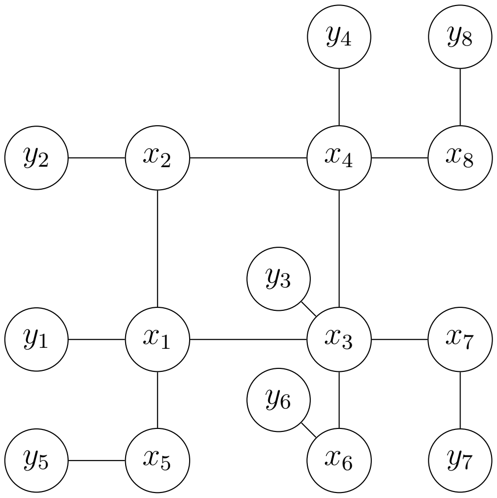
CSE 515 HW

Alexander Van Roijen

February 24, 2020

I 4.6

I.I a



This represents one corner of our graph representing our image, so it would expand out much further, but this should suffice.

I.II b

```
For the background, we have
mu = array([0.32488191, 0.3359058 , 0.15492575])

cov = [array([[0.07608524, 0.0709183 , 0.04026234],
[0.0709183 , 0.06727482, 0.03855529],
[0.04026234, 0.03855529, 0.02645081]]),
For the forground we have
mu = array([0.70059092, 0.45043831, 0.04376236])

cov = array([[0.05134831, 0.05985912, 0.00137747],
[0.05985912, 0.08341228, 0.00235105],
[0.00137747, 0.00235105, 0.00110494]])
```

