

## **PROXIMAL SUPPORT VECTOR MACHINE CLASSIFICATION ON SEISMIC DATA – PROGRAM **psvm3d****

### **Contents**

Overview .....	1
Theory .....	2
Computation flow chart.....	5
Step-by-step instruction on program <b>psvm3d</b> .....	5
Horizon definition.....	10
References .....	13

### **Overview**

Support vector machine (SVM) is a recent supervised machine learning technique that is widely used in text detection, image recognition and protein classification. In exploration geophysics, it can be used in seismic facies classification, petrophysics parameter estimation, and correlation of seismic attributes with engineering data. Proximal support vector machine (PSVM) is a variant of SVM, which has comparable classification performance to standard SVM but at considerable computational savings (Fung and Mangasarian, 2001, 2005; Mangasarian and Wild, 2006) that is critical when handling large 3D seismic surveys. This documentation provides an overview of the arithmetic of PSVM and step-by-step instructions on an AASPI implementation of PSVM for seismic data – **psvm3d**.

Comparing to the most popular artificial neural network (ANN) algorithms that are available in many commercial software, SVM and its variants benefit from the fact that they are based on convex optimization which is free of local minima (Shawe-Taylor and Cristianini, 2004), therefore providing a constant and robust classifier, once training samples and model parameters are determined. Such classifier can then generate stable, reproducible classification result (Bennett and Campbell, 2000). Also, SVM has fewer parameters to pick than ANNs and the number of kernel functions is automatically selected, which makes it easier to reach the optimal model (Bennett and Campbell, 2000). Some researchers have compared the capability of SVM with ANN in pressure-wave velocity prediction in mining geophysics (Verma et al., 2014) and other non-geophysics disciplines (Wong and Hsu, 2005; Balabin and Lomakina, 2011) and found SVM is superior in most cases.

### Theory

Because SVMs are originally developed to solve binary classification problems, the arithmetic we show here are the steps to generate a binary PSVM classifier. Strategy of extending binary PSVM to a multiclass classifier is in the later sections of this chapter.

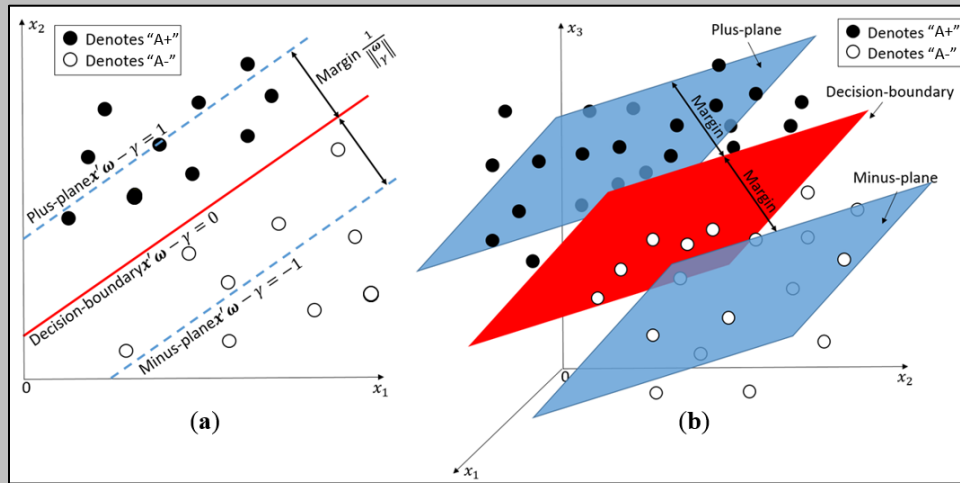
Similar to SVM, a PSVM decision condition is defined as (Figure 1):

$$\mathbf{x}'\boldsymbol{\omega} - \gamma \begin{cases} > 0, & \mathbf{x} \in A+; \\ = 0, & \mathbf{x} \in A+ \text{ or } A-; \\ < 0, & \mathbf{x} \in A-, \end{cases} \quad (1)$$

where  $\mathbf{x} \in R^n$  is an  $n$  dimensional vector data point to be classified,  $\boldsymbol{\omega} \in R^n$  implicitly defines the normal of the decision-boundary,  $\gamma \in R$  defines the location of the decision-boundary, and "A +" and "A -" are two classes of the binary classification. PSVM solves an optimization problem and takes the form of (Fung and Mangasarian, 2001):

$$\min_{\boldsymbol{\omega}, \gamma, \mathbf{y}} \nu \frac{1}{2} \|\mathbf{y}\|^2 + \frac{1}{2} (\boldsymbol{\omega}'\boldsymbol{\omega} + \gamma^2), \quad (2)$$

subject to



**Figure 1.** (a) Scratch of a two-class PSVM in 2-D space. Class "A+" and "A-" are approximated by two parallel lines that being pushed as far apart as possible. The decision boundary then sits right at the middle of these two lines. In this case, maximizing the margin is equivalent to minimizing  $(\mathbf{W}^T\mathbf{W} + \gamma^2)^{1/2}$ . (b) Two-class PSVM in 3D space. In this case the decision boundary becomes a plane.

$$\mathbf{D}(\mathbf{A}\boldsymbol{\omega} - \mathbf{e}\gamma) + \mathbf{y} = \mathbf{e}. \quad (3)$$

In this optimization problem,  $\mathbf{y} \in R^m$  is the error variable;  $\mathbf{A} \in R^{m \times n}$  is a sample matrix composed of  $m$  samples, which can be divided into two classes,  $A +$  and  $A -$ ;  $\mathbf{D} \in R^{m \times m}$  is a diagonal matrix of labels with a diagonal composed of "+1" for  $A +$  and "-1" for  $A -$ ;  $\nu$  is a non-negative parameter; and  $\mathbf{e} \in R^m$  is a column vector of ones. This optimization problem can be solved by using a Lagrangian multiplier  $\mathbf{u} \in R^m$ :

$$L(\boldsymbol{\omega}, \gamma, \mathbf{y}, \mathbf{u}) = \nu \frac{1}{2} \|\mathbf{y}\|^2 + \frac{1}{2} (\boldsymbol{\omega}'\boldsymbol{\omega} + \gamma^2) - \mathbf{u}'(\mathbf{D}(\mathbf{A}\boldsymbol{\omega} - \mathbf{e}\gamma) + \mathbf{y} - \mathbf{e}). \quad (4)$$

