# Machine Learning Mid-term Assignment 02

18M30396 Cho Bunhin

August 2, 2018

Problems done: problem 1(all), problem 3(1,2,3), problem 6(all).
MATLAB codes can be found at
`https://github.com/chobunhin/ART.T458-Machine-Learning-Assignment`

## Problem 1

Linear logistic regression with Batch steep descent

- For $i$-th sample, corresponding function value and gradient read

$$J_i(w) = \ln(1 + \exp(-y_i w^T x_i)), \tag{1}$$

$$\frac{\partial J_i}{\partial w}(w) = -\frac{\exp(-y_i w^T x_i)}{1 + \exp(-y_i w^T x_i)} y_i x_i, \tag{2}$$

- for $t$-th iteration, we update $w$ by a batch of samples as well as the $L^2$ penalty:

$$w^{t+1} = w^t - \eta \cdot \left( \sum_{i \in I^t} \frac{\partial J_i}{\partial w} + \lambda w^t \right). \tag{3}$$

Linear logistic regression with AdaGrad method

- Diagonal Hessian

$$H_t = diag(G_t^{1/2}), \tag{4}$$

where $\{G_t\}_i = \sum_{\tau=1}^{t} \{g^\tau\}_i^2$, and $g$ is the gradient of full $J(w)$.

- Accumulated gradient

$$d_t = \sum_{\tau=1}^{t} g_\tau \tag{5}$$

- Updating rule:

$$w^{t+1} = w^t - \eta_0 H_t^{-1} d_t \tag{6}$$

MATLAB code see `problem1.m`
Comparisons: Toy Dataset II; Running spec see `problem1.m`; We observed that AdaGrad method converges considerably faster than batch steepest descent method.

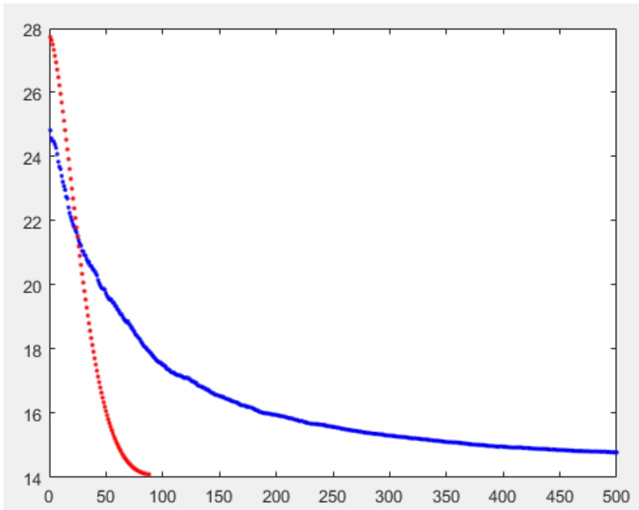


Figure 1: Function values comparison: blue-batch, red-adagrad

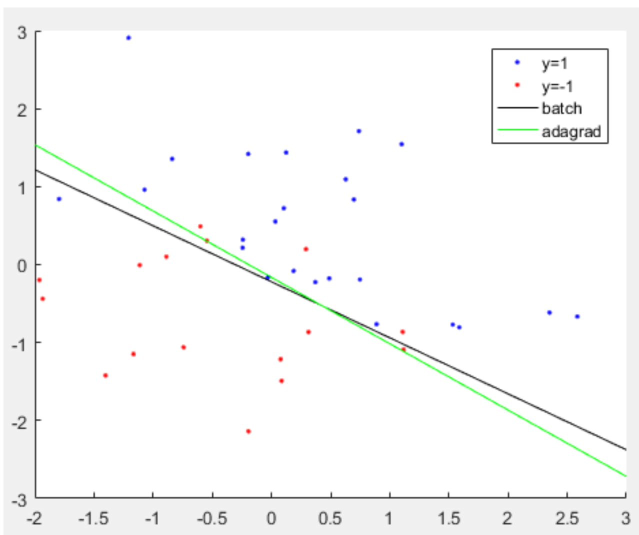


Figure 2: Separation results comparison

## Problem 3

**proof of 1.** The original problem is equivalent to

$$\begin{cases} \min_{w\in\mathbb{R}^d, z\in\mathbb{R}^n} & z^T\mathbf{1} + \lambda\|\mathbf{w}\|_\mathbf{2}^\mathbf{2} \\ \text{s.t.} & z \geq 0 \\ & z_i + y_i w^T x_i - 1 \geq 0 \end{cases} \tag{7}$$

with Lagrangian ($\mu \geq 0, \alpha \geq 0$):

$$L(w, z; \mu, \alpha) = \langle z, \mathbf{1}\rangle + \lambda\langle \mathbf{w}, \mathbf{w}\rangle - \langle \mu, \mathbf{z}\rangle - \langle \alpha, \mathbf{z}\rangle - \sum_{\mathbf{i=1}}^{\mathbf{n}}(\mathbf{y_i}\langle \mathbf{w}, \mathbf{x_i}\rangle - \mathbf{1}). \tag{8}$$

