Homework Assignment 1

SDS 385 Statistical Models for Big Data

Part of the solutions are due to Y. Xie

Please upload the HW on canvas before class Oct 11th by 10am. Please type up your homework using latex. We will not accept handwritten homeworks.

1. (10 pts) **Convex functions:** Using the definition of convex function, i.e. $f(tx+(1-t)y) \leq tf(x)+(1-t)f(y)$ show that the following functions are convex.

   (a) (3pts) e^x

   Proof. For any fixed x and y, $\forall t \in [0,1]$, let

$$g(t) \triangleq f(tx+(1-t)y) - tf(x) - (1-t)f(y) = e^{tx+(1-t)y} - te^x - (1-t)e^y$$

   By definition, $e^x = 1 + \sum_{i=1}^{\infty} \frac{x^i}{i!}$. Without loss of generality, suppose $x \geq y$, we have

$$\begin{aligned} g(t) &= e^y(e^{t(x-y)} - te^{x-y} - 1 + t) \\ &= e^y\left(1 + \sum_{i=1}^{\infty} \frac{t^i(x-y)^i}{i!} - t\left(1 + \sum_{i=1}^{\infty} \frac{(x-y)^i}{i!}\right) - 1 + t\right) \qquad \text{[by def of } e^x\text{]} \\ &= e^y \sum_{i=1}^{\infty} (t^i - t)\frac{(x-y)^i}{i!} \leq 0 \end{aligned}$$

   Since $\forall i \geq 1$, $t^i \leq t, \forall t \in [0,1]$ and $(x-y)^i \geq 0$, each term $(t^i - t)\frac{(x-y)^i}{i!} \leq 0$. Hence, $g(t) \leq 0, \forall t \in [0,1] \Rightarrow e^{tx+(1-t)y} \leq te^x + (1-t)e^y, \forall t \in [0,1] \Rightarrow e^x$ is convex. $\square$

   (b) (2pts) If $f(x)$ is convex for $x \in \Re^p$, show that so is $f(Ax+b)$ for $A \in \Re^{p\times p}$ and $b \in \Re^p$.

   Proof.

$$\begin{aligned} f(A(tx+(1-t)y)+b) &= f(t(Ax+b) + (1-t)(Ay+b)) \qquad [b = tb + (1-t)b] \\ &\leq tf(Ax+b) + (1-t)f(Ay+b) \qquad [f(x) \text{ is convex}] \end{aligned}$$

   $\square$

   (c) (2pts) If $f_i(x), i \in [k]$ are convex functions, show that the pointwise maximum, i.e. $g(x) = \max_{i\in[k]} f_i(x)$ is also convex.

Proof.

$$\begin{aligned}
g(tx + (1-t)y) &= \max_{i\in[k]} f_i(tx + (1-t)y) && [\text{by def of } g(x)] \\
&\le \max_{i\in[k]} tf_i(x) + (1-t)f_i(y) && [f_i(x) \text{ are convex}] \\
&\le \max_{i\in[k]} tf_i(x) + \max_{i\in[k]}(1-t)f_i(y) && [\max a + b \le \max a + \max b, \forall a, b] \\
&= t\max_{i\in[k]} f_i(x) + (1-t)\max_{i\in[k]} f_i(y) && [t > 0, 1-t > 0] \\
&= tg(x) + (1-t)g(y) && [\text{by def of } g(x)]
\end{aligned}$$

□

(d) (3 pts) Consider the logistic regression problem. For $x \in \Re^p$, You have

$$y \sim Bernoulli\left(\frac{1}{1+e^{-\theta^T x}}\right)$$

i. (1pt) Write down the log likelihood function.

