Multiclass Classification of gases by detecting gas particles with nanosensors using Support Vector Machines

Anamitra Datta under supervision of Navnidhi Upadhyay
Department of Electrical & Computer Engineering, University of Massachusetts Amherst
Nanodevices and Integrated Systems Laboratory

Abstract:

The olfactory system is how humans are able to process many different kinds of smells. The simulation of this organ system through electronic devices has been of interest to many industries, engineers, and academicians for quite some time. One of the ways this system can be replicated by hardware is through the use of gas sensor arrays, which can detect gas molecules by changing their resistivity. The detection of multiple gases by using a single gas sensor array within a controlled environment has been implemented in various ways with machine learning algorithms, namely: linear regression, K-nearest neighbors, and neural networks. This paper presents a feasibility study of using the Support Vector Machine algorithms to classify multiple gases by analyzing the signals generated by the gas sensor array. In this paper, we will show how to extract patterns from the sensor data and present the results of our experiment and measure the performance. The results show that Support Vector Machines, implemented through the Sequential Minimal Optimization techniques, are effective and perform better than other established methods such as Neural Networks for gas classification.

Introduction:

Our gas identification system has three layers. The first layer is a sensor array which is responsible for generating multiple time variable resistance value depending on the gas medium it is exposed to. The second layer records the change in resistivity by sampling the output of each of the sensors in the array. Finally, the data is fed into the third layer: a pattern recognition system.

Our job is to create a mathematical model, which will train on this data to extract the pattern of a particular gas and properly classify different test samples of gases. We have used a Support Vector Machine, which is a model based on linear regression, but more sophisticated. It is a method of supervised learning which tries to find an optimal hyperplane to classify the data by solving a convex objective function formulated by the training data. We then test our model on a subset of the data to do cross-validation and further refine our hyperplane equation. In the end, we run our model on the actual data and measure its performance for accuracy, time, and complexity.

We have used a two-dimensional model for detecting, at most, five different kinds of gases successfully, with ~90% accuracy. The dimensions of the classification model was determined by the number of sensors used. The sensors were chosen using sequential feature selection method which chose the two best features (sensors) that separated the data with reasonably high confidence levels. This feature selection method was done with cross-validation on our data set manually. Our algorithm is extensible to higher dimensions as well. But for the sake of visual clarity, we have presented two-dimensional and three-dimensional models in this report.

Theory:

The Support Vector Machine algorithm can be represented as an optimization problem in the form of a quadratic equation. For a meaningful solution to optimization problem, the equation has to be convex with a global minimum. The equation can be represented in two parts. The first part is to maximize the margins generated by a hyperplanes. The best margins will separate the data in positive and negative classes with the highest confidence level. The second part of the equation is the summation of the errors from points in which fall within the bounds of the margin. Our job is to find a tradeoff between these two opposing terms and hence, it is an optimization problem. The primal form of the equation is shown in figure 1.

$$\begin{aligned} \underset{\mathbf{w},b,\xi_i}{\text{minimize}} \quad & \tfrac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N}\xi_i \\ \text{subject to} \quad & y_i(\mathbf{w}^T\mathbf{x}_i - b) + \xi_i - 1 \geq 0, \quad 1 \leq i \leq N \\ & \xi_i \geq 0, \qquad\qquad\qquad 1 \leq i \leq N \end{aligned}$$

Figure 1. Primal form of the Objective Function

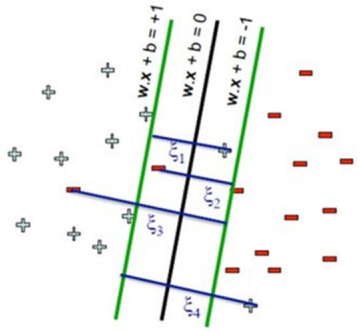


Figure 2. Hyperplane line, margins, and classification error

- w is the weight vector defining the hyperplane equation. Shown in figure 2.
- C is a regularization constant which will balance the relative importance of how much error we can allow in order to fit the data within the margin boundaries
- ξ is the error term of points which are misclassified or in between the margins. Shown in figure 2.
- y is the actual label of the data
- x is the sample vector in N-dimensions. N is the number of features used.

- b is a bias term which is inherent and a characteristic of the data set itself, but not related to the individual values of the training points themselves.

The objective function, which is in a multi-dimensional space can be solved by converting it into a sequence of subproblems each in a one-dimensional space. This can be done in two ways. By directly applying our algorithm of the primal form of the equation and looking for the minimum point by continuously taking the gradient of each point and adjusting the weights until we reach the global minimum. This is called the Stochastic Gradient Descent (SGD) algorithm. Figure 3 gives a visual representation of how this algorithm works, which is similar to dropping a ball at any arbitrary point inside a convex shape, such as a bowl. The ball will settle by finding the minimum point, in this example, due to gravity, after multiple oscillatory movements, which are similar to the iterative processes in the algorithm.

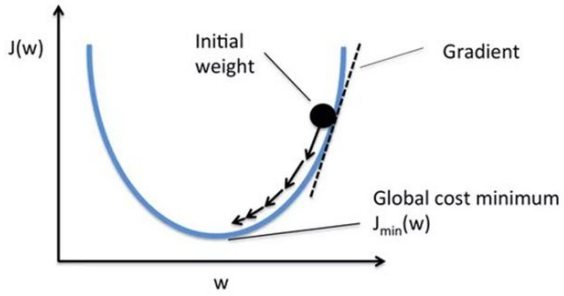


Figure 3. Gradient Descent Algorithm Representation

The other approach is called Sequential Minimal Optimization (SMO) algorithm which is also a method of solving quadratic problem by first transforming it into a problem, called the dual form, in which it finds the greatest lower bound given its constraints with respect to a single variable called a Lagrange Multiplier for a few selective points (support vectors) and then decomposes the problem into a series of subproblems, each optimizing an objective function of a small number of variables, typically changing one variable, alpha, the Lagrange Multiplier value, and treating all other variables as constant. By applying Lagrange multipliers, shown in figure 4, we arrive at the dual form of the objective function given in figure 5 after applying feasibility conditions for the greatest lower bound.

