CSE 250A. Assignment 6

Hao-en Sung (A53204772)
wrangle1005@gmail.com

November 8, 2016

6.1 EM algorithm

(a) Posterior probability

Sol. From figure, I can derive the probability as follows.

$$
\begin{aligned}
P(A, C|B, D) &= \frac{P(A, B, C, D)}{\sum_{a,c} P(A, B, C, D)} \\
&= \frac{P(A) \cdot P(B|A) \cdot P(C|A, B) \cdot P(D|A, B, C)}{\sum_{a,c} P(A) \cdot P(B|A) \cdot P(C|A, B) \cdot P(D|A, B, C)}
\end{aligned}
$$

□

(b) Posterior probability

Sol. According to marginization property, I have

$$
\begin{aligned}
P(A|B, D) &= \sum_c P(A, C|B, D) \\
P(C|B, D) &= \sum_a P(A, C|B, D).
\end{aligned}
$$

□

(c) Log-likelihood

Sol.

$$
\begin{aligned}
L &= \sum_t \log P(B = b_t, D = d_t) \\
&= \sum_t \log \sum_{a,c} P(A = a, B = b_t, C = c, D = d_t) \\
&= \sum_t \log \sum_{a,c} P(A = a) \cdot P(B = b_t|A = a) \cdot P(C = c|A = a, B = b_t) \cdot P(D|A = a, B = b_t, C = 1)
\end{aligned}
$$

□

(d) EM algorithm

Sol.

$$
\begin{aligned}
P(A = a) &= \frac{\sum_t P(A = a|B = b_t, D = d_t)}{\sum_t \sum_a P(A = a|B = b_t, D = d_t)} \\
&= \frac{\sum_t P(A = a|B = b_t, D = d_t)}{T} \\
P(B = b|A = a) &= \frac{\sum_t P(B = b, A = a|B = b_t, D = d_t)}{\sum_t \sum_b P(B = b, A = a|B = b_t, D = d_t)} \\
&= \frac{\sum_t I(b, b_t) \cdot P(A = a|B = b_t, D = d_t)}{\sum_t P(A = a|B = b_t, D = d_t)} \\
P(C = c|A = a, B = b) &= \frac{\sum_t P(C = c, A = a, B = b|B = b_t, D = d_t)}{\sum_t \sum_c P(C = c, A = a, B = b|B = b_t, D = d_t)} \\
&= \frac{\sum_t I(b, b_t) \cdot P(A = a, C = c|B = b_t, D = d_t)}{\sum_t I(b, b_t) \cdot P(A = a|B = b_t, D = d_t)} \\
P(D = d|A = a, B = b, C = c) &= \frac{\sum_t P(D = d, A = a, B = b, C = c|B = b_t, D = d_t)}{\sum_t \sum_c P(D = d, A = a, B = b, C = c|B = b_t, D = d_t)} \\
&= \frac{\sum_t I(b, b_t) \cdot I(d, d_t) \cdot P(A = a, C = c|B = b_t, D = d_t)}{\sum_t I(b, b_t) \cdot P(A = a, C = c|B = b_t, D = d_t)}
\end{aligned}
$$

□

6.2 EM algorithm for noisy-OR

(a) Show that this "extended" belief network defines the same conditional distribution $P(Y|X)$ as the original one. In particular, starting from

$$P(Y=1|X) = \sum_{Z\in\{0,1\}^n} P(Y=1, Z|X),$$

show that the right hand side of this equation reduces to the noisy-OR CPT with parameters p_i. To perform this marginalization, you will need to exploit various conditional independence relations.

