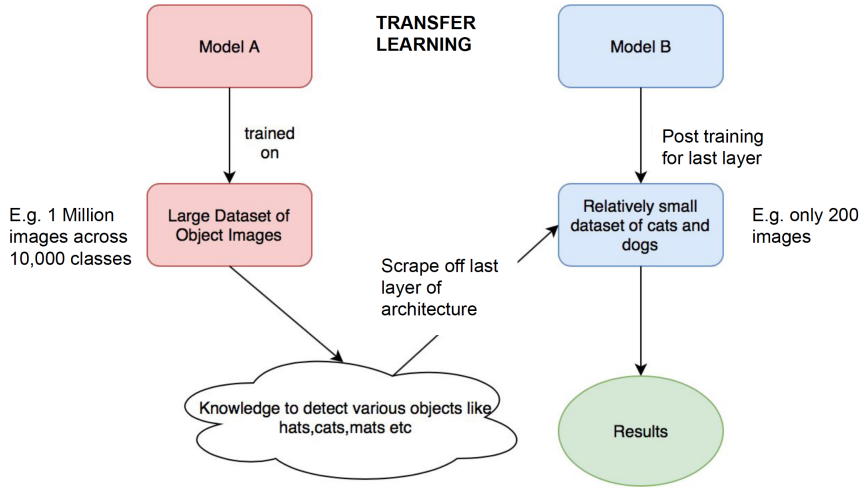
W e will be considering the Classifier that sorts between fish and dogs. The goal of the attacker will be to chose a specific picture of a fish and trick the ML model to think it is a dog. To do so he and insert a poisoned instance for fish class from dog a base instance. This poisoned picture which looks like a dog will be labeled accordingly and therefore trick the ML model.

Here are examples of poisoned dog images associated with the fish images they are targeting. We can see that the difference between the original and poisoned dog is indistinguishable to the human eye.

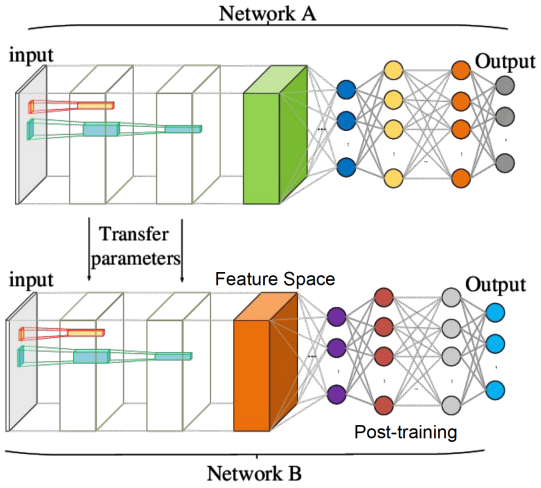


3.2.4 Transfer learning

On way to train a new ML model is to use an existing working model that has been trained on a significantly big data set (Model A) and get rid of the last couple of layers. We then reconstruct the last layers using a relatively small data set to fine tune the model to distinguish cats and dogs for instance (Model B).



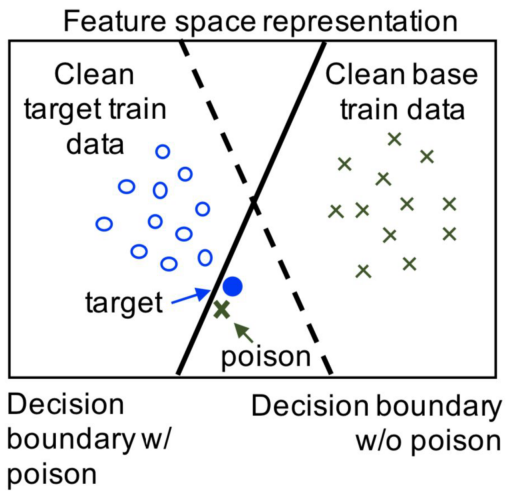
These last layers are very susceptible to change making it an idea target for data poisoning. Transfer learning is an effective way to train a new model without requiring all the resources necessary to train a ML model from scratch. Unfortunately if the data set used to train the last layer has been tampered with classification accuracy can be easily tampered with. Here is another representation of Transfer learning.



