Hand Gesture Detection Using Least Squares and SVM

Alexander Bush	abush4@wisc.edu
Alex Pletta	apletta@wisc.edu
Ting-Hung Lin	tlin89@wisc.edu

Abstract

So far in class we've looked into multiclass classifiers that are built and designed to do just that, such as neural networks. While binary classifiers are intrinsically simpler to understand, the inputs can only be "True" or "False", real world datas are generally more complicated with multiple potential classes. With the intention to apply binary classifiers to a more applicable setting, this activity will introduce multiclass classification algorithms with the foundation of binary classifiers.

In order to perform multiclass classification with models that aren't designed with this capability, two common heuristics are the One vs Rest and the One vs One algorithm. Later in this activity, we will investigate in depth the combinational use of SVM with the two heuristics. Least squares will also be included in this activity as a comparison baseline.

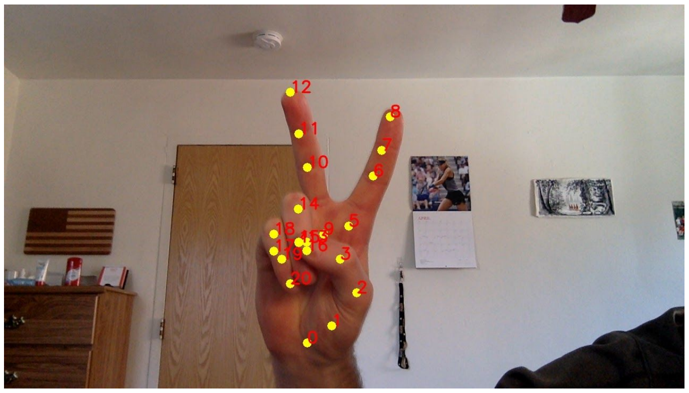
To help understand the concepts and operations of the One vs Rest and the One vs One, we will first quickly review how the binary classifier SVM works. In the main activity section, we have prepared a series of guiding questions meant to accelerate the understanding of this topic, a simple data set will be provided for easy visualization.

If the simple data set isn't challenging enough, we have also collected a set of labeled hand-gesture features for further exploring. This will give some insights to a realistic application for these heuristic multiclass classifiers and they perform respectively.

Background

- *Problem description*

  The final goal of this activity is to use binary classifiers to perform multiclass classification; The specific application we have chosen is to detect hand gestures with SVM classifier + two different heuristics . The hand features have been extracted using OpenCV and a hand model developed and published by Carnegie Mellon University [1], this hand model is available at [2].



(Image above is an actual data sample we've collected with the aforementioned feature extraction method)

After the initial data collection, raw feature data is stored in ‘data_train.json’ and ‘data_test.json’. In order to have a more meaningful feature set, we preprocessed the data into pixel distance between the feature point 0 and feature points 1~20. The json parser and feature processing code are provided. Finally, these features will be classified, trained, and tested using three different methods: Least Squares, SVM and Least Squares with One vs Rest, and SVM with One vs One.

- *History and importance*

  Mark Weiser once said, “The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it.”

  As the 21st century has progressed, people have become more and more accustomed to computers. This has led to more interaction between humans and computers. However, computers have not always been able to recognize gestures that people have. Recently, in the last ten years, there has been an increasing amount of research that has gone into gesture detection to allow computers to make classifications based on images. These gestures could ultimately be useful to interact with computers to play a video game. Many of these projects use SVM’s to train the models[3].

  In addition hand gestures have always been an important source of non-verbal communication. Sign language is fundamentally built on hand gestures, and computers being able to recognize hand gestures may provide a vast improvement for the deaf population to communicate with others virtually. Recently there have been projects to allow this to happen using complex neural networks[4]. While this project will not focus on complex neural networks, it will focus on the ability to classify these gestures into multiple groups.

