PSET 0

Mark Viti

Due: Feb. 1, 2024

Problem 1

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a be a differentiable convex function. We claim that the gradient ∇f is monotone.

Proof. First, let us recall the definition of monotone as given by Boyd. For a function ψ to be monotone, it must be the case that for all $x, y \in \operatorname{dom}(\psi)$, we have that

$$(\psi(x) - \psi(y))^T (x - y) \geq 0 \tag{1}$$

Now, let us consider the first order condition for convexity. That is, we have that

$$f(y) \geq f(x) + \nabla f(x)^T (y - x) \tag{2}$$

and

$$f(x) \geq f(y) + \nabla f(y)^T (x - y) \tag{3}$$

Thus, all we need to do is combine these two inequalities to get the desired result. To that end, we have that

$$\begin{aligned} f(y) - \left[f(y) + \nabla f(y)^T (x - y)\right] &\geq f(x) + \nabla f(x)^T (y - x) - [f(x)] && (4) \\ 0 &\geq \nabla f(x)^T (y - x) - \nabla f(y)^T (x - y) && (5) \\ 0 &\leq [\nabla f(x) - \nabla f(y)]^T (x - y) && (6) \end{aligned}$$

And so we have that ∇f is monotone. ■

Additional Problem

To show the second part of the problem, we will consider the function $f(x) = \left(x_1, \frac{x_1}{2} + x_2\right)^T$. We will first show that the function is monotone. Note then that we have that for some $x, y \in \mathbb{R}^2$, we have that

$$\begin{aligned} f(x) - f(y) &= \left(x_1 - y_1, \frac{x_1}{2} - \frac{y_1}{2} + x_2 - y_1\right)^T && (7) \\ &= \left(x_1 - y_1, \frac{x_1 - y_1}{2} + x_2 - y_2\right)^T && (8) \end{aligned}$$

Now, we need to see if it is monotone. To that end, we have that

$$\begin{aligned}(f(x)-f(y))^T(x-y) &= \left(x_1-y_1, \frac{x_1-y_1}{2}+x_2-y_2\right)^T(x-y)\\ &= (x_1-y_1)(x_1-y_1)+\frac{x_1-y_1}{2}(x_1-y_1)+(x_2-y_2)(x\\ &= (x_1-y_1)^2+\frac{1}{2}(x_1-y_1)^2+(x_2-y_2)^2\end{aligned}$$

Now, notice that the first term is always non-negative. The second term is also non-negative. The third term is also non-negative. Thus, we have that the function is monotone.

To show that the function is not a gradient, we will show that the partial derivatives are not equal (from Complex Analysis, Multivariate Calculus, etc.)

$$P = x_1 \tag{12}$$

$$Q = \frac{x_1}{2}+x_2 \tag{13}$$

$$\implies \frac{\partial P}{\partial x_2} = 0 \neq \frac{1}{2} = \frac{\partial Q}{\partial x_1} \tag{14}$$

And so we have that the function is not a gradient. Thus, we have a monotone function that is not the gradient of another function. ■

Problem 2

Let $f : \mathbb{R}^n \to \mathbb{R}$ be the log-sum-exp function defined as

$$f(x) = \log\left(\sum_{i=1}^{n} e^{x_i}\right) \tag{15}$$

We will show that f is convex by showing that the Hessian is positive semidefinite.

Proof. First, let us compute the Hessian of f. We will compute the Hessian by first computing the gradient of f. To that end, we have that

$$LSE(x) = \log\left(\sum_{i=1}^{n} e^{x_i}\right) \tag{16}$$

$$\nabla LSE(x) = \frac{1}{\sum_{i=1}^{n} e^{x_i}}\begin{pmatrix} e^{x_1}\\ e^{x_2}\\ \vdots\\ e^{x_n}\end{pmatrix} \qquad \text{By the Chain Rule} \tag{17}$$

Now, to find the Hessian we will need to find the gradient of each element of the vector $\nabla LSE(x)$. Let us denote the ith element of $\nabla LSE(x)$ as $lse_i(x)$. For clarity, this means that

$$lse_i(x) = \frac{e^{x_i}}{\sum_{i=1}^{n} e^{x_i}} \tag{18}$$

Now, we can make a useful simplification by saying that $a_i = e^{x_i}$ and that $A = \sum_{i=1}^n e^{x_i}$. With this notation, we have that

$$lse_i(x) = \frac{a_i}{A} \tag{19}$$

And then we have two cases. First, let us consider the case such that $i = j$. In this case, we have that

$$\frac{\partial lse_i(x)}{\partial x_i} = \frac{A\frac{\partial a_i}{\partial x_i} - a_i\frac{\partial A}{\partial x_i}}{A^2} \tag{20}$$

$$= \frac{a_i}{A} - \frac{a_i^2}{A^2} \tag{21}$$

And then if $i \neq j$, we have that

$$\frac{\partial lse_i(x)}{\partial x_j} = \frac{A\frac{\partial a_i}{\partial x_j} - a_i\frac{\partial A}{\partial x_j}}{A^2} \tag{22}$$

$$= -\frac{a_i a_j}{A^2} \tag{23}$$

Now, we want to write this in a more compact form. For simplicity again, let us call $a = (e^{x_1}, \ldots e^{x_n})$

$$\nabla^2 LSE(x) = \frac{1}{\sum_{i=1}^n e^{x_i}}\left[\left(\sum_{i=1}^n e^{x_i}\right)\mathbf{diag}(a) - aa^T\right] \tag{24}$$

Now, we want to show that this is positive semidefinite. To that end, let us consider the following vector $v \in \mathbb{R}^n$.

$$v^T\nabla^2 LSE(x)v = \frac{1}{\sum_{i=1}^n e^{x_i}}\left[\left(\sum_{i=1}^n e^{x_i}\right)v^T\mathbf{diag}(a)v - v^T aa^T v\right] \tag{25}$$

$$= \frac{1}{\sum_{i=1}^n e^{x_i}}\left[\sum_{i=1}^n e^{x_i}v_i^2 - \sum_{i=1}^n\sum_{j=1}^n e^{x_i}e^{x_j}v_iv_j\right] \tag{26}$$

$$\tag{27}$$

Now, notice that the first term is always non-negative. The second term is also non-negative. Thus, all eigenvalues are non-negative and so we have that the Hessian is positive semidefinite. ■

