Homework 11 Solutions

1. (10 points) *Convexity (10 points).*

a. For any $0 \leq \theta \leq 1$ and any $x, y \in \mathbb{R}$,

$$\begin{aligned}
f(\theta x + (1-\theta) y) &= \sum_{i=1}^{n} a_i f_i(\theta x + (1-\theta) y) && (1)\\
&\leq \sum_{i=1}^{n} a_i (\theta f_i(x) + (1-\theta) f_i(y)) \quad \text{by convexity of the } f_i && (2)\\
&= \theta \sum_{i=1}^{n} a_i f_i(x) + (1-\theta) \sum_{i=1}^{n} a_i f_i(y) && (3)\\
&= \theta f(x) + (1-\theta) f(y). && (4)
\end{aligned}$$

b. For any $0 \leq \theta \leq 1$ and any $x, y \in \mathbb{R}$,

$$\begin{aligned}
f(\theta x + (1-\theta) y) &= \max_{1 \leq i \leq m} f_i(\theta x + (1-\theta) y) && (5)\\
&\leq \max_{1 \leq i \leq m} \theta f_i(x) + (1-\theta) f_i(y) \quad \text{by convexity of the } f_i && (6)\\
&\leq \theta \max_{1 \leq i \leq m} f_i(x) + (1-\theta) \max_{1 \leq j \leq m} f_j(y) \quad \text{because } \theta, 1-\theta \geq 0 && (7)\\
&= \theta f(x) + (1-\theta) f(y). && (8)
\end{aligned}$$

c. Take $f_1(x) = x^2$ and $f_2(x) = -1$, both functions are convex (their second derivatives are nonnegative), but $h(x) = -x^2$ is strictly concave (its second derivative is -2).

2. (10 points) *1D optimization (10 points).*

a. The code is

```
def derivative_descent(x_ini, der_f, step, eps):
    res = []
    x = x_ini
    der_x = der_f(x)
    res.append(x)
    while np.abs(der_x) > eps:
        x = x - step * der_x
        der_x = der_f(x)
        res.append(x)
    return np.array(res)

def newton_method(x_ini, der_f, der2_f, eps):
    res = []
    x = x_ini
    der_x = der_f(x)
    res.append(x)
    while np.abs(der_x) > eps:
        der2_x = der2_f(x)
        x = x - der_x / der2_x
        der_x = der_f(x)
```

```
        res.append(x)
    return np.array(res)

def quadratic_approx(x, point, f, df, d2f):
    return f(point) + df(point) * (x - point) + d2f(point)* (x-point) ** 2 / 2
```

3. (10 points) *Projected gradient descent (10 points).*

   a.

$$\min_{u \in \mathbb{R}^n_+} ||x-u||_2^2 = \min_{u_1,\dots,u_n \geq 0} \sum_{i=1}^{n} |x_i - u_i|^2 \tag{9}$$

$$= \sum_{i=1}^{n} \min_{u_i \geq 0} |x_i - u_i|^2 . \tag{10}$$

The minimum of $|x_i - u_i|^2$ is achieved by either $u_i = x_i$ if x_i is nonnegative or $u_i = 0$ if x_i is negative. The projection consequently satisfies

$$\mathcal{P}_{\mathbb{R}^n_+}(x)_i = \max\{0, x_i\}, \quad 1 \leq i \leq n. \tag{11}$$

   b. If $x \in \mathcal{B}_{\ell_2}$ then it is obviously equal to the projection, so let us assume that $||x||_2 > 1$.

If we write the vector u in terms of its projection onto $\text{span}(x)$, which we call u_x, and its projection onto the orthogonal complement of $\text{span}(x)$, which we call $u_\perp$ we have

$$||x-u||_2^2 = ||x-u_x||_2^2 + ||u_\perp||_2^2 . \tag{12}$$

No matter what the value of u_x is, this expression is minimized by setting $u_\perp = 0$. We can consequently restrict u to lie on the span of x, i.e. $u = \alpha \frac{x}{||x||_2}$ where $-1 \leq \alpha \leq 1$.

