Name: YongKyung Oh
Andrew ID: yongkyuo

Machine Learning for Text Mining

Homework 4

1. Statement of Assurance

1. Did you receive any help whatsoever from anyone in solving this assignment? Yes / **No**.

If you answered 'yes', give full details? (e.g."Jane explained to me what is asked in Question 3.4").

2. Did you give any help whatsoever to anyone in solving this assignment? Yes / **No**.

If you answered 'yes', give full details? (e.g. "I pointed Joe to section 2.3 to help him with Question 2").

3. Did you find or come across code that implements any part of this assignment? Yes / **No**.

If you answered 'yes', give full details? (e.g. book & page, URL & location within the page, etc)

2. Writeup (40 pts)

(1) [10 pts] Gradient

$$f(\boldsymbol{w}) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + \frac{\lambda}{n}\sum_{i=1}^{n}\max(1 - y_i\boldsymbol{w}^T\boldsymbol{x}_i, 0)^2 = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + \frac{\lambda}{n}L$$

$$\nabla f(\boldsymbol{w}) = \boldsymbol{w} + \frac{\lambda}{n}\nabla L$$

Here L is non differentiable function, so we should conduct sub-gradient method. Assume that $g(\boldsymbol{x}_i) = \boldsymbol{w}^T\boldsymbol{x}_i = \sum_j^d (w_i x_{ij})$ where $\boldsymbol{x}_i \in \mathbb{R}^d$

$$L = \sum_{i=1}^{n}(1 - y_i * g(\boldsymbol{x}_i))^2 \ \ where\ \forall i \in I$$

$$\nabla L = \sum_{i=1}^{n} 2(1 - y_i * g(\boldsymbol{x}_i))\left(-y_i * \frac{\partial g}{\partial \boldsymbol{w}}\right)$$

$g(\boldsymbol{x}_i)$ is dot product and it can be re-written as follow.

$$g(\boldsymbol{x}_i) = \sum_j^d (w_i x_{ij}) = sum(\boldsymbol{w} \otimes \boldsymbol{x}) = u = sum(v)$$

$$\frac{\partial \boldsymbol{v}}{\partial \boldsymbol{w}} = \frac{\partial(\boldsymbol{w} \otimes \boldsymbol{x})}{\partial \boldsymbol{w}} = diag(\boldsymbol{x})$$

$$\frac{\partial u}{\partial \boldsymbol{v}} = \frac{\partial(sum(\boldsymbol{v}))}{\partial \boldsymbol{v}} = \vec{1}^T$$

$$\frac{\partial u}{\partial \boldsymbol{w}} = \frac{\partial u}{\partial \boldsymbol{v}}\frac{\partial \boldsymbol{v}}{\partial \boldsymbol{w}} = \vec{1}^T * diag(\boldsymbol{x}) = \boldsymbol{x}^T$$

Therefore, our original equation can be changed as follow

$$L = \sum_{i=1}^{n} (1 - y_i \boldsymbol{w}^T \boldsymbol{x}_i)^2 \ \ where \ \forall i \in I$$

$$\begin{aligned} \nabla L &= \sum_{i=1}^{n} 2(1 - y_i \boldsymbol{w}^T \boldsymbol{x}_i)(-y_i \boldsymbol{x}_i^T) \\ &= \sum_{i=1}^{n} 2\boldsymbol{x}_i^T (y_i \boldsymbol{w}^T \boldsymbol{x}_i y_i - y_i) \\ &= \sum_{i=1}^{n} 2\boldsymbol{x}_i^T (\boldsymbol{w}^T \boldsymbol{x}_i - y_i) \\ &= 2\boldsymbol{X}_I^T (\boldsymbol{X}_I \boldsymbol{w} - \boldsymbol{y}_I) \end{aligned}$$

Since $y_i \in \{-1.1\}$, so $y_i^2 = 1$.

