SVM
Support Vector Machine

Supervised Learning - Classification

Ricco Rakotomalala
Université Lumière Lyon 2
Outline

1. Binary classification – Linear classifier
2. Maximize the margin (I) – Primal form
3. Maximize the margin (II) – Dual form
4. Noisy labels – Soft Margin
5. Nonlinear classification – Kernel trick
6. Estimating class membership probabilities
7. Feature selection
8. Extension to multiclass problem
9. SVM in practice – Tools and software
10. Conclusion – Pros and cons
11. References
Binary classification

LINEAR SVM
Binary classification
Data linearly separable

Supervised learning : \( Y = f(x_1,x_2,...,x_p ; \beta) \) for a binary problem i.e. \( Y \in \{+, -\} \) or \( Y \in \{+1, -1\} \)

<table>
<thead>
<tr>
<th>x1</th>
<th>x2</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>-1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>-1</td>
</tr>
<tr>
<td>6</td>
<td>9</td>
<td>-1</td>
</tr>
<tr>
<td>8</td>
<td>7</td>
<td>-1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>13</td>
<td>6</td>
<td>1</td>
</tr>
</tbody>
</table>

The aim is to find a hyperplane which enables to separate perfectly the “+” and “-”. The classifier comes in the form of a linear combination of the variables.

\[
f(x) = x^T \beta + \beta_0 = x_1 \beta_1 + x_2 \beta_2 + \cdots + \beta_0
\]
Once the "shape" of the decision boundary defined, we have to choose a solution among the infinite number of possible solutions.

Two keys issues always in the supervised learning framework:

1. Choosing the “Representation bias” or “hypothesis bias” → we define the shape of the separator
2. Choosing the search bias i.e. the way to select the best solution among all the possible solutions → it often boils down to set the objective function to optimize

Example: Linear Discriminant Analysis

The separating hyperplane is to halfway between the two conditional centroids within the meaning of the Mahalanobis distance.
Primal problem

MAXIMIZE THE MARGIN (I)
The optimal separating hyperplane separates the two classes and maximizes the distance to the closest point from either class (Vapnik, 1996) [HTF, page 132]

- Distance from any point $x$ with the boundary (see projection)
  \[ d = \frac{|x^T \beta + \beta_0|}{\|\beta\|} \]

- The maximum margin is
  \[ \delta = \frac{2}{\|\beta\|} \]

- The instances from which rely the margins are “support vectors”. If we remove them from the sample, the optimal solution is modified.

- Several areas are defined in the representation space
  - $f(x) = 0$, we have the maximum margin hyperplane
  - $f(x) > 0$, the area of « + » instances
  - $f(x) < 0$, the area of « - » instances
  - $f(x) = +1$ or -1, these hyperplanes are the margins
Maximizing the margin
Mathematical formulation

\[
\begin{align*}
\min_{\beta, \beta_0} & \|\beta\| \\
\text{Subject to} & \quad y_i \times f(x_i) \geq 1, \ i = 1, \ldots, n
\end{align*}
\]

Maximize the margin is equivalent to minimize the norm of the vector of parameters \( \beta \)

\[
\max \frac{2}{\|\beta\|} \iff \min \|\beta\|
\]

- Norm:
  \[
  \|\beta\| = \sqrt{\beta_1^2 + \cdots + \beta_p^2}
  \]

- Constraints point out that all points are on the right side, at least they are on the hyperplane of support vectors.

- We have a problem of convex optimization (quadratic objective function, linear constraints). A global optimum exists.

- But there is no literal solution. It must pass through numerical optimization programs.

Note: There are often also writing

\[
\min_{\beta, \beta_0} \frac{1}{2} \|\beta\|^2
\]
Maximizing the margin
A toy example under EXCEL (!)

We use the SOLVER to solve the optimization problem.

