CPSC 540 Assignment 4

The assignment instructions are the same as for the previous assignment.

1. Name(s): Cody Griffith, Ziming Yin, Tim Jaschek
2. Student ID(s): Cody: 88416169 , Ziming: 88489166 , Tim: 91220160.

1 Discrete Markov Chains

1.1 Sampling, Inference, and Decoding

The function *example_markovChain.jl* loads the initial state probabilities and transition probabilities for three Markov chain models on d binary variables,

$$p(x_1, x_2, \dots, x_d) = p(x_1) \prod_{j=2}^{d} p(x_j \mid x_{j-1}).$$

It then finds the optimal decoding (the most likely assignment to the variables $\{x_1, x_2, \dots, x_d\}$) in the first two chains. In the demo, decoding is done by enumerating all possible assignments to the variables. This works for the first two chains as they only have 4 variables, but is too slow to decode the last chain because it has 31 variables. In this question you'll explore two ways to estimate the marginals in the third Markov chain and two ways to estimate the most-probable sequence (you only need to report results on the third Markov chain for this question, but the first two may be helpful for debugging).

1. Write a function, *sampleAncestral.m*, that uses ancestral sampling to sample a sequence x. Hand in this code and report all the univariate marginal probabilities using a Monte Carlo estimate based on 10000 samples.

   Answer: Using ancestral sampling, we computed the following marginal probabilities: Marginals of chain short1:

$$\begin{pmatrix} 0.5962 & 0.3223 & 0.3622 & 0.13 \\ 0.4038 & 0.6777 & 0.6378 & 0.87 \end{pmatrix}$$

Marginals of chain short2:

$$\begin{pmatrix} 0.5979 & 0.3428 & 0.2376 & 0.7722 \\ 0.4021 & 0.6572 & 0.7624 & 0.2278 \end{pmatrix}$$

Marginals of chain long: (due to formatting issues given as in Julia output)

```
M_long = [0.5959 0.464 0.9002 0.9722 0.7779 0.2296 0.0577
   0.3388 0.3213 0.5978 0.3509 0.744 0.371 0.
1254 0.3752 0.4176 0.7205 0.7928 0.3138 0.6691 0.5796 0.2539
   0.4076 0.2442 0.7409 0.6201 0.6051 0.73
95 0.4659 0.4415 0.1825; 0.4041 0.536 0.0998 0.0278 0.2221
   0.7704 0.9423 0.6612 0.6787 0.4022 0.6491
 0.256 0.629 0.8746 0.6248 0.5824 0.2795 0.2072 0.6862 0.3309
    0.4204 0.7461 0.5924 0.7558 0.2591 0.3
799 0.3949 0.2605 0.5341 0.5585 0.8175]
```

Code of sampleAncestral.jl

```
function sampleAncestral(p0,pT,n)
        # n is number of samples
        # p0 is initial distribution
        # pT are transition probabilities
        k = length(p0)
        d = size(pT,3)+1
        X = zeros(n,d)
        for i in 1:n
                X[i,1] = sampleDiscrete(p0)
                for j in 2:d
                        X[i,j] = sampleDiscrete(pT[Int(X[i,j-1]),:,j-1])
                end
        end
        return X
end

function sampleDiscrete(p)
        minimum(find(cumsum(p[:]).> rand()))
end

function computeMarginals(X,p0)
        # computes the marginals from a given sample matrix X
        k = length(p0)
        (n,d) = size(X)
        M = zeros(k,d)
        for i=1:d
```

```
            for j = 1:(k-1)
                M[j,i] = length(find(x->x==j,X[:,i]))/n
            end
            M[k,i] = 1 - sum(M[:,i])
        end
        return M
end
```

2. Write a function, *marginalCK.m*, that uses the CK equations to compute the exact univariate marginals. Hand in this code, report all exact univariate marginals, and report how this differs from the marginals in the previous question.

   Answer: Using the Chapman Kolmogorov Equations (CK), we computed the exact univariate marginals: Marginals of chain short1:

$$\begin{pmatrix} 0.6 & 0.332608 & 0.370063 & 0.128596 \\ 0.4 & 0.667392 & 0.629937 & 0.871404 \end{pmatrix}$$

   Marginals of chain short2:

$$\begin{pmatrix} 0.6 & 0.34826 & 0.233741 & 0.776004 \\ 0.4 & 0.65174 & 0.766259 & 0.223996 \end{pmatrix}$$

   Marginals of chain long: (due to formatting issues given as in Julia output)

```
M_long = [0.6 0.468067 0.899785 0.971825 0.77789 0.226686
   0.0542476 0.335907 0.320373 0.590169 0.360
051 0.747861 0.372626 0.131324 0.374426 0.410331 0.723989
   0.787671 0.314746 0.669811 0.581134 0.2541
49 0.403661 0.240327 0.737836 0.621143 0.605971 0.74189
   0.464382 0.437986 0.178229; 0.4 0.531933 0.1
00215 0.0281746 0.22211 0.773314 0.945752 0.664093 0.679627
   0.409831 0.639949 0.252139 0.627374 0.86
8676 0.625574 0.589669 0.276011 0.212329 0.685254 0.330189
   0.418866 0.745851 0.596339 0.759673 0.262
164 0.378857 0.394029 0.25811 0.535618 0.562014 0.821771]
```

   Code of marginalCK.jl

```
function marginalCK(p0,pT)
    k = length(p0)
    d = size(pT,3)+1
    M = zeros(k,d)
    #use dynamic programming
    M = chapmanStep(d,p0,pT,M)
    return M
```

```
end

function chapmanStep(j,p0,pT,M)
    if j==1
        M[:,1] = p0
    elseif (M[1,j] == 0)
        previousProb = chapmanStep(j-1,p0,pT,M)[:,j-1]
        M[:,j] = pT[:,:,j-1]'*previousProb
    end
    return M
end
```

3. Write a function, *viterbiDecode.m*, that implements the Viterbi decoding algorithm for Markov chains. Hand in this code and report the optimal decoding of the third Markov chain, and how this differs from the sequence defined by the most likely state at each time (the states maximizing the marginal probabilities).

