Huberized support vector machine

Lingyu Zhang

Mar 16, 2018

Theoretical part

0.1 Stochastic Gradient Descent for elastic-net regularized HSVM

The objective function is

$$F = \frac{1}{N}\sum_{i=1}^{N} h_\delta(y_i(b + x_i^T w)) + \lambda_1 ||w||_1 + \frac{\lambda_2}{2}||w||_2^2$$

where

$$h_\sigma(z_i) = \begin{cases} 0 & \text{if } z_i > 1 \\ \frac{(1-z_i)^2}{2\delta} & \text{if } 1-\delta < z_i \leq 1 \\ 1 - z_i - \frac{\delta}{2} & \text{if } z_i \leq 1-\delta \end{cases}$$

$$z_i = y_i(b + x_i^T w)$$

We need to find the optimal w and b which minimize the objective function. Therefore, we compute the gradient of the objective function with regard to the parameters we need to update. Let $z_i = y_i(b + x_i^T w)$ where i is randomly selected from $i \in \{1, N\}$
Now we compute ∇w and ∇b

$$\nabla w = \frac{1}{N}\sum_{i=1}^{N} \begin{cases} 0 & \text{if } z > 1 \\ \frac{z-1}{\delta}\frac{\partial z}{\partial w} & \text{if } 1-\delta < z \leq 1 \\ -1\frac{\partial z}{\partial w} & \text{if } z \leq 1-\delta \end{cases} + \lambda_1 * sign(w) + \lambda_2 * w$$

with $\frac{\partial z}{\partial w} = x_i * y_i$

$$\nabla b = \frac{1}{N}\sum_{i=1}^{N} \begin{cases} 0 & \text{if } z > 1 \\ \frac{z-1}{\delta}\frac{\partial z}{\partial b} & \text{if } 1-\delta < z \leq 1 \\ -1\frac{\partial z}{\partial b} & \text{if } z \leq 1-\delta \end{cases}$$

with $\frac{\partial z}{\partial b} = y_i$
For stochastic gradient descent, at each iteration ∇w and ∇b are computed based on a randomly selected sample with index i
We update the parameters as below:

$$w^{k+1} = w^k - \eta * \nabla w$$

$$b^{k+1} = b^k - \eta * \nabla b$$

Note that in the experimental part, we are using mini-batch stochastic gradient, where ∇w is the average of the gradient of w computed based on all the samples within a mini-batch, and ∇b is the average of the gradient of b computed based on all the samples within a mini-batch.

0.2 Stochastic Gradient Descent for Hinge-loss based support vector machine with regularization term

The objective function is

$$F = \frac{1}{N} max\Big(0, 1 - y_i(b + x_i^T w)\Big) + \lambda_1 ||w||_1 + \frac{\lambda_2}{2}||w||_2^2$$

We need to find the optimal w and b which minimize the objective function. Therefore, we compute the gradient of the objective function with regard to the parameters we need to update. Let $z_i = 1 - y_i(b + x_i^T w)$ where i is randomly selected from $i \in \{1, N\}$
Now we compute ∇w and ∇b

$$\nabla w = \frac{1}{N}\sum_{i=1}^{N} \begin{cases} \frac{\partial z}{\partial w} & \text{if } z > 0 \\ 0 & \text{if } z \leq 0 \end{cases} + \lambda_1 * sign(w) + \lambda_2 * w$$

with $\frac{\partial z}{\partial w} = -x_i * y_i$

$$\nabla b = \frac{1}{N}\sum_{i=1}^{N} \begin{cases} \frac{\partial z}{\partial b} & \text{if } z > 0 \\ 0 & \text{if } z \leq 0 \end{cases}$$

with $\frac{\partial z}{\partial b} = -y_i$
For stochastic gradient descent, at each iteration ∇w and ∇b are computed based on a randomly selected sample with index i.
We update the parameters as below:

$$w^{k+1} = w^k - \eta * \nabla w$$

$$b^{k+1} = b^k - \eta * \nabla b$$

Note that in the experimental part, we are using mini-batch stochastic gradient, where ∇w is the average of the gradient of w computed based on all the samples within a mini-batch, and ∇b is the average of the gradient of b computed based on all the samples within a mini-batch.

