Assignment – 3

By: Maor Ashkenazi, Pan Eyal

Question 1

Section A

i. e^{ax} is convex. Proof:

- Using the second derivative definition, we see that $\frac{d^2}{dx^2}e^{ax} = a^2e^{ax} > 0, \forall a, x \in R$.

ii. $-\log(x)$ is convex in the function's domain $x \in (0, \infty)$. Proof:

- We will assume that $\log(x)$ is in base 10 as said in class.
  Using the second derivative definition, we see that:
  $$\frac{d^2}{dx^2} - \log(x) = \frac{1}{x^2 \ln(10)} > 0, \forall x \in (0, \infty)$$

iii. $\log(x)$ is concave in the function's domain $x \in (0, \infty)$. Proof:

- Using the definition for concave functions, a concave function is a negative of a convex function.

iv. $|x|^a$ is convex. Proof:

We'll split the proof into two cases:

- $a = 1$ => In this case we see that for every $\alpha \in [0,1]$ it holds that
  $$f(\alpha x_1 + (1-\alpha)x_2) = |\alpha x_1 + (1-\alpha)x_2|$$
  $$\leq |\alpha x_1| + |(1-\alpha)x_2| = \alpha|x_1| + (1-\alpha)|x_2| = \alpha f(x_1) + (1-\alpha)f(x_2)$$
  Which means that for every two points in the function's domain, the function lies beneath the line between the two points.
- $a > 1$=> In this case we can see that the derivative exists.
  It is easy to see that for $x > 0$ the derivative exists and is $\frac{d}{dx}f(x) = ax^{a-1}$. This also hold for $x < 0$ where the derivative is $\frac{d}{dx}f(x) = -a(-x)^{a-1}$.
  For $x = 0$ we can use the derivative definition:
  $$\frac{d}{dx}f(0) = \lim_{h\to 0}\frac{f(h) - f(0)}{h} = \underbrace{\lim_{h\to 0}\frac{|h|^a}{h}}_{(*)}$$

  - For $h \to 0$ where $h > 0$, $(*) = \lim_{h\to 0}\frac{h^a}{h} = \lim_{h\to 0} h^{a-1} \underbrace{=}_{a>1} 0$
  - For $h \to 0$ where $h < 0$, $(*) = \lim_{h\to 0}\frac{(-h)^a}{h} = \lim_{h\to 0} -\frac{(-h)^a}{-h} = \lim_{h\to 0} -(-h)^{a-1} \underbrace{=}_{a>1} 0$

  Finally, we get:
  $$\frac{d}{dx}f = \begin{cases} ax^{a-1}, & x > 0 \\ 0, & x = 0 \\ -a(-x)^{a-1}, & x < 0 \end{cases}$$

Now we can calculate the second derivative:

$$\frac{d^2}{dx^2}f(x) = \begin{cases} a(a-1)x^{a-2}, & x > 0 \\ 0, & x = 0 \\ a(a-1)(-x)^{a-2}, & x < 0 \end{cases} = \begin{cases} a(a-1)|x|^{a-2}, & x \neq 0 \\ 0, & x = 0 \end{cases}$$

We can see that $\frac{d^2}{dx^2}f(x) \geq 0, \forall x \in R$, thus the function is convex.

v. x^3 is not convex and not concave.

This can be seen by looking at the second derivative of the function, $\frac{d^2}{dx^2}f(x) = 6x$.

$$\frac{d^2}{dx^2}f(x) > 0\,, \qquad \forall x > 0$$
$$\frac{d^2}{dx^2}f(x) < 0\,, \qquad \forall x < 0$$

Section B

$f(x)$ is convex if and only if A is PSD. Proof:
We will show that:

$$f(\alpha x_1 + (1-\alpha)x_2) \leq \alpha f(x_1) + (1-\alpha)f(x_2) \Leftrightarrow$$

We will start from LHS:

$$f(\alpha x_1 + (1-\alpha)x_2) =$$
$$= (\alpha x_1 + (1-\alpha)x_2)^T A(\alpha x_1 + (1-\alpha)x_2) + b^T(\alpha x_1 + (1-\alpha)x_2) + c$$
$$= (\alpha x_1 + (1-\alpha)x_2)^T(\alpha Ax_1 + (1-\alpha)Ax_2) + \alpha b^T x_1 + (1-\alpha)b^T x_2 + c$$
$$= (\alpha x_1)^T(\alpha Ax_1 + (1-\alpha)Ax_2) + \left((1-\alpha)x_2\right)^T(\alpha Ax_1 + (1-\alpha)Ax_2) + \alpha b^T x_1 + (1-\alpha)b^T x_2 + c$$
$$= (\alpha x_1)^T Ax_1\alpha + (\alpha x_1)^T(1-\alpha)Ax_2 + \left((1-\alpha)x_2\right)^T\alpha Ax_1 + \left((1-\alpha)x_2\right)^T(1-\alpha)Ax_2 + \alpha b^T x_1 + (1-\alpha)b^T x_2 + c$$
$$= \alpha^2(x_1)^T Ax_1 + (1-\alpha)\alpha(x_1)^T Ax_2 + (1-\alpha)\alpha(x_2)^T Ax_1 + (1-\alpha)^2(x_2)^T Ax_2 + \alpha b^T x_1 + (1-\alpha)b^T x_2 + c$$

