1. Algorithms

1.1. Stochastic coordinate descent

Algorithm 1 Stochastic coordinate descent

**Input:** Data (feature vectors) \( x_i \in \mathbb{R}^d \) with labels \( y_i \in \{+1, -1\} \), number of votes \( rr \in \mathbb{N} \) (Natural numbers), number of iterations per vote \( it \in \mathbb{N} \) (Natural numbers), batch size as a percent of training data \( p \in [0, 1] \) (set to 0.75 by default), and \( w_{inc} \in \mathbb{R} \) (set to 0.17 by default)

**Output:** Total of \( rr \) pairs of \((\text{best}w \in \mathbb{R}^d, \text{best}w_0 \in \mathbb{R})\) after each vote

**Procedure:**

1. Set \( j = 0 \)
2. While \( j < rr \) do
   1. Set \( \text{best}w = \text{null}, \text{best}w_0 = \text{null}, \text{bestloss} = \infty \)
   2. For \( i = 0 \) to \( it \) do
      1. Randomly pick \( p \) percent of rows as input training data to the coordinate descent algorithm and run it to completion starting with the values of \( w \) and \( w_0 \) from the previous call to it (if \( i == 0 \) we set \( w = \text{null}, w_0 = \text{null} \)).
      2. In the next step we calculate objectives on the full input training set
         1. If \( \text{objective}(w, w_0) < \text{objective}(\text{best}w, \text{best}w_0) \) then
            1. Set \( \text{best}w = w \), \( \text{best}w_0 = w_0 \), and \( \text{bestloss} = \text{objective}(w, w_0) \)
         end if
   end for
3. Output \( \text{best}w \) and \( \text{best}w_0 \)
4. Set \( j = j + 1 \).
end while

We output all \((\text{best}w, \text{best}w_0)\) pairs across the votes. We can use the pair with the lowest objective or the majority vote of all pairs for prediction.

1.2. Optimal threshold \( w_0 \) and 01 loss

Algorithm 2 Opt

**Input:** \( w^T x_i \in \mathbb{R}^d \) for \( i = 0..n-1 \) with labels \( y_i \in \{+1, -1\}, \text{start}, \text{end} \)

**Output:** Optimal \( w_0 \in \mathbb{R} \) with minimum (balanced) 01 loss and the loss value \( \text{obj} \)

**Procedure:**

1. For \( i = \text{start} \) to \( \text{end} - 1 \) do
2. Set \( w_0' = w^T x_i + w^T x_{i+1} \)
3. If \( y_i(w^T x_i + w_0') == 0 \) then
   1. If \( y_i == 1 \) then errorplus++
   end if
4. Else if \( y_i(w^T x_i + w_0') > 0 \) then
   1. If \( y_i == 1 \) then errorplus++ else errorminus--
   end if
5. Else if \( y_i(w^T x_i + w_0') < 0 \) then
   1. If \( y_i == 1 \) then errorplus++ else errorminus++
   end if
6. If \( \text{errorplus} + \text{errorminus} < \text{obj} \) is lower than current best objective \( \text{obj} \) then \( \text{obj} = \text{obj}' \) and \( w_0 = w_0' \).
7. End if
8. End for
9. Return \((w_0, \text{obj})\)

---

\(^1\)Department of Computer Science, New Jersey Institute of Technology, Newark, NJ 07102, USA. Correspondence to: Usman Roshan <usman@njit.edu>. 

---

Supplementary Material for "Robust binary classification with the 01 loss"

Yunzhe Xue \(^1\) Meiyan Xie \(^1\) Usman Roshan \(^1\)
1.3. Black box adversarial attacks

Algorithm 3

**Input:** Model $M$ to be attacked, adversarial attacker $B$, $\lambda$ and $\epsilon$ that determine amount of adversarial perturbation in each sample where $\lambda$ is used in training the substitute model and $\epsilon$ is to generate adversaries to attack the target model, data $x_i \in \mathbb{R}^d$ with labels $y_i \in \{+1, -1\}$, number of epochs $ep \in \mathbb{N}$ (Natural numbers)

**Procedure:**

Set the initial data $D = \{x_i\}$ as 200 random samples from the test dataset.

for $i = 0$ to $ep$ do

1. Obtain predictions $y'_i$ of $D$ from black box model $M$
2. Set adversarial training data $A$ to be $D = \{x_i, y'_i\}$
3. Train attacker $B$ with $A$ as input training data
4. With $B$’s gradient we produce adversarial examples as augmented data to train the substitute with the step below.
5. For each sample $a_i$ in $A$ create adversary $a_i = a_i \pm \lambda \text{sign}(\nabla f)$ where $\nabla f$ is the gradient of $B$ and $\lambda$ is given in the input. We randomly decide to add or subtract $\lambda$ by a coin flip and found this trick to improve the substitute model test accuracy and produce more effective adversarial examples.
6. Add new adversarial samples $\{a_i\}$ to $D$. This doubles the number of adversarial samples after each iteration.

end for

Now that our attacked $B$ is trained we produced adversaries for the remaining test datapoints. For each datapoint $x$ in the test dataset minus the 200 selected initially to train the substitute we produce adversaries using $x' = x + \epsilon \text{sign}(\nabla f)$ as in step 5 above but now we use $\epsilon$ instead of $\lambda$. We now test the accuracy of the target model $M$ with the newly generated adversaries.

In the above procedure we set $\lambda = 0.1$ for MNIST and CIFAR10 and $\lambda = 0.01$ for STL10 and ImageNet since these values produce the most effective attack. We use different values of $\epsilon$ that we show in the main paper and in this Supplementary Material below.

2. Results

2.1. CIFAR10, STL10, and ImageNet lower $\epsilon$ values

![Accuracy of adversarial samples generated at each epoch during substitute model training on CIFAR10, STL10, and ImageNet. At epoch 0 we have the accuracy of the target model on clean test data (without adversaries) as shown in the tables.](image)
2.2. MNIST lower $\epsilon$ values

Figure 2. Accuracy of adversarial samples on MNIST lower $\epsilon$ values (see Figure 2 caption for more)

<table>
<thead>
<tr>
<th>Epoch</th>
<th>SCD01majvote</th>
<th>SVM</th>
<th>MLP01majvote</th>
<th>MLP</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>20</td>
<td>.7</td>
<td>.79</td>
<td>.8</td>
<td>.91</td>
</tr>
</tbody>
</table>

2.3. Comparison to prior work

Figure 3. Accuracy of adversarial samples of the previous stochastic coordinate descent 01 loss solver and our SCD01
2.4. GTSRB and CelebA lower $\epsilon$ values

![Diagram of GTSRB eps=0.015625](image)

<table>
<thead>
<tr>
<th>Epoch</th>
<th>MLP01majvote</th>
<th>SVM</th>
<th>MLP</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>.98</td>
<td>.99</td>
<td>.98</td>
</tr>
<tr>
<td>20</td>
<td>.71</td>
<td>.65</td>
<td>.66</td>
</tr>
</tbody>
</table>

![Diagram of CelebA eps=0.03125](image)

<table>
<thead>
<tr>
<th>Epoch</th>
<th>MLP01majvote</th>
<th>SVM</th>
<th>MLP</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>.79</td>
<td>.76</td>
<td>.78</td>
</tr>
<tr>
<td>20</td>
<td>.58</td>
<td>.14</td>
<td>.2</td>
</tr>
</tbody>
</table>

Figure 4. Accuracy of adversarial samples on GTSRB and CelebA (see Figure 2 caption for more)