I.III c

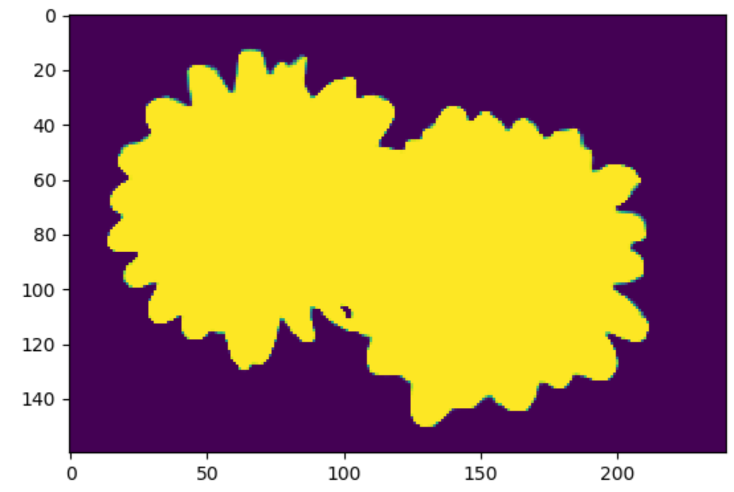


Figure 1: after 30 iterations of BP, $\epsilon = 0.01$

I.IV c

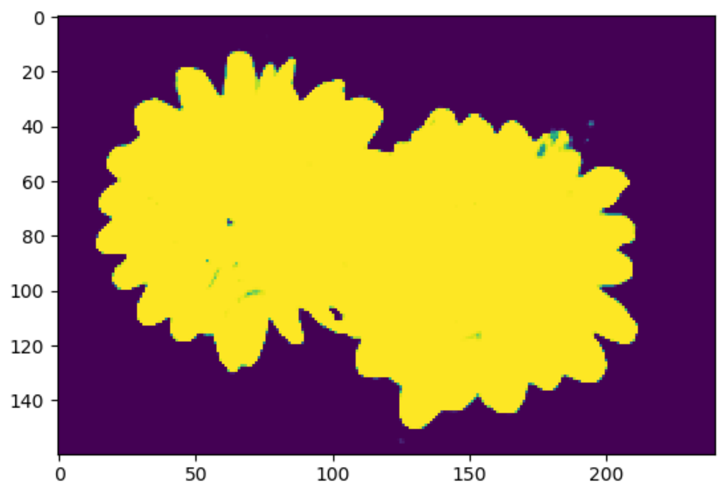


Figure 2: after 1 BP iteration, $\epsilon = 0.01$

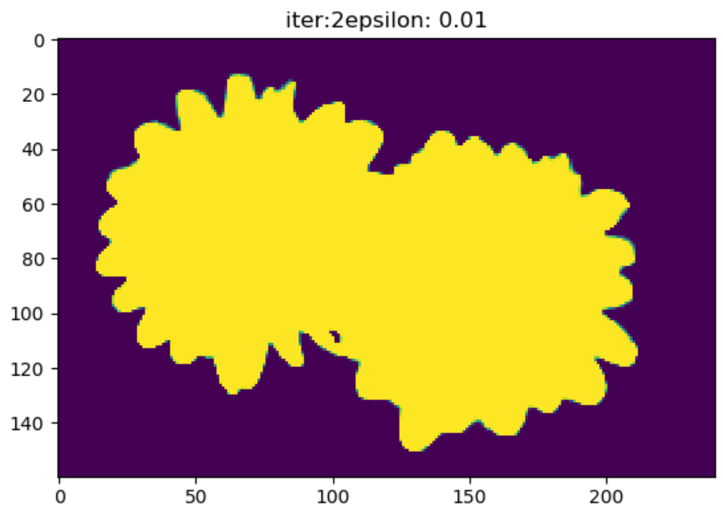


Figure 3: after 2 BP iterations, $\epsilon = 0.01$

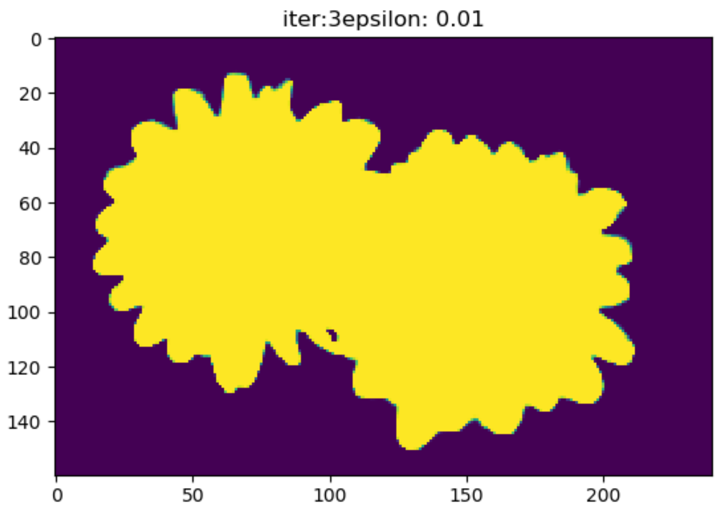


Figure 4: after 3 BP iterations, $\epsilon = 0.01$

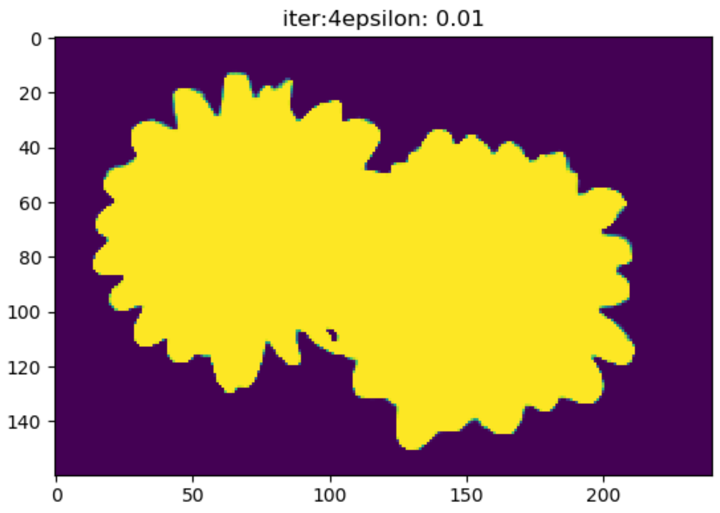
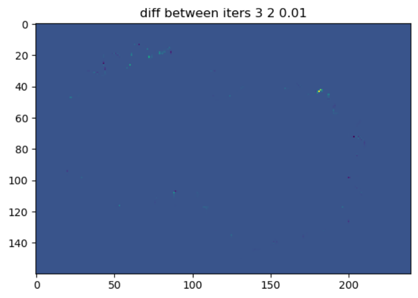
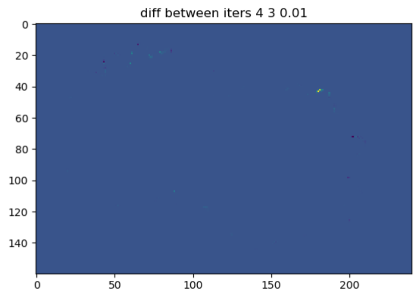


