Changmao Li

Homework 0
CS534 Machine Learning, Spring 2019

This timed homework assignment tests prerequisite knowledge in linear algebra, statistics, and programming. Please observe the time limits in each problem (these limits include time to lookup facts you may have forgotten). Show your work, attach any code in subsequent pages, and include a filled and signed copy of the honor pledge.

Problem 1 (linear algebra - 2 hours)

1.a. Given the pairs of vectors $v_1 = [\frac{\sqrt{3}}{2}, \frac{1}{2}, 0]^T, v_2 = [0, \frac{1}{2}, \frac{\sqrt{3}}{2}]^T$ and $\hat{v}_1 = [\frac{1}{\sqrt{2}}, 0, \frac{-1}{\sqrt{2}}]^T, \hat{v}_2 = [\frac{-1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0]^T$, use software to find the matrix $A \in \mathbb{R}^{3x3}$ such that

$$[\hat{v}_1, \hat{v}_2] = A[v_1, v_2],\ [v_1, v_2] = A^{-1}[\hat{v}_1, \hat{v}_2]. \quad (1)$$

1.b. Show that the inverse of a nonsingular skew-symmetric matrix $A^T = -A$ is also skew-symmetric.

1.c. Show that the eigenvalues of a real symmetric matrix $A = A^T$ are always real (use the conjugate transpose).

1.a on the later page

1.b $(A^{-1})^T = (A^T)^{-1} = (-A)^{-1} = -A^{-1}$

1.c we have $\begin{cases} \bar{A}\bar{V} = \bar{\lambda}\bar{V} \\ AV = \lambda V \\ A = A^T \\ A = \bar{A} \end{cases}$ $\bar{V}^TAV = \bar{V}^T(AV) = \bar{V}^T(\lambda V) = \lambda(\bar{V}\cdot V)$

$\therefore \bar{V}^TAV = \bar{V}^TA^TV = (A\bar{V})^TV = (\bar{A}\bar{V})^TV = (\bar{\lambda}\bar{V})^TV = \bar{\lambda}(\bar{V}\cdot V)$

$\because V \neq 0 \therefore \bar{V}\cdot V \neq 0 \therefore \lambda = \bar{\lambda} \therefore \lambda \in R$

```
%1.a
syms a11 a12 a13 a21 a22 a23 a31 a32 a33
A = [a11 a12 a13; a21 a22 a23; a31 a32 a33];
v1 = [sqrt(3)/2;0.5;0];
v2 = [0;0.5;sqrt(3)/2];
v3 = [1/sqrt(2);0;-1/sqrt(2)];
v4 = [-1/sqrt(2);1/sqrt(2);0];
v6 = [v3, v4];
v7 = [v1, v2];
[s11,s12,s13,s21,s22,s23,s31,s32,s33,params,conditions] = solve([v6==A*v7,v7
== inv(A)*v6],[a11,a12,a13,a21,a22,a23,a31,a32,a33],'ReturnConditions',true);
A = [s11 s12 s13;s21 s22 s23; s31 s32 s33]
params
conditions
```

A =

[z1 + (2*2^(1/2)*3^(1/2))/3, - 3^(1/2)*z1 - 2^(1/2), z1]
[z - (2^(1/2)*3^(1/2))/3, 2^(1/2) - 3^(1/2)*z, z]
[z2 - (2^(1/2)*3^(1/2))/3, -3^(1/2)*z2, z2]

params =

[z, z1, z2]

conditions =

z + z1 + z2 ~= 0

Problem 2 (programming)

Gradient-based optimization is a fundamental technique in machine learning. This technique seeks to minimize a cost f that is a function of some parameters β by iteratively updating the parameters using information from the gradient of the cost.

