

SeparableVSSVM: An Implementation of Version Space Support Vector Machines for Separable Data

Evgueni N. Smirnov

MICC-IKAT, Faculty of General Sciences
Maastricht University
Maastricht 6200 MD
The Netherlands

smirnov@cs.unimaas.nl

September 28, 2006

1. Introduction

Version Space Support Vector Machines (*VSSVMs*) form a new approach to reliable instance classification. They are an implementation of version spaces that uses support vector machines (*SVMs*). The nice feature of *VSSVMs* is that their volume grows with parameters *Gamma* (*Exponent*) and *C* of *SVMs*. In this way, instances with unreliable classifications are not classified; i.e., *VSSVMs* classify only those instances of which the classifications are reliable¹.

For more details please read:

Smirnov, E.N., Sprinkhuizen-Kuyper, I.G., Nalbantov, G.I., and Vanderlooy, S. (2006). *Version Space Support Vector Machines*. in Proceedings of the 17th European Conference on Artificial Intelligence (ECAI-2006), pp.809-811.

2. Implementation

VSSVMs have been implemented as a *Java* class for the *WEKA* machine-learning environment. The class name is `SeparableVSSVM`. The class extends the *WEKA* class `Classifier` and can be plugged in *WEKA* in a standard way. For more details please read:

Witten, I., and Frank, E., *Data Mining: Practical Machine Learning Tools and Techniques*, Morgan Kaufman, 2005.

The `SeparableVSSVM` class uses the *SMO* implementation of the *SVM* representing *VSSVMs*. This means that the `SeparableVSSVM` class can be used only in those *WEKA* installations that have the *SMO* class.

The `SeparableVSSVM` class implements *VSSVMs* for separable data only. The data are separable if the parameters *Gamma* (*Exponent*) and *C* of *SVM* (that represents *VSSVMs*) are

¹ Using version-space terminology: *VSSVMs* allow constructing version spaces that contain hypotheses of the target class or of its close approximations. In this way the unanimous-voting classification rule of version spaces does not misclassify instances; i.e., instance classifications with *VSSVMs* become reliable.

chosen s.t. the *SVM* hyperplane separates these data (i.e., there exists at least one hyperplane that is consistent with the data). This implies that the version space (*VSSVM*) in the hypothesis space of oriented hyperplanes is nonempty; i.e., the version space (*VSSVM*) can be used for classification.

The `SeparableVSSVM` class allows the parameters *Gamma* (*Exponent*) and *C* to be chosen manually or to be found by binary search. If the parameters are chosen s.t. no separability exists or the binary search ends without success, then the `SeparableVSSVM` class throws an `exception` indicating that the version space (*VSSVM*) is empty.

To start binary search for parameters *Gamma* (*Exponent*) and *C*, we assign `true` value to the option `ToDoSearch`². The search for the parameter *Gamma* (*Exponent*) is realized in the range $[lGE, rGE]$ where *lGE* is the value of the option `LeftBoundSearchGamma_Exp` and *rGE* is the value of the option `RightBoundSearchGamma_Exp`. We note that we search for the parameter *Gamma* if the option is `UserRBF` set to be true; otherwise, we search for the parameter *Exponent*. The search for the parameter *C* is realized in the range $[lC, rC]$ where *lC* is the value of the option `LeftBoundSearchC` and *rC* is the value of the option `RightBoundSearchC`. The found values of the parameters are recorded into a file. We can specify the name of the file by the option `file`. If the option is not specified the `SeparableVSSVM` class assign its own name of the file. It is:

```
<name_of_the_data>.GE=lGE-rGE_C=lC-rC.pol.exp or
```

```
<name_of_the_data>.GE=lGE-rGE_C=lC-rC.rbf.exp
```

depending on whether *polynomial* or *RBF* kernels are used³.

Once the parameter file is ready, we can use it for latter experiments. For that purpose, we assign `false` value to the option `ToDoSearch`. In this case the values of the parameters *Gamma* (*Exponent*) and *C*, will be read from the file, so no search will be preformed and the experiments become with one order faster!