For RHS:

$$\alpha f(x_1) + (1-\alpha)f(x_2) =$$
$$= \alpha(x_1^T Ax_1 + b^T x_1 + c) + (1-\alpha)(x_2^T Ax_2 + b^T x_2 + c)$$
$$= \alpha x_1^T Ax_1 + \alpha b^T x_1 + \alpha c + (1-\alpha)(x_2^T Ax_2 + b^T x_2) + (1-\alpha)c$$
$$= \alpha x_1^T Ax_1 + \alpha b^T x_1 + \alpha c + (1-\alpha)(x_2^T Ax_2 + b^T x_2) + c - \alpha c$$
$$= \alpha x_1^T Ax_1 + \alpha b^T x_1 + (1-\alpha)(x_2^T Ax_2 + b^T x_2) + c$$

Now we will return to the inequality:

$$\alpha^2(x_1)^T Ax_1 + (1-\alpha)\alpha(x_1)^T Ax_2 + (1-\alpha)\alpha(x_2)^T Ax_1 + (1-\alpha)^2(x_2)^T Ax_2 + \alpha b^T x_1 + (1-\alpha)b^T x_2 + c \le \alpha x_1^T Ax_1 + \alpha b^T x_1 + (1-\alpha)(x_2^T Ax_2 + b^T x_2) + c$$

$$\Leftrightarrow$$

$$\alpha^2(x_1)^T Ax_1 + (1-\alpha)\alpha(x_1)^T Ax_2 + (1-\alpha)\alpha(x_2)^T Ax_1 + (1-\alpha)^2(x_2)^T Ax_2 + (1-\alpha)b^T x_2 \le \alpha x_1^T Ax_1 + (1-\alpha)(x_2^T Ax_2 + b^T x_2)$$

$$\Leftrightarrow$$

$$\alpha^2 {x_1}^T Ax_1 + (1-\alpha)\alpha {x_1}^T Ax_2 + (1-\alpha)\alpha {x_2}^T Ax_1 + (1-\alpha)^2 {x_2}^T Ax_2 + (1-\alpha)b^T x_2 \le \alpha x_1^T Ax_1 + (1-\alpha)x_2^T Ax_2 + (1-\alpha)b^T x_2$$

$$\Leftrightarrow$$

$$\alpha^2 {x_1}^T Ax_1 + (1-\alpha)\alpha {x_1}^T Ax_2 + (1-\alpha)\alpha {x_2}^T Ax_1 + (1-\alpha)^2 {x_2}^T Ax_2 \le \alpha x_1^T Ax_1 + (1-\alpha)x_2^T Ax_2$$

$$\Leftrightarrow$$

$$\alpha^2 {x_1}^T Ax_1 - \alpha x_1^T Ax_1 + (1-\alpha)\alpha {x_1}^T Ax_2 + (1-\alpha)\alpha {x_2}^T Ax_1 + (1-\alpha)^2 {x_2}^T Ax_2 - (1-\alpha)x_2^T Ax_2 \le 0$$

$$\Leftrightarrow (\alpha-1)\alpha {x_1}^T Ax_1 + (1-\alpha)\alpha {x_1}^T Ax_2 + (1-\alpha)\alpha {x_2}^T Ax_1 + (1-\alpha)^2 {x_2}^T Ax_2 - (1-\alpha)x_2^T Ax_2 \le 0$$

$$\Leftrightarrow (\alpha-1)\alpha {x_1}^T Ax_1 + (1-\alpha)\alpha {x_1}^T Ax_2 + (1-\alpha)\alpha {x_2}^T Ax_1 + (\alpha-1)\alpha {x_2}^T Ax_2 \le 0$$

$$\Leftrightarrow -(1-\alpha)\alpha {x_1}^T Ax_1 + (1-\alpha)\alpha {x_1}^T Ax_2 + (1-\alpha)\alpha {x_2}^T Ax_1 - (1-\alpha)\alpha {x_2}^T Ax_2 \le 0$$

$$\Leftrightarrow (1-\alpha)\alpha(-{x_1}^T Ax_1 + {x_1}^T Ax_2 + {x_2}^T Ax_1 - {x_2}^T Ax_2) \le 0$$

$$\Leftrightarrow (-{x_1}^T Ax_1 + {x_1}^T Ax_2 + {x_2}^T Ax_1 - {x_2}^T Ax_2) \le 0$$

$$\Leftrightarrow (-{x_1}^T Ax_1 + {x_2}^T Ax_1 + {x_1}^T Ax_2 - {x_2}^T Ax_2) \le 0$$

$$\Leftrightarrow ((-{x_1}^T + x_2^T)Ax_1 + ({x_1}^T - {x_2}^T)Ax_2) \le 0$$

$$\Leftrightarrow (-({x_1}^T - x_2^T)Ax_1 + ({x_1}^T - {x_2}^T)Ax_2) \le 0$$

$$\Leftrightarrow (x_1 - x_2)^T A(x_2 - x_1) \le 0$$

$$\underbrace{\Leftrightarrow}_{*(-1)} (x_1 - x_2)^T A(x_1 - x_2) \ge 0$$

This inequality holds for any $x_1, x_2 \in \mathbb{R}^n$ when A is PSD.