  The use of gesture detection will always have ethical questions about it. The foundation of this problem is to use images, and naturally people do not always want to be photographed. Therefore, this technology should not be used unless the person being photographed is willing to have their picture taken. However, this does not always happen. A similar process that we will be using in this project can be applied to facial technology[5], and while some people may not care if their hand is pictured, using facial recognition to unwilling participants is oftentimes viewed as an invasion of privacy. Some other potential applications are to have automated bartenders. One problem in this application is that many hand gestures mean different things to different people around the world. In addition the recognition would be different if someone did not have 2 arms and 10 fingers[6].

- *Mathematical formulation. Lets review!*

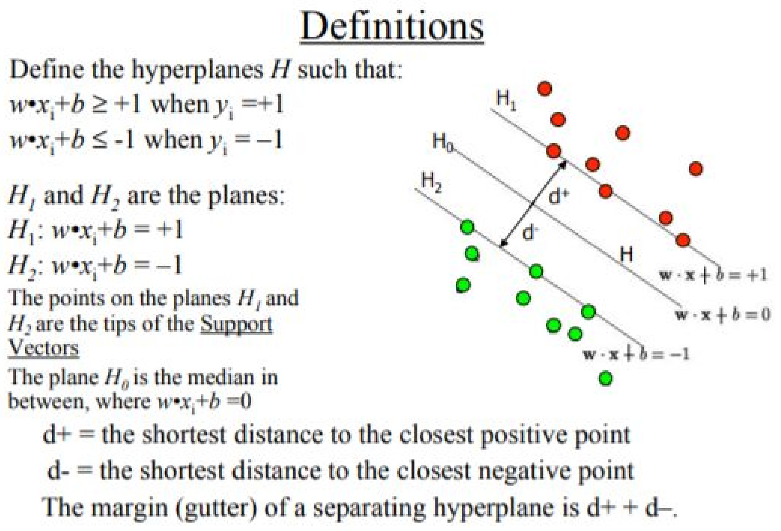
Least Squares

The least squares is often thought as the line of best fit for a set of data. It will minimize the squared error of every point in the data set. It has a closed form solution. For X feature data, w weights and b labels, the optimization problem and solution statement is:

$$Xw = b$$
$$min_w = \lVert Xw - b \rVert_2^2$$
$$w = (X^T X)^{-1} X^T b$$

SVM

The main difference between the SVM and the least squares methods is that the SVM approach uses an iterative process to determine the weights, whereas the Least Squares is a closed form solution. In addition SVM maximizes the margin between a decision boundary and boundaries through points close to the boundary, called support vectors, by using hinge loss. A great advantage of using hinge loss is that the calculated boundary has no error for all points labeled correctly. SVM can additionally use kernels to allow for more complex decision boundaries.



Definitions

Define the hyperplanes H such that:
$w \bullet x_i + b \geq +1$ when $y_i = +1$
$w \bullet x_i + b \leq -1$ when $y_i = -1$

H_1 and H_2 are the planes:
H_1: $w \bullet x_i + b = +1$
H_2: $w \bullet x_i + b = -1$
The points on the planes H_1 and H_2 are the tips of the Support Vectors
The plane H_0 is the median in between, where $w \bullet x_i + b = 0$

d+ = the shortest distance to the closest positive point
d- = the shortest distance to the closest negative point
The margin (gutter) of a separating hyperplane is d+ + d–.

H_1
H_0
H_2
d+
d-
H
$\mathbf{w} \cdot \mathbf{x} + b = +1$
$\mathbf{w} \cdot \mathbf{x} + b = 0$
$\mathbf{w} \cdot \mathbf{x} + b = -1$

(Figure above extracted from" An Idiot's guide to Support vector machines (SVMs)" [7], for more information on SVM, check out this simple and informative read.)

SVM solves the following optimization problem. Since there is no closed form solution iterative techniques must be used to minimize the loss function. For gradient descent, many techniques exist for running until convergence including specifying a number of iterations and checking the size of the loss at each step.

$$min_w = \sum_i^n (1 - b_i X_i^T w)_+ + \lambda \lVert w \rVert_2$$

until converged:

$$w_{k+1} = w_k - \nabla f(w) = w_k - \left(\sum_i^n (-b_i X_i I_{\{b_i X_i^T w_k < 1\}}) + 2\lambda w\right)$$

Warm up

1) After reviewing the binary classification techniques that were learned in class, Compare the Least Squares and SVM methods for binary classification. Give at least two differences between the two methods.

