Support Vector Machines (SVM) in bioinformatics

Day 1:
Introduction to SVM

Jean-Philippe Vert

Bioinformatics Center, Kyoto University, Japan
Jean-Philippe.Vert@mines.org

Human Genome Center, University of Tokyo, Japan, July 17-19, 2002.
3 days outline

- Day 1: Introduction to SVM
- Day 2: Applications in bioinformatics
- Day 3: Advanced topics and current research
Today’s outline

1. SVM: A brief overview (FAQ)
2. Simplest SVM: linear classifier for separable data
3. More useful SVM: linear classifiers for general data
4. Even more useful SVM: non-linear classifiers for general data
5. Remarks
Part 1

SVM: a brief overview (FAQ)
What is a SVM?

- a family of learning algorithm for classification of objects into two classes (works also for regression)

- Input: a training set

\[ S = \{(x_1, y_1), \ldots, (x_N, y_N)\} \]

of objects \( x_i \in \mathcal{X} \) and their known classes \( y_i \in \{-1, +1\} \).

- Output: a classifier \( f : \mathcal{X} \rightarrow \{-1, +1\} \) which predicts the class \( f(x) \) for any (new) object \( x \in \mathcal{X} \).
Examples of classification tasks (more tomorrow)

- **Optical character recognition**: $x$ is an image, $y$ a character.

- **Text classification**: $x$ is a text, $y$ is a category (topic, spam / non spam...)

- **Medical diagnosis**: $x$ is a set of features (age, sex, blood type, genome...), $y$ indicates the risk.

- **Protein secondary structure prediction**: $x$ is a string, $y$ is a secondary structure
Pattern recognition example
Are there other methods for classification?

- Bayesian classifier (based on maximum a posteriori probability)
- Fisher linear discriminant
- Neural networks
- Expert systems (rule-based)
- Decision tree
- ...
Why is it gaining popularity

- Good performance in real-world applications
- Computational efficiency (no local minimum, sparse representation...)
- Robust in high dimension (e.g., images, microarray data, texts)
- Sound theoretical foundations
Why is it so efficient?

- Still a research subject
- Always try to classify objects with large confidence, which prevent from overfitting
- No strong hypothesis on the data generation process (contrary to Bayesian approaches)
What is overfitting?

- There is always a trade-off between good classification of the training set, and good classification of future objects (generalization performance).

- Overfitting means fitting too much the training data, which degrades the generalization performance.

- Very important in large dimensions, or with complex non-linear classifiers.
Overfitting example
What is Vapnik’s Statistical Learning Theory

- The mathematical foundation of SVM
- Gives conditions for a learning algorithm to generalize well
- The “capacity” of the set of classifiers which can be learned must be controlled
Why is it relevant for bioinformatics?

- Classification problems are very common (structure, function, localization prediction; analysis of microarray data; ...)
- Small training sets in high dimension is common
- Extensions of SVM to non-vector objects (strings, graphs...) is natural
Part 2

Simplest SVM: Linear SVM for separable training sets
We suppose that the object are finite-dimensional real vectors: $\mathcal{X} = \mathbb{R}^n$ and an object is:

$$\vec{x} = (x_1, \ldots, x_m).$$

$x_i$ can for example be a feature of a more general object.

Example: a protein sequence can be converted to a 20-dimensional vector by taking the amino-acid composition.
**Vectors and inner product**

**inner product:**

\[ \vec{x}.\vec{x}' = x_1 x'_1 + x_2 x'_2 \quad ( + \ldots + x_m x'_m ) \]  
\[ = ||\vec{x}||.||\vec{x}'||. \cos(\vec{x}, \vec{x}') \]
Linear classifier

Classification is based on the sign of the decision function:

\[ f_{\vec{w}, b}(\vec{x}) = \vec{w} \cdot \vec{x} + b \]
Linearly separable training set

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

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

\[ w \cdot x + b > 0 \]
Which one is the best?
Vapnik’s answer: LARGEST MARGIN
How to find the optimal hyperplane?

For a given linear classifier $f_{\mathbf{w}, b}$ consider the tube defined by the values $-1$ and $+1$ of the decision function:

$\mathbf{w} \cdot \mathbf{x} + b > +1$

$\mathbf{w} \cdot \mathbf{x} + b < -1$

$\mathbf{w} \cdot \mathbf{x} + b = +1$

$\mathbf{w} \cdot \mathbf{x} + b = -1$
The width of the tube is $1/||\vec{w}||$.