Solution. The distribution is $\mathbb{P}(y|\theta, x) = (\frac{1}{1+e^{-\theta^T x}})^y(\frac{e^{-\theta^T x}}{1+e^{-\theta^T x}})^{1-y}$, then

$$\begin{aligned}
L(\theta) &= \log \mathbb{P}(y_1, \ldots, y_n | x_1, \ldots, x_n; \theta) = \log \prod_{i=1}^n \mathbb{P}(y_i|\theta, x_i) \\
&= \log \prod_{i=1}^n (\frac{1}{1+e^{-\theta^T x_i}})^{y_i}(\frac{e^{-\theta^T x_i}}{1+e^{-\theta^T x_i}})^{1-y_i} \\
&= \sum_{i=1}^n -y_i \log(1+e^{-\theta^T x_i}) + (1-y_i)\log(\frac{e^{-\theta^T x_i}}{1+e^{-\theta^T x_i}}) \\
&= \sum_{i=1}^n y_i\theta^T x_i - \log(1+e^{\theta^T x_i})
\end{aligned}$$

■

ii. (2pt) Show that this is concave. *Hint: for part d, you can use first/second order conditions and properties of convex functions to prove convexity*

Proof. Since $L(\theta)$ is twice-differentiable, take derivatives and consider the Hessian matrix:

$$\nabla L(\theta) = \sum_{i=1}^n y_i x_i - \frac{e^{\theta^T x_i}}{1+e^{\theta^T x_i}} x_i = \sum_{i=1}^n (y_i - p_i)x_i \quad [p_i = \frac{1}{1+e^{-\theta^T x_i}} = \frac{e^{\theta^T x_i}}{1+e^{\theta^T x_i}}]$$

$$\nabla p_i(\theta) = \frac{e^{-\theta^T x_i} x_i}{(1+e^{-\theta^T x_i})^2} = p_i(1-p_i)x_i$$

$$\begin{aligned}
H(\theta) = \nabla^2 L(\theta) &= \frac{\partial^2 L(\theta)}{\partial\theta\partial\theta^T} = \frac{\partial}{\partial\theta}\left[\frac{\partial L(\theta)}{\partial\theta}\right]^T = \sum_{i=1}^n -\nabla p_i(\theta)x_i^T \\
&= \sum_{i=1}^n -p_i(1-p_i)x_i x_i^T \triangleq \sum_{i=1}^n a_i x_i x_i^T
\end{aligned}$$

where $a_i = -p_i(1-p_i) \le 0, \forall i$ since $0 \le p_i \le 1$. And for any vector v,

$$v^T \nabla^2 L(\theta) v = v^T (\sum_{i=1}^{n} a_i x_i x_i^T) v = \sum_{i=1}^{n} a_i v^T x_i x_i^T v = \sum_{i=1}^{n} a_i (x_i^T v)^2 \le 0$$

Hence, $H(\theta) \preccurlyeq 0$, by the definition of concave function, $L(\theta)$ is concave. □

2. (10 pts) **Convergence of gradient descent:** In class, we used strong convexity to show convergence of GD. In this homework we will revisit this for Lipschitz functions. To be concrete, suppose the function f is convex and differentiable and its gradient is Lipschitz condition with constant $L > 0$, i.e. we have

$$\|\nabla f(x) - \nabla f(y)\| \le L\|x-y\|_2, \qquad \text{For any } x, y$$

In this problem we run GD for k iterations with a fixed step size $t < 1/L$.

(a) (1 pt) First show that for any y,

$$f(y) \le f(x) + \nabla f(x)^T (y-x) + \frac{L}{2}\|y-x\|^2$$

Proof. Let $h(t) = f((1-t)x + ty) = f(t(y-x)+x)$, $f'(t) = \nabla f((y-x)t + x)^T (y-x)$. Then, by fundamental theorem of calculus,

$$\begin{aligned}
f(y) - f(x) = h(1) - h(0) = \int_0^1 h'(t)dt = \int_0^1 \nabla f((y-x)t+x)^T (y-x)dt \\
&[\text{since } \int_0^1 \nabla f(x)^T (y-x)dt = t\nabla f(x)^T(y-x)\Big|_0^1 = \nabla f(x)^T (y-x)\ ] \\
&= \nabla f(x)^T (y-x) + \int_0^1 [\nabla f((y-x)t+x) - \nabla f(x)]^T (y-x)dt \\
&[\text{by Cauchy Schwartz inequality and integral interval is positive}] \\
&\le \nabla f(x)^T (y-x) + \int_0^1 \|\nabla f((y-x)t+x) - \nabla f(x)\| \|y-x\| dt \\
&[\|\nabla f((y-x)t+x) - \nabla f(x)\| \le L\|(y-x)t + x - x\| = Lt\|y-x\|] \\
&\le \nabla f(x)^T (y-x) + \int_0^1 Lt\|y-x\|^2 dt \\
&= \nabla f(x)^T (y-x) + \frac{Lt^2}{2}\|y-x\|^2\Big|_0^1 = \nabla f(x)^T (y-x) + \frac{L}{2}\|y-x\|^2
\end{aligned}$$

Hence, we have $f(y) \le f(x) + \nabla f(x)^T (y-x) + \frac{L}{2}\|y-x\|^2$, $\forall x, y$.

