Assignment 6

Question 1.1

$$y_i \in \{-1, 1\}$$

Odds ratio

$$\frac{p(y_i|w^T x_i)}{p(-y_i|w^T x_i)}$$

Linear model

$$\log\left(\frac{p(y_i|w^T x_i)}{p(-y_i|w^T x_i)}\right) = w^T x.$$

Objective function

$$\underset{w\in\mathbb{R}^d}{\operatorname{argmin}} \sum_{i=1}^{n} -\log(p(y_i|w^T x_i)).$$

Question 1.1

Odds ratio

$$\frac{p(y_i|w^T x_i)}{p(-y_i|w^T x_i)}$$

Linear model

$$\boxed{\log\left(\frac{p(y_i|w^T x_i)}{p(-y_i|w^T x_i)}\right) = w^T x.}$$

Objective function

$$\underset{w\in\mathbb{R}^d}{\operatorname{argmin}} \sum_{i=1}^{n} -\log(p(y_i|w^T x_i)).$$

Starting from equation 1

First step

replace $p(-y_i|w^T x_i)$ with $p(y_i|w^T x_i)$ using the fact that,

$$p(y_i|w^T x_i) + p(-y_i|w^T x_i) = 1$$

Second step

Apply "exp" on both sides to get rid of the log

Third step

Solve for $p(y_i|w^T x_i)$ and plug it into the objective function

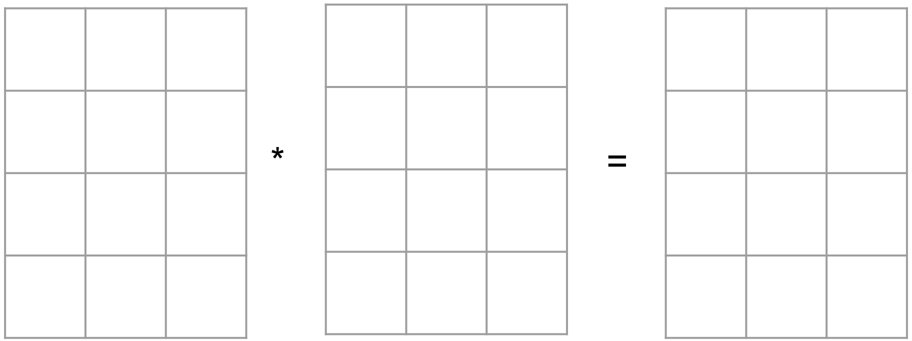
Question 1.2 One-vs-all Logistic Regression

```
% Classification using one-vs-all least squares

% Compute sizes
[n,d] = size(X);
k = max(y);

W = zeros(d,k); % Each column is a classifier
for c = 1:k
    yc = ones(n,1); % Treat class 'c' as (+1)
    yc(y ~= c) = -1; % Treat other classes as (-1)
    W(:,c) = (X'*X)\(X'*yc);
end

model.W = W;
model.predict = @predict;
end

function [yhat] = predict(model,X)
W = model.W;
    [~,yhat] = max(X*W,[],2);
end
```

Question 1.2 One-vs-all Logistic Regression

```
% Classification using one-vs-all least squares

% Compute sizes
[n,d] = size(X);
k = max(y);

W = zeros(d,k); % Each column is a classifier
for c = 1:k
    yc = ones(n,1); % Treat class 'c' as (+1)
    yc(y ~= c) = -1; % Treat other classes as (-1)
    W(:,c) = (X'*X)\(X'*yc);
end

model.W = W;
model.predict = @predict;
end

function [yhat] = predict(model,X)
W = model.W;
    [~,yhat] = max(X*W,[],2);
end
```

Matrix X   Matrix W   Matrix y



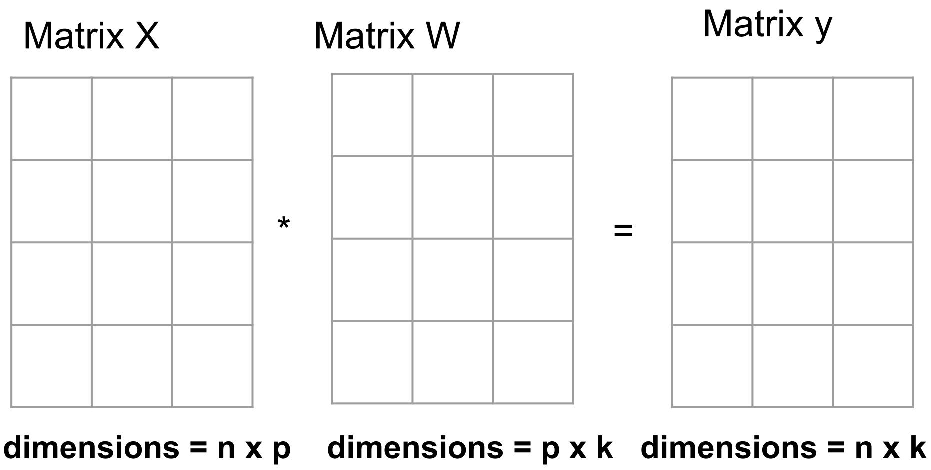
dimensions = ?   dimensions = ?   dimensions = ?