Given Himmelblau's function

$$f(\beta_1, \beta_2) = (\beta_1^2 + \beta_2^2 - 11)^2 + (\beta_1 + \beta_2^2 - 7)^2 \tag{2}$$

2.a. Identify the *critical points* $\beta^* = [\beta_1^*, \beta_2^*]^T$ of f using a software solver. Classify each critical point as a local minimum, maximum, or saddle point by analyzing the Hessian matrix of second partial derivatives H at β^*

$$H\, f = \left[\begin{array}{cc} \frac{\partial^2 f}{\partial \beta_1^2} & \frac{\partial^2 f}{\partial \beta_1 \partial \beta_2} \\ \frac{\partial^2 f}{\partial \beta_2 \partial \beta_1} & \frac{\partial^2 f}{\partial \beta_2^2} \end{array} \right] \Bigg|_{\beta=\beta^*} \tag{3}$$

Generate a contour plot of the cost f in the range $\hat{\beta}_1, \hat{\beta}_2 \in [-5, 5]$. Plot the critical points and indicate their classifications.

2.b. Implement gradient descent to find solutions minimizing f using the updating function

$$\hat{\beta}^{i+1} = \hat{\beta}^i - \gamma \nabla f(\hat{\beta}^i) \tag{4}$$

where $\hat{\beta}^i = [\hat{\beta}_1^i, \hat{\beta}_2^i]^T$ is the solution at iteration i, and the *learning rate* $\gamma = 1e{-}3$. Plot the trajectory of your solutions $\hat{\beta}^0, \hat{\beta}^1, \ldots \hat{\beta}^N$ where $N = 10^5$ iterations on the plot from **2.a.**. Illustrate trajectories for ten random initializations of $\hat{\beta}^0 \in [-5, 5]$.

2.c. Implement Newton's method to find solutions minimizing f using the updating function

$$\hat{\beta}^{i+1} = \hat{\beta}^i - \gamma [H\, f(\hat{\beta}^i)]^{-1} \nabla f(\hat{\beta}^i), \tag{5}$$

