SVC:

For this report, SVM was a polynomial kernel, with C=1. Multiclass SVM performance using the two techniques, one vs rest, and one vs one, was staggering. In the one vs rest implementation I trained 10 classifiers, with the one being labeled as 1, and the rest being labeled as -1. This achieved an accuracy of .758. In the one vs one implementation, we trained 45 classifiers, trained on only the two in question, with the first one being labeled as 1, and the other one being labeled as -1. This achieved an accuracy of .122. This drop was very surprising to me because I believed that the one vs one classification would achieve a greater degree of accuracy because it is more separable, and is more specific, even though it takes longer to train. The confusion matrix for the one vs rest classification indicates that there were many misclassifications of 3s as 2s, and of 5s as 3s which make sense. Most misclassifications were of 2s. The confusion matrix of the one vs one classification shows that every point was classified as a two, which makes some sense because the other classification misclassified as 2 a lot. I'm not quite sure why this happened, as it happened when I tested with sklearn's SVC as well. I believe it is because there were more 2s in the sample than other classes, which leads to weights that favor 2 because there are more possible range of 2 values.

C was determined by trial, choosing one that increased accuracy. This is exactly how it is determined normally, in a grid search that determines optimal hyper parameters.

Graphical Model:

This algorithm will not always converge, and will sometimes get stuck in local minimums. In order to guarantee lowest value, we should randomize the initial xi and try it multiple times and pick the lowest energy overall, much like how k means is determined. We will not always find the lowest energy, and get stuck in local minimums. The lowest energy could be much different than the original image. The function is determined by the weights of the parameters, not by the original image. My best solution involved the parameters suggested by the book, with 2 iterations. This led to an accuracy of .976. I could not fight better parameters than this, regardless of modification of beta, eta, or h. Beta is responsible for how close a pixel is to its neighbors, and eta is how close a pixel is to its value in the noisy. H is responsible for how close a pixel is to its previous value. In theory a large beta should be good in this problem, but there is a limit. In order to find proper parameters, we would have to do a grid search.