To grow the volume of *VSSVMs* we increase the values of the parameters *Gamma* (*Exponent*) and *C* (Increasing the values of these parameters enrich the hypothesis space of *VSSVMs* s.t. the volume of *VSSVMs* grows.). For that purpose, we can add extra values to these parameters specified by the values of the options *UpdateGamma* (*UpdateExp*) and *UpdateC*. We note that the volume-extension approach can be applied for both cases: when the search for the parameters is on (very slow) and when the search for the parameters is off (relatively fast).

3. Options

The `SeparableVSSVM` class has two sets of options: parameters of *SVM* (*SMO* implementation from *WEKA*) and specific parameters of *VSSVMs*.

² We use here names of options that appear on the option window of the `SeparableVSSVM` classifier when it is used in the *WEKA Explorer* or *WEKA Experimenter*. For the names of the options in console windows please see the code of the `SeparableVSSVM` class.

³ For example, a file `labor.GE=0-10_C=10-55.rbf.exp` contains parameter values for the parameters *Gamma* and *C* that were found in the ranges $[0,10]$ and $[10, 55]$, respectively.

3.1 **SVM options.** *SVM* options start with key *SVM*. They are:

SVM_BuildLogisticModels -- Whether to fit logistic models to the outputs (for proper probability estimates).

SVM_CacheSize -- The size of the kernel cache (should be a prime number).

SVM_Epsilon -- The epsilon for round-off error (shouldn't be changed).

SVM_FeatureSpaceNormalization -- Whether feature-space normalization is performed (only available for non-linear polynomial kernels).

SVM_FilterType -- Determines how/if the data will be transformed.

SVM_LowerOrderTerms -- Whether lower order polynomials are also used (only available for non-linear polynomial kernels).

SVM_NumFolds -- The number of folds for cross-validation used to generate training data for logistic models (-1 means use training data).

SVM_RandomSeed -- Random number seed for the cross-validation AAAA.

SVM_ToleranceParameter -- The tolerance parameter (shouldn't be changed).

SVM_UseRBF -- Whether to use an RBF kernel instead of a polynomial one.

3.2 **VSSVM options.** *VSSVM* options start with key *VSSVM*. They are:

VSSVM_File -- Name of the parameter file.

VSSVM_LeftBoundSearchC -- The Left Bound for the binary search of the parameter *C*. If *VSSVM_LeftBoundSearchC* equals *VSSVM_RightBoundSearchC*, then no search for *C* is performed.

VSSVM_LeftBoundSearchGamma_Exp -- The Left Bound for the binary search of *Gamma* (*Exponent*). It is not recommended for *Exponent* (since monotonicity of the separability is not preserved). If *VSSVM_LeftBoundSearchGamma_Exp* equals *VSSVM_RightBoundSearchGamma_Exp*, then no search is performed.

VSSVM_PrecisionBinarySearchC -- The Precision for the Binary Search for the *C* search.

VSSVM_PrecisionBinarySearchGamma_Exp -- The Precision for the Binary Search for the *Gamma* (*Exponent*) search.

VSSVM_RightBoundSearchC -- The Right Bound for the binary search of *C*. If *VSSVM_RightBoundSearchC* equals *VSSVM_LeftBoundSearchC*, then no search for *C* is performed.

VSSVM_ToDoSearch -- Binary Value if `true` Binary Search for *Gamma* (*Exponent*) and *C* is performed.

VSSVM_UpdateC -- The update of *C*. The update is added to the current value of *C*. In this way the volume-extension approach is applied on *VSSVMs*.

VSSVM_UpdateExp -- The update of *Exponent*. The update is added to the current value of *Exponent*. In this way the volume-extension approach is applied on VSSVMs. Note that *Exponent* is not monotonic with the volume. So, the volume-extension approach cannot be realized with the *Exponent* update for all cases.

VSSVM_UpdateGamma -- The update of *Gamma*. The update is added to the current value of *Gamma*. In this way the volume-extension approach is applied on VSSVMs.

