Technical University of Crete

School of Electrical and Computer Engineering

Course: **Optimization**

Exercise Bonus(2) (50/1000)

Angelopoulos Dimitris, 2020030038

In this exercise, we will perform data classification using Logistic Regression (LR) and Support Vector Machines (SVMs).

A. First of all we will generate an approximately linearly separable dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ with $\mathbf{x}_i \in \mathbb{R}^N$ and

(a) $y_i \in \{1, 0\}$ for LR,

(b) $y_i \in \{+1, -1\}$ for SVMs.

We generate a hyperplane $\mathbb{H}_{b,\mathbf{w}} = \{\mathbf{x} \in \mathbb{R}^N \mid \mathbf{w}^T\mathbf{x} = b\}$ with $\mathbf{w} \in \mathbb{R}^N$, $b \in \mathbb{R}$ chosen randomly. Then we project a random point $\mathbf{x}_0' \in \mathbb{R}^N$ in the hyperplane, using :

$$\mathcal{P}_{\mathbb{H}_{b,\mathbf{w}}}(\mathbf{x}) = \mathbf{F}^{-1}(1:n,:)\begin{bmatrix}\mathbf{x}\\ \mathbf{b}\end{bmatrix}, \quad \mathbf{F} = \begin{bmatrix}\mathbb{I}_{N\times N} & \mathbf{w}\\ \mathbf{w}^T & 0\end{bmatrix}$$

After finding a random point $\mathbf{x}_0 \in \mathbb{H}_{b,\mathbf{w}}$, we generate the labels y_i randomly, for the given problem (LR or SVM) and the data points $\mathbf{x}_0$ as follows :

$$\mathbf{x}_i = \begin{cases}\mathbf{x}_0 + \mathbf{w} + \sigma * \text{randn(N,1)}, & \text{if } y_i = 1\\ \mathbf{x}_0 - \mathbf{w} + \sigma * \text{randn(N,1)}, & \text{if } y_i \neq 1\end{cases}$$

We check whether the data are separable with respect to the original hyperplane. This is important as we have to make sure to generate a separable dataset for the hard-SVMs case that follows.

Last but not least, we set $\boldsymbol{\theta} = (b, \mathbf{w})$ and augment the data as following :

$$\mathbf{X} = \begin{bmatrix} -1 & \dots & -1 \\ \mathbf{x}_1 & \dots & \mathbf{x}_n \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_1 & \dots & y_n \end{bmatrix}^T$$

B. In this problem, we will perform classification choosing a $\boldsymbol{\theta}$ that minimizes the following cost function :

$$J_R(\boldsymbol{\theta}) = -\frac{1}{n}\sum_{j=1}^{n} y_j \log h_{\boldsymbol{\theta}}(\mathbf{x}_j) + (1-y_j)\log(1-h_{\boldsymbol{\theta}}(\mathbf{x}_j)) + \frac{\lambda}{2}\|\boldsymbol{\theta}\|_2^2$$

The gradient of this function is :

$$\nabla J_R(\boldsymbol{\theta}) = \frac{1}{n}\sum_{j=1}^{n}(h_{\boldsymbol{\theta}}(\mathbf{x}_j) - y_j)\mathbf{x}_j + \lambda\boldsymbol{\theta}$$

If we compute the Hessian we can prove that it is convex. Alse note that the logistic function used is :

$$h_{\boldsymbol{\theta}}(\mathbf{x}) = \frac{1}{1+e^{-\boldsymbol{\theta}^T\mathbf{x}}}$$

For 50 data samples and for different values for λ we get the following plot :