Problem 3

We are going to show that the information inequality holds for the KL divergence. Due to the hint, we know that the negative log function is convex. Thus, let us recall the definition of the KL divergence. We have that for all $u, v \in \mathbb{R}^n_{++}$:

$$D_{kl}(u,v) = f(u) - f(v) - \nabla f(v)^T(u - v) \tag{28}$$

Such that $f(v) - \sum_{i=1}^n v_i \log v_i$ is the negative log function. Now, since the negative log function is convex, we have that

$$f(u) \geq f(v) + \nabla f(v)^T(u-v) \tag{29}$$
$$f(u) - f(v) \geq \nabla f(v)^T(u-v) \tag{30}$$
$$f(u) - f(v) - \nabla f(v)^T(u-v) \geq 0 \tag{31}$$
$$D_{kl}(u,v) \geq 0 \tag{32}$$

Now, we want to show that the inequality is strict if and only if $u = v$. To that end, let us consider the case where $u = v$. In this case, we have that

$$D_{kl}(u,v) = f(u) - f(v) - \nabla f(v)^T(u-v) \tag{33}$$
$$= f(u) - f(u) - \nabla f(u)^T(u-u) \tag{34}$$
$$= 0 \tag{35}$$

Now, if $D_{kl}(u,v), D_{kl}(v,u) = 0$, then we have that

$$D_{kl}(u,v) = f(u) - f(v) - \nabla f(v)^T(u-v)$$
$$D_{kl}(v,u) = f(v) - f(u) - \nabla f(u)^T(v-u)$$
$$D_{kl}(u,v) + D_{kl}(v,u) = f(u) - f(v) - \nabla f(v)^T(u-v) + f(v) - f(u) - \nabla f(u)^T$$
$$0 = -\nabla f(v)^T(u-v) - \nabla f(u)^T(v-u)$$
$$0 = \nabla f(v)^T(u-v) + \nabla f(u)^T(v-u)$$
$$0 = \nabla f(v)^T(u-v) - \nabla f(u)^T(u-v)$$
$$0 = [\nabla f(v) - \nabla f(u)]^T(u-v) \implies \nabla f(v) = \nabla f(u) \text{ or } u =$$

Now, we know that the gradient of the negative log function is given by

$$\nabla f(v) = \begin{pmatrix} -\frac{1}{v_1} \\ -\frac{1}{v_2} \\ \vdots \\ -\frac{1}{v_n} \end{pmatrix} \tag{43}$$

Thus, we have that $\nabla f(v) = \nabla f(u)$ if and only if $u = v$. Thus, we have that $D_{kl}(u,v) = 0$ if and only if $u = v$. ■

Additional Questions

(a)

Let us define Bregman divergence as follows:

$$D_f(x,y) = f(x) - f(y) - \nabla f(y)^T(x-y) \tag{44}$$

Such that f is the quadratic function $f(x) = ||x||_2^2$. Then, we claim that $D_f(x,y) = ||x-y||_2^2$.

Proof. If $f(x) = ||x||_2^2$, then we have that $\nabla f(x) = 2x$. Thus, we have that

$$\begin{aligned} D_f(x,y) &= f(x) - f(y) - \nabla f(y)^T(x-y) && (45)\\ &= ||x||_2^2 - ||y||_2^2 - 2y^T(x-y) && (46)\\ &= ||x||_2^2 - ||y||_2^2 - 2y^Tx + 2y^Ty && (47)\\ &= xx^T - 2y^Tx + y^Ty && (48)\\ &= (x-y)(x-y)^T && (49)\\ &= ||x-y||_2^2 && (50) \end{aligned}$$

And this completes the proof. ■

(b)

We will now show that for any three points $x, y, z \in \mathbb{R}^n$, we have that

$$D_f(x,y) - \langle \nabla f(x) - \nabla f(y), z - y\rangle = D_f(z,x) - D_f(y,x) \qquad (51)$$

Proof. We will show that $D_f(x,y) - \langle \nabla f(x) - \nabla f(y), z-y\rangle = D_f(z,x) - D_f(z,y)$. To that end, we will simplify both sides to the same expression, proving equivalence. First, we will simplify the left hand side.

$$D_f(x,y) - \langle \nabla f(x) - \nabla f(y), z-y\rangle = f(x) - f(y) - \nabla f(y)^T(x-y) - \langle \nabla f(x)$$

We will now expand the inner product.

$$\begin{aligned} D_f(x,y) - \langle \nabla f(x) - \nabla f(y), z-y\rangle &= f(x) - f(y) - \nabla f(y)^T(x-y) - \Big([\nabla f(x\\ &= f(x) - f(y) - \nabla f(y)^T(z-y) - \nabla f(x)^T\\ &= f(x) - f(y) - \nabla f(x)^T(z-y) \end{aligned}$$

Now for the RHS, we have that

$$\begin{aligned} D_f(z,x) - D_f(z,y) &= f(z) - f(x) - \nabla f(x)^T(z-x) - f(z) + f(y) + \nabla f(x)^T(\\ &= f(y) - f(x) - \nabla f(x)^T(z-x) + \nabla f(x)^T(y-x)\\ &= f(y) - f(x) - \nabla f(x)^Tz + \nabla f(x)^Tx + \nabla f(x)^Ty - \nabla f(x\\ &= f(y) - f(x)\nabla f(x)^T(z-y) \end{aligned}$$

Thus, it is clear that the RHS and the LHS are equivalent. ■

Problem 4

Part (A)

We will show that the the function $f(x) = \sum_{i=1}^{r} |x|_{[i]}$ is convex.

Proof. First, we ought to note that the domain is $\mathbb{R}^n$. which is clearly convex.