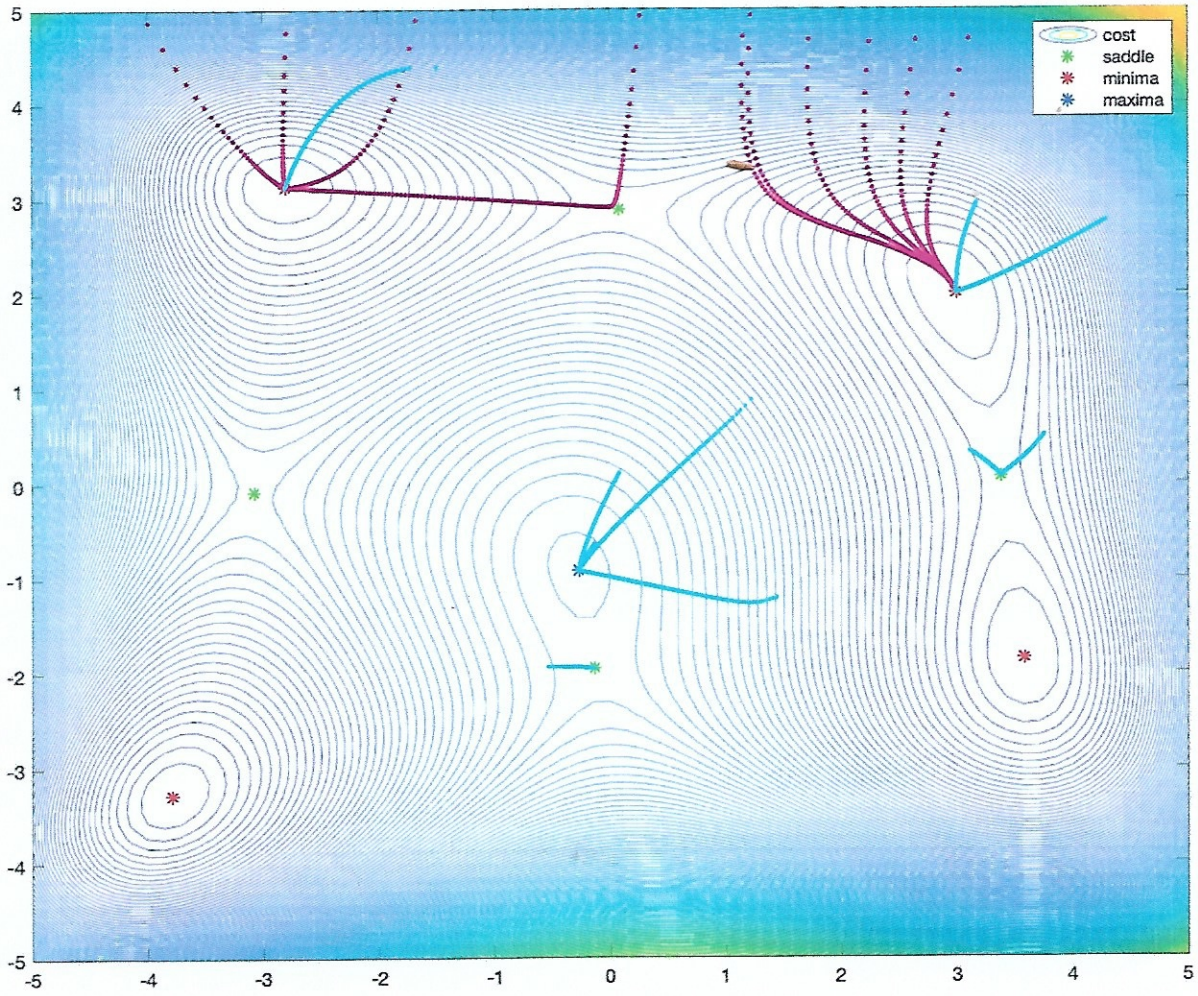
where H is the Hessian. Use *learning rate* $\gamma = 1e-2$. Illustrate trajectories for ten random initializations as in as in **2.b**.

```
% 2.a
syms X Y
f = (X*X+Y-11)^2 + (X+Y*Y-7)^2;
[a,b] = solve([diff(f, X) == 0, diff(f, Y) == 0],[X Y]);
df_dx = diff(f, X);
df_dy = diff(f, Y)
ddf_dxdx = diff(df_dx,X);
ddf_dydy= diff(df_dy,Y);
ddf_dxdy = diff(df_dx,Y);
fc = fcontour(f);
fc.LevelStep = 5;
hold on;
axis([-5,5,-5,5]);
saddle_x = [];
saddle_y = [];
minima_x = [];
minima_y = [];
maxima_x = [];
maxima_y = [];
for i=1:length(a)
      ddf_dxdx_i = subs(ddf_dxdx, [X,Y], [a(i),b(i)]);
      ddf_dydy_i = subs(ddf_dydy, [X,Y], [a(i),b(i)]);
      ddf_dxdy_i = subs(ddf_dxdy, [X,Y], [a(i),b(i)]);
      det_H_f = ddf_dxdx_i * ddf_dydy_i - ddf_dxdy_i^2;
      df_dx_i = subs(df_dx,[X,Y], [a(i),b(i)];
      df_dy_i = subs(df_dy,[X,Y], [a(i),b(i)];
      if det_H_f <= 0
            saddle_x = [saddle_x;a[i]]
            saddle_y = [saddle_y:b[i]]
      else
            if df_dx_i>0&& df_dy_i>0
                  minima_x = [minima_x;a(i)];
                  minima_y = [minima_y;b(i)];
            elseif df_dx_i<0 && df_dy_i<0
                  maxima_x = [maxima_x;a(i)];
                  maxima_y = [maxima_y;b(i)]
            end
      end
end
plot(saddle_x,saddle_y,'*g');
plot(minima_x,minima_y,'*r');
plot(maxima_x,maxima_y,'*b');
legend("cost","saddle","minima","maxima");

% 2.b
for k=1:10
    x(1) = -5+10*rand();
    y(1) = 5+10*rand();
    e = 10^(-8);
    i = 1;
    r = 0.001;
    while True
        deltaf = [subs(df_dx,[X,Y], [x(i),y(i)]) subs(df_dy, [X,Y],
[x(i),y(i)])];
        x(i+1) = x(i)- r*deltaf(1);
        y(i+1) = y(i)- r*deltaf(2);
        i = i+1;
        if norm(deltaf) < e || i> 100000
```

```
            break;
        end
    end
    h(i) = plot(x,y,'.m');
    set(h(i),'handlevisibility','off');
    disp(k)
end

% 2.c
for k=1:10
    x(1) = -5+10*rand();
    y(1) = -5+10*rand();
    e = 10^(-8);
    r = 1e-2;
    i = 1;
    while True
        deltaf = [subs(df_dx,[X,Y], [x(i),y(i)]) subs(df_dy, [X,Y],
[x(i),y(i)])];
        ddf_dxdx_i = subs(ddf_dxdx, [X,Y], [x(i),y(i)]);
        ddf_dydy_i = subs(ddf_dydy, [X,Y], [x(i),y(i)]);
        ddf_dxdy_i = subs(ddf_dxdy, [X,Y], [x(i),y(i)]);
        Hf = [ddf_dxdx_i, ddf_dxdy_i; ddf_dxdy_i, ddf_dydy_i];
        H_inv = inv(Hf);
        x(i+1) = x(i) - r*H_inv(1,:)*deltaf';
        y(i+1) = y(i) - r*H_inv(2,:)*deltaf';
        i = i+1;
        if norm(deltaf) < e || i> 100000
            break;
        end
    end
    h(i) = plot(x,y,'.c');
    set(h(i),'handlevisibility','off');
    disp(k)
end
hold off;
```


