Assignment 11

Benjamin Jakubowski

December 3, 2015

1. Convexity

a. Sum of convex functions is convex

$f := \sum_{i=1}^{m} a_i f_i$ is a convex function if the constants $a_1, \dots, a_m$ are nonnegative.

Proof:

Let $x, y \in \mathbb{R}, \theta \in (0,1)$ be given. Then

$$\begin{aligned}
\theta f(x) + (1-\theta) f(y) &= \theta \sum_{i=1}^{m} a_i f_i(x) + (1-\theta) \sum_{i=1}^{m} a_i f_i(y) \\
&= \sum_{i=1}^{m} a_i \left(\theta f_i(x) + (1-\theta) f_i(y)\right) \\
&\geq \sum_{i=1}^{m} a_i f_i \left(\theta x + (1-\theta) y\right) \qquad \text{(since every } f_i \text{ is convex)} \\
&= f\left(\theta x + (1-\theta) y\right)
\end{aligned}$$

So $\theta f(x) + (1-\theta) f(y) \geq f\left(\theta x + (1-\theta) y\right)$ and f is convex. □

b. Maximum of convex functions is convex

Now let

$$g(x) = \max_{1 \leq i \leq m} f_i(x)$$

We show $g(x)$ is itself convex. Again, let $x, y \in \mathbb{R}$ be given. Now let

$$i_x = \arg\max_{1 \leq i \leq m} f_i(x)$$

$$i_y = \arg\max_{1 \leq i \leq m} f_i(y)$$

Then

$$\theta g(x) + (1-\theta) g(y) = \theta f_{i_x}(x) + (1-\theta) f_{i_y}(y)$$

So now, let i be given. Then note

$$\begin{aligned} & f_{i_x}(x) \geq f_i(x) & & f_{i_y}(y) \geq f_i(y) \\ \iff & \theta f_{i_x}(x) \geq \theta f_i(x) & & (1-\theta) f_{i_y}(y) \geq (1-\theta) f_i(y) \\ \implies & \theta f_{i_x}(x) + (1-\theta) f_{i_y}(y) \geq \theta f_i(x) + (1-\theta) f_i(y) \end{aligned}$$

Thus,

$$\begin{aligned} \theta g(x) + (1-\theta) g(y) &= \theta f_{i_x}(x) + (1-\theta) f_{i_y}(y) \\ &\geq \theta f_i(x) + (1-\theta) f_i(y) \qquad \text{for all } i, 1 \leq i \leq m \end{aligned}$$

Next since f_i is convex, for any i

$$\theta f_i(x) + (1-\theta) f_i(y) \geq f_i(\theta x + (1-\theta) y)$$

Hence, for all $i, 1 \leq i \leq m$,

$$\begin{aligned} \theta g(x) + (1-\theta) g(y) &= \theta f_{i_x}(x) + (1-\theta) f_{i_y}(y) \\ &\geq \theta f_i(x) + (1-\theta) f_i(y) \\ &\geq f_i\left(\theta x + (1-\theta) y\right) \end{aligned}$$

But then, in particular,

$$\begin{aligned} \theta g(x) + (1-\theta) g(y) &\geq \max_{1 \leq i \leq m} f_i\left(\theta x + (1-\theta) y\right) \\ &= g\left(\theta x + (1-\theta) y\right) \end{aligned}$$

So g is convex. □

c. Product of convex functions not necessarily convex

Let $h := \prod_{i=1}^{m} f_i$. We aim to show that h is not necessarily convex.

Let $f_1 = x^2 - 5x$. Note f_1 is convex since $f_1''(x) = 2 > 0$ for all x.
Let $f_2 = x^2 + 5x$. Note f_2 is convex since $f_2''(x) = 2 > 0$ for all x.
Now,

$$\begin{aligned} h(x) &= f_1(x) f_2(x) \\ &= (x^2 - 5x)(x^2 + 5x) \\ &= x^4 - 25x^2 \end{aligned}$$

So $h''(x) = 12x^2 - 50 < 0$ if $|x| \leq \sqrt{\frac{50}{12}}$. Hence, h is not convex.