Sol. It can be derived that

$$\begin{aligned}
\sum_{Z\in\{0,1\}^n} P(Y=1, Z|X) &= \sum_{Z\in\{0,1\}^n} P(Y=1|X,Z)\cdot P(Z|X) \\
&= \sum_{Z\in\{0,1\}^n} P(Y=1|Z)\cdot P(Z|X) \\
&= \sum_{Z\in\{0,1\}^n\backslash 0^n} \prod_i^n P(Z_i|X_i) \\
&= 1 - \prod_i^n P(Z_i=0|X_i) \\
&= 1 - \prod_i^n (1-p_i)^{X_i}.
\end{aligned}$$

□

(b) Compute the posterior probability that appears in the E-step of this EM algorithm. In particular, for joint observations $x \in \{0,1\}^n$ and $y \in \{0,1\}$, use Bayes rule to show that:

$$P(Z_i=1, X_i=1|X=x, Y=y) = \frac{yx_ip_i}{1-\prod_j(1-p_j)^{x_j}}$$

Sol. I first consider the case of $y=1$. It is noticeable that $P(Y=1|Z)=1$ because of $Z_i=1$.

$$\begin{aligned}
P(Z_i=1, X_i=1|X=x, Y=y) &= \frac{\sum_{z=\{0,1\}^n} P(Z_i=1, X_i=1, X=x, Z=z, Y=1)}{P(X=x, Y=y)} \\
&= \frac{P(Z_i=1, X_i=1)\cdot P(X_i=1)\cdot \sum_{z_j, j\neq i}\prod_j P(Z_j=z_j, X_j=x_j)\cdot P(Y=1|Z=z)}{P(Y=1|X=x)} \\
&= \frac{p_i\cdot x_i}{1-\prod_j(1-p_j)^{x_j}}
\end{aligned}$$

It is clear that when $y=0$, the results will be 0. Thus, we can combine two cases and have

$$P(Z_i=1, X_i=1|X=x, Y=y) = \frac{p_i\cdot x_i\cdot y}{1-\prod_j(1-p_j)^{x_j}}$$

□

(c) For the data set $\{\vec{x}(t), y(t)\}_{t=1}^T$, show that the EM update for the parameters p_i is given by:

$$p_i \leftarrow \frac{1}{T_i}\sum_t P(Z_i=1, X_i=1|X=x^{(t)}, Y=y^{(t)},$$

where T_i is the number of examples in which $X_i=1$. (You should derive this update as a special case of the general form presented in lecture.)

Sol. Update for M-step can be written as

$$\begin{aligned}
p_i = P(Z_i=1|X_i=1) &= \frac{\sum_t P(Z_i=1, X_i=1|X=x^{(t)}, Y=y^{(t)})}{\sum_t\sum_{z_i} P(Z_i=z_i, X_i=1|X=x^{(t)}, Y=y^{(t)})} \\
&= \frac{\sum_t P(Z_i=1, X_i=1|X=x^{(t)}, Y=y^{(t)})}{\sum_t P(X_i=1|X=x^{(t)}, Y=y^{(t)})} \\
&= \frac{\sum_t P(Z_i=1, X_i=1|X=x^{(t)}, Y=y^{(t)})}{T_i}
\end{aligned}$$

□

(d) Complete the following table:

iteration	number of mistakes M	log-likelihood L
0	195	-1.044560
1	60	-0.504941
2	43	-0.410764
4	42	-0.365127
8	44	-0.347663
16	40	-0.334677
32	37	-0.322593
64	37	-0.314831
128	36	-0.311156
256	36	-0.310161

Table 1: EM algorithm for noisy-OR

Sol. Learned results are recorded in Table 1 □

(e) Turn in your source code. As always, you may program in the language of your choice.

Sol. The code to solve this problem is included as Code 1. □

6.3 Auxiliary function

(a) Consider the function $f(x) = \log\cosh(x)$. Show that the minimum occurs at $x = 0$.