Indeed, the points $\vec{x}_1$ and $\vec{x}_2$ satisfy:

$$\begin{aligned}
\vec{w}.\vec{x}_1 + b &= 0, \\
\vec{w}.\vec{x}_2 + b &= 1.
\end{aligned}$$

By subtracting we get $\vec{w}.(\vec{x}_2 - \vec{x}_1) = 1$, and therefore:

$$\gamma = ||\vec{x}_2 - \vec{x}_1|| = \frac{1}{||\vec{w}||}.$$
All training points should be on the right side of the tube

For positive examples \( (y_i = 1) \) this means:

\[ \vec{w} \cdot \vec{x}_i + b \geq 1 \]

For negative examples \( (y_i = -1) \) this means:

\[ \vec{w} \cdot \vec{x}_i + b \leq -1 \]

Both cases are summarized as follows:

\[ \forall i = 1, \ldots, N, \quad y_i (\vec{w} \cdot \vec{x}_i + b) \geq 1 \]
Finding the optimal hyperplane

The optimal hyperplane is defined by the pair \((\vec{w}, b)\) which solves the following problem:

Minimize:

\[ ||\vec{w}||^2 \]

under the constraints:

\[ \forall i = 1, \ldots, N, \quad y_i (\vec{w} \cdot \vec{x}_i + b) - 1 \geq 0. \]

This is a classical quadratic program.
How to find the minimum of a convex function?

If \( h(u_1, \ldots, u_n) \) is a convex and differentiable function of \( n \) variable, then \( \vec{u}^* \) is a minimum if and only if:

\[
\nabla h(u^*) = \begin{pmatrix}
\frac{\partial h}{\partial u_1}(u^*) \\
\vdots \\
\frac{\partial h}{\partial u_1}(u^*)
\end{pmatrix} = \begin{pmatrix}
0 \\
\vdots \\
0
\end{pmatrix}
\]
How to find the minimum of a convex function with linear constraints?

Suppose that we want the minimum of $h(u)$ under the constraints:

$$g_i(\vec{u}) \geq 0, \quad i = 1, \ldots, N,$$

where each function $g_i(\vec{u})$ is affine.

We introduce one variable $\alpha_i$ for each constraint and consider the Lagrangian:

$$L(\vec{u}, \vec{\alpha}) = h(\vec{u}) - \sum_{i=1}^{N} \alpha_i g_i(\vec{u}).$$
Lagrangian method (ctd.)

For each $\tilde{\alpha}$ we can look for $\tilde{u}_\alpha$ which minimizes $L(\tilde{u}, \tilde{\alpha})$ (with no constraint), and note the dual function:

$$L(\tilde{\alpha}) = \min_{\tilde{u}} L(\tilde{u}, \tilde{\alpha}).$$

The dual variable $\tilde{\alpha}^*$ which maximizes $L(\tilde{\alpha})$ gives the solution of the primal minimization problem with constraint:

$$\tilde{u}^* = \tilde{u}_{\alpha^*}.$$
Application to optimal hyperplane

In order to minimize:
\[ \frac{1}{2} \| \vec{w} \|^2 \]
under the constraints:
\[ \forall i = 1, \ldots, N, \quad y_i (\vec{w} \cdot \vec{x}_i + b) - 1 \geq 0. \]

we introduce one dual variable \( \alpha_i \) for each constraint, i.e., for each training point. The Lagrangian is:
\[ L(\vec{w}, b, \vec{\alpha}) = \frac{1}{2} \| \vec{w} \|^2 - \sum_{i=1}^{N} \alpha_i (y_i (\vec{w} \cdot \vec{x}_i + b) - 1). \]
Solving the dual problem

The dual problem is to find $\alpha^*$ maximize

$$L(\bar{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j \vec{x}_i \cdot \vec{x}_j,$$

under the (simple) constraints $\alpha_i \geq 0$ (for $i = 1, \ldots, N$), and

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

$\bar{\alpha}^*$ can be easily found using classical optimization softwares.
Recovering the optimal hyperplane

Once \( \vec{\alpha}^* \) is found, we recover \((\vec{w}^*, b^*)\) corresponding to the optimal hyperplane. \( \vec{w}^* \) is given by:

\[
\vec{w}^* = \sum_{i=1}^{N} \alpha_i \vec{x}_i,
\]

and the decision function is therefore:

\[
f^*(\vec{x}) = \vec{w}^* \cdot \vec{x} + b^* = \sum_{i=1}^{N} \alpha_i \vec{x}_i \cdot \vec{x} + b^*.
\]  

(3)
Interpretation: support vectors

$\alpha = 0$

$\alpha > 0$
Simplest SVM: conclusion

- Finds the optimal hyperplane, which corresponds to the largest margin
- Can be solved easily using a dual formulation
- The solution is sparse: the number of support vectors can be very small compared to the size of the training set
- Only support vectors are important for prediction of future points. All other points can be forgotten.
Part 3

More useful SVM: Linear SVM for general training sets
In general, training sets are not linearly separable
What goes wrong?