$$\nabla f(\boldsymbol{w}) = \boldsymbol{w} + \frac{\lambda}{n} \nabla L = \boldsymbol{w} + \frac{2\lambda}{n} \boldsymbol{X}_I^T (\boldsymbol{X}_I \boldsymbol{w} - \boldsymbol{y}_I)$$

(2) [10 pts] Hessian

$$\nabla f(\boldsymbol{w}) = \boldsymbol{w} + \frac{\lambda}{n} \nabla L = \boldsymbol{w} + \frac{2\lambda}{n} \boldsymbol{X}_I^T (\boldsymbol{X}_I \boldsymbol{w} - \boldsymbol{y}_I)$$

$$\begin{aligned} \nabla^2 f(\boldsymbol{w}) &= \frac{\partial \boldsymbol{w}}{\partial \boldsymbol{w}} + \frac{\partial}{\partial \boldsymbol{w}} \left\{ \frac{2\lambda}{n} \boldsymbol{X}_I^T (\boldsymbol{X}_I \boldsymbol{w} - \boldsymbol{y}_I) \right\} \\ &= I_d + \frac{2\lambda}{n} \boldsymbol{X}_I^T (\boldsymbol{X}_I) \\ &= I_d + \frac{2\lambda}{n} \boldsymbol{X}^T D \boldsymbol{X} \end{aligned}$$

Where I_d is the identity matrix of dimension d and $D \in \mathbb{R}^{n \times n}$ is a diagonal matrix with the following elements.

$$D_{ii} = \begin{cases} 1, & if \ i \in I \\ 0, & otherwise \end{cases}$$

(3) [10 pts] Optimality

$$\min_{\boldsymbol{w},\xi} \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + \frac{\lambda}{n}\sum_{i=1}^{n}\xi^2$$
$$s.t.\ \ y_i\boldsymbol{w}^T\boldsymbol{x}_i \geq 1 - \xi_i$$
$$\xi_i \geq 0$$
$$\xi_i \geq 1 - y_i\boldsymbol{w}^T\boldsymbol{x}_i$$

When $1 - y_i\boldsymbol{w}^T\boldsymbol{x}_i > 0$, $\xi_i = 1 - y_i\boldsymbol{w}^T\boldsymbol{x}_i$, because of the objective function is min.

$$\begin{cases} 1 - y_i\boldsymbol{w}^T\boldsymbol{x}_i > 0 \ \rightarrow \min \xi^2 = (1 - y_i\boldsymbol{w}^T\boldsymbol{x}_i)^2 \\ 1 - y_i\boldsymbol{w}^T\boldsymbol{x}_i = 0 \ \rightarrow \min \xi^2 = 0 \\ 1 - y_i\boldsymbol{w}^T\boldsymbol{x}_i < 0 \ \rightarrow \min \xi^2 = 0 \geq 0 \end{cases}$$
$$\therefore \xi_i = \max(0, 1 - y_i\boldsymbol{w}^T\boldsymbol{x}_i)$$

Substitute to the original equation

$$f(\boldsymbol{w}) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + \frac{\lambda}{n}\sum_{i=1}^{n}\max(1 - y_i\boldsymbol{w}^T\boldsymbol{x}_i, 0)^2$$

(4) [10 pts] Algorithm Pseudo Code

(Outline the pseudo code of the optimization update procedure for mini-batch stochastic gradient method and Newton method)

Mini-batch stochastic gradient method (Pegasos Algorithm)[1]

Input: a list of feature vectors X; a list of output y; regularization parameter λ

Output: weight vector w

$w = (0,0, \ldots 0)$

for t in max_iteration

select a position i randomly (mini-batch set with batchsize n)

$\eta = \frac{1}{\lambda t}$ step_size (learning rate with decreasing factor)

$score = y_i(\boldsymbol{w}^T\boldsymbol{x}_i)$

if $score < 1$

$\boldsymbol{w} = (1 - \eta\lambda)\boldsymbol{w} + (\eta y_i)x_i$

else

$\boldsymbol{w} = (1 - \eta\lambda)\boldsymbol{w}$

the end result is $\boldsymbol{w}$

[1] https://svn.spraakdata.gu.se/repos/richard/pub/ml2015_web/a2_clarification.pdf