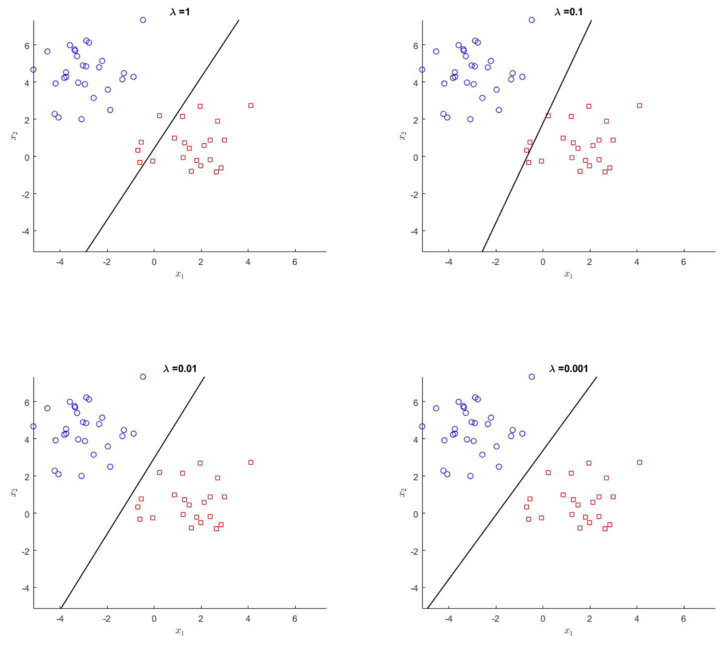

Fig. 1

We can see that as λ increases from 0 to 1, the solution becomes sub-optimal. That is expected as the regularization term acts as a penalty to the cost function, increasing its value and thus as we increase λ the cost can not reach the true minimum.

Now we will fix $\lambda = 0.1$ and develop the following the algorithms.

Algorithm 1 Gradient Descent using Backtracking

```
$\boldsymbol{\theta}_0 \in \mathbb{R}^N$, $k = 0$, $\alpha_0 = 1$, $a \in (0, 0.5)$ and $\beta \in (0, 1)$
While ($||\nabla J_R(\boldsymbol{\theta}_k)|| > \epsilon$)
  1. $\Delta\boldsymbol{\theta}_k := -\nabla J_R(\boldsymbol{\theta}_k)$
  2. While ($J_R(\boldsymbol{\theta}_k + \alpha_k \Delta\boldsymbol{\theta}_k) > J_R(\boldsymbol{\theta}_k) + a\alpha_k \nabla J_R(\boldsymbol{\theta}_k)^T \Delta\boldsymbol{\theta}_k$)
       1. $\alpha_k := \beta\alpha_k$
  3. $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \alpha_k \Delta\boldsymbol{\theta}_k$
  4. $k := k + 1$
```

Algorithm 2 Accelerated Gradient Descent using Backtracking

```
$\boldsymbol{\theta}_0 \in \mathbb{R}^N$, $\mathbf{y}_0 = \boldsymbol{\theta}_0$, $k = 0$, $\alpha_0 = 1$, $t_0 = 1$, $a \in (0, 0.5)$ and $\beta \in (0, 1)$
While ($\frac{\|\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k\|_2}{\|\boldsymbol{\theta}_k\|_2} > \epsilon$)
  1. $\Delta\mathbf{y}_k := -\nabla J_R(\mathbf{y}_k)$
  2. While ($J_R(\mathbf{y}_k + \alpha_k \Delta\mathbf{y}_k) > J_R(\mathbf{y}_k) + a\alpha_k \nabla J_R(\mathbf{y}_k)^T \Delta\mathbf{y}_k$)
       1. $\alpha_k := \beta\alpha_k$
  3. $t_{k+1} = \frac{1 + \sqrt{1 + 4t_k^2}}{2}$
  4. $\boldsymbol{\theta}_{k+1} = \mathbf{y}_k + \alpha_k \Delta\mathbf{y}_k$
  5. $\mathbf{y}_{k+1} = \boldsymbol{\theta}_{k+1} + \frac{t_k - 1}{t_{k+1}}(\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k)$
  6. $k := k + 1$
```

Algorithm 3 Stochastic Gradient Descent

```
$\boldsymbol{\theta}_0 \in \mathbb{R}^N$, $k = 0$
For $i$ in $1 \dots n_E$
    1. Uniformly shuffle $\mathcal{D}$
    For $j$ in $1 \dots \frac{n}{n_\mathcal{B}}$
        1. $\mathcal{B}_k = \mathcal{D}[(j-1)n_\mathcal{B} + 1 : jn_\mathcal{B}]$
        2. $\boldsymbol{\xi}_k = \frac{1}{|\mathcal{B}_k|} \sum_{j \in \mathcal{B}_k} \nabla J_{R,j}(\boldsymbol{\theta}_k)$
        3. $t_k = \frac{2}{\lambda(k+1)}$
        4. $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - t_k \boldsymbol{\xi}_k$
    End
    2. $k := k + 1$
End
```

After implementing every algorithm in MATLAB and choosing $|\mathcal{B}| = 10$ for 100 data samples we can get the following convergence figures.