- Everything till penultimate layer is borrowed - static knowledge
- Knowledge is used to convert input to feature space : x to f(x)
- Post training is done on a smaller dataset - poison it!
- Attack happens in post training

3.2.5 Finding the poisoning

The poisoning image must look like a dog but must have all the qualities of it's target (the fish). This will move the decision boundary from the initial doted line to the full line (see diagram). Now any pictures close enough to the targets characteristics will be on the "dog" side of the decision line and therefore be seen as a dog instead of a fish.



In order to create such a poisonous image we must find an image to satisfy this equation with X the image we want to find, t the target, b the initial "dog" image that we will modify and beta the importance we give to the similarity between the original a poisonous image:

$$\mathbf{p} = \underset{\mathbf{x}}{\operatorname{argmin}} \ \|f(\mathbf{x}) - f(\mathbf{t})\|_2^2 + \beta \|\mathbf{x} - \mathbf{b}\|_2^2$$

Make the poison instance move towards the target instance in feature space

Make the poison appear like a base class instance to a human labeller

3.2.6 Approach

The protocol to follow to poison the data set is the following:

1. Choose a target instance to misclassify
2. Choose a base instance & make imperceptible changes to it to get a poison
3. Poison is created though an optimization based equation
4. Inject poison into training data and let model be trained on poisoned dataset

3.2.7 Algorithm

The algorithm to find the image consists of a forward backward iterative splitting procedure to find poison Iteratively. This is achieved by iterating these 2 steps:

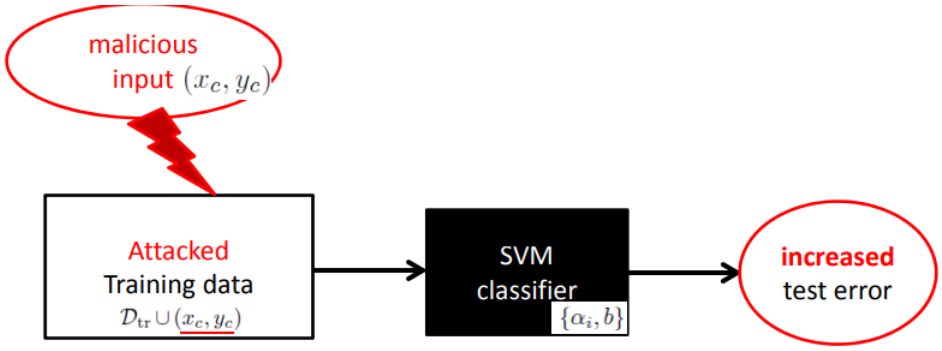
1. minimize distance to the target instance in feature space
2. minimize distance from the base instance in input space

3.3 Poisoning Attack Against Support Vector Machines(SVM)

3.3.1 Overview of Data Poisoning Against SVM

Poisoning attack against Support Vector Machines is an instance of causative attacks. Causative attack is an attack which takes advantage of many people's assumption that machine learning algorithms receive well-behaved data. However that is not so. An attacker that is sneaky enough can temper with optimal solutions to the Support Vector Machine by injecting a specific and well crafted attack example. Injection of such point into the training data is called Poisoning Attack.

The attacking scheme is rather simple (See picture below). The point (x_c, y_c) is the desired attack point that will enable the attacker to temper with optimal solutions to the SVM, D_{tr} is the training data, and $\{\alpha_i, b\}$ is the solution to the SVM. As can be seen in the figure, the attack leverages access to the training data. This is more of a realistic assumption than an exploratory attack during which the adversary has access to the model itself. One way to inject some malicious input would be to simply upload the malicious data online or create a set of fake accounts online.



3.3.2 Brief overview of optimization

Solving for optimal solutions to SVM can be achieved by mathematical optimization technique called quadratic programming. More precisely we solve a quadratic programming problem using Lagrange multipliers:

$$L_p = \frac{1}{2}||\beta||^2 + C\sum_{i=1}^{N}\zeta_i - \sum_{i=1}^{N}\alpha_i[y_i(x_i^T\beta + \beta_0) - (1-\zeta_i)] - \sum_{i=1}^{N}\mu_i\zeta_i \tag{3.1}$$

The function described in (3.1) is called the Lagrange Primal Function, abbreviated as L_P. Our goal is to minimize this function. We now minimize the above function with respect to β, β_0, ζ_i $\forall$ i to derive the following result.

$$\beta = \sum_{i=1}^{N} \alpha_i y_i x_i$$
$$0 = \sum_{i=1}^{N} \alpha_i y_i$$
$$\alpha_i = C - \mu_i$$
$$\forall i, i = 1, .., N$$

Now we substitute the above back into our primal equation (3.1). Then we have the following:

$$L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j x_i^T x_j \tag{3.2}$$

The formulation (3.2) is called the Lagrange Dual function.