   Answer: Viterbi decoding gives us the following decoding of the Markov chains: chain short1:

   $$\begin{pmatrix} 1.0 & 2.0 & 2.0 & 2.0 \end{pmatrix}$$

   chain short2:

   $$\begin{pmatrix} 1.0 & 2.0 & 2.0 & 1.0 \end{pmatrix}$$

   chain long: (due to formatting issues given as in Julia output)

```
M_long = [1  1  1  1  1  2  2  2  2  1  2  1  2  2  2  2  2  1  2  1  1  2  2  2  1  1
   1  1  1  1  2]
```

   For a comparison, here are the maximizing marginal probabilities from the CK Equations in Problem 1.1.2.

```
M_long = [1  2  1  1  1  2  2  2  2  1  2  1  2  2  2  2  1  1  2  1  1  2  2  2  1  1
   1  1  2  2  2]
```

   They differ in positions 2,17,29 and 30 and therefore have ℓ^1-distance 4. Code of viterbiDecode.jl

```
function viterbiDecode(p0,pT)
    k = length(p0)
    d = size(pT,3)+1
    M = zeros(k,d)
    state = zeros(k,d-1)
    B = zeros(1,d)
    M[:,1]=p0
    for i in 2:d
```

```
            for u in 1:k # later state
                comparematrix = zeros(1,k)
                for v in 1:k #previous state
                    comparematrix[1,v] = pT[v,u,i-1]'*M[v,i-1]
                end
                (M[u,i], state[u,i-1]) = findmax(comparematrix)
            end
        end
        (~,B[1,d]) = findmax(M[:,d])
        #backtrack
        for j in 1:d-1
            previous = Int(B[1,d-j+1])
            B[1,d-j] = state[previous,d-j]
        end
        return B
    end
```

Hint: for parts 2-3, you can use a 2 by d matrix M to represent the dynamic programming table, and for part 3 you can use another matrix B containing the argmax values that lead to each entry in the table.

1.2 Simple Conditioning Queries

The long sequence from the previous question usually starts with state 1 and most of the time ends in state 2. In this question you'll consider conditioning on these events not happening. First, compute the following quantities which can be done using your functions from the previous question:

1. Report all the univariate conditional probabilities $p(x_j \mid x_1 = 2)$ obtained using a Monte Carlo estimate based on 10000 samples.

   Answer: The code

```
print("\n ******* Problem 1.2.1. *******\n")
X = sampleAncestral(p0,pT_long,10000)
M_long = computeMarginals(X,p0)
# Normalization accounts for a different number of examples in Y
Y = X[find(x->x==2,X[:,1]),:]
(numsize,densize)=(size(Y,1),size(X,1))
M_start2 = computeMarginals(Y,p0)
normalization = densize/numsize
p2 = (M_start2/M_long[2,1])/normalization
print("\nConditional marginals of chain long using MC\n")
```

```
display(p2)
```

yields

$$\begin{pmatrix} 0.0 & 0.0650888 & 0.931213 & 0.978057 & \dots & 0.741864 & 0.471154 & 0.435897 & 0.183679 \\ 1.0 & 0.934911 & 0.068787 & 0.0219428 & \dots & 0.258136 & 0.528846 & 0.564103 & 0.816321 \end{pmatrix}$$

for more entries please run the file example_Markov.jl.

2. Report all the exact univariate conditionals $p(x_j \mid x_1 = 2)$.

   Answer: The code

```
print("\n ******* Problem 1.2.2. *******\n")
# Start using given conditional information
y0 = [0  1]
M_start2 = marginalCK(y0, pT_long)
p2 = M_start2
print("\nConditional marginals of chain long using CK\n")
display(p2)
```

   yields

$$\begin{pmatrix} 0.0 & 0.0634045 & 0.929715 & 0.97558 & \dots & 0.74189 & 0.464382 & 0.437986 & 0.178229 \\ 1.0 & 0.936595 & 0.0702846 & 0.0244196 & \dots & 0.25811 & 0.535618 & 0.562014 & 0.821771 \end{pmatrix}$$

   for more entries please run the file example_Markov.jl. Note that our MC estimation in 1.2.1 gave us very close results.

3. Report the sequence beginning with $x_1 = 2$ that has the highest probability. How does this differ from the (unconditional) optimal decoding?

   Answer: Viterbi Decoding with $x_1 = 2$ gives the decoding:

$$(22111222212122222122112221111112)$$

   It is remarkable, that this differs for example in the first two positions from the unconditional optimal decoding. Changing the first slot therefore can change the entire decoding. ℓ^1-distance to unconditioned optimal decoding: 2.

4. Report the sequence ending with $x_d = 1$ that has the highest probability. How does this differ from the (unconditional) optimal decoding?

   Answer: Viterbi Decoding with $x_1 = 1$ gives the decoding:

$$(11111222212122222122112221111111)$$

   Once again we see that conditioning on the first variable can effect the parameter of the last variable in the decoding (this is one of the one that differs with the unconditional optimal decoding). ℓ^1-distance to unconditioned optimal decoding: 1.

Hint: these conditions can be done by changing the input to the functions from the previous question.

1.3 Conditioning Queries Requiring Inference

Next consider the following cases (which require implementing an extra rejection step or backward phase):

1. Report all the univariate conditional probabilities $p(x_j \mid x_d = 1)$ obtained using a Monte Carlo estimate based on 10000 samples and rejection sampling. Also report the number of samples accepted among the 10000 samples.

   Answer: The code

```
X = sampleAncestral(p0,pT_long,10000)
M_long = computeMarginals(X,p0)
# Normalization accounts for a different number of examples in Y
Y = X[find(x->x==1,X[:,end]),:]
(numsize,densize)=(size(Y,1),size(X,1))
M_start2 = computeMarginals(Y,p0)
normalization = densize/numsize
p2 = (M_start2/M_long[1,end])/normalization
print("\nConditional marginals with xd=1 of chain long using MC\n")
display(p2)
print("The number of samples in Y is: ", numsize, "\n")
```

   yields the conditional marginals with $x_d = 1$ of chain long using MC

$$\begin{pmatrix} 0.593768 & 0.46119 & 0.896317 & 0.967705 & \dots & 0.726912 & 0.447592 & 0.325779 & 1.0 \\ 0.406232 & 0.53881 & 0.103683 & 0.0322946 & \dots & 0.273088 & 0.552408 & 0.674221 & 0.0 \end{pmatrix}$$

   The number of samples in Y is: 1765. For more entries please run the file example_Markov.jl.

2. Write a function, *sampleBackwards.m* that uses backwards sampling to sample sequences where $x_d = 1$). Hand in this code and report all the univariate conditional probabilities $p(x_j \mid x_d = 1)$ obtained using a Monte Carlo estimate based on 10000 samples.