For N = 2 :

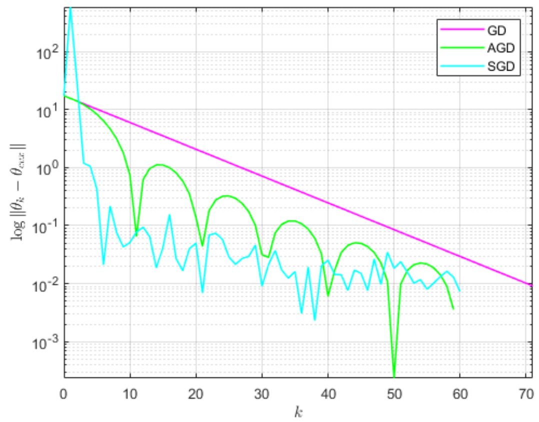

Fig. 2

For N = 20 :

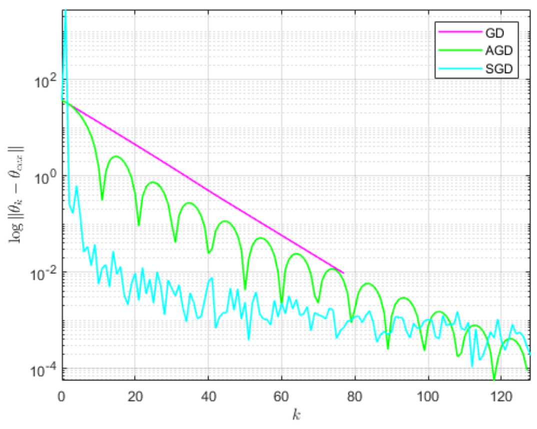

Fig. 3

For both cases we observe that the Stochastic GD converges faster than the Accelerated GD. One other noticeable difference is that the SGD is noisy compared to the (A)GD, that is because the data are uniformly shuffled. Generally SGD does not compute the gradient upon every sample as this can be computationally heavy for large datasets. Instead it "breaks apart" the data in smaller sets, called batches, and computes the gradients for every batch. This method relies on the fact that the data have common characteristics.

C. In this problem we will use another classification model called SVMs

1. We consider the Hard-SVM problem which converges only when our data are linearly separable, as follows.

$$\begin{aligned} \underset{\mathbf{w},b}{\text{minimize}} \quad & \|\mathbf{w}\|_2^2 \\ \text{s.t.} \quad & y_i(\mathbf{w}^T\mathbf{x}_i - b) \geq 1, i = 1 \dots n \end{aligned}$$

For $N = 2$ we solve the hard-SVM problem via CVX. The resulting figure is :

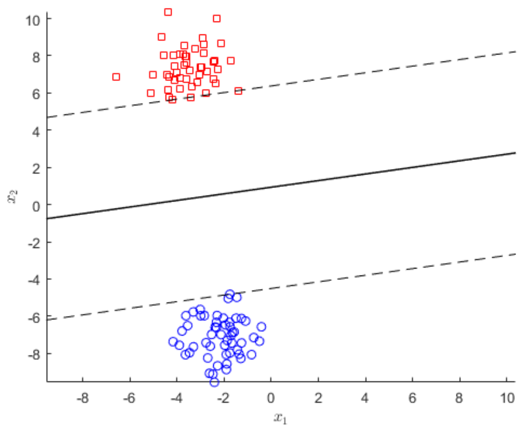

Fig. 4

The Homogeneous Hard-SVMs problem is given by the optimization problem that uses the augmented variable as shown below.

$$\begin{aligned} \underset{\boldsymbol{\theta}}{\text{minimize}} \quad & \frac{1}{2}\|\boldsymbol{\theta}\|_2^2 \\ \text{s.t.} \quad & y_i\boldsymbol{\theta}^T\mathbf{x}_i \geq 1, i = 1 \dots n \end{aligned}$$

Solving the homogeneous Hard-SVMs problem for the same dataset we get :

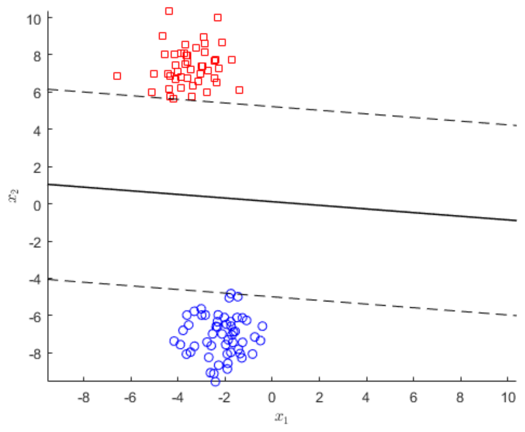

Fig. 5

We can see that for the homogeneous case the maximum margin is more tight than the original case. That happens because the homogeneous hard-SVM minimizes the norm of the augmented vector $\boldsymbol{\theta}$.

Now we will solve the dual homogeneous hard-SVM and then find the essentially nonzero Lagrange multipliers.