$$L_p(\mathbf{w}, b, \xi, \alpha, \mu) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{n=1}^{N}\xi_n - \sum_{n=1}^{N}\alpha_n[y_n(\mathbf{w}^T\mathbf{x} + b) + \xi_n - 1] - \sum_{n=1}^{N}\mu_n\xi_n$$

Figure 4. Applying Lagrange multipliers to Primal Form

$$\text{maximize } L_d(\alpha) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} \alpha_n \alpha_m y_n y_m \mathbf{x}_m^T \mathbf{x}_n + C \sum_{n=1}^{N} \xi_n - \sum_{n=1}^{N} \alpha_n \xi_n - \sum_{n=1}^{N} \mu_n \xi_n$$

$$= \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} \alpha_n \alpha_m y_n y_m \mathbf{x}_m^T \mathbf{x}_n$$

$$\text{subject to } 0 \leq \alpha_n \leq C, \quad \sum_{n=1}^{N} \alpha_n y_n = 0$$

Figure 5. Dual form the objective function

Experiment:

Figure 6 shows the overall architecture of our Support Vector Machine multi-class classifier.

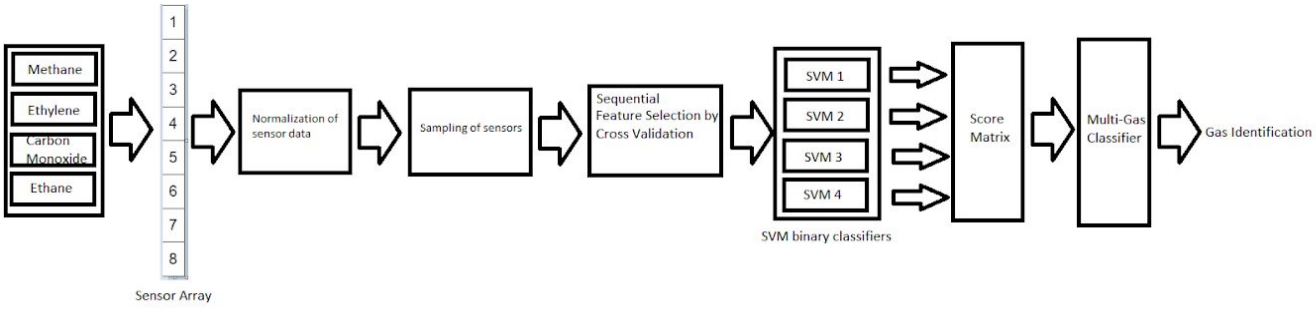


Figure 6. Support Vector Machine multi-class classifier architecture

We trained our support vector machine using 4 different gases, each with varying concentrations from 10 ppm to 100 ppm. Eight sensors were used to collect data. The data showed how the sensors changed in resistivity based on the gas it was exposed to. Figure 7 shows the typical response and the spike pattern that occurs when sensors come in contact with gas molecules.

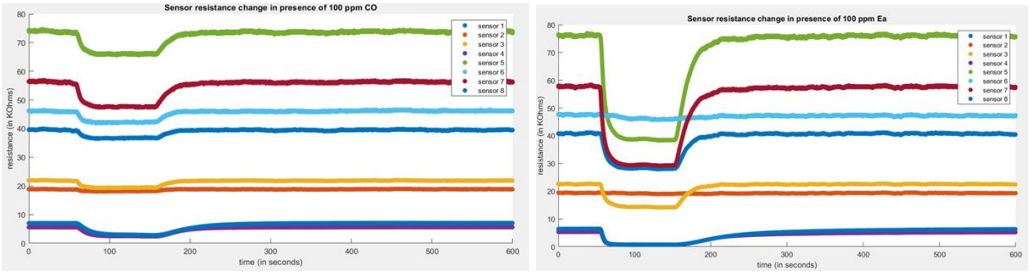


Figure 7. Sensor data showing change in resistivity in presence of different gases indicated by spikes

The data is normalized using the following formula to minimize the effect of concentration and noise captured during data collection so we can compare two different gases. The sample points are scaled from 0 to 1, 0 being the lowest and 1 being the highest.

$$X_{i,\,0\text{ to }1} = \frac{X_i - X_{Min}}{X_{Max} - X_{Min}}$$

We then sample the sensor data only at the spike to remove redundancy in the points and so the data represents the true characteristics of the spike. This is shown in figure 8.

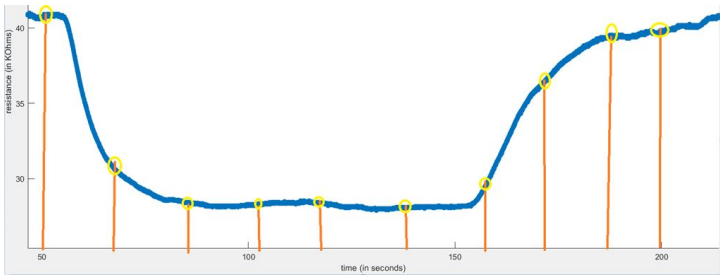


Figure 8. Sampling along the spike pattern in the sensor data

The best features (sensors) are selected from an array of 8 columns by a technique called cross-validation, which is partitioning the data into different subsets, including a test subset, and running our training algorithm repeatedly on these subsets with different combinations to see which sensors produce the best separation of positive and negative values. This process is done sequentially to extract the best features. In our experiment, we extracted the two best features as we will be running our Support Vector Machine algorithms in a two-dimensional space. Each sample point will be represented by a tuple (x_1,x_2).