<table>
<thead>
<tr>
<th></th>
<th>beta.1</th>
<th>beta.2</th>
<th>beta.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.667</td>
<td>-0.667</td>
<td>-1.667</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>n°</th>
<th>x1</th>
<th>x2</th>
<th>y</th>
<th>f(x)</th>
<th>f(x)*y</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>-1</td>
<td>-3</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>5</td>
<td>-1</td>
<td>-2.33333333</td>
<td>2.33333333</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>9</td>
<td>-1</td>
<td>-3.66666667</td>
<td>3.66666667</td>
</tr>
<tr>
<td>5</td>
<td>8</td>
<td>7</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>1</td>
<td>1</td>
<td>2.33333333</td>
<td>2.33333333</td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td>4</td>
<td>1</td>
<td>1.66666667</td>
<td>1.66666667</td>
</tr>
<tr>
<td>9</td>
<td>12</td>
<td>7</td>
<td>1</td>
<td>1.66666667</td>
<td>1.66666667</td>
</tr>
<tr>
<td>10</td>
<td>13</td>
<td>6</td>
<td>1</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

Norme.Beta 0.943

Objective cell: \(\|\beta\|\)

<table>
<thead>
<tr>
<th></th>
<th>x1</th>
<th>x2</th>
<th>y</th>
<th>f(x)</th>
<th>f(x)*y</th>
</tr>
</thead>
<tbody>
<tr>
<td>n°</td>
<td>x1</td>
<td>x2</td>
<td>y</td>
<td>f(x)</td>
<td>f(x)*y</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>-1</td>
<td>-3</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>5</td>
<td>-1</td>
<td>-2.33333333</td>
<td>2.33333333</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>9</td>
<td>-1</td>
<td>-3.66666667</td>
<td>3.66666667</td>
</tr>
<tr>
<td>5</td>
<td>8</td>
<td>7</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>1</td>
<td>1</td>
<td>2.33333333</td>
<td>2.33333333</td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td>4</td>
<td>1</td>
<td>1.66666667</td>
<td>1.66666667</td>
</tr>
<tr>
<td>9</td>
<td>12</td>
<td>7</td>
<td>1</td>
<td>1.66666667</td>
<td>1.66666667</td>
</tr>
<tr>
<td>10</td>
<td>13</td>
<td>6</td>
<td>1</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

Saturated constraints: 3 support vectors were found (n°2, 5 et 6)

\[0.667x_1 - 0.667x_2 - 1.667 = 0\]

\[0.667x_1 - 0.667x_2 - 1.667 + 1 = 0\]

\[0.667x_1 - 0.667x_2 - 1.667 - 1 = 0\]

\[x^T\beta + \beta_0 \pm 1 = 0\]
Primal problem

Comments

Rule assignment for the instance \( i^* \) based on the estimated coefficients \( \hat{\beta}_j \)

\[
\begin{align*}
f(x_i^*) \left\{ \begin{array}{cc}
\ge 0 & \Rightarrow \hat{y}_{i^*} = 1 \\
< 0 & \Rightarrow \hat{y}_{i^*} = -1
\end{array} \right.
\end{align*}
\]

Drawbacks of this primal form

(1) Algorithms for numerical optimization (quadratic prog.) are not operational when “p” is large (> a few hundred). This often happens when we handle real problems (e.g. text mining, image,...) (few examples, many descriptors)

(2) This primal form does not highlight the possibility of using "kernel" functions that enable to go beyond to the linear classifiers
Dual problem

MAXIMIZE THE MARGIN (II)
Dual problem
Lagrangian multiplier method

A convex optimization problem has a dual form by using the Lagrange multipliers.

The primal problem...

\[
\min_{\beta, \beta_0} \frac{1}{2}\|\beta\|^2
\]

s.c. \( y_i \times (x_i^T \beta + \beta_0) \geq 1, \forall i = 1, \ldots, n \)

...becomes under the dual form

\[
L_P(\beta, \beta_0, \alpha) = \frac{1}{2}\|\beta\|^2 - \sum_{i=1}^{n} \alpha_i \left[ y_i \times (x_i^T \beta + \beta_0) - 1 \right]
\]

Where \( \alpha_i \) are the Lagrange multipliers

By setting each partial derivate equal to zero

\[
\frac{\partial L}{\partial \beta} = \beta - \sum_{i=1}^{n} \alpha_i y_i x_i = 0
\]

\[
\frac{\partial L}{\partial \beta_0} = \sum_{i=1}^{n} \alpha_i y_i = 0
\]

\[
\frac{\partial L}{\partial \alpha_i} = -y_i \times (x_i^T \beta + \beta_0) + 1 \leq 0, \forall i
\]

The solution must satisfy the **Karush-Kuhn-Tucker (KKT) conditions**

\( \alpha_i [y_i \times (x_i^T \beta + \beta_0) - 1] = 0, \forall i \)
Dual problem
Optimization

