Data-analysis and Retrieval
Support Vector Machines

Ad Feelders

June 13, 2014
Overview

1. Linearly Separable Case
2. Kernel Functions
3. Allowing Errors
4. SVM’s in R
Outline

1. Linearly Separable Case
2. Kernel Functions
3. Allowing Errors
4. SVM’s in R
Support vector machines

- Machine-learning research in the last two decades has improved classifier effectiveness.
- New generation of state-of-the-art classifiers: support vector machines (SVMs), boosted decision trees, regularized logistic regression, neural networks, and random forests
- Applications to IR problems, particularly text classification

**SVMs: A kind of large-margin classifier**

Vector space based machine-learning method aiming to find a decision boundary between two classes that is maximally far from any point in the training data (possibly discounting some points as outliers or noise)
Linear Classifier for two classes

Linear model

\[ f(\vec{x}) = \vec{w}^\top \vec{x} + b \]

with \( y_i \in \{-1, +1\} \).

- Predict \( y_0 = +1 \) if \( f(\vec{x}_0) \geq 0 \) and \( y_0 = -1 \) otherwise.
- The decision boundary is given by \( f(\vec{x}) = 0 \).
Data linearly separable

Assume training data is linearly separable, so there is at least one choice of \( \mathbf{w}, b \) such that:

1. \( f(\mathbf{x}_i) > 0 \) for \( y_i = +1 \);
2. \( f(\mathbf{x}_i) < 0 \) for \( y_i = -1 \);

that is, all training points are classified correctly.

Putting 1. and 2. together:

\[
y_i f(\mathbf{x}_i) > 0 \quad i = 1, \ldots, N
\]
Two-class training data
Many Linear Separators
SVM Decision Boundary
Maximize Margin
Support Vectors

[Diagram showing support vectors and the decision boundary]
Maximum Margin

- There may be many solutions that separate the classes exactly.
- Which one gives smallest prediction error?
- SVM chooses line with maximal margin, where the margin is the distance between the line and the closest data point.
- Points near decision surface → uncertain classification decisions (50% either way).

A classifier with a large margin makes no low certainty classification decisions.
Gives classification safety margin w.r.t slight errors in measurement or doc. variation
Let $\vec{x}_a$ and $\vec{x}_b$ be two points on the decision boundary $f(\vec{x}) = \vec{w}^T \vec{x} + b = 0$. Since $f(\vec{x}_a) = f(\vec{x}_b) = 0$, we have

$$(\vec{w}^T \vec{x}_a + b) - (\vec{w}^T \vec{x}_b + b) = \vec{w}^T (\vec{x}_a - \vec{x}_b) = 0,$$

so the vector $\vec{w}$ is orthogonal to the decision boundary.
Distance of a point to a line

\[ \vec{w}^\top \vec{x} + b = 0 \]
Distance to decision surface

We have

\[ \vec{x} = \vec{x}_\perp + r \frac{\vec{w}}{|\vec{w}|}. \]

where \( \vec{x}_\perp \) is the orthogonal projection of \( \vec{x} \) onto the line \( f(\vec{x}) = 0 \), \( r \) is the (signed) distance of \( \vec{x} \) to the line, and \( \frac{\vec{w}}{|\vec{w}|} \) is the unit vector in the direction of \( \vec{w} \). Multiply left and right by \( \vec{w}^\top \) and add \( b \):

\[ \vec{w}^\top \vec{x} + b = \vec{w}^\top \vec{x}_\perp + b + r \frac{\vec{w}^\top \vec{w}}{|\vec{w}|} \]

So we get

\[ r = f(\vec{x}) \frac{|\vec{w}|}{|\vec{w}|^2} = \frac{f(\vec{x})}{|\vec{w}|} \]
The distance of $\vec{x}_i$ to the decision boundary is:

$$d = \frac{|f(\vec{x}_i)|}{|\vec{w}|}$$

For lines that separate the data perfectly, we have $y_i f(\vec{x}_i) = |f(\vec{x}_i)|$, so that the distance is given by

$$d = \frac{y_i f(\vec{x}_i)}{|\vec{w}|} = \frac{y_i (\vec{w}^T \vec{x}_i + b)}{|\vec{w}|}$$
Maximum margin solution

Now we can formalize the problem.

