libSVM

LING572

Fei Xia
Documentation


• The libSVM directory on Patas:
  /NLP_TOOLS/ml_tools/svm/libsvm/latest/
  – README
  – FAQ.html
  – svm-train, svm-predict, etc.

• More info:
  – [A practical guide to support vector classification](http://www.csie.ntu.edu.tw/~cjlin/libsvm/tutorial.html)
  – [LIBSVM : a library for support vector machines](http://www.csie.ntu.edu.tw/~cjlin/libsvm/)
Steps for using libSVM

• Define features in the input space (if use one of the pre-defined kernel functions)

• Scale the data before training/test

• Choose a kernel function

• Tune parameters using cross-validation
Main commands

• svm-scale: scaling the data

• svm-train: training

• svm-predict: decoding
Scaling the data

• To avoid features with larger variance to dominate those with smaller variance.

• Scale each feature to the range \([-1,+1]\) or \([0,1]\).
  – \([0,1]\) is faster than \([-1,1]\)
svm-scale

- svm-scale -l -1 -u 1 -s range_file training_data > training_data.scale

- svm-scale -r range_file test_data > test_data.scale

- Scale feature values to [-1, 1] or [0,1]

- No need to scale the data for Hw8.
svm-train

• svm-train [options] training_data model_file

• Options:
  -t [0-3]: kernel type
  -g gamma: used in polynomial, RBF, sigmoid
  -d degree: used in polynomial
  -r coef0: used in polynomial, sigmoid

• Type “svm-train” to see options
Kernel functions

-t kernel_type : set type of kernel function (default 2)

0: linear: \( u' \times v \)

1: polynomial: \((\gamma u' \times v + \text{coef0})^{\text{degree}}\)

2: RBF: \(\exp(-\gamma |u-v|^2)\)

3: sigmoid: \(\tanh(\gamma u' \times v + \text{coef0})\)
svm-predict

- `svm-predict test_data model_file output_file`

- `svm-predict` produces only the system prediction in `output_file`.

- You will implement your own decoder in Hw8.
The format of training/test data

• Sparse format: no need to include features with value zero.

• Mallet format:
  truelabel  f1: v1  f2: v2 ..... 

• libSVM format:
  truelabel_idx  feat_idx1:v1  feat_idx2:v2  .... 

  (feat_idx, v) is sorted according to feat_idx in ascending order.
  Ex:  1  20:1  23:0.5  34:-1  ...
When there are two classes
The format of the model file

svm_type  c_svc
kernel_type  rbf
gamma  0.5
nr_class  2
total_sv  535
rho  0.281122
label  0 1
nr_sv  272 263
SV

0.98836
0:1 1:1 2:1 3:1 4:1 5:1 ...

... This is weight for the support vector, which is equal to $\alpha_i y_i$.

This is a support vector with the format f1:v1 f2:v2 ...
Classifying an instance $x$

$$f(x) = \sum_i \alpha_i y_i K(x_i, x) - \rho$$

$$= \sum_i \text{weight}_i K(x_i, x) - \rho$$

where $y_i$ (i.e., $x_i$’s label) is $+1$ ("$c_0$") or $-1$ ("$c_1$”).

if $f(x) > 0$
then label it with $c_0$
else label it with $c_1$
## Notation differences

<table>
<thead>
<tr>
<th></th>
<th>In SVM paper</th>
<th>In libSVM</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Model</strong></td>
<td>$x_i, y_i, \alpha_i$</td>
<td>$weight_i, x_i$</td>
</tr>
<tr>
<td></td>
<td>$b$</td>
<td>$\rho$</td>
</tr>
<tr>
<td><strong>Prediction</strong></td>
<td>$\sum_i \alpha_i y_i K(x_i, x) + b$</td>
<td>$\sum_i weight_i K(x_i, x) - \rho$</td>
</tr>
<tr>
<td>Representing $y_i$ in training/test/output</td>
<td>+1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>-1</td>
<td>1</td>
</tr>
</tbody>
</table>
System output of svm-predict

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

## \( c_0 \)

## \( c_1 \)
Additional slides
When there are C classes
Handling a multi-class task

• All-pair

• Build a classifier for every \((c_m, c_n)\) pairs
  – There are \(C(C-1)/2\) classifiers

• The classifiers are stored in a compact format.
The format of the model file (when there are C>2 classes)

```
svm_type  c_svc
kernel_type  rbf
gamma  0.5
nr_class  3
total_sv  2698
rho  -0.0111642 -0.00216906 0.00951624
label 0 1 2
nr_sv  900 898 900
SV
0.98836  0.9975  0:1 1:1 2:1 3:1 4:1 5:1 ...
...
```
The rho array

It contains $C(C-1)/2$ elements, one per classifier

0 vs. 1, 0 vs. 2, ..., 0 vs. C-1,
1 vs. 2, 1 vs. 3, ..., 1 vs. C-1
2 vs. 3, ..., 2 vs. C-1
...
C-2 vs. C-1
The format of the SV line

Each line includes C-1 weights (i.e., $y_i \alpha_i$) followed by the vector.

$w_1 w_2 ... w_{C-1} f_1:v_1 f_2:v_2 ....$

Suppose the current vector belongs to the i-th class, the weights are ordered as follows:

0 vs. i  1 vs. i  2 vs i  ....  i-1 vs i
i vs. i+1  i vs i+2  i vs i+3  ....  i vs C-1

Ex1: i=0
0 vs. 1, 0 vs. 2, 0 vs. 3, ...., 0 vs. C-1

Ex2: i=4
0 vs 4, 1 vs 4, 2 vs. 4, 3 vs. 4, 4 vs. 5, 4 vs. 6, ...., 4 vs. C-1
Classifying an instance $x$

$\text{win}[m] = 0$ for every class $m$

For each classifier for $(m,n)$

$$f(x) = \sum_i \alpha_i y_i K(x_i, x) - \rho$$

$$= \sum_i \text{weight}_i K(x_i, x) - \rho$$

where $x_i$ is a training instance with label $c_m$ or $c_n$.

if $f(x) > 0$

then $\text{win}[m]++$

else $\text{win}[n]++$

$$\text{sysLabel} = \arg \max_m \text{win}[m]$$
To classify $x$ with a m-vs-n classifier ($m < n$):

$\rho$ is stored at what position?

For each $x_i$ belonging to $c_m$

- 0 vs. $m$, 1 vs. $m$, ..., $m-1$ vs. $m$,
- $m$ vs. $m+1$, $m$ vs. $m+2$, ..., $m$ vs. $n$, ...

the weight for m-vs-n is stored at position $n-1$

For each $x_i$ belonging to $c_n$

- 0 vs. $n$, 1 vs. $n$, 2 vs. $n$, ..., $m$ vs. $n$, ...

the weight for m-vs-n is stored at position $m$