By using information from the partial derivative of the Lagrangian, the results rely only on multipliers

\[
\max_{\alpha} L_D(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{i'=1}^{n} \alpha_i \alpha_{i'} y_i y_{i'} \langle x_i, x_{i'} \rangle
\]

Subject to

\[
\begin{align*}
\alpha_i & \geq 0, \forall i \\
\sum_{i=1}^{n} \alpha_i y_i &= 0
\end{align*}
\]

- \(\langle x_i, x_{i'} \rangle\) is the scalar product between the vectors of values for the instances \(i\) and \(i'\)

\[
\langle x_i, x_{i'} \rangle = \sum_{j=1}^{p} x_{ij} \times x_{i'j}
\]

- \(\alpha_i > 0\) define the important instances i.e. the support vectors

- Inevitably, there will be support vectors with different class labels, otherwise this condition cannot be met.
Example
Using Excel again

We use again the SOLVER to solve the optimization problem.

<table>
<thead>
<tr>
<th>n°</th>
<th>x1</th>
<th>x2</th>
<th>y</th>
<th>alpha</th>
<th>y*alpha</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>-1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>-1</td>
<td>0.33333</td>
<td>-0.3333</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>5</td>
<td>-1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>9</td>
<td>-1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>8</td>
<td>7</td>
<td>-1</td>
<td>0.11111</td>
<td>-0.1111</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>0.44444</td>
<td>0.44444</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>12</td>
<td>7</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>13</td>
<td>6</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Only the support vectors have a « weight » $\alpha_i > 0$ (n°2, 5 and 6)

<table>
<thead>
<tr>
<th>n°</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0.6</td>
<td>0</td>
<td>0</td>
<td>0.9</td>
<td>-1.6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1.4</td>
<td>-2.3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>-1.6</td>
<td>0</td>
<td>0</td>
<td>-2.3</td>
<td>5.1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Somme $0.88889$, $7.8E-16$

Somme $0.889$

The matrix of the scalar product $\langle x_i, x_i' \rangle$ is called Gram matrix

$$\sum_{i=1}^{n} \sum_{i'=1}^{n} \alpha_i \alpha_{i'} y_i y_{i'} \langle x_i, x_{i'} \rangle = 0.889$$

Only the support vectors have a « weight » $\alpha_i > 0$ (n°2, 5 and 6)

Recall that the margin is equal to $\delta = \frac{2}{\|\beta\|}$

Objective function $L_D(\alpha)$

$$\sum_{i=1}^{n} \alpha_i y_i = 0$$
From the support vectors to the hyperplane coefficients (I)
Computing $\beta$ from the $\alpha$ (Lagrange multipliers)

From the partial derivative of the Lagrangian with respect to $\beta$

$$\frac{\partial L}{\partial \beta} = \beta - \sum_{i=1}^{n} \alpha_i y_i x_i = 0 \Rightarrow \beta = \sum_{i=1}^{n} \alpha_i y_i x_i$$

Only the support vectors points are involved in the calculation of the coefficients, since they are the only ones for which ($\alpha_i > 0$)

Form KKT conditions, computed from any support vector ($\alpha_i$ must be $> 0$), we can obtain $\beta_0$

$$\alpha_i [ y_i \times (x_i^T \beta + \beta_0) - 1] = 0$$

$$\Rightarrow \beta_0 = \frac{1 - y_i x_i^T \beta}{y_i}$$

Note: Because $y_i \in \{-1, +1\}$, we can write also:

$$\beta_0 = y_i - x_i^T \beta$$

Since the primal and dual expressions are two facets of the same problem, one must be able to pass from one to the other.
From the support vectors to the hyperplane coefficients (II)

Numerical example

For $\beta_1$, only the variable $X_1$ participates in the calculations

$$\beta_1 = 0.333 \times (-1) \times 2 + 0.111 \times (-1) \times 8 + 0.444 \times (1) \times 5 = 0.6667$$

We use the support vector n°2

$$\beta_0 = \frac{1 - y_i x_i^T \beta}{y_i} = \frac{1 - (-1) \times [0.667 \times 2 + (-0.667) \times 1]}{(-1)} = -1.6667$$

The result is the same whatever the support vector used.
Classification of an unseen instance (I)
Using the support vectors

The classification function can be written based on the coefficients $\beta$ or the Lagrange multipliers $\alpha$

$$f(x) = x^T \beta + \beta_0$$

$$= \sum_{i=1}^{n} \alpha_i y_i \langle x_i, x \rangle + \beta_0$$

$$= \sum_{i \in S} \alpha_i y_i \langle x_i, x \rangle + \beta_0$$

S is the set of support vectors.