Sol. Origin formula can be rewritten as $f(x) = \log\frac{e^x+e^{-x}}{2}$. Then $f'(x)$ can be obtained as follows.

$$
\begin{aligned}
f'(x) &= \frac{2}{e^x + e^{-x}} \cdot \frac{e^x - e^{-x}}{2} \\
&= \frac{e^x - e^{-x}}{e^x + e^{-x}} \\
&= \text{tanh(x)}
\end{aligned}
$$

By setting $x = 0$, we have $f'(x) = 0$. In other words, $f(x)$ has minimum value at $x = 0$. □

(b) Show that $f''(x) \leq 1$ for all x

Sol. One can rewrite $f'(x)$ as $1 + \frac{-2}{e^{2x}+1}$. Then $f''(x)$ can be obtained as follows.

$$
\begin{aligned}
f''(x) &= 2 \cdot (e^{2x} + 1)^{-2} \cdot 2 \cdot e^{2x} \\
&= 4 \cdot \frac{e^{2x}}{(e^{2x}+1)^2} \\
&= \left(\frac{2}{e^x + e^{-x}}\right)^2 \\
&= \text{sech}^2(x)
\end{aligned}
$$

Since $\text{sech}(x)$ has the range $(0, 1)$, $\text{sech}^2(x)$ also has the range $(0, 1)$. □

(c) Consider the function $Q(x, y) = f(y) + f'(y)(x - y) + \frac{1}{2}(x - y)^2$. Plot $f(x), Q(x, -2)$, and $Q(x, 1)$ as a function of x.

Sol. I use *Wolfram Alpha* to draw the corresponding figure, as shown in Fig. 1

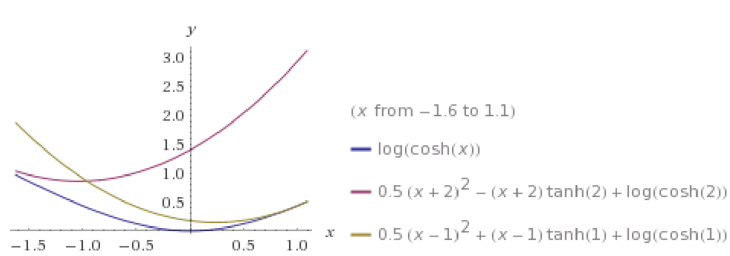

Figure 1: 6-3-(c)

□

(d) Prove that $Q(x, y)$ is an auxiliary function for $f(x)$. In particular, show that it satisfies:

i. $Q(x,x) = f(x)$

Sol. It is obvious that

$$\begin{aligned} Q(x,x) &= f(x) + f'(x)(x-x) + \frac{1}{2}(x-x)^2 \\ &= f(x) \end{aligned}$$

□

ii. $Q(x,y) \geq f(x)$

Sol. Start from $f(x)$, I have

$$\begin{aligned} f(x) &= f(y) + \int_y^x du \left[f'(y) + \int_y^u dv f''(v) \right] \\ &\leq f(y) + \int_y^x du \left[f'(y) + \int_y^u dv \right] \\ &= f(y) + \int_y^x du \left[f'(y) + u - y \right] \\ &= f(y) + \left(f'(y) \cdot u + \frac{1}{2} \cdot u^2 - y \cdot u \right) \Big|_y^x \\ &= f(y) + f'(y) \cdot (x-y) + \frac{1}{2} \cdot x^2 - \frac{1}{2} \cdot y^2 - y \cdot x + y \cdot y \\ &= f(y) + f'(y) \cdot (x-y) + \frac{1}{2} \cdot (x-y)^2 = Q(x,y) \end{aligned}$$

□

(e) Derive the form of the update rule $x_{n+1} = \arg\min_x Q(x, x_n)$.

Sol. By calculating the derivative of $Q(x,y)$, I have

$$\frac{\partial}{\partial x} Q(x, x_n) = f'(x_n) + (x - x_n).$$

To find out the optimum, I can derive

$$\begin{aligned} 0 &= \frac{\partial}{\partial x} Q(x, x_n) \\ x_{n+1} = x &= x_n - f'(x_n) \\ &= x_n - \tanh(x_n) \end{aligned}$$

□

(f) Write a simple program to show that your update rule in (e) converges numerically for the initial guesses $x_0 = -2$ and $x_0 = 1$. Turn in your source code as well as plots of x_n versus n.