cost
saddle
minima
maxima
5
4
3
2
1
0
-1
-2
-3
-4
-5
-5
-4
-3
-2
-1
0
1
2
3
4
5

Problem 3 (statistics & probability - 1 hour)

A discrete-valued random variable is defined by its probability mass function $P = \{p_1, p_2, \ldots p_n\}$, where $0 \leq p_i \leq 1$ is the probability of outcome X_i.

One interpretation of the *Shannon entropy* of a random variable $\mathbf{X}$ is the degree of uncertainty in the outcome of $\mathbf{X}$. For discrete-valued random variables the Shannon entropy is calculated as

$$S(\mathbf{X}) = -\sum_i p_i \, log(p_i). \tag{6}$$

3.a. Suppose there is a game with n possible outcomes and probability mass function $p = \{p_1, p_2, \ldots p_n\}$. What P maximizes the entropy of the game? Explain your answer in plain English.

3.b. Show that this choice of p maximizes S (Note that for a concave function f, $\sum_i a_i f(x_i) \leq f(\sum_i a_i x_i)$ for $a_i \geq 0$, $\sum_i a_i = 1$).

3.a $P_1 = P_2 = \cdots = P_n = \frac{1}{n}$

For, every outcome, when they has same probability, their distribution turns to be the most uncertainty, so the value of $S(X)$ will be maximize.

3.b Assum concave function: $f(P_i) = P_i \log(P_i)$

since $\sum_i a_i f(x_i) \leq f(\sum_i a_i x_i)$ for $a_i \geq 0, \sum_i a_i = 1$

$\therefore \left(\frac{1}{n}\sum_{i=1}^{n} P_i\right) \log\left(\frac{1}{n}\sum_{i=1}^{n} P_i\right) \leq \frac{1}{n}\sum_{i=1}^{n} P_i \log P_i = \frac{1}{n}(-S(X))$

$\therefore S(X) \leq -n \cdot \frac{1}{n} \log\frac{1}{n} = \log n$

the maximum achived when $P_1 = P_2 = \cdots P_n = \frac{1}{n}$