Name: Joao Miguel Coelho
Andrew ID: jmcoelho

Machine Learning for Text Mining

Homework 4 – Template

1. Statement of Assurance

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

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? Yes. This paper and this book helped me understand the different ways of expressing the optimization problem. This github repo helped me understand that the direction of my gradient was wrong on pmf. I did not copy from any of the sources.

2. Writeup (40 pts)

(1) [10 pts] Gradient

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

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

The max operation is piece-wise linear.
In this case, it is not differentiable when $1 - y_i w^T x_i = 0$.
We can resort to sub-gradients:

- $1 - y_i w^T x_i > 0$:

  $\nabla_w \max(1 - y_i w^T x_i, 0)^2 = 2(1 - y_i w^T x_i)(-y_i x_i^T)$

- $1 - y_i w^T x_i < 0$:

  $\nabla_w \max(1 - y_i w^T x_i, 0)^2 = 0$

As such, using $I \equiv \{i : 1 - y_i w^T x_i > 0\}$, and $y_i \in \{-1, 1\} \Rightarrow y_i^2 = 1$:

$$\nabla f(w) = w + \frac{2\lambda}{n} \sum_{i \in I} (1 - y_i w^T x_i)(-y_i x_i^T)$$

$$= w + \frac{2\lambda}{n} \sum_{i \in I} -y_i x_i^T + y_i^2 w^T x_i x_i^T$$

$$= w + \frac{2\lambda}{n} \sum_{i \in I} -y_i x_i^T + 1\, w^T x_i x_i^T$$

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

(2) [10 pts] Hessian

$$\nabla^2 f(w) = \nabla_w(w) + \nabla_w\left(\frac{2\lambda}{n} X_{I,:}^T (X_{I,:} w - y_I)\right)$$
$$= I_d + \frac{2\lambda}{n} \nabla_w \left(X^T D X w - X_{I,:}^T y_I\right)$$
$$= I_d + \frac{2\lambda}{n} \nabla_w \left(X^T D X w\right)$$
$$= I_d + \frac{2\lambda}{n} X^T D X \quad \blacksquare$$

(3) [10 pts] Optimality

The two are equivalent formulations of the primal optimization problem. Both try to minimize the sum between the regularization term and the loss term.

In the unconstrained problem, the hinge loss is $max(1 - y_i\, w^T x_i, 0)^2$. In the constrained problem, the first term is constrained to be ≥ 0 by leveraging the slack variables ξ_i.

If there is a solution to the constrained problem, then it should satisfy both constraints. Following constraint 1, if $y_i\, w^T x_i \geq 1 - \xi_i$, then $1 - y_i\, w^T x_i \leq \xi_i$. Constraint 2 implies that $\xi_i \geq 0$, so $\xi_i = max(1 - y_i\, w^T x_i, 0)$, i.e., the minimization process ensures that $\xi_i = max(1 - y_i\, w^T x_i, 0)$. As such, if (w, ξ) is a solution to the constrained problem, the unconstrained problem will also have a minimum value at w. In a similar fashion, the same applies to a potential solution x to the unconstrained problem.

Summarily, both optimization problems are equivalent in their minimization objective. The constrained version leverages slack variables to ensures non-negativity of hinge loss, aligning the optimal function values.

(4) [10 pts] Algorithm Pseudo Code

Assuming bias terms are already present on X and y as an additional dimension for both approaches.

Mini batch SGD:
Input: X, y, N, w, lr, bs
Initialization: w~Gaussian(0,1); Split (X, y) in batches of size bs
For N epochs:
    For each batch (Xi, yi):
        $w \leftarrow w - lr\nabla f(Xi, yi; w)$