The steps of the optimization algorithms given below have been implemented and tested in MATLAB. Both SGD and SMO algorithms are given below.

The following is the Stochastic Gradient Descent algorithm we have followed in our project:

1. Initialize weight vectors to zero
2. Set learning rate to one. Learning rate determines at what rate we modify the weight vectors. Learning rate will decay with each iteration.
3. Set the batch size of all our training samples
4. Set number of iterations. It is approximately ten times the size of our data set
5. Iterate over each sample in the data set starting at random points every time

6. Calculate the gradient and update the weights with the partial derivative of the objective function
7. If properly classified, the change will only take care of the margin, if not, it will take care of error as well.

The following is the Sequential Minimal Optimization algorithm we have followed in our project, which iterates over support vectors and checks KKT conditions for necessary and sufficient conditions for optimality and adjusts alphas accordingly:

1. Initialize all alphas and bias terms to zero
2. Loop while alpha values stop changing
3. For every alpha, do the following:
    a. Find the prediction error
    b. Multiply the prediction term with the label
    c. Check if KKT conditions are violated
        i. If so, find another alpha and find its prediction error
        ii. Determine upper and lower limits of alpha depending on the label
        iii. Find eta (a term relating the two alphas) for updating the second alpha value
        iv. Clip alphas for their lower and upper limits
        v. Update the other alpha value
        vi. Update the bias term
4. If alphas stop changing, exit the loop
5. Calculate weights, bias values, and number of support vectors

In order to implement multi-class classification, we create a distance vector for all of the test points by gathering the scores of positively classified samples from all of the Support Vector Machines. The score is indicative of the distance of the point to the separating hyperplane for a particular model. The maximum score indicates the highest likelihood of a sample to be positively identified as a particular class of gas. Hence, the column of the distance matric with the highest value is chosen as the predicted class. Distances for each point are calculated as shown in figure 9.

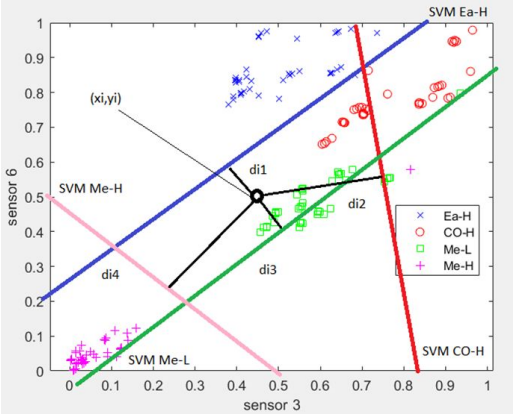


Figure 9. Distances of the point (x_i,y_i) from the four SVMs

Results:

Sampled Two-Gas Separation:

I first tested the Support Vector Machine using two different gases. High concentrations of Ethane and High concentrations of Ethylene. Figure 10 shows the data points of the two sampled gases. The sensors were chosen using sequential feature selection.

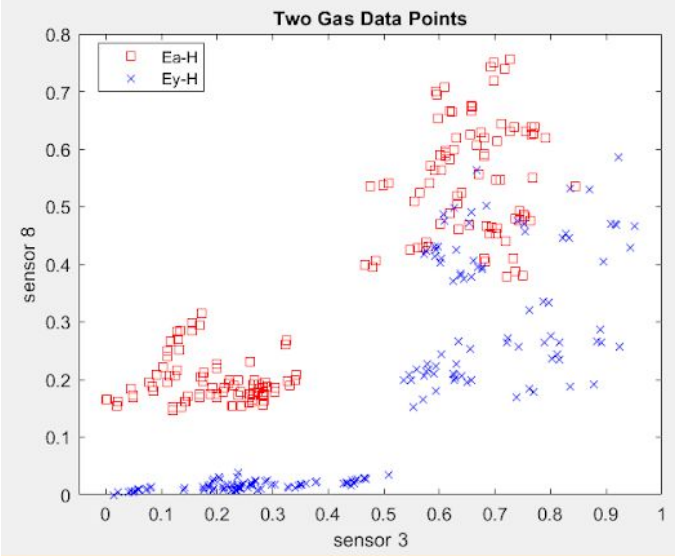


Figure 10. Data points of high concentrations of Ethane and Ethylene

The two figures below are the Support Vector Machine hyperplanes and margins generated by our algorithms. The colored regions indicate the prediction of what gas a point on the graph is most likely to be. The third figure is the hyperplane generated by the inbuilt MATLAB API using their version of Sequential Minimal Optimization, which is similar to our results.