There are the only ones which have a weight ($\alpha_i > 0$)

Only the support vectors participate in the classification process!

We have a kind of nearest neighbors algorithm where only the instances corresponding to the support vectors participate to the classification process. These instances are weighted ($\alpha_i$)

The intercept $\beta_0$ can be obtained from the KKT conditions applied on the support vectors (see previous pages)
Classification of an unseen instance (II)
Numerical example

For the classification of the instance n°1

\[ f(x) = \sum_{i \in S} \alpha_i y_i \langle x_i, x \rangle + \beta_0 \]

\[ = 0.333 \times (-1) \times (2 \times 1 + 1 \times 3) + 0.111 \times (-1) \times (8 \times 1 + 7 \times 3) + 0.444 \times (1) \times (5 \times 1 + 1 \times 3) + (-1.667) \]

\[ = -3.0 \]

<table>
<thead>
<tr>
<th>n°</th>
<th>x1</th>
<th>x2</th>
<th>y</th>
<th>f(x)</th>
<th>prediction</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>-1</td>
<td>-3.000</td>
<td>-1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>-1</td>
<td>-1.000</td>
<td>-1</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>5</td>
<td>-1</td>
<td>-2.333</td>
<td>-1</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>9</td>
<td>-1</td>
<td>-3.667</td>
<td>-1</td>
</tr>
<tr>
<td>5</td>
<td>8</td>
<td>7</td>
<td>-1</td>
<td>-1.000</td>
<td>-1</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1.000</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>1</td>
<td>1</td>
<td>2.333</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td>4</td>
<td>1</td>
<td>1.667</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>12</td>
<td>7</td>
<td>1</td>
<td>1.667</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>13</td>
<td>6</td>
<td>1</td>
<td>3.000</td>
<td>1</td>
</tr>
</tbody>
</table>

Utilization of the 3 support vectors.
Dual form
Comments

1. This formulation is completely consistent with the primal form.

2. It highlights the part of the support vectors with the weights $\alpha_i$.

3. It highlights also the importance of the scalar product $\langle x_i, x_j \rangle$ during the calculations (Gram matrix).

4. Facing the high dimensionality ("p" is very high, e.g. text mining), this formulation makes calculations tractable for optimization techniques.
Noisy dataset labels

SOFT MARGIN
“Slack variables”

Using the slack variables $\xi_i$ to handle misclassified instances

In real problems, a perfect classification is not feasible. Some instances are in the wrong side of the margins.

- $\xi$ is a vector of size n
- $\xi_i \geq 0$ marks the misclassified instances
- $\xi_i = 0$, the instance is in the right side of the margin
- $\xi_i < 1$, the instance is in the right side of the maximum margin hyperplane, but it exceeds its margin
- $\xi_i > 1$, the instance is misclassified i.e. it is in the wrong side of the maximum margin hyperplane
Reformulation of the problem
Introduction of the cost parameter “C”

We should penalize the error, more or less strongly depending on whether you want a model that more or less fits to the training data.

Primal form

\[
\min_{\beta, \beta_0, \xi_i} \frac{1}{2} \|\beta\|^2 + C \sum_{i=1}^{n} \xi_i
\]

\[s.c.
\]

\[y_i \times (x_i^T \beta + \beta_0) \geq 1 - \xi_i, \forall i = 1, \ldots, n
\]

\[\xi_i \geq 0, \forall i
\]

The tolerance for errors is more or less accentuated with the \( C \) parameter ("cost" parameter)

\[\rightarrow \text{C is too high: overfitting}
\]

\[\rightarrow \text{C is too low: underfitting}
\]

Dual form

\[
\max_{\alpha} L_D(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{i'=1}^{n} \alpha_i \alpha_i' y_i y_{i'} \langle x_i, x_{i'} \rangle
\]

\[s.c.
\]

\[\sum_{i=1}^{n} \alpha_i y_i = 0
\]

\[0 \leq \alpha_i \leq C, \forall i
\]
Soft-margin - An example

Primal form

• Minimization of the objective function w.r.t. $\beta$ and $\xi$
• $C$ is a parameter, we set $C = 5$

The value of $C$ is an important issue in practice
Kernel trick

NONLINEAR CLASSIFICATION
Transformed feature space
Feature construction

By performing appropriate transformations of variables, we can make linearly separable a problem which was not linearly separable in the original representation space.