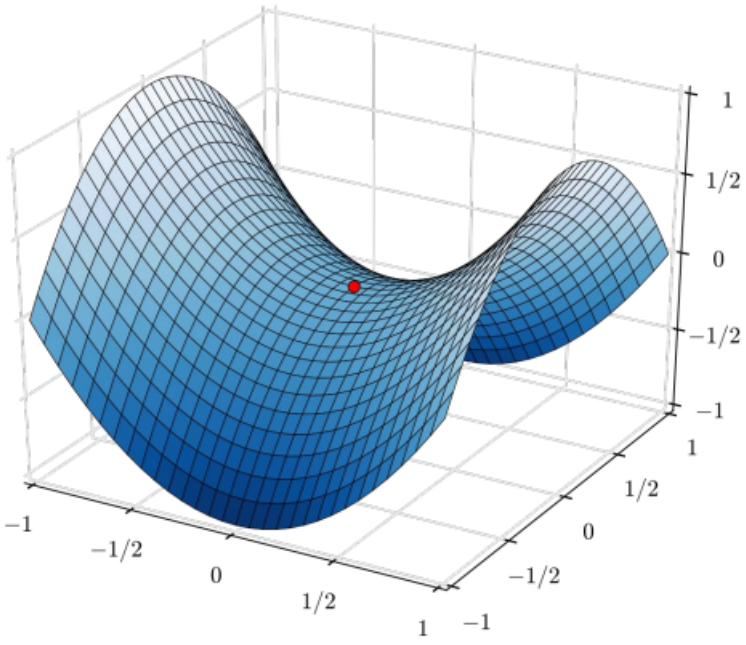
Notice that now we have a function we can optimize with respect to only one variable, α. We optimize the Dual formulation of the Lagrangian subject to: $0 \leq \alpha_i \leq C$ and $\sum_{i=1}^{N} \alpha_i y_i$. There is however one more condition that needs to be satisfied and it is called the KKT (Karsh-Kuhn-Tucker). This condition gives us an extra set of three constraints that need to be satisfied on top of the ones we have seen so far. Here are the extra constraints that come from KKT:

$$\alpha_i [y_i (x_i^T \beta + \beta_0) - (1 - \zeta_i)] = 0$$
$$\mu_i \zeta_i = 0$$
$$y_i (x_i^T \beta + \beta 0) - (1 - \zeta_i) \geq 0$$

Keeping all these constraints in mind, we solve the dual formulation of the Lagrangian. Let us say our optimal alpha that optimizes the Dual formulation is: α_i^*. we can now substitute it back into the equation (3.1) and its constraints to derive the solutions for the optimal weights, say β^*:

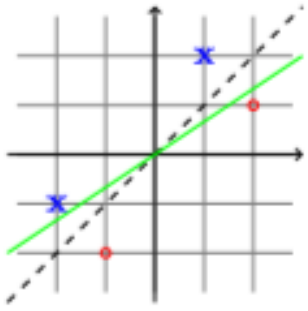
$$\beta^* = \sum_{i=1}^{N} \alpha^* y_i x_i \tag{3.3}$$

where of course, as can be seen in the original constrains, the coefficients are not zero. Here is a pictorial representation of what we described above. The saddle point (coloured in red) is the point at which our function achieves its minimum.



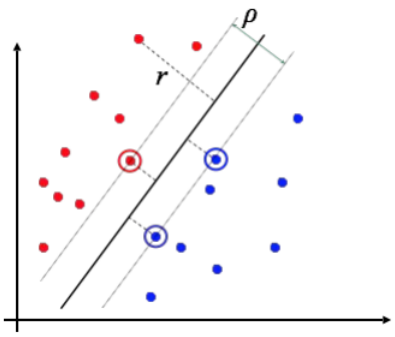
3.3.3 Refresher on SVM

We assume that our data is linearly separable and our goal is to build a classifier (linear) to predict unobserved instances' labels.However, needless to say, this is an idea that comes from Perceptron. The goal of SVM is to find the "best" classifier. See below.