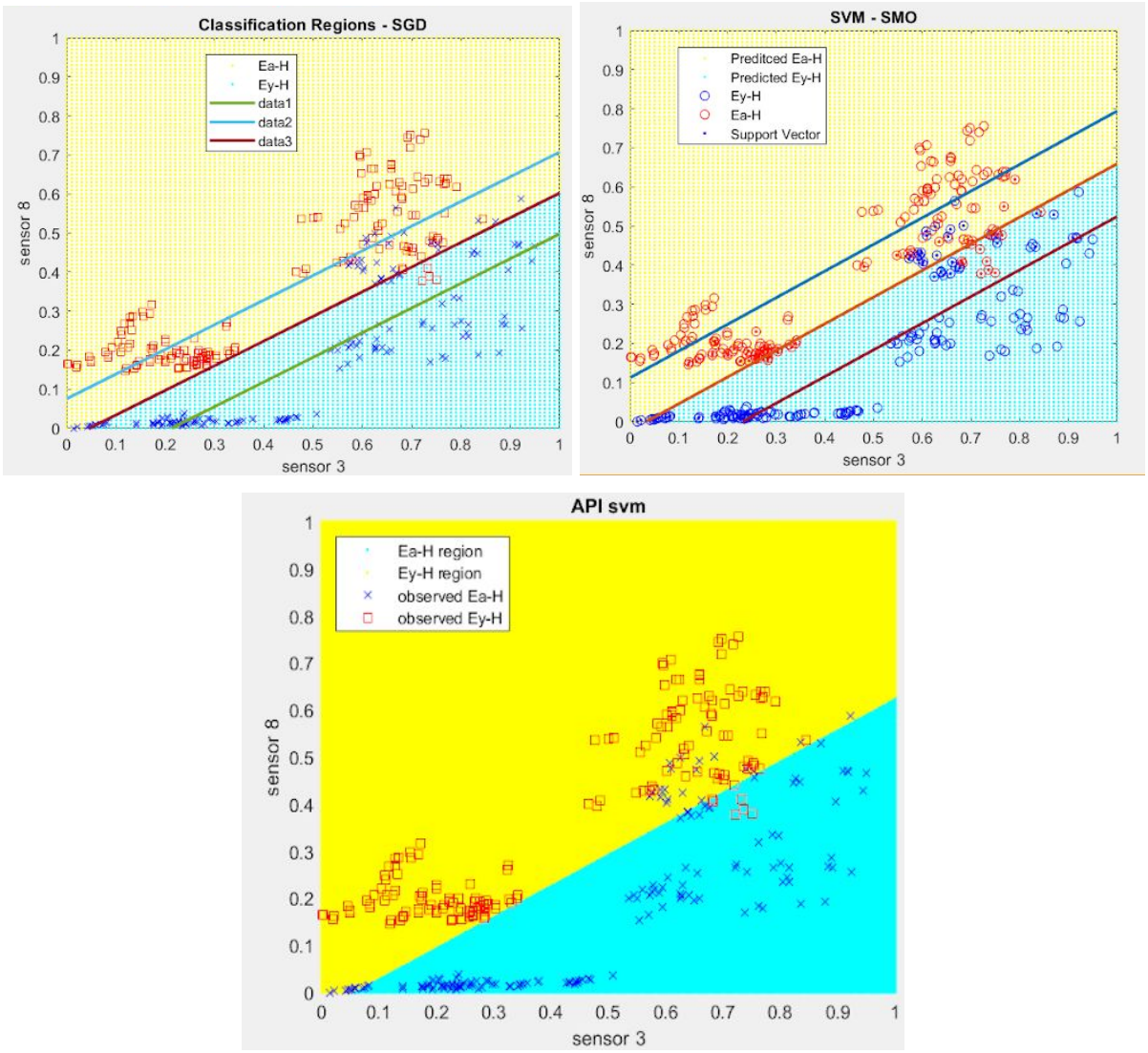


Figure 11. SGD, SMO, and API Support Vector Machines

Four Gas Multi-Class Classification:

I further tested the Support Vector Machine algorithms for multi-class classification. I tested this using for different classes: High concentrations of Ethane, Carbon Monoxide, and Methane, and a low concentration of Methane.

Figure 12 shows the plot of the data points of the four classes. The two sensors were chosen using sequential feature selection.

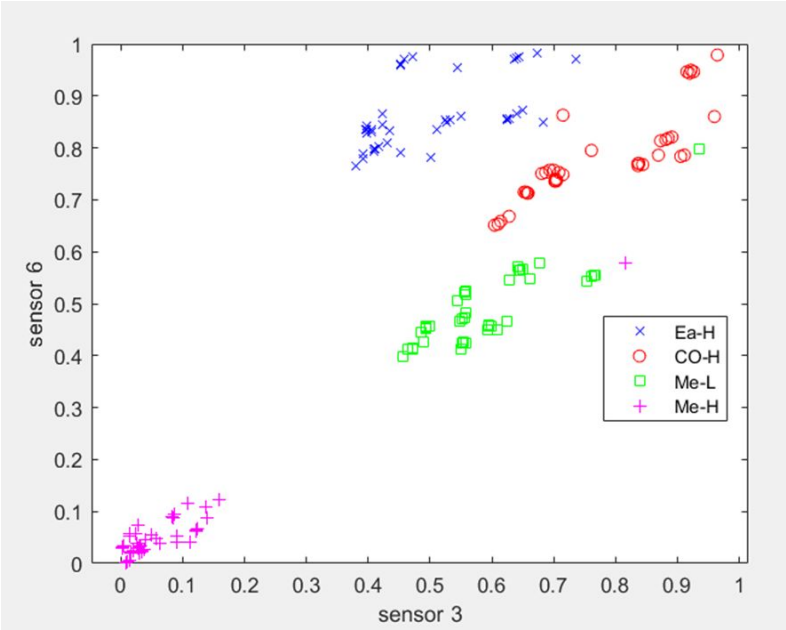
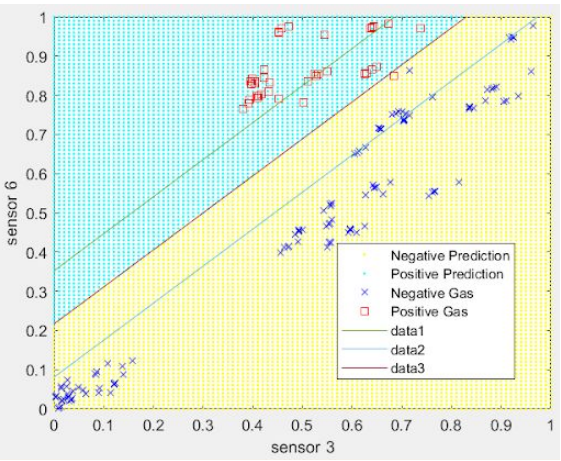


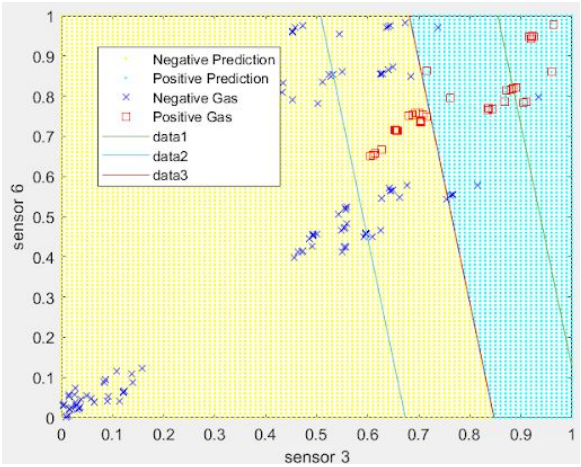
Figure 12. Plot of four different gases

We then used our Support Vector Machine algorithms to separate each class from all of the other classes, giving four SVMs as a result, as shown in figure 13.

