Support Vector Machine With Data Reduction
# Table of Contents

Summary ............................................................................................................................................. 3

1. Introduction of Support Vector Machines .................................................................................. 3
   1.1 Brief Introduction of Support Vector Machines ................................................................... 3
   1.2 SVM Simple Experiment ...................................................................................................... 4
      1.2.1 SVM Algorithm ........................................................................................................... 4
      1.2.2 Simple SVM Output .................................................................................................... 4

2. Data Reduction ............................................................................................................................. 5
   2.1 Idea Proposal ....................................................................................................................... 5
   2.2 Brief Introduction of Convex Hulls ..................................................................................... 6
   2.3 Experiment ......................................................................................................................... 7

3. Real Data Experiment ................................................................................................................. 7

4. Conclusion ..................................................................................................................................... 10

5. Reference ....................................................................................................................................... 10
Summary

The principal component analysis (PCA) is unsupervised, it is difficult to cluster the data into groups. However, if data information is available, then labeled training data can be applied to supervise learning, which is easier and more precisely. Support vector machine (SVM) is one of the most popular supervised methods in machine learning. It can efficiently perform the linear or non-linear classification using kernel trick. However, SVM has noticeable limitations. For example, only the support vectors define the boundary. The rest of data points are meaningless for the classification. Therefore, I am investigating a faster and more convenient method to improve support vector machine.

1. Introduction of Support Vector Machines

1.1 Brief Introduction of Support Vector Machines

The support vector machine is a supervised classification algorithm in machine learning. Given a set of training data that contain information for blood glucose and BMI (figure 1), SVM can be used to identify the classification boundary for predicting whether diabetes is associated with the sample donor.

In SVM, it often utilizes $w^T x = 0$ for finding the hyperplane more than uses $y - ax - b = 0$. In figure 2, a point, $x$, is on the plane if

\[
\begin{align*}
(x - x_0) \perp w \\
w \cdot (x - x_0) &= 0 \\
w \cdot x - w \cdot x_0 &= 0 \\
w \cdot x + b &= 0
\end{align*}
\]
The goal of SVM is to maximize the margin, which is the distance between two hyperplanes. In figure 3, (a) will be the optimal solution for SVM.

\[ w \cdot x + b = 0 \]

\[ y_i = +1 \]

\[ y_i = -1 \]

(a) \hspace{2cm} (b) \hspace{2cm} (c)

Figure 3 illustrate 3 sets of hyperplanes, where the hyperplanes in 3(a) has the maximum margin, or largest \( d \) value.

1.2 SVM Simple Experiment

1.2.1 SVM Algorithm

\[
\text{svm() built-in function for SVM classification} \\
\text{# compute the SVM model} \\
svm.model <- svm(type ~ ., data=data, type='C-classification', kernel = 'linear', scale =FALSE) \\
\text{# get the support vectors' index} \\
svi <- svm.model$index \\
\text{# get parameters of hyperplane} \\
w <- t(svm.model$coefs) %*% svm.model$SV \\
b <- -svm.model$rho \\
\]

Table 1 SVM Algorithm with simply using svm function in R

Here, I put “linear” of kernel in svm function which gives me the idea that I probably don’t have to use svm to deal with linear separation dataset.

1.2.2 Simple SVM Output

I randomly generated a set of data and use the svm function in R. Figure 4 demonstrates the data set and the calculated hyperplane. The hyperplane is estimated using a few data points that are closest to the plane (circled data). Other data points are not contributing to the data classification.
2. Data Reduction

2.1 Idea Proposal

SVM has a very high cost in computation. As shown in some publication, SVMs have good generalization performance, but they tend to be ineffective during test phase (Burges, 1996; Burgess 1998; Osuna and Girosi, 1998). The SVMs array dimension is limited to 2E9 elements, hence it is not suitable for analyzing large data set.

In order to overcome this limitation, some studies explore a reduce support machine. Kuan-Ming Lin and Chih-Jen Lin published “A Study on Reduced Support Vector Machine” (https://www.csie.ntu.edu.tw/~cjlin/papers/rsvmTEX.pdf) and Juh-Jye, Lee wrote “RSVM: Reduced Support Vector Machine” (http://www.siam.org/meetings/sdm01/pdf/sdm01_13.pdf). Building on the reduction concept, I purpose to show a simplified approach for linear binary classification without the complicated mathematics, thus avoid the heavy computation in SVM.

I started by assuming that the linear support vectors have certain features. First, support vectors are in-between two classes. I already proved this assumption in figure 4, in which the support vectors belong to 2 different clusters that are close to each other. Second, support vectors are outside of clusters and farthest from clusters centroid. In figure 5(a), the circled data in each cluster represent four of the farthest data from the centroid. However, the support vectors do not contain all of these data as shown in figure 5(b).

Figure 4 SVM Output

Figure 5a) Circled data are the four perimeter data points furthest from centroid.

5b) Circled data are the Support Vectors from SVM
Subsequently, I applied convex hull algorithm to find the perimeter points. After connected the perimeter points, I identified the support vectors within those data point.

2.2 Brief Introduction of Convex Hulls

Figure 6 shows the step-by-step approach of convex hull.

- **Step 1**: Choose the two points furthest to the left as starting point.
- **Step 2 and 3**: Add the third and fourth points.
- **Step 4**: Remove the point because of maintaining the upper hull when adding the fourth point.

This approach continues until all data points are analyzed. When finished, an example of the results is illustrated in figure 6.

![Figure 6 Example of Convex Hull](image)

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>CONVEXHULL(P)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input</strong></td>
<td>A set $P$ of points in the plane.</td>
</tr>
<tr>
<td><strong>Output</strong></td>
<td>A list containing the vertices of $CH(P)$ in clockwise order.</td>
</tr>
<tr>
<td>1.</td>
<td>Sort the points by $x$-coordinate, resulting in a sequence $p_1, \ldots, p_n$.</td>
</tr>
<tr>
<td>2.</td>
<td>Put the points $p_1$ and $p_2$ in a list $L_{upper}$, with $p_1$ as the first point.</td>
</tr>
<tr>
<td>3.</td>
<td>for $i \leftarrow 3$ to $n$ do Append $p_i$ to $L_{upper}$.</td>
</tr>
<tr>
<td>5.</td>
<td>while $L_{upper}$ contains more than two points and the last three points in $L_{upper}$ do not make a right turn do Delete the middle of the last three points from $L_{upper}$.</td>
</tr>
</tbody>
</table>

**Table 1 Convex Hull Algorithm**
2.3 Experiment

After studying the convex hull algorithm, I applied it to determine the perimeter points. As shown in figure 7, I found the support vectors were within the perimeter points.

![Convex Hull Perimeter and Support Vectors](image)

Even though using convex hull algorithm could save a significant amount of time, not all perimeter points are support vectors. Only the few perimeter points in between two clusters are support vectors. Therefore, the convex hull algorithm needs a new constraint to precisely determine the support vectors. One of the approaches is to identify vectors that points at each other centroids. For each vector, select the data on the perimeter that are within ± 95° from the vectors (figure 8a). This approach identified the circled data as shown in right plot of figure 8a, which matches the support vectors from SVM. After the data reduction experiment, I compare the running time with SVM and perimeter solution. There is a significant time reduction using this method, with the timing saving being proportional to the data size (figure 8b).

![Points on the perimeter ≤ ± 95°](image)

3. Real Data Experiment

In order to study the application of this approach, I use Iris plants data for my experiment. There are 4 attributes: sepal length, sepal width, petal length, and petal width; and 3 classes: Iris Setosa, Iris Versicolour,
and Iris Virginica. Because this project is for binary linear classification, I use two attributes at a time and merge Iris Versicolour and Iris Virginica into one class.

From figure 9 to figure 11, convex hull algorithm works very well. I got support vectors by using convex hull algorithm as same as using SVM function in R. Also, the most important is saving a lot of time.

![Iris data set with sepal width vs. petal length](image1)

![Iris data set with sepal vs. petal lengths](image2)

![Iris data set with petal length vs. petal width](image3)

In figures 9 - 11, there are only a few data points that are support vectors, which are convenient when there are only two sets of data. However, when there are more points in-between two classes, I need to get more than one convex hull. Because a data could be important to determine the support vector in two classes but insignificant in other classes, I don’t want a single convex hull which will remove some important points which may potentially be support vectors. In figure 12 to 14, I compare that only run one convex hull (r=1) and run several times of convex hull (r>1).
Figure 12 Iris data set with sepal length vs. sepal width

Figure 13 Iris Data set with sepal length vs. petal width

Figure 14 Iris data set with sepal vs. petal widths
4. Conclusion

Except using SVM function for binary linear classification, using convex hull algorithm to find the perimeter points for boundary reduces the runtime significantly. It also removes the limitation of 2E9 data points issue. From my experiment, over all, using convex hull algorithm works fine. But this method only works for linearly separable data. For non-linearly separable data, we may consider to use RSVM (Reduced Support Vector Machines).

5. Reference

Computational Geometry: Lecture 1 Introduction and Convex Hulls