By setting the gradients of  $L$  to zero, we obtain expressions for  $\omega$ ,  $\gamma$  and  $\mathbf{y}$  explicitly in the knowns and  $\mathbf{u}$ , where  $\mathbf{u}$  can further be represented by  $\mathbf{A}$ ,  $\mathbf{D}$  and  $v$ . Then by changing  $\omega$  in equations 2 and 3 using its dual equivalent  $\omega = \mathbf{A}'\mathbf{D}\mathbf{u}$ , we can arrive at (Fung and Mangasarian, 2001):

$$\min_{\omega, \gamma, \mathbf{y}} v \frac{1}{2} \|\mathbf{y}\|^2 + \frac{1}{2} (\mathbf{u}'\mathbf{u} + \gamma^2), \quad (5)$$

subject to

$$\mathbf{D}(\mathbf{A}\mathbf{A}'\mathbf{D}\mathbf{u} - \mathbf{e}\gamma) + \mathbf{y} = \mathbf{e}. \quad (6)$$

Equations 5 and 6 provide a more desirable version of the optimization problem since one can now insert kernel methods to solve nonlinear classification problems made possible by the term  $\mathbf{A}\mathbf{A}'$  in Equation 6. Utilizing the Lagrangian multiplier again (this time we denote the multiplier as  $\mathbf{v}$ ), we can minimize the new optimization problem against  $\mathbf{u}$ ,  $\gamma$ ,  $\mathbf{y}$  and  $\mathbf{v}$ . By setting the gradients of these four variables to zero, we can express  $\mathbf{u}$ ,  $\gamma$  and  $\mathbf{y}$  explicitly by  $\mathbf{v}$  and other knowns, where  $\mathbf{v}$  is solely a dependent on the data matrices. Then for  $\mathbf{x} \in R^{1 \times n}$  we write the decision conditions as

$$\mathbf{x}'\mathbf{A}'\mathbf{D}\mathbf{u} - \gamma \begin{cases} > 0, & \mathbf{x} \in A+; \\ = 0, & \mathbf{x} \in A+ \text{ or } A-; \\ < 0, & \mathbf{x} \in A-, \end{cases} \quad (7)$$

with

$$\mathbf{u} = \mathbf{D}\mathbf{K}'\mathbf{D} \left( \frac{1}{v} + \mathbf{G}\mathbf{G}' \right)^{-1} \mathbf{e}, \quad (8)$$

$$\gamma = \mathbf{e}'\mathbf{D} \left( \frac{1}{v} + \mathbf{G}\mathbf{G}' \right)^{-1} \mathbf{e}, \quad (9)$$

and

$$\mathbf{G} = \mathbf{D}[\mathbf{K} \quad -\mathbf{e}]. \quad (10)$$

Instead of  $\mathbf{A}$ , we have  $\mathbf{K}$  in equations 8 and 10, which is a Gaussian kernel function of  $\mathbf{A}$  and  $\mathbf{A}'$  that has the form:

$$\mathbf{K}(\mathbf{A}, \mathbf{A}')_{ij} = \exp \left( -\sigma \|\mathbf{A}'_i - \mathbf{A}'_j\|^2 \right), i, j \in [1, m], \quad (11)$$

where  $\sigma$  is a scalar parameter. Finally, by replacing  $\mathbf{x}'\mathbf{A}'$  by its corresponding kernel expression, the decision condition can be written as:

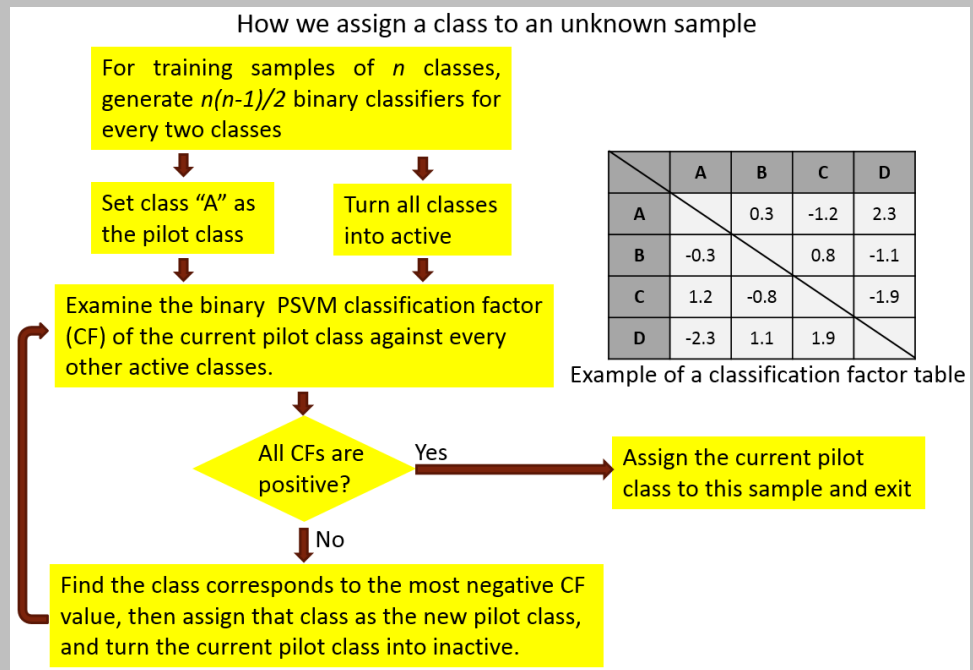
$$\mathbf{K}(\mathbf{x}', \mathbf{A}')\mathbf{D}\mathbf{u} - \gamma \begin{cases} > 0, & \mathbf{x} \in A+; \\ = 0, & \mathbf{x} \in A+ \text{ or } A-; \\ < 0, & \mathbf{x} \in A-. \end{cases} \quad (12)$$

and

$$\mathbf{K}(\mathbf{x}', \mathbf{A}')_{ij} = \exp(-\sigma \|\mathbf{x} - \mathbf{A}'_i\|^2), i \in [1, m]. \quad (13)$$

The formulations above represent a nonlinear PSVM classifier.

