Name: Zihao **ZHOU**
Andrew ID: zihaozho

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.

Ta Zihang Dai helps me to solve the code problem in OH(bias and the efficient use of Numpy)

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

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

2. Writeup (40 pts)

(1) [10 pts] Gradient

(1) Proof

$$\min f(w) = \frac{1}{2}w^Tw + \frac{\lambda}{n}\sum_{i=1}^{n}\max(1-y_iw^Tx_i,0)^2$$

for every item in $\max(1-y_iw^Tx_i,0)$, if $1-y_iw^Tx_i<0$, $\max(1-y_iw^Tx_i,0)=0$,

so, original formula

$$= \frac{1}{2}ww^T + \frac{\lambda}{n}\sum_{i=1}^{n}(1-2y_iw^Tx_i+y_iw^Tx_ix_i^Tw)_I$$

$$= \frac{1}{2}ww^T + \frac{\lambda}{n}(I_I - 2y_I^Tx_Iw + w^Tx_I^Tx_Iw)$$

derivation of w:

$$\nabla f(w) = \nabla\left(\frac{1}{2}ww^T\right) + \frac{\lambda}{n}\nabla(-2y_I^Tx_Iw + w^Tx_I^Tx_Iw)$$

$$= w - \frac{2\lambda}{n}x_I^Ty + \frac{2\lambda}{n}x_I^Tx_Iw$$

$$= w + \frac{2\lambda}{n}x_{I,:}^T(x_{I,:}w - y_I)$$

prove up

(2) [10 pts] Hessian

(2) Proof

From (1), we got:

$$\nabla f(w) = w + \frac{2\lambda}{n} X_I^T X_I w - \frac{2\lambda}{n} X_I^T y$$

$$\nabla^2 f(w) = \nabla(\nabla f(w))$$

$$= \nabla(w) + \nabla\left(\frac{2\lambda}{n} X_I^T X_I w\right)$$

$$= I_d + \frac{2\lambda}{n} \nabla(X_I^T X_I w)$$

for $i \notin I$, the second is 0,

for $i \in I$, $\nabla(x_i^T x_i w) = x_i^T x_i$,

Therefore:

$$\nabla^2 f(w) = Id + \frac{2\lambda}{n} X^T \Sigma X,$$

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

(3) [10 pts] Optimality

(3) Proof

Let $1 - y_i w^T x_i = \beta \xi_i$, $\xi_i \geq 0$, $\beta \in [0,1]$

$\beta \xi_i \leq \xi_i$, thus the following conditions are met:

$$y_i w^T x_i = 1 - \beta \xi_i \geq 1 - \xi_i$$
$$\xi_i \geq 0$$

Therefore,

$$\min_w \frac{1}{2} w^T w + \frac{\lambda}{n} \sum_{i=1}^{n} \max(1 - y_i w^T x_i, 0)^2$$

can be written as:

$$\min_{w,\xi} \frac{1}{2} w^T w + \frac{\lambda}{n} \sum_{i=1}^{n} \xi_i^2$$

(4) [10 pts] Algorithm Pseudo Code

(4) Algorithm Pseudo Code

1. mini-batch SGD

Input: $n, \lambda, T, \gamma_0, \beta$

Initialization: $w \leftarrow 0$ (with bias)

```
for t in T:
    γ_t = γ_0 / (1 + β t)
    for I in mini-batches:
        grad ← w + (2λ/n) X_{I,:}^T (X_{I,:} w - y_I)
        w ← w - γ_t · grad
    end
end
```

2. Newton method

Input: n, λ, T

Initialization: $w \leftarrow 0$ (with bias)

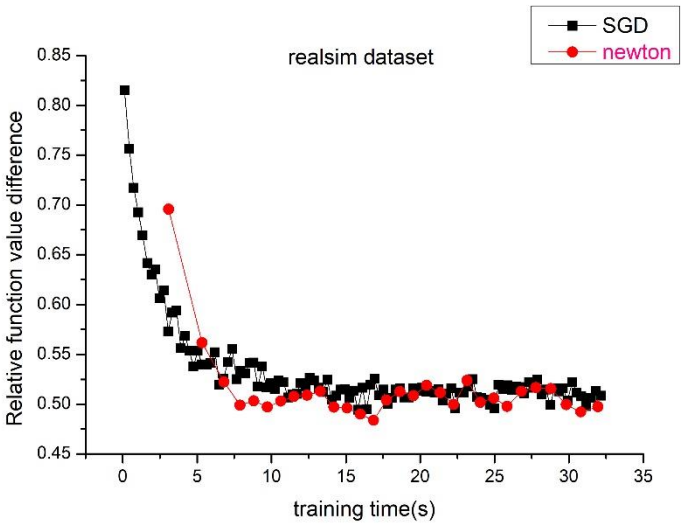
```
for t in T:
    ∇f(w) ← w + (2λ/n) X_{I,:}^T (X_{I,:} w - y_I)
    d ← -[∇²f(w)]^{-1} · ∇f(w)
    w ← w + d
end
```