   Answer: Code of sampleBackwards.jl

```
include("problem1_functions.jl")

function sample_backwards(p0,pT,n,xd)
    # compute p(x)
    M=marginalCK(p0,pT)
    # p(y|x)*p(x)/sum p(y|x)*p(x)
    k = length(p0)
    d = size(pT,3)+1
    X = zeros(n,d)
```

```
    X[:,d]=xd
    for i in 1:n
        for j in d-1:-1:1
            p_numerator = pT[:, Int(X[i,j+1]), j] .* M[:,j]
            p_dominator = sum(p_numerator)
            p_back =  p_numerator /p_dominator
            X[i,j] = sampleDiscrete(p_back)
        end
    end
    return X
end
```

conditional probabilities:

$$\begin{pmatrix} 0.598 & 0.4665 & 0.8975 & 0.9719 & \dots & 0.7411 & 0.4384, & 0.3225 & 1.0 \\ 0.402 & 0.5335 & 0.1025 & 0.0281 & \dots & 0.2589 & 0.5616 & 0.6775 & 0.0 \end{pmatrix}$$

For more entries please run the file Prob1_3_2.jl.

3. Write a function, *forwardBackwards.m* that is able compute all exact univariate conditionals $p(x_j \mid x_d = 1)$ in $O(dk^2)$. Hand in the code and report all the exact univariate conditionals $p(x_j \mid x_d = 1)$.

   Answer: Code of forwardBackwards.jl

```
include("problem1_functions.jl")

function forwardBackward(p0,pT,xd)
    M=marginalCK(p0,pT)
    k = length(p0)
    d = size(pT,3)+1
    V = backward(xd,p0,pT)
    Z = M.*V
    for j in 1:d
        Z[:,j] = Z[:,j]/sum(Z[:,j])
    end
    return Z
end


function backward(xd,p0,pT)
    k = length(p0)
    d = size(pT,3)+1
    V = zeros(k,d)
    V[xd,d]=1
```

```
    for i in d-1:-1:1
        V[:,i] = pT[:,:,i]*V[:,i+1]
    end
    return V
end
```

conditional probabilities:

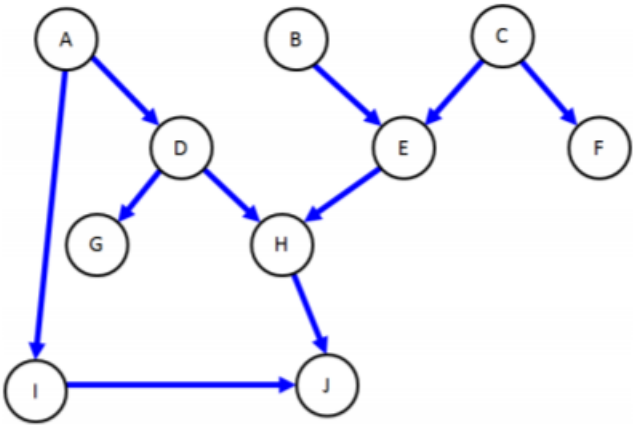
$$\begin{pmatrix} 0.6 & 0.468067 & 0.899785 & \dots & 0.732424 & 0.431823 & 0.329598 & 1.0 \\ 0.4 & 0.531933 & 0.100215 & \dots & 0.267576 & 0.568177 & 0.670402 & 0.0 \end{pmatrix}$$

For more entries please run the file Prob1_3_3.jl.

2 Directed Acyclic Graphical Models

2.1 D-Separation

Consider a directed acyclic graphical (DAG) model with the following graph structure:



Assuming that the conditional independence properties are faithful to the graph, using d-separation briefly explain why the following are true or false:

1. $B \perp F$.

   Answer: True. E is not observed, thus B and C are independent. F is child of C, $B \perp F$.

2. $B \perp F \mid A$.

   Answer: True. A does not block the path.

3. $B \perp F \mid C$.

   Answer: True. Observe C blocks E and F. E is only child of B, B and F now independent.

4. $B \perp F \mid E$.

   Answer: False. Observe E create a path, now it is not d-separated

5. $B \perp F \mid I$.

   Answer: True. I does not derive from E.

6. $B \perp F \mid J$.

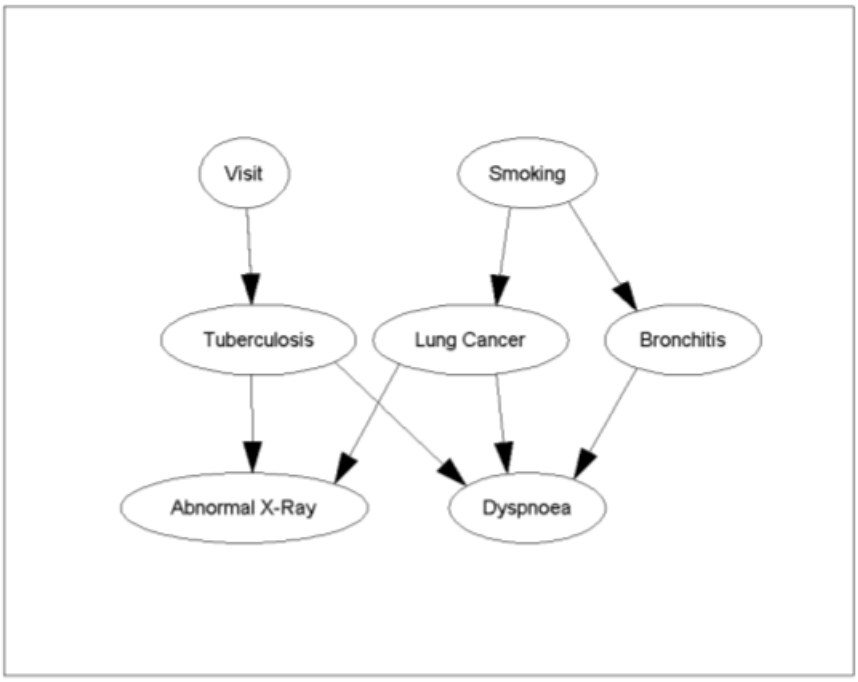
   Answer: False. J is a grandchild of E.

7. $B \perp F \mid C, E$.

   Answer: True. Observe both C and E blocks the path.

2.2 Exact Inference

While DAGs can be used as a visual representation of independence assumptions, they can also be used to simplify computations. This question will give you practice using the basic properties which allow efficient computations in graphical models. Consider the DAG model below, for distinguishing between different causes of shortness-of-breath (dyspnoea) and the causes of an abnormal lung x-ray, while modelling potential causes of these diseases too (whether the person is a smoker or had a 'visit' to a country with a high degree of tuberculosis).