To extend this binary classifier to handle multiclass classification problems, some strategies have been developed by researchers, which generally lie into three categories: “one-versus-all”, “one-versus-one” and “all together”. The two former strategies, as one can tell from the names, build several binary classifiers individually ( $n(n - 1)/2$  for “one-versus-one” and  $n$  for “one-versus-all”, where  $n$  is the number of class), then use these classifiers to conclude the final classification decision. While “all together” will solve multiclass problems in one step. Experiments conducted by some researchers indicate a superiority of “one-versus-one” methods on large problems for practical use (Hsu and Lin, 2002). There are two popular particular algorithms for “one-versus-one” strategies, namely “Max Wins” (KreBel, 1999) and directed acyclic graph (DAG) (Platt et al., 2000). Both algorithms can give comparable results while surpassing the “one-versus-all” method in accuracy and computational efficiency. In our implementation, an approach similar to DAG is adopted and is described below.



**Figure 2.** Workflow of assigning a class to an unknown sample using a classification factor based scheme.

Our approach uses a classification factor table to assign classes to unknown samples. A classification factor of an unknown sample point for a certain pilot class “A”, is the normalized distance to the binary decision boundary between “A” and the other class used when generating this binary decision boundary. An example of a classification factor table is shown in Figure 2, and based on this table, the unknown sample point belongs to class “D”.

### Computation flow chart

The program **psvm3d** takes an ASCII format training file to build the PSVM classifier, then apply it to seismic taking multiple AASPI format seismic attribute input volumes. The training file consists of input/output pairs and can be constructed either from hand picking facies, or edited well logs. Several commercial software (e.g. point set in Petrel) and **make\_training\_clusters** can be used to manually pick facies of interest to be used as output in the training file. A well log property can also be used as output, as long as such property is discretized. After having the output ready, users can use **make\_training\_clusters** to extract attributes at corresponding locations to generate the input attributes in the training file. The flow chart of **psvm3d** is shown in Figure 3.

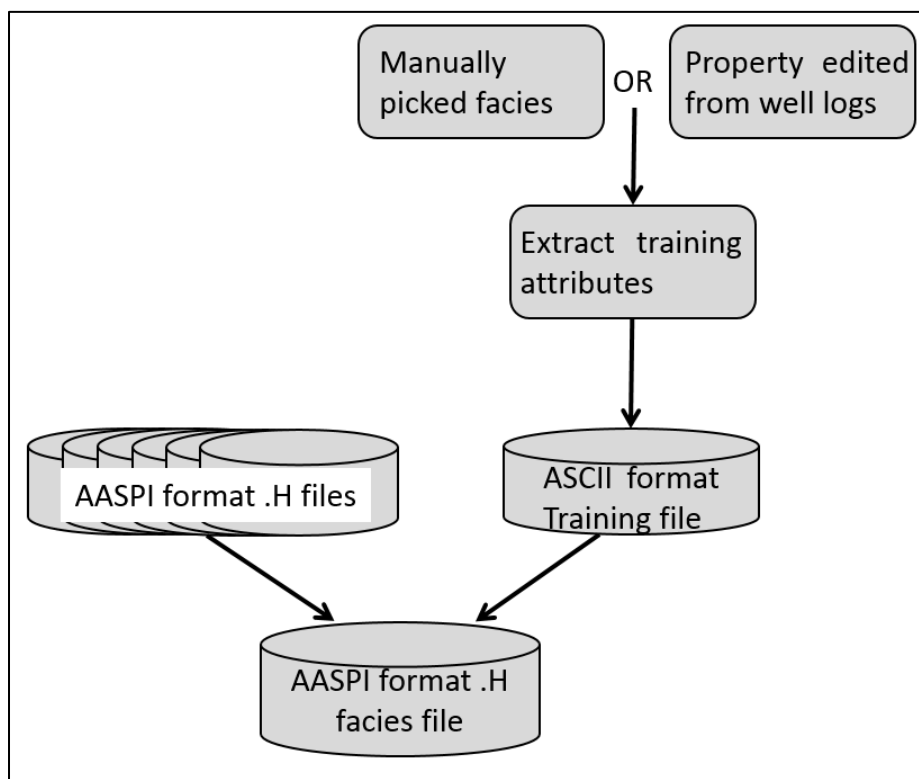


Figure 3. Flow chart of program **psvm3d**.

### Step-by-step instruction on program **psvm3d**

This Program **psvm3d** is launched from the *Volumetric Classification* in the main **aaspi\_util** GUI (Figure 4).

# Volumetric\_Classification: Program psvm3d

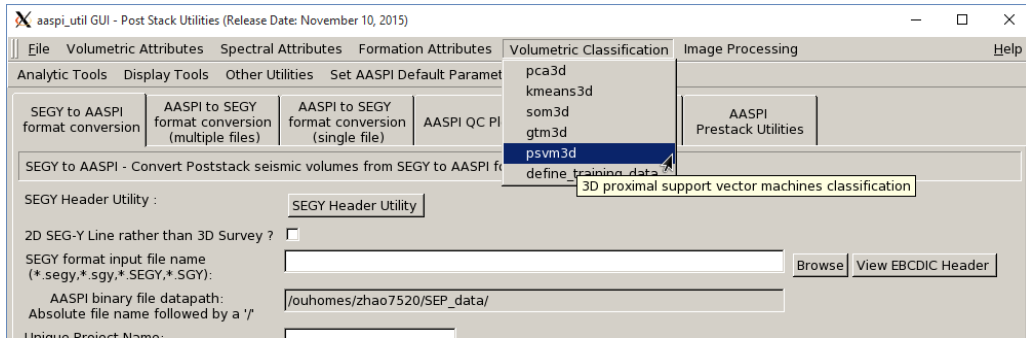


Figure 4. Launching the program PSVM Well Log Analysis.

The interface of **psvm3d** is shown below. We will go through all the options in detail.

