Benjamin Limoges
Assignment 4

1. For the Kth Nearest Neighbor algorithm, the normalization does not seem to help much for either data set when there is no noise present. The vertebrate data set does slightly worse under normalization, while the sonar data set only does slightly better. Normalization seems to only put the units in the same scale – while important, does not seem to significantly affect our results

   This does not hold when noisy features are added to the data; there are serious gains to normalization, especially in the sonar data set. Z-score normalization works better for the noisy data, than the no-noise data. I believe the Z-scores help put all of the features on the same scale (as said above) so that way one feature does not dominate the calculation of the class. Noisy data might be of a different scale than the original "no-noise" data, allowing it to dominate the kth nearest neighbor algorithm. It most likely helps the sonar data more than the vertebrate dataset due to this.

2. The reason to not use an even k is that it would introduce another layer of tiebreakers. Since the algorithm uses an unweighted majority class vote, then it would be easier for there to be a tie if there were an even number of classes. For instance, in the sonar data set the class is either 1 or -1; if k = 2 then there could be ties, which adds a layer of unnecessary complication. For odd numbers of classes, ties are impossible if there an even number of class labels.

3. For the perceptron learning algorithm, the learning rate does not appear to have a large effect, except for the noisy data in the vertebrate dataset. Decreasing the learning rate to 0.001 from 0.01 seems to severely affect the data set. This is most likely because the stochastic gradient descent algorithm settles into a local minimum from which it cannot escape; as we learned in class, it is typically advised to start with a larger learning rate and then taper to a smaller rate. What was more surprising to me at least, is that I got identical answers between learning rates in both no-noise data sets and the sonar noisy data set. This must be because the two learning rates settled into the same local minimum after 500 iterations.

4.
   a) On the vertebrate data set with no noise, the perceptron algorithm performed better than the kth nearest neighbor algorithm, except for when k = 5. When k = 5, then it yielded the same results as the perceptron (for both values of n). For larger values of k, it exposes more of the clustering of the two classes; by exposing the clustering, you could imagine drawing your hyperplane between the two clusters. Since there is no noise in the data set, by increasing k you get more accurate results, which would result to being on the right side of the perceptron's hyperplane. This is most likely why the two results were identical for k=5.

   b) When we add noise to the dataset, the perceptron performs significantly better than the kth nearest neighbor. This is most likely since the perceptron algorithm looks through the training data 500 times before it sees the test data; in this process it discounts the outliers (as long as there are only a small number of them) as it is overwhelmed by the positive effects of the correct data. In the kth nearest neighbor, each outlier is given equal weight as the rest of the data points, which can magnify the error.

   c) For the sonar data set, without noise, the kth nearest neighbor performs better than the perceptron (for both normalized and unnormalized data). This may have been because the sonar dataset may not have been linearly separable. If the dataset is not separable, then the perceptron

will be much less accurate, as it cannot perfectly separate the data, even if given infinite time. The kth nearest neighbor has a much less stringent requirement that the data simply needs to cluster around the different class labels in the feature space.

d) When noise is added to the sonar data set, the perceptron performs better for both learning rates, than the kth nearest neighbor (except for k = 1). There exception is most likely do to chance – theoretically we would expect there to be greater volatility in lower k values. The perceptron probably performs better since it discounts the noise in the dataset more than the kth nearest neighbor. As was described in part (b), the kth nearest neighbor allows outlier data points to have the same weight as good data points, whereas the perceptron minimizes this over the 500 iterations.

e) I found that the average percentage point drop in correct identification on the kth nearest neighbor algorithm to be worse for the vertebrate dataset. For the perceptron, I had the vertebrate dataset do better with noise (an unusual result). I think the major reason the vertebrate dataset did so much poorly with noise is that the vertebrate dataset only contains 6 true features, whereas the sonar dataset has 60 features. Thus if 10 "noisy" features are added to both datasets, it will affect the vertebrate dataset more than the sonar.

	Vertebrate				Sonar			
	Unnormalized		Normalized		Unnormalized		Normalized	
	Original	Noisy	Original	Noisy	Original	Noisy	Original	Noise
K=1	77%	57%	74%	65%	82%	69%	81%	80%
K=3	81%	60%	77%	65%	81%	61%	83%	80%
K=5	85%	60%	77%	65%	76%	59%	82%	78%

	Vertebrate		Sonar	
	Original	Noise	Original	Noise
N = 0.01	85%	97%	73%	68%
N = 0.001	85%	77%	73%	68%