Section C

$$\underline{ii \to i}$$

Assumption: $f(y) \geq f(x) + \nabla f(x)^T(y-x), \ \ \forall x, y \in \Omega$.

Let there be a point $t = (1-\theta)x + \theta y$ s.t $\theta \in (0,1)$, $t \in \Omega$.

Using the assumption above, we know that:

1. $f(y) \geq f(t) + \nabla f(t)^T(y-t) = f(t) + \nabla f(t)^T\big((1-\theta)y - (1-\theta)x\big) =$
$$f(t) + (1-\theta)\nabla f(t)^T(y-x)$$
2. $f(x) \geq f(t) + \nabla f(t)^T(x-t) = f(t) + \nabla f(t)^T(\theta x - \theta y) = f(t) + \theta\nabla f(t)^T(x-y)$

We'll rewrite these inequalities a bit differently, by moving $f(t)$ to the lefthand side and dividing by the positive scalar:

1. $\frac{f(y)-f(t)}{1-\theta} \geq \nabla f(t)^T(y-x)$
2. $\frac{f(x)-f(t)}{\theta} \geq \nabla f(t)^T(x-y) \underbrace{\rightarrow}_{*(-1)} \frac{f(t)-f(x)}{\theta} \leq \nabla f(t)^T(y-x)$

From these two inequalities we get:

$$\frac{f(y)-f(t)}{1-\theta} \geq \frac{f(t)-f(x)}{\theta}$$

After rearranging everything, we get:

$$(1-\theta)f(t) + \theta f(t) = f(t) \leq \theta f(y) + (1-\theta)f(x)$$

$$\rightarrow f\big((1-\theta)x + \theta y\big) \leq \theta f(y) + (1-\theta)f(x)$$

Which is the definition for a convex function.

$$\underline{i \to ii}$$

Assumption: f is convex. This means that

$$f\big((1-\theta)x + \theta y\big) \leq \theta f(y) + (1-\theta)f(x), \ \ \forall x, y \in \Omega \ \forall \theta \in (0,1)$$

We rearrange the expression:

$$f(x - \theta x + \theta y) \leq \theta f(y) + f(x) - \theta f(x) \rightarrow \underbrace{f\big(x + \theta(y-x)\big) \leq f(x) + \theta(f(y) - f(x))}_{(*)}$$

Now let's use the Taylor expansion, around x where $\epsilon = \theta(y-x)$:

$$f\big(x + \theta(y-x)\big) = f(x) + \nabla f(x)^T\big(\theta(y-x)\big) + O(\big||\theta(y-x)|\big|^2)$$

For $\theta \to 0$, as instructed in the assignment, we get:

$$\underbrace{f\big(x+\theta(y-x)\big)=f(x)+\theta\nabla f(x)^T(y-x)}_{(**)}$$

By combining $(*)$, $(**)$, we get the following:

$$f(x)+\theta\nabla f(x)^T(y-x)\le f(x)+\theta(f(y)-f(x))$$

$$\rightarrow \theta\nabla f(x)^T(y-x)\le \theta\big(f(y)-f(x)\big)$$

$$\rightarrow \nabla f(x)^T(y-x)\le f(y)-f(x)$$

$$\rightarrow f(y)\ge f(x)+\nabla f(x)^T(y-x)$$

As instructed.

Combining the two subsections, we see that $i \Leftrightarrow ii$ (the two are equivalent).

Question 2

Section A

To solve a least squares problem with a regularization term:

$$\underset{x}{\operatorname{argmin}}(||Ax-b||_2^2+\lambda||Cx||_2^2)$$

Our closed form solution (using the normal equations) is:

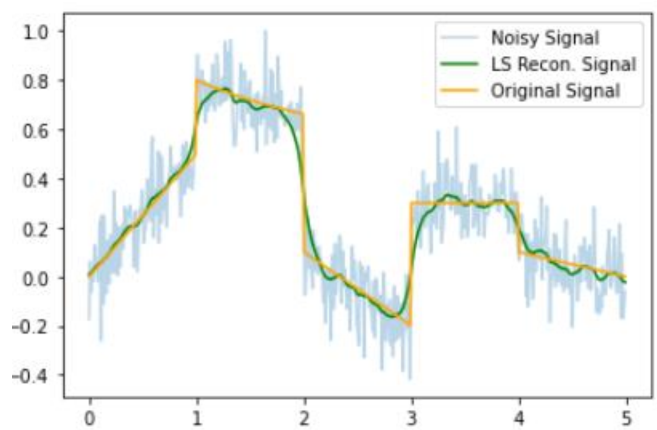
$$(A^TA+\lambda C^TC)x=A^Tb$$

In our case, $A=I$ & $C=G$ & $\frac{\lambda}{2}=\frac{80}{2}$ & $b=y$, thus:

$$(I+40G^TG)x=y \Rightarrow x=(I+40G^TG)^{-1}y$$

We used Python to solve this and got the following solution:

```
reg_factor = 80
res = sparselinalg.inv(scipy.sparse.eye(n)+(reg_factor/2)*G.transpose()@G)@y
```



We can also see that using a larger λ results in a smoother (yet less precise) reconstruction, which makes sense because the regularization term forces the adjacent samples to be similar.