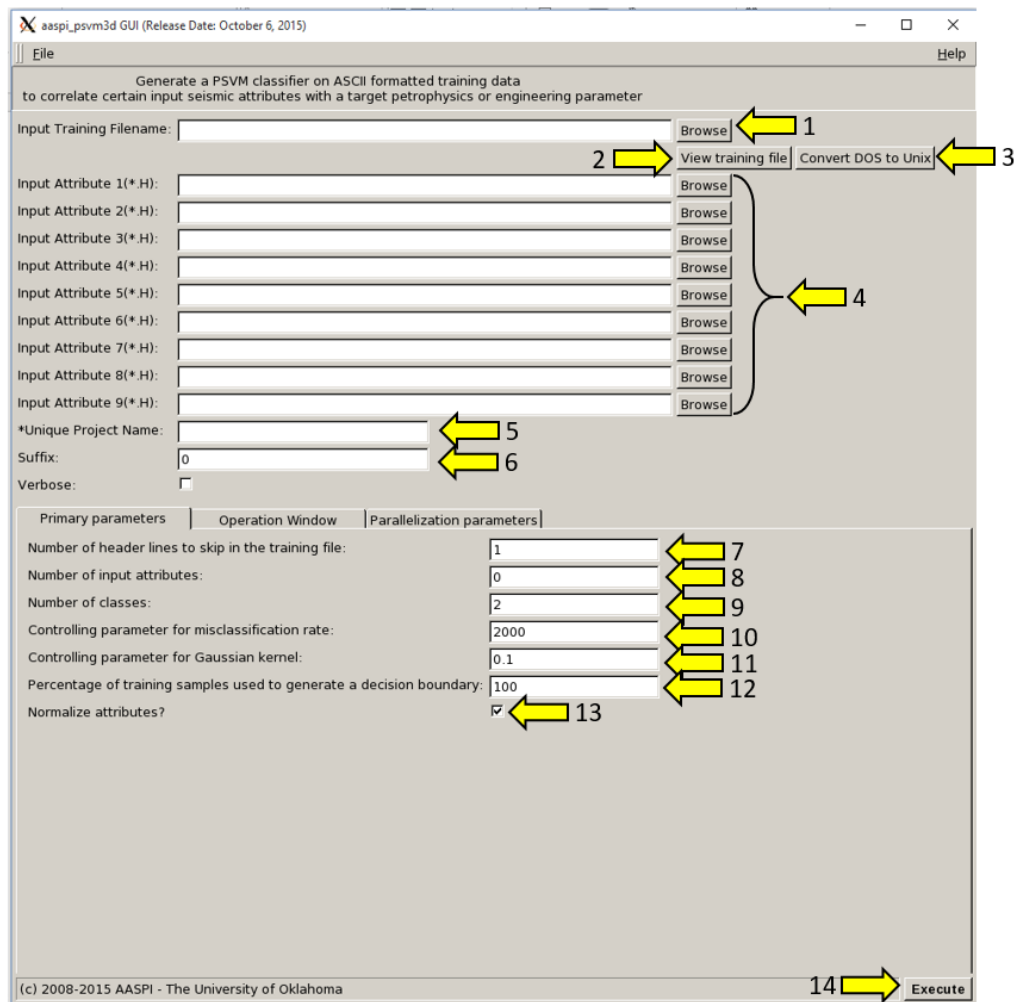


Figure 5. Interface of psvm3d.

**Button 1:** Browse training file.

Note: The program can only handle ASCII format training file, so please hand edit or using program **make\_training\_clusters** to generate a .txt or .dat file in the following format

## Volumetric\_Classification: Program psvm3d

(Figure 6). The format is: from left to right, each column is an input dimension (e.g. one attribute), then a column of label (positive integer numbers for facies or a discrete property). All other columns after “label” are ignored. You can have arbitrary number of header lines, which will be skipped during the importing of the files. **The sequence of input attributes in the training file must agree with the input attributes file list!**

log1	log2	log3	log4	label	extra columns... (e.g. MD)
51841.64063	26935.35742	1.704349708	3811383228	1	7502
33774.44531	17715.05469	1.634891917	1656611034	6	7660.5
50315.04688	26632.26758	1.569270777	3504752383	1	7923
53169.19922	27760.99414	1.6681764	4051182345	1	8502.5
34095.53906	16539.62695	2.249557077	1845954646	2	7868.5
48726.94141	24698.78906	1.89212672	3458510965	1	8597.5
38455.96484	19724.96484	1.800974794	2169707788	2	7705
45310.55469	24011.8457	1.560800845	3026602008	7	8262.5
52936.375	27013.35938	1.840176505	4029289962	1	7490
28180.46094	15991.72363	1.105314593	1059605743	6	8099
54165.91797	27750.04492	1.809998668	4244495243	1	8516.5
29320.0625	17191.31055	0.908785852	1142603987	8	8299.5
54166.74609	27742.24023	1.812259132	4257033610	1	8517
53954.65625	27858.05859	1.7510788	4216501385	1	8534.5
31412.4375	19265.32031	0.658586632	1171121773	9	8225.5
51861.29688	27149.27539	1.648970041	3758128825	1	7399.5
30382.01758	18928.20703	0.576405696	1066315103	8	8403
52596.90234	27241.33789	1.727893156	3928474660	1	7465
45016.66406	24360.98633	1.414734147	2743924255	1	7679
54823.67969	28755.60742	1.634889911	4182686811	1	8633.5
51951.76172	27140.26953	1.664141484	3777469633	1	7413.5
35710.83203	20000.30664	1.188061043	1720712563	8	8205.5
50521.89844	26961.88867	1.511225421	3523451743	1	7910
27744.17773	14988.05859	1.426517131	1118232594	2	8076.5
52579.58984	27344.14258	1.697479367	3906468779	1	7393.5
50714.21875	26632.41211	1.626089165	3592529109	1	7430
52662.21875	27469.63477	1.67529773	3906856425	1	7826
39965.76563	20995.17969	1.623573544	2275429956	1	7901.5
34559.80859	19429.59961	1.163842922	1624689227	4	8118.5
32354.3418	19198.0332	0.8402178	1324971963	8	8214
51962.21094	27150.45313	1.662866463	3792142635	1	7412
49247.73047	25697.74219	1.672675537	3424101737	1	7946
30129.89648	17727.28516	0.888755428	1134988504	8	8018
28798.59961	17923.15039	0.581749341	1000516282	10	8178.5
52114.26563	27085.1582	1.702120226	3833284544	1	7460.5
31283.79102	17445.36719	1.215722451	1320017984	5	8092
53879.60938	28386.38281	1.602705695	4044150337	1	8528.5

**Figure 6.** An example of a supported training file format.

**Button 2:** View the training file content (Figure 7).

**Button 3:** If the file is generated from Windows based software (e.g. Petrel), they will have the annoying carriage return (^M) at the end of each line (Shown in Figure 7). Use this button to delete those carriage returns if you prefer to (result shown in Figure 8).