3.3 Use of the Parameters

The use of the parameters is straightforward. **If you want to start VSSVMs with certain value *g* of the parameter *Gamma* (*Exponent*), then set option `LeftBoundSearchGamma_Exp` equal to *g* and option `RightBoundSearchGamma_Exp` equal to *g*. If you want to start VSSVMs with certain value *c* of the parameter *C*, then set option `LeftBoundSearchC` equal to *c* and option `RightBoundSearchC` equal to *c*. Please note that when the range options for a parameter are equal no search takes place.**

4. Examples

We have to build a `SeparableVSSVM` classifier for the labor data using the *RBF* kernel and to evaluate this classifier using 10-fold cross validation. The parameters for parameters *Gamma* (*Exponent*) and *C* are unknown but we have borders for these parameters. We set the borders (see **Figure 1**) and then we set the option `ToDoSearch` to `true`. Then, we start *WEKA Explorer* to build a VSSVM for labor data using the *RBF* kernel. After 2 minutes the evaluation process finishes and we have the following print ⁴:

=== Summary ===

Correctly Classified Instances	40	70.1754 %
Incorrectly Classified Instances	2	3.5088 %
Kappa statistic	0.8833	
Mean absolute error	0.0351	
Root mean squared error	0.1873	
Relative absolute error	10.858 %	
Root relative squared error	47.712 %	
UnClassified Instances	15	26.3158 %
Total Number of Instances	57	

=== Detailed Accuracy By Class ===

TP Rate	FP Rate	Precision	Recall	F-Measure	Class
0.917	0.033	0.917	0.917	0.917	bad
0.967	0.083	0.967	0.967	0.967	good

=== Confusion Matrix ===

```

a b <-- classified as
11 1 | a = bad
1 29 | b = good

```

⁴ We note that evaluation statistics are provided only for classified instances. Therefore, please take into account the statistics *UnClassified Instances*.

After the first experiment we have a new parameter file that contains the values for the parameters *Gamma* (*Exponent*) and *C* for each folder. The file contains 11 lines. The first line contains the value of the *Gamma* parameter and the value of the *C* parameter when the `SeparableVSSVM` classifier is built on all the data. The $n+1$ line contains the value of the *Gamma* parameter and the value of the *C* parameter when the `SeparableVSSVM` classifier is built for n -th folder of the data.

```
labor.rbf.exp Print:
```

```
0.078125 15.1824951171875
0.078125 55.9234619140625
0.078125 10.528564453125
0.078125 5.4931640625
0.078125 6.7901611328125
0.078125 10.9100341796875
0.078125 14.190673828125
0.078125 5.2642822265625
0.078125 6.1798095703125
0.078125 12.20703125
0.078125 15.716552734375
```

We can use the file to experiment with growing the volume of *VSSVMs*. This experiment we can realized by increasing the parameter *C*. So, we start `SeparableVSSVM` with the option `updateC` equal to 1 without search (option `ToDoSearch` is `false`). The time for model evaluation is around 20 seconds and we get the following print:

```
=== Summary ===
```

```
Correctly Classified Instances    37      64.9123 %
Incorrectly Classified Instances    1      1.7544 %
Kappa statistic                   0.9343
Mean absolute error                0.0175
Root mean squared error            0.1325
Relative absolute error            6.0903 %
Root relative squared error       36.0334 %
UnClassified Instances           19      33.3333 %
Total Number of Instances         57
```

```
=== Detailed Accuracy By Class ===
```

```
TP Rate  FP Rate  Precision  Recall  F-Measure  Class
1        0.036    0.909     1       0.952     bad
0.964    0        1         0.964   0.982     good
```

```
=== Confusion Matrix ===
```

```
a b <-- classified as
10 0 | a = bad
1 27 | b = good
```