□

(b) (3 pts) Let $y' = x - t\nabla f(x)$. Now show:

$$f(y') \le f(x) - t\|\nabla f(x)\|^2/2$$

Proof.

$$
\begin{aligned}
f(y') &= f(x - t\nabla f(x)) \quad [\text{plug in } y'] \\
&\le f(x) + \nabla f(x)^T (x - t\nabla f(x) - x) + \frac{L}{2}\|x - t\nabla f(x) - x\|^2 \quad [\text{from (a)}] \\
&= f(x) + (\frac{tL}{2} - 1)t\|\nabla f(x)\|^2 \\
&\le f(x) - t\|\nabla f(x)\|^2/2 \quad [0 < t < 1/L \text{ and } t\|\nabla f(x)\|^2 \ge 0]
\end{aligned}
$$

□

(c) (3 pts) Now show that $f(y') - f(x^*) \le \frac{1}{2t}(\|x - x^*\|^2 - \|y' - x^*\|^2)$

Proof. From (a), plug in $y'\&x$ and $x\&x^*$, we have

$$
\begin{aligned}
f(y') - f(x) &\le \nabla f(x)^T (y' - x) + \frac{L}{2}\|y' - x\|^2 \\
f(x) - f(x^*) &\le \nabla f(x^*)^T (x - x^*) + \frac{L}{2}\|x - x^*\|^2
\end{aligned}
$$

Since x^* is the optimal solution, $\nabla f(x^*) = 0$. Add up the above two, we have

$$
\begin{aligned}
f(y') - f(x^*) &\le \nabla f(x)^T (y' - x) + \frac{L}{2}(\|y' - x\|^2 + \|x - x^*\|^2) && [\nabla f(x^*) = 0] \\
&= (\frac{L}{2} - \frac{1}{t})\|y' - x\|^2 + \frac{L}{2}\|x - x^*\|^2 && [\nabla f(x) = -\frac{1}{t}(y' - x)] \\
&\le \frac{1}{2t}(\|x - x^*\|^2 - \|y' - x^*\|^2) && [t < \frac{1}{L} \Rightarrow L < \frac{1}{t}]
\end{aligned}
$$

□

(d) (3 pts) Using this, show that

$$
f(x^{(k)}) - f(x^*) \le \frac{\|x^{(0)} - x^*\|^2}{2tk}
$$

Proof. From (b), the update is a descent algorithm since $f(x^{(i+1)}) \le f(x^{(i)}) - t\|\nabla f(x^{(i)})\|^2/2$ $\le f(x^{(i)}), \forall i \ge 0$. Then, $f(x^{(k)}) \le f(x^{(i)}), \forall k \ge i$. Hence,

$$
\begin{aligned}
f(x^{(k)}) - f(x^*) &\le \frac{1}{k}\sum_{i=1}^{k}(f(x^{(i)}) - f(x^*)) && [f(x^{(k)}) \le f(x^{(i)}), \forall k \ge i] \\
&\le \frac{1}{k}\sum_{i=1}^{k}\frac{1}{2t}(\|x^{(i-1)} - x^*\|^2 - \|x^{(i)} - x^*\|^2) && [\text{from (c)}] \\
&= \frac{\|x^{(0)} - x^*\|^2}{2tk} && [\text{from telescoping sum}]
\end{aligned}
$$