Note: This function depends on your Linux environment, and therefore may not always work. However, **it will not affect reading in the file.**

**Buttons 4:** Browse input AASPI format attribute files. The files must agree with the order in the training file.

**Blank 5 and 6:** Project name and suffix. You can put the parameters as suffix.

**Blank 7:** Number of header lines to skip in the training file.

**Blank 8:** Number of input dimensions, i.e. number of attributes.

**Blank 9:** Number of classes (facies) within the data.

Note: Classes that do not appear in the training file cannot be predicted.

**Blank 10 and 11:** PSVM classifier parameters (must be positive real numbers). Generally, **Blank 10** controls how tight the classifier fits the training data, which will scarify the ability of



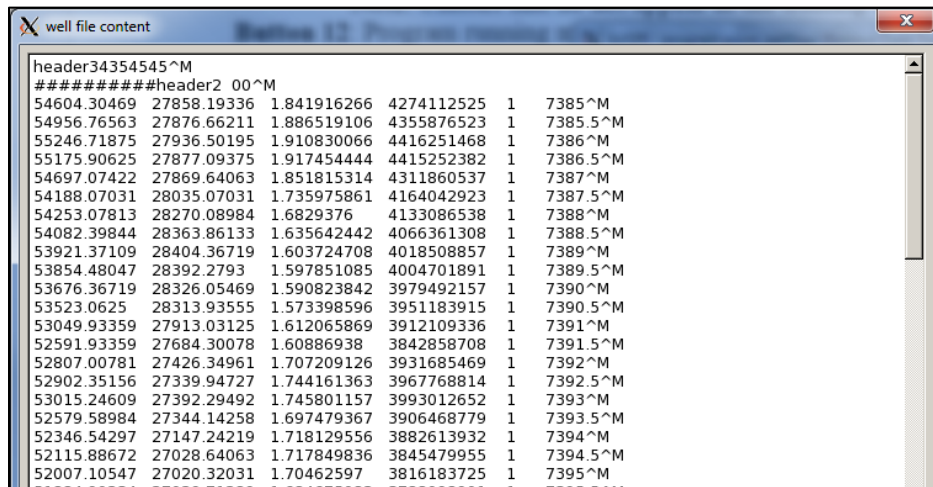
## Volumetric\_Classification: Program psvm3d

generalization. **Blank 11** is the standard deviation of a Gaussian function used in kernel mapping. **The classifier's performance is more sensitive to Blank 11 based on our test.**

**Blank 12:** Amount of samples out of the training file that are actually used for training. More training samples will have more computation cost, and sometimes not using all the available training samples may provide a more generalized classifier.

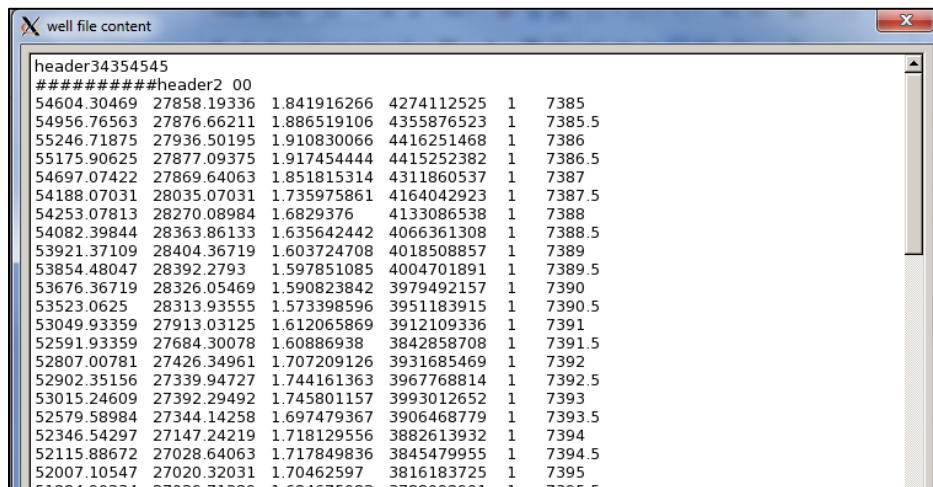
**Checkmark 13:** Check if you want to normalize attributes using z-score. Default setting is to normalize attributes, but in some circumstances it may produce better result without normalization.

**Button 14:** Run the program.



```
header34354545^M
#####header2 00^M
54604.30469 27858.19336 1.841916266 4274112525 1 7385^M
54956.76563 27876.66211 1.886519106 4355876523 1 7385.5^M
55246.71875 27936.50195 1.910830066 4416251468 1 7386^M
55175.90625 27877.09375 1.917454444 4415252382 1 7386.5^M
54697.07422 27869.64063 1.851815314 4311860537 1 7387^M
54188.07031 28035.07031 1.735975861 4164042923 1 7387.5^M
54253.07813 28270.08984 1.6829376 4133086538 1 7388^M
54082.39844 28363.86133 1.635642442 4066361308 1 7388.5^M
53921.37109 28404.36719 1.603724708 4018508857 1 7389^M
53854.48047 28392.2793 1.597851085 4004701891 1 7389.5^M
53676.36719 28326.05469 1.590823842 3979492157 1 7390^M
53523.0625 28313.93555 1.573398596 3951183915 1 7390.5^M
53049.93359 27913.03125 1.612065869 3912109336 1 7391^M
52591.93359 27684.30078 1.60886938 3842858708 1 7391.5^M
52807.00781 27426.34961 1.707209126 3931685469 1 7392^M
52902.35156 27339.94727 1.744161363 3967768814 1 7392.5^M
53015.24609 27392.29492 1.745801157 3993012652 1 7393^M
52579.58984 27344.14258 1.697479367 3906468779 1 7393.5^M
52346.54297 27147.24219 1.718129556 3882613932 1 7394^M
52115.88672 27028.64063 1.717849836 3845479955 1 7394.5^M
52007.10547 27020.32031 1.70462597 3816183725 1 7395^M
```