Newton method[2]

Input: a list of feature vectors X; a list of output y; regularization parameter λ

Output: weight vector w

$w = (0,0, \dots 0)$

for t in max_iteration

if $f(w_t)$ is sufficiently small **then**

$\boldsymbol{w = w_t}$

return $\boldsymbol{w^*}$

end

$w_{t+1} = w_t - \frac{f(w_t)}{f\prime(w_t)}$

if $|w_{t+1} - w_t|$ is sufficiently small **then**

$\boldsymbol{w = w_{t+1}}$

return $\boldsymbol{w^*}$

end

the end result is $\boldsymbol{w}$

[2] https://www.math.usm.edu/lambers/mat460/fall09/lecture10.pdf

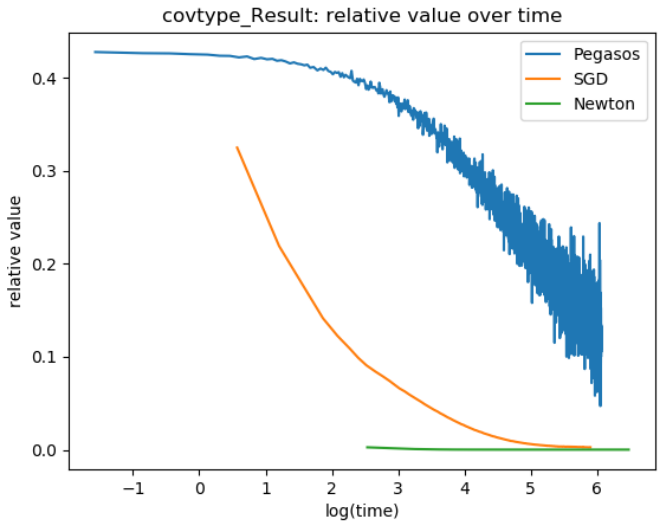
3. Experiments (20 pts)

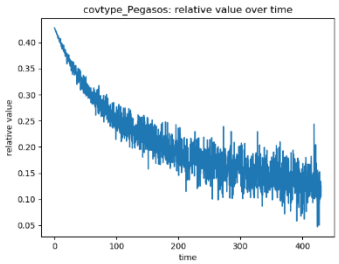
Plot the figures for **both** of two datasets and **both** the approaches

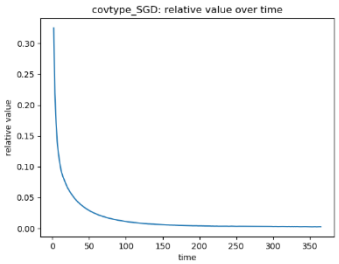
I conducted both mini-batch SGD (Pegasos) and SGD to compare the result. Three different approaches are plotted with log(time) axis.

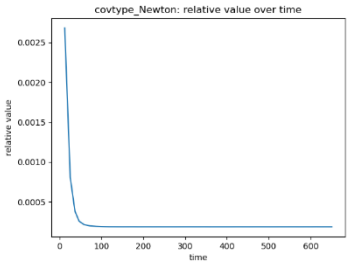
(1) [5 pts] Relative function value difference versus training time