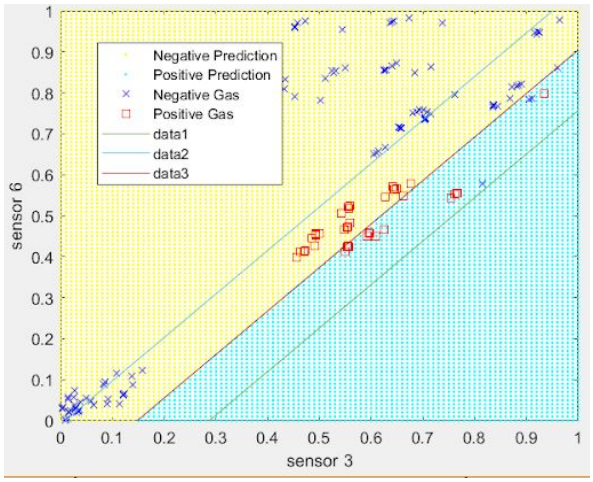
Figure 13. SVMs generated from SGD algorithm



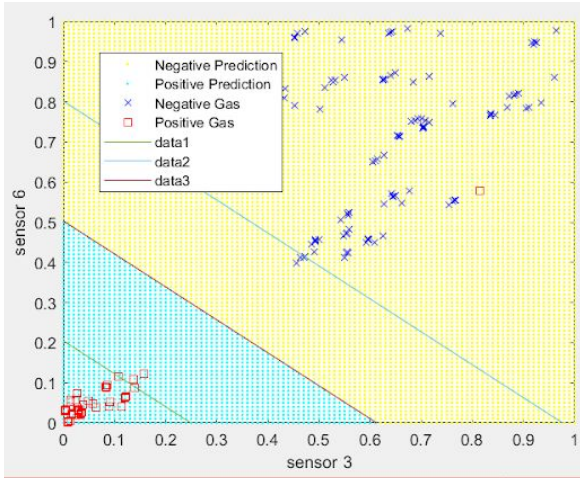
Ethane Separation-SGD



Carbon Monoxide Separation-SGD



Methane Low Separation- SGD

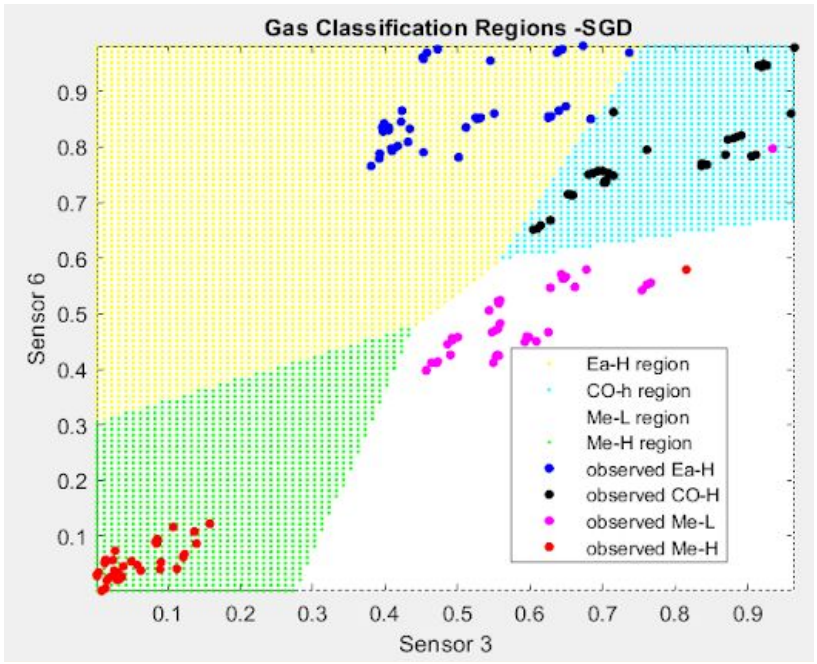


Methane High Separation- SGD

We then superimposed these four support vector machines generated from the SGD algorithm on each other to implement multi-classification.

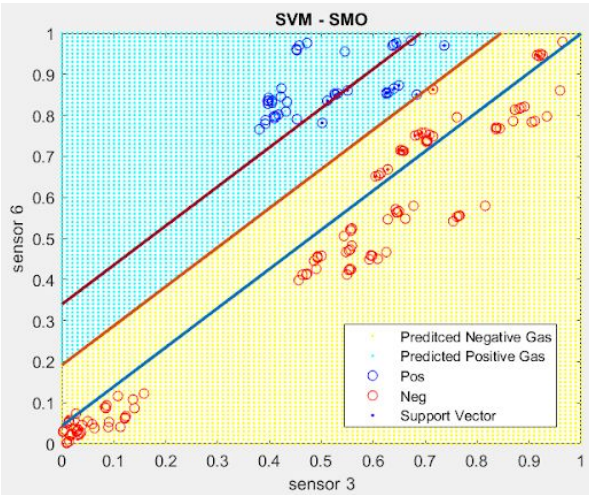
The result of performing multi-classification on this data set with SGD is shown in figure 14.