For this question, let's assume that we use the following parameterization of the network:

$$\text{Visit}$$
$$p(V = 1) = 0.01$$
$$\text{Smoking}$$
$$p(S = 1) = 0.2$$
$$\text{Tuberculosis}$$
$$p(T = 1 \mid V = 1) = 0.05$$
$$p(T = 1 \mid V = 0) = 0.01$$
$$\text{Lung Cancer}$$
$$p(L = 1 \mid S = 1) = 0.10$$
$$p(L = 1 \mid S = 0) = 0.01$$
$$\text{Bronchitis}$$
$$p(B = 1 \mid S = 1) = 0.60$$
$$p(B = 1 \mid S = 0) = 0.30$$
$$\text{Abnormal X-Ray}$$
$$p(X = 1 \mid T = 1, L = 1) = 1.00$$
$$p(X = 1 \mid T = 1, L = 0) = 0.98$$
$$p(X = 1 \mid T = 0, L = 1) = 0.9$$

$$p(X = 1 \mid T = 0, L = 0) = 0.05$$

Dyspnoea

$$p(D = 1 \mid T = 1, L = 1, B = 1) = 0.90$$
$$p(D = 1 \mid T = 1, L = 1, B = 0) = 0.70$$
$$p(D = 1 \mid T = 1, L = 0, B = 1) = 0.85$$
$$p(D = 1 \mid T = 1, L = 0, B = 0) = 0.65$$
$$p(D = 1 \mid T = 0, L = 1, B = 1) = 0.82$$
$$p(D = 1 \mid T = 0, L = 1, B = 0) = 0.60$$
$$p(D = 1 \mid T = 0, L = 0, B = 1) = 0.80$$
$$p(D = 1 \mid T = 0, L = 0, B = 0) = 0.10$$

Compute the following quantities (hints are given on the right, and these will be easier to do in order and if you use conditional independence properties to simplify the calculations):

1. $p(S = 0)$ (negation of marginal of root node; use sum to one constraint).

   Answer: $p(S = 0) = 1 - p(S = 1) = 0.8$

2. $p(L = 1)$ (marginal of child node; marginalize over parent).

   Answer: $p(L = 1)$
   $= p(L = 1, S = 1) + p(L = 1, S = 0)$
   $= p(L = 1 \mid S = 1)p(S = 1) + p(L = 1 \mid S = 0)p(S = 0)$
   $= 0.1 * 0.2 + 0.01 * 0.8$
   $= 0.028$

3. $p(X = 1 \mid T = 1)$ (conditional of child with missing parent; marginalize over missing parent).

   Answer: $p(X = 1 \mid T = 1)$
   $= p(X = 1, L = 1 \mid T = 1) + p(X = 1, L = 0 \mid T = 1)$
   $= (p(X = 1, L = 1, T = 1) + p(X = 1, L = 0, T = 1))/p(T = 1)$
   $= (p(X = 1 \mid L = 1, T = 1)p(L = 1, T = 1) + p(X = 1 \mid L = 0, T = 1)p(L = 0, T = 1))/p(T = 1)$
   $= p(X = 1 \mid L = 1, T = 1)p(L = 1 \mid T = 1) + p(X = 1 \mid L = 0, T = 1)p(L = 0 \mid T = 1)$
   $= p(X = 1 \mid L = 1, T = 1)p(L = 1) + p(X = 1 \mid L = 0, T = 1)p(L = 0)$
   $= 0.028 + 0.98 * 0.972$
   $= 0.98056$

4. $p(X = 1 \mid T = 1, S = 1)$ (conditional of child given parent and grand-parent, marginalize over missing parent).

Answer:

$$
\begin{aligned}
& p(X=1 \mid T=1, S=1) \\
=& (p(X=1, L=1, T=1, S=1)+p(X=1, L=0, T=1, S=1)) / p(T=1, S=1) \\
=& [p(X=1 \mid L=1, T=1, S=1) p(L=1, S=1, T=1) \\
& +p(X=1 \mid L=0, T=1, S=1) p(L=0, T=1, S=1)] / p(T=1, S=1) \\
=& p(X=1 \mid L=1, T=1, S=1) p(L=1 \mid S=1, T=1) \\
& +p(X=1 \mid L=0, T=1, S=1) p(L=0 \mid T=1, S=1) \\
=& p(X=1 \mid L=1, T=1, S=1) p(L=1 \mid S=1)+p(X=1 \mid L=0, T=1, S=1) p(L=0 \mid S=1) \\
=& p(X=1 \mid L=1, T=1, S=1) * 0.1+p(X=1 \mid L=0, T=1, S=1) * 0.9 \\
=& p(X=1 \mid L=1, T=1) * 0.1+p(X=1 \mid L=0, T=1) * 0.9 \\
=& 0.1+0.98 * 0.9 \\
=& 0.982
\end{aligned}
$$

5. $p(X=1)$ (marginal of leaf node; marginalize over parents and use independence to simplify).

Answer:

$$
\begin{aligned}
p(T=1) =& p(T=1, V=1)+p(T=1, V=0) \\
=& p(T=1 \mid V=1) p(V=1)+p(T=1 \mid V=0) p(V=0) \\
=& 0.05 * 0.01+0.01 * 0.99 \\
=& 0.0104
\end{aligned}
$$

$$
\begin{aligned}
& p(X=1) \\
=& \sum p(X=1, T, L) \\
=& \sum p(X=1 \mid T, L) p(T, L) \\
=& \sum p(X=1 \mid T, L) p(T) p(L) \\
=& p(X=1 \mid T=0, L=0) p(T=0) p(L=0)+p(X=1 \mid T=0, L=1) p(T=0) p(L=1) \\
& +p(X=1 \mid T=1, L=0) p(T=1) p(L=0)+p(X=1 \mid T=1, L=1) p(T=1) p(L=1) \\
=& 0.05 * 0.9896 * 0.972+0.9 * 0.9896 * 0.028+0.98 * 0.0104 * 0.972+1 * 0.0104 * 0.028 \\
=& 0.0832
\end{aligned}
$$

6. $p(T=1 \mid X=1)$ (conditional of parent given child; use Bayes rule).

Answer:

$$p(T=1 \mid X=1)$$

$$
\begin{aligned}
&= p(T=1, X=1)/p(X=1) \\
&= p(X=1 \mid T=1) * p(T=1)/p(X=1) \\
&= 0.98056 * 0.0104/0.0832 \\
&= 0.12257
\end{aligned}
$$

7. $p(T=1 \mid L=1)$ (conditional of parent given co-parent; use independence and then marginal).

   Answer:

$$
\begin{aligned}
& p(T=1 \mid L=1) \\
&= p(T=1) \\
&= 0.0104
\end{aligned}
$$

8. $p(T=1 \mid X=1, L=1)$ (conditional of parent given child and co-parent; use Bayes rule).

   Answer:

$$
\begin{aligned}
& p(X=1 \mid L=1) \\
&= p(X=1, T=1 \mid L=1) + p(X=1, T=0 \mid L=1) \\
&= [p(X=1, T=1, L=1) + p(X=1, T=0, L=1)]/p(L=1) \\
&= p(X=1 \mid T=1, L=1)p(T=1) + p(X=1 \mid T=0, L=1)p(T=0) \\
&= 1 * 0.0104 + 0.9 * (1 - 0.0104) \\
&= 0.90104
\end{aligned}
$$

$$
\begin{aligned}
& p(T=1 \mid X=1, L=1) \\
&= p(T=1, X=1, L=1)/p(X=1, L=1) \\
&= p(X=1 \mid T=1, L=1)p(T=1, L=1)/p(X=1, L=1) \\
&= p(X=1 \mid T=1, L=1)p(T=1)(L=1)/p(X=1, L=1) \\
&= p(X=1 \mid T=1, L=1)p(T=1)/p(X=1 \mid L=1) \\
&= 1 * 0.0104/0.90104 \\
&= 0.01154
\end{aligned}
$$

2.3 Inpainting

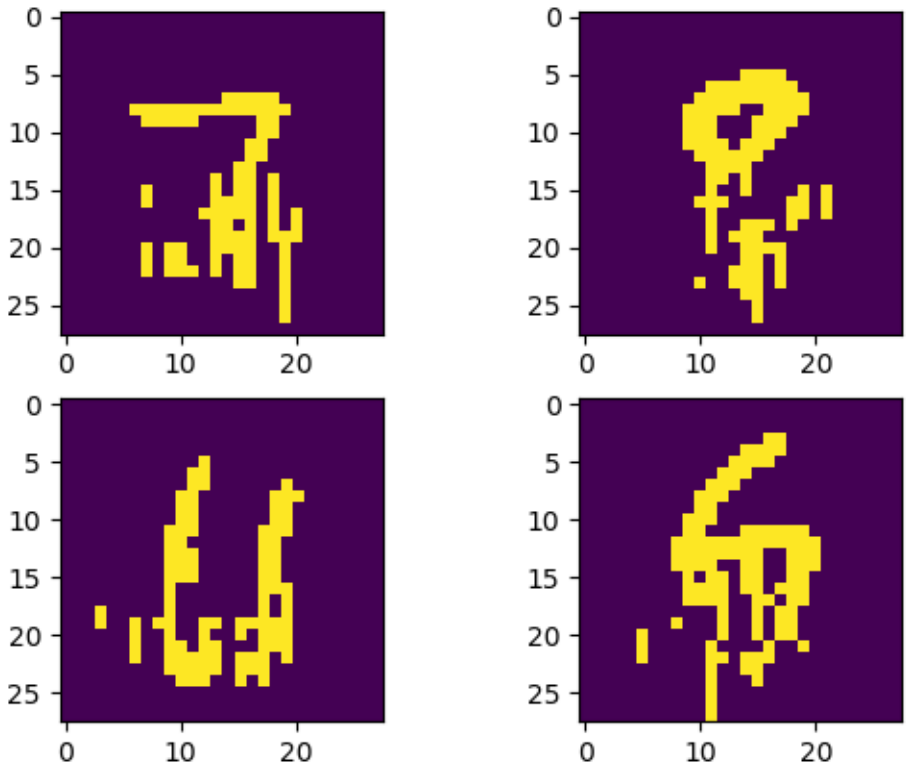
The function *example_fil.jl* loads a variant of the MNIST dataset. It contains all the training images but the test images are missing their bottom half. Running this function fits an independent Bernoulli model to the training set, and then shows the result of applying the density model to "fill in" four random test examples. It performs pretty badly because the independent model can't condition on the known top-half of the images.

1. Make a variant of the demo where you fit an inhomogeneous Markov chain to each image column. Hand in your code and an example of using this model to fill in 4 random test images.

   Answer: Code of example_inhomMarkov.jl

```
# Load X and y variable
using JLD
data = load("MNIST_images.jld")
(X,Xtest) = (data["X"],data["Xtest"])

m = size(X,1)
n = size(X,3)
t = size(Xtest,3)

# change start
# Train an inhomogeneous Markov chain
p_ijk = zeros(m,m,2)
for j in 1:m
    p_ijk[1,j,:] = sum(X[1,j,:] .== 1)/n
end

for i in 2:m
    for j in 1:m
        p_ijk[i,j,1] = sum( (X[i,j,:] .==1) .& (X[i-1,j,:]
            .==0) )/sum(X[i-1,j,:] .==0)
        p_ijk[i,j,2] = sum( (X[i,j,:] .==1) .& (X[i-1,j,:]
            .==1) )/sum(X[i-1,j,:] .==1)

    end
end
# change end

# Show parameters
using PyPlot
```

```
# Fill-in some random test images
figure(1)
for image in 1:4
    subplot(2,2,image)

    # Grab a random test example
    ind = rand(1:t)
    I = Xtest[:,:,ind]

    # Fill in the bottom half using the model
    for i in 1:m
        for j in 1:m
            if isnan(I[i,j])
                if I[i-1,j] ==0
                    I[i,j] = rand() < p_ijk[i,j,1]
                else
                    I[i,j] = rand() < p_ijk[i,j,2]
                end
            end
        end
    end
    imshow(I)
end
```



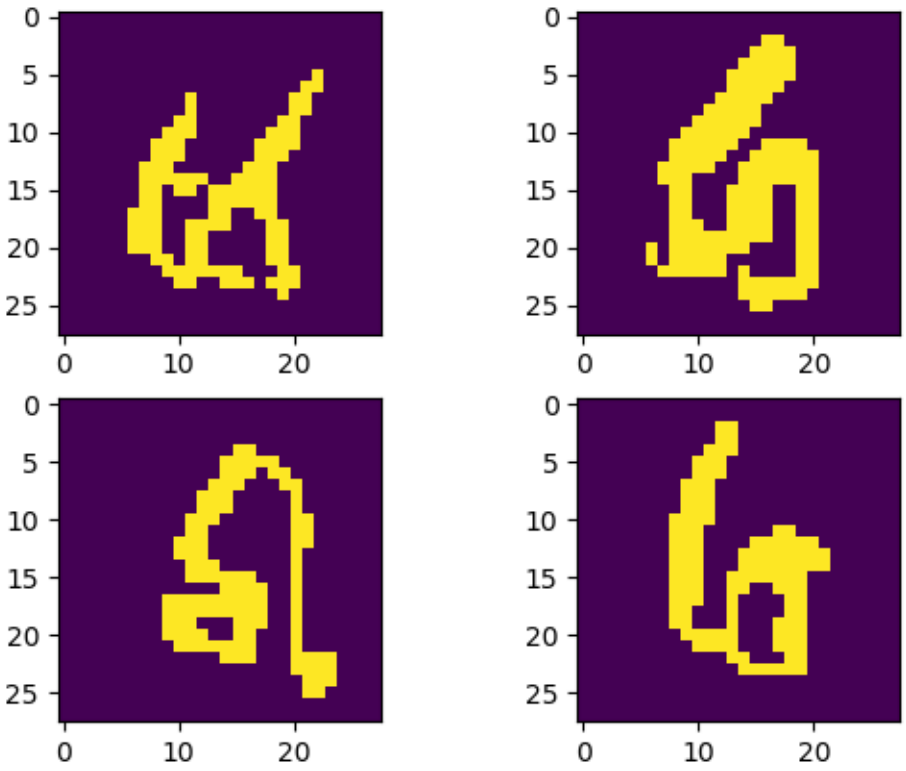
2. Make a variant of the demo where you fit a directed acyclic graphical model to the data, using general discrete conditional probabilities and where the parents of pixel (i, j) are the other 8 pixels in the region $(i-2:i, j-2:j)$.Hand in your code and an example of using this model to fill in 4 random test images.

   Answer: Code of example_DAG.jl

```
# Load X and y variable
using JLD
data = load("MNIST_images.jld")
(X, Xtest) = (data["X"], data["Xtest"])
include("problem2_functions.jl")

m = size(X,1)
n = size(X,3)
t = size(Xtest,3)

# Training
# Put all 784 models in dictionary
```

```
models = Dict()
print("***** Training phase ***** \n")
for i=1:m
    for j=1:m
        (ind,np)=parents(i,j)
        # Setup supervised learning problem
        ybar = X[i,j,:]
        Xbar = X[ind[1],ind[2],:]
        # Put in matrix form and remove last line as it is not a parent
        Xbar = reshape(Xbar,(np+1,n))'
        Xbar = Xbar[:,1:(end-1)]
        #solve it
        models[(i-1)*m+j] = tabular(Xbar,ybar)
    end
    @printf "progress: %.1f " i/m*100
    print("% \n")
end

println("Done with training")

# Prediction
using PyPlot
figure(1)
for image in 1:4
    subplot(2,2,image)
    # Grab a random test example
    index = rand(1:t)
    I = Xtest[:,:,index]
    # Fill in the bottom half using the model
    for i in 1:m
        for j in 1:m
            if isnan(I[i,j])
                (ind,np)=parents(i,j)
                Xtilde = I[ind[1],ind[2]]
                Xtilde = reshape(Xtilde,(np+1,1))'
                Xtilde = Xtilde[1:(end-1)]
                I[i,j] = models[(i-1)*m+j].sample(Xtilde)
            end
        end
    end
    imshow(I)
end
```

```
function parents(i,j)
    leftboundary  = max(j-2,1)
    upperboundary = max(i-2,1)
    np = (i-upperboundary+1)*(j-leftboundary+1) - 1
    ind = [upperboundary:i,leftboundary:j]
    return ind,np
end
```



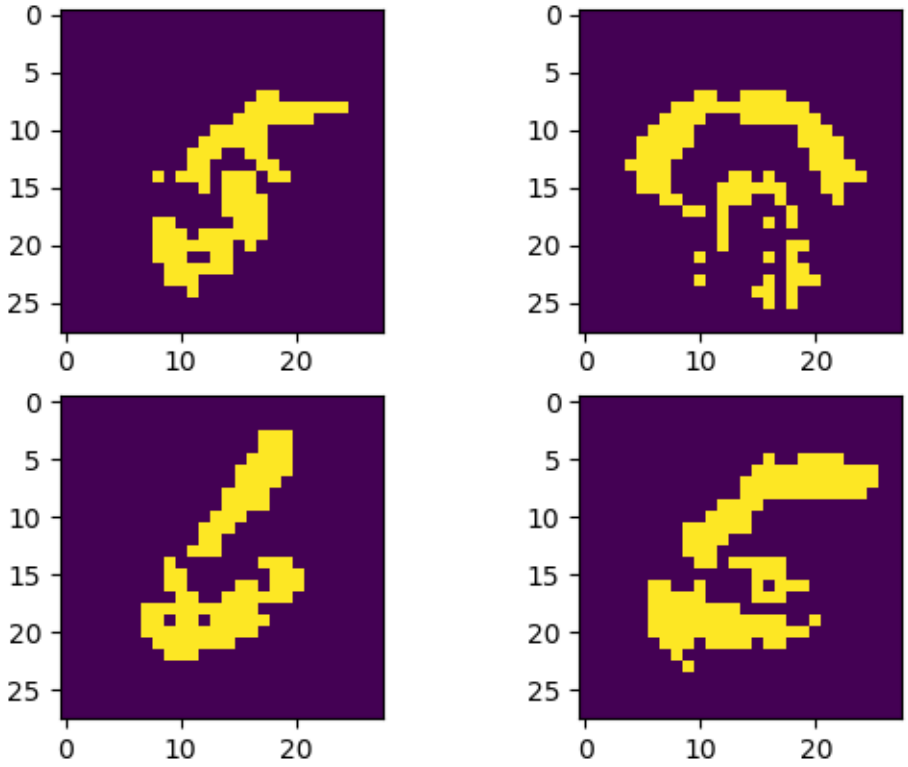
3. Make a variant of the demo where you fit a sigmoid belief network to the data, where the parents of pixel (i, j) are the other pixels in the region $(1 : i, 1 : j)$. Hand in your code and an example of using this model to fill in 4 random test images.

   Answer: Code of example_sigmoidnet.jl

```
# Load X and y variable
using JLD
data = load("MNIST_images.jld")
```

```
(X, Xtest) = (data["X"], data["Xtest"])
include("problem2_functions.jl")

m = size(X,1)
n = size(X,3)
t = size(Xtest,3)

# Training
# Create a dictionary for every model
models = Dict()
print("***** Training phase ***** \n")
for i in 1:m
    for j in 1:m
        ybar = X[i,j,:]
        Xbar = X[1:i,1:j,:]
        Xbar = reshape(Xbar,(i*j,n))'
        Xbar = Xbar[:,1:(end-1)]
        models[(i-1)*m+j] = logReg(Xbar,ybar)
    end
    @printf "progress: %.1f " i/m*100
    print("% \n")
end
println("Done with training")

# Prediction
using PyPlot
figure(1)
for image in 1:4
    subplot(2,2,image)
    # Grab a random test example
    index = rand(1:t)
    I = Xtest[:,:,index]
    # Fill in the bottom half using the model
    for i in 1:m
        for j in 1:m
            if isnan(I[i,j])
                Xtilde = I[1:i,1:j]
                Xtilde = reshape(Xtilde,(i*j,1))'
                Xtilde = Xtilde[1:(end-1)]
                I[i,j] = models[(i-1)*m+j].sample(Xtilde)
            end
        end
```

```
        end
        imshow(I)
    end
```



3 Very-Short Answer Questions

Give a short and concise 1-sentence answer to the below questions.

1. What are two reasons that we might want to be able to sample from a distribution?
   If we can sample cheaply, we can use Monte Carlo estimates to get expected quantities or visualize more what a sample may look like. Another reason is that we could use the samples to decide weather the assumptions on the distribution in a model make sense.

2. What is the inverse transform method used for?
   Inverse methods are used to sample from a CDF using the fact that a uniform random

number generator is easy to implement.

3. Describe how we could use ancestral sampling to sample from a naive Bayes model.
   Since naive Bayes almost has the same form as ancestral sampling, we compute all conditionals and then sample the from each conditional to get a (x^i, y^i) pair.

4. What is the cost of generating a sample from a Markov chain of length d with k possible states for each time? What is the cost of decoding?
   To generate samples, each component requires k $O(1)$ calculations, thus $O(k^d)$ for a chain of length d. To decode, we have the Viterbi algorithm to compute the next message, which has cost $O(k^2)$ so then we need d messages, total cost is $O(dk^2)$.

5. Suppose we have a black-box procedure for sampling from a student t distribution. Describe how we could use this black box to get an approximation of the variance of the distribution.
   Just take enough samples to approximate the first and second moments to get the variance with a Monte Carlo scheme.

6. What is the difference between inference and decoding in Markov chains?
   Inference is a probability that we can talk about. Decoding is the sequence of most likely probabilities to give the maximum likelihood.

7. What is "explaining away"?
   In a probabilistic sense, if you have two parents of the same child and given information about one of the parents, this causes the probability of the other parent to decrease.

8. If two variables are not d-separated, are they necessarily depenent? If two variables are d-separated, are they necessarily independent?
   No, consider a basic V-structure, in this case our parents may be independent but they are d-connected by the common child. No again, they are only independent if the d-separating variables are given, and example would be upside down V-structure (two children, one parent), in this case the only path between the children is through the parent and thus only information about the parent will break their dependence.

9. Why do we enforce that DAG models are acyclic?
   They need to be trees, cyclic graphs are not trees and there would be probabilities that have infinite sequences.

10. What is the relationship between multivariate Gaussians and UGMs?
    The multivariate Gaussian is just a particular choice for the potential function $\phi(x^i, x^j)$, in a UGM you can choose many others.

11. For each of the following, name a method we covered in class that applies for UGMs:

    (a) Approximate decoding.
    Iterated conditional mode

(b) Approximate sampling.
Gibbs Sampling or more generally MCMC

(c) Approximate inference.
Block structured inference

4 Relevant Papers for Project

4.1 Finding Relevant Papers

To help you make progress on your project, for this part you should hand in a list of 10 academic papers related to your current project topic. Finding related work is often one of the first steps towards getting a new project started, and it gives you an idea of what has (and has not) been explored. Some strategies for finding related papers are:

1. Use Google: try the keywords you think are most relevant. Asking people in your lab (or related labs) for references is also often a good starting point.
2. Once you have found a few related papers, read their introduction section to find references that these papers think are worth mentioning.
3. Once you have found a few related papers, use Google Scholar to look through the list of references that are *citing* these papers. You may have to do some sifting if there are a lot of citations. Reasonable criteria to sift through large reference lists include looking for the ones with the most citations or focusing on the more recent ones (then returning to Step 2 to find the more-relevant older references).

For this question you only need to provide a list, but in Assignment 5 you will have to do a survey of 10 papers. So it's worth trying to identify papers that are both relevant and important at this point. For some types of projects it will be easier to find papers than others. If you are having trouble, post on Piazza.

Although the papers do not need to all be machine learning papers, the course project does need to be related to machine learning in some way, so at least a subset of the papers should be machine learning papers. Here is a rough guide to some of the most reputable places to where you see machine learning works published:

- The International Conference on Machine Learning (ICML) and the conference on Advances in Neural Information Processing (NIPS) are the top places to publish machine learning work. The Journal of Machine Learning Research (JMLR) is the top journal, although in this field conference publications are usually viewed as more prestigious.
- Other good venues include AISTATS (emphasis on statistics), UAI (emphasis on graphical models), COLT (emphasis on theory), ICLR (emphasis on deep learning), ECML-PKDD (European version of ICML), CVPR and ICCV/ECCV (emphasis on computer

vision), ACL and EMNLP (emphasis on language), KDD (emphasis on data mining), AAAI/IJCAI (emphasis on AI more broadly), JRSSB and Annals of Stats (emphasis on statistics more broadly), and Science and Nature (emphasis on science more broadly).

4.2 Paper Review

Among your list of 10 papers, choose one paper and write a review of this paper. It makes sense to choose a paper that is closely-related to your project or to choose one of the most important-looking papers. The review should have two parts:

1. A short summary of the contributions of the paper. Say what problem the paper is addressing, why this is an important problems, what is proposed, and how it is being evaluated.
2. A list of strengths and weaknesses of the paper, and whether the claims are appropriately evaluated. For ideas of what issues to discuss, see the JMLR guidelines for reviewers:
   `http://www.jmlr.org/reviewer-guide.html`

Note that you should include a non-empty list of strengths *and* weaknesses. Many students when doing their first reviews focus either purely on strengths or purely on weaknesses. It's important to recognize that all papers have weaknesses or limitations (even ones written by famous people or that are published in impressive places or that proved to be historically important) and all papers have strengths or at least a motivation (the authors must have thought it was worth writing for some reason).

Literature Review - Principal Component Analysis in Predictive Policing

by Cody Griffith, Ziming Yin and Tim Jaschek

March 17, 2018

Abstract

In our final project we would like to apply unsupervised machine learning techniques for predictive policing. We decided to focus on unsupervised clustering. A powerful and frequently used tool for such tasks is principal component analysis.

Summaries

Article [2]: K-means Clustering via Principal Component Analysis

The paper *K-means Clustering via Principal Component Analysis* written by Chris Ding (Berkeley) and Xiaofeng He (Berkeley) appeared in the *Proceedings of the* 21^{st} *International Conference on Machine Learning*. It contributes to the research in machine learning by showing that there is a fundamental connection between principal component analysis and K-means clustering. It begins with a brief introduction to the K-means algorithm and the principal component analysis (PCA) algorithm. The introduction ends with the claim that PCA can be used as a tool to obtain the result of a K-means performance. In the following the authors give a discussion of a two-class model, where the mathematics are fairly straight forward and the result can be visualized. In the third section the main result for a general K-class model is stated and followed by a rigorous proof. A final section that shows results of numerical experiments completes the paper.