Following standard dual problem generating, we have

$$\frac{\partial L}{\partial w} = 0 \Rightarrow \hat{w} = \frac{1}{2\lambda}\sum_{i=1}^{n}\alpha_i y_i x_i, \tag{9}$$

$$\frac{\partial L}{\partial z} = 0 \Rightarrow \mathbf{1} - \mu - \alpha = \mathbf{0}, \tag{10}$$

and the dual objective function now reads:

$$L(\hat{w}, \hat{z}; \mu, \alpha) = -\frac{1}{4\lambda}\langle \sum_{i=1}^{n}\alpha_i y_i x_i, \sum_{j=1}^{n}\alpha_j y_j x_j\rangle + \langle \alpha, \mathbf{1}\rangle, \tag{11}$$

which gives the conclusion that

$$K_{ij} = y_i y_j \langle x_i, x_j\rangle, \tag{12}$$

and the dual problem reads:

$$\begin{cases} \max_{\alpha\in\mathbb{R}^n} & -\frac{1}{4\lambda}\langle \alpha, K\alpha\rangle + \langle \alpha, \mathbf{1}\rangle \\ \text{s.t.} & \mathbf{0} \leq \alpha \leq \mathbf{1} \end{cases} \tag{13}$$

□

**2.** KKT condition $\Rightarrow \frac{\partial L}{\partial w} = 0 \Rightarrow \hat{w} = \frac{1}{2\lambda}\sum_{i=1}^{n}\alpha_i y_i x_i$ according to (9). □

**3.** MATLAB code see `problem3.m`
Settings: Toy Dataset II; number of samples: 60; learning rate $\eta_t = \eta = 0.005$; iterate 1000 times. The primal-dual objective values as well as final result are shown in figure 3 and figure 4.

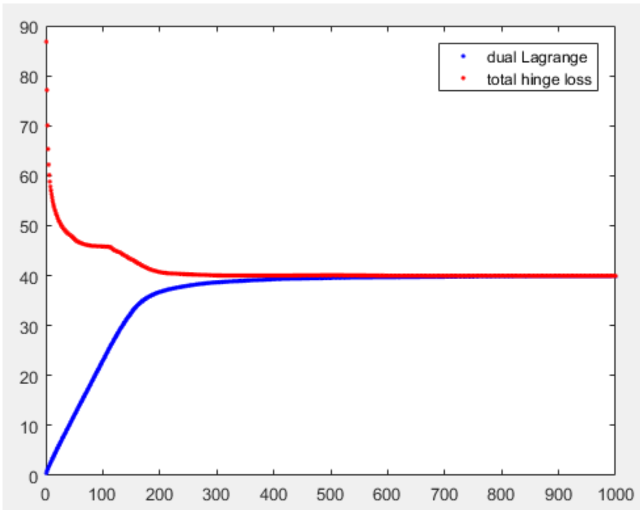


Figure 3: primal dual objective value wrt. iterate

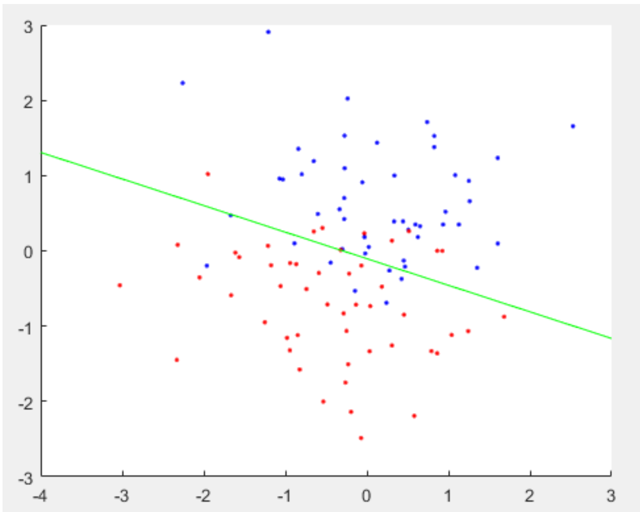


Figure 4: result given by SVM

# Problem 6

**1. nuclear norm** For matrix $Z \in \mathbb{R}^{m \times n}$, let the singular values of $Z$ be

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{\min(m,n)}, \tag{14}$$

then the nuclear norm of $Z$ reads

$$\|Z\|_* = \sum_{j=1}^{\min(m,n)} \sigma_j. \tag{15}$$

$\square$

**2. proximal operator of nuclear norm** Let $Z = U\Sigma V^T$ be the SVD of $Z$, where $\Sigma = diag\{\sigma_1, \sigma_2, \ldots, \sigma_{\min(m,n)}\}$. Then proximal operator

$$\begin{aligned} Prox_{\lambda\|\cdot\|_*}(Z) &= \arg\min_X \frac{1}{2}\|X - Z\|_F^2 + \lambda\|X\|_* &(16)\\ &= U \cdot diag\{S_\lambda(\sigma_1), \ldots, S_\lambda(\sigma_{\min(m,n)})\} \cdot V^T, &(17)\end{aligned}$$

where soft shrinkage $S_\lambda(\sigma) = \max(\sigma - \lambda, 0)$. $\square$