The Lagrangian of the primal is :

$$L(\boldsymbol{\theta}, \boldsymbol{\alpha}) = \frac{1}{2}\|\boldsymbol{\theta}\| + \sum_{i=1}^{n} \alpha_i (1 - y_i \boldsymbol{\theta}^T \mathbf{x}_i)$$

We get the dual by minimizing the Lagrangian over $\boldsymbol{\theta}$ as shown below.

$$g(\boldsymbol{\alpha}) = \underset{\boldsymbol{\theta}}{\text{minimize}}\, L(\boldsymbol{\theta}, \boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

The dual function g is concave and we get its solution by maximizing it over all the positive $\boldsymbol{\alpha}$.

$$\underset{\boldsymbol{\alpha} \in \mathbb{R}_+^n}{\text{maximize}} \quad g(\boldsymbol{\alpha})$$

The resulting SVM boundary is :

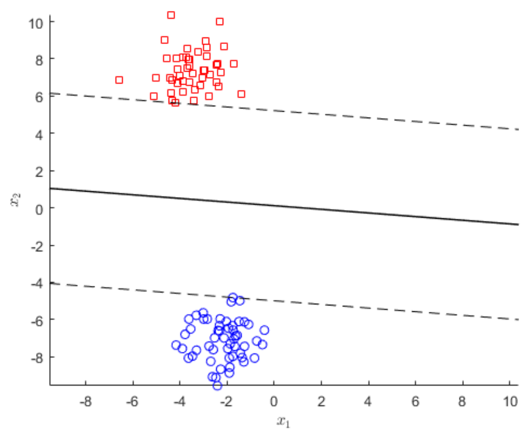

Fig. 6

First of all we observe that the result of the dual is the same with the one of the primal in Fig. 5. That is true because the optimization problem for hard-SVM is convex.

The data samples at the indicies of the non-zero Lagrange multipliers are $\mathbf{x}_{36} = (-4.2, 5.63)$ and $\mathbf{x}_{87} = (-1.75, -4.82)$. We can verify that these data points are the 2 support vectors as shown in the plots.

Lastly we use the following expression:

$$\boldsymbol{\theta} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$$

This computes the primal optimal through the dual. Implementing this, we verify that indeed the problems converge to the same value.

2. Now we will solve the homogeneous soft-SVM problem using the Stochastic Sub-gradient Descent(SSG).

$$\begin{aligned} \underset{\boldsymbol{\theta},\boldsymbol{\xi}}{\text{minimize}} \quad & \frac{\lambda}{2}\|\boldsymbol{\theta}\|_2^2 + \frac{1}{n}\mathbf{1}^T\boldsymbol{\xi} \\ \text{s.t.} \quad & 1 - \xi_i - y_i\boldsymbol{\theta}^T\mathbf{x}_i \leq 0, i = 1 \dots n \\ & -\boldsymbol{\xi} \leq \mathbf{0} \end{aligned}$$

In the Soft-SVMs problem we essentially relax our inequallity constraints introducing the variable $\boldsymbol{\xi}$. The constant λ determines how strict is the relaxation of the problem.
An equivalent problem is also :

$$\underset{\boldsymbol{\theta}}{\text{minimize}}\, f(\boldsymbol{\theta}) := \frac{\lambda}{2}\|\boldsymbol{\theta}\|_2^2 + \frac{1}{n}\sum_{i=1}^{n} h_{\mathbf{x}_i,y_i}(\boldsymbol{\theta})$$

With

$$h_{\mathbf{x}_i,y_i}(\boldsymbol{\theta}) = \max\{0, 1 - y_i\boldsymbol{\theta}^T\mathbf{x}_i\}$$

Stochastic Sub-GD is essentially SGD as described in Algorithm 3 with the difference that the cost function is non-deifferentiable and thus we use the sub-gradient as follows :

$$\partial h_{\mathbf{x}_i,y_i}(\boldsymbol{\theta}) = \begin{cases} -y_i\mathbf{x}_i, & 1 - y_i\boldsymbol{\theta}^T\mathbf{x}_i > 0 \\ 0, & \text{otherwise} \end{cases}$$

Then we compute $\boldsymbol{\xi}_k$ per batch using the subgradient and update $\boldsymbol{\theta}$ as follows :

$$\boldsymbol{\xi}_k = \frac{1}{|\mathcal{B}_k|}\sum_{j\in\mathcal{B}_k} \partial h_{\mathbf{x}_i,y_i}(\boldsymbol{\theta})$$

The update step becomes :

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \frac{2}{\lambda(k+1)}(\boldsymbol{\xi}_k + \lambda\boldsymbol{\theta}_k)$$

Choosing Epoch size $k = 100$ for $N = 2$, we plot the convergence rates of the SSG for different batch numbers.

