CS 228 Problem Set 1

Hugh Zhang

February 19, 2017

Problem 1

Let the structure be a two node binary Bayes net with one edge from A to B. Let $P(A = 1) = 0.51, P(B = 1 \mid A = 1) = .5, P(B = 1 \mid A = 0) = .99$

Then, $P(B = 1) = 0.51 * .5 + 0.49 * .99 = .7401$, so the most likely outcome based on marginals without looking at conditionals is $A = 1, B = 1$. However, the $P(A = 1, B = 1) = .51 * .5 = .255$. This is not as likely as the most likely outcome, A=0,B=1 which has $P(A = 0, B = 1) = .49 * .99 = .4851$.

Problem 2

Basic message passing is nd^2 for a chain. Labeling the nodes from A to Y WLOG, you can factor

$$P(A \dots Y) = \phi(A, B) * \phi(B, C) \dots \phi(X, Y)$$

Z, which is the normalizing constant is just

$$\sum_A \cdots \sum_Y P(A \dots Y) = \sum_A \sum_B \phi(A, B) * \sum_C \phi(B, C) \cdots \sum_Y \phi(X, Y)$$

Z takes nd^2 time to calculate because this is just message passing across the entire array. Note that when we try calculating the marginals for a pair of variables, the variables cut out three parts of the chain. WLOG call them x_i and x_j with $i < j$. For all x_k with $k < i$ or x_m with $m > j$, then you can do basic message propogation to calculate each chain for nd^2 using basic message passing. Then, to get the middle part [bottom of the equation listed below], you can choose to extend the left side (WLOG) and calculate the middle chain for each fixed value of x_i, and then unite it with the right chain, then normalize to have a complete probability. Thus, since you do message passing d times when you try to marginalize the variable out (once for each value of the variable), it takes nd^3 The left and right chains eventually cancel out, but the middle calculation still dominates the run time. To calculate all n^2 pairs, the runtime is then n^3d^3

Let r(x) be the right chain and l(x) be the left chain. They then cancel out.

Then

$$P(x) = \frac{l(x)\phi(X_i, x_{i+1}) \dots \phi(x_{j-1}, X_j) * r(x)}{l(x) \sum_{x_i} \phi(x_i, x_{i+1}) \cdots \sum_{x_j} \phi(x_j - 1, x_j) * r(x)}$$

$$= \frac{\phi(X_i, x_{i+1}) \dots \phi(x_{j-1}, X_j)}{\sum_{x_i} \phi(x_i, x_{i+1}) \cdots \sum_{x_j} \phi(x_j - 1, x_j)}$$

CHECK THIS DO THEY CANCEL

2.2

Note that X_i and X_j are independent conditioned on X_{j-1}

$$\sum_{x_{j-1}} P(X_i, X_{j-1}) P(X_j \mid X_{j-1}) = \sum_{x_{j-1}} P(X_i, X_{j-1}, X_j) = P(X_i, X_j)$$

2.3

I claim you can do this in $O(n^2 d^3)$ time. First you calculate Z, which takes $O(nd^2)$, which is fine. Then we calculate all the individual marginals $P(x_i)$ for all i. This is also just basic message passing, so its $O(nd^2)$ Then, you calculate all adjacent pairs using our above algorithm. $P(x_1, x_2) \dots P(x_{n-1}, x_n)$ This takes nd^3 time per pair, and N pairs, so we are at $O(n^2 d^3)$. Then, to calculate all pairs of distance x apart given that you've calcualted all pairs of distance x-1 apart and below, we use the formula above. (This is the inductive step)

$$\begin{aligned} & P(X_i, X_j) \\ = & \sum_{x_{j-1}} P(X_i, X_{j-1}) P(X_j \mid X_{j-1}) \\ = & \sum_{x_{j-1}} P(X_i, X_{j-1}) \frac{P(X_j, X_{j-1})}{P(X_{j-1})} \end{aligned}$$

We have all of these values, since $j - i - 1 = x - 1 < x$, and we can use our previously calculated values. There are d^2 possible values of X_i and X_j, and N possible pairs of distance X. Marginalizing over all values of X_{j-1} adds another factor of d to our calculations for a total of nd^3 per inductive step. Since we have n inductive steps, this is a total of $n^2 d^3$, exactly what we want.