Figure 14. Four Gas Multi-Class Classification using SGD algorithm

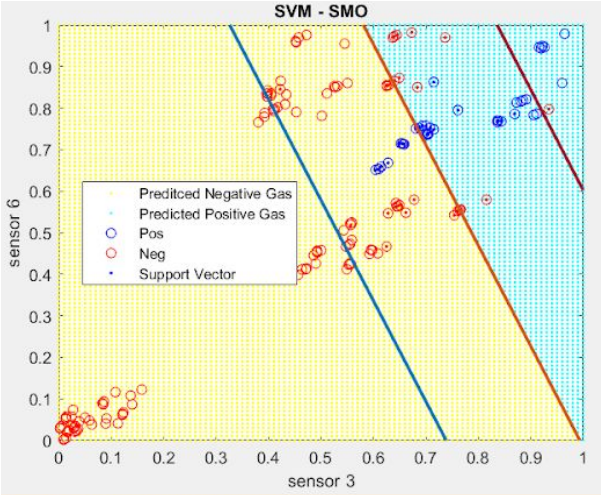


We did the same thing using the SMO algorithm. Figure 15 shows the SVMs generated using SMO for the same data set.

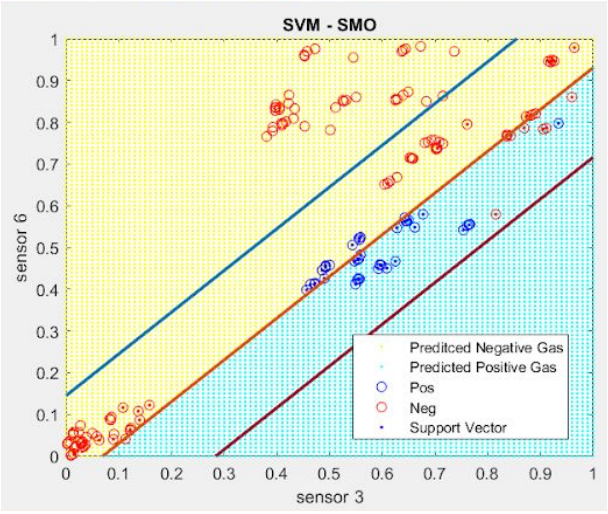
Figure 15. SVMs generated from SMO algorithm