□

3. (20 pts) **Programming question** Logistic regression is a simple statistical classification method which models the conditional distribution of the class variable y being equal to class c given an input $x \in \mathbb{R}^p$. We will examine two classification tasks, one classifying newsgroup posts, and the other classifying digits. In these tasks the input x is some description of the sample (e.g. word counts in the news case) and y is the category the sample belongs to (e.g. sports, politics). The Logistic Regression model assumes the class distribution conditioned on x is log-linear. For C classes, the goal is to learn $\beta_1, \dots \beta_{C-1} \in \mathbb{R}^p$. We use the K^{th} class as a pivot.

$$\log \frac{p(Y=1|X=x;\beta_1,\dots,\beta_{C-1})}{p(Y=C|X=x;\beta_1,\dots,\beta_{C-1})} = \beta_1^T x$$

Another way to think about this is to take β_C as all zeros. Thus,

$$P(Y=c|X=x,\beta_1,\dots,\beta_C) = \frac{\exp(\beta_c^T x)}{\sum_{j=1}^C \exp(\beta_j^T x)}. \tag{1}$$

Once the model is learned, one can classify a new point by picking the class that maximizes the posterior probability of belonging to that class (see Eq 1). You can measure convergence by the relative error of the concantenated parameter vector $\beta = [\beta_1^T \dots \beta_{K-1}^T] \in \mathbb{R}^{p(C-1)}$. You should write your loss function as an average, and you can use the regularization parameter to be $1/n$.

(a) Write down the log likelihood of this model for n datapoints.

Solution. The probability for all data points is

$$P(\beta) = \prod_{i=1}^n \mathbb{P}(y_i|\beta,x_i) = \prod_{i=1}^n \prod_{c=1}^C \left(\frac{\exp(\beta_c^T x_i)}{\sum_{j=1}^C \exp(\beta_j^T x_i)} \right)^{\mathbb{1}(y_i=c)}$$

where $\mathbb{1}(y_i=c)=1$, if $y_i=c$; $=0$ otherwise. Then, the log likelihood of the model is

$$\begin{aligned} L(\beta) = \log P(\beta) &= \sum_{i=1}^n \sum_{c=1}^C \mathbb{1}(y_i=c) \log \frac{\exp(\beta_c^T x_i)}{\sum_{j=1}^C \exp(\beta_j^T x_i)} \\ &= \sum_{i=1}^n \left(\sum_{c=1}^C \mathbb{1}(y_i=c)\beta_c^T x_i - \log \sum_{j=1}^C e^{\beta_j^T x_i} \right) \qquad [\sum_{c=1}^C \mathbb{1}(y_i=c)=1, \forall y_i] \end{aligned}$$

■

(b) Is this concave? Why?

Solution. Yes, it is concave. First, we derive the first derivative as follows:

$$\frac{\partial L(\beta)}{\partial \beta_j} = \sum_{i=1}^n (\mathbb{1}(y_i=j) - p_{ij}) x_i$$

where $p_{ij} = \frac{e^{\beta_j^T x_i}}{\sum_{c=1}^C e^{\beta_c^T x_i}}$, is the probability of the label of x_i, i.e. y_i, to be class j.

Then the block matrix H_{jk} of the Hessian matrix H is:

$$\begin{aligned}
H_{jk} = \frac{\partial^2 H}{\partial \beta_k \partial \beta_j^T} &= -\sum_{i=1}^n \frac{\partial p_{ij}}{\partial \beta_k} x_i^T = -\sum_{i=1}^n \frac{\delta_{jk} e^{\beta_j^T x_i} (\sum_{c=1}^C e^{\beta_c^T x_i}) - e^{\beta_j^T x_i} e^{\beta_k^T x_i}}{(\sum_{c=1}^C e^{\beta_c^T x_i})^2} x_i x_i^T \\
&= -\sum_{i=1}^n (\delta_{jk} p_{ij} - p_{ij} p_{ik}) x_i x_i^T = -\sum_{i=1}^n (\delta_{jk} - p_{ik}) p_{ij} x_i x_i^T
\end{aligned}$$

where $\delta_{jk} = 1$ if $j = k$; $= 0$, otherwise. If $j = k$, the diagonal block of H is

$$H_{jj} = \sum_{i=1}^n (p_{ij}^2 - p_{ij}) x_i x_i^T, \quad j = 1, 2, \ldots, C$$