Sol. I implement my algorithm in MATLAB as Code 2. The convergence result is shown in Fig. 2.

□

(g) Repeat parts (e) and (f) using the update rule for Newton's method: namely, $x_{n+1} = x_n - f'(x_n)/f''(x_n)$. What happens and why? Determine an upper bound on $|x_0|$ so that Newton's method converges. (Hint: require $|x_1| < |x_0|$.)

Sol. The update rule for Newton's Method can be written as

$$\begin{aligned} x_{n+1} &= x_n - \frac{\tanh(x_n)}{\operatorname{sech}^2(x_n)} \\ &= x_n - \sinh(x_n) \cdot \cosh(x_n) \\ &= x_n - \frac{1}{2} \cdot \sinh(2x_n). \end{aligned}$$

However, with the Code 2 and Fig. 3, it can be found that $x_0 = -2$ will diverge to infinity rather than converge to 0 when n increases.

To make sure Newton's Method to converge, I have the constraint that

$$\begin{aligned} |x_1| &< |x_0| \\ |x_0 - \frac{1}{2} \cdot \sinh(2x_0)| &< |x_0|. \end{aligned}$$

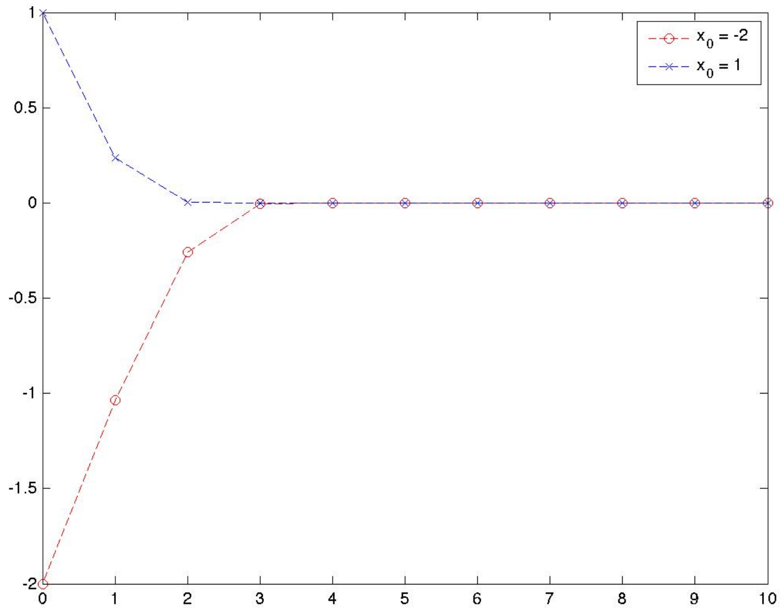

Figure 2: 6-3-(f)