But multiplying concretely intermediate variables in the database is expensive, without any assurance that we obtain an efficient transformation.
Kernel functions
Applied to scalar product

The dot product between vectors has an important place in the calculations (dual form). SVM can take advantage of the "kernel" functions.

Let a transformation function $\phi(x)$ of initial variables

\[ x = (x_1, x_2) \rightarrow \phi(x) = (x_1^2, \sqrt{2}x_1x_2, x_2^2) \]

With the dual form, to optimize the Lagrangian, we calculate the scalar product $<\phi(x_i), \phi(x_{i'})>$ for each pair of instances $(i, i')$

We should handle 3 variables instead of 2, the calculations are more expensive, not to mention the storage of additional variables.

We can find a function $K(.)$, called Kernel Function, such as

\[ K(x_i, x_{i'}) = \langle \phi(x_i), \phi(x_{i'}) \rangle \]

The main consequence is that we simply calculate the scalar product $<x_i, x_{i'}>$, and we transform the result with the Kernel function.

We handle only the 2 initial variables for calculations. But the algorithm fits the classifier in a 3 dimensional space!
**Polynomial kernel**

**Examples**

Dot product between two instances (vectors) \( u \) and \( v \) with the following values

\[
\begin{align*}
\mathbf{u} &= (4, 7) \\
\mathbf{v} &= (2, 5)
\end{align*}
\]

\[
\langle \mathbf{u}, \mathbf{v} \rangle = 4 \times 2 + 7 \times 5 = 43
\]

Transformation (1)

\[
\phi(x) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)
\]

\[
\begin{align*}
\phi(u) &= (16, 39.6, 49) \\
\phi(v) &= (4, 14.1, 25)
\end{align*}
\]

\[
\langle \phi(u), \phi(v) \rangle = 1849
\]

Corresponding function (1)

\[
K_1(u, v) = \left( \langle u, v \rangle \right)^2 = 43^2 = 1849
\]

Transformation (2)

\[
\phi(x) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2)
\]

\[
\begin{align*}
\phi(u) &= (1, 5.7, 9.9, 16, 49, 39.6) \\
\phi(v) &= (1, 2.8, 7.1, 4, 25, 14.1)
\end{align*}
\]

\[
\langle \phi(u), \phi(v) \rangle = 1936
\]

Corresponding function (2)

\[
K_2(u, v) = \left( 1 + \langle u, v \rangle \right)^2 = (1 + 43)^2 = 1936
\]

The results are equivalent. With \( K(\cdot) \), we work in a higher dimensional space without having to explicitly create the variables.

We work in a 5-dimensional space in this configuration.
Dual form
Including the kernel function $K()$

$$\max_{\alpha} L_D(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{i'=1}^{n} \alpha_i \alpha_i' y_i y_i' K(x_i, x_{i'})$$

subject to

$$\sum_{i=1}^{n} \alpha_i y_i = 0$$

$$0 \leq \alpha_i \leq C, \forall i$$

It is no longer possible to obtain an explicit classification function, we must use the support vectors in order to assign a class to unseen instances i.e. we must store them (values and weights) for the deployment (see PMML)

$$f(x) = \sum_{i' \in S} \alpha_{i'} y_{i'} K(x_{i'}, x) + \beta_0$$

$\beta_0$ can be obtained from the Karush-Kuhn-Tucker (KKT) conditions, but by using the kernel functions (see page 12 and following)
Some kernel functions

The most popular functions in tools (e.g. Scikit-learn package for Python - SVC)

- **Polynomial**
  \[ K(u, v) = (\text{coef0} + \langle u, v \rangle)^{\text{degree}} \]
  - coef0 = 0 and degree = 1, we have the “linear” kernel

- **Gaussian radial basis function (RBF)**
  \[ K(u, v) = \exp\left(-\gamma \times \|u - v\|^2\right) \]
  - if it is not specified, the tools set by default (p: number of variables)
  \[ \gamma = \frac{1}{p} \]

- **Hyperbolic tangent**
  \[ K(u, v) = \tanh\left(\gamma \times \langle u, v \rangle + \text{coef0}\right) \]

Setting the right value of the parameters is the key issue, including the “cost parameter” C.