0.3 SAGA for elastic-net regularized HSVM

The objective function is

$$F = \frac{1}{N}\sum_{i=1}^{N} h_\delta(y_i(b + x_i^T w)) + \lambda_1 ||w||_1 + \frac{\lambda_2}{2}||w||_2^2$$

where

$$h_\sigma(z) = \begin{cases} 0 & \text{if } z > 1 \\ \frac{(1-z)^2}{2\delta} & \text{if } 1-\delta < z \leq 1 \\ 1 - z - \frac{\delta}{2} & \text{if } z \leq 1-\delta \end{cases}$$

We need to find the optimal w and b which minimize the objective function. Therefore, we compute the gradient of the objective function with regard to the parameters we need to update.
Let $z_i = y_i(b + x_i^T w)$ where i is the index of sample, $i \in \{1, N\}$
We construct a table g with the dimension $p * N$ where p is the dimension of w (w is $p * 1$) and N is the total number of samples. Firstly g is initialized as w^0

$$g^0 = \begin{bmatrix} g_1^0 & g_2^0 & ... & g_N^0 \end{bmatrix} = \begin{bmatrix} w_1^0 & w_2^0 & ... & w_N^0 \end{bmatrix}$$

For the update of table g, at each iteration, we only update one element of g, with the index m, where m is randomly generated from $\{1, N\}$
At each iteration, we update g_m as below:

$$g_m^{k+1} = \nabla w_m^k = \begin{cases} 0 & \text{if } z_m > 1 \\ \frac{z_m - 1}{\delta} \frac{\partial z_m}{\partial w} & \text{if } 1-\delta < z_m \leq 1 \\ -1 \frac{\partial z_m}{\partial w} & \text{if } z_m \leq 1-\delta \end{cases}$$

For the rest (N-1) elements of g, the values remain the same, therefore at each iteration, g tables becomes as below:

$$g^{k+1} = \begin{bmatrix} g_1^{k+1} & g_2^{k+1} & ... & g_m^{k+1} & ... & g_N^{k+1} \end{bmatrix} = \begin{bmatrix} g_1^k & g_2^k & ... & g_m^{k+1} & ... & g_N^k \end{bmatrix}$$

Therefore, the parameter w at each iteration will be updated as below:

$$w^{k+1} = w^k - \eta * \left[g_m^{k+1} - g_m^k + \frac{1}{N}(g_1^k + g_2^k + g_3^k + ... + g_N^k + \lambda_1 * sign(w^k) + \lambda_2 * w^k) \right]$$

$$w^{k+1} = w^k - \eta * \frac{1}{N} \left[N(g_m^{k+1} - g_m^k) + (g_1^k + g_2^k + g_3^k + ... + g_N^k) + \lambda_1 * sign(w^k) + \lambda_2 * w^k \right]$$

$$w^{k+1} = w^k - \eta * \frac{1}{N} \left[(N-1)(g_m^{k+1} - g_m^k) + (g_1^k + g_2^k + g_3^k + ... + g_m^{k+1} + ... + g_N^k) + \lambda_1 * sign(w^k) + \lambda_2 * w^k \right]$$

0.4 SVRG for elastic-net regularized HSVM

The objective function is

$$F = \frac{1}{N} \sum_{i=1}^{N} h_\delta(y_i(b + x_i^T w)) + \lambda_1 ||w||_1 + \frac{\lambda_2}{2} ||w||_2^2$$

where

$$h_\sigma(z) = \begin{cases} 0 & \text{if } z > 1 \\ \frac{(1-z)^2}{2\delta} & \text{if } 1-\delta < z \leq 1 \\ 1 - z - \frac{\delta}{2} & \text{if } z \leq 1-\delta \end{cases}$$

We need to find the optimal w and b which minimize the objective function. Therefore, we compute the gradient of the objective function with regard to the parameters we need to update.
Let $z_i = y_i(b + x_i^T w)$ where i is the index of sample, $i \in \{1, N\}$

$$\widetilde{z} = y_i(b + x_i^T \widetilde{w})$$

We choose $\widetilde{w}$ as the average of all the historical value of w

$$\widetilde{w} = \overline{w}$$

At each iteration, we update the parameter w as below:

$$w^{k+1} = w^k - \eta\left[\frac{\partial h_m(w^k)}{\partial w^k} - \frac{\partial h_m(\widetilde{w})}{\partial \widetilde{w}} + \frac{1}{N}\Big(\sum_{i=1}^{N} \frac{\partial h_i(\widetilde{z})}{\partial \widetilde{w}} + \lambda_1 ||w^k||_1 + \frac{\lambda_2}{2}||w^k||_2^2\Big)\right]$$

$$w^{k+1} = w^k - \eta * \frac{1}{N}\left[N\Big(\frac{\partial h_m(w^k)}{\partial w^k} - \frac{\partial h_m(\widetilde{w})}{\partial \widetilde{w}}\Big) + \Big(\sum_{i=1}^{N} \frac{\partial h_i(\widetilde{z})}{\partial \widetilde{w}} + \lambda_1 ||w^k||_1 + \frac{\lambda_2}{2}||w^k||_2^2\Big)\right]$$

where

$$\frac{\partial h_i(w)}{\partial w} = \frac{1}{N}\sum_{i=1}^{N} \begin{cases} 0 & \text{if } z > 1 \\ \frac{z-1}{\delta}\frac{\partial z}{\partial w} & \text{if } 1-\delta < z \leq 1 \\ -1\frac{\partial z}{\partial w} & \text{if } z \leq 1-\delta \end{cases}$$

with $\frac{\partial z}{\partial w} = x_i * y_i$

Experimental part

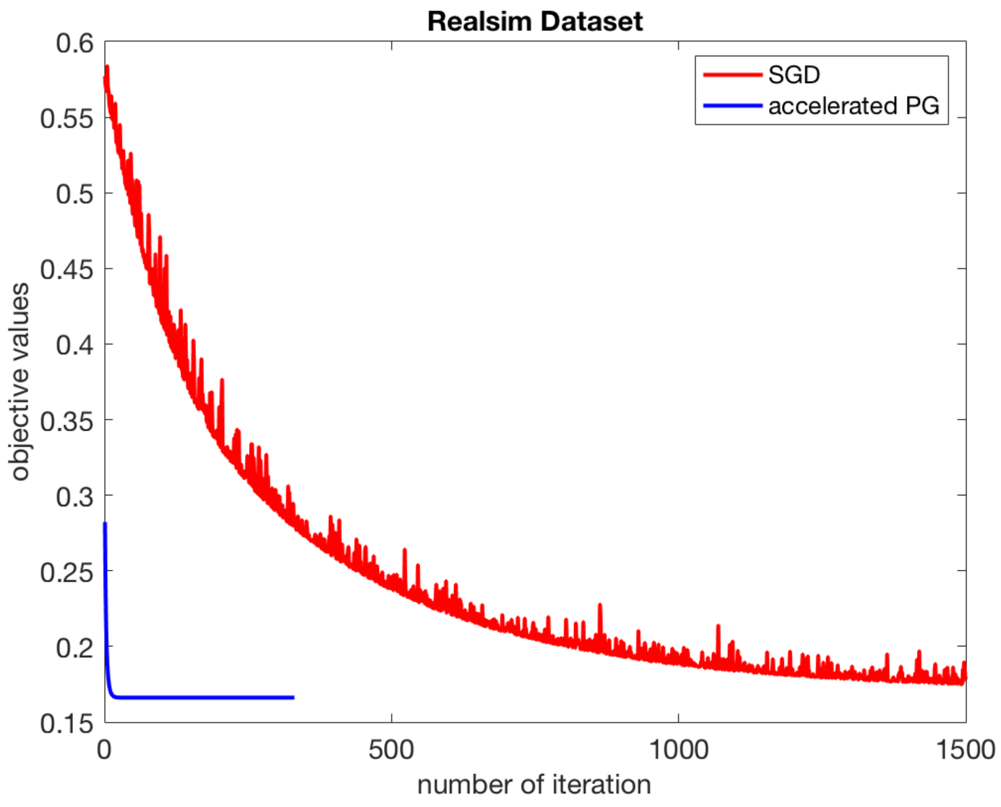
Compare SGD and Accelerated proximal gradient for Elastic-net regularized HSVM

Firstly for data preparation, we shuffle the data to enable the stochastic selection of the batch data. We initialize the parameters, the weight matrix is initialized by normal distribution with mean = 0 and standard derivation = 0.1 For the bias vector, we initialized them as zero vectors. We tune the hyper-parameters and the optimal ones are as below: We choose the mini-batch size as 50, we firstly use a learning rate and then tune it based on the speed of convergence.

$$\text{learning rate } \eta = 1.5$$

$$\text{batch size} = 50$$

Below is the objective values plot for stochastic gradient over iterations. In order to compare the performance of different models, we plot the objective values acquired by Acclerated Proximal Gradient developed by Prof. Xu to see the difference.



We can find that the loss is going down with the training iteration increases and finally converged. Below is the prediction accuracy for testing set after 1500 iterations

```
SGD: suc_rate = 0.94, time = 188.3498

APG: suc_rate = 0.95, time = 5.6813
```

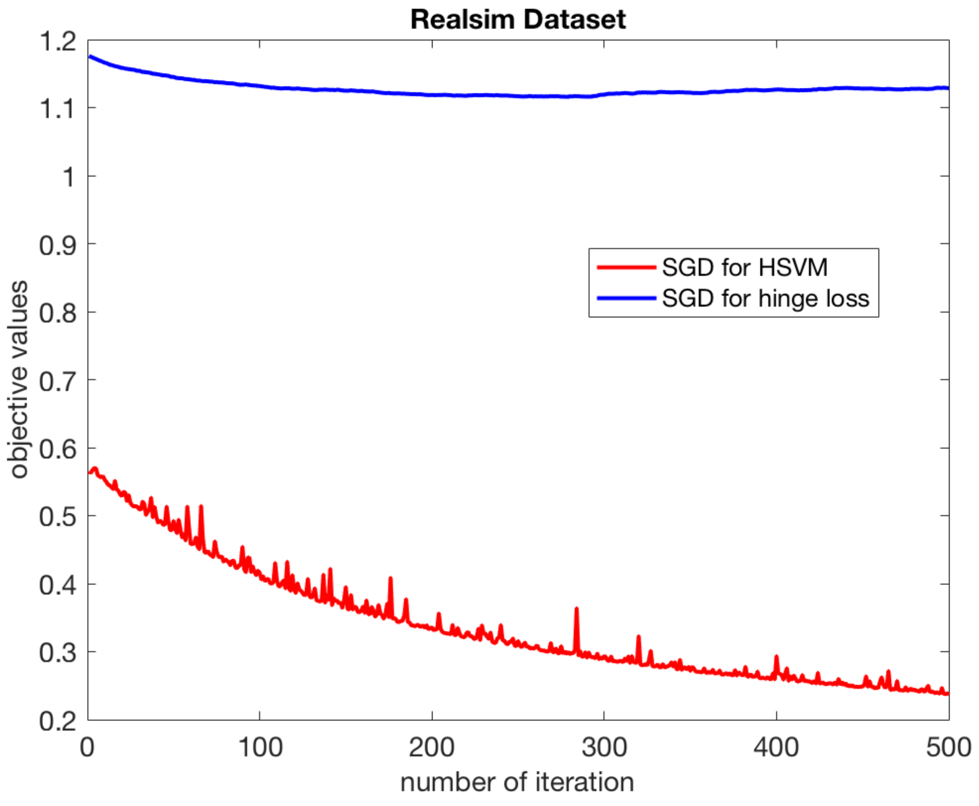
Compare the performance of SGD on Elastic-net regularized HSVM and Hinge loss regularized SVM

For data preparation, we firstly shuffle the data to enable the stochastic selection of the batch data. For SGD on hinge loss regularized SVM, firstly we initialize the parameters, the weight matrix is initialized by normal distribution with mean = 0 and standard derivation = 0.1 For the bias vector, we initialized them as zero vectors. We tune the hyper-parameters and the optimal ones are as below: We choose the mini-batch size as 50, we firstly use a learning rate and then tune it based on the speed of convergence.

$$\text{learning rate } \eta = 6.0$$

$$\text{batch size} = 50$$

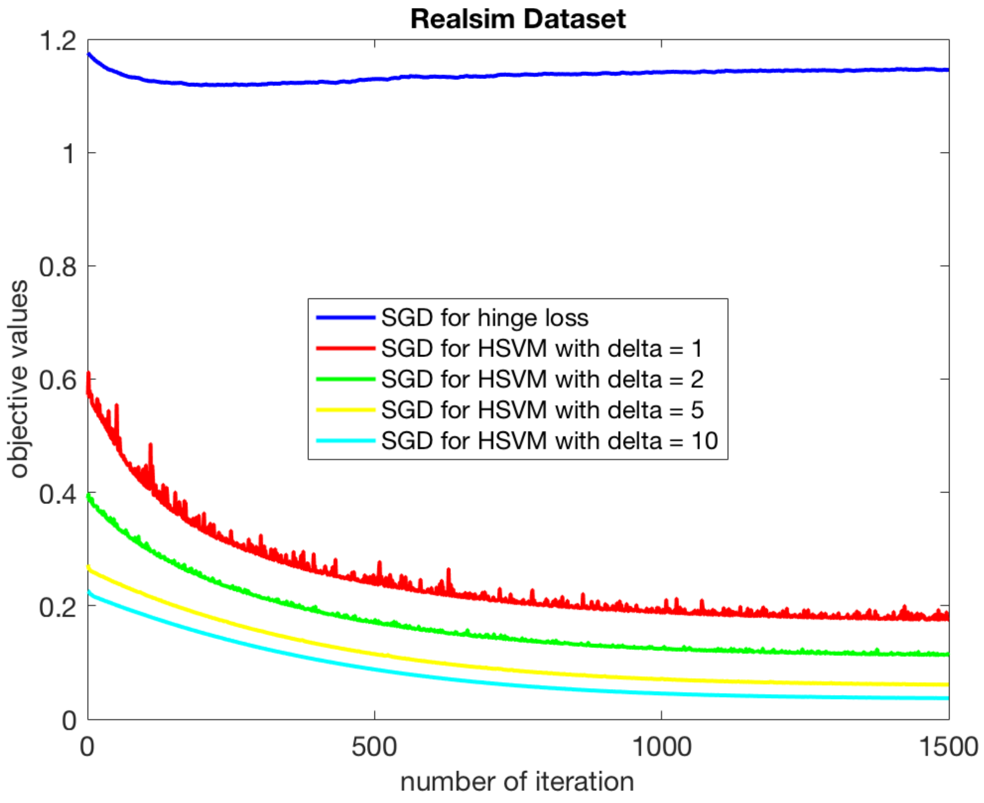
Below is the objective values plot for stochastic gradient over iterations on both objective functions.



We can find that the loss is going down with the training iteration increases and finally converged. Below is the prediction accuracy for testing set after 500 iterations

```
SGD for HSVM: suc_rate = 0.93, time = 71.6951

startSGD for Hinge loss: suc_rate = 0.92, time = 65.9800
```

In order to evaluate the effect of smooth parameter δ , we vary the value of δ and compare the objective functions over iteration as below:



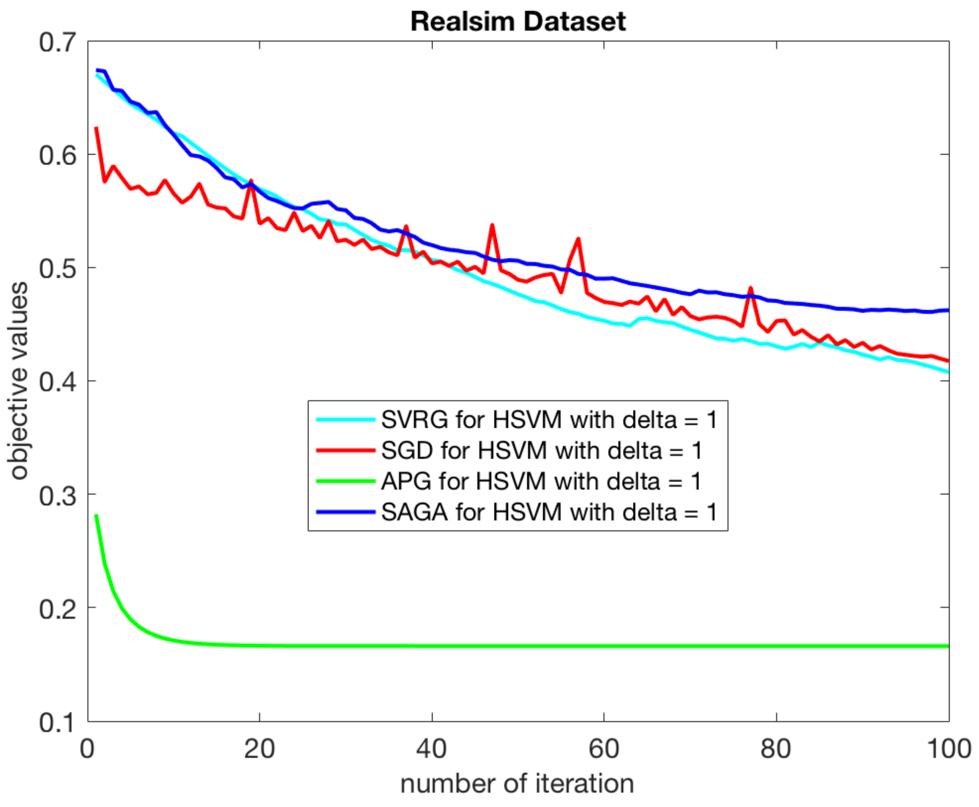
We can find that the loss is going down with the training iteration increases and finally converged. The speed of convergence is nearly the same for different *delta*. Below is the prediction accuracy for testing set after 1500 iterations

```
>> sgd_hinge_test_lingyu
start optimization based on hinge loss
SGD for Hinge loss: suc_rate = 0.89, time = 201.7073

start sgd based on Elastic net HSVM loss
SGD for HSVM with delta = 1.00: suc_rate = 0.93, time = 194.5040

start sgd based on Elastic net HSVM loss
SGD for HSVM with delta = 2.00: suc_rate = 0.92, time = 190.5488

start sgd based on Elastic net HSVM loss
SGD for HSVM with delta = 5.00: suc_rate = 0.82, time = 200.2859

start sgd based on Elastic net HSVM loss
SGD for HSVM with delta = 10.00: suc_rate = 0.74, time = 205.3726
```

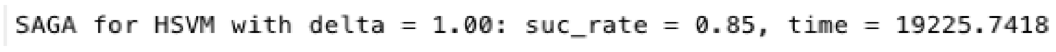
We can find that although the final converged loss is reduced with the increase of δ, the prediction accuracy acutally decreases, it might because the δ affects the calculation of loss function value, smooth the convergence while not improve the performance of the model.

Compare the performance of SVRG, SAGA, SGD and Accelerated proximal gradient on Elastic-net HSVM

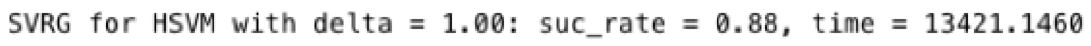
For SAGA on Elastic-net HSVM, firstly we initialize the parameters, the weight matrix is initialized by normal distribution with mean = 0 and standard derivation = 0.1 For the bias vector, we initialized them as zero vectors. We tune the hyper-parameters and the optimal ones are as below: we firstly use a learning rate and then tune it based on the speed of convergence, the optimal learning rate is $\eta = 1.5$. For SVRG on Elastic-net HSVM, firstly we initialize the parameters, the weight matrix is initialized by normal distribution with mean = 0 and standard derivation = 0.1 For the bias vector, we initialized them as zero vectors. We tune the hyper-parameters and the optimal ones are as below: we firstly use a learning rate and then tune it based on the speed of convergence, the optimal learning rate is $\eta = 1.5$.
Below is the objective values of different models over iterations.



We can find that after 100 iterations, the APG model has already converged, while the loss functions for SVRG, SAGA and SGD are continued to converge. The testing accuracy for the SAGA and SVRG after 100 iterations are shown as below:

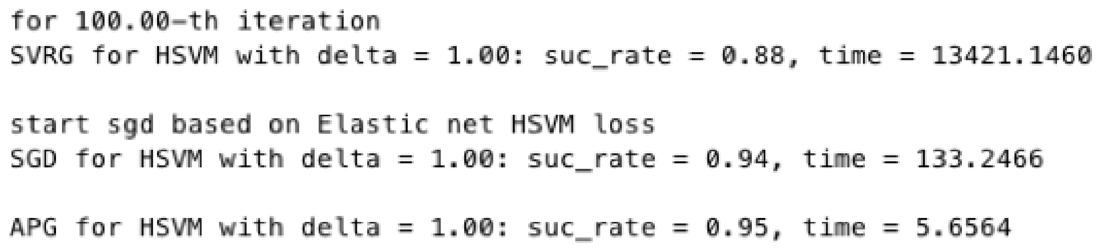


```
SAGA for HSVM with delta = 1.00: suc_rate = 0.85, time = 19225.7418
```

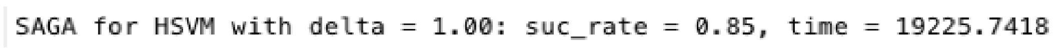


```
SVRG for HSVM with delta = 1.00: suc_rate = 0.88, time = 13421.1460
```

We can find that after 100 iterations, the testing accuracy for SAGA is 85% and for SVRG is 88%, since after 100 iterations, the loss function has not converged, we need to do more iterations until it converges to get higher accuracy. Also note that it taks around 5 hours for either SAGA and SVRG to run 100 iterations, we then fix the iterations for SAGA and SVRG to be 100 and increase the number of iterations for SGD and APG to be 1000 and the performance is as below:



```
for 100.00-th iteration
SVRG for HSVM with delta = 1.00: suc_rate = 0.88, time = 13421.1460

start sgd based on Elastic net HSVM loss
SGD for HSVM with delta = 1.00: suc_rate = 0.94, time = 133.2466

APG for HSVM with delta = 1.00: suc_rate = 0.95, time = 5.6564
```



```
SAGA for HSVM with delta = 1.00: suc_rate = 0.85, time = 19225.7418
```

Note that the number of iterations should be increased for SAGA and SVRG to get final optimal model, tried to modify the code to be more efficient through using gpuArray in matlab utilizing the GPU of the server and convering the for loop to be parfor loop for parallel computing, but not got significant improved speed yet, might need to improve the efficiency of the code by other possible ways.