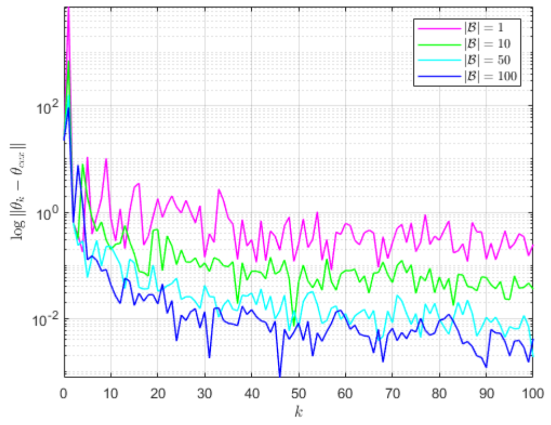

Fig. 6

We observe that as we increase the batch size the convergence rate increases. This is expected as the gradient is computed over a larger sample of data. If we use batch size equal to the number of data points, computation time can be very slow for larger datasets. On the other hand, if we choose batch size equal to 1, then the convergence rate is not very efficient. Also for larger datasets it is more noisy as shown in the figure below (n = 500).

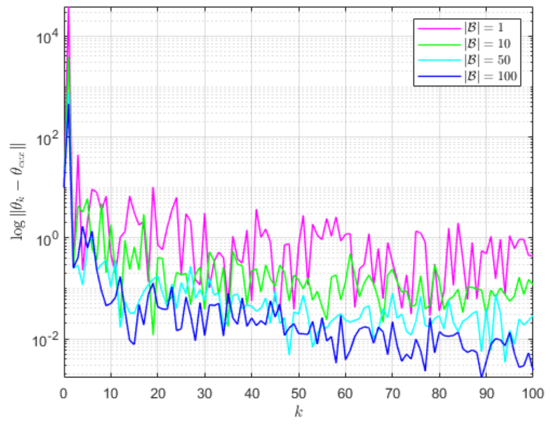

Fig. 7

D. In the final problem we will solve the dual homogeneous soft-SVMs problem using kernels. The dual function is the same as in problem C with the following difference.

$$g(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \boldsymbol{\kappa}(\mathbf{x}_i, \mathbf{x}_j)$$

First of all we generate a non-separable dataset as follows :

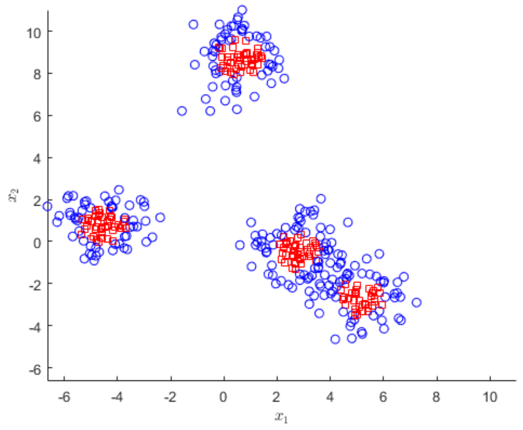

Fig. 8

We choose the Gaussian given by the following function :

$$\boldsymbol{\kappa}(\mathbf{x}, \mathbf{x}') = \exp\left(- \frac{\|\mathbf{x} - \mathbf{x}'\|_2^2}{2l^2} \right)$$

Choosing $l = 1$ we solve the dual problem using CVX and get the following 2-dimensional grid with every point being classified using :

$$y_{\text{new}} = \text{sign}((\mathbf{y} \odot \boldsymbol{\alpha}_*)^T \boldsymbol{\kappa}(\mathbf{X}, \mathbf{x}_{\text{new}}))$$

The classified grid gives us the decision boundary of the homogeneous soft-SVM problem using the Gaussian kernel.

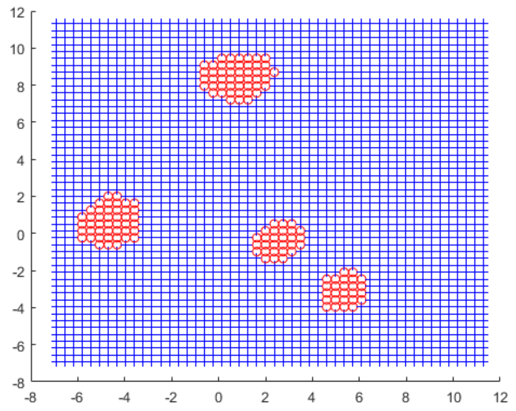

Fig. 9