Barbora Hladká                 Martin Holub

{Hladka | Holub}@ufal.mff.cuni.cz

Charles University,
Faculty of Mathematics and Physics,
Institute of Formal and Applied Linguistics
library(e1071), but there are also other libraries (kernlab, shogun ...)  
training: function svm()  
prediction: function predict()  
svm() can work in both classification and regression mode  
if target attribute (response variable) is categorical (factor) the engine switches to classification
model = svm(formula, data=, kernel=, cost=, cross=, ...)

- ?svm
- kernel defines the kernel used in training and prediction. The options are: linear, polynomial, radial basis and sigmoid, default = radial
- cost – cost of constraint violation (default: 1)
- cross – optional, with the value k the k-fold cross-validation is performed
<table>
<thead>
<tr>
<th>Kernel name</th>
<th>Formula</th>
<th>Learning parameters and their default values</th>
</tr>
</thead>
<tbody>
<tr>
<td>linear</td>
<td>$x_i \cdot x_j$</td>
<td></td>
</tr>
<tr>
<td>polynomial</td>
<td>$(\gamma x_i \cdot x_j + c)^d$</td>
<td>$\gamma$, gamma=$1/(data\ dimension)$ $c$, coef0=0 $d$, degree=3</td>
</tr>
<tr>
<td>radial</td>
<td>$\exp(-\gamma(</td>
<td></td>
</tr>
<tr>
<td>sigmoid</td>
<td>$\tanh(\gamma x_i \cdot x_j + c)$</td>
<td>$\gamma$, gamma=$1/(data\ dimension)$ $c$, coef0=0</td>
</tr>
</tbody>
</table>
SVM – non-linear kernel functions

• polynomial kernel
  – smaller degree can generalize better
  – higher degree can fit (only) training data better

• radial basis
  – very robust
  – you should try and use it when polynomial kernel is weak to fit your data
SVM Parameter tuning with `tune.svm`

- SVM is a more complicated method in comparison with the previous and usually requires parameter tuning!
- Parameter tuning can take a very long time on big data, use a reasonably smaller part is often recommended

```r
> model.tune = tune.svm(class ~ ., data=train.small,
  kernel = "radial",
  gamma = c(0.001, 0.005, 0.01, 0.015, 0.02),
  cost = c(0.5, 1, 5, 10))

> model.tune
Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
  gamma  cost
  0.01    1

- best performance: 0.739
Built-in cross-validation

K-fold cross-validation

- parameter cross

```r
> model.best <- svm(class ~ ., train.small,
>        kernel = "radial",
>        gamma = 0.01,
>        cost = 10,
>        cross = 10)

> model.best$accuracies
[1] 26.81 30.90 36.36 28.63 38.18 28.18 37.72 35.90 34.09 30.90

> model.best$tot.accuracy
[1] 32.77

> prediction.best <- predict(model.best, test, type="class")
> mean(prediction.best==test$class)
[1] 33.36
```
Class weighting

- **class.weights** parameter
  In case of asymmetric class sizes you may want to avoid possibly overproportional influence of bigger classes. Weights may be specified in a vector with named components, like:

  ```r
  m <- svm(x, y, class.weights = c(A = 0.3, B = 0.7))
  ```
General hints on practical use of `svm()`

- Note that SVMs may be very sensible to the proper choice of parameters, so always check a range of parameter combinations, at least on a reasonable subset of your data.
- Be careful with large datasets as training times may increase rather fast.
- C-classification with the RBF kernel (default) can often be a good choice because of its good general performance and the few number of parameters (only two: cost and gamma).
- When you use C-classification with the RBF kernel: try small and large values for cost first, then decide which are better for the data by cross-validation, and finally try several gamma values for the better cost.
Evaluation of multi-class classification task

<table>
<thead>
<tr>
<th>target class</th>
<th>True Positive</th>
<th>False Positive</th>
<th>False Negative</th>
<th>class weight</th>
<th>Precision</th>
<th>Recall</th>
<th>F1 score</th>
</tr>
</thead>
<tbody>
<tr>
<td>C₁</td>
<td>TP₁</td>
<td>FP₁</td>
<td>FN₁</td>
<td>w₁</td>
<td>P₁</td>
<td>R₁</td>
<td>F₁</td>
</tr>
<tr>
<td>C₂</td>
<td>TP₂</td>
<td>FP₂</td>
<td>FN₂</td>
<td>w₂</td>
<td>P₂</td>
<td>R₂</td>
<td>F₂</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Cₖ</td>
<td>TPₖ</td>
<td>FPₖ</td>
<td>FNₖ</td>
<td>wₖ</td>
<td>Pₖ</td>
<td>Rₖ</td>
<td>Fₖ</td>
</tr>
</tbody>
</table>

- class weight $w_i$ is the relative frequency of $C_i$ class in the data
- macro-averaged F1 score $= \sum_{i=1}^{k} F_i / k$
- weighted-averaged F1 score $= \sum_{i=1}^{k} w_i F_i / k$
Evaluation of multi-class classification task

- In general, if you are working with an imbalanced dataset where all classes are equally important, using the macro average would be a good choice.
- If you have an imbalanced dataset but want to assign greater contribution to classes with more examples in the dataset, then the weighted average is preferred.