Problem 3

If you change a factor (by changing its value, or by adding an edge) in a clique, check all its neighbors and see if they are affected, AKA the seperation set contains the factor that has been changed. If they are affected, then everything in that entire subtree needs to be updated, since they all depend on that factor in the message passing.

3.2

If you only want the marginal over a single variable, you only need to pass the message along the single path to the variable you want to calculate, up to d-separation. In addition, you don't need to update the entire tree, just for every node that you are d-seperated on since you marginalize the probability out otherwise.

EQUATION FOR MESSAGE DOESN"T DEPEND ON THE OLD GUY. Calibrated only if all the messages that are coming in are ok.

Problem 4

4.2.1

Inference is exponential. No

Yes, forward prop

Problem 5

a

CODE HERE XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

G.varToFac = [[i] for i in range(M)]

for i in range(N): for var in np.where(H[i]==1)[0]: G.varToFac[var].append(i+M)

G.factor = [] G.M = M G.N = N G.p = p

for i in range(M):

if yhat[i] == 1: vals = [p, 1.-p] if yhat[i] == 0: vals = [1.-p, p] G.factor.append(Factor(None, [i], [2], np.array(vals))) for i in range(N): scope = np.where(H[i]==1)[0]

val = np.zeros([2]*scope.size) for index, value in np.ndenumerate(val): s = 0 for i in index: s += i if sval[index] = 1.

G.factor.append(Factor(None, scope, [2]*scope.size, (val)))

return G

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

Test cases: ytest1 = [1, 1, 1, 1, 1] ytest2 = [1, 0, 1, 0, 1] ytest3 = [0, 0, 0, 0, 0]

If you include the error rate factors: 0, 0, 0.77378094 If you don't: 0, 0, 1

c

All values essentially 0, as expected. Thus, the code word is properly decoded.

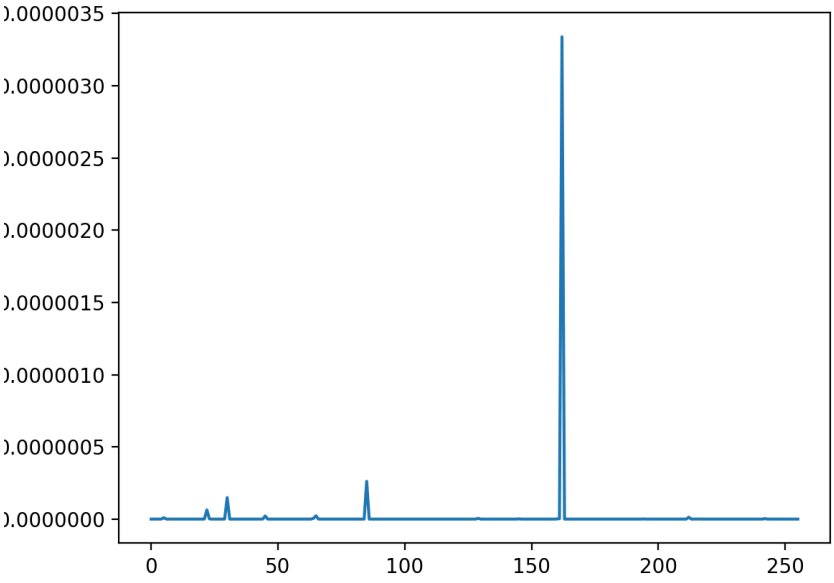

Figure 1: Plot with the ith bit on the X axis and the probability of it being 1 on the y axis.

d

It's moderately reliable. For a 0.06 error rate, all but 1 message converged to the correct message after only 50 iterations.

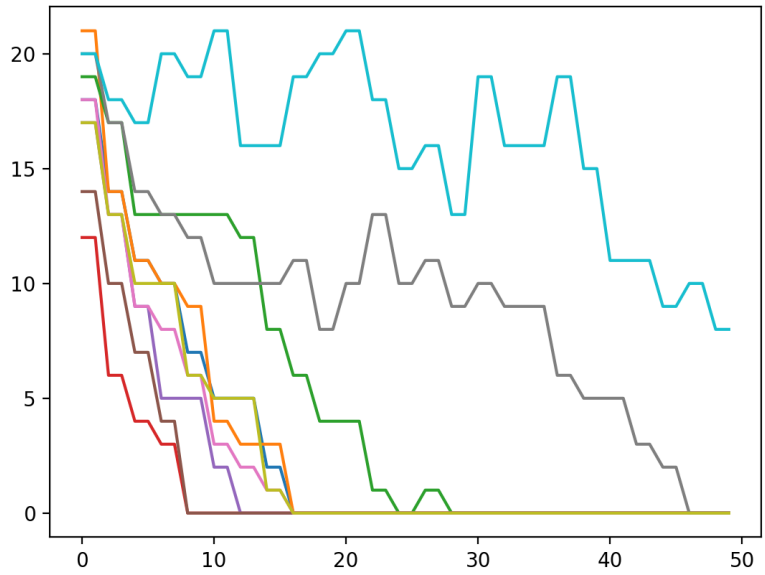

Figure 2: Error rate of 0.06. X axis is the number of iterations. Y axis is the hamming distance between decoded message and real message

e

As the error rate increases, it gets harder and harder for loopy BP to correctly converge onto the correct message. Only 5/10 messages converged to a 0 hamming distance for error rate of 0.08. For the error rate of .1, many didn't even come close and didn't show much sign of going down.

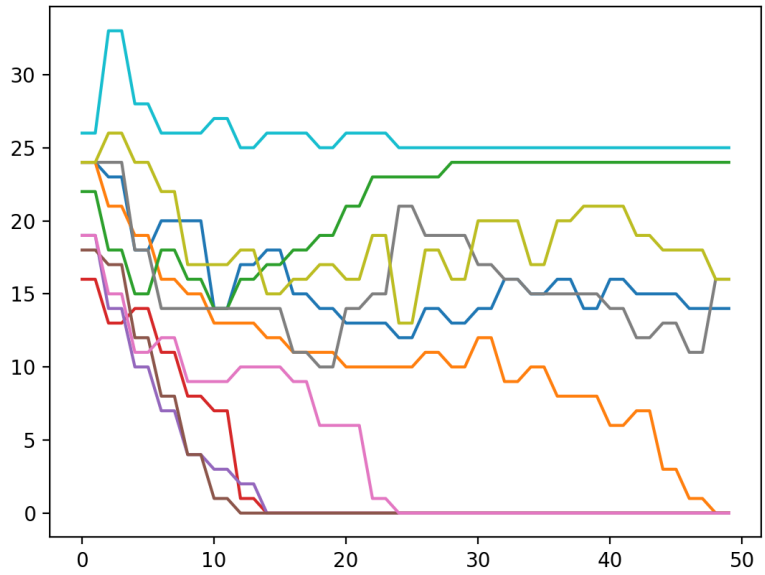

Figure 3: Error rate of 0.08. X axis is the number of iterations. Y axis is the hamming distance between decoded message and real message

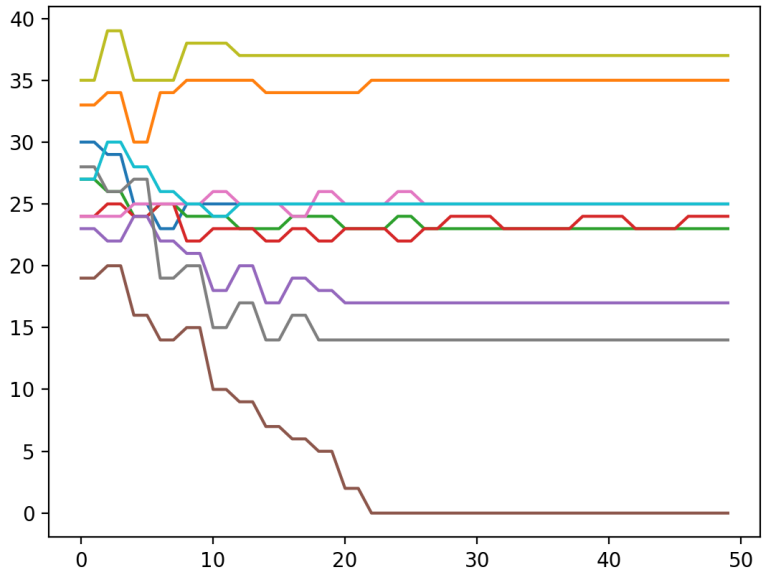

Figure 4: Error rate of 0.1. X axis is the number of iterations. Y axis is the hamming distance between decoded message and real message

f

Correctly decodes picture after X iterations. At iteration 0, the picture is non existent since all the incoming messages to variables are initialized uniformly.

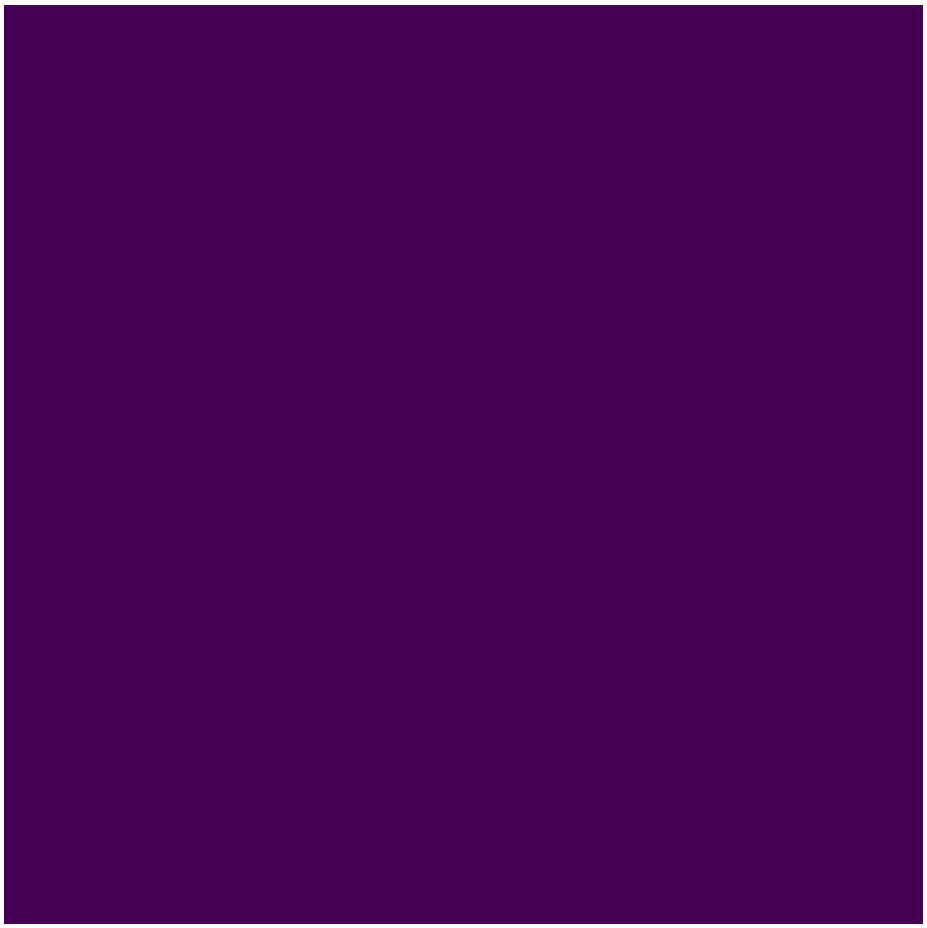

Figure 5: With 0.06 error rate and 0 iteration

Figure 6: With 0.06 error rate and 1 iteration

Figure 7: With 0.06 error rate and 2 iteration

Figure 8: With 0.06 error rate and 3 iteration

Figure 9: With 0.06 error rate and 5 iteration

Figure 10: With 0.06 error rate and 10 iteration

Figure 11: With 0.06 error rate and 20 iteration

Figure 12: With 0.06 error rate and 30 iteration

g

Correctly decodes picture after X iterations. At iteration 0, the picture is non existent since all the incoming messages to variables are initialized uniformly. Naturally, since there is a higher error rate, the picture looks more blurry at first and takes a longer time to come to a good picture, but it still does eventually.