Before we state the main result of Ding and He, let us recap their notation. As in our lecture let $X \in \mathbb{R}^{n\times d}$ denote the matrix of test data, *i.e.* the collection of the values of d features for n examples. The K-means method assigns the data to K disjoint clusters $C_1, \dots, C_K$. The clusters are determined by their centroids, which are determined by minimizing the ℓ^2-error

$$J_K = \sum_{k=1}^{K} \sum_{i \in C_k} (x_i - m_k)^2,$$

where $(x_1, \dots, x_d) = X$ represent the feature vectors and $m_k = \sum_{i\in C_k} x_i/n_k$ is the centroid of cluster C_k and n_k is the number of points in C_k.
For the PCA, let $Y = (y_1, \dots, y_d)$ represent the centered data matrix, *i.e.* the matrix defined by $y_i = x_i - \overline{x}$, where $\overline{x} = \sum_{i=1}^{d} x_i/d$. Using Y, the covariance matrix has the nice

form

$$(\mathrm{Cov}(x_j, x_k))_{j,k} \propto \sum_{i=1}^{d} (x_i - \overline{x})(x_i - \overline{x})^T = YY^T,$$

where the constant is given by $1/d$. Principal components $(u_k)_{k=1,\ldots,d}$ and principal directions $(v_k)_{k=1,\ldots,d}$ are eigenvectors of the covariance and it's transpose respectively

$$YY^T u_k = \lambda_k u_k, \quad Y^T Y v_k = \lambda_k v_k, \qquad \text{where } \lambda_k \in \mathbb{R}^+ \text{ for } k = 1, \ldots, d.$$

By the single value decomposition Y is given by $\sum_{k=1}^{d} \lambda_k u_k v_k^T$ and can be approximated by considering partial sums of this expansion. The main result is that principle component analysis can be viewed as a tool to perform K-means clustering. We quote their main theorem:

Theorem 1. *When optimizing the K-means objective function, the continuous solutions for the transformed discrete cluster membership indicator vectors Q_{K-1} are the $K-1$ principal components: $Q_{K-1} = (v_1, \ldots, v_{K-1})$. J_K satisfies the upper and lower bounds*

$$d\overline{y^2} - \sum_{k=1}^{K-1} \lambda_k < J_K < d\overline{y^2},$$

where $d\overline{y^2}$ is the total variance and λ_k are the principal component eigenvalues of the covariance matrix YY^T.

In two dimensions, Ding and He provide a figure which shows how this clustering is happening:

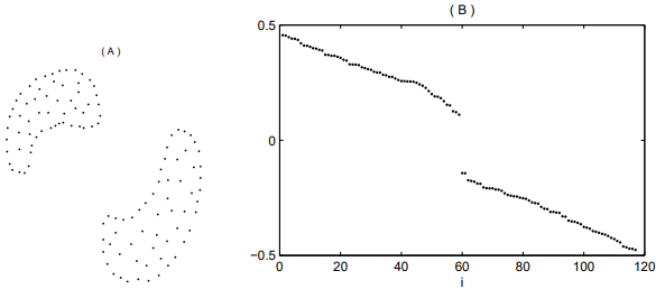


Figure 1. (A) Two clusters in 2D space. (B) Principal component $\mathbf{v}_1(i)$, showing the value of each element i.

They show that we can assign the K-means cluster by

$$C_1 = \{i | v_1(i) \leqslant 0\}, \quad C_2 = \{i | v_1(i) > 0\}.$$

The result for a more general K-class model is similar but we will not recall it here.

Strengths

- The paper follows a clear structure. We are not confronted right away with the rather complicated result for a general K-class model and even better, the simple 2-class model lets us understand the main ideas and makes us really excited to see how it works in the general case.
- The paper is very focused on the specific theorem that the authors prove. There is no unnecessary information. The introduction is brief and K-means and PCA are defined sufficiently.
- For us, who are not-yet experts in PCA and K-means but have a mathematical and ML foundation, the paper is written in a way that we can understand (most of) it. That is a desirable property of academic work.
- Is it written in a clear, readable style, with good grammar and few (if any) typographical errors.

Weaknesses

- The paper presents rigorous theory and illustrates it with examples. However, in the discussion the authors do not mention further questions and open problems. It would be helpful for the scientific community to get ideas what problems it would be interesting to look at.
- The authors do not mention that performing a PCA can take a long time as the single value decomposition of the covariance matrix has to be computed. It would have been a more complete picture if there was a comparison of the runtime of K-means and PCA on certain examples.
- There are no sections of pseudo code. That would be a useful add on.

Citations

The citations are organized in the following way: [4], [2], [11], [1], [3] and [8], [5] are PCA, [7], [12], [6] (THESIS) and [9] (THESIS) are Predictive Policing. [10] is a very detailed PhD thesis on the Karhunen-Loéve expansions, which are the continuous time generalization of PCA. Our group member Tim Jaschek wrote his B.Sc. thesis on Karhunen-Loéve expansions.

References

[1] R. Bro and A. K. Smilde. Principal component analysis. *Analytical Methods*, 6(9):2812–2831, 2014.

[2] C. Ding and X. He. K-means Clustering via Principal Component Analysis. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, pages 29–, New York, NY, USA, 2004. ACM.

[3] T. Howley, M. G. Madden, M.-L. O'Connell, and A. G. Ryder. The Effect of Principal Component Analysis on Machine Learning Accuracy with High Dimensional Spectral Data. In *Applications and Innovations in Intelligent Systems XIII*, pages 209–222. Springer, London, 2006.

[4] K. I. Kim, K. Jung, and H. J. Kim. Face recognition using kernel principal component analysis. *IEEE Signal Processing Letters*, 9(2):40–42, Feb. 2002.

[5] A. Levey and M. Lindenbaum. Sequential Karhunen-Loeve basis extraction and its application to images. *IEEE Transactions on Image Processing*, 9(8):1371–1374, Aug. 2000.

[6] Á. M. Márquez. A machine learning approach for studying linked residential burglaries, 2014.

[7] L. McClendon and N. Meghanathan. Using machine learning algorithms to analyze crime data. *Machine Learning and Applications: An International Journal (MLAIJ)*, 2(1), 2015.

[8] F. Nie, J. Yuan, and H. Huang. Optimal Mean Robust Principal Component Analysis. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, pages II–1062–II–1070, Beijing, China, 2014. JMLR.org.

[9] M. Vaquero Barnadas. Machine learning applied to crime prediction. B.S. thesis, Universitat Politècnica de Catalunya, 2016.

[10] L. Wang. *Karhunen-Loeve expansions and their applications*. London School of Economics and Political Science (United Kingdom), 2008.

[11] Q. Wang. Kernel Principal Component Analysis and its Applications in Face Recognition and Active Shape Models. *arXiv:1207.3538 [cs]*, July 2012. arXiv: 1207.3538.

[12] T. Wang, C. Rudin, D. Wagner, and R. Sevieri. Learning to detect patterns of crime. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 515–530. Springer, 2013.