2) Write out the optimization problem and steps to solve for the weights that could classify between two classes A and B using Least Squares. Include matrix dimensions.

3) Write out the optimization problem and steps to solve for the weights that could classify between two classes A and B using SVM's. Include matrix dimensions and remember to include the constant term at the end of the features. Should the constant term be the same for the features and weights?

Main Activity

Now that we have reviewed the binary classification methods, we see how we can use these techniques for multi-class classification. The methods that we will look at are Least Squares, One-vs-Rest, and One-vs-One.

Least Squares approach

The first potential solution would be to use least squares. In some applications it is possible to calculate the least squares as we have done numerous times in class, but instead of using the sign function to classify the features into binary classes, use other indicators to classify the resulting prediction. As an example, later we will be looking at hand gestures. Our classes will be grouping the number of fingers we are holding up. If we classify a hand, and we are holding four fingers up, then it would be penalized more to classify to the group where we are holding no fingers up, than the class that we are holding one finger up.

One-vs-Rest

The One-vs-Rest(OvR) also known as the One-vs-All(OvA) approach is exactly like the title says. For every class that we have, we are going to do a binary classification to determine whether a specific sample will be classified in that group or not. As an example, if we had four classes, A,B,C, and D. The first binary classification test would be comparing class A to class B,C, and D. If we are classifying with *n* classes, there will be *n* binary classifications required.

The class that is predicted for each sample is by selecting the classifier with the highest evaluation score, such evaluation score is simply $y = x^T w$, classifier weights applied to its corresponding feature values.

One-vs-One

The One-vs-One(OvO) method is similar to the OvR method, but instead of comparing each classification to the set of remaining classes, each class is compared to a single other class at a time. All pairs of samples are compared and the class that is best after comparing all of the pairs of data is selected. This method is used to solve the potential issue of data imbalance in the OvR method.

As an example, if we had four classes, A,B,C, and D. The first binary classification test would be comparing class A to class B. If we are classifying with *n* classes, there will be *n*(n-1)/2* binary classifications required. The predicted class that is selected for each sample is the class that was chosen the most in the m binary classifications. In example above, if one sample was actually A, then we would expect class A to be selected the most in the m comparisons. In addition, every other class is expected to be selected fewer times than A.

Main Activity Questions

1) What values are typically used as known labels for Least Squares binary classification? Can these values change? Write a sentence or two for how to train and test a Least Squares classifier for a dataset containing 3 different class labels.

2) Write out what binary comparisons would be needed for One-vs-Rest using 3 classes A, B, C? Repeat for using 4 classes A, B, C, D?

3) Write out what binary comparisons would be needed for One-vs-One using classes A, B, C? Repeat for classes A, B, C, D?

4) After being introduced to these techniques for multi-classification, what advantages and disadvantages do you see? What problems could arise for each technique?

Coding

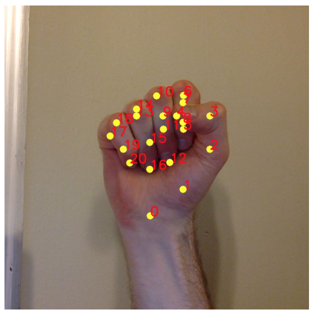
We will now implement the code. First we will write the code for the toy model final. This contains three clusters of data. Skeleton code has been provided to show the points, and additional code has been provided to check the error of the results. The red cluster of points is labeled 0, the green is 1, and the blue is 2. This code will then be used on the gesture detection data.