<contype>

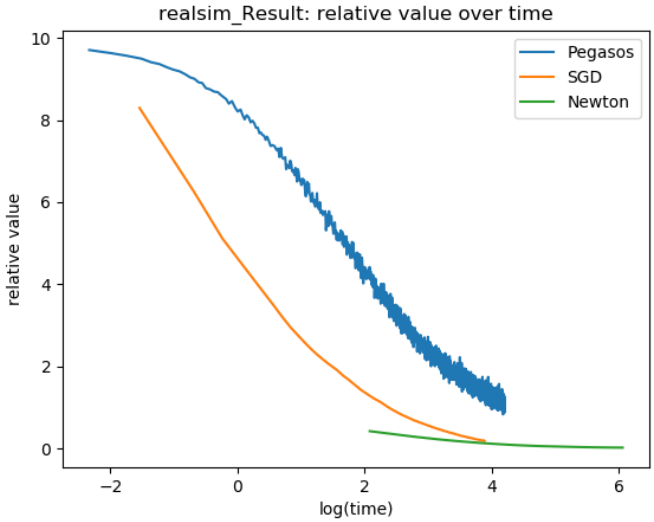


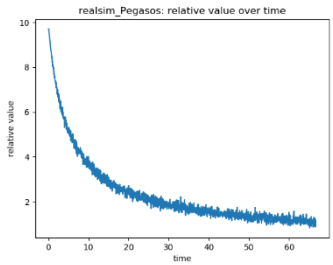


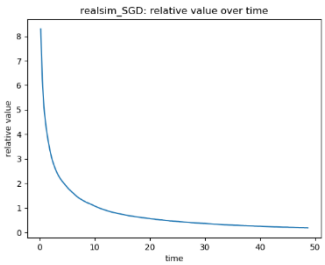


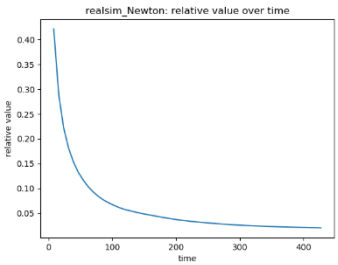


<realsim>


realsim_Result: relative value over time
Pegasos
SGD
Newton
relative value
log(time)

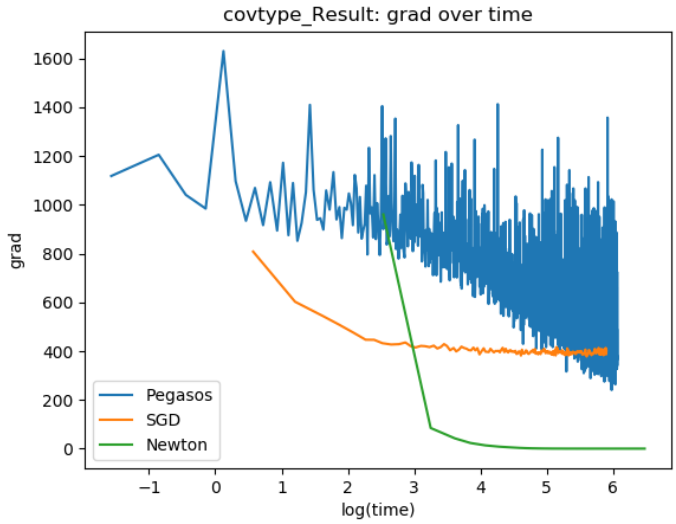

realsim_Pegasos: relative value over time
relative value
time

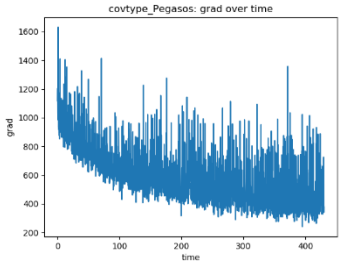

realsim_SGD: relative value over time
relative value
time

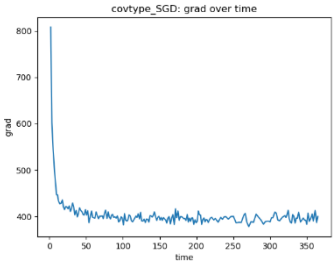

realsim_Newton: relative value over time
relative value
time

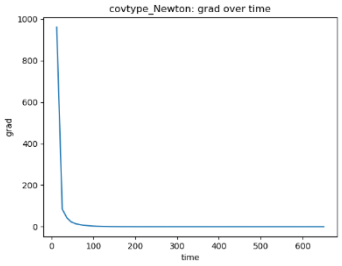
(2) [5 pts] Gradient norm versus training time