$$\min_{u \in \mathcal{B}_{\ell_2}} ||x-u||_2^2 = \min_{-1 \leq \alpha \leq 1} \left|\left| x - \alpha \frac{x}{||x||} \right|\right|_2^2 \tag{13}$$

$$= \min_{0 \leq \alpha \leq 1} \left(1 - \frac{\alpha}{||x||}\right)^2 ||x||_2^2 \tag{14}$$

$$= \min_{0 \leq \alpha \leq 1} \left(||x||_2 - \alpha\right)^2 . \tag{15}$$

The derivative of $(||x||_2 - \alpha)^2$ with respect to α is $-2(||x||_2 - \alpha)$ which is negative since we are assuming that $||x||_2 > 1 \geq \alpha$, so if α is restricted to $-1 \leq \alpha \leq 1$ then the minimum is achieved at $\alpha = 1$. Thus,

$$\mathcal{P}_{\mathcal{B}_{\ell_2}} = \frac{x}{\max\{||x||_2, 1\}}. \tag{16}$$

   c. The code is

```
def projected_gradient_descent(x_ini, f, grad, step, eps, proj, n_iter):
    res = []
    x = x_ini
    grad_x = grad(x)
    res.append(x)
```

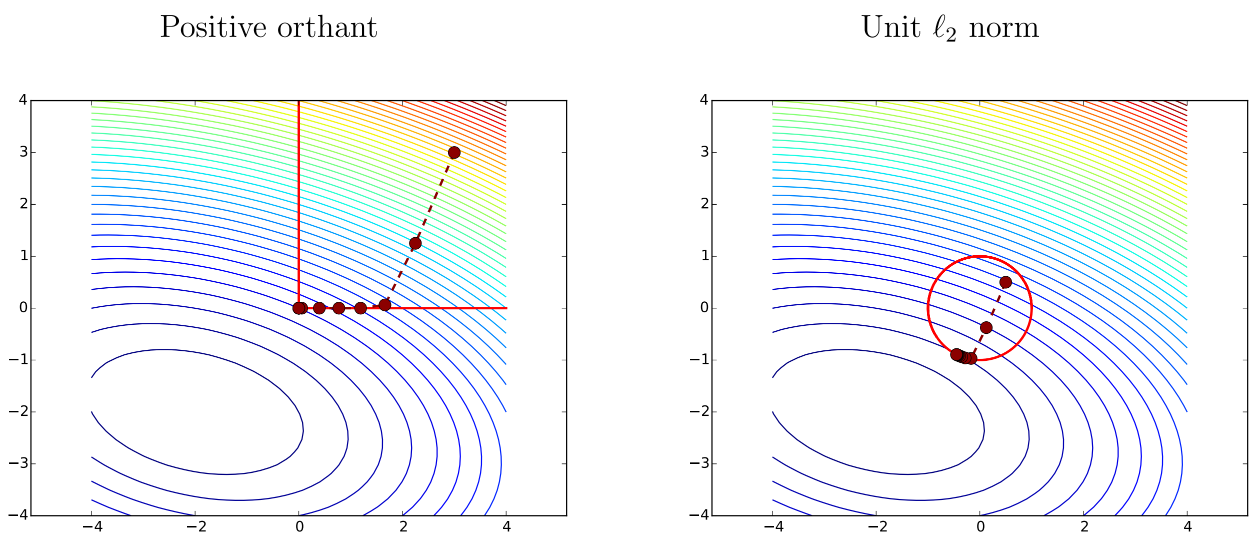

Figure 1: Results for Problem 3.

```
    for i in range(n_iter):
        x = x - step * grad_x
        x = proj(x)
        grad_x = grad(x)
        res.append(x)
    return np.array(res)


def projection_positive(x):
    y = x.clip(min=0)
    return y


def projection_l2(x):
    norm_x = np.linalg.norm(x)
    if norm_x < 1:
        y = x
    else:
        y = x / norm_x
    return y
```

The results are shown in Figure 1.

d. From the images, in both cases the iterations of the algorithm converge to a point on the contour line that is closest to the minimum and touches the set of interest. Any other point in the set of interest is at a contour line that is further out, so the point reached by the method achieves the smallest function value among all points in the set.