n samples, k classes, p features

use *findMin* with *LogisticLoss* instead (see assignment 4 for the *LogisticLoss* function)

Question 1.2 One-vs-all Logistic Regression

```
% Classification using one-vs-all least squares

% Compute sizes
[n,d] = size(X);
k = max(y);

W = zeros(d,k); % Each column is a classifier
for c = 1:k
    yc = ones(n,1); % Treat class 'c' as (+1)
    yc(y ~= c) = -1; % Treat other classes as (-1)
    W(:,c) = (X'*X)\(X'*yc);
end

model.W = W;
model.predict = @predict;
end

function [yhat] = predict(model,X)
W = model.W;
    [~,yhat] = max(X*W,[],2);
end
```



n samples, k classes, p features

use *findMin* with *LogisticLoss* instead (see assignment 4 for the *LogisticLoss* function)

Question 1.3 Softmax Loss and derivative

- Get the negative log likelihood loss and its derivative using softmax function

$$p(y_i|W, x_i) = \frac{\exp(w_{y_i}^T x_i)}{\sum_{c'=1}^{k} \exp(w_c'^T x_i)}.$$

Question 1.3 Softmax Loss and derivative

- Get the negative log likelihood loss and its derivative using softmax function

$$p(y_i|W, x_i) = \frac{\exp(w_{y_i}^T x_i)}{\sum_{c'=1}^{k} \exp(w_c'^T x_i)}.$$

- We will derive it for the one training example 2 class case

$$p(y_{11}|W, x_1) = \frac{\exp(w_{11} x_1)}{\exp(w_{11} x_1)+\exp(w_{12} x_1)}$$

$$p(y_{12}|W, x_1) = \frac{\exp(w_{12} x_1)}{\exp(w_{11} x_1)+\exp(w_{12} x_1)}$$

Question 1.3 Softmax Loss and derivative

- Get the negative log likelihood loss and its derivative using softmax function

$$p(y_i|W, x_i) = \frac{\exp(w_{y_i}^T x_i)}{\sum_{c'=1}^{k} \exp(w_c'^T x_i)}.$$

- We will derive it for the one training example 2 class case

$$p(y_{11}|W, x_1) = \frac{\exp(w_{11}x_1)}{\exp(w_{11}x_1)+\exp(w_{12}x_1)} \qquad p(y_{12}|W, x_1) = \frac{\exp(w_{12}x_1)}{\exp(w_{11}x_1)+\exp(w_{12}x_1)}$$

$$\begin{array}{c} x_1 \\ \boxed{} \end{array} * \begin{array}{cc} w_{11} & w_{12} \\ \boxed{} & \boxed{} \end{array} = \begin{array}{cc} p(y_{11}|W, x_1) & p(y_{12}|W, x_1) \\ \boxed{} & \boxed{} \end{array}$$

- The negative logarithmic for $p(y_{11}|W, x_1)$ is,

$\log(p(y_{11}|W, x_1)) =$? (apply log)

$-\log(p(y_{11}|W, x_1)) =$? (multiply by -1)

Question 1.3 Softmax Loss and derivative

- Get the negative log likelihood loss and its derivative using softmax function

$$p(y_i|W, x_i) = \frac{\exp(w_{y_i}^T x_i)}{\sum_{c'=1}^{k} \exp(w_c'^T x_i)}.$$

- We will derive it for the one training example 2 class case

$$p(y_{11}|W, x_1) = \frac{\exp(w_{11}x_1)}{\exp(w_{11}x_1)+\exp(w_{12}x_1)} \qquad p(y_{12}|W, x_1) = \frac{\exp(w_{12}x_1)}{\exp(w_{11}x_1)+\exp(w_{12}x_1)}$$

$$\begin{array}{c} x_1 \\ \boxed{} \end{array} * \begin{array}{cc} w_{11} & w_{12} \\ \boxed{} & \boxed{} \end{array} = \begin{array}{cc} p(y_{11}|W, x_1) & p(y_{12}|W, x_1) \\ \boxed{} & \boxed{} \end{array}$$

- The negative logarithmic for $p(y_{11}|W, x_1)$ is,

$$\log(p(y_{11}|W, x_1)) = \log(\frac{\exp(w_{11}x_1)}{\exp(w_{11}x_1)+\exp(w_{12}x_1)}) = \log(\exp(w_{11}x_1)) - \log(\exp(w_{11}x_1) + \exp(w_{12}x_1))$$ (apply log)

$$-\log(p(y_{11}|W, x_1)) = -\log(\exp(w_{11}x_1)) + \log(\exp(w_{11}x_1) + \exp(w_{12}x_1))$$ (multiply by -1)

Question 1.3 Softmax Loss and derivative

- Get the negative log likelihood loss and its derivative using softmax function

$$p(y_i|W, x_i) = \frac{\exp(w_{y_i}^T x_i)}{\sum_{c'=1}^{k} \exp(w_c'^T x_i)}.$$

- We will derive it for the one training example 2 class case

$$p(y_{11}|W,x_1) = \frac{\exp(w_{11}x_1)}{\exp(w_{11}x_1)+\exp(w_{12}x_1)} \qquad p(y_{12}|W,x_1) = \frac{\exp(w_{12}x_1)}{\exp(w_{11}x_1)+\exp(w_{12}x_1)}$$

$$\begin{array}{c} x_1 \\ \boxed{} \end{array} * \begin{array}{cc} w_{11} & w_{12} \\ \boxed{} & \boxed{} \end{array} = \begin{array}{cc} p(y_{11}|W,x_1) & p(y_{12}|W,x_1) \\ \boxed{} & \boxed{} \end{array}$$

- The negative logarithmic for $p(y_{11}|W,x_1)$ is,

$$\log(p(y_{11}|W,x_1)) = \log\left(\frac{\exp(w_{11}x_1)}{\exp(w_{11}x_1)+\exp(w_{12}x_1)}\right) = w_{11}x_1 - \log(\exp(w_{11}x_1) + \exp(w_{12}x_1))$$ (apply log)

$$-\log(p(y_{11}|W,x_1)) = -w_{11}x_1 + \log(\exp(w_{11}x_1) + \exp(w_{12}x_1))$$ (multiply by -1)

Question 1.3 Softmax Loss and derivative

- Get the negative log likelihood loss and its derivative using softmax function

$$p(y_i|W,x_i) = \frac{\exp(w_{y_i}^T x_i)}{\sum_{c'=1}^{k}\exp(w_c'^T x_i)}.$$

$$y_1 = [1,0]$$

- We will derive it for the one training example 2 class case

$$p(y_{11}|W,x_1) = \frac{\exp(w_{11}x_1)}{\exp(w_{11}x_1)+\exp(w_{12}x_1)}$$

$$p(y_{12}|W,x_1) = \frac{\exp(w_{12}x_1)}{\exp(w_{11}x_1)+\exp(w_{12}x_1)}$$

$$\begin{array}{c} x_1 \\ \boxed{} \end{array} * \begin{array}{cc} w_{11} & w_{12} \\ \boxed{} & \boxed{} \end{array} = \begin{array}{cc} p(y_{11}|W,x_1) & p(y_{12}|W,x_1) \\ \boxed{} & \boxed{} \end{array}$$

- The negative logarithmic for $p(y_{11}|W,x_1)$ is,

$$\log(p(y_{11}|W,x_1)) = \log\left(\frac{\exp(w_{11}x_1)}{\exp(w_{11}x_1)+\exp(w_{12}x_1)}\right) = w_{11}x_1 - \log(\exp(w_{11}x_1)+\exp(w_{12}x_1))$$ (apply log)

$$\boxed{-\log(p(y_{11}|W,x_1)) = -w_{11}x_1 + \log(\exp(w_{11}x_1)+\exp(w_{12}x_1))}$$ (multiply by -1)

Question 1.3 Softmax derivative

- Therefore the negative log likelihood,

$$f(W) = -y_{11}\log(p(y_{11}|W,x_1)) - y_{12}\log(p(y_{12}|W,x_1))$$

Question 1.3 Softmax derivative

- Therefore the negative log likelihood,

$$f(W) = -y_{11}\log(p(y_{11}|W,x_1)) - y_{12}\log(p(y_{12}|W,x_1))$$

- Let $y_1 = [1,0]$

Question 1.3 Softmax derivative

- Therefore the negative log likelihood,

$$f(W) = -y_{11}\log(p(y_{11}|W,x_1)) - y_{12}\log(p(y_{12}|W,x_1))$$

- Let $y_1 = [\underset{y_{11}}{1}, \underset{y_{12}}{0}]$

- Therefore,

$$f(W) = -w_{11}x_1 + \log(\exp(w_{11}x_1) + \exp(w_{12}x_1))$$

- The derivative with respect to w_{11} and w_{12} are,

$$\frac{\partial f}{\partial w_{11}} = ?$$

$$\frac{\partial f}{\partial w_{12}} = ?$$

Question 1.3 Softmax derivative

- Therefore the negative log likelihood loss function,

$$f(W) = -w_{11}x_1 + \log(\exp(w_{11}x_1) + \exp(w_{12}x_1))$$

$$y_1 = [1, 0]$$

- The derivative with respect to w_{11} and w_{12} are,

$$\frac{\partial f}{\partial w_{11}} = -x_1 + \frac{1}{\exp(w_{11}x_1) + \exp(w_{12}x_1)} \cdot \exp(w_{11}x_1) \cdot (x_1)$$

$$\frac{\partial f}{\partial w_{12}} = \frac{1}{\exp(w_{11}x_1) + \exp(w_{12}x_1)} \cdot \exp(w_{12}x_1) \cdot (x_1)$$

Question 1.4 - Softmax Classifier

```
function [model] = leastSquaresClassifier(X,y)
% Classification using one-vs-all least squares

% Compute sizes
[n,d] = size(X);
k = max(y);

W = zeros(d,k); % Each column is a classifier
for c = 1:k
    yc = ones(n,1); % Treat class 'c' as (+1)
    yc(y ~= c) = -1; % Treat other classes as (-1)
    W(:,c) = (X'*X)\(X'*yc);
end

model.W = W;
model.predict = @predict;
end
```

Question 1.4 - Softmax Classifier

```
function [model] = leastSquaresClassifier(X,y)
% Classification using one-vs-all least squares

% Compute sizes
[n,d] = size(X);
k = max(y);

W = zeros(d,k); % Each column is a classifier
for c = 1:k
    yc = ones(n,1); % Treat class 'c' as (+1)
    yc(y ~= c) = -1; % Treat other classes as (-1)
    W(:,c) = (X'*X)\(X'*yc);
end

model.W = W;
model.predict = @predict;
end
```

Use findMin instead with the softmax loss grad function

Question 1.4 - Softmax Classifier

```matlab
function [model] = leastSquaresClassifier(X,y)
% Classification using one-vs-all least squares

% Compute sizes
[n,d] = size(X);
k = max(y);

W = zeros(d,k); % Each column is a classifier
for c = 1:k
    yc = ones(n,1); % Treat class 'c' as (+1)
    yc(y ~= c) = -1; % Treat other classes as (-1)
    W(:,c) = (X'*X)\(X'*yc);
end

model.W = W;
model.predict = @predict;
end
```

```matlab

W = zeros(d,k); % Each column is a classifier
% findMin expects 1-dimensional parameter vector
% Therefore use W(:) to get W's 1-dimensional form
W(:) = findMin(@yourSoftmaxLossFunction, W(:), ....)

model.W = W;
model.predict = @predict;
end

function [loss, grad] = yourSoftmaxLossFunction(w, X, y, k)
    % reshape w's dimensions to "p x k"
    % p is the number of features, k is the number of classes
    W = reshape(w, [p k]);

    % Compute loss
    loss = the softmax loss function you derived for Q1.3

    % Compute gradient
    grad = the softmax gradient function you derived for Q1.3

    % reshape grad's dimensions to "1 x (p * k)"
    % i.e. convert the grad matrix to a 1-dimensional vector
    grad = reshape(grad, [p*k 1]);
end
```

Change the contents of the green box on the left using that of the green boxes on the right

Question 1.4 - Softmax Classifier

```matlab
function [model] = leastSquaresClassifier(X,y)
% Classification using one-vs-all least squares

% Compute sizes
[n,d] = size(X);
k = max(y);

W = zeros(d,k); % Each column is a classifier
for c = 1:k
    yc = ones(n,1); % Treat class 'c' as (+1)
    yc(y ~= c) = -1; % Treat other classes as (-1)
    W(:,c) = (X'*X)\(X'*yc);
end

model.W = W;
model.predict = @predict;
end
```

```matlab

W = zeros(d,k); % Each column is a classifier
% findMin expects 1-dimensional parameter vector
% Therefore use W(:) to get W's 1-dimensional form
W(:) = findMin(@yourSoftmaxLossFunction, W(:), ....)

model.W = W;
model.predict = @predict;
end

function [loss, grad] = yourSoftmaxLossFunction(w, X, y, k)
    % reshape w's dimensions to "p x k"
    % p is the number of features, k is the number of classes
    W = reshape(w, [p k]);

    % Compute loss
    loss = the softmax loss function you derived for Q1.3

    % Compute gradient
    grad = the softmax gradient function you derived for Q1.3

    % reshape grad's dimensions to "1 x (p * k)"
    % i.e. convert the grad matrix to a 1-dimensional vector
    grad = reshape(grad, [p*k 1]);
end
```

Change the contents of the green box on the left using that of the green boxes on the right

Question 1.5 - Cost of Multinomial Logistic Regression

Time complexity for processing one example = ?

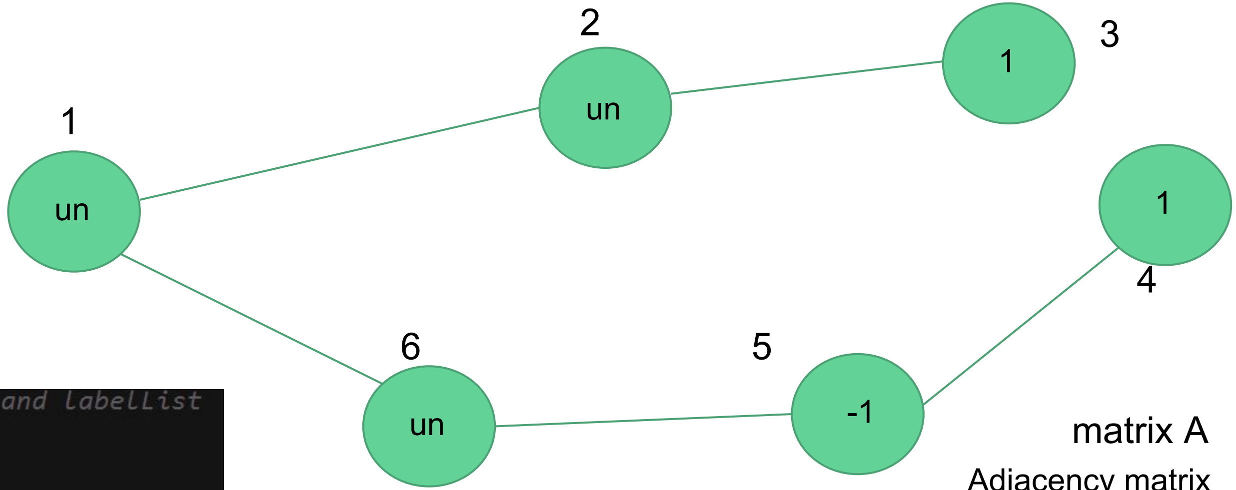
```
# Training
# Run for T iterations
for t = 1 to T
    # Loop over training examples
    for i = 1 to n
            for k = 1 to K
                softmax_value(i,k) = compute softmax for class k for training example i over the 'd' features
    for j = 1 to d
            for k = 1 to K
                softmax_gradient(j, k) = compute the gradient for the cofficient of feature j of class k using softmax_value
    for j = 1 to d
        for k = 1 to K
            update w(j,k) using softmax_gradient(j, k)
# Testing
# Loop over test examples
for i = 1 to n_test
    for k = 1 to K
        softmax_value(i,k) = compute softmax for class k for test example i  over the 'd' features
end for

for i = 1 to n_test
    yhat(i) = argmax of softmax_value(i,k) over 'k'
end for
```

Time complexity for predicting one example = ?

Question 1.5 - Cost of Multinomial Logistic Regression

```
# Training
# Run for T iterations
for t = 1 to T
    # Loop over training examples
    for i = 1 to n
            for k = 1 to K
                softmax_value(i,k) = compute softmax for class k for training example i over the 'd' features
    for j = 1 to d
            for k = 1 to K
                softmax_gradient(j, k) = compute the gradient for the cofficient of feature j of class k using softmax_value
    for j = 1 to d
        for k = 1 to K
            update w(j,k) using softmax_gradient(j, k)
# Testing
# Loop over test examples
for i = 1 to n_test
    for k = 1 to K
        softmax_value(i,k) = compute softmax for class k for test example i  over the 'd' features
end for

for i = 1 to n_test
    yhat(i) = argmax of softmax_value(i,k) over 'k'
end for
```

Question 2
random walk




```
load simpleGraph.mat % Loads adjacency A and labelList

n = length(A);
p = zeros(n,2);
r = 100;


for i = 2:n
    for j = 1:r
        % Run random walk
        yhat = runRandomWalk(A,labelList,i);
        if yhat == 1
            p(i,1) = p(i,1) + 1;
        elseif yhat == -1
            p(i,2) = p(i,2) + 1;
        end
    end
end

% Output final probabilities
probabilities = p/r
```

matrix labelList

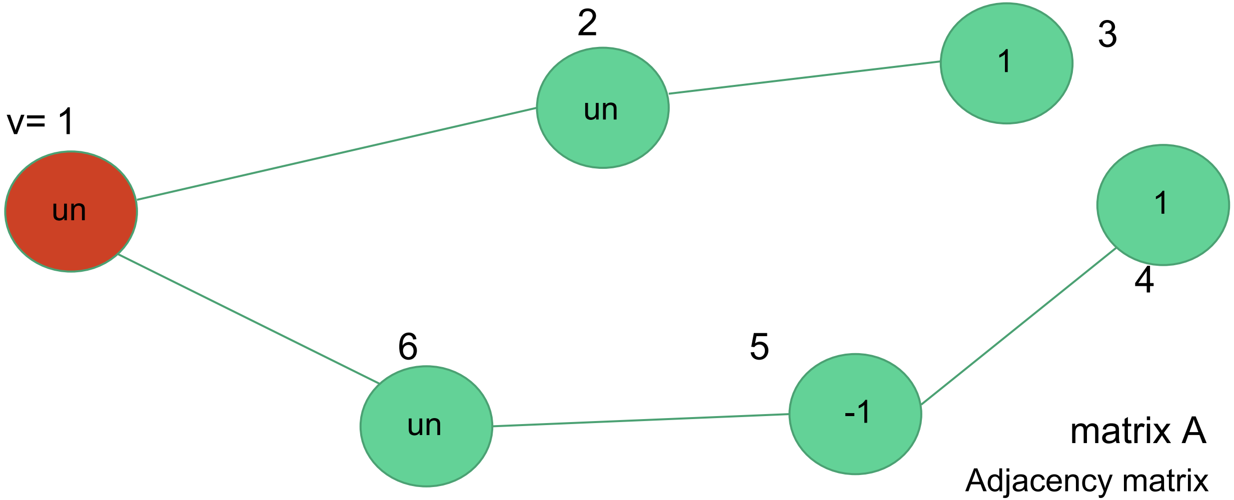
3	1
4	1
5	-1

matrix A

Adjacency matrix

	1	2	3	4	5	6
1	0	1	0	0	0	1
2	1	0	1	0	0	0
3	0	1	0	0	0	0
4	0	0	0	0	1	0
5	0	0	0	1	0	1
6	1	0	0	0	1	0

Question 2
random walk



matrix A

Adjacency matrix

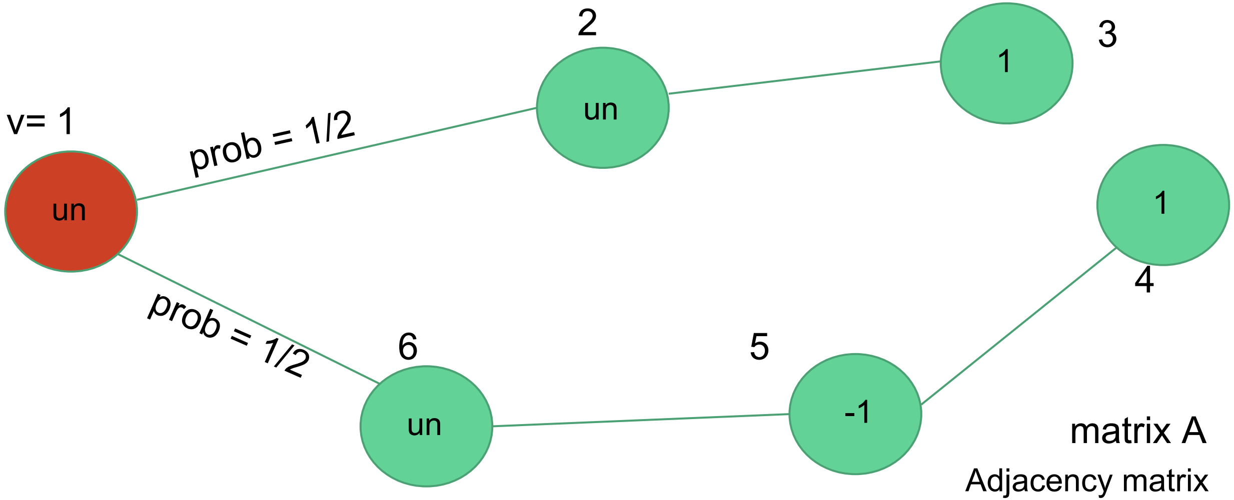
	1	2	3	4	5	6
1	0	1	0	0	0	1
2	1	0	1	0	0	0
3	0	1	0	0	0	0
4	0	0	0	0	1	0
5	0	0	0	1	0	1
6	1	0	0	0	1	0

matrix labelList

3	1
4	1
5	-1

```
% Run random walk
yhat = runRandomWalk(A,labelList,i);
if yhat == 1
    p(i,1) = p(i,1) + 1;
elseif yhat == -1
    p(i,2) = p(i,2) + 1;
end
```

Question 2
random walk



matrix labelList

3	1
4	1
5	-1

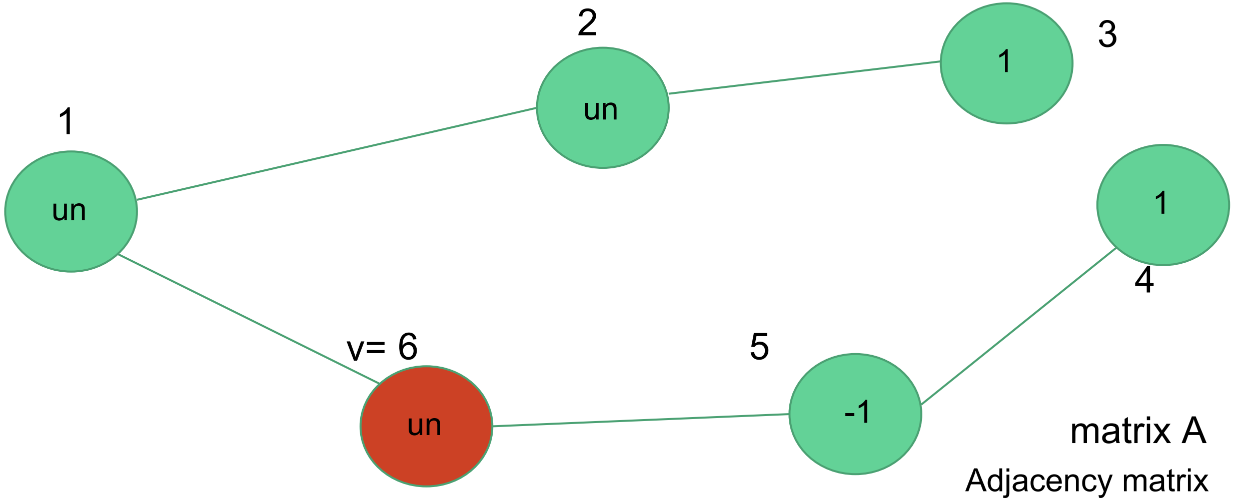
matrix A

Adjacency matrix

	1	2	3	4	5	6
1	0	1	0	0	0	1
2	1	0	1	0	0	0
3	0	1	0	0	0	0
4	0	0	0	0	1	0
5	0	0	0	1	0	1
6	1	0	0	0	1	0

```
% Run random walk
yhat = runRandomWalk(A,labelList,i);
if yhat == 1
    p(i,1) = p(i,1) + 1;
elseif yhat == -1
    p(i,2) = p(i,2) + 1;
end
```

Question 2
random walk



matrix labelList

3	1
4	1
5	-1

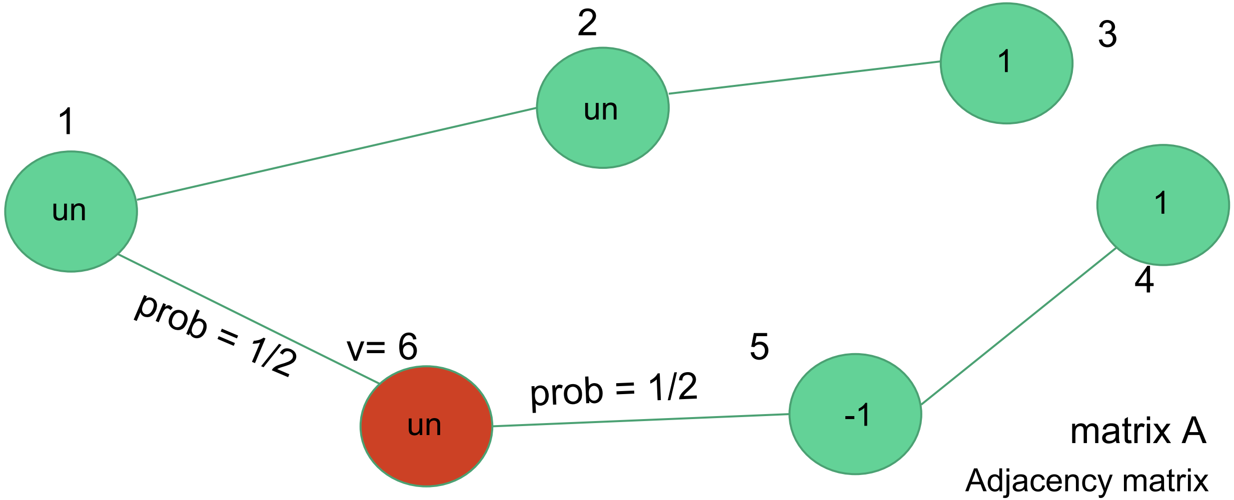
matrix A

Adjacency matrix

	1	2	3	4	5	6
1	0	1	0	0	0	1
2	1	0	1	0	0	0
3	0	1	0	0	0	0
4	0	0	0	0	1	0
5	0	0	0	1	0	1
6	1	0	0	0	1	0

```
% Run random walk
yhat = runRandomWalk(A,labelList,i);
if yhat == 1
    p(i,1) = p(i,1) + 1;
elseif yhat == -1
    p(i,2) = p(i,2) + 1;
end
```

Question 2
random walk



matrix labelList

3	1
4	1
5	-1

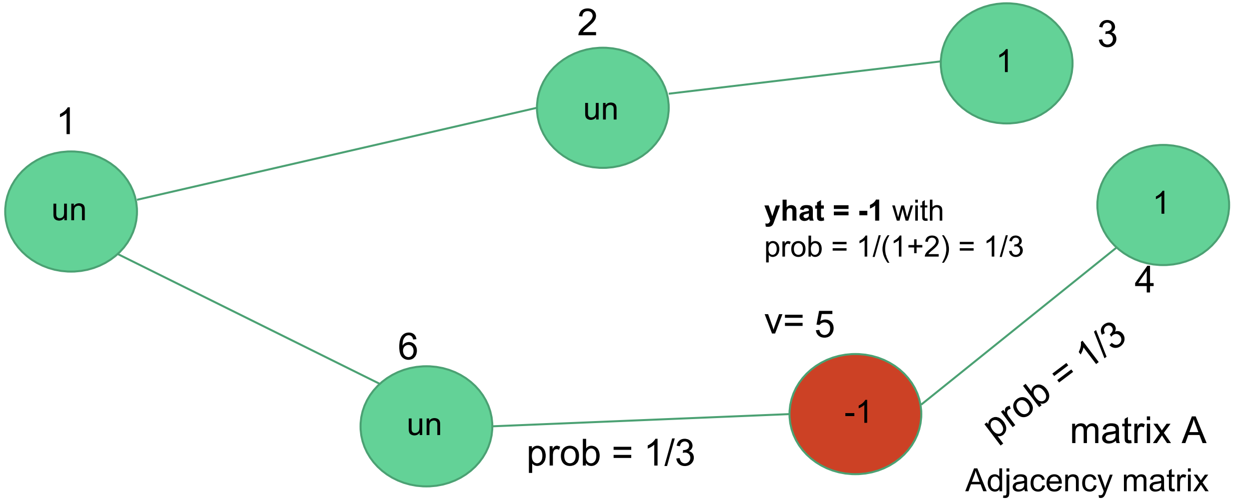
matrix A

Adjacency matrix

	1	2	3	4	5	6
1	0	1	0	0	0	1
2	1	0	1	0	0	0
3	0	1	0	0	0	0
4	0	0	0	0	1	0
5	0	0	0	1	0	1
6	1	0	0	0	1	0

```
% Run random walk
yhat = runRandomWalk(A,labelList,i);
if yhat == 1
    p(i,1) = p(i,1) + 1;
elseif yhat == -1
    p(i,2) = p(i,2) + 1;
end
```

Question 2
random walk



matrix labelList

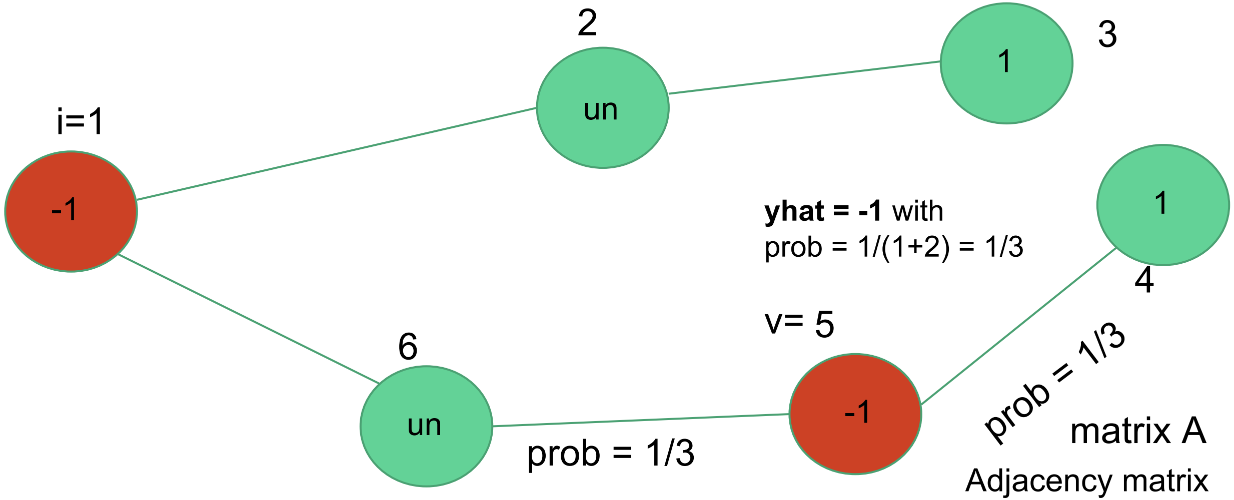
3	1
4	1
5	-1

matrix A
Adjacency matrix

	1	2	3	4	5	6
1	0	1	0	0	0	1
2	1	0	1	0	0	0
3	0	1	0	0	0	0
4	0	0	0	0	1	0
5	0	0	0	1	0	1
6	1	0	0	0	1	0

```
% Run random walk
yhat = runRandomWalk(A,labelList,i);
if yhat == 1
    p(i,1) = p(i,1) + 1;
elseif yhat == -1
    p(i,2) = p(i,2) + 1;
end
```

Question 2
random walk



matrix A

Adjacency matrix

	1	2	3	4	5	6
1	0	1	0	0	0	1
2	1	0	1	0	0	0
3	0	1	0	0	0	0
4	0	0	0	0	1	0
5	0	0	0	1	0	1
6	1	0	0	0	1	0

matrix labelList

3	1
4	1
5	-1

```
% Run random walk
yhat = runRandomWalk(A,labelList,i);
if yhat == 1
    p(i,1) = p(i,1) + 1;
elseif yhat == -1
    p(i,2) = p(i,2) + 1;
end
```