<contype>

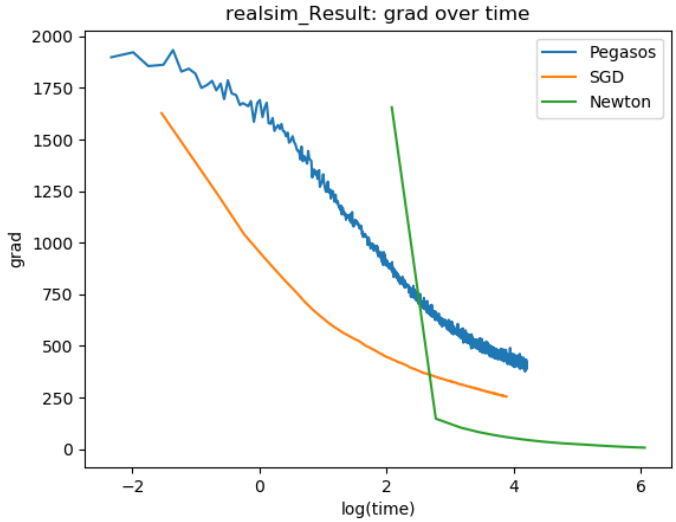


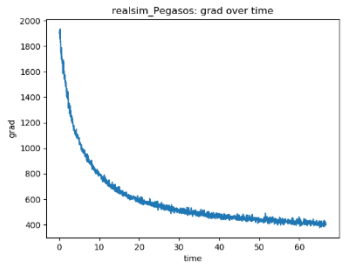


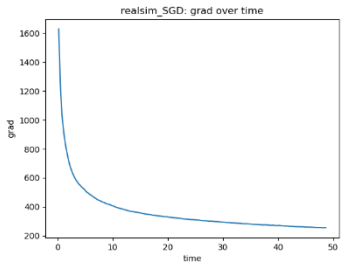


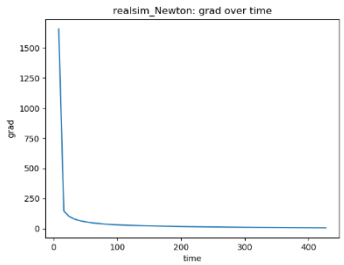


<realsim>


realsim_Result: grad over time
Pegasos
SGD
Newton
grad
log(time)