With a higher error rate, naturally the or

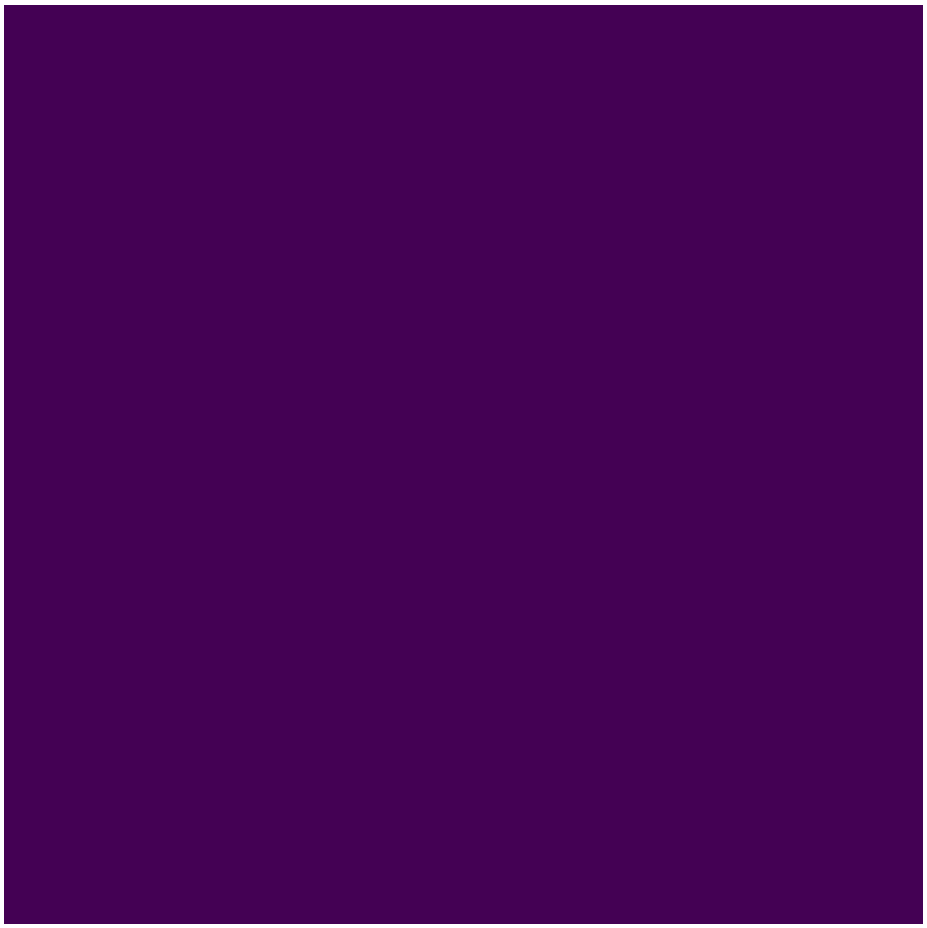

Figure 13: With 0.1 error rate and 0 iteration

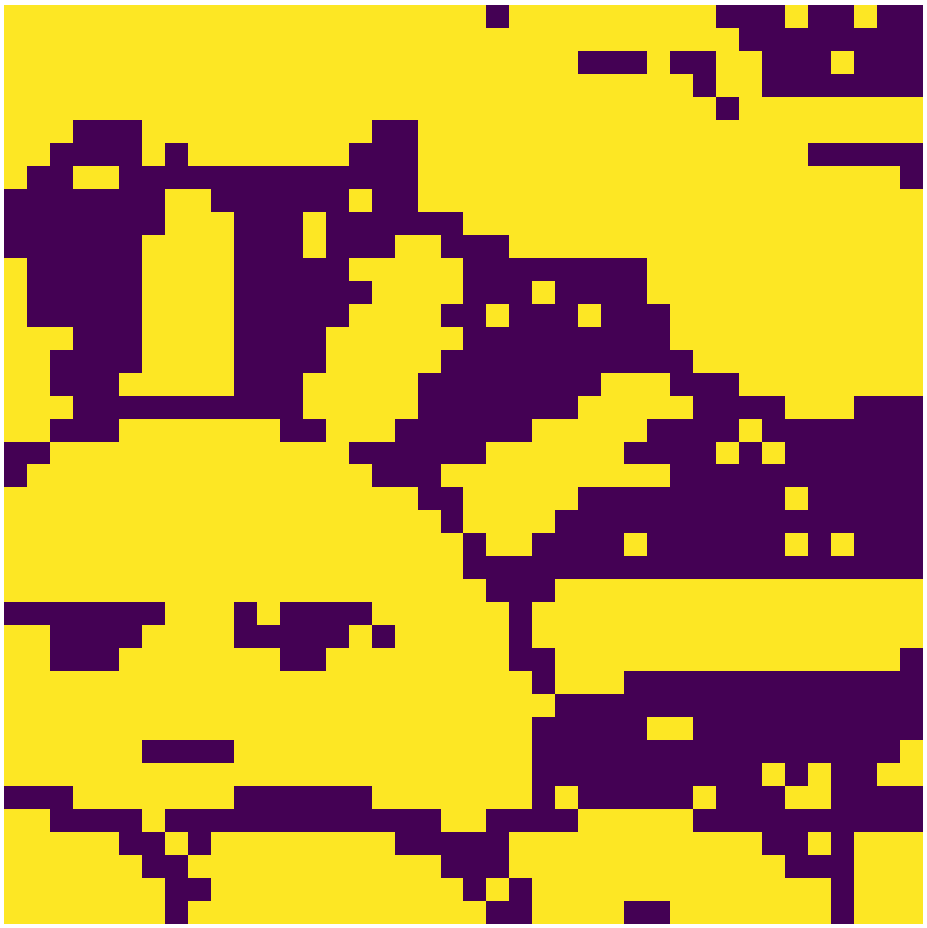

