MACHINE LEARNING 2018

Homework 1's Solutions

November 10, 2018

This homework consists of:

1. Problem 1 (30 pts.)

2. Problem 2 (20 pts.)

3. Problem 3 (50 pts.)

Total: (100 pts.)

Problem 1. *(30 points)*

(a) *(15 points)*
We prove this part by induction on s.
Base cases: For $s = 0$: 1 is clearly an eigenvalue of I_n (where $n \times n$ is also the dimension of A). We also see for $s = 1$, λ is an eigenvalue of A, as given by the problem statement. However, it suffices to list the case $s = 0$ as the base case of the induction.
Inductive step: Suppose the statement holds for $s \geq 0$, that is, λ^s is an eigenvalue of A^s with corresponding eigenvector x. Therefore, $A^s x = \lambda^s x$. Then, $A^{s+1}x = A(A^s x) = A(\lambda^s x) = \lambda^s(Ax) = \lambda^s(\lambda x) = \lambda^{s+1}x$. Therefore, λ^{s+1} is an eigenvalue of A^{s+1} with corresponding eigenvector x. $\blacksquare$

(b) *(15 points)*
Let $L = (A + XBX^T)$ and $R = (A^{-1} - A^{-1}X(B^{-1} + X^T A^{-1} X)^{-1} X^T A^{-1})$. It suffices to show $L * R = I$ and $R * L = I$. We will show $L * R = I$. Proof for the second identity is similar and is, thus, omitted.

$$
\begin{aligned}
&(A + XBX^T)(A^{-1} - A^{-1}X(B^{-1} + X^T A^{-1} X)^{-1} X^T A^{-1})\\
=&(I - X(B^{-1} + X^T A^{-1} X)^{-1} X^T A^{-1}) +\\
&(XBX^T A^{-1} - XBX^T A^{-1} X(B^{-1} + X^T A^{-1} X)^{-1} X^T A^{-1})\\
=&(I + XBX^T A^{-1}) - (X + XBX^T A^{-1} X)(B^{-1} + X^T A^{-1} X)^{-1} X^T A^{-1})\\
=&(I + XBX^T A^{-1}) - XB(B^{-1} + X^T A^{-1} X)(B^{-1} + X^T A^{-1} X)^{-1} X^T A^{-1})\\
=&I + XBX^T A^{-1} - XBX^T A^{-1}\\
=&I \quad \blacksquare
\end{aligned}
$$

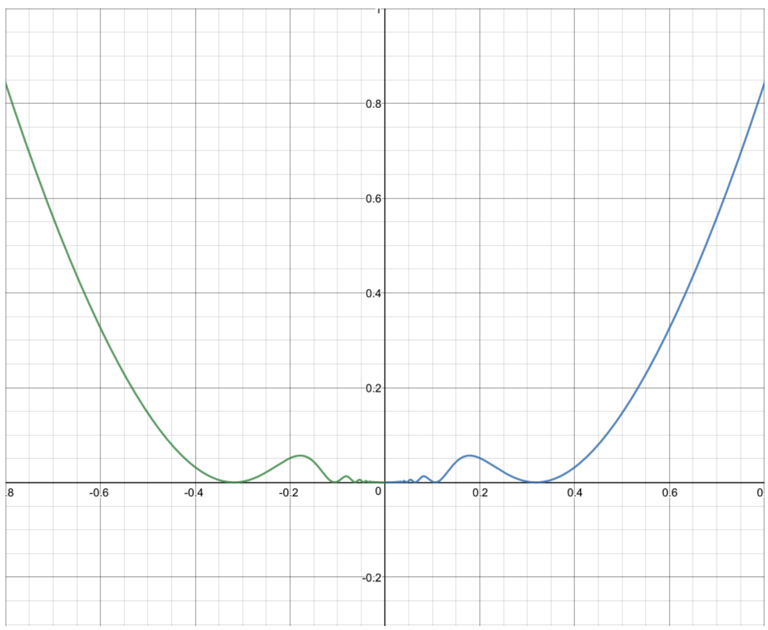
Problem 2 *(20 points)*
Note that a neighborhood N_ϵ of a point x^* is defined as $N_\epsilon(x^*) = \{x : |x - x^*| < \epsilon\}$.

Now, let x^* be an isolated local minimizer of a function f. By definition, there is a neighborhood $N_\epsilon(x^*)$ such that $f(x^*) \leq f(x)$ $\forall x \in N_\epsilon(x^*)$ and that there is no other local minimizer of f in $N_\epsilon(x^*)$. Suppose, for the sake of contradiction, that x^* is **not** a strict local minimizer of f. Then, there is a point $\bar{x} \in N_\epsilon(x^*)$ such that $\bar{x} \neq x^*$ and $f(\bar{x}) = f(x^*)$.

Let $\bar{\epsilon} = \epsilon - |\bar{x} - x^*|$. Note that $\bar{\epsilon} > 0$ because $\bar{x} \in N_\epsilon(x^*)$. $\forall x \in N_{\bar{\epsilon}}(\bar{x})$, $|x - x^*| \leq |x - \bar{x}| + |\bar{x} - x^*| \leq \bar{\epsilon} + |\bar{x} - x^*| = \epsilon$. Therefore, every point in $N_{\bar{\epsilon}}(\bar{x})$ is also in $N_\epsilon(x^*)$, which means $\forall x \in N_{\bar{\epsilon}}(\bar{x})$, $f(x) \leq f(x^*) = f(\bar{x})$. Therefore,

$\bar{x} \in N_\epsilon(x^*)$ is a local minimizer of f, which contradicts the fact that x^* is an isolated local minimizer of f. ■

Note that not all strict local minimizers are isolated. As a counter-example, consider function $f : \mathbb{R} \to \mathbb{R}$ such that $f(x) = x^2(1 + cos(\frac{1}{x}))$ for $x \neq 0$ and $f(0) = 0$. Clearly 0 is a strict local minimizer of f. But 0 is not an isolated local minimizer of f (See below figure for the graph of f around 0).



Problem 3 *(50 points)*
There are multiple ways to do this. Below is a simple snippet that doesn't do anything fancy for initialization and step size tuning. As long as a student demonstrates his/her understanding of the updates, some concept of early stopping, and initialization, it should be fine. Note that $\nabla f(x) = Ax + b$

1 Sample python code

```
import numpy as np
def objective(A,b,x):
    return (np.dot(x.T,np.dot(A,x)))+np.dot(b,x)
```

```python
def gradient_descent(A,b,stepSize=0.01,niters=1000,epsilon=0.01):
    n=A.shape[0]
    x=np.random.rand(n)
    for i in range(niters):
        print("iteration %d"%(i))
        grad=np.dot(A,x)+b
        print("   gradient norm = %.4f"%(np.linalg.norm(grad)))
        print("   function value = %.4f"%(objective(A,b,x)))
        if np.linalg.norm(grad)<epsilon:
            print("***optimal solution found")
            break
        x=x-stepSize*grad
        print('---')
    return x,objective(A,b,x)
#
A=np.array([[1,0,0],[0,1,0],[0,0,1]])
b=np.array([0,0,0])
x,obj=gradient_descent(A,b)
print("\n\n\nbest solution found: {}".format(x))
print("best objective value: {}".format(obj))
```