2. Implementing 1D optimization algorithms

Here is the code added to implement derivative descent and Newton's method.

```
def derivative_descent(x_ini, f, der_f, step, eps=0.01):
    x_i = x_ini
    der_f_at_x = der_f(x_i)
    path = [x_i]
    while np.abs(der_f_at_x) > eps:
        x_i = x_i - step*der_f(x_i)
        path.append(x_i)
        der_f_at_x = der_f(x_i)
    return np.array(path)

def newton_method(x_ini, f, der_f, der2_f, eps=0.01):
    x_i = x_ini
    der_f_at_x = der_f(x_i)
    path = [x_i]
    while np.abs(der_f_at_x) > eps:
        x_i = x_i - der_f(x_i)/der2_f(x_i)
        path.append(x_i)
        der_f_at_x = der_f(x_i)
    return np.array(path)

def quadratic_approx(x, point, f, df, d2f):
    approx = f(point) + df(point)*(x-point)\
    + 1.0/2.0 * d2f(point)*(x-point)**2.0
    return approx
```

3. Projected gradient descent

a. Projection onto positive orthant $\mathbb{R}^n_+$

Take $\boldsymbol{x} \in \mathbb{R}^n$. We can represent this vector using the standard basis as

$$\boldsymbol{x} = \sum_{i=1}^{n} \alpha_i \boldsymbol{e}_i$$

Now, if $\alpha_i \geq 0$ for all $i, 1 \leq i \leq n$, then $\boldsymbol{x} \in \mathbb{R}^n_+$, so $\mathcal{P}_{\mathbb{R}^n_+} \boldsymbol{x} = \boldsymbol{x}$.

If not, then let

$$\tilde{\alpha}_i = \begin{cases} 0 & \text{if } \alpha_i \leq 0 \\ \alpha_i & \text{otherwise} \end{cases}$$

Then

$$\mathcal{P}_{\mathbb{R}^n_+} \boldsymbol{x} = \sum_{i=1}^{n} \tilde{\alpha}_i \boldsymbol{e}_i$$

To see this is in fact the projection onto the positive orthant, consider that:

$$\begin{aligned}\mathcal{P}_{\mathbb{R}^n_+}\boldsymbol{x} &= \arg\min_{\boldsymbol{u}\in\mathbb{R}^n_+} ||\boldsymbol{x}-\boldsymbol{u}||_2^2 \\ &= \arg\min_{\boldsymbol{u}\in\mathbb{R}^n_+} \sum_{i=1}^{n}(x_i-u_i)^2\end{aligned}$$

Now take an arbitrary i. There are two cases to consider:

- If $x_i > 0$, $\arg\min_{u_i\in R_+}(x_i - u_i) = x_i$.
- If $x_i \leq 0$, $\arg\min_{u_i\in R_+}(x_i - u_i) = 0$.

But this is exactly how we defined the projection:

$$\mathcal{P}_{\mathbb{R}^n_+}\boldsymbol{x} = \sum_{i=1}^{n}\tilde{\alpha}_i\boldsymbol{e}_i$$

B. Projection onto the unit ℓ_2 ball

Take $\boldsymbol{x}\in\mathbb{R}^n$. First, note that if $||\boldsymbol{x}||_2 \leq 1, \mathcal{P}_{\mathcal{B}_{\ell_2}}\boldsymbol{x} = \boldsymbol{x}$.
'
If not, then

$$\mathcal{P}_{\mathcal{B}_{\ell_2}}\boldsymbol{x} = \frac{\boldsymbol{x}}{||\boldsymbol{x}||_2}$$

We can see this is in fact the projection by noting $\mathcal{P}_{\mathcal{B}_{\ell_2}}\boldsymbol{x}$ is also the projection of $\boldsymbol{x}$ onto the hyperplane tangent to the surface of $\mathcal{B}_{\ell_2}$ at $\frac{\boldsymbol{x}}{||\boldsymbol{x}||_2}$. This immediately implies that the distance between $\boldsymbol{x}$ and any other vector in $\mathcal{B}_{\ell_2}$ must be greater than $\mathcal{P}_{\mathcal{B}_{\ell_2}}\boldsymbol{x}$.

(To see this, note this distance is the norm of a vector, call it $\boldsymbol{z}$. If $\boldsymbol{z}$ passes through the tangent hyperplane at some point other than $\frac{\boldsymbol{x}}{||\boldsymbol{x}||_2}$, then the norm of $\boldsymbol{z}$ is greater than $||\boldsymbol{x}-\mathcal{P}_{\mathcal{B}_{\ell_2}}\boldsymbol{x}||_2$, so our alternate candidate cannot be the projection).