Figure 7. An example of viewing a training file content. Carriage returns are visible as “^M”.



```
header34354545
#####header2 00
54604.30469 27858.19336 1.841916266 4274112525 1 7385
54956.76563 27876.66211 1.886519106 4355876523 1 7385.5
55246.71875 27936.50195 1.910830066 4416251468 1 7386
55175.90625 27877.09375 1.917454444 4415252382 1 7386.5
54697.07422 27869.64063 1.851815314 4311860537 1 7387
54188.07031 28035.07031 1.735975861 4164042923 1 7387.5
54253.07813 28270.08984 1.6829376 4133086538 1 7388
54082.39844 28363.86133 1.635642442 4066361308 1 7388.5
53921.37109 28404.36719 1.603724708 4018508857 1 7389
53854.48047 28392.2793 1.597851085 4004701891 1 7389.5
53676.36719 28326.05469 1.590823842 3979492157 1 7390
53523.0625 28313.93555 1.573398596 3951183915 1 7390.5
53049.93359 27913.03125 1.612065869 3912109336 1 7391
52591.93359 27684.30078 1.60886938 3842858708 1 7391.5
52807.00781 27426.34961 1.707209126 3931685469 1 7392
52902.35156 27339.94727 1.744161363 3967768814 1 7392.5
53015.24609 27392.29492 1.745801157 3993012652 1 7393
52579.58984 27344.14258 1.697479367 3906468779 1 7393.5
52346.54297 27147.24219 1.718129556 3882613932 1 7394
52115.88672 27028.64063 1.717849836 3845479955 1 7394.5
52007.10547 27020.32031 1.70462597 3816183725 1 7395
```

Figure 8. An example of a training file after deleting carriage returns.

In the **Operation Window** tab, all functions are explained in Figure 9 (The panel shown is from **som3d**, and the **psvm3d** has an identical operation window panel).



# Volumetric\_Classification: Program psvm3d

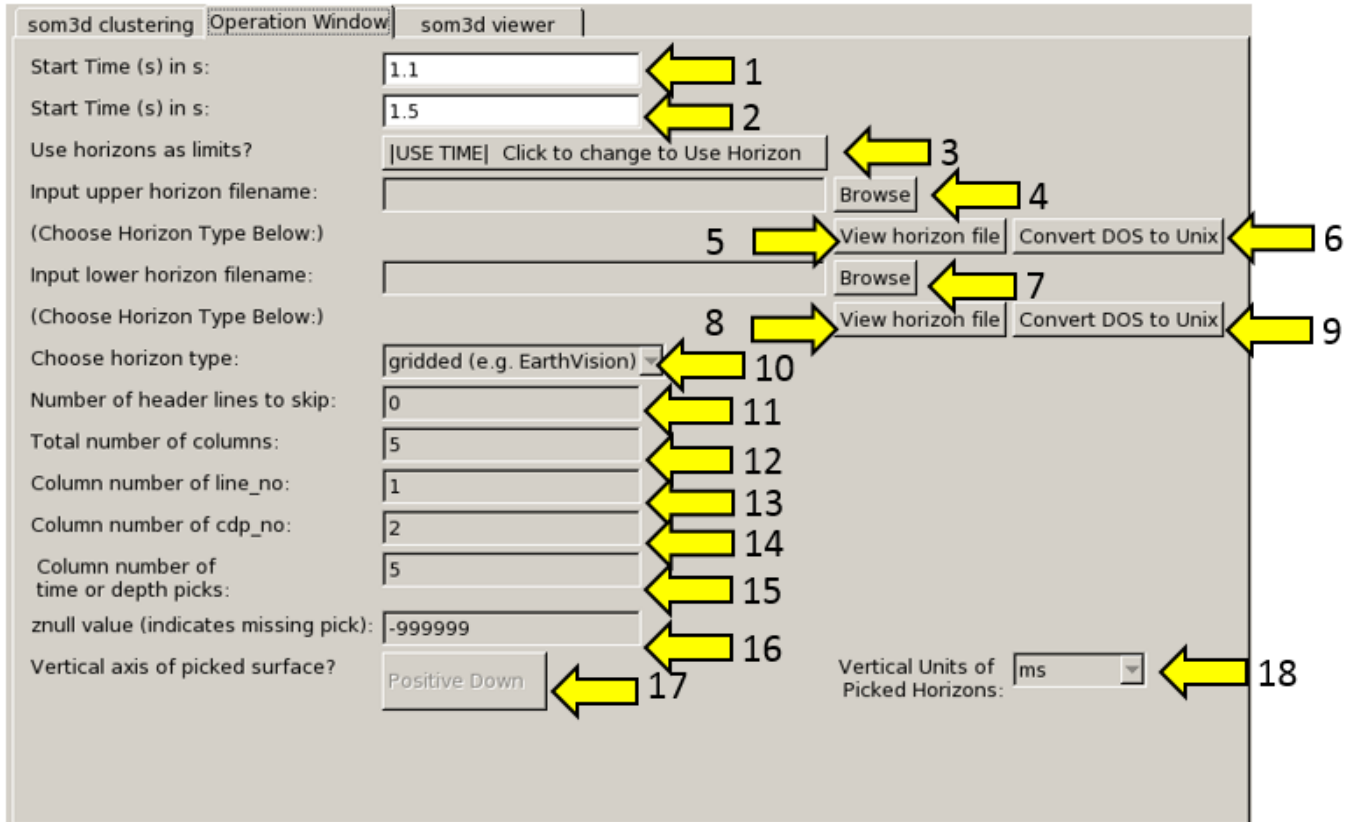


Figure 9. Operation Window tab in som3d (identical to psvm3d).