Now, we will use the heading of the section to inform our approach. We want to show that this sum is a point-wise maxima of a set of convex functions. We can easily rewrite the function as

$$f(x) = \max |x|_{[1]} + |x|_{[2]} + \cdots + |x|_{[r]} \qquad (60)$$

Now, we need to show that each term of the summation is convex. WLOG, we will show that the absolute value function is convex. To that end, we have that for some $\lambda \in [0,1]$, we have that

$$\begin{aligned} f(\lambda x + (1-\lambda)y) &= |\lambda \cdot x + (1 - \lambda \cdot y)| && (61)\\ &\leq |\lambda \cdot x| + |(1-\lambda)\cdot y| && (62)\\ &= \lambda \cdot f(x) + (1-\lambda)\cdot f(y) && (63)\end{aligned}$$

Thus, each term of the summation is convex. Thus, the function is point-wise maxima of a finite number of convex functions. (To show that it is finite point-wise, we can notice r, n are finite and thus this there are $\binom{n}{r}$ terms in the summation.) Thus, we have that the function is convex. ■

Part (B)

We will show that the function $f(x) = -\log(-\log(\sum_{i=1}^m e^{a_i^T x + b_i}))$ is convex on $\mathbf{dom} f = \left\{x | \sum_{i=1}^m e^{a_i^T x + b_i} < 1\right\}$.

First, we must show that the domain is convex. Let $x, y \in \mathbf{dom} f$. Then, we have that for all $i \in \{1, \ldots, m\}$, we have that

$$\sum_{i=1}^m e^{a_i^T x + b_i} < 1 \text{ and } \sum_{i=1}^m e^{a_i^T y + b_i} < 1 \quad (64)$$

Now, we want to show that for all $\lambda \in [0,1]$, we have that

$$\sum_{i=1}^m e^{a_i^T(\lambda x + (1-\lambda)y) + b_i} < 1 \quad (65)$$

Notice then that since e^x is convex, we have that

$$e^{\lambda z + (1-\lambda w)} \leq \lambda e^z + (1-\lambda)e^w \quad (66)$$

Thus in our case, we have

$$e^{a_i^T(\lambda x + (1-\lambda)y) + b_i} \leq \lambda e^{a_i^T x + b_i} + (1-\lambda)e^{a_i^T y + b_i} \quad (67)$$

This shows domain convexity because if $x, y \in \mathbf{dom} f$, then $e^{a_i^T x + b_i} < 1$ and $e^{a_i^T y + b_i} < 1$. Thus, we have that for each i, $e^{a_i^T(\lambda x + (1-\lambda)y) + b_i} < \lambda e^{a_i^T x + b_i} + (1-\lambda)e^{a_i^T y + b_i}$. Thus it follows for the sums

$$\sum_{i=1}^m e^{a_i^T(\lambda x + (1-\lambda)y) + b_i} < \underbrace{\sum_{i=1}^m \lambda e^{a_i^T x + b_i}}_{<1} + \underbrace{\sum_{i=1}^m (1-\lambda) e^{a_i^T y + b_i}}_{<1} \quad (68)$$

Now, since each component of the total sum is less than 1, their convex combination for $\lambda \in [0,1]$ must also be less than 1. Hence, $\lambda x + (1-\lambda)y \in \mathbf{dom} f$. Thus, the domain is convex.

Having shown the domain is convex, we can now proceed towards using the composition rule to establish that the function itself is convex. As we have already shown, we know that the LSE function is convex. Moreover, we know that we can define $y_i = a_i \cdot x + b_i$. Since this is just a linear transformation, we know that this function is convex. Thus, we can let $g(x) = \log\left(\sum_{i=1}^n e^{a_i^t x + b_i}\right) = g(y) = \log\left(\sum_{i=1}^n e^{y_i}\right)$ and change our expression to

$$f(x) = -\log(-g(y)) \tag{69}$$

And since $g(y)$ is convex, by the composition theorem and Example 3.13, we know that $f(x)$ is convex. ■

Part (C)

We will use te composition rule. Before we proceed, it is helpful to recall Example 3.13. This example lists two key results:

1. If g is convex, then $-\log(-g(x))$ is convex on $\{x | g(x) < 0\}$.
2. If g is convex and non-negative and $p \geq 1$, then g^p is convex.
3. *From the hint* we know that $||x||_p^p / u^{p-1}$ is convex in (x, u) for $u > 0$.
4. f is convex if h is convex and non-increasing, and g is concave for $f(x) = h(g(x))$.

Now we will show that $f(x, t) = -\log(t^p - ||x||_p^p)$ is convex on $\mathbf{dom} f = \{(x, t) t > ||x||_p\}$.

First, using log properties, we can rewrite the function as

$$f(x, t) = -\log\left(t^{p-1}\right) - \log\left(t - \frac{||x||_p^p}{t^{p-1}}\right) \tag{70}$$

$$= -(p-1)\log(t) - \log\left(t - \frac{||x||_p^p}{t^{p-1}}\right) \tag{71}$$

We will now examine each part one at a time. First, we note that $-(p-1)\log(t)$ is convex. This is because the negative log function is convex and $p \geq 1$. Thus, we have a positive scalar times a convex function, which is still convex.

Next, we will examine the second part. We will use the hint to show that the function is convex. To that end, we will let $h(x, t) = \frac{||x||_p^p}{t^{p-1}}$ Then we have that this second part is equal to

$$-\log(t - \frac{||x||_p^p}{t^{p-1}}) = -\log(t - h(x, t)) \tag{72}$$

Now, we want to bne able to apply property 4 and argue that since $-log(x)$ is convex and non-increasing, then the function is convex if $t - h(x, t)$ is concave. Notice that t is linear. Moreover from the hint, we know that $h(x, t)$ is convex. Thus, we have that $t - h(x, t)$ is concave. Therefore, we have that $-\log(t - h(x, t))$ is convex by the composition rule.

So to recap, we have that t $-(p-1)\log(t)$ is convex and that $-\log(t - h(x, t))$ is convex. Thus, we have the sum of two convex functions:

$$f(x,t) = +\underbrace{-(p-1)\log(t)}_{\text{convex}} + \underbrace{-\log\left(t - \frac{||x||_p^p}{t^{p-1}}\right)}_{\text{convex}} \tag{73}$$

And since the sum of two convex functions is convex, we have that $f(x,t)$ is convex.