If $j \neq k$, the other parts are

$$H_{jk} = \sum_{i=1}^n p_{ij} p_{ik} x_i x_i^T$$

$$\begin{aligned}
u^T H u &= \sum_k u_k^T H_{kk} u_k + 2 \sum_{j<k} u_j^T H_{jk} u_k \\
&= -\sum_k \sum_{i=1}^n (u_k^T x_i)^2 p_{ik} (1 - p_{ik}) + 2 \sum_{j<k} \sum_i p_{ij} p_{ik} (u_j^T x_i)(u_k^T x_i) \\
&= -\sum_{i=1}^n \sum_k (u_k^T x_i)^2 p_{ik} - \sum_k (p_{ik} u_k^T x_i)^2 + \sum_{i=1}^n \left((\sum_k p_{ik} u_k^T x_i)^2 - \sum_k (p_{ik} u_k^T x_i)^2 \right) \\
&= -\sum_{i=1}^n \sum_k (u_k^T x_i)^2 p_{ik} + \sum_{i=1}^n (\sum_k u_k^T x_i p_{ik})^2 \\
&\leq -\sum_{i=1}^n \sum_k (u_k^T x_i)^2 p_{ik} + \sum_{i=1}^n \sum_k (u_k^T x_i)^2 p_{ik} = 0
\end{aligned}$$

Thus, we see that the Hessian is negative semidefinite. Hence the likelihood is concave. ■

(c) For the two datasets in the provided zip file, implement the following five methods. You will use ℓ_2 regularization.

i. Gradient descent

ii. Newton Raphson

iii. Stochastic gradient descent

Solution. See "sds385_hw1_lr_yx.html" for codes and results. To maximize the log likelihood, the loss function we need to minimize using ℓ_2 regularization with $\mu = \frac{1}{n}$ is

$$\min_\beta L(\beta) = -\frac{1}{n} \sum_{i=1}^n \sum_{c=1}^C \mathbb{1}(y_i = c) \log \left(\frac{\exp(\beta_c^T x_i)}{\sum_{j=1}^C \exp(\beta_j^T x_i)} \right) + \frac{1}{n} \|\beta\|^2$$

■

(d) For each method, plot the loglikelihood as a function of number of iterations.

Solution. In Figure 1, I only plot 100 iterations for demonstration. For Newton-Raphson, since it takes time for it to train and it converges very quick, I just train for 20 iterations. ■

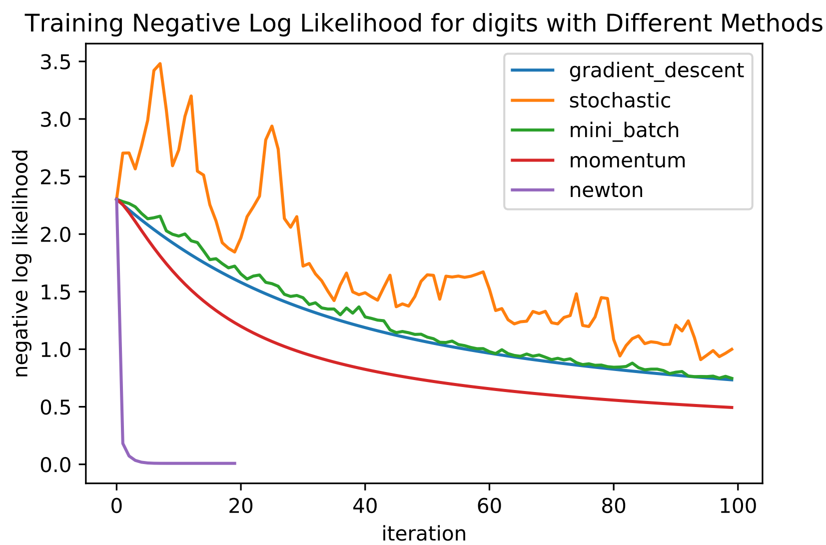


Figure 1: Plot for (d). Parameters: stepsize $\eta = 0.001$; mini batch size = 8; momentum parameter: $\theta = 0.5$; iterations for Newton Raphson: 20, for others: 100.

