Classification by Support Vector Machines

Florian Markowetz
Max-Planck-Institute for Molecular Genetics
– Computational Molecular Biology –
Berlin

Practical DNA Microarray Analysis 2003
Overview

I Large Margin Classifiers

II The Kernel Trick

III R package: e1071
Gene expression is a complex process we cannot describe explicitly. ⇒ try to learn patterns from examples.

Given: $\mathcal{X} = \{x_i, y_i\}_{i=1}^n$ training set patients you’ve already seen consisting of

- $x_i \in \mathbb{R}^g$ points expression profiles
- $y_i \in \{+1, -1\}$ labels 2 kinds of cancer

Goal: Learn a decision function that describes the data well.

$$f_\mathcal{X} : \mathbb{R}^g \mapsto \{+1, -1\}$$

Diagnosis = $f_\mathcal{X}(\text{new patient})$
Problems of learning

- too simple
- too complex
- tradeoff
- new patient
- negative example
- positive example
- new patient
Linear separation

Most easy case: data set is linearly separable.

We need only a very simple classifier:

\[ S = \{ x \mid \langle w, x \rangle + b = 0 \} \]

Choose \( w \) and \( b \) from the training set \( \mathcal{X} \).

Prediction: On which side of the hyperplane does the new point lie?

Decision function: 

\[ f_{\mathcal{X}}(x_{\text{new}}) = \text{sign} \left( \langle w, x_{\text{new}} \rangle + b \right) \]
Which hyperplane is the best?
Separate the training set with maximal margin.

Samples with positive label

Samples with negative label

Margin

Separating Hyperplane
Non-separable training sets

Use linear separation, but admit training errors.

Penalty of error: distance to hyperplane multiplied by error cost $C$. 
Maximizing the margin is a problem of constrained optimisation, which can be solved by Lagrange Method.

Each training point $x_i$ is described by a Lagrange multiplier $\alpha_i$:

$\alpha_i = 0 \quad \Rightarrow \quad x_i$ has no influence on the hyperplane

$\alpha_i > 0 \quad \Rightarrow \quad x_i$ determines the sep. hyperplane

These points are called Support Vectors. They lie nearest to the hyperplane.
Solution

Solution: \[ w = \sum_{i=1}^{\#SV} \alpha_i y_i x_i^{sv} \]

Diagnosis: \[ f(x_{new}) = \text{sign} \left( \sum_{i=1}^{\#SV} \alpha_i y_i \langle x_i^{sv}, x_{new} \rangle + b \right) \]

The decision function only depends on the Support Vectors.

They are the critical elements of the training set.

All other points could be removed without changing the solution.
What’s next?

I Large Margin Classifiers

II The Kernel Trick

III R package: e1071
Separation may be easier in higher dimensions

complex in low dimensions → simple in higher dimensions

Florian Markowetz, *Classification by SVM*, Practical DNA Microarray Analysis 2003
The kernel trick

Classification is easier in high dimensions.

In the construction of the maximal margin hyperplane, we have to evaluate high dimensional inner products of the form

$$\langle \Phi(x_1), \Phi(x_2) \rangle_H$$

where $\Phi : \mathcal{L} \rightarrow \mathcal{H}$ is the feature map from a low to a high dimensional space.

Problem: Computationally expensive!

Idea: do the feature map implicitly!
Kernel Mapping

Mercer Theorem:
Under some conditions on $\mathcal{K}$ there exists an inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ and a mapping $\Phi: \mathcal{L} \rightarrow \mathcal{H}$ such that

$$\langle \Phi(x_1), \Phi(x_2) \rangle_{\mathcal{H}} = \mathcal{K}(x_1, x_2)$$

Using this kernel the decision function becomes

$$f(x_{\text{new}}) = \text{sign} \left( \sum_{i=1}^{\#SV} \alpha_i y_i \mathcal{K}(x_i, x_{\text{new}}) + b \right)$$
Examples of Kernels

**linear** \( \mathcal{K}(x_1, x_2) = \langle x_1, x_2 \rangle \)

**polynomial** \( \mathcal{K}(x_1, x_2) = (\gamma \langle x_1, x_2 \rangle + c_0)^d \)

**radial basis function** \( \mathcal{K}(x_1, x_2) = \exp \left( -\gamma \| x_1 - x_2 \|^2 \right) \)
Parameters of SVM

Kernel Parameters

$\gamma$: width of rbf coeff. in polynomial ($= 1$)

$d$: degree of polynomial

$c_0$: additive constant in polynomial ($= 0$)

Error weight

$C$: influence of training errors
More than 2 classes: ONE-versus-ALL
ONE-versus-ONE
Literature on SVM

- http://www.kernel-machines.org

- Vladimir Vapnik. 
  _The comprehensive treatment of statistical learning theory, including a large amount of material on SVMs_

  _An overview of statistical learning theory, containing no proofs, but most of the crucial theorems and milestones of learning theory. With a detailed chapter on SVMs for pattern recognition and regression_

- Bernhard Schölkopf and Alex Smola. 
  _An introduction and overview over SVMs. A free sample of one third of the chapters (Introduction, Kernels, Loss Functions, Optimization, Learning Theory Part I, and Classification) is available on the book website._
What’s next?

I. Large Margin Classifiers

II. The Kernel Trick

III. R package: e1071
Misc Functions of the Department of Statistics (e1071), TU Wien

Functions for latent class analysis, short time Fourier transform, fuzzy clustering, support vector machines, shortest path computation, bagged clustering, ...

Source + Reference Manual:
The Comprehensive R Archive Network:

http://cran.r-project.org/
svm.model <- svm(x=data, y=labels, type="C-classification",
                 kernel="linear")

svm.model <- svm(x=data, y=labels, type="C-classification",
                 kernel="linear", cost="42")

svm.model <- svm(x=data, y=labels, type="C-classification",
                 kernel="polynomial", degree="2")

svm.model <- svm(x=data, y=labels, type="C-classification",
                 kernel="radial", gamma="0.002")
> svm.model <- svm(data, labels, type="C-classification",
                  kernel="linear", cross=10)
> svm.model

Call:
  svm.default(x = data, y = labels, type = "C-classification",
               kernel = "linear", cross = 10)

Parameters:
  SVM-Type:  C-classification
  SVM-Kernel:  linear
    cost:  1
    gamma:  0.0001402721

Number of Support Vectors:  42 ( 22 20 )
Number of Classes: 2

Levels: (as integer)
  1 -1

Rho:
  -0.2118043

10-fold cross-validation on training data:

Total Accuracy: 87.7551
Single Accuracies:
  75 80 100 80 100 100 80 80 100 80
predicted <- predict(svm.model, data)  # test on training set
sum(predicted != labels)              # count differences

table(true=labels, pred=predicted)     # confusion matrix

<table>
<thead>
<tr>
<th>pred</th>
<th>true</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>24</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Thank you!