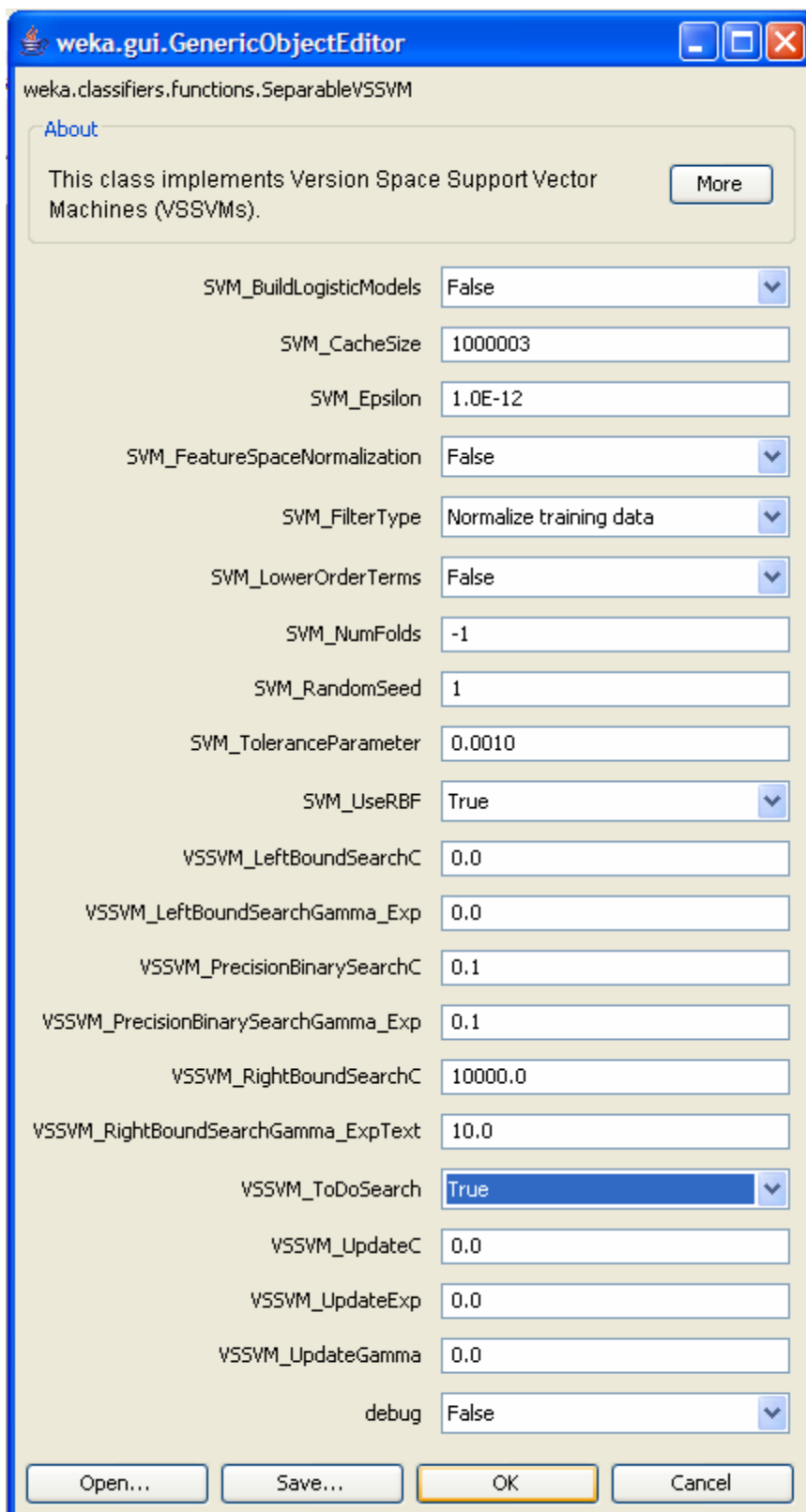


Figure 1 The option window of SeparableVSSVM for the *labor* data when we start search.

5. Use of the Parameter Files

Make sure that the values of the parameters for each experiment are saved in a file with unique name. Otherwise, all the values of the parameters will be saved in one file. If we will use this file to avoid search we will use actually the parameter values of the very first experiment.

6. Remarks

If you have any questions about *VSSVMs* and *SeparableVSSVM*, you can send an e-mail to smirnov@micc.unimaas.nl. We will be obliged if you can send us information where and how you use *VSSVMs*.