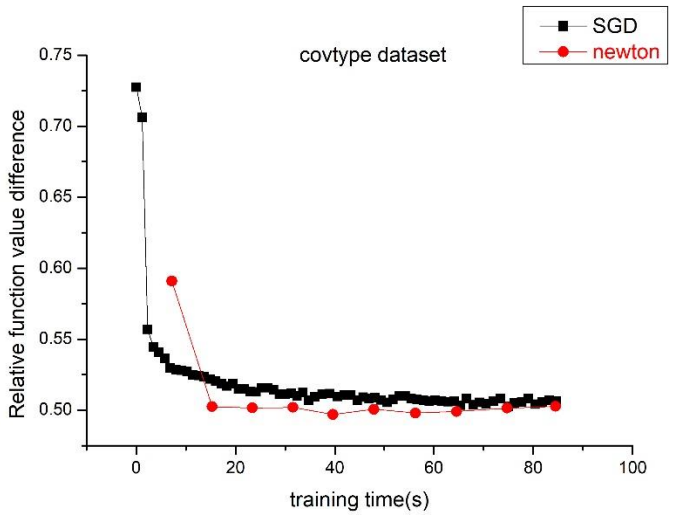
Output: $w^{(T)}$

3. Experiments (20 pts)

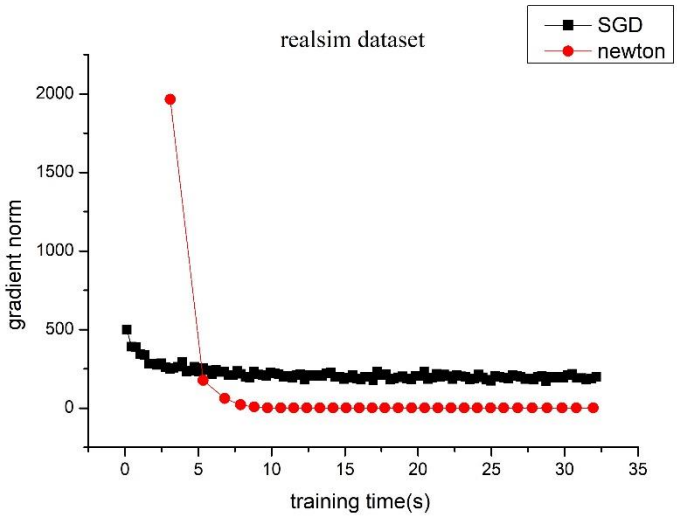
Plot the figures for **both** of two datasets and **both** the approaches

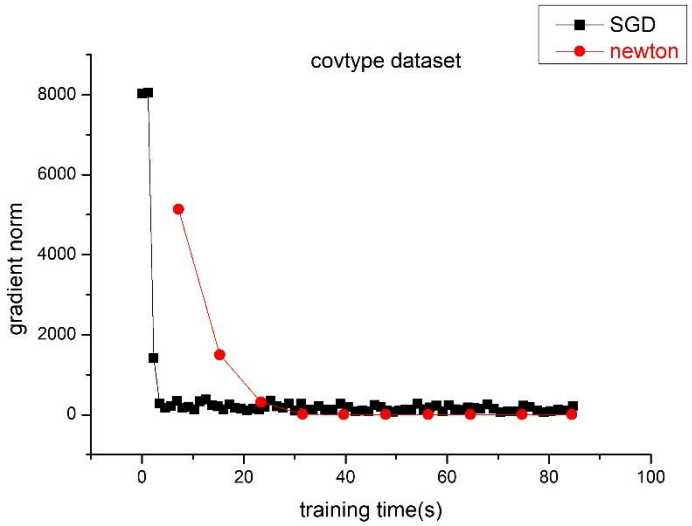
(1) [5 pts] Relative function value difference versus training time