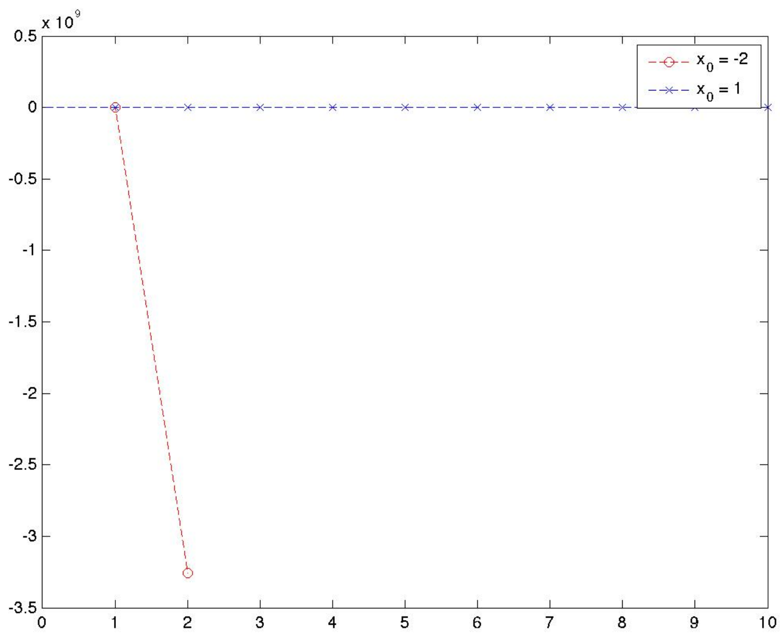

Figure 3: 6-3-(g)

There are overall four cases. Assume $x_0 > 0, x_0 > \frac{1}{2} \cdot \sinh(2x_0)$, then

$$\begin{aligned} x_0 - \frac{1}{2} \cdot \sinh(2x_0) &< x_0 \\ \frac{1}{2} \cdot \sinh(2x_0) &> 0 \\ x_0 &> 0. \end{aligned}$$

However, there is no addition constraint observed.

Then I assume $0 < x_0 < \frac{1}{2} \cdot \sinh(2x_0)$, then

$$\begin{aligned} \frac{1}{2} \cdot \sinh(2x_0) - x_0 &< x_0 \\ \sinh(2x_0) &< 4x_0 \end{aligned}$$

It can be approximated that $0 < x < 1.08866$.

Similarly consider two cases for $x_0 < 0$, I can conclude the approximation of range x_0 as $|x_0| < 1.08866$. $\square$

(h) Plot the function $g(x) = \frac{1}{10} \sum_{k=1}^{1} 0 \log \cosh(x + \frac{1}{k}$. Is it still simple to find the exact minimum?

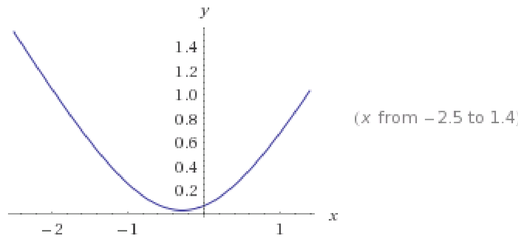

Figure 4: 6-3-(h)

Sol. I use *Wolfram Alpha* to draw the corresponding figure, as shown in Fig. 4.

It is harder to find out the exact minimum, and *Wolfram Alpha* encounters *Standard computation time exceeded.* □

(i) Consider the function $R(x, y) = g(y) + g'(y)(x - y) + \frac{1}{2}(x - y)^2$. Prove that $R(x, y)$ is an auxiliary function for $g(x)$.

i. $R(x, x) = g(x)$

Sol. It is obvious that

$$\begin{aligned} R(x,x) &= g(x) + g'(x)(x-x) + \frac{1}{2}(x-x)^2 \\ &= g(x) \end{aligned}$$

□

ii. $R(x, y) \geq g(x)$

Sol. Exactly the same proof process as (d), except the substitution of $f(\cdot)$ for $g(\cdot)$ and $Q(\cdot,\cdot)$ for $R(\cdot,\cdot)$. □

(j) Derive the form of the update rule $x_{n+1} = \arg\min_x R(x, x_n)$.

Sol. By calculating the derivative of $R(x, y)$, I have

$$\frac{\partial}{\partial x} R(x, x_n) = g'(x_n) + (x - x_n).$$

To find out the optimum, I can derive

$$\begin{aligned} 0 &= \frac{\partial}{\partial x} R(x, x_n) \\ x_{n+1} = x &= x_n - g'(x_n) \\ &= x_n - \frac{1}{10}\sum_{k=1}^{10} \tanh(x_n + \frac{1}{k}) \end{aligned}$$

□

(k) Use the update rule from part (j) to locate the minimum of $g(x)$ to four significant digits. In addition to your answer for the minimum, turn in your source code as well as plots of x_n versus n.