(e) For gradient descent try different step-sizes and provide a discussion on the effect of stepsize on the convergence.

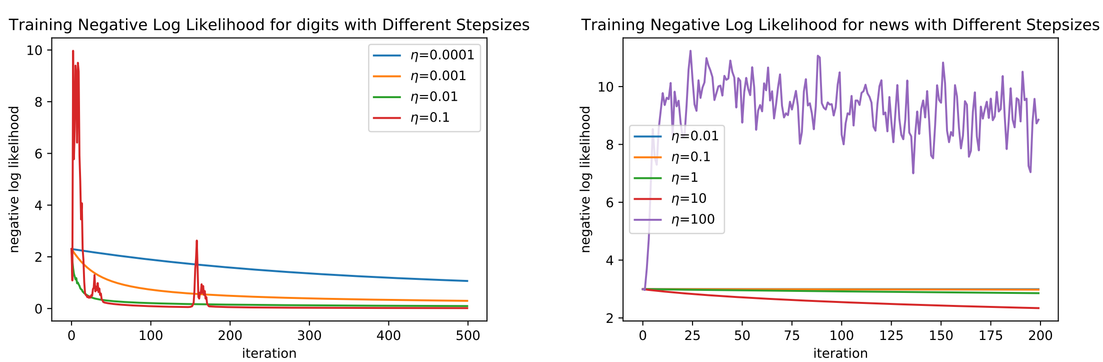
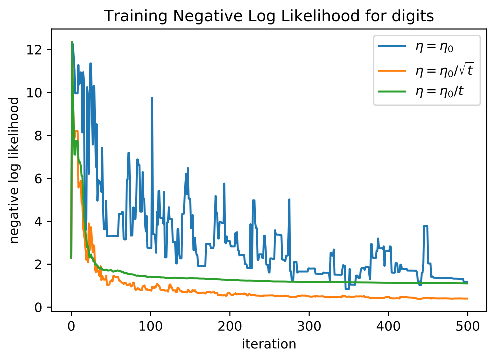


Figure 2: Plot for (e). Left: digits; Right: news.

Solution. As it is shown in Figure 2, for both two datasets, if the stepsizes are too big (digits: $\eta \geq 0.1$; news: $\eta \geq 100$), Gradient Descent do not converge; if the stepsizes are too small (digits: $\eta \leq 0.0001$; news: $\eta \leq 0.01$), it takes more iterations to converge. Hence, a proper stepsize is between the two cases: digits case is $\eta = 0.01$; news case is $\eta = 10$, which makes GD converge fast. ■

(f) For SGD, how are you choosing your step-size? Show a plot with decreasing step-size and for fixed step-size.

Solution. As it is shown in Figure 3 (SGD), the method with constant stepsize fluctuates. With decreasing stepsize, the loss decreases slower but smoother. Both stochastic methods may diverge with constant stepsize but converge with decreasing stepsize. The performance of square root decreasing ($\eta = \frac{\eta_0}{\sqrt{t}}$) is between linear decreasing ($\eta = \frac{\eta_0}{t}$) and constant stepsizes ($\eta = \eta_0$), we can choose $\eta = \frac{\eta_0}{\sqrt{t}}$ to converge both smoothly and quickly to the local or global optimum.



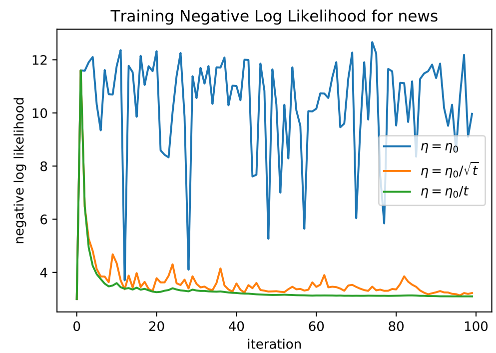


Figure 3: Plot for (g) using SGD. Left: digits with $\eta_0 = 0.01$; Right: news with $\eta_0 = 10$.

■

(g) Finally compute the test set error and compare the methods on both of the datasets.