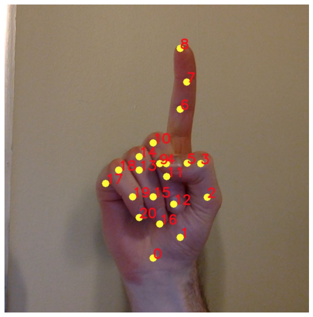
5) Implement the Least Squares, One-vs-One, and One vs-Rest code in the sections provided in the Toy Data file. What model worked best with this data? Why do you think that this is the case?

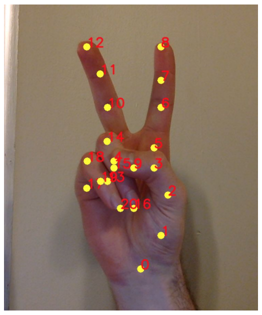
6) Try increasing the noise_multiplier to 3 to increase the test data variance. (see code block under ‘Toy Data’. What do you notice about the performance differences LS alone and One vs Rest? How about between SVM and LS for One vs Rest? Could either SVM or LS these methods be adapted for non-separable data and if so how?

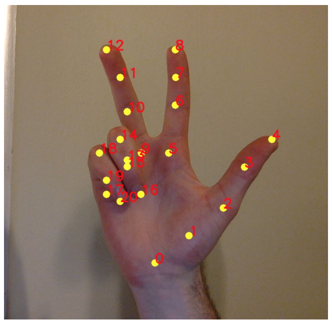
Hand Model

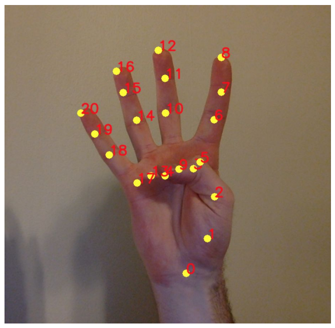
Now we will implement the code in the Hand detection file. As discussed previously, instead of 3 classes we will have 6 classes. The code for the Toy data should be very similar to the hand data. The features will be the distance from the reference point, 0, to every other point not including 0. The training and testing features were all very similar to these classes and no other gestures were used. Here we strictly enforce the data collection to only these 6 types of gestures for a clean data set.

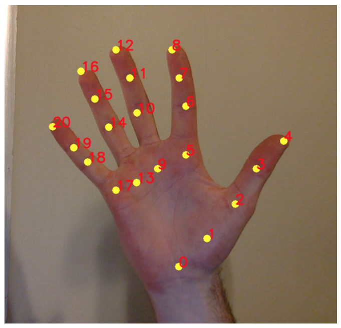

Class 0


Class 1


Class 2


Class 3


Class 4


Class 5

7) Each of the images above represent a gesture class. The spots represent detected key points on the gesture. How many classes and features are there? Write out the matrices that will be used to solve this problem. Remember to include differences for Least Squares and for SVM.

8) Implement the Least Squares, One-vs-One, and One-vs-Rest code in the sections provided in the Hand Data file. What model worked best with this data? Why do you think that this is the case?

9) What would we need to change if we did not have a dataset of linearly separable data? What other ways could we improve the code to classify more complex hand gestures?

10) Try out data_NOISY.json. What do you think is wrong with it?

Bonus Activity:
This is not a part of the activity. If you wish to try this yourself feel free to look at the code.

- Clone our repo[8]
- install opencv
    - Ex. $ pip install opencv-contrib-python
- Take a test picture of your own hand
    - $ python3 take_pic.py
- Calculate keypoints
    - $ python3 detect_keypoints.py
- Try running hand_model.ipynb
- See github readme for help

References

[1] https://arxiv.org/pdf/1704.07809.pdf
[2] http://posefs1.perception.cs.cmu.edu/OpenPose/models/hand/pose_iter_102000.caffemodel
[3] https://shop.tarjomeplus.com/UploadFileEn/TPLUS_EN_4807.pdf
[4] American Sign Language Hand Gesture Recognition
[5] Face Detection For Beginners
[6] https://arxiv.org/pdf/1705.05884.pdf
[7] An Idiot's guide to Support vector machines (SVMs)
[8] https://github.com/apletta/hand-detection

APPENDIX SOLUTION!!!

Warm up

Q1

- LS = easy, closed-form solution, only works with linearly separable data
- SVM = solved with iterative methods, not as sensitive to outliers, can be used with kernels for non-separable data

Q2 something similar to the following is acceptable, this can be taken directly from the background information

$$Xw = b$$
$$dim(X) = n\,x\,m,\; dim(b) = n\,x\,1$$
$$X = [x_1\; x_2\; ...\; x_n\; 1]$$
$$\widehat{w} = [w_1\; w_2\; ...\; w_m]$$
$$w = (X^T X)^{-1} X^T b$$

Q3 something similar to the following is acceptable, this can be taken directly from the background information

- $Xw = b$
- $X = [x_1\; x_2\; ...\; x_n\; 1]$
- $\widehat{w} = [w\; 0]$
- $min_w = \sum_i^n (1 - b_i X_i^T w)_+ + \lambda \| w \|_2$
- until converged:

$$w_{k+1} = w_k - \nabla f(w) = w_k - \left(\sum_i^n (-b_i X_i I_{\{b_i X_i^T w_k < 1\}}) + 2\lambda w\right)$$

Main activity

Q1 Write out how you could use Least Squares to classify a dataset that contains 3 different classes.

- Typical labels are b = +/- 1
- Labels can be any value

- Answers may vary, code will help direct this later on. Recommended solution:
    1) training
        a) “Normal” approach: $w = (X^T X)^{-1} X^T b$
    2) testing
        a) round up or down between cutoff values where the true label is also within the cutoff values

Q2

3 classes - There 3 classifications required

Binary Classification 1: A vs B,C
Binary Classification 2: B vs A,C
Binary Classification 3: C vs A,B

4 classes - There are 4 classifications required

Binary Classification 1: A vs B,C,D
Binary Classification 2: B vs A,C,D
Binary Classification 3: C vs A,B,D
Binary Classification 4: D vs A,B,C

Q3

3 classes - There are 3(2)/2 = 3 classifications required

Binary Classification 1: A vs B
Binary Classification 2: A vs C
Binary Classification 3: B vs C

4 classes - There are 4(3)/2 = 6 classifications required

Binary Classification 1: A vs B
Binary Classification 2: A vs C
Binary Classification 3: A vs D
Binary Classification 4: B vs C
Binary Classification 5: B vs D
Binary Classification 6: C vs D

Q4

- LS pros=easy, closed-form solution, only run once, can work with different labels cons=must have linearly separable data and classes must be ordered
- OvR pros=relatively easy, linear scale w/ number of classes (only run number of classes) cons=impacted by data imbalance, can be affected by data size and slow models
- OvO pros= data imbalance (can be very important) cons=tie resolution, slightly more difficult to code, need to run more classification tests

Data imbalance when applying OvR algorithms may become an issue. If there are a lot of classes, then there will be a very low likelihood that any of the values will classify as true. For example if there is a class of size 5, and a data set of a million points, that class will almost never be true. Another issue would be if we had too large of a data set, since we are using all of the samples it will be slow.

One of the big issues with OvO is that it does not account for ties. If in the binary classification and the value that is positive is the class that it is classified to. If

+A vs -B
+B vs -C
+C vs -A

then all of the classes will have an equal probability of being classified, but only one of those classes will be chosen. OvO is not as prone to data imbalance, and with large data sets it is less prone to problems with the data size. For example if the size of the rest in OvR is 1 million, and the size of O is 100, then it will not need to run through the extra almost 1 million values that the other classes run through.

Least Squares on its own the data might not be ordered or separable. In addition it is very sensitive to outliers. However it is very easy to code and fast. It is much quicker than the other methods.

Q5 See attached code

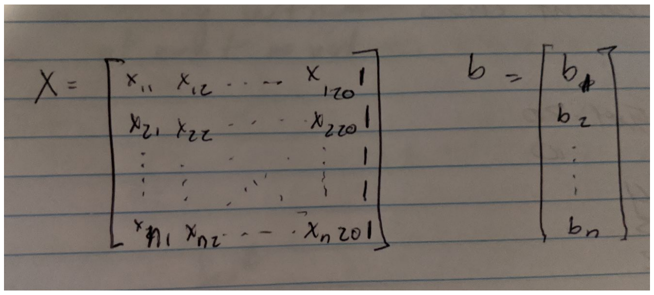
OvO and OvR both have 0% error rate. It's difficult to judge which model is better just with this information. Nice if data imbalance of OvR is mentioned, artificially introducing a larger classes into training the model not ideal compare to OvO

Q6

- LS alone becomes very difficult to classify with increased noise
- SVM performs slightly better than LS for One vs Rest
- LS cannot be adapted for non-separable data, SVM can be by using kernels

Q7 There are 6 classes, and 20 features.

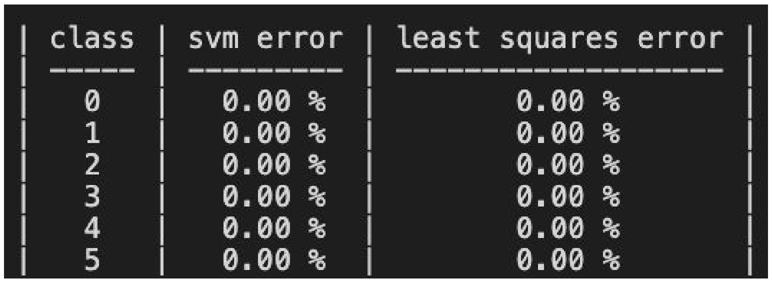
For Least Squares:



$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{120} & 1 \\ x_{21} & x_{22} & \cdots & x_{220} & 1 \\ \vdots & & & \vdots & 1 \\ \vdots & \vdots & \ddots & \vdots & 1 \\ x_{n1} & x_{n2} & \cdots & x_{n20} & 1 \end{bmatrix} \quad b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ \vdots \\ b_n \end{bmatrix}$$

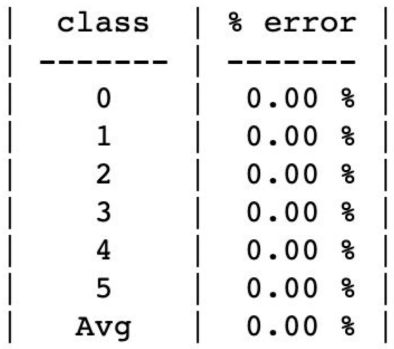
For SVM:

Q8 See attached code



```
| class | svm error | least squares error |
| ----- | --------- | ------------------- |
|   0   |  0.00 %   |       0.00 %        |
|   1   |  0.00 %   |       0.00 %        |
|   2   |  0.00 %   |       0.00 %        |
|   3   |  0.00 %   |       0.00 %        |
|   4   |  0.00 %   |       0.00 %        |
|   5   |  0.00 %   |       0.00 %        |
```

expected answer for OvR



```
|  class  | % error |
| ------- | ------- |
|    0    |  0.00 % |
|    1    |  0.00 % |
|    2    |  0.00 % |
|    3    |  0.00 % |
|    4    |  0.00 % |
|    5    |  0.00 % |
|   Avg   |  0.00 % |
```

expected answer for OvO

Horrible LS, arbitrarily assigning cutoffs that have no particular meaning in this context.
Both models excel at classifying the hand data. Given that data acquisition was under controlled environment and gesture was restricted to one type per class , such behavior is expected. Good if they question if something this good actually exists in a real world data set. Error would be added if there was more variation in each class gesture.

Q9

Using kernels will allow significantly more flexibility when creating decision boundaries. These would also allow you to classify non-linearly separable data. The code could efit the features to include ratios of distances, or have some comparable value to scale the distances between photos. In addition, different hand gestures could be used to allow a wider range of images to be detected

Q10 features collected was too noisy, and the features were too complex to classify correctly with code provided.