There is a bit of polysemy in the parameters, but they have been popularized by the famous **LIBSVM** package, included in several data mining tools (Scikit-Learn - Python, e1071 - R, Tanagra, etc.)
Scores and probabilities

CLASS MEMBERSHIP PROBABILITIES
Class membership probability

The output of the classification function $f(x)$ enables to assign a class to an instance.

\[
\begin{align*}
&f(x) \begin{cases} 
\geq 0 \Rightarrow \hat{y} = 1 \\
< 0 \Rightarrow \hat{y} = -1 
\end{cases}
\end{align*}
\]

The two points are assigned to the "positive" class, but one is more positive than the other!

We need an indication about the credibility of the response.

$|f(x)|$ is already a good indication. It allows to rank individuals according to their level of "positivity" (e.g. scoring, targeting, etc.)

But...

In many areas, we need an estimation of the class membership probability (e.g. interpretation, combination with a cost matrix, comparison with the outputs of other methods, etc.)
Platt scaling
Maximum likelihood estimation

We use a sigmoid function in order to map \( f(x) \) in the interval \([0, 1]\)

\[
P(Y = 1/ x) = \frac{1}{1 + \exp[-f(x)]}
\]

We can develop a more sophisticated solution by using a parameterized expression and estimate the values of the coefficients by maximum likelihood estimation

\[
P(Y = 1/ x) = \frac{1}{1 + \exp[-(a \times f(x) + b)]}
\]

A logistic regression program can estimate easily the values of “a” and “b”
Platt scaling

An example

Let us see our first toy example (Page 9)

\[
P(Y = 1/ x) = \frac{1}{1 + \exp[-(1.32 \times f(x) + 0.02)]}
\]

<table>
<thead>
<tr>
<th>n°</th>
<th>x1</th>
<th>x2</th>
<th>y</th>
<th>f(x)</th>
<th>P(y=1/x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>-1</td>
<td>-3.000</td>
<td>0.019</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>-1</td>
<td>-1.000</td>
<td>0.214</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>5</td>
<td>-1</td>
<td>-2.333</td>
<td>0.045</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>9</td>
<td>-1</td>
<td>-3.667</td>
<td>0.008</td>
</tr>
<tr>
<td>5</td>
<td>8</td>
<td>7</td>
<td>-1</td>
<td>-1.000</td>
<td>0.214</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1.000</td>
<td>0.792</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>1</td>
<td>1</td>
<td>2.333</td>
<td>0.957</td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td>4</td>
<td>1</td>
<td>1.667</td>
<td>0.902</td>
</tr>
<tr>
<td>9</td>
<td>12</td>
<td>7</td>
<td>1</td>
<td>1.667</td>
<td>0.902</td>
</tr>
<tr>
<td>10</td>
<td>13</td>
<td>6</td>
<td>1</td>
<td>3.000</td>
<td>0.982</td>
</tr>
</tbody>
</table>

\[
a = 1.32 \\
b = 0.02
\]

Classes membership probabilities are consistent with the position of the point and its distance from the frontier (maximum margin hyperplane).
Detecting the relevant variables

FEATURE SELECTION
Not embedded approaches
Filter and wrapper

Filter methods
The selection is done before and independently of the subsequent learning algorithm. Often based on the concept of "correlation" in the broad sense.

Pros: quickness, generic.
Cons: Not connected to the characteristics of the subsequent learning method, nothing says that the selected variables will be the right ones.

Wrapper methods
Use the classifier as a black box. Searching (e.g. forward, backward) of the best subset of variables which maximizes a performance criterion (e.g. cross-validation error rate).

Pros: selection directly related to a performance criterion.
Cons: very computationally intensive, risk of overfitting, do not use the internal characteristics of the learning algorithm (e.g. maximum margin for SVM).
Embedded approach
Maximum margin criterion

Measuring the contribution of the variable “$x_j$” in the classifier, without having to explicitly launching the learning process without “$x_j$”

Backward searching process to detect the best subset of relevant variables

2 main issues: measuring the contribution of the variable in the margin, organizing an algorithm from this criterion [ABE, page 192]

The variable $x_j$ is disabled by setting its values at zero

$$
\Delta^{(j)} \| \beta \|^2 = \sum_{i,i' \in S} \alpha_i \alpha_i' y_i y_i' \left( K(x_i, x_{i'}) - K(x_i^{(j)}, x_{i'}^{(j)}) \right)
$$

For a linear kernel, it is equivalent to test the significance of the coefficient $\beta_j$ (is it significantly different to zero)

1. Compute $\delta_0$, the initial margin with all the features
2. Find $j^*$ such as $\Delta^{(j^*)} \| \beta \|^2$ is minimum, and put it aside
3. Launch the learning without $x_{j^*}$, calculate the new margin $\delta$
4. If $\frac{\delta_0 - \delta}{\delta} < \varepsilon$ Then remove $x_{j^*}$, set $\delta_0 = \delta$ and go to 2, Else STOP the searching process.