Solution. I use hyperparameters which work better in above experiments. Result of digits dataset is in Figure 4 and news dataset is in Figure 5. Batch methods have better results but they take more time per iteration while Stochastic methods are quick. Newton method takes less than 10 iterations to converge and has good performance on digits test data. But it takes too long time per iteration when data is large, for example, news data, since I use Kronecker product and for loop for each data. I tried diagonal Newton to approximate NR in the Jupyter Notebook, but the performance is not good as gradient descent. Hence, I only compare four methods for news data. ■

```
p_i("digits", train_d, test_d, C=10, T=int(2e3))
```

```
Loss log of method: gradient_descent
iteration = 0: train loss = 2.3025750930440454, test loss = 2.3025750930440454
iteration = 1999: train loss = 0.04448019328960056, test loss = 0.10240669626438234
Loss log of method: stochastic
iteration = 0: train loss = 2.3025750930440454, test loss = 2.3025750930440454
iteration = 1999: train loss = 0.24383257891919535, test loss = 0.2852865803844827
Loss log of method: mini_batch
iteration = 0: train loss = 2.3025750930440454, test loss = 2.3025750930440454
iteration = 1999: train loss = 0.23102823640648126, test loss = 0.25769453850714236
Loss log of method: momentum
iteration = 0: train loss = 2.3025750930440454, test loss = 2.3025750930440454
iteration = 1999: train loss = 0.02892203250032551, test loss = 0.09927627195583788
Loss log of method: newton
iteration = 0: train loss = 2.3025750930440454, test loss = 2.3025750930440454
iteration = 19: train loss = 0.008528481045289847, test loss = 0.1156053695901
```

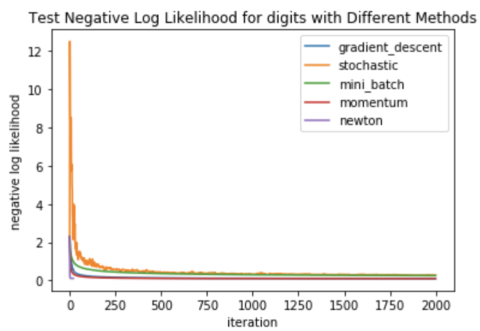


Figure 4: Plot for (i) with dataset "digits". Parameters: stepsize $\eta = 0.01$ (SGD and minibatch SGD are with $\frac{0.01}{\sqrt{t}}$); mini batch size $= 64$; momentum parameter: $\theta = 0.8$; iterations for Newton Raphson: 20, for others: 2000.

```
p_i("news", train_n, test_n, eta=10, C=20, T=int(2e3))
```

```
Loss log of method: gradient_descent
iteration = 0: train loss = 2.9957122737539867, test loss = 2.995712273753989
iteration = 1999: train loss = 1.8722600105140152, test loss = 2.0138910752639476
Loss log of method: stochastic
iteration = 0: train loss = 2.9957122737539867, test loss = 2.995712273753989
iteration = 1999: train loss = 2.662023951794948, test loss = 2.7090895116445575
Loss log of method: mini_batch
iteration = 0: train loss = 2.9957122737539867, test loss = 2.995712273753989
iteration = 1999: train loss = 2.599181290767331, test loss = 2.6419835869118136
Loss log of method: momentum
iteration = 0: train loss = 2.9957122737539867, test loss = 2.995712273753989
iteration = 1999: train loss = 1.8658595713427513, test loss = 2.0090436303356145
```

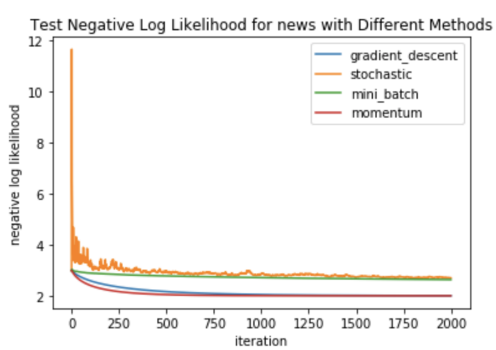


Figure 5: Plot for (i) with dataset "digits". Parameters: stepsize $\eta = 10$ (SGD and minibatch SGD are with $\frac{10}{\sqrt{t}}$); mini batch size $= 64$; momentum parameter: $\theta = 0.8$; iterations: 2000.