Sol. I implement my algorithm in MATLAB as Code 2. The convergence result $g(x) = 0.0395$ occurs when $x = -0.2830$ as shown in Fig. 5.

□

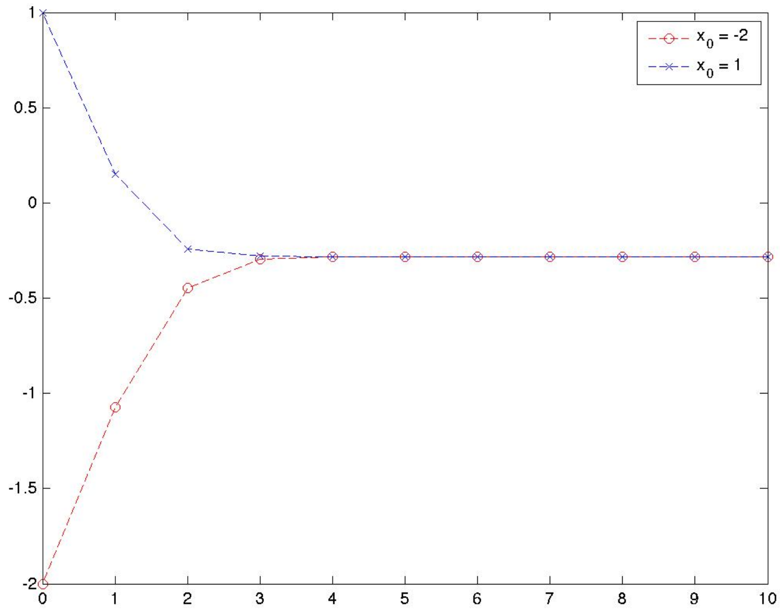

Figure 5: 6-3-(k)

Appendix

```
#include <cstdio>
#include <cstdlib>
#include <cmath>
#include <vector>
#include <unordered_set>

using namespace std;

const int MAX_ITER = 256;
const int NUM_I = 267;
const int NUM_F = 23;

double calLL(vector<vector<int>>& dat_x, vector<int>& dat_y, vector<double>& prob) {
    double ret = 0;
    for (int i = 0; i < NUM_I; i++) {
        double val = 1.0;
        for (int j = 0; j < NUM_F; j++) {
            val *= pow(1-prob[j], dat_x[i][j]);
        }

        if (dat_y[i] == 0) {
            ret += log(val);
        } else {
            ret += log(1-val);
        }
    }
    return ret / NUM_I;
}

int calERR(vector<vector<int>>& dat_x, vector<int>& dat_y, vector<double>& prob) {
    int cnt = 0;
    for (int i = 0; i < NUM_I; i++) {
        double val = 1.0;
        for (int j = 0; j < NUM_F; j++) {
            val *= pow(1-prob[j], dat_x[i][j]);
        }

        if (dat_y[i] == (val >= 0.5)) {
            cnt += 1;
        }
    }
    return cnt;
}

int main() {
    vector<vector<int>> dat_x(NUM_I, vector<int>(NUM_F, 0));
    { // read x
        FILE* pf = fopen("../dat/spectX.txt", "r");
        if (pf == NULL) {
            fprintf(stderr, "cannot open x file\n");
            exit(EXIT_FAILURE);
        }

        int d;
        for (int i = 0; i < NUM_I; i++) {
            for (int j = 0; j < NUM_F; j++) {
```

```
                fscanf(pf, "%d", &d);
                dat_x[i][j] = d;
            }
        }

        fclose(pf);
    }

    vector<int> dat_y(NUM_I, 0);
    { // read y
        FILE* pf = fopen("../dat/spectY.txt", "r");
        if (pf == NULL) {
            fprintf(stderr, "cannot open y file\n");
            exit(EXIT_FAILURE);
        }

        int d;
        for (int i = 0; i < NUM_I; i++) {
            fscanf(pf, "%d", &d);
            dat_y[i] = d;
        }

        fclose(pf);
    }

    vector<double> prob(NUM_F, 1.0 / NUM_F);
    printf("Iter %d: %d %f\n", 0,
      calERR(dat_x, dat_y, prob), calLL(dat_x, dat_y, prob));

    unordered_set<int> um;
    for (int i = 0; i <= 8; i++) {
        um.insert((1 << i));
    }

    for (int times = 1; times <= MAX_ITER; times++) {
        vector<double> nprob(NUM_F, 0);
        vector<int> cnt(NUM_F, 0); // T_i
        for (int i = 0; i < NUM_I; i++) {
            double val = 1.0;
            for (int j = 0; j < NUM_F; j++) {
                cnt[j] += dat_x[i][j];
                val *= pow(1-prob[j], dat_x[i][j]);
            }
            for (int j = 0; j < NUM_F; j++) {
                nprob[j] += dat_y[i] * dat_x[i][j] * prob[j] / (1-val);
            }
        }
        for (int j = 0; j < NUM_F; j++) {
            nprob[j] /= cnt[j];
        }
        prob = nprob;

        if (um.find(times) != um.end()) {
            printf("Iter %d: %d %f\n", times,
              calERR(dat_x, dat_y, prob), calLL(dat_x, dat_y, prob));
        }
    }
}
```