Solve

$$\arg \max_{\vec{w}, b} \left\{ \frac{1}{|\vec{w}|} \min_i [y_i (\vec{w}^\top \vec{x}_i + b)] \right\}$$

Since $\frac{1}{|\vec{w}|}$ does not depend on $i$ it can be moved outside of the minimization.

- Direct solution of this problem would be rather complex.
- A more convenient representation is possible.
Canonical Representation

The decision boundary is defined by

$$\vec{w}^\top \vec{x} + b = 0$$

Then also

$$\kappa(\vec{w}^\top \vec{x} + b) = \kappa \vec{w}^\top \vec{x} + \kappa b = 0$$

so rescaling $\vec{w} \rightarrow \kappa \vec{w}$ and $b \rightarrow \kappa b$ gives just another representation of the same decision boundary. Choose scaling factor such that

$$y_i(\vec{w}^\top \vec{x}_i + b) = 1$$

for the point $\vec{x}_i$ closest to the decision boundary.
Canonical Representation (square=1, circle=−1)
Canonical Representation

In this case we have

\[ y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 \quad i = 1, \ldots, N \]

Optimization Problem

\[
\arg \min_{\mathbf{w}, b} \frac{1}{2} |\mathbf{w}|^2
\]

subject to the constraints

\[ y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 \quad i = 1, \ldots, N \]

This optimization problem (quadratic program) has a unique global minimum.
Example: find the SVM decision boundary

- Direction of weight vector?
- What are the support vectors?
- Support vectors have $f(\vec{x}) = 1$ or $f(\vec{x}) = -1$
Working geometrically

- The maximum margin weight vector will be parallel to the shortest line connecting points of the two classes, that is, the line between (1, 1) and (2, 3), giving a weight vector of (1, 2).
- The optimal decision surface is orthogonal to that line and intersects it at the halfway point. Therefore, it passes through (1.5, 2).
- The SVM decision boundary is:

\[ w_1 x_1 + w_2 x_2 + b = x_1 + 2x_2 + b = 0 \]
\[ 1.5 + 4 + b = 0 \]
\[ b = -1.5 - 4 = -5.5 \]
\[ f(x_1, x_2) = x_1 + 2x_2 - 5.5 = 0 \]
Working algebraically

- We know that the solution is $\vec{w} = (c, 2c)$ for some $c$. So (use the support vectors):

  $$c + 2c + b = -1$$
  $$2c + 6c + b = 1$$

- Hence, $c = 2/5$ and $b = -11/5$. So the optimal hyperplane is given by $\vec{w} = (2/5, 4/5)$ and $b = -11/5$.

- Equation of decision boundary:

  $$f(x_1, x_2) = \frac{2}{5}x_1 + \frac{4}{5}x_2 - \frac{11}{5} = 0$$

- Same decision boundary but all coefficients multiplied by $\frac{2}{5}$. 

Support vector machines
Dual Representation

Maximize

\[ L(\vec{a}) = \sum_{i=1}^{N} a_i - \frac{1}{2} \sum_{i,j=1}^{N} a_i y_i a_j y_j \vec{x}_i^\top \vec{x}_j \]

with respect to \( \vec{a} \) and subject to the constraints

\[ a_i \geq 0, \quad i = 1, \ldots, N \]

\[ \sum_{i=1}^{N} a_i y_i = 0. \]

The \( a_i \) are called Lagrange multipliers.
Outline

1. Linearly Separable Case
2. Kernel Functions
3. Allowing Errors
4. SVM’s in R
Mapping

\[ f(\mathbf{x}) = \mathbf{w}^\top \phi(\mathbf{x}) + b = 0 \]

\( \phi \) maps \( \mathbf{x} \) into higher dimensional space where data is linearly separable.
Kernel Function

- We map $\vec{x}$ to a high-dimensional space $\phi(\vec{x})$ in which data is linearly separable.
- Performing computations in this high-dimensional space may be very expensive.
- Use a kernel function $k$ that computes a dot product in this space (without making the actual mapping):

$$k(\vec{x}, \vec{x}') = \phi(\vec{x})^\top \phi(\vec{x}')$$
Example: polynomial kernel

Suppose $\vec{x} \in \mathbb{R}^3$ and $\phi(\vec{x}) \in \mathbb{R}^{10}$ with

$$\phi(\vec{x}) = (1, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_3, x_1^2, x_2^2, x_3^2, \sqrt{2}x_1x_2, \sqrt{2}x_1x_3, \sqrt{2}x_2x_3)$$