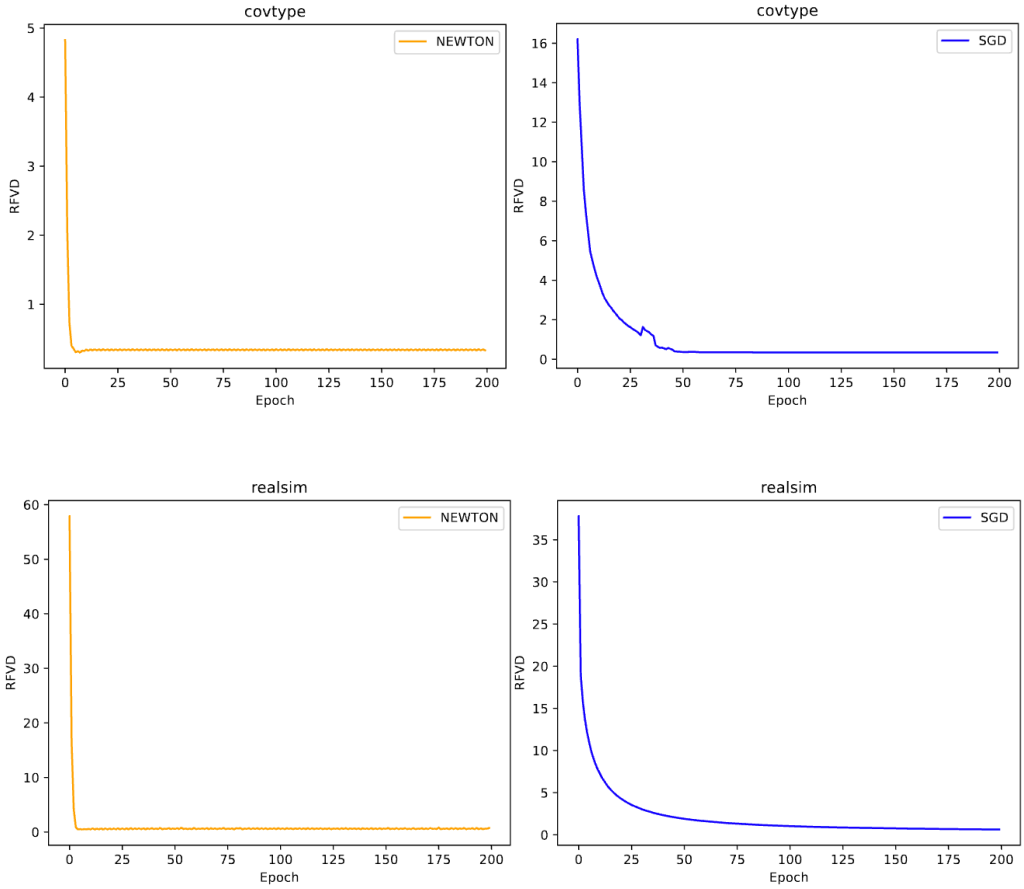
Newton-Raphson:
Input: X, y, N, w
Initialization: w~Gaussian(0,1);
For N epochs:
    $d = (\nabla^2 f(X, y;w))^{-1}\nabla f(X, y; w)$
    $w \leftarrow w + d$

3. Experiments (20 pts)

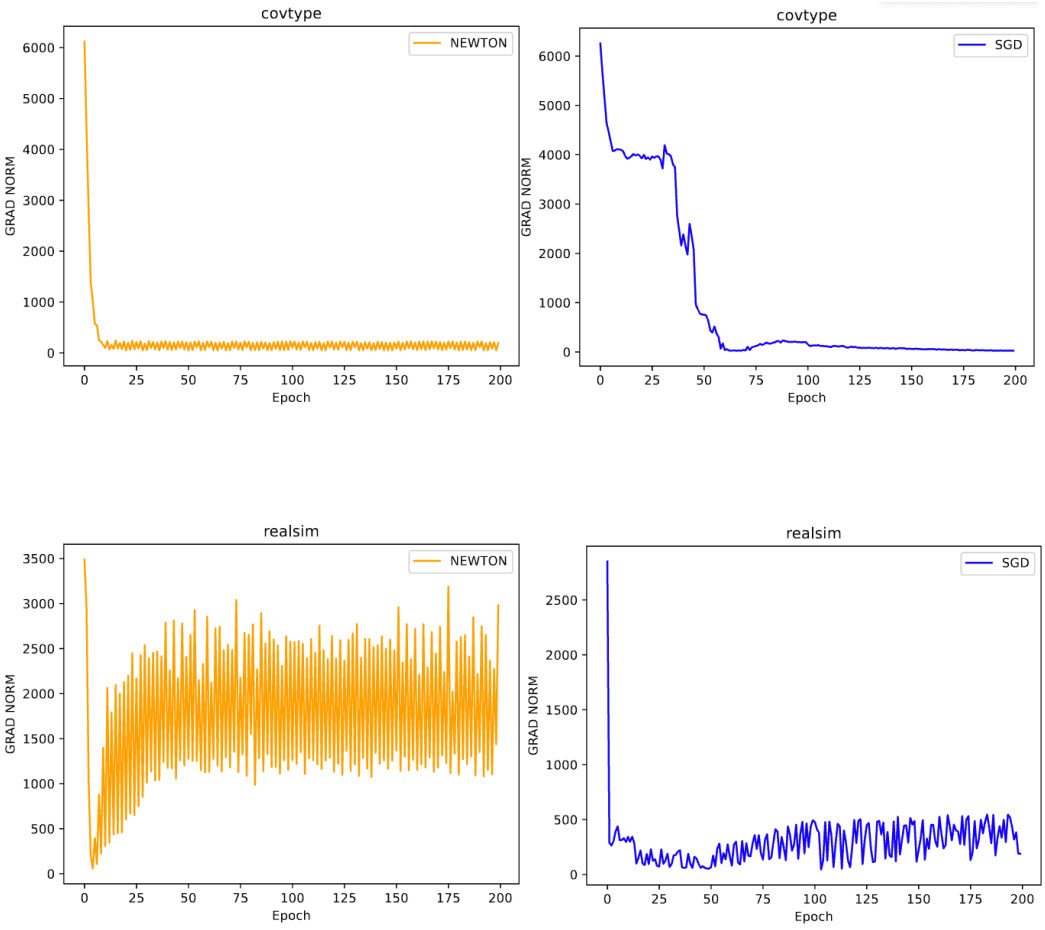
Plot the figures for **both** of two datasets and **both** the approaches