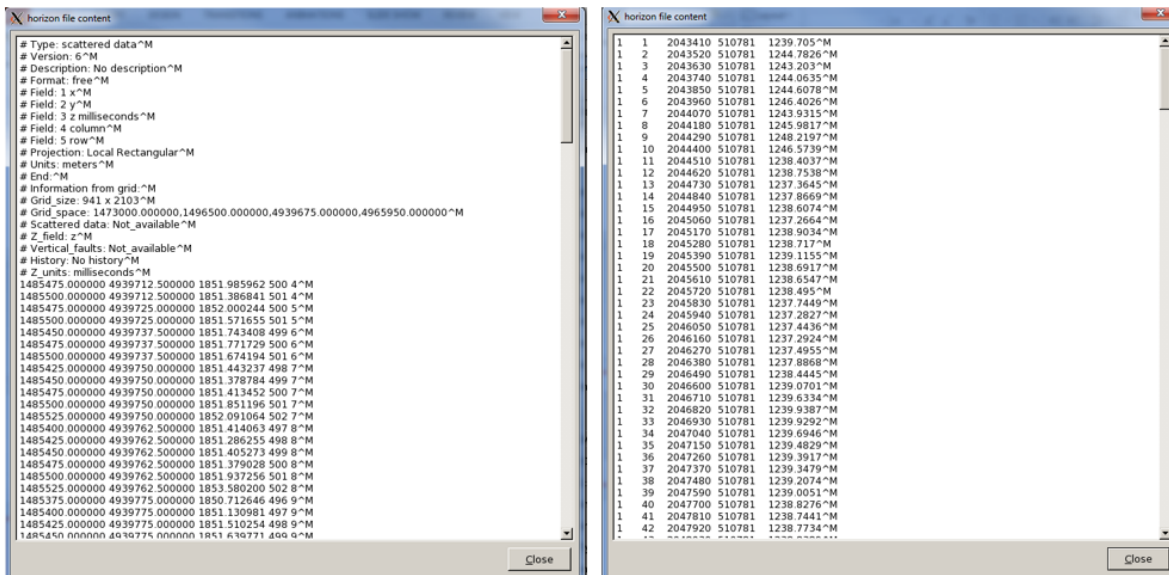


Figure 10. (left) A gridded horizon file (EarthVision format). (right) An interpolated horizon file with five columns (ASCII free format).

### Horizon definition

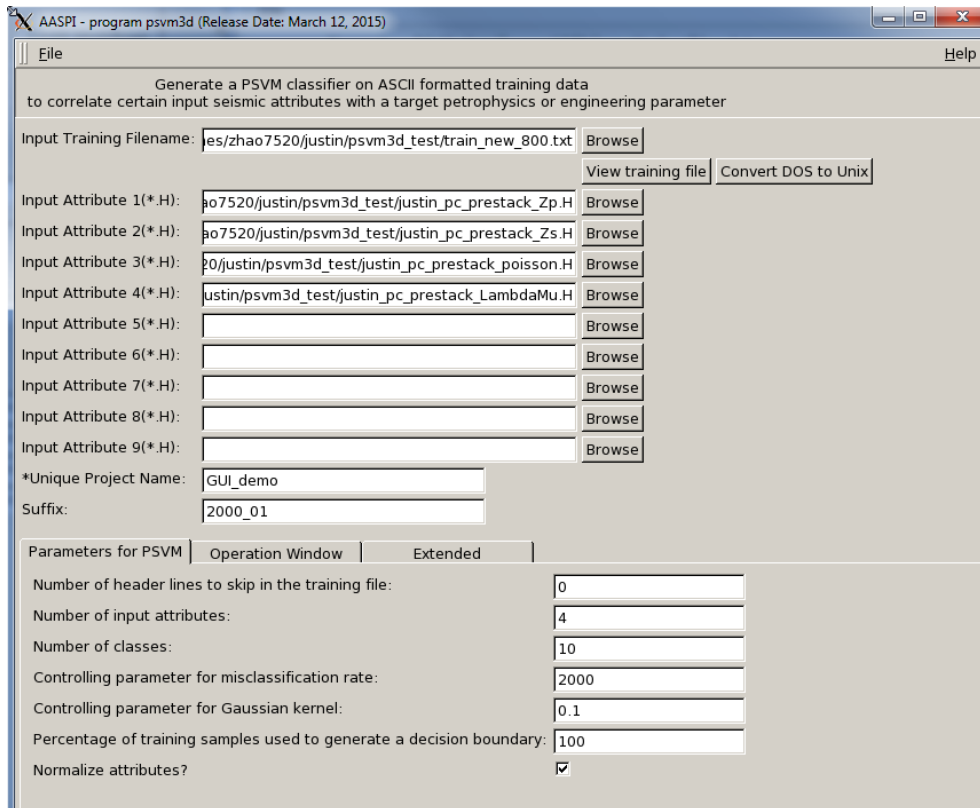
The horizon definition panel will look the same for almost all AASPI GUIs:

1. Start time (upper boundary) of the analysis window.
2. End time (lower boundary) of the analysis window.
3. Toggle that allows one to do the analysis between the top and bottom time slices described in 1 and 2 above, or alternatively between two imported horizons. If *USE HORIZON* is selected, all horizon related options will be enabled. If the horizons extend beyond the window limits defined in 1 and 2, the analysis window will be clipped.
4. Browse button to select the name of the upper (shallower) horizon.
5. Button that displays the horizon contents (see Figure 10).
6. Button to convert horizons from Windows to Linux format. If the files are generated from Windows based software (e.g. Petrel), they will have the annoying carriage return (^M) at the end of each line (Shown in Figure 10). Use these two buttons to delete those carriage returns. Note: This function depends on your Linux environment. If you do not have the program **dos2unix** it may not work. In these situations, the files may have been automatically converted to Linux and thus be properly read in.
7. Browse button to select the name of the lower (deeper) horizon.
8. Button that displays the horizon contents (see Figure 10).
9. Button to convert horizons from Windows to Linux format. (see 6 above).
10. Toggle that selects the horizon format. Currently *gridded* (e.g. EarthVision in Petrel) and *interpolated* (ASCII free format, e.g. SeisX) formats are supported. The *gridded* horizons are nodes of B-splines used in mapping and have no direct correlation to the seismic data survey. For example, *gridded* horizons may be computed simply from well tops. The x and y locations are aligned along north and east axes. In contrast *interpolated* horizons are defined by *line\_no*, *cdp\_no* (*crossline\_no*) and *time* triplets for each trace location. Examples of both formats are shown in Figure 10. If *interpolated* is selected, the user needs to manually define each column in the file.
11. Number of header lines to skip in the *interpolated* horizon files.
12. Total number of columns in the *interpolated* horizon files.
13. Enter the column number containing the *line\_no* (*inline\_no*) of the interpolated data triplet.
14. Enter the column number containing the *cdp\_no* (*crossline\_no*) of the interpolated data triplet.
15. Enter the column number containing the *time* or *depth* value of the interpolated data triplet.
16. *Znull* value (indicate missing picks) in the horizon files.
17. Toggle to choose between *Positive Down* and *Negative Down* for the horizon files (e.g. Petrel uses negative down).
18. Choose the vertical units used to define the horizon files (either *s*, *ms*, *kft*, *ft*, *km*, or *m*).