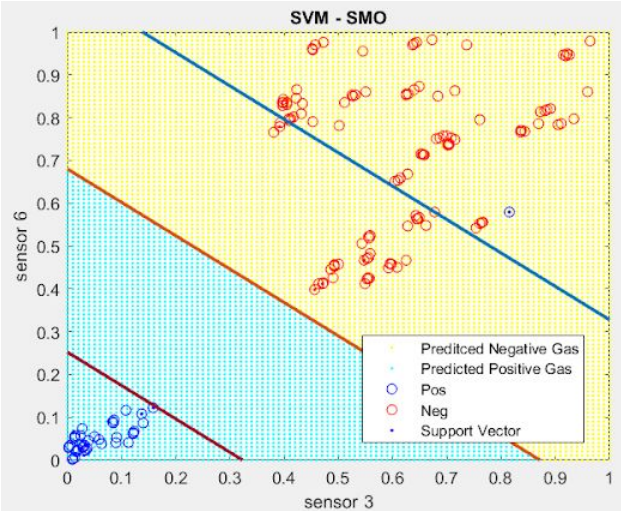
Ethane Separation-SMO



Carbon Monoxide Separation-SMO



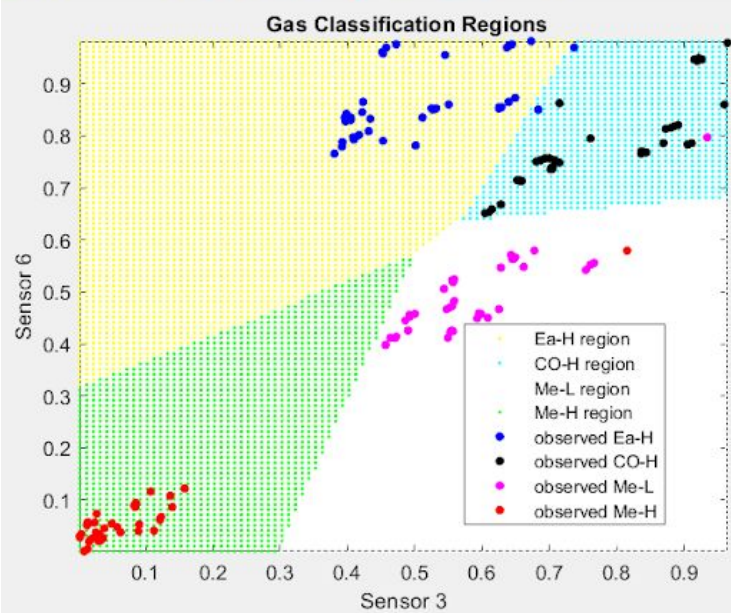
Methane Low Separation- SMO



Methane High Separation- SMO

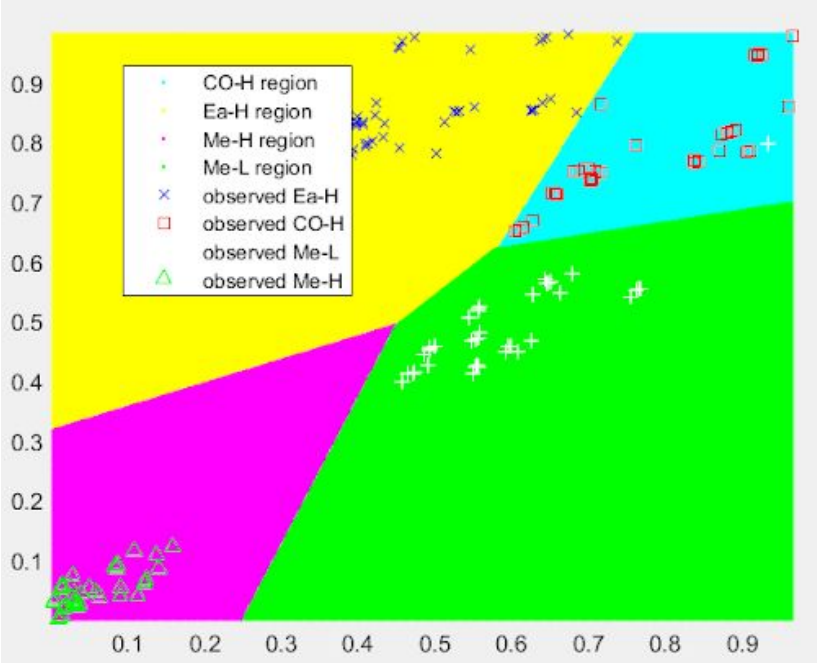
We then superimposed these four support vector machines generated from the SMO algorithm on each other to implement multi-classification. The result of performing multi-classification on this data set with SMO is shown in figure 16.

Figure 16. Four Gas Multi-Class Classification using SMO algorithm



We compared our Multi-Class SVM with the one generated from the MATLAB API using their own version of SMO. This is shown in figure 17.

Figure 17. Four Gas Multi-Class Classification using MATLAB API



Our Support Vector Machine multi-class classifier is similar to the one produced by the MATLAB API

Five Gas Multi-Class Classification:

The maximum number of classes we are able to classify at a time with two sensors in two dimensions is five, as shown in figure 18 We can use three dimensions (three features/sensors) to plot five different gases and use a multi-classifier Support Vector Machine to distinguish the class as shown in figure 19. The sensors were chosen with sequential feature selection.

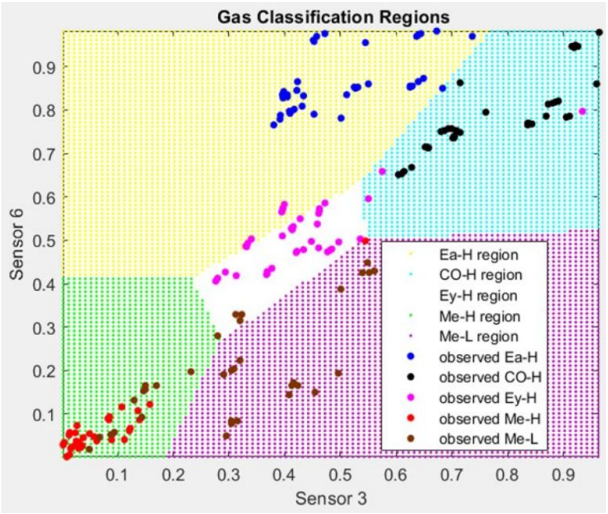


Figure 18. Five Gas Classification with SGD in 2D

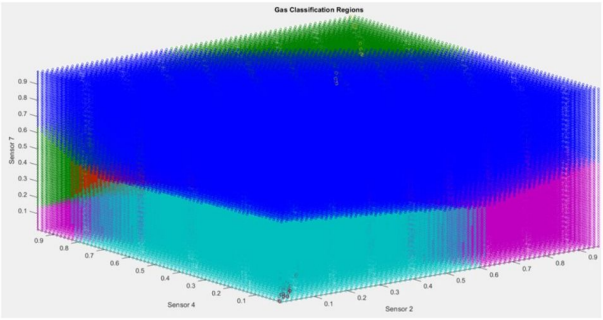


Figure 19. Five Gas Classification with SGD in 3D

Discussion:

While Stochastic Gradient Descent and Sequential Minimal Optimization algorithms perform with about the same accuracy, they have a few differences.

Stochastic Gradient Descent

- Works directly on the primal form of objective function
- Manipulates weights without any constraints
- Can only do linear classification
- Manipulates all points

Sequential Minimal Optimization

- Works directly on dual form of objective function
- Maintains a balance of sum derived from dual form
- Can accommodate non-linear kernels
- Only manipulates a small subset of points called Support Vectors
- Works well in high dimensions compared to SGD

Stochastic Gradient Descent has a complexity of about $O(N^2)$, N being the number of samples in the data set. Sequential Minimal Optimization also has about $O(N^2)$ or about $O(N^3)$ complexity. SMO takes much less time and has many benefits Stochastic Gradient Descent doesn't have,

such as providing information about Support Vectors and accommodating non-linear kernels. It is also more efficient that Stochastic Gradient Descent since it only iterates through Support Vectors, which are the only points with non-zero alpha values that make up a small subset of the data, while SGD has to iterate through the entire data set and could potentially take more time than SMO. Figure 20 shows the complexity analysis of SGD and SMO

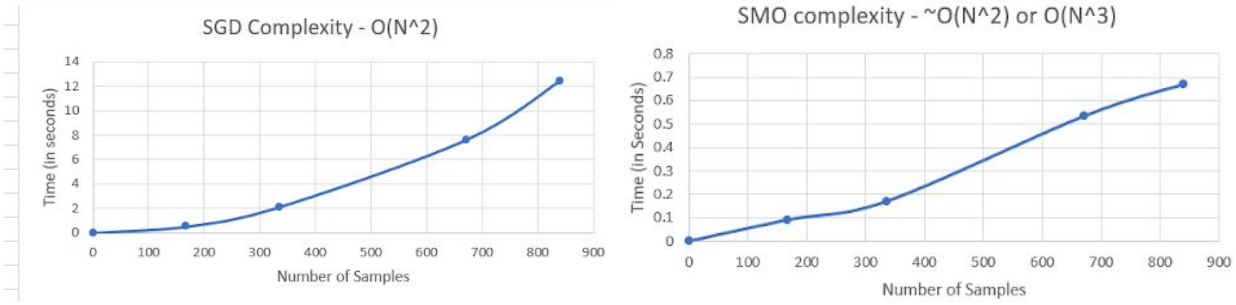


Figure 20. Complexity of SGD and SMO algorithms

Figure 21 shows the accuracy of SMO vs. SGD. While both algorithms converge to the same accuracy, SGD fluctuates and converges less quickly than SMO does, which converges quickly and stabilizes much better and more smoothly.