Then

$$\phi(\vec{x})^\top \phi(\vec{z}) = 1 + 2x_1z_1 + 2x_2z_2 + 2x_3z_3 + x_1^2z_1^2 + x_2^2z_2^2 + x_3^2z_3^2$$

$$+ 2x_1x_2z_1z_2 + 2x_1x_3z_1z_3 + 2x_2x_3z_2z_3$$

But this can be written as

$$(1 + \vec{x}^\top \vec{z})^2 = (1 + x_1z_1 + x_2z_2 + x_3z_3)^2$$

which costs much less operations to compute.
Polynomial kernel: numeric example

Suppose \( \vec{x} = (3, 2, 6) \) and \( \vec{z} = (4, 1, 5) \).

Then

\[
\phi(\vec{x}) = (1, 3\sqrt{2}, 2\sqrt{2}, 6\sqrt{2}, 9, 4, 36, 6\sqrt{2}, 18\sqrt{2}, 12\sqrt{2}) \\
\phi(\vec{z}) = (1, 4\sqrt{2}, 1\sqrt{2}, 5\sqrt{2}, 16, 1, 25, 4\sqrt{2}, 20\sqrt{2}, 5\sqrt{2})
\]

Then

\[
\phi(\vec{x})^\top \phi(\vec{z}) = 1 + 24 + 4 + 60 + 144 + 4 + 900 + 48 + 720 + 120 = 2025.
\]

But

\[
(1 + \vec{x}^\top \vec{z})^2 = (1 + (3)(4) + (2)(1) + (6)(5))^2 = 45^2 = 2025
\]

is a more efficient way to compute this dot product.
Linear kernel

\[ k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^\top \mathbf{x}' \]

Two popular non-linear kernels are the polynomial kernel

\[ k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^\top \mathbf{x}' + c)^Q \]

and Gaussian (or radial) kernel

\[ k(\mathbf{x}, \mathbf{x}') = \exp\left(-|\mathbf{x} - \mathbf{x}'|^2 / 2\sigma^2\right), \]

or

\[ k(\mathbf{x}, \mathbf{x}') = \exp\left(-\gamma |\mathbf{x} - \mathbf{x}'|^2\right), \]

where \( \gamma = \frac{1}{2\sigma^2} \).
Maximize

\[
L(\vec{a}) = \sum_{i=1}^{N} a_i - \frac{1}{2} \sum_{i,j=1}^{N} a_i y_i a_j y_j \phi(\vec{x}_i)^\top \phi(\vec{x}_j)
\]

with respect to \( \vec{a} \).

To solve the optimization problem it is not required to map \( \vec{x} \) into the high dimensional space because we can compute the required dot product using the kernel function.
Prediction

Once the optimization problem has been solved, the values of the Lagrange multipliers $a_i, i = 1, \ldots, N$ are determined. We have

$$a_i > 0 \Leftrightarrow \vec{x}_i \text{ is a support vector.}$$

All other data points have $a_i = 0$.

To predict the class label of a data point $\vec{x}$ we compute:

$$f(\vec{x}) = \sum_{i=1}^{N} a_i y_i k(\vec{x}, \vec{x}_i) + b,$$

and take the sign to predict $+1$ or $-1$. 

(1)
How to determine the value of $b$

Since for any support vector $\vec{x}_i$, $y_i f(\vec{x}_i) = 1$, for support vector $\vec{x}_i$, we have

$$y_i \left( b + \sum_{j=1}^{N} a_j y_j k(\vec{x}_i, \vec{x}_j) \right) = 1$$

Hence we have

$$y_i b + y_i \sum_{j=1}^{N} a_j y_j k(\vec{x}_i, \vec{x}_j) = 1$$

$$y_i b = 1 - y_i \sum_{j=1}^{N} a_j y_j k(\vec{x}_i, \vec{x}_j)$$

$$b = y_i - \sum_{j=1}^{N} a_j y_j k(\vec{x}_i, \vec{x}_j) \quad \text{(2)}$$

since $y_i \in \{-1, +1\}$ and so $1/y_i = y_i$. 
Prediction

- Note that only the support vectors play a role in the prediction, since all other data points have $a_i = 0$.
- This makes prediction more efficient, especially if there are only a few support vectors (compare to prediction with kNN).
- Prediction is performed using the kernel function, so again, we don’t have to make the mapping $\phi(\vec{x})$. 
We receive the following output from the optimization software for fitting a support vector machine with linear kernel and perfect separation of the training data:

<table>
<thead>
<tr>
<th>$i$</th>
<th>$x_{i,1}$</th>
<th>$x_{i,2}$</th>
<th>$y_i$</th>
<th>$a_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-2</td>
<td>2</td>
<td>-1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>1</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>6</td>
<td>+1</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>4</td>
<td>+1</td>
<td>$\frac{9}{8}$</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>5</td>
<td>+1</td>
<td>0</td>
</tr>
</tbody>
</table>
The figure below is a plot of the same data set, where the dots represent points with class $-1$, and the crosses points with class $+1$. 
Prediction: Example

(a) Compute the value of the SVM bias term $b$.
Data points with $a > 0$ are support vectors.
Let's take the point $x_1 = 4, x_2 = 4$ with class label +1:

$$b = y_m - \sum_{i=1}^{N} a_i y_i \vec{x}_m \vec{x}_i = 1 + [4 \ 4] \begin{bmatrix} 1 \\ 3 \end{bmatrix} + [4 \ 4] \begin{bmatrix} 3 \\ 1 \end{bmatrix} - \frac{9}{8} [4 \ 4] \begin{bmatrix} 4 \\ 4 \end{bmatrix} = -3$$

(b) Which class does the SVM predict for the data point $x_1 = 5, x_2 = 2$?

$$f(\vec{x}) = b + \sum_{i=1}^{N} a_n y_i \vec{x}_m \vec{x}_i = -3 - [5 \ 2] \begin{bmatrix} 1 \\ 3 \end{bmatrix} - [5 \ 2] \begin{bmatrix} 3 \\ 1 \end{bmatrix} + \frac{9}{8} [5 \ 2] \begin{bmatrix} 4 \\ 4 \end{bmatrix} = \frac{1}{2}$$

Since the sign is positive, we predict class +1.
Decision boundary and support vectors.
Outline

1. Linearly Separable Case
2. Kernel Functions
3. Allowing Errors
4. SVM’s in R
Allowing Errors

- So far we assumed that the training data points are linearly separable in feature space $\phi(\vec{x})$.
- Resulting SVM gives exact separation of training data in original input space $\vec{x}$, with non-linear decision boundary.
- Class distributions typically overlap, in which case exact separation of the training data leads to poor generalization (overfitting).
Allowing Errors

- Data points are allowed to be on the wrong side of the margin boundary, but with a penalty that increases with the distance from that boundary.
- For convenience we make this penalty a linear function of the distance to the margin boundary.
- Introduce slack variables $\xi_i \geq 0$ with one slack variable for each training data point.
- Support vectors lie on the margin boundary, or on the wrong side of the margin boundary.
Definition of Slack Variables

We define $\xi_i = 0$ for data points that are on the inside of the correct margin boundary and $\xi_i = |y_i - f(\vec{x}_i)|$ for all other data points.
New objective function

Our goal is to maximize the margin while softly penalizing points that lie on the wrong side of the margin boundary. We therefore minimize

$$\frac{1}{2} |\vec{w}|^2 + C \sum_{i=1}^{N} \xi_i$$

where the parameter $C > 0$ controls the trade-off between the slack variable penalty and the margin.
Outline

1. Linearly Separable Case
2. Kernel Functions
3. Allowing Errors
4. SVM’s in R
Example: Conn’s syndrome

Two possible causes:

(a) Benign tumor (adenoma) of the adrenal cortex.
(b) More diffuse affection of the adrenal glands (bilateral hyperplasia).