To show convexity of the domain, we will let $(x,t),(y,s) \in \textbf{dom} f$. Then we have that $t > ||x||_p$ and $s > ||y||_p$. Now, we want to show that for all $\lambda \in [0,1]$, we have that

$$\lambda t + (1-\lambda)s > ||\lambda x + (1-\lambda)y||_p \tag{74}$$

Now, we will use the Minkowski inequality to show that this is true. To that end, we have that

$$||\lambda x + (1-\lambda)y||_p \le ||\lambda x||_p + ||(1-\lambda)y||_p \tag{75}$$
$$= \lambda||x||_p + (1-\lambda)||y||_p \tag{76}$$
$$\le \lambda t + (1-\lambda)s \tag{77}$$

And so we have that the domain is convex. ■

Problem 5

Generate m data points $\{y_i, x_i\}$ satisfying $y_i = x_i^\top \beta$, where $\beta \in \mathbb{R}^n$.

In [ ]:
```python
import numpy as np

seed = 432
np.random.seed(seed)
def generate_data(m, n):
    """
    Generate data for a noiseless linear model y = X beta.

    Parameters:
    - m: Number of data points (rows of X).
    - n: Number of features (columns of X and size of beta).

    Returns:
    - X: Random matrix of size m x n.
    - y: Vector of size m obtained using the linear model.
    - beta: Gaussian random vector of size n.
    """

    # Generate random matrix X of size m x n
    X = np.random.rand(m, n)

    # Generate Gaussian random vector beta of size n
    beta = np.random.randn(n)

    # Compute y using the linear model
    y = X @ beta

    return X, y, beta
```

```
# Example usage:
# m, n = 100, 5
# X, y, beta = generate_data(m, n)
```

Consider the rank deficient case where $m < n$. Show that there are multiple solutions to the linear equation $y = X\beta$.

We generate some data X, y with $m = 50$ and $n = 200$. We can see that $y = Xb$ has at least two solutions.

```
In [ ]: def find_two_solutions_corrected(X, y):
            # Compute the pseudo-inverse of X
            X_pseudo = np.linalg.pinv(X)

            # Compute the minimum norm solution
            beta_min_norm = np.dot(X_pseudo, y)

            # Compute a basis for the null space of X
            _, _, Vt = np.linalg.svd(X)
            null_space_basis = Vt[-(X.shape[1]-np.linalg.matrix_rank(X)):].T

            # Generate a vector in the null space of X
            arbitrary_coefficients = np.random.rand(null_space_basis.shape[1])
            null_space_vector = np.dot(null_space_basis, arbitrary_coefficients)

            # Compute the arbitrary solution
            beta_arbitrary = beta_min_norm + null_space_vector

            return beta_min_norm, beta_arbitrary

        m = 100
        n = 400
        X, y, beta = generate_data(m, n)
        b1, b2 = find_two_solutions_corrected(X, y)

        print(f"b1 and b2 are solutions to $y = X *beta$ because the residue are {np
```

```
b1 and b2 are solutions to $y = X *beta$ because the residue are 3.049105001
863942e-13,2.554003338504377e-13
```

Min-norm solution via cvxpy

The mathematical formulation for finding the minimum norm solution $\mathbf{b}$ subject to the constraint $y = X\mathbf{b}$ is as follows:

$$\begin{aligned} \text{minimize} \quad & ||\mathbf{b}||_2 \\ \text{subject to} \quad & X\mathbf{b} = y \end{aligned}$$

Where:

- $||\mathbf{b}||_2$ is the Euclidean norm of $\mathbf{b}$.
- X is the given matrix.
- y is the given vector.

Once you run this code in your local environment with CVXPY installed, it will return the minimum norm solution $\mathbf{b}$ for the equation $y = X\mathbf{b}$.

In [ ]:
```
import cvxpy as cp

def min_norm_solution_cvxpy(X, y):
    # Define the variable
    b = cp.Variable(X.shape[1])

    # Complete this function
    # Define the objective
    objective = cp.Minimize(cp.norm(b, 2))

    # Define the constraints
    constraints = [X @ b == y]

    # Create the problem and solve it
    problem = cp.Problem(objective, constraints)
    problem.solve()

    return b.value

b = min_norm_solution_cvxpy(X, y)
print(f"b is a solution to $y = X *beta$ because the residue is {np.linalg.r
```

```
b is a solution to $y = X *beta$ because the residue is 1.1427512163646394e-
13
```

```
/Users/markviti/opt/anaconda3/envs/data/lib/python3.10/site-packages/cvxpy/r
eductions/solvers/solving_chain.py:336: FutureWarning:
    Your problem is being solved with the ECOS solver by default. Starting i
n
    CVXPY 1.5.0, Clarabel will be used as the default solver instead. To con
tinue
    using ECOS, specify the ECOS solver explicitly using the ``solver=cp.ECO
S``
    argument to the ``problem.solve`` method.

  warnings.warn(ECOS_DEPRECATION_MSG, FutureWarning)
```

Gradient descent starting from zero.

The gradient descent algorithm updates the parameters iteratively using the gradient of the objective function with respect to the parameters.

For the linear system $y = X\beta$ and the objective function $J(\beta) = \frac{1}{m}\|X\beta - y\|_2^2$, the gradient with respect to β is:

$$\nabla J(\beta) = \frac{2}{m} X^T (X\beta - y)$$

Using the gradient descent update rule, we can iteratively update the solution β as:

$$\beta_{\text{new}} = \beta_{\text{old}} - \alpha \nabla J(\beta)$$

Where α is the learning rate. You are going to implement gradient descent, initialized from a zero vector.

```
In [ ]: def gradient_descent(X, y, learning_rate = 0.001, num_iterations = 10000):
            # Initialize beta as a zero vector
            beta = np.zeros(X.shape[1])

            # Perform gradient descent
            for t in range(num_iterations):
                loss_value, grad_norm = loss_and_gradient(X, y, beta)

                beta = beta - learning_rate * grad_norm

            return beta

        def loss_and_gradient(X, y, beta):
            # compute least-squres loss and the norm of gradient
            grad_norm = 2/X.shape[0] * X.T @ (X @ beta - y)
            loss = np.linalg.norm(X @ beta - y)**2
            return loss, grad_norm
```