Here we see an application using the program **psvm3d** to predict brittleness index (BI) from four inversion derived seismic attributes. Before running **psvm3d**, it is strongly advised to test the training data using program **PSVM Well Log Analysis** which is used for testing, validating, and predicting ASCII files. In **PSVM Well Log Analysis**, the user can manually divide the training file into two portions, training and testing, or use the whole training file in cross-validation model to find the best parameters. Once the optimal parameters are found, the user can use this training file, seismic attributes (in the corresponding order), and the PSVM parameters that

## Volumetric\_Classification: Program **psvm3d**

tested out in **PSVM Well Log Analysis**, to perform classification using **psvm3d**. The parameters used in our example are shown in Figure 11.



The screenshot shows the AASPI program psvm3d interface. The window title is "AASPI - program psvm3d (Release Date: March 12, 2015)". The main area contains the following fields and controls:

- Input Training Filename:
- Input Attribute 1(\*.H):
- Input Attribute 2(\*.H):
- Input Attribute 3(\*.H):
- Input Attribute 4(\*.H):
- Input Attribute 5(\*.H):
- Input Attribute 6(\*.H):
- Input Attribute 7(\*.H):
- Input Attribute 8(\*.H):
- Input Attribute 9(\*.H):
- \*Unique Project Name:
- Suffix:

Below these fields are three tabs: "Parameters for PSVM", "Operation Window", and "Extended". The "Parameters for PSVM" tab is active and shows the following settings:

Parameter	Value
Number of header lines to skip in the training file:	0
Number of input attributes:	4
Number of classes:	10
Controlling parameter for misclassification rate:	2000
Controlling parameter for Gaussian kernel:	0.1
Percentage of training samples used to generate a decision boundary:	100
Normalize attributes?	<input checked="" type="checkbox"/>

**Figure 11.** PSVM parameter settings for BI prediction.

As shown in Figure 11, we prepared a training file from one study well, and use the parameters listed in the panel. The input attributes are P-impedance, S-impedance, Poisson's Ratio, and Lambda/Mhu Ratio, where the target properties is brittleness which is digitalized in to 10 classes, 1 being the least brittle and 10 being the most. In the **Operation Window** tab (Figure 12), we use two horizons as upper and lower limits of the operation window, and the horizons are **interpolated**, which means we need to define the columns. And finally I choose to use 50 processors to speed up the job (Figure 13).

## Volumetric\_Classification: Program psvm3d

Parameters for PSVM | Operation Window | Extended

Start Time (s) in s: 1.1  
Start Time (s) in s: 1.5  
Use horizons as limits? [USE HORIZON] Click to change to Use Time  
Input upper horizon filename: 520/justin/psvm3d\_test/Lower\_Barnett\_pc.txt Browse  
(Choose Horizon Type Below): View horizon file Convert DOS to Unix  
Input lower horizon filename: es/zhao7520/justin/psvm3d\_test/Viola\_pc.txt Browse  
(Choose Horizon Type Below): View horizon file Convert DOS to Unix  
Choose horizon type: interpolated (e.g. SeisX)  
Number of header lines to skip: 0  
Total number of columns: 5  
Column number of line\_no: 1  
Column number of cdp\_no: 2  
Column number of time or depth picks: 5  
znull value (indicates missing pick): -9999.25  
Vertical axis of picked surface? Positive Down  
Vertical Units of Picked Horizons: ms

Figure 12. Operation window options for BI prediction.

Primary parameters | Operation Window | Parallelization parameters

Use MPI:   
Processors per node: 50 Determine Maximum Processors on localhost  
Node list (separated by blanks): localhost  
Build an LSF Script? Do Not Run Under LSF  
Build a PBS Script? Do Not Run Under PBS  
Maximum LSF run time (hrs): 10  
Available batch processors: 0 Determine Optimum Number of Batch Processors  
LSF Batch Queue:

Figure 13. Extended (MPI) options for BI prediction.

After clicking the **Execute** button, the user is able to supervise the running progress (Figure 14).

```
zhao7520@ediakaran:~/justin/psvm3d_test
Number of samples used for generating this boundary is 425
0 Matrix inverse finished
Matrix nu generated before matmul
after matmul
Boundary No. 1 is generated successfully

-----
Boundary No. 2 is generating...
Number of samples used for generating this boundary is 390
0 Matrix inverse finished
Matrix nu generated before matmul
after matmul
Boundary No. 2 is generated successfully

-----
Boundary No. 3 is generating...
Number of samples used for generating this boundary is 381

zhao7520@ediakaran:~/justin/psvm3d_test
Performing PSVM classification: jline_end_line 3 198
Performing PSVM classification: jline_end_line 4 198
Performing PSVM classification: jline_end_line 5 198
Performing PSVM classification: jline_end_line 6 198
Performing PSVM classification: jline_end_line 7 198
Performing PSVM classification: jline_end_line 8 198
Performing PSVM classification: jline_end_line 9 198
Performing PSVM classification: jline_end_line 10 198
Performing PSVM classification: jline_end_line 11 198
Performing PSVM classification: jline_end_line 12 198
Performing PSVM classification: jline_end_line 13 198
Performing PSVM classification: jline_end_line 14 198
Performing PSVM classification: jline_end_line 15 198
Performing PSVM classification: jline_end_line 16 198
Performing PSVM classification: jline_end_line 17 198
Performing PSVM classification: jline_end_line 18 198
Performing PSVM classification: jline_end_line 19 198
Performing PSVM classification: jline_end_line 20 198
Performing PSVM classification: jline_end_line 21 198
Performing PSVM classification: jline_end_line 22 198
Performing PSVM classification: jline_end_line 23 198
Performing PSVM classification: jline_end_line 24 198
Performing PSVM classification: jline_end_line 25 198
```

Figure 14. Running progress windows.

Once finished, the user will locate the generated classification file in the same directory named as “psvm3d\_classification\_projectname\_suffix.H”, which can be display using either the **AASPI\_QC\_Plot** or other commercial software. Figure 15 shows the final classification result displayed in Petrel.