Listing 1: Code for 6-2

```
%% Parameters
max_iter = 10;


%% 6-3 (f)
res = figure('visible', 'off');

x1 = -2;
x2 = 1;
lst1 = [x1; zeros(max_iter, 1)];
lst2 = [x2; zeros(max_iter, 1)];
for i = 1:max_iter
    x1 = x1 - tanh(x1);
    lst1(i+1) = x1;
    x2 = x2 - tanh(x2);
    lst2(i+1) = x2;
end

plot(0:max_iter, lst1, '--ro', 0:max_iter, lst2, '--bx');
legend('x_0 = -2', 'x_0 = 1');
saveas(res, '../res/6-3-f.jpg');


%% 6-3 (g)
```

```
res = figure('visible', 'off');

x1 = -2;
x2 = 1;
lst1 = [x1; zeros(max_iter, 1)];
lst2 = [x2; zeros(max_iter, 1)];
for i = 1:max_iter
    x1 = x1 - tanh(x1) / sech(x1)^2 ;
    lst1(i+1) = x1;
    x2 = x2 - tanh(x2) / sech(x2)^2;
    lst2(i+1) = x2;
end

plot(0:max_iter, lst1, '--ro', 0:max_iter, lst2, '--bx');
legend('x_0 = -2', 'x_0 = 1');
saveas(res, '../res/6-3-g.jpg');


%% 6-3 (k)
res = figure('visible', 'off');

x1 = -2;
x2 = 1;
lst1 = [x1; zeros(max_iter, 1)];
lst2 = [x2; zeros(max_iter, 1)];
for i = 1:max_iter
    t1 = 0;
    t2 = 0;
    for j = 1:10
        t1 = t1 + tanh(x1 + 1/j);
        t2 = t2 + tanh(x2 + 1/j);
    end
    x1 = x1 - 1/10 * t1;
    lst1(i+1) = x1;
    x2 = x2 - 1/10 * t2;
    lst2(i+1) = x2;
end

fprintf('when x = %f, g(x) has minimum value %f\n', x2, log(cosh(x2)));

plot(0:max_iter, lst1, '--ro', 0:max_iter, lst2, '--bx');
legend('x_0 = -2', 'x_0 = 1');
saveas(res, '../res/6-3-k.jpg');
```

Listing 2: Code for 6-3