Now test thse two functions. We want to show that gradient descent finds the min-norm solution.

```
In [ ]: m = 100
        n = 400
        X, y, beta = generate_data(m, n)

        beta_min_norm = min_norm_solution_cvxpy(X,y)
        beta_gd = gradient_descent(X, y, learning_rate = 0.002,num_iterations= 50000
        error = np.linalg.norm(beta_gd - beta_min_norm)
        if error < 1e-5:
            print(f"the difference between the two solutions are bounded by {error}"
            print("gradient descent with a very small learning rate coincides with t
            print(f"the loss and gradient of loss of GD solution is {loss_and_gradie
        else:
            print(f"the difference between the two solutions are bounded by {error}"
            print("gradient descent does not coincide with min-norm solution")
            print(f"the loss and gradient of loss of GD solution is {loss_and_gradie
       the difference between the two solutions are bounded by 9.40898184878866e-09
       gradient descent with a very small learning rate coincides with the min-norm
       solution
       the loss and gradient of loss of GD solution is (3.7326855852289988e-16, arr
       ay([-3.45787555e-11,  1.87498045e-12,  2.19052477e-11,  5.41291469e-11,
               8.10265126e-11, -6.71255989e-11,  1.54865667e-11, -4.60151769e-11,
               1.06811038e-10,  2.01946858e-12, -7.75006042e-11, -2.06031065e-11,
               6.11811574e-12, -7.07058071e-11, -4.89753733e-11,  3.83058510e-11,
              -4.01253420e-12, -4.12629302e-11, -2.54371349e-11, -3.50446351e-11,
               8.05248713e-11,  3.16513814e-11, -2.01335399e-11, -3.30012027e-11,
              -4.90791139e-11, -1.55308653e-10, -2.87315929e-11, -1.85749911e-10,
               1.31246944e-10, -1.05758916e-10, -4.23603822e-11,  6.71450395e-11,
              -6.56058062e-11, -7.16361190e-11, -5.82282568e-11, -1.34370069e-11,
               1.46278732e-11,  9.73296576e-11, -1.93937200e-11, -1.55175058e-11,
              -1.34346562e-10,  7.57277953e-11, -3.41892937e-11, -6.63680980e-11,
              -5.56919782e-11, -7.33934154e-11,  8.23265813e-12,  1.31337422e-10,
              -1.58013968e-11,  6.64332234e-11,  3.41479626e-11, -2.08842063e-11,
              -2.21918275e-11,  2.23322076e-11,  4.54600740e-11, -3.63135408e-11,
              -2.53453541e-11,  4.26219983e-12,  3.93371202e-11,  1.52341397e-11,
               7.04684516e-11,  3.36496411e-11, -1.05560879e-11,  4.17901614e-11,
              -5.16249601e-11,  1.25858264e-12, -1.39341168e-11, -4.31292019e-11,
               6.79216801e-11,  3.10635184e-11,  5.15291244e-11,  1.09791973e-10,
               9.21053465e-11, -6.71945260e-11, -6.80035543e-12, -7.55817336e-11,
              -1.11439636e-11, -1.90376755e-11, -2.17102301e-11, -6.40714906e-11,
              -1.31598699e-10,  2.43380326e-11, -1.82698206e-11, -5.10360463e-11,
              -2.71984694e-11,  3.23719149e-11,  2.38419820e-11,  5.58172623e-11,
              -5.85937970e-11,  1.08253278e-11, -9.06843788e-12,  7.29951806e-11,
              -1.87204826e-11, -1.95038011e-11,  1.56515486e-11, -8.84928959e-11,
              -4.72033382e-11,  1.89596204e-11,  7.69140208e-12,  1.02168885e-12,
               6.72662179e-11, -3.91634345e-11,  2.19576430e-11,  5.73051663e-12,
               4.66969703e-11, -1.05789592e-12, -1.12277195e-10, -4.65252084e-11,
              -2.57278474e-11, -4.37363208e-11, -4.00273743e-11, -5.91093947e-11,
               6.92239242e-11,  2.02218052e-11,  3.78616046e-11,  1.93346806e-11,
               6.37626294e-11,  1.30826442e-11,  3.90069369e-11, -4.08627954e-11,
              -2.52754580e-12, -1.25622227e-10,  4.39045521e-11,  1.75373363e-11,
               9.56982368e-11,  1.51705826e-11, -8.82916569e-12, -1.14754573e-12,
              -9.96652087e-11,  8.97665324e-11, -7.92993893e-11,  7.19717898e-11,
              -6.81417614e-11,  5.06560600e-11,  2.18881205e-11, -2.26136259e-11,
              -1.36883985e-11, -1.00113332e-11, -6.36281835e-11, -9.10034238e-12,
              -1.22658351e-11, -1.09635168e-10,  4.13647774e-12,  1.59944568e-10,
              -2.03018686e-11, -6.62491627e-11,  3.67216768e-11,  4.75362596e-11,
               6.54089822e-11, -3.38686882e-12, -3.31378119e-11, -6.58433793e-11,
               1.60828507e-10,  5.07298647e-11,  1.84529382e-12,  3.15602430e-11,
              -4.96121698e-11,  3.06417447e-11,  5.43608204e-11,  1.65989623e-11,
               2.76414073e-12, -7.45176043e-11,  1.10039863e-10, -6.05361162e-11,