Figure 5: after 4 BP iterations, $\epsilon = 0.01$

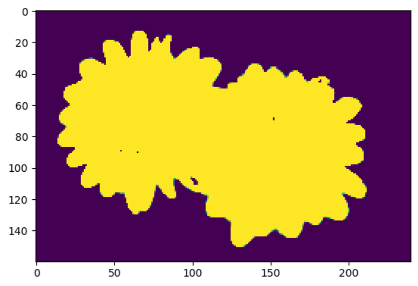
As to be expected in a background foreground classification task, the greatest regions of uncertainty are around the edges of our flowers petals, as this is where nearby pixels are meant to be quite different from each other.
Overall, we see a great deal of change in labels and the confidence in them during the first

and second iteration, but harder to tell in future iterations. Looking at figures below, we can see the edges are still being determined slightly during the next iterations between 2 and 3 and 3 and 4.

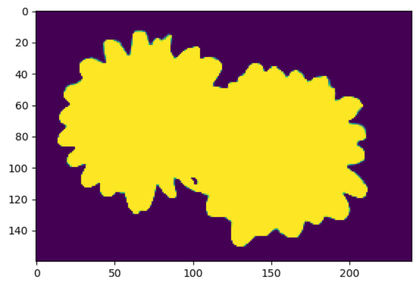




Looking at these, we can see small regions of change around the upper right and left edges of our flowers.

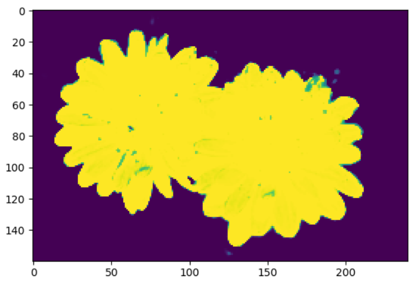


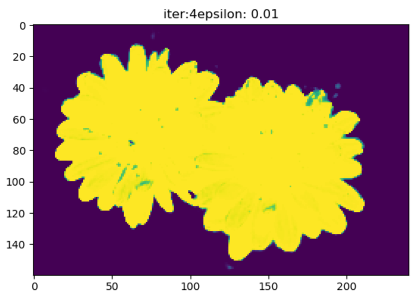
(a) 30 iterations with $\epsilon = 0$



(b) 30 iterations with $\epsilon = 0.01$

The results are slightly similar, with a slight change in their edge cases, as to be expected. Overall, edges seem to be less sharply defined, a bit more rounded with a greater deal of uncertainty between background and foreground within different regions of hte flow


(a) 30 iterations with .6 and .4



(b) 4 iterations with .6 and .4

II 5.1

II.I a

$v^{t+1}_{i\to a}(x_i) = P(y_i|x_i)\Pi_{b\in\partial i - a}\tilde{v}^t_{b\to i}(x_i)$
$\tilde{v}^{t+1}_{a\to i}(x_i) = \Sigma_{x_{\partial a - i}}\Pi_{j\in\partial a - i} v^t_{j\to a}(x_j) I(\bigotimes x_{\partial a} = 1)$

II.II b

For $v^{t+1}_{i\to a}(x_i)$ it will take $O(d_i)$ where d_i is the number of factors that neighbor or connect to i which is in general $O(l)$.

For $\tilde{v}^{t+1}_{a\to i}(x_i)$ our time complexity is $O((|\partial a|-1)2^{|\partial a|-1}) = O(r2^r)$. This is because we need to assure that our logic holds for every possible permutation of our nodes $I(\bigotimes x_{\partial a} = 1)$. The additional r comes from a need to multiply r different messages together for each permutation.

To reduce the complexity, we can expand our equation $\tilde{v}^{t+1}_{a\to i}(x_i)$ to get a general form of $v^t_{j\to a}(+1) * v^t_{h\to a}(+1) ... * v^t_{z\to a}(+1) I(\bigotimes x_{\partial a} = 1)$
$+ v^t_{j\to a}(-1) * v^t_{h\to a}(+1) ... * v^t_{z\to a}(+1) I(\bigotimes x_{\partial a} = 1)$
$+ v^t_{j\to a}(+1) * v^t_{h\to a}(-1) ... * v^t_{z\to a}(+1) I(\bigotimes x_{\partial a} = 1)$
$+ ... v^t_{j\to a}(-1) * v^t_{h\to a}(-1) ... * v^t_{z\to a}(-1) I(\bigotimes x_{\partial a} = 1)$. Which then becomes a function of x_i such that