The dual problem, maximize

\[ L(\bar{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j \vec{x}_i \cdot \vec{x}_j, \]

under the constraints \( \alpha_i \geq 0 \) (for \( i = 1, \ldots, N \)), and

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

has no solution: the larger some \( \alpha_i \), the larger the function to maximize.
Forcing a solution

One solution is to limit the range of $\vec{\alpha}$, to be sure that one solution exists. For example, maximize

$$L(\vec{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j \vec{x}_i \cdot \vec{x}_j,$$

under the constraints:

$$\begin{cases} 
0 \leq \alpha_i \leq C, & \text{for } i = 1, \ldots, N \\
\sum_{i=1}^{N} \alpha_i y_i = 0.
\end{cases}$$
Interpretation

\[ \alpha = 0 \]
\[ 0 < \alpha < C \]
\[ \alpha = C \]
Remarks

- This formulation finds a trade-off between:
  - minimizing the training error
  - maximizing the margin

- Other formulations are possible to adapt SVM to general training sets.

- All properties of the separable case are conserved (support vectors, sparseness, computation efficiency...).
Part 4

General SVM: Non-linear classifiers for general training sets
Sometimes linear classifiers are not interesting
Solution: non-linear mapping to a feature space
Let $\Phi(\vec{x}) = (x_1^2, x_2^2)'$, $\vec{w} = (1, 1)'$ and $b = 1$. Then the decision function is:

$$f(\vec{x}) = x_1^2 + x_2^2 - R^2 = \vec{w} \cdot \Phi(\vec{x}) + b,$$
Kernel \((simple\ but\ important)\)

For a given mapping \(\Phi\) from the space of objects \(\mathcal{X}\) to some feature space, the kernel of two objects \(x\) and \(x'\) is the inner product of their images in the features space:

\[
\forall x, x' \in \mathcal{X}, \quad K(x, x') = \vec{\Phi}(x) \cdot \vec{\Phi}(x').
\]

**Example:** if \(\vec{\Phi}(\vec{x}) = (x_1^2, x_2^2)',\) then

\[
K(\vec{x}, \vec{x}') = \vec{\Phi}(\vec{x}) \cdot \vec{\Phi}(\vec{x}') = (x_1)^2(x_1')^2 + (x_2)^2(x_2')^2.
\]
Training a SVM in the feature space

Replace each $\vec{x}.\vec{x}'$ in the SVM algorithm by $K(x, x')$

The dual problem is to maximize

$$L(\vec{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j K(x_i, x_j),$$

under the constraints:

$$\begin{align*}
0 &\leq \alpha_i \leq C, \quad \text{for } i = 1, \ldots, N \\
\sum_{i=1}^{N} \alpha_i y_i &= 0.
\end{align*}$$
Predicting with a SVM in the feature space

The decision function becomes:

\[ f(x) = w^* \cdot \Phi(x) + b^* \]

\[ = \sum_{i=1}^{N} \alpha_i K(x_i, x) + b^*. \]  \hspace{1cm} (4)
The kernel trick

- The explicit computation of $\Phi(x)$ is not necessary. The kernel $K(x, x')$ is enough. SVM work implicitly in the feature space.

- It is sometimes possible to easily compute kernels which correspond to complex large-dimensional feature spaces.
Kernel example

For any vector $\vec{x} = (x_1, x_2)'$, consider the mapping:

$$\Phi(\vec{x}) = \left( x_1^2, x_2^2, \sqrt{2}x_1x_2, \sqrt{2}x_1, \sqrt{2}x_2, 1 \right)' .$$

The associated kernel is:

$$K(\vec{x}, \vec{x}') = \Phi(\vec{x}).\Phi(\vec{x}')$$

$$= (x_1x_1' + x_2x_2' + 1)^2$$

$$= (\vec{x}.\vec{x}' + 1)^2$$
Classical kernels for vectors

- **Polynomial:**
  \[ K(x, x') = (x.x' + 1)^d \]

- **Gaussian radial basis function**
  \[ K(x, x') = \exp \left( \frac{||x - x'||^2}{2\sigma^2} \right) \]

- **Sigmoid**
  \[ K(x, x') = \tanh(\kappa x.x' + \theta) \]
Example: classification with a Gaussian kernel

\[ f(\vec{x}) = \sum_{i=1}^{N} \alpha_i \exp\left( \frac{||\vec{x} - \vec{x}_i||^2}{2\sigma^2} \right) \]
Part 4

Conclusion (day 1)
Conclusion

- SVM is a simple but extremely powerful learning algorithm for binary classification

- The freedom to choose the kernel offers wonderful opportunities (see day 3: one can design kernels for non-vector objects such as strings, graphs...)


- Lecture notes (draft) on [my homepage](#)