              -1.38504955e-11, -1.54797376e-11,  3.09903552e-11, -8.24652512e-11,
               1.65971504e-12, -3.10697646e-12, -2.25663894e-11,  8.32256859e-12,
               5.12627984e-11,  6.16782320e-11, -6.72562628e-11, -2.95913551e-11,
              -2.00760282e-11, -3.66267805e-11,  4.96268940e-11,  3.10599948e-11,
               1.64819715e-11,  1.70005528e-11,  9.73139041e-12,  1.43213764e-11,
              -2.88565830e-11, -5.26282558e-11, -8.07155559e-12, -6.94154196e-11,
               1.03964238e-10, -8.57521838e-11,  3.57320949e-13, -3.43585894e-11,
               2.98278435e-11,  3.19451845e-11,  4.84667423e-11,  3.23774109e-11,
              -8.85630515e-11,  4.91721948e-11,  2.73268890e-11,  3.27069569e-11,
              -1.07480118e-10,  5.28811026e-12, -6.41145885e-11,  7.89314153e-12,
               3.54522757e-11, -5.29633737e-11,  1.02597546e-10,  3.38339540e-11,
        3.00364553e-11,  7.17789669e-11,  3.33807115e-11,  1.02199626e-11,
       -7.24422756e-11, -6.23297917e-11,  3.36800329e-12,  5.75673118e-11,
       -1.98174071e-11,  1.07375577e-10,  1.78044313e-11,  3.56005589e-12,
       -5.63922540e-12,  4.23210587e-11, -6.25782930e-11,  6.17299364e-11,
        1.35932210e-10,  3.81154151e-12, -8.52544915e-11, -4.57057374e-12,
       -2.48871486e-11,  6.33833067e-11,  1.40440110e-10, -6.80142332e-11,
        5.27906237e-11, -4.41473991e-12,  4.44316719e-11,  9.85773857e-13,
       -2.49520513e-11, -1.23973427e-10, -6.25760083e-11,  3.88915590e-11,
       -4.64450920e-11, -2.53556453e-11, -1.18231938e-10,  2.62680853e-11,
        4.20478901e-11,  1.70246578e-10, -5.20585975e-11,  3.28612804e-11,
       -3.88046899e-11, -8.25624793e-11, -6.86958536e-11, -2.76000599e-11,
       -2.81620733e-11,  2.26128673e-11, -4.48596058e-11, -1.63159424e-10,
       -4.43907944e-11,  1.25682429e-11,  8.00645910e-11, -8.08838576e-11,
       -8.39339142e-11, -1.42574657e-11, -4.06959704e-11, -1.28873426e-10,
       -3.83885924e-11,  5.00650808e-11,  5.80643158e-11, -2.75403456e-11,
        2.00966031e-11,  5.65169258e-11,  9.03237849e-11,  1.94326606e-11,
        8.69126942e-12,  1.08561818e-10, -6.73756285e-11,  3.10337465e-11,
       -6.31273780e-11,  1.05673910e-10, -2.40428921e-11,  3.97305860e-11,
        4.15189189e-11, -2.26404509e-11,  7.63914907e-11, -2.87857160e-11,
        1.01111717e-11,  1.04272160e-11, -6.42174275e-11, -9.57990195e-11,
        6.41345973e-11,  5.17349186e-11, -1.50303919e-11, -9.92396607e-11,
        6.94146938e-11, -6.00887061e-11,  6.35250312e-12,  5.12837680e-11,
       -4.42014116e-11, -3.54877045e-11,  1.12991602e-10, -3.33358177e-11,
        4.47507267e-11, -5.22939101e-11,  3.77054672e-11, -2.66994007e-11,
        1.98787722e-11,  2.54804157e-11, -7.83671527e-11,  8.56891952e-12,
       -3.12566130e-11,  5.80679428e-11, -5.63557545e-11, -3.07619268e-11,
        1.01614964e-10, -1.03000294e-11,  1.44169883e-10,  9.52874276e-11,
       -1.31683450e-11, -4.11417850e-11,  1.46259687e-10, -1.12441181e-11,
        4.82664038e-11,  5.76291160e-11,  1.26174173e-10,  7.53190371e-11,
        6.26345372e-11,  1.98221081e-12,  1.86471258e-11,  7.28248203e-11,
       -8.09822640e-11,  6.65503305e-11, -2.05361681e-11, -7.33398198e-11,
       -6.14369183e-12,  1.70220918e-11, -4.14388027e-11,  1.22009480e-11,
       -1.29526172e-10,  3.85749399e-11,  1.01945570e-10,  3.31973918e-11,
        2.02799601e-11,  1.95429159e-11, -6.94509275e-11, -3.97046692e-11,
        1.32323216e-10, -4.81582330e-11, -1.56269744e-11, -1.75001310e-11,
        1.50294799e-11, -6.38868658e-11,  7.76874414e-12,  5.91267096e-11,
       -2.62295649e-11, -2.00464982e-11, -1.33544659e-11, -6.97310245e-11,
        1.62307039e-12,  8.40746143e-11,  2.66269198e-11,  5.46156743e-11,
       -1.56018211e-11,  3.08254815e-11,  2.92245678e-11,  1.04895005e-11,
       -5.49009411e-11,  4.69056241e-11, -6.79305907e-11,  8.65864834e-11,
       -6.87177210e-11, -2.80902391e-11, -1.53322393e-11,  3.88085987e-11,
        5.02071168e-11,  5.78713035e-11, -2.87705711e-11, -1.14057916e-10,
        3.65032873e-11, -6.37990465e-11, -4.50518833e-11, -6.40912720e-12,
        4.20289319e-11, -2.44986647e-11,  3.06455080e-11, -4.17232528e-11,
       -3.86569123e-11,  1.22798595e-10, -6.33821665e-11,  1.02595926e-10,
        3.76328473e-11, -1.43000353e-10, -1.59027521e-11, -3.72458875e-12,
        3.14095226e-11,  8.10727771e-11, -7.04669136e-11,  2.10445346e-11,
       -6.05303917e-11,  3.40293723e-11, -9.13139181e-11,  2.36080176e-11]))
```