Section B

Our formula for the IRLS iterations:

$$x^{(k+1)}=\underset{x}{\operatorname{argmin}}\left(||x-y||_2^2+\lambda||Gx||^2{}_{W^{(k)}}\right)$$

We'll find a closed form solution for the iteration step using the weighted normal equations:

$$\nabla_{x^{(k+1)}}\left(\left|\left|x^{(k+1)}-y\right|\right|_2^2+\lambda\left|\left|Gx^{(k+1)}\right|\right|^2{}_{W^{(k)}}\right)=$$

$$=\nabla_{x^{(k+1)}}\left(\left(x^{(k+1)}-y\right)^T\left(x^{(k+1)}-y\right)+\lambda\left(Gx^{(k+1)}\right)^TW^{(k)}Gx^{(k+1)}\right)=$$

$$=\nabla_{x^{(k+1)}}\left(\left(x^{(k+1)}-y\right)^T\left(x^{(k+1)}-y\right)+\lambda x^{(k+1)^T}G^TW^{(k)}Gx^{(k+1)}\right)=0$$

$$2\left(x^{(k+1)} - y\right) + \lambda 2G^T W^{(k)} G x^{(k+1)} = 0$$

$$\left(I + \lambda G^T W^{(k)} G\right) x^{(k+1)} = y$$

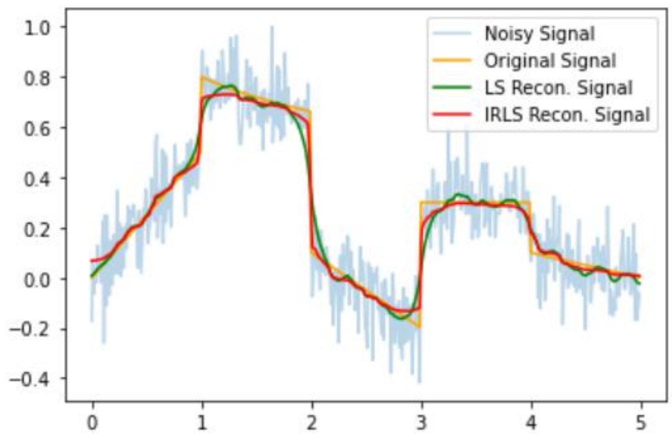
$$x^{(k+1)} = \left(I + \lambda G^T W^{(k)} G\right)^{-1} y$$

Because we are solving for l_1 norm, we use the following weight matrix:

$$W^{(k)} = diag\left(\frac{1}{\left|Gx^{(k)}\right| + \epsilon}\right)$$

We used Python to solve this and got the following solution:

```
for i in range(num_iterations):
    res2 = sparselinalg.inv(scipy.sparse.eye(n) + reg_factor*G.transpose()@W@G)@y
    W = scipy.sparse.diags(1/(abs(G@res2) + epsilon))
```



We can clearly see that the IRLS algorithm obtained a more precise and smoother (in between subsections) signal.

<u>Question 3</u>

Section A

$$\nabla_\theta F(\theta) = \nabla_\theta\left(\frac{1}{2}\left\|f(\theta) - y^{obs}\right\|_2^2\right) = 2 * \frac{1}{2}\nabla_\theta\big(f(\theta)\big)\big(f_\theta - y^{obs}\big) = J(\theta)^T(f(\theta) - y^{obs})$$

Section B

We'll derive the function and find a minimum to obtain d_{opt}

$$\nabla_d\ \left(\frac{1}{2}\left\|f\big(\theta^{(k)}\big) + J\big(\theta^{(k)}\big)d - y^{obs}\right\|_2^2\right) = 0$$

$$\nabla_d\ \left(\frac{1}{2}\left\|r^{(k)} + J\big(\theta^{(k)}\big)d\right\|_2^2\right) = 0$$

$$\nabla_d\ \left(\frac{1}{2}\left({r^{(k)}}^T r^{(k)} + d^T J\big(\theta^{(k)}\big)^T J\big(\theta^{(k)}\big)d + 2d^T J\big(\theta^{(k)}\big)^T r^{(k)}\right)\right) = 0$$

$$J\big(\theta^{(k)}\big)^T J\big(\theta^{(k)}\big)d + J\big(\theta^{(k)}\big)^T r^{(k)} = 0$$

$$J\big(\theta^{(k)}\big)^T J\big(\theta^{(k)}\big)d = -J\big(\theta^{(k)}\big)^T r^{(k)} = -J\big(\theta^{(k)}\big)^T (f(\theta) - y^{obs})$$