Figure 14: With 0.1 error rate and 1 iteration

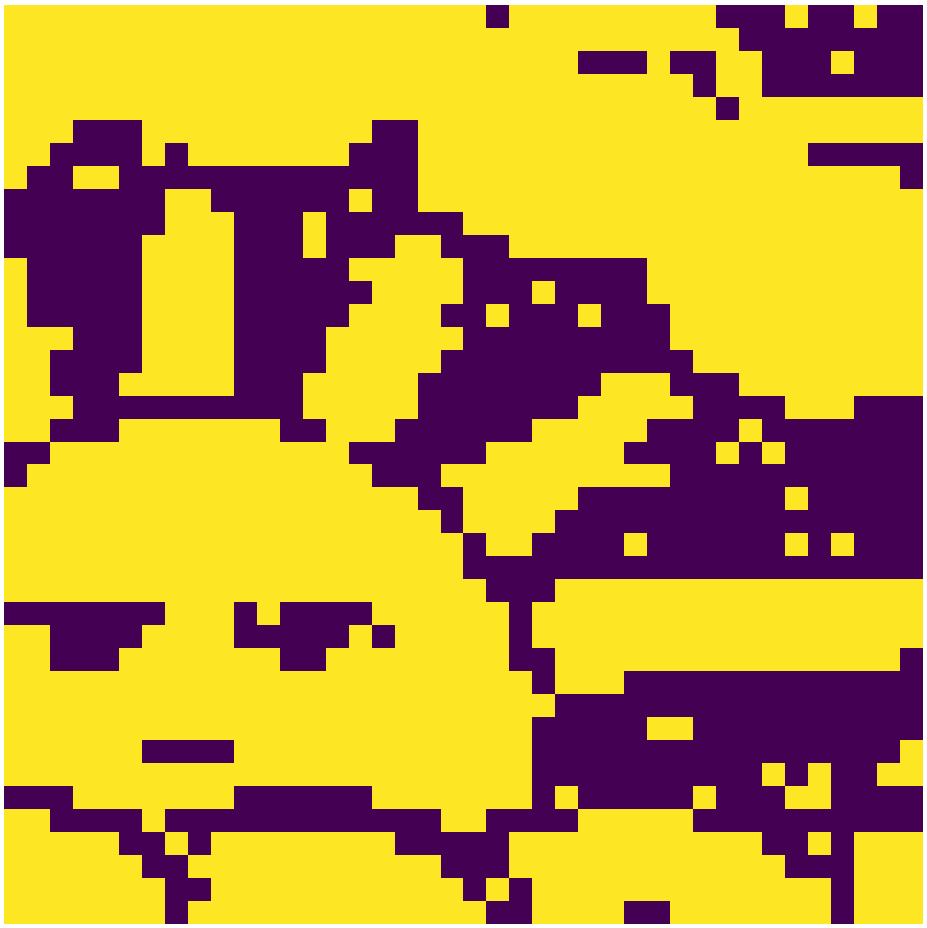

Figure 15: With 0.1 error rate and 2 iteration

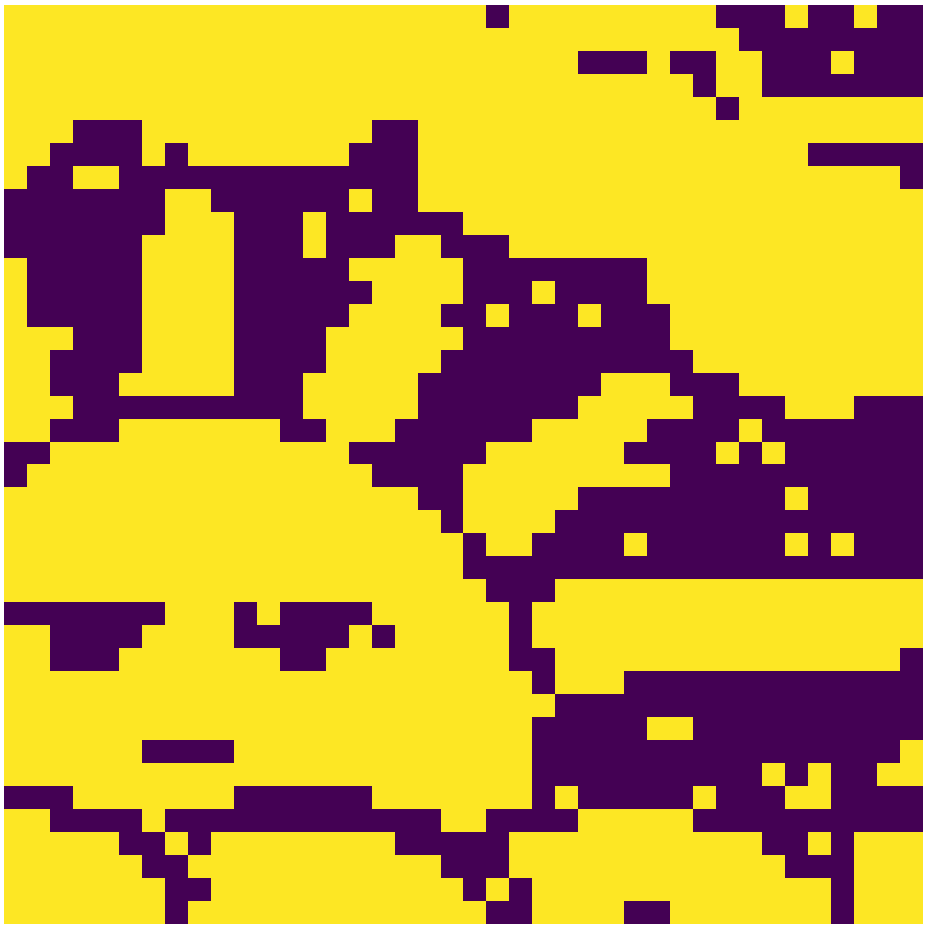

Figure 16: With 0.1 error rate and 3 iteration

Figure 17: With 0.1 error rate and 5 iteration

Figure 18: With 0.1 error rate and 10 iteration

Figure 19: With 0.1 error rate and 20 iteration

Figure 20: With 0.1 error rate and 30 iteration