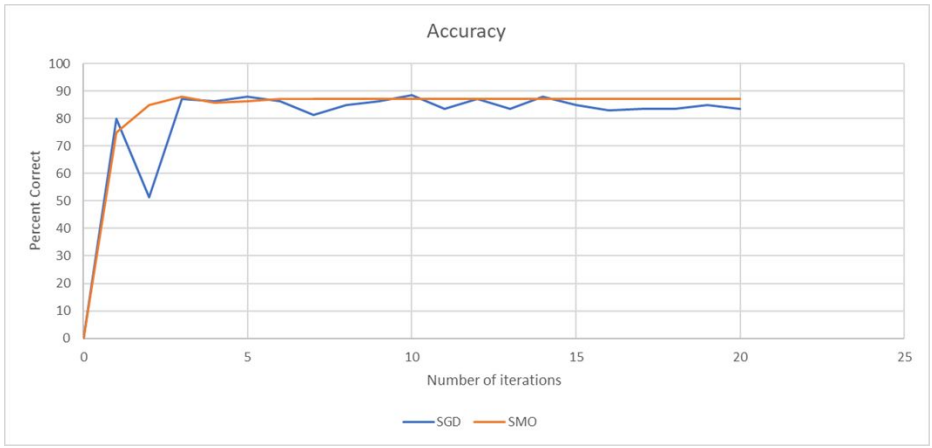


Figure 21. Accuracy of SGD vs. SMO

Next, we compared our SMO SVM and the SMO SVM generated by MATLAB with Neural Networks. We used the Neural Network API generated by the MATLAB toolbox as our test neural network for comparison. Figure 22 shows the time each algorithm took. The MATLAB API was the fastest and most reliable for a small training set compared to Neural Networks, proving that SVMs are generally faster and much more efficient than Neural Networks when given a small data set. The point to note is that our SMO code is not precompiled machine code, whereas the APIs for both SMO and Neural Networks have the advantage of calling the pre-compiled library, which is optimized binary code.

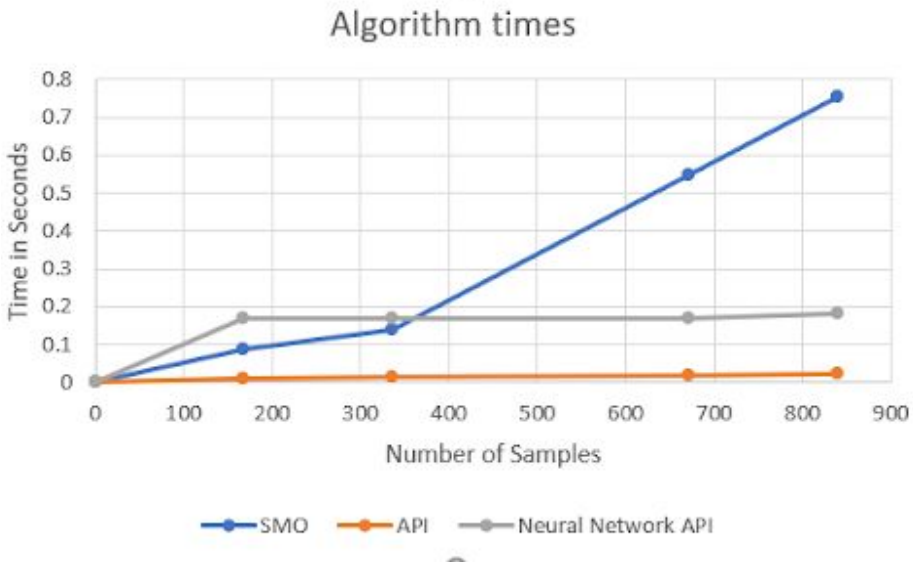


Figure 22. Algorithm Times Comparison for different machine learning algorithms

Conclusion:

We were able to successfully classify up to five different gases using Support Vector Machine algorithms. However, in most cases, more than five gases did not produce satisfactory results because at least one of the combinations for the binary classifier were not linearly separable. The introduction of another dimension improved the situation, but not dramatically. We have not used other kernels such as RBF/Gaussian kernels or Polynomial kernels. Using those kernels might help. Improving the quality of the sensor data might also help as well.

However, the performance of SMO was very promising and showed better results than the Neural Network did for a small data set. If we could run our MATLAB code as precompiled library code, just like the MATLAB API code, we would have better results, even for higher sample sizes.

Next, we are going to implement a Support Vector Machine on hardware involving the use of a memristor crossbar array and measure the performance with other existing systems.

Acknowledgements:

I would like to thank Navnidhi Upadhyay, who was my mentor for this project. He was very generous with his time and his knowledge in this subject. Also, let me express my sincere gratitude to Professors Qiangfei Xia and Joshua Yang for giving me the opportunity to work in their labs and for their encouragement during this experience. Lastly, I would like to thank the NSF for funding this research project.

References:

1. Platt, John. "Sequential minimal optimization: A fast algorithm for training support vector machines." (1998).
2. Drakos, Nikos, and Ross Moore. “Support Vector Machine.” *Sequential Minimal Optimization (SMO) Algorithm*, fourier.eng.hmc.edu/e176/lectures/ch9/node9.html.
3. Daumé III, Hal. *A Course in Machine Learning*. Self Published, 2017.
4. Ng, Andrew. *“Machine Learning.” Support Vector Machines.* Youtube. Dec 31, 2016 https://www.youtube.com/watch?v=hCOIMkcsm_g