[next page]

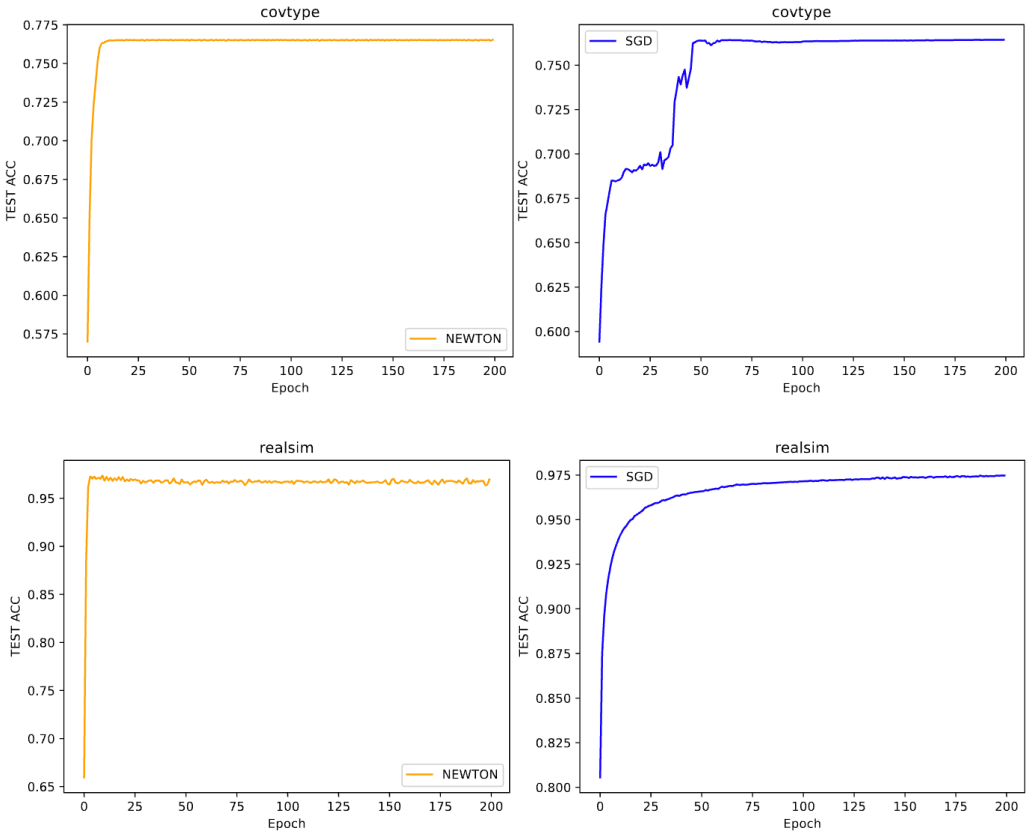
(1) [5 pts] Relative function value difference versus training time



(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

First, both approaches achieve accuracies that are similar to what is described on the statement:
Accuracy with SGD on realsim: 0.9746
Accuracy with NEWTON on realsim: 0.9697
Accuracy with SGD on covtype: 0.7643
Accuracy with NEWTON on covtype: 0.7653

Both optimization techniques converge to very similar values, most likely due to the fact that the L2 SVM loss is convex. In terms of run-time, SGD is faster than Newton, as expected. On the realism dataset, SGD runs 57 epochs per second, while newton runs 9 epochs per second. On covtype, SGD runs 11 epochs per second, and newton 1 epoch per second. This is due to the newton method processing the whole data at once, and requires heavier computations (approximation of the Hessian matrix).

Regarding the three types of plots:
Starting by the test accuracy per epoch, we can see that the Newton method converges faster, i.e., a couple epochs were enough for the model to reach the optimal accuracy value. SGD needed roughly 50 epochs. This may be because SGD method takes heuristically based steps (based on the learning rate) and uses mini-batches, while the newton method leverages the whole dataset (which may be more informative than mini-batches), and is a second-order optimization algorithm, hence taking into account the curvature of the loss landscape when computing the update step. This behavior was observed for both datasets.

As for the gradient norm, this metric can help identify training stability problems, i.e., it can be seen as a proxy as to how well the algorithm is navigating the loss landscape. For the covtype dataset, alike the test accuracy, the newton method converges very early, and SGD requires more epochs to reach the low values. This can be due to the same intuition as above, where the more informed steps taken by the Newton method resulted in a faster reduction, suggesting a good navigation towards the optimum, while the SGD exhibits more variability due to the minibatches, due to this being an approximation that introduces noise to the computations. As for the realism dataset, we can see that the gradient is very noisy for the Newton method, and also somewhat noisy for the SGD method. This can be due e.g. to instabilities when computing the Hessian for the Newton method, and noise in the data for both methods.

Finally, looking into the relative function value difference, we can again see signs of successful converge for both approaches in both datasets, as the relative difference between f(w) and f(w*) approaches 0, faster in terms of epochs for the Newton method for the reasons above.