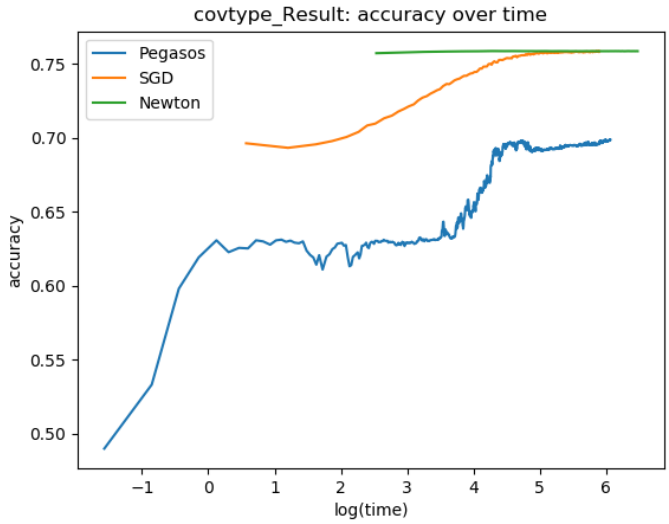

realsim_Pegasos: grad over time
grad
time

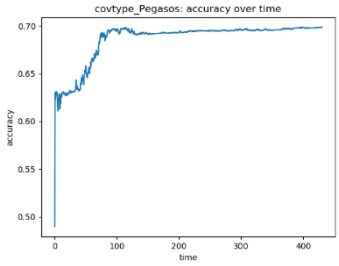

realsim_SGD: grad over time
grad
time

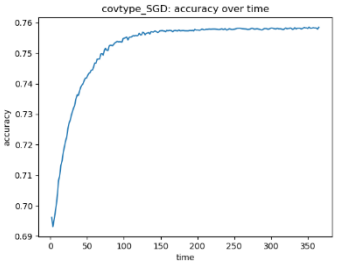

realsim_Newton: grad over time
grad
time

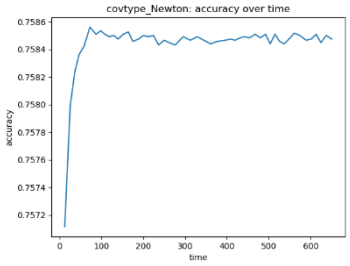
(3) [5 pts] Test set accuracies versus training time

<contype>

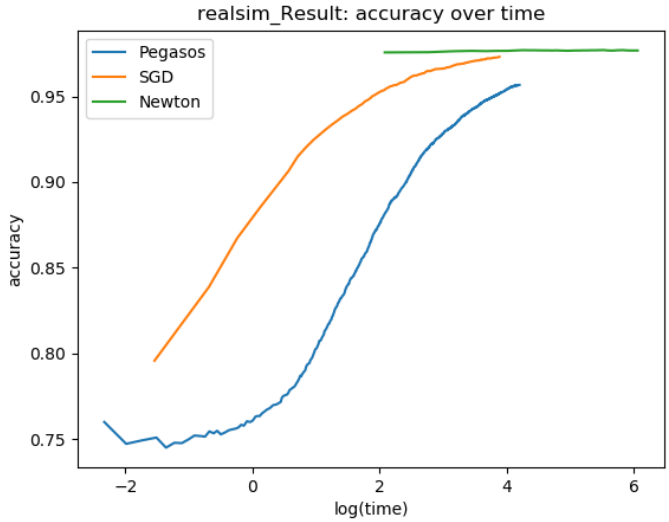


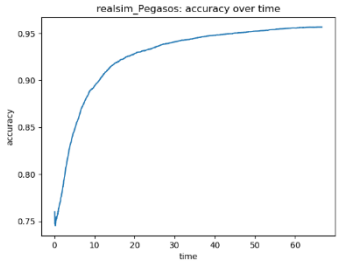


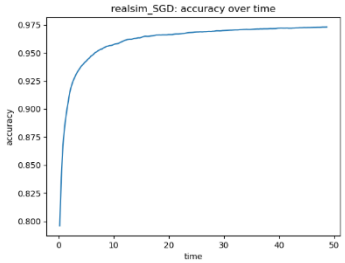


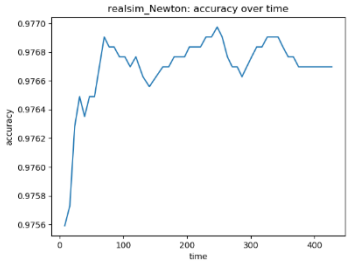


<realsim>


realsim_Result: accuracy over time
Pegasos
SGD
Newton
accuracy
log(time)


realsim_Pegasos: accuracy over time
accuracy
time


realsim_SGD: accuracy over time
accuracy
time


realsim_Newton: accuracy over time
accuracy
time

(4) [5 pts] Discuss the difference between mini-batch SGD and Newton method in terms of the three types of figures

- All methods work well and almost reach the target values. Mini-batch SGD requires more iteration to converge. Appropriate stopping rule is needed.
- We use the conjugate gradient method in the Newton's method. Therefore, Newton's method is converging faster than mini batch gradient decent or SGD.
- In the mini-batch gradient, there are fluctuation in the gradient value because of the variation of each batches. Annealing the learning rate help to converge comparatively stable.
- Newton's method is a second order optimization technique and it can sometimes outperform than stochastic gradient decent. In some cases, it requires more computation cost. So, appropriate method should be considered.