$\tilde{v}^{t+1}_{a\to i}(x_i = +1) = v^t_{j\to a}(+1) * v^t_{h\to a}(+1) ... * v^t_{z\to a}(+1) * 1$
$+ v^t_{j\to a}(-1) * v^t_{h\to a}(+1) ... * v^t_{z\to a}(+1) * 0$
$+ v^t_{j\to a}(+1) * v^t_{h\to a}(-1) ... * v^t_{z\to a}(+1) * 0$
$+ ... v^t_{j\to a}(-1) * v^t_{h\to a}(-1) ... * v^t_{z\to a}(-1) * 1$ (Note this last condition may be 0 or one depending on the carndinality of r).

Similarly, the indicators would flip in $\tilde{v}^{t+1}_{a\to i}(x_i = -1)$. Thus, since both values end up with opposite signs, you would get the same results if you let each one cancel instead, meaning you add its negative counterpart $a + b + -b = a + 0 * b = a$. With this in mind, you can rewrite our equation into a product of

$\tilde{v}^{t+1}_{a\to i}(x_i = +1) \propto (v^t_{j\to a}(+1) + v^t_{j\to a}(-1))$
$* (v^t_{h\to a}(+1) + v^t_{h\to a}(-1))...$
$* (v^t_{z\to a}(+1) + v^t_{z\to a}(-1))$
$+$
$(v^t_{j\to a}(+1) - v^t_{j\to a}(-1))$
$* (v^t_{h\to a}(+1) - v^t_{h\to a}(-1))...$
$* (v^t_{z\to a}(+1) - v^t_{z\to a}(-1))$

Taking the time to write this out, you can see you get some nice cancellations such as

$v^t_{j\to a}(-1)v^t_{h\to a}(+1)...v^t_{z\to a}(+1) - v^t_{j\to a}(-1)v^t_{h\to a}(+1)...v^t_{z\to a}(+1) = 0$ which satisfies exactly $v^t_{j\to a}(-1) * v^t_{h\to a}(+1)... * v^t_{z\to a}(+1) * 0$.

Thus, we get $\tilde{v}^{t+1}_{a\to i}(x_i = +1) \propto \Pi_{j\in\partial a-i} v^t_{j\to a}(+1) + v^t_{j\to a}(-1) + \Pi_{j\in\partial a-i} v^t_{j\to a}(+1) + v^t_{j\to a}(-1)$ This satisfies our reduction in computation time to $O(d_a)$ as we only need to iterate over the neighbors values once

II.III c

Initially, we have that $w^0 = P(W^0 = -1) = P(\text{randomly chosen message from variable to factor is flipped}) = \epsilon$ as they are independent and we assume that $\forall x_i$ a message of 1 was sent.

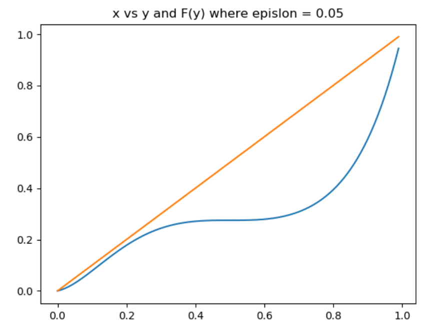
For subsequent time steps, with our messages as described, we get the probability of a message from a factor to a node taking value -1 only occurs when that factor received an odd number of -1 from the r variable nodes it receives from.

So we can think of $z^t = P(Z^t = -1)$ as a binomial distribution with the parameters r,$p = (w^t)$. And we want the sum of all instance where the number of successes is odd. $z^t = P(Z^t = -1) = \Sigma_{i=1,3,..r} p^i(1-p)^{n-i}$. However, this can be reduced, as the number of odd values for a Binomial distribution r,p gives$\frac{1-(1-2p)^r}{2}$

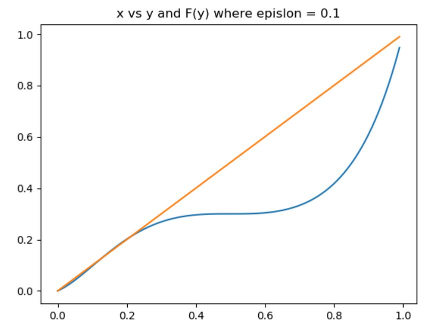
Similarly, $w^{t+1} = P(W^{t+1} = -1) = (z^t)^{l-1} + \epsilon(1 - (z^t)^{l-1} - (1 - z^t)^{l-1})$ as all incoming messages from our factor must be -1 in order for it to be -1.

II.IV d

$w^{t+1} = (\frac{1-(1-2p)^r}{2})^l + (1 - \frac{1-(1-2p)^r}{2}^{l-1} - (1 - \frac{1-(1-2p)^r}{2})^{l-1})\epsilon$ where $p = w^t$



(a) $F(y)$ over various values from 0 to 1 with $\epsilon = 0.05$



(b) $F(y)$ over various values from 0 to 1 with $\epsilon = 0.1$

Figure 9: Convergence to 0 for $\epsilon = 0.05$ but not 0.1

III 6.3

III.I a

for both $\rho = 0.39, 0.4$ it is positive definite. The code to determine this is shown below this section in the appendix. The variances were computed as

for $\rho = 0.39$

var
2.09554555
1.99476782
2.09554555
1.28016209

For $\rho = 0.4$

var
2.24867725
2.14285714
2.24867725
1.2962963

III.II b

Our recursion for our information matrix J is $J_{i\to j} = J_{ii} - \Sigma_{\partial i-j} J_{ik} J^{-1}_{k\to i} J_{ki}$
Implementation details are in the appendix.
The results for when $rho = 0.39$ is below

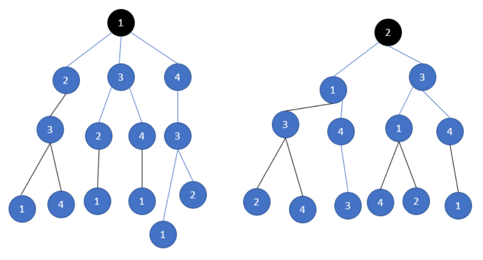
```
[[1.         0.51728683 0.56910096 0.51728683]
[0.70596584 1.         0.70596584 0.41193168]
[0.56910096 0.51728683 1.         0.51728683]
[0.70596584 0.41193168 0.70596584 1.        ]]
```

However, this is not the case for when $rho = 0.40$. Instead here we see some lack of convergence

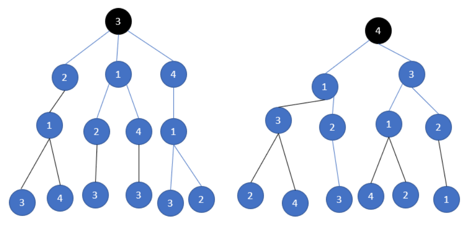
```
diff between iterations 100 and 99
[[0.         0.00439133 0.00367744 0.00439133]
[0.00434545 0.         0.00434545 0.00869089]
[0.00367744 0.00439133 0.         0.00439133]
[0.00434545 0.00869089 0.00434545 0.        ]]
diff between iterations 1000 and 999
[[0.         0.00368532 0.00306086 0.00368532]
[0.00353689 0.         0.00353689 0.00707378]
[0.00306086 0.00368532 0.         0.00368532]
[0.00353689 0.00707378 0.00353689 0.        ]]
```

Clearly, despite increasing number of iterations, there is no consensus for this value of $rho = 0.4$ in its message passes.

III.III c



(a) Unrolled comp trees for snodes 1 2



(b) Unrolled comp trees for snodes 3 4

Figure 10: unrolled to depth 3

The below demonstrates the information matrix for our computation tree starting at node 1 after unrolling 2 of the three levels listed above. Note that r = ρ

```
[[1,-r,r,r,0,0,0,0],
[-r,1,0,0,r,0,0,0],
[r,0,1,0,0,r,r,0],
[r,0,0,1,0,0,0,r],
[0,r,0,0,1,0,0,0],
[0,0,r,0,0,1,0,0],
[0,0,r,0,0,0,1,0],
[0,0,0,r,0,0,0,1]]
```

As to its positive definiteness, it is in fact positive definite. Again you can see the code used to calculate this in the appendix. This holds for both values 0.39 and 0.40, but appears to break down at 0.5. If we were to continue to expand the tree and test the matrix, we would see that it fails to be positive definite at even earlier values. Due to this, I suspect that it may even become non positive definite at 0.4, which would make sense why our naive approach of belief propagation would not converge.

IV Appendix

Code for problem 6.3