2 important notes:

- The removing of a variable always reduce the margin. The question is: how significant is the reduction? If it is significant, we cannot remove the variable.
- $\varepsilon$ is a parameter of the algorithm ($\varepsilon$ high, we obtain less variable)
Extension to multiclass problems (K number of classes, K > 2)

MULTICLASS SVM
The SVM approach is formally defined for binary problems $Y \in \{+, -\}$, how to extend it (simply) to $K$ classes problems $Y \in \{y_1, ..., y_K\}$? The most popular approaches reduce the multiclass problem into multiple binary classification problems.

- $K$ binary classifiers which distinguish one of the classes $y_k$ from the rest (one vs. rest or one vs. all) ($Y' \in \{y_k=+1, y_{(k)}=-1\}$).
- We obtain $K$ classification functions $f_k(x)$.
- For the prediction, we pick the class which has the highest score i.e. $\hat{y} = \arg\max_k f_k(x)$

This strategy is consistent with the maximum a posteriori (MAP) scheme.

- **Properties**: $K$ learning processes to perform on the dataset
- **Pros**: simplicity
- **Cons**: we can artificially insert an imbalance of the classes in the construction of individual models. If the scores are not-well calibrated, comparisons of the output of the classification functions are biased.
Multiclass SVM
One vs. one (pairwise) approach

- Building $K(K-1)/2$ classifiers which distinguish every pair of classes ($Y' \in \{y_k=+1, y_j=-1\}$). We obtain $K(K-1)/2$ classification functions $f_{k,j}(x)$.
- In prediction, we predict the class using a voting system i.e. the one which has the maximum number of wins.

$$D_k(x) = \sum_{j \neq k, j=1}^{K} \text{sign}[f_{k,j}(x)]$$

$$\hat{y} = \arg \max_k D_k(x)$$

- **Note**: When two classes or more have the same number of votes, we use the sum of the scores $f_{k,j}(x)$ and we select the one corresponds to the max.
- **Properties**: the approach is computationally intensive, but each classifier is learned on the subset of the whole dataset.
- **Pros**: less problem of imbalance of the classes, the scores are better calibrated.
- **Cons**: computing time when $K$ increases (e.g. $K = 10 \Rightarrow 45$ classifiers to build).
Tools, packages, settings (Python, R and Tanagra)

SVM IN PRACTICE
#importing the training set
import pandas
dtrain = pandas.read_table("ionosphere-train.txt",sep="\t",header=0,decimal=".")
print(dtrain.shape)
y_app = dtrain.as_matrix()[:,32]
X_app = dtrain.as_matrix()[:,0:32]

#importing the module
from sklearn.svm import SVC
svm = SVC()  # instanciation de l’objet

#displaying the settings (default kernel “rbf”)
#the variables are not standardized (scale)
print(svm)

#learning process
svm.fit(X_app,y_app)

#importing the test set
dtest = pandas.read_table("ionosphere-test.txt",sep="\t",header=0,decimal=".")
print(dtest.shape)
y_test = dtest.as_matrix()[:,32]
X_test = dtest.as_matrix()[:,0:32]

#prediction on the test set
y_pred = svm.predict(X_test)