C. Implementing projected gradient descent

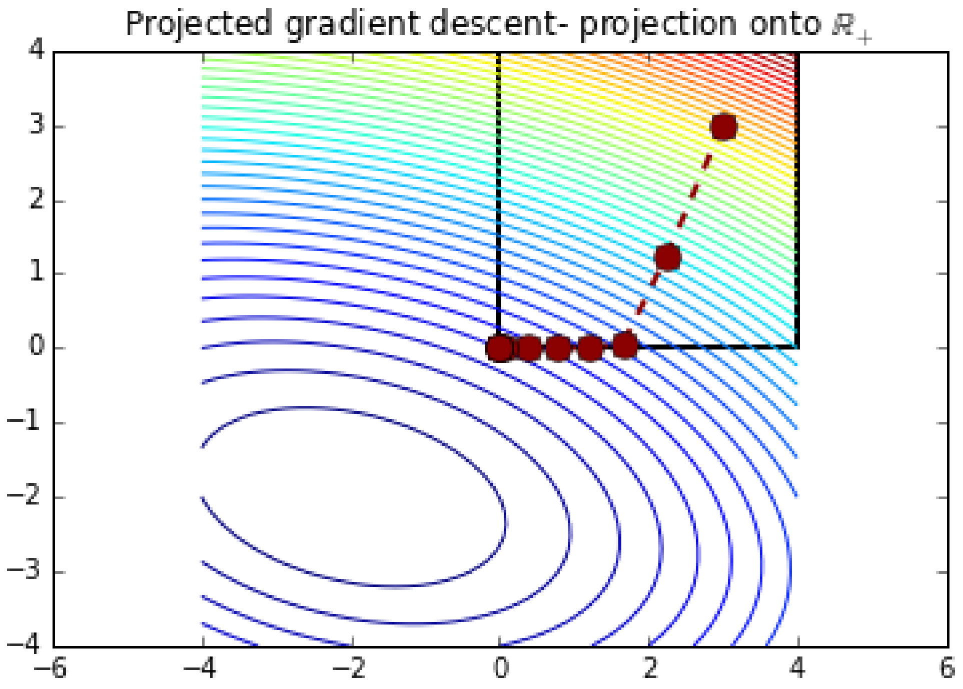
Here is the code added to *hw11_pb3.py* to implement projected gradient descent:

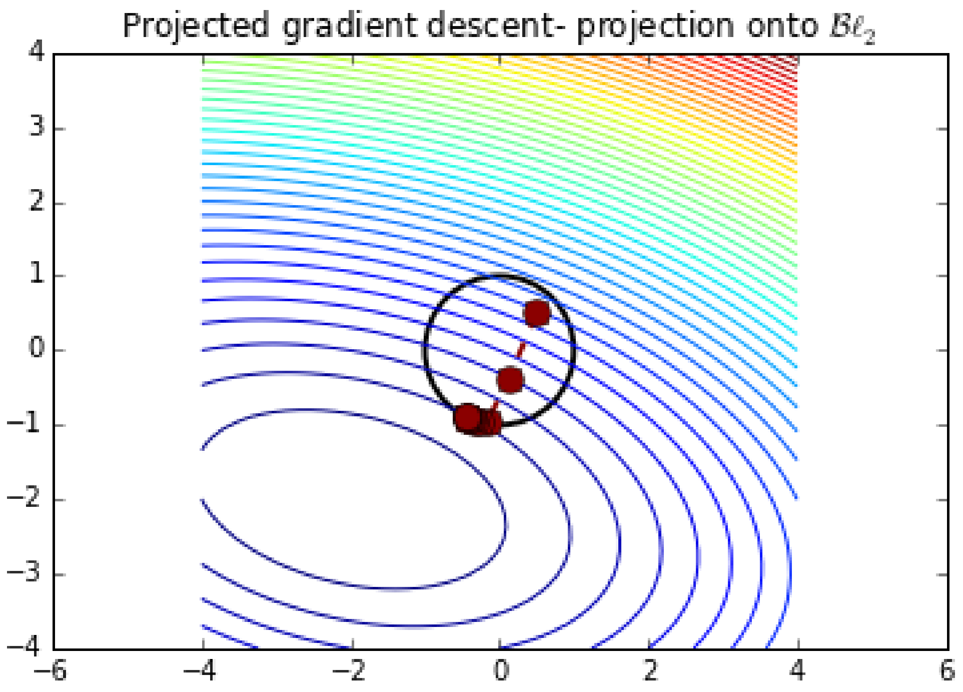
```
def projected_gradient_descent(x_ini, f, grad, step, eps, proj, n_iter):
    x_i = proj(x_ini)
    path = [x_i]
    grad_at_x = grad(x_i)
    iterations = 1
    while np.sqrt(np.dot(grad_at_x, grad_at_x)) > eps\
    and iterations <= n_iter:
        x_i = x_i - step*grad(x_i)
        x_i = proj(x_i)
        path.append(x_i)
        grad_at_x = grad(x_i)
        iterations +=1
    return np.array(path)

def projection_positive(x):
    func = lambda x_i: (0 if x_i<0 else x_i)
    v_func = np.vectorize(func)
    proj = v_func(x)
    return proj

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

The figures are:





D. Intuitive justification for optimal point in constraint set

The intuitive justification that the algorithm reaches the optimal point is based on the observation that the final solution (reached after 10 iterations) appears to be at the lowest point within the set. We can see this in the contour lines/level sets- in both cases (i.e. constrained to $\mathbb{R}_+$ and to $\mathcal{B}_{\ell_2}$) the final solution is at or below the lowest level set depicted in the contour plot.