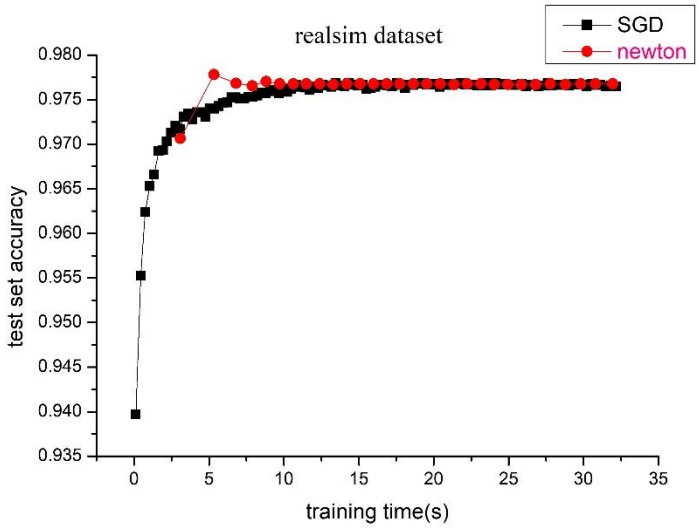


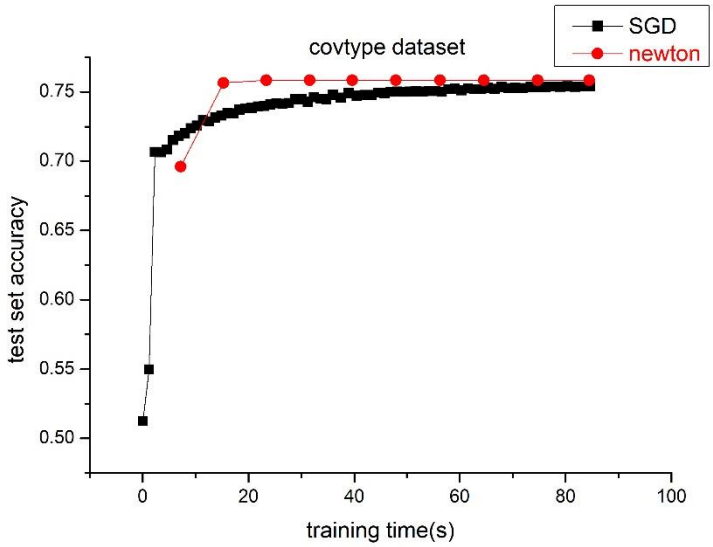
(2) [5 pts] Gradient norm versus training time





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





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

From the perspective of the behavior of Relative function value difference (RFVD):

With the increase of training time, for the Newton method, the RFVD drops rapidly after only one update and reach the optimal value at 4 updates (7.5s) in realism and 2 updates(15s) in covtype.

For mini-batch SGD, the RFVD drops gradually to reach the optimal value. It will take about 15s for realism and 50s for covtype to reach the optimal value.

From the perspective of the behavior of the gradient norm:

With the increase of training time, for the Newton method, the gradient norm also drops rapidly and reach the optimal value at about 5 updates(10s) in realism and 4 updates(30s) in covtype.

For mini-batch SGD, the gradient norm also drops gradually to reach the optimal value. It will take about 20s for realism but for covtype, it only takes 4 mini-batches (less than 3s) updates to reach the optimal.

From the perspective of the behavior of test set accuracy:

With the increase of training time, Newton method immediately gets a good result of the test set accuracy in both dataset after 2 updates, using 5s and 15s respectively.

For mini-batch SGD, the test set accuracies also increase gradually to reach the optimal value. It will take about 15s for realism and 80s for covtype,

In conclusion:

The time used in the Newton method is much shorter than SGD. There are many reasons for this:

1. Newton method uses **whole train data** to update the parameters every step. In the above six figures, every point for SGD is the result after one batch update(i.e. 1000 samples in realism and 10000 samples in coytype).

2. Newton method is a method for second-order convergence, which is faster than SGD's first-order convergence.

However, there are also some limitations for Newton method:

1.compuatational and memory resource

Allocate space for the whole dataset is impossible if we have a large dataset and iterate to get conjugate gradient is computationally complex.

2. Hessian should be a real and positive-definite matrix

If Hessian is not a real and positive-definite matrix, we can't get the inverse of Hessian and use the conjugate gradient method to get every update.