kMeans Style

1. Per attribute kMeans using global data
    1. Train
        1. Construct a kMeans with 2 clusters (OK and anomalous) for each attribute xi not including time stamp or node ID.
    2. Test
        1. Majority vote. Pass xtest to classifier and classify each element individually. Sum the OK/anomalous readings and classify example by majority vote.
        2. Single detection. Pass xtest to classifier and classify each element individually. If any element is anomalous, report data point as anomalous.
2. Per attribute kMeans using local data
    1. Train
        1. Split data into subsets based on NodeID. Construct a per attribute using global data kMeans for each NodeID.
    2. Test
        1. Majority vote. Pass xtest to classifier and have each of the Nodes classify data point. Sum the OK/anomalous readings and classify example by majority vote.
        2. Percentage of classifiers report anomalous, report data point.
3. Per point kMeans using global data
    1. Train
        1. Construct kMeans with 2 clusters (OK and anomalous). No including time stamp or nodeID in data points.
    2. Test
        1. Pass xtest to classifier and classify data point. Return classification.
4. Per point kMeans using local data
    1. Train
        1. Split data into subsets based on NodeID. Construct a per point kMeans using global data for each NodeID.
    2. Test
        1. Majority vote. Pass xtest to classifier and have each of the Nodes classify data point. Sume the OK/anomalous readings and classify eample by majority vote.
5. Needed items
    1. Ability to compute distance between points.
    2. Ability to test for anomalous point: Distance of point to cluster center > distance of farthest point in cluster to cluster center? → anomalous
6. Considerations
    1. Pruning. After creating the clusters, can we prune away the edges iteratively until it no longer results in better accuracy? This would, possibly, remove anomalous readings in the data that we accidentally used as normal data points.
    2. Could use EM to cluster instead of kMeans. Classification is the same. Pruning could be easier: if a point has > 5% probability to be anomalous, prune it, etc. and rebuild the model.
        1. Problems include initial conditions.
        2. Multivariate or univariate gaussian? Univariate would fit the above statements better.

Univariate Gaussian Style (Modified Andrew Ng approach)

1. Per attribute univariate gaussian using global data
    1. Train
        1. Fit a univariate gaussian to each attribute xi not including or nodeID.
    2. Test
        1. Calculate gaussian value of each attribute in xtest. Multiply values of each gaussian together. If < epsilon → anomalous!
2. Per attribute univariate gaussian using local data
    1. Train
        1. Split data into subsets based on NodeID.
        2. Train each classifier as with the per attribute univariate gaussian using global data.
    2. Test
        1. Same as with univariate gaussian using global data, but then do majority vote.
        2. Any classifier reports anomalous, report data point.
        3. Percentage of classifiers report anomalous, report data point.
3. Needed items
    1. Gaussian function
    2. A good epsilon
        1. Use CV to determine a good one...
            1. Randomly guess and use CV to test it.
4. Considerations:
    1. Pruning. After fitting the initial gaussians, can we prune away the edges iteratively until it no longer results in better accuracy? Prune some points, recalculate entire model, if it's better keep it, if it's not, stop.
    2. Transforming data to where it's more gaussian-style. Gives better results, not entirely necessary?
    3. If we graph the data and see that there are correlations (like temp and voltage) we should create a new attribute that contains the relationship between the two. And that relationship should change a lot whenever the two fall out of correlation.
        1. Temp/Voltage
        2. Light/Time

Multivariate Gaussian Style (Modified Andrew Ng Approach)

1. Global multivariate gaussian using global data
    1. Train
        1. Construct a multivariate gaussian model of the data.
        2. Construct the covariance matrix.
            1. Actually not huge considering we've only got 6 attributes.
    2. Test
        1. Same as univariate gaussian.
    3. Needed Items
        1. Multivariate gaussian creator
        2. Matrix inverter
        3. Multivariate gaussian evaluator
2. Considerations
    1. Same as with univariate gaussians
    2. The only model of the ones proposed to include temporal and nodeID in the data points! This would be really cool, as I'm sure there's some temporal correlation to the data. Light readings would obviously go down at night!

Overall considerations

1. Data set size
    1. 3.2M examples is a lot. Model 6 attributes as floats, that's 24 bytes per example. And that's 80MB of RAM just for the data itself.
2. Classification
    1. Do we need to partition the data for using with CV?
    2. Do we need to partition out some test data?
3. Algorithm accuracy
    1. How do we compute the accuracy of these algorithms? We don't actually know what the class of these data points is, they are all unlabeled.
        1. Recall, precision, F1? F1 seems like it might be good.
        2. http://stackoverflow.com/questions/657890/how-to-compute-precision-and-recall-in-clustering inter/intra cluster thing seems cool
4. Complexity
    1. Computation time should be fast for training and for testing. Especially for testing since I said I wanted this to be real time.
5. Once we detect anomalous data, what do we do with it?
    1. Sure we can report it. That's cool.
    2. What if we can smooth it out? So a robot doesn't freak the heck out?
        1. If we see too many anomalous readings though, flag it!
6. Partitioning the data into subsets to make a classifier based on each node gives us the ability to detect anomalous readings for each node better. Overfitting?
7. Creating these classifiers is going to take a lot of time. Need a way to save our work! And load later!