From section A, we get:

$$J^T J d = -\nabla F\big(\theta^{(k)}\big)$$

Section C

We'll derive the function and find a minimum to obtain d_{LM}

$$\nabla_d\ \left(\frac{1}{2}\left\|f\big(\theta^{(k)}\big) + J\big(\theta^{(k)}\big)d - y^{obs}\right\|_2^2 + \frac{\mu}{2}\|d\|_2^2\right) =$$

$$\nabla_d\ \left(\frac{1}{2}\left\|f\big(\theta^{(k)}\big) + J\big(\theta^{(k)}\big)d - y^{obs}\right\|_2^2\right) + \nabla_d\ \left(\frac{\mu}{2}\|d\|_2^2\right)$$

From section B, we get the left side gradient. We find a minimum by comparing the gradient to zero:

$$J\big(\theta^{(k)}\big)^T J\big(\theta^{(k)}\big)d_{LM} + J\big(\theta^{(k)}\big)^T r^{(k)} + \mu d_{LM} = 0$$

$$\left(J\big(\theta^{(k)}\big)^T J\big(\theta^{(k)}\big) + \mu I\right) d_{LM} = -J\big(\theta^{(k)}\big)^T r^{(k)}$$

Again, from section A we get:

$$(J^T J + \mu I)d_{LM} = -\nabla F\big(\theta^{(k)}\big)$$

We see that $(J^T J + \mu I)$ is invertible because $J^T J$ is SPSD & $\mu > 0$ thus the whole expression is invertible.

Section D

A descent direction d is one who holds:

$$\langle \mathrm{d}, \nabla F \rangle < 0$$

The descent direction for Levenberg-Marquardt is $d_{LM} = -(J^T J + \mu I)^{-1} \nabla F\big(\theta^{(k)}\big)$.

$$-\langle (J^T J + \mu I)^{-1} \nabla F, \nabla F \rangle = -\nabla F^T ((J^T J + \mu I)^{-1})^T \nabla F = -\nabla F^T (J^T J + \mu I)^{-1} \nabla F$$

We can see that $(J^T J + \mu I)$ is SPD because for every $x \neq 0$:

$$x^T (J^T J + \mu I) x = x^T J^T J x + \mu x^T x > 0$$

Since $J^T J$ is $SPSD$ and $\mu > 0$.

Thus, we know that $\nabla F^T (J^T J + \mu I)^{-1} \nabla F > 0$ and finally $\langle d_{LM}, \nabla F \rangle < 0$.

Section E

To compute the descent direction in each of the algorithms, we need the Jacobian of the model:

$$\mathrm{J} = \nabla f(\theta) = \begin{bmatrix} \mathrm{e}^{-\theta_2 (x-\theta_3)^2} \\ -\theta_1 (x-\theta_3)^2 \mathrm{e}^{-\theta_2 (x-\theta_3)^2} \\ 2\theta_1 \theta_2 (x-\theta_3) \mathrm{e}^{-\theta_2 (x-\theta_3)^2} \end{bmatrix}$$

We ran the algorithms using the following code:

```python
def linesearch(iterations, theta, d, grad, alpha=1, beta=0.5, c=1e-4):
    for j in range(iterations):
        if F(*(theta+alpha*d)) <= F(*theta) + c*alpha*(d@grad):
            break
        else:
            alpha = alpha*beta
    return alpha

def opt(iterations, theta, method="gn"):
    Fs = [F(*theta)]
    residual = model(*theta) - data
    for i in range(iterations):
        J = jacobian(*theta)
        F_grad = J.transpose()@residual
        if method == "gn":
            d = -np.linalg.inv(J.transpose()@J)@F_grad
        elif method == "sd":
            d = -F_grad
        else:
            print("Unknown method")
            return
        alpha = linesearch(100, theta, d, F_grad)
        theta = theta + alpha*d
        residual = model(*theta) - data
        Fs.append(F(*theta))

        if np.linalg.norm(residual) < 1e-3:
            return theta, residual

    return theta, Fs
```

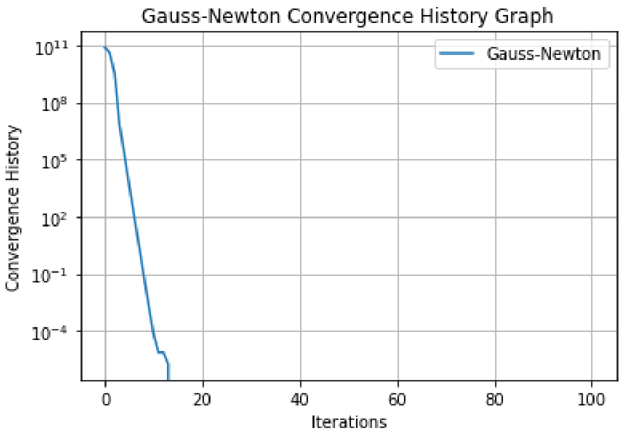
```python
def model(t1, t2, t3):
    return t1*np.exp(-t2*((X-t3)**2))

def F(t1, t2, t3):
    pred = model(t1, t2, t3)
    return 0.5*((pred-data)@(pred-data))
```