Pre-operative diagnosis on basis of

1. Sodium concentration (mmol/l)
2. CO₂ concentration (mmol/l)
Conn’s syndrome: the data

\[a=1, \ b=0\]

<table>
<thead>
<tr>
<th></th>
<th>sodium</th>
<th>co2</th>
<th>cause</th>
<th></th>
<th>sodium</th>
<th>co2</th>
<th>cause</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>140.6</td>
<td>30.3</td>
<td>0</td>
<td>16</td>
<td>139.0</td>
<td>31.4</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>143.0</td>
<td>27.1</td>
<td>0</td>
<td>17</td>
<td>144.8</td>
<td>33.5</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>140.0</td>
<td>27.0</td>
<td>0</td>
<td>18</td>
<td>145.7</td>
<td>27.4</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>146.0</td>
<td>33.0</td>
<td>0</td>
<td>19</td>
<td>144.0</td>
<td>33.0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>138.7</td>
<td>24.1</td>
<td>0</td>
<td>20</td>
<td>143.5</td>
<td>27.5</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>143.7</td>
<td>28.0</td>
<td>0</td>
<td>21</td>
<td>140.3</td>
<td>23.4</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>137.3</td>
<td>29.6</td>
<td>0</td>
<td>22</td>
<td>141.2</td>
<td>25.8</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>141.0</td>
<td>30.0</td>
<td>0</td>
<td>23</td>
<td>142.0</td>
<td>22.0</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>143.8</td>
<td>32.2</td>
<td>0</td>
<td>24</td>
<td>143.5</td>
<td>27.8</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>144.6</td>
<td>29.5</td>
<td>0</td>
<td>25</td>
<td>139.7</td>
<td>28.0</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>139.5</td>
<td>26.0</td>
<td>0</td>
<td>26</td>
<td>141.1</td>
<td>25.0</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>144.0</td>
<td>33.7</td>
<td>0</td>
<td>27</td>
<td>141.0</td>
<td>26.0</td>
<td>1</td>
</tr>
<tr>
<td>13</td>
<td>145.0</td>
<td>33.0</td>
<td>0</td>
<td>28</td>
<td>140.5</td>
<td>27.0</td>
<td>1</td>
</tr>
<tr>
<td>14</td>
<td>140.2</td>
<td>29.1</td>
<td>0</td>
<td>29</td>
<td>140.0</td>
<td>26.0</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>144.7</td>
<td>27.4</td>
<td>0</td>
<td>30</td>
<td>140.0</td>
<td>25.6</td>
<td>1</td>
</tr>
</tbody>
</table>
Conn’s Syndrome: Plot of Data

Support vector machines
How to in R: linear kernel

```r
> conn.svm.lin <- svm(cause ~ sodium + co2,
data=conn.dat,kernel="linear")
> plot(conn.svm.lin,conn.dat)
> conn.svm.lin.predict <- predict(conn.svm.lin,conn.dat[,1:2])
> table(conn.dat[,3],conn.svm.lin.predict)

conn.svm.lin.predict
   0 1
  0 17 3
  1  2  8
```
Conn’s syndrome: linear kernel

Red: class a. Black: class b. $x =$ support vector.
Note that axes have switched!
How to in R: radial kernel with $C = 1$

```r
> conn.svm.rad <- svm(cause ~ sodium + co2, data=conn.dat)
> plot(conn.svm.rad,conn.dat)
> conn.svm.rad.predict <- predict(conn.svm.rad,conn.dat[,1:2])
> table(conn.dat[,3],conn.svm.rad.predict)

<table>
<thead>
<tr>
<th>conn.svm.rad.predict</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>17</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>8</td>
</tr>
</tbody>
</table>
```
Conn’s syndrome: radial kernel, $C = 1$
How to in R: radial kernel with $C = 100$

```r
> conn.svm.rad <- svm(cause ~ sodium + co2,
data=conn.dat,cost=100)
> plot(conn.svm.rad,conn.dat)
> conn.svm.rad.predict <-
predict(conn.svm.rad,conn.dat[,1:2])
> table(conn.dat[,3],conn.svm.rad.predict)

   conn.svm.rad.predict
     0 1
   0 19 1
   1 1 9
```
Conn’s syndrome: radial kernel, \( C = 100 \)
LIBSVM is available in package e1071 in R. It can also perform non-binary classification.

Non-binary classification is performed (automatically) as follows:

- It trains $|C|(|C| - 1)/2$ binary SVM’s on all possible pairs of classes.
- To classify a new point, it is classified by every binary SVM, and the class with the highest number of votes is predicted.
Exercise SVM’s

Circles: \( y = -1 \); Crosses: \( y = +1 \)

(a) Give the linear SVM decision boundary. Scale coefficients so that \( y(w_1 x_1 + w_2 x_2 + b) = 1 \) for the support vectors.

(b) Give the Rocchio decision boundary.