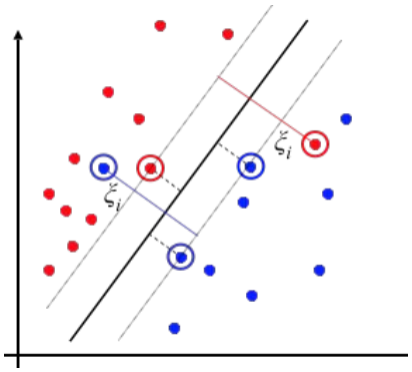
It is clear from the picture which of the two linear boundaries is better. SVM's goal is to find such boundary. Thus more concretely, the goal is to maximize the margin ρ and mathematically we come to the following formulation.

$$|\beta^T x + \beta_0| = 1 \tag{3.4}$$
$$\min \Psi(\beta) = \beta^T \beta \tag{3.5}$$
$$\text{subject to} \tag{3.6}$$
$$y_i(\beta^T x_i + \beta_0) \geq 1 \tag{3.7}$$



However, we may not always guarantee that our data is perfectly linearly separable. In that case, we only need to slightly modify our formulation. Observe:

$$\min \Psi(\beta) = \beta^T \beta + C\Sigma\zeta_i \tag{3.8}$$
$$\text{subject to} \tag{3.9}$$
$$y_i(\beta^T x_i + \beta_0) \geq 1 - \zeta_i \tag{3.10}$$
$$\zeta_i \geq 0 \tag{3.11}$$



Remark: $\Sigma\zeta_i$ is also called the hinge loss

After some algebra, it is easy to see that $(\beta^T\beta)^{-1} = ||\beta||^{-1}$ is the margin we wish to maximize (we could also say we want to minimize: $||\beta|| = (\beta^T\beta)$ as suggested above) with the slack variables ζ. Above minimization problem can be translated to the quadratic formulation that was introduced earlier and optimal solutions can be derived in very much the same process illustrated above in our discussion of optimization.

3.3.4 Poisoning Attack

So what can possibly go wrong? Quite a bit, as we will see from this point on. As mentioned in our earliest discussion of poisoning attacks, a sneaky and intelligent adversary can construct an instance that needs to only wait patiently until collection time of data for the training algorithm to rather maximize the hinge loss on the validation data. Such point can be found using the gradient ascent algorithm. The key is to update the attacking point with the addition of some appropriate gradient after each iteration. Thus, our problem is now to compute such gradient. This can also be formulated as a maximization problem: find x_c such that the hinge loss function is maximized. The process of arriving to the solution is beyond the scope of this summary and one can study the referenced paper to see the derivation process. Nonetheless, we present the high-level idea of the algorithm.

3.3.5 The Algorithm

In this section, we introduce the algorithm to compute the attacking point. As will be shown, the highlight of the algorithm is really the computation of the "toxic" gradient u, which will cause the loss function to increase (we often call this gradient ascend for this reason). As mentioned in the earlier section, the derivation and computation of the gradient u is beyond the scope of this summary, but can be seen in detail in the original paper.

Algorithm 1: Find attack point

Input: Training Data (D_{tr}), validation data, attack point label, initial attack point (x_c^0, y_c), step size (t)
Output: final attack point
1. $\{\alpha, b\} \longleftarrow$ solve SVM on D_{tr}
2. pick initial attacking point x_c^0
3. p $\longleftarrow$ 0

while $L(x_c^p)$ - $L(x_c^{p-1}) > \epsilon$ **do**
  perform Gradient Ascent
  recompute $\{\alpha, b\}$ on D_{tr} U $\{x_c^p, y_c\}$
  compute the "poison" gradient u.
  normalize u
  $x_c^{p+1} \longleftarrow x_c^p + tu$
end
return: $x_c = x_c^p$

With respect to the algorithm above, please keep the following in mind.

1. Adversary knows the training data used by the learner.
2. initial attack point is picked from a region sufficiently deep within the attacking class's margin.
3. L(x_c^j) is the loss function with respect to the attacking point at each iteration, j = 1,2,...,m.