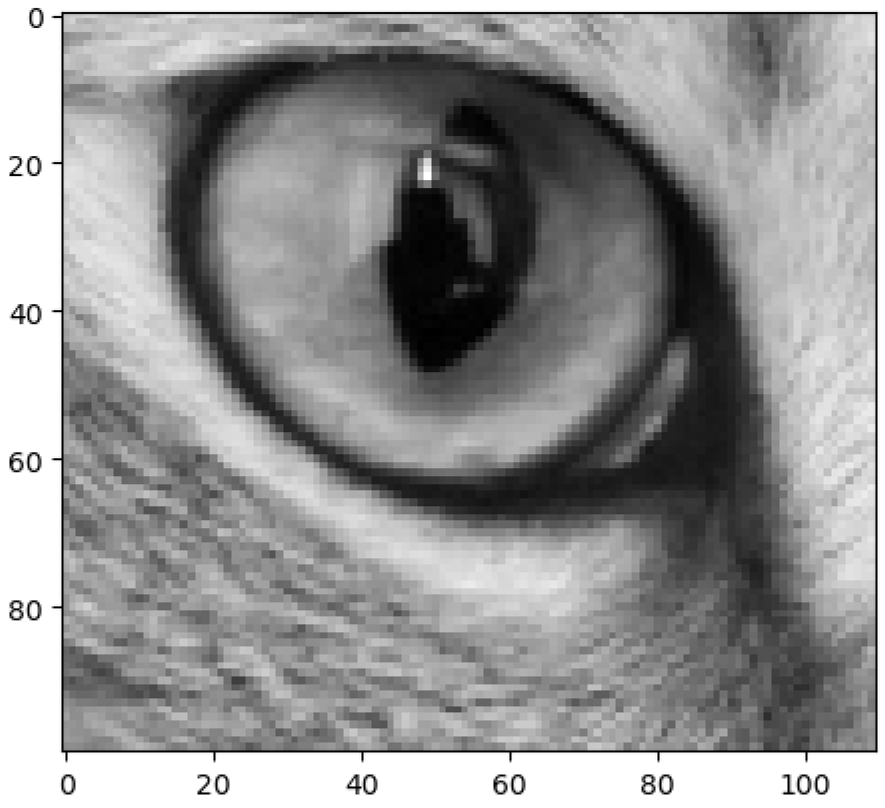
Problem 6

In [ ]:
```
%matplotlib inline
import numpy as np
import scipy
from scipy import ndimage
```

```
import matplotlib.pyplot as plt
from skimage import data, img_as_float, color
from skimage.restoration import denoise_tv_chambolle, denoise_bilateral
seed = 432
np.random.seed(seed)
```

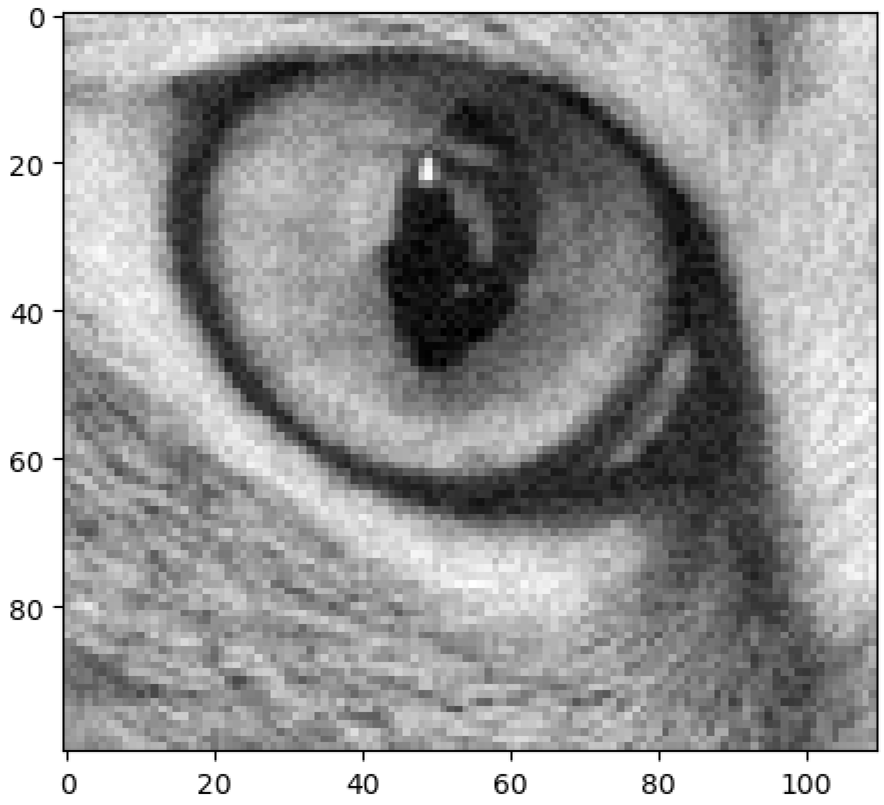
In [ ]:
```
# Import image Cat and convert to grayscale
cat = img_as_float(data.cat())
cat = cat[80:180, 120:230]
cat = color.rgb2gray(cat)

plt.gray()
plt.imshow(cat)
```

Out[ ]:
```
<matplotlib.image.AxesImage at 0x7f79a10cf520>
```



In [ ]:
```
noisy = cat + 0.6 * cat.std() * np.random.random(cat.shape)
noisy = np.clip(noisy, 0, 1)
plt.gray()
plt.imshow(noisy)

X = noisy
print(X.shape)
```

```
(100, 110)
```