#measuring the test error rate: 0.07
from sklearn import metrics
err = 1.0 - metrics.accuracy_score(y_test,y_pred)
print(err)
Scikit-learn provides a mechanism for searching the optimal parameters in a cross-validation process. The test sample is not used here, thus we can use it in order to estimate the generalization error rate.

```python
# grid search tool
from sklearn.grid_search import GridSearchCV

# parameters to evaluate - modifying the kernel and the 'cost parameter'
parametres = {
    "kernel": ['linear', 'poly', 'rbf', 'sigmoid'],
    "C": [0.1, 0.5, 1.0, 2.0, 10.0]
}

# the classifier to use
svmc = SVC()

# creating the object
grille = GridSearchCV(estimator=svmc, param_grid=parametres, scoring="accuracy")

# launching the exploring
resultats = grille.fit(X_app, y_app)

# best settings: {‘kernel’ : ‘rbf’, ‘C’ : 10.0}
print(resultats.best_params_)

# prediction with the best classifier
ypredc = resultats.predict(X_test)

# error rate on the test set = 0.045 (!)
err_best = 1.0 - metrics.accuracy_score(y_test, ypredc)
print(err_best)
```
e1071 – svm() from **LIBSVM**

```r
# importing the learning set
dtrain <- read.table("ionosphere-train.txt",header=T,sep="\t")
dtest <- read.table("ionosphere-test.txt",header=T,sep="\t")

# package "e1071"
library(e1071)

# learning process
# the variables are automatically scaled
m1 <- svm(class ~ ., data = dtrain)

# displaying
print(m1)

# prediction
y1 <- predict(m1,newdata=dtest)

# confusion matrix and error rate = 0.04
mc1 <- table(dtest$class,y1)
err1 <- 1 - sum(diag(mc1))/sum(mc1)
print(err1)
```

Compared with scikit-learn, the variables are automatically standardized. This is preferable in most cases.
```r
# grid search using cross-validation
set.seed(1000) # to obtain the same results for each session
obj <- tune(svm, class ~ ., data = dtrain, ranges =
           list(kernel=c('linear','polynomial','radial', 'sigmoid'), cost =
                c(0.1,0.5,1.0,2.0,10)), tunecontrol = tune.control(sampling="cross"))

# displaying
print(obj)

# build the classifier with the new parameters
m2 <- svm(class ~ ., data = dtrain, kernel='radial', cost = 2)

# displaying
print(m2)

# prediction
y2 <- predict(m2,newdata=dtest)

# confusion matrix - test error rate = 0.035
mc2 <- table(dtest$class,y2)
err2 <- 1 - sum(diag(mc2))/sum(mc2)
print(err2)
```

*e1071* provides also, like scikit-learn, a tool for searching the “optimal” settings.
The SVM component provides an explicit model when using a linear kernel.

The data have been merged into a single file with an additional column indicating the type of the sample (train or test).

Test error rate = 0.165. Clearly, the 'linear' kernel is not suitable for these data.
C-SVC comes from the famous LIBSVM library

Les caractéristiques fournies se limitent au nombre de points supports (comme R)

With the same settings (rbf kernel, scale = FALSE, C = 10), we obtain exactly the same classifier than under Python (scikit-learn)
OVERVIEW

Pros and cons of SVM
### SVM — Pros and cons

**Pros**

- Ability to handle high dimensional dataset (high #variables)
- Robust even if the ratio “#observations / #variables” is inverted
- Efficient processing of the nonlinear problems with the kernel mechanism
- Nonparametric
- Robust against the outliers (controlled with the parameter $C$)
- #support vector provides a good indication about the complexity of the problem to handle
- Often effective compared with other approaches
- The possibilities of parameters adjustments allow flexibility (e.g. linear vs. nonlinear, regularization, etc.)

**Cons**

- Identifying the optimal values of the parameters is not obvious (SVM may be highly sensitive to the parameters)
- Difficulty in processing large dataset (#instances)
- Problems when we deal with noisy data labels (proliferation of the number of support vectors)
- No explicit model when we use nonlinear kernel
- Interpretation of the classifier for nonlinear kernel. Difficulty in the identification of the influence of the descriptors.
- The handling of the multiclass problem remains an open issue
Popularized in the classification task, SVMs can be applied to other kind of problems:

• semi-supervised learning (partially labeled data)

• support vector regression (regression problem)

• support vector clustering (clustering problem)

Specific kernel functions are developed (text mining, image mining, speech recognition,...). They must be adapted to the notion of similarity between observations in the domain.

The SVM approach is very close to research, with often new developments and improvements. All these new variants are not always available in the usual tools.
REFERENCES
References


... and many course materials found on the web...

Tanagra Tutorial, « Implementing SVM on large dataset », July 2009; comparison of various tools (Tanagra, Orange, RapidMiner, Weka) on a high dimensional dataset (31809 descriptors).

Tanagra Tutorial, « SVM using the LIBSVM library », November 2008; using the LIBSVM library from Tanagra.

Tanagra Tutorial, « CVM and BVM from the LIBCVM library », July 2012; extension of LIBSVM, this library can handle large dataset (high number of instances).

Tanagra Tutorial, « Support Vector Regression », April 2009; SVM in the regression context under Tanagra and R (e1071).