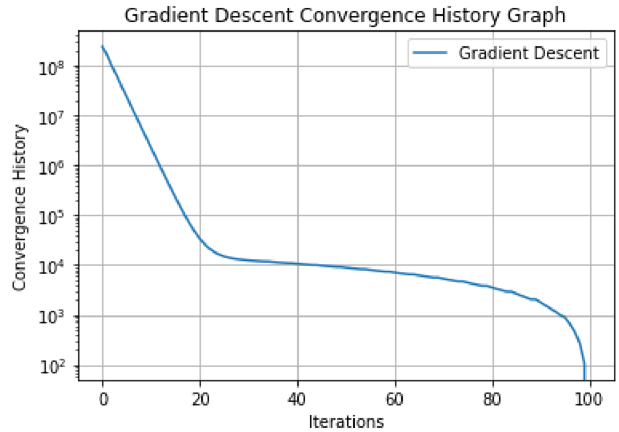
```python
theta_gn, Fs_gn = opt(100, init_theta, method="gn")
theta_sd, Fs_sd = opt(100, init_theta, method="sd")
```

The initial model prediction is:

```
array([     6,      8,     10,     13,     16,     20,     24,     30,
           37,     45,     55,     67,     81,     99,    120,    145,
          175,    210,    253,    303,    363,    433,    516,    613,
          728,    862,   1018,   1201,   1414,   1661,   1947,   2279,
         2661,   3101,   3606,   4186,   4848,   5605,   6467,   7446,
         8557,   9813,  11231,  12829,  14625,  16639,  18892,  21407,
        24209,  27323,  30776,  34596,  38813,  43456,  48557,  54149,
        60265,  66937,  74199,  82084,  90627,  99858, 109810, 120512,
       131993, 144279, 157394, 171358, 186187, 201896, 218493, 235981,
       254361, 273624, 293757, 314742, 336552, 359155, 382510, 406569,
       431279, 456576, 482391, 508647, 535261, 562142, 589193, 616313,
       643392, 670320, 696978, 723250, 749012, 774141, 798516, 822012,
       844508, 865887, 886033])
```

The following plots show the convergence history:





The optimal parameters for SD/GD are the following:

$$\theta = [1000000, 0.001017, 110]$$

The model prediction is:

```
array([     6,      7,      9,     11,     14,     17,     22,     27,
           33,     40,     49,     60,     74,     90,    109,    132,
          159,    192,    231,    278,    333,    398,    476,    567,
          674,    799,    946,   1118,   1318,   1551,   1822,   2135,
         2498,   2915,   3396,   3948,   4581,   5304,   6128,   7067,
         8133,   9341,  10706,  12247,  13981,  15927,  18109,  20548,
        23267,  26294,  29655,  33377,  37491,  42027,  47016,  52491,
        58486,  65034,  72169,  79925,  88336,  97435, 107253, 117823,
       129174, 141331, 154321, 168164, 182879, 198480, 214977, 232376,
       250674, 269868, 289945, 310886, 332667, 355254, 378610, 402686,
       427428, 452774, 478655, 504994, 531705, 558700, 585879, 613140,
       640374, 667466, 694301, 720755, 746707, 772031, 796602, 820295,
       842988, 864559, 884891])
```

The optimal parameters for GN are the following:

$$\theta = [956035, 0.0018835, 101.27]$$

The model prediction is:

```
array([     0,      0,      0,      0,      0,      0,      0,      0,
            0,      0,      0,      0,      0,      0,      0,      1,
            1,      2,      2,      3,      5,      6,      9,     12,
           16,     22,     29,     38,     51,     66,     87,    113,
          147,    189,    244,    312,    399,    507,    643,    811,
         1020,   1278,   1594,   1982,   2455,   3029,   3723,   4559,
         5562,   6760,   8185,   9873,  11864,  14203,  16940,  20128,
        23826,  28097,  33010,  38635,  45050,  52331,  60561,  69822,
        80196,  91766, 104609, 118802, 134413, 151504, 170125, 190318,
       212106, 235500, 260491, 287050, 315129, 344653, 375526, 407626,
       440806, 474895, 509696, 544991, 580539, 616081, 651340, 686028,
       719847, 752493, 783662, 813053, 840375, 865349, 887715, 907236,
       923699, 936925, 946767])
```

We can see that the predictions obtained with GN are far more accurate, and that the predictions obtained with SD/GD are very similar to the initial results (θ did not change by a lot).

In addition, when tracking the optimization process, we saw that the line search algorithm provided **very** small step sizes for SD/GD (close to zero). We deduced that the optimization process in SD/GD is suboptimal because of the different orders of magnitude in the gradient's scale. That is, small updates in the different entries in θ results in large differences in the objective function. Thus, when applying the line search algorithm, we obtained very small step sizes due to the smaller scales in θ, which negatively affects the optimization process for the rest of the values.

On the contrary, in GN the gradient is multiplied by $(J^T J)^{-1}$ which counteracts the effect of the vastly different scales.

Question 4

Section A

The following function ($logistic\_regression\_loss$) returns three inline methods:

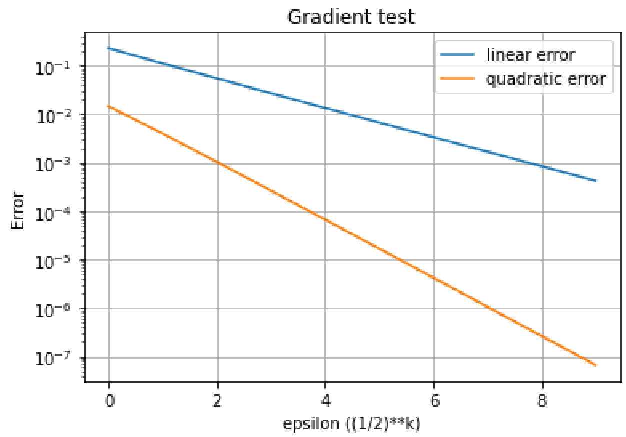
1. Computes the **objective function** for logistic regression
2. Computes the **gradient** for logistic regression
3. Computes the **hessian** for logistic regression

```
def sigmoid(x):
    return 1/(1+np.exp(-x))

def logistic_regression(X, w):
    return sigmoid(X.transpose()@w)

def logistic_regression_loss(X, y):
    c1 = y
    c2 = 1-c1
    m = X.shape[1]

    sig = lambda w: logistic_regression(X, w)
    loss = lambda w: (-1/m)*(c1.transpose()@np.log(sig(w))+c2.transpose()@np.log(1-sig(w)))
    grad = lambda w: (1/m)*X@(sig(w)-c1)
    hess = lambda w: (1/m)*X@np.diag(sig(w)*(1-sig(w)))@X.transpose()

    return loss, grad, hess
```

Section B

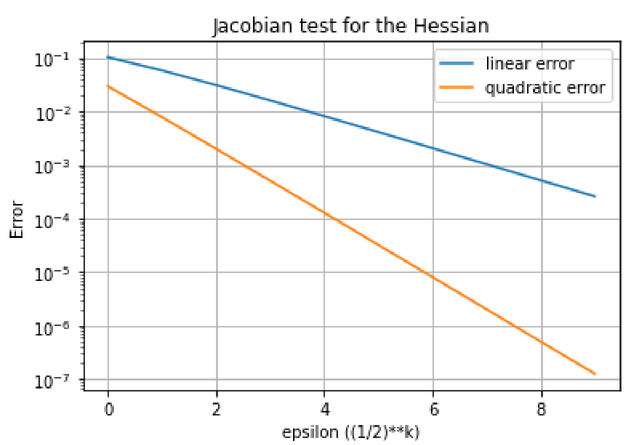
To make sure our gradient works, we used the gradient test, as specified in the lecture notes:

```
X = np.random.rand(10, 10)
y = np.rint(np.random.rand(10))
loss, grad, hess = logistic_regression_loss(X, y)
w = np.random.rand(10)
d = np.random.rand(10)
d = d/np.linalg.norm(d)

O1 = []
O2 = []
eps = 1
for k in range(10):
    eps /= 2
    O1.append(np.abs(loss(w+eps*d)-loss(w)))
    O2.append(np.abs(loss(w+eps*d)-loss(w)-eps*d@grad(w)))
```



To make sure our Hessian works, we used the Jacobian test. The Jacobian of the gradient vector is in fact the hessian. Thus, we used the Jacobian on g, where $g = \nabla f$.

```
X = np.random.rand(10, 10)
y = np.rint(np.random.rand(10))
loss, grad, hess = logistic_regression_loss(X, y)
w = np.random.rand(10)
d = np.random.rand(10)
d = d/np.linalg.norm(d)

O1 = []
O2 = []
eps = 1
for k in range(10):
    eps /= 2
    O1.append(np.linalg.norm(grad(w+eps*d)-grad(w)))
    O2.append(np.linalg.norm(grad(w+eps*d)-grad(w)-eps*hess(w)@d))
```