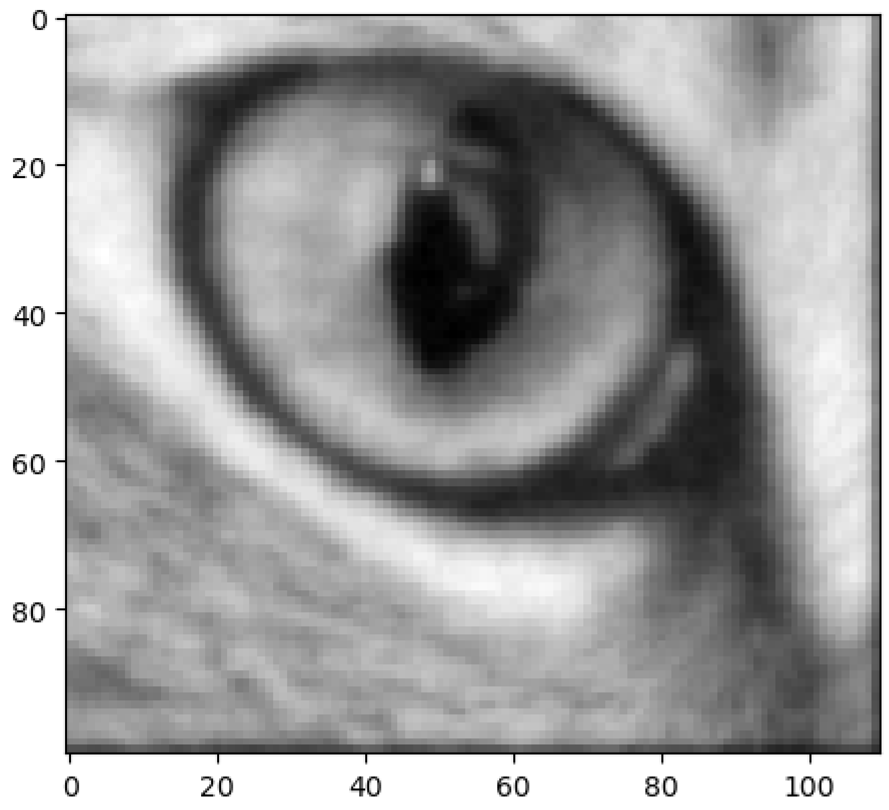
In [ ]:
```
import cvxpy as cp
import numpy as np


# Assuming X is the given noisy image of size m x n
m, n = X.shape
Y = cp.Variable((m, n))

# Pad Y with zeros to handle boundary conditions
Y_padded = cp.hstack([Y, cp.Constant(np.zeros((m, 1)))])
Y_padded = cp.vstack([Y_padded, cp.Constant(np.zeros((1, n+1)))])

# Regularization parameter
reg = 5

###

# Frobenius norm of (Y - X)
fro_norm = cp.norm(Y - X, 'fro')

# Total Variation (TV) in both the x and y directions
tv_x = cp.norm(cp.diff(Y_padded, axis=1), 'fro')
tv_y = cp.norm(cp.diff(Y_padded, axis=0), 'fro')
objective = cp.Minimize(1/2*fro_norm**2 + reg*(tv_x + tv_y))
###
# Define the problem
problem = cp.Problem(objective)

# Set a value for lambda_ and solve the problem
```

```
problem.solve(solver=cp.SCS, verbose = True)

# The optimal denoised image is now in Y.value
denoised_image = Y.value


plt.gray()
plt.imshow(denoised_image)
```

```
=============================================================================
===
                                     CVXPY
                                     v1.4.2
=============================================================================
===
(CVXPY) Jan 31 09:45:34 AM: Your problem has 11000 variables, 0 constraints,
and 0 parameters.
(CVXPY) Jan 31 09:45:34 AM: It is compliant with the following grammars: DC
P, DQCP
(CVXPY) Jan 31 09:45:34 AM: (If you need to solve this problem multiple time
s, but with different data, consider using parameters.)
(CVXPY) Jan 31 09:45:34 AM: CVXPY will first compile your problem; then, it
will invoke a numerical solver to obtain a solution.
(CVXPY) Jan 31 09:45:34 AM: Your problem is compiled with the CPP canonicali
zation backend.
-----------------------------------------------------------------------------
---
                                  Compilation
-----------------------------------------------------------------------------
---
(CVXPY) Jan 31 09:45:34 AM: Compiling problem (target solver=SCS).
(CVXPY) Jan 31 09:45:34 AM: Reduction chain: Dcp2Cone -> CvxAttr2Constr -> C
oneMatrixStuffing -> SCS
(CVXPY) Jan 31 09:45:34 AM: Applying reduction Dcp2Cone
(CVXPY) Jan 31 09:45:34 AM: Applying reduction CvxAttr2Constr
(CVXPY) Jan 31 09:45:34 AM: Applying reduction ConeMatrixStuffing
(CVXPY) Jan 31 09:45:34 AM: Applying reduction SCS
(CVXPY) Jan 31 09:45:34 AM: Finished problem compilation (took 8.455e-02 sec
onds).
-----------------------------------------------------------------------------
---
                                Numerical solver
-----------------------------------------------------------------------------
---
(CVXPY) Jan 31 09:45:34 AM: Invoking solver SCS  to obtain a solution.
------------------------------------------------------------------
               SCS v3.2.4 - Splitting Conic Solver
        (c) Brendan O'Donoghue, Stanford University, 2012
------------------------------------------------------------------
problem:  variables n: 11003, constraints m: 33213
cones:    q: soc vars: 33213, qsize: 3
settings: eps_abs: 1.0e-05, eps_rel: 1.0e-05, eps_infeas: 1.0e-07
          alpha: 1.50, scale: 1.00e-01, adaptive_scale: 1
          max_iters: 100000, normalize: 1, rho_x: 1.00e-06
          acceleration_lookback: 10, acceleration_interval: 10
lin-sys:  sparse-direct-amd-qdldl
          nnz(A): 54793, nnz(P): 1
------------------------------------------------------------------
 iter | pri res | dua res |   gap   |   obj   |  scale  | time (s)
------------------------------------------------------------------
     0| 1.40e+02  5.00e+00  1.40e+03 -6.96e+02  1.00e-01  6.05e-02
   100| 5.17e-05  2.08e-07  8.40e-05  5.82e+01  1.00e-01  2.46e-01
------------------------------------------------------------------
status:  solved
timings: total: 2.46e-01s = setup: 5.60e-02s + solve: 1.90e-01s
```

```
         lin-sys: 1.49e-01s, cones: 1.23e-02s, accel: 3.84e-03s
------------------------------------------------------------------
objective = 58.199153
------------------------------------------------------------------
---------------------------------------------------------------------------
---
                                    Summary
---------------------------------------------------------------------------
---
(CVXPY) Jan 31 09:45:34 AM: Problem status: optimal
(CVXPY) Jan 31 09:45:34 AM: Optimal value: 5.820e+01
(CVXPY) Jan 31 09:45:34 AM: Compilation took 8.455e-02 seconds
(CVXPY) Jan 31 09:45:34 AM: Solver (including time spent in interface) took
2.539e-01 seconds
```

Out[ ]: `<matplotlib.image.AxesImage at 0x7f79e08ff2b0>`



In [ ]:
```python
# Display orginal, noised, and denoised images
plt.figure(figsize=(12, 4))
plt.gray()
plt.subplot(131)
plt.imshow(cat)
plt.axis('off')
plt.title('Original Image')
plt.subplot(132)
plt.imshow(noisy)
plt.axis('off')
plt.title('Noisy Image')
plt.subplot(133)
plt.imshow(denoised_image)
plt.axis('off')
plt.title('Denoised Image')
plt.show()
```