Here is the trajectory of the attacking point as the algorithm proceeds. (This experiment was done on artificial data)

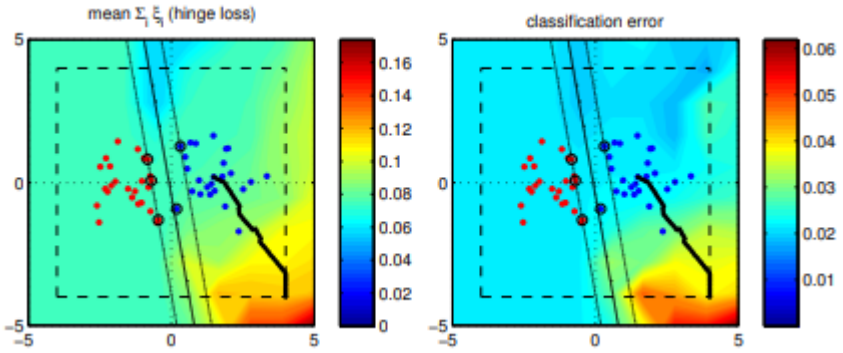
1. Heatmap background is the error surface

2. Dashed lines (the inner square) is the co-domain of attack points

3. Thick black line is the trajectory of the attack point toward the local maximum

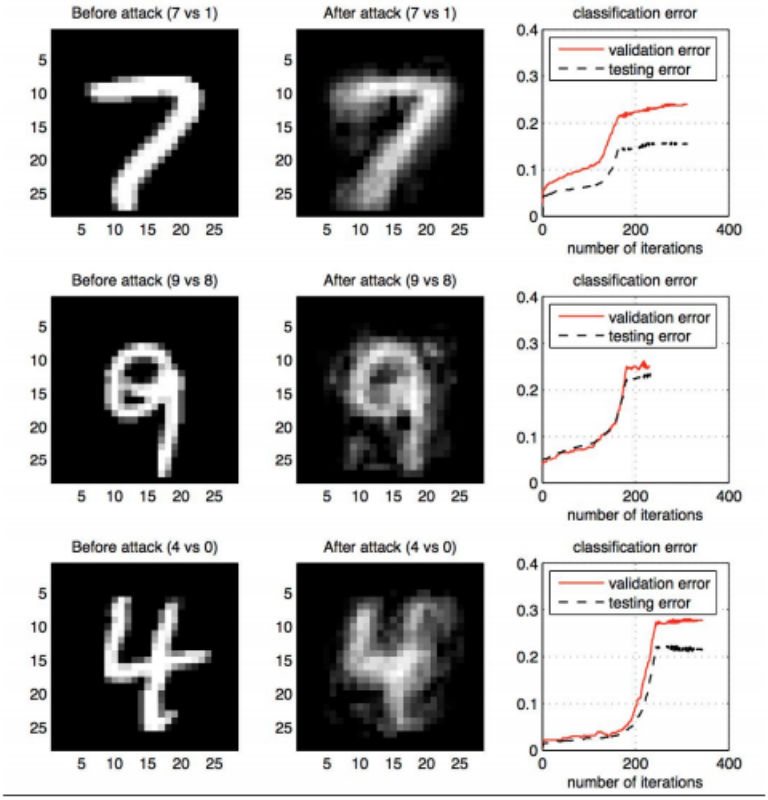
4. Normal black line is the decision boundary

5. Mini black circles are the support vectors



3.3.6 Final Result and evaluation

We end our discussion with some empirical data on the effectiveness of poisoning attack against SVM. Below is the summary of the results. As can be seen, the experiment was done on three pairs of digits: (7,1), (9,8), and (4,0). There is a stark contrast between the digits before the attack and after the attack as can be seen in the first two columns of the figure. The last column of the figure shows the increasing error over the course of the attack.


Before attack (7 vs 1)
After attack (7 vs 1)
classification error
validation error
testing error
number of iterations
Before attack (9 vs 8)
After attack (9 vs 8)
classification error
validation error
testing error
number of iterations
Before attack (4 vs 0)
After attack (4 vs 0)
classification error
validation error
testing error
number of iterations

3.4 References

B. Biggio, B. Nelson, and P. Laskov. Poisoning attacks against support vector machines. In International Conference on Machine Learning (ICML), pages 1467–1474, 2012.

T. Hastie, R. Tibshirani, and J. Friedman. The Elements of Statistical Learning: Data Mining, Inference, and Prediction

S. Ben-David and S.Shalev-Schwartz. Understanding Machine Learning: From Theory to Algorithm

Wu, Cheng-Ju(2017) Poisoning Attacks Against Support Vector Machines[PowerPoint slide 14]. Retrieved from https://people.eecs.berkeley.edu/ roydong/fa17files/EECS290OIEOR290-StudentPresentations-Wu-01.pdf