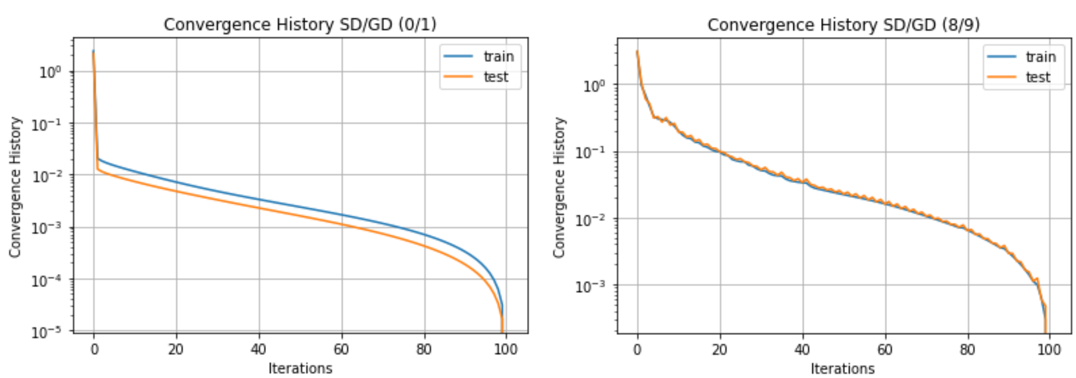


Section C

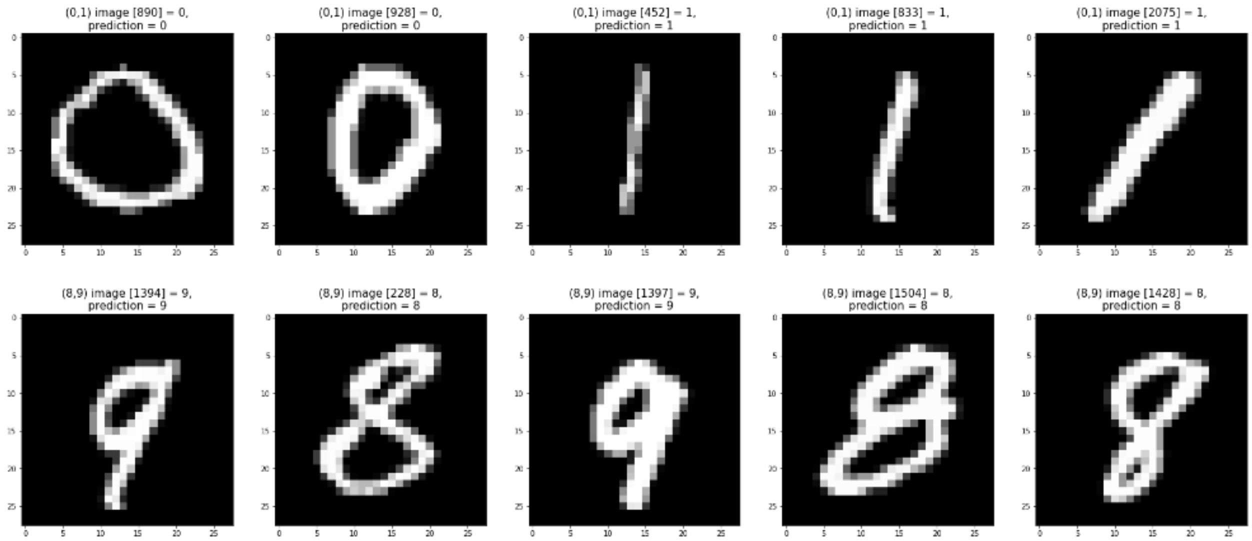
The following is our implementation of SD/GD & Exact-Newton:

```
def linesearch(loss_fn, iterations, w, d, grad, alpha=1, beta=0.5, c=1e-4):
    for j in range(iterations):
        if loss_fn(w+alpha*d) <= loss_fn(w) + c*alpha*(d@grad):
            break
        else:
            alpha = alpha*beta
    return alpha

def opt(loss_fn, grad_fn, hessian_fn, loss_fn_test, iterations=100, method="sd", eps=1e-3):
    train_losses = [loss_fn(w_init)]
    test_losses = [loss_fn_test(w_init)]
    w_opt_test = w_init
    w = w_init

    for i in (prog_bar := tqdm(range(iterations))):
        grad = grad_fn(w)
        hessian = hessian_fn(w)
        if method == "newton":
            d = -(np.linalg.inv(hessian + 0.01*np.eye(hessian.shape[0]))@grad)
        elif method == "sd":
            d = -grad
        else:
            print("Unknown method")
            return
        alpha = linesearch(loss_fn, 10, w, d, grad)
        w = np.clip(w + alpha*d, -1, 1)
        train_losses.append(loss_fn(w))
        test_losses.append(loss_fn_test(w))

        if test_losses[-1] < loss_fn_test(w_opt_test):
            w_opt_test = w
        else:
            print("Overfitting!")

        if np.linalg.norm(grad_fn(w)) / np.linalg.norm(grad) < eps:
            break

    return (train_losses, w), (test_losses, w_opt_test)
```

The following plots show the convergence history of SD/GD (0/1 classification on the left & 8/9 classification on the right):

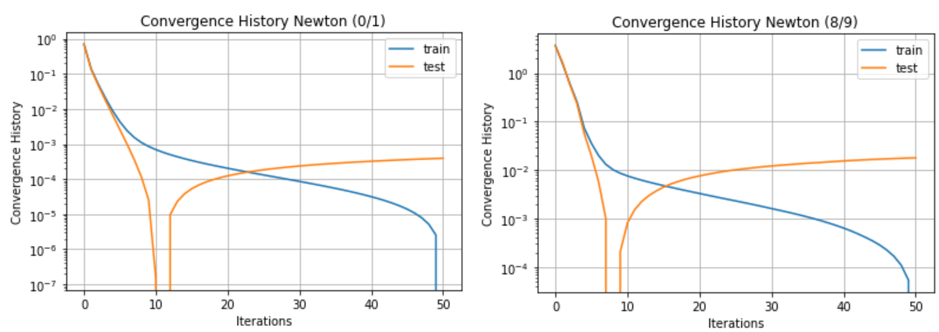


Also, here are some qualitative results of our model:



When applying the Newton method, we encountered a singular instance of the Hessian, so we added a scaled identity to the hessian, to solve the singularity.

The following plots show the convergence history of Newton (0/1 classification on the left & 8/9 classification on the right):



We can clearly see the model overfitted the training data.

We tweaked the hyperparameters (specifically the scale of an identity added to the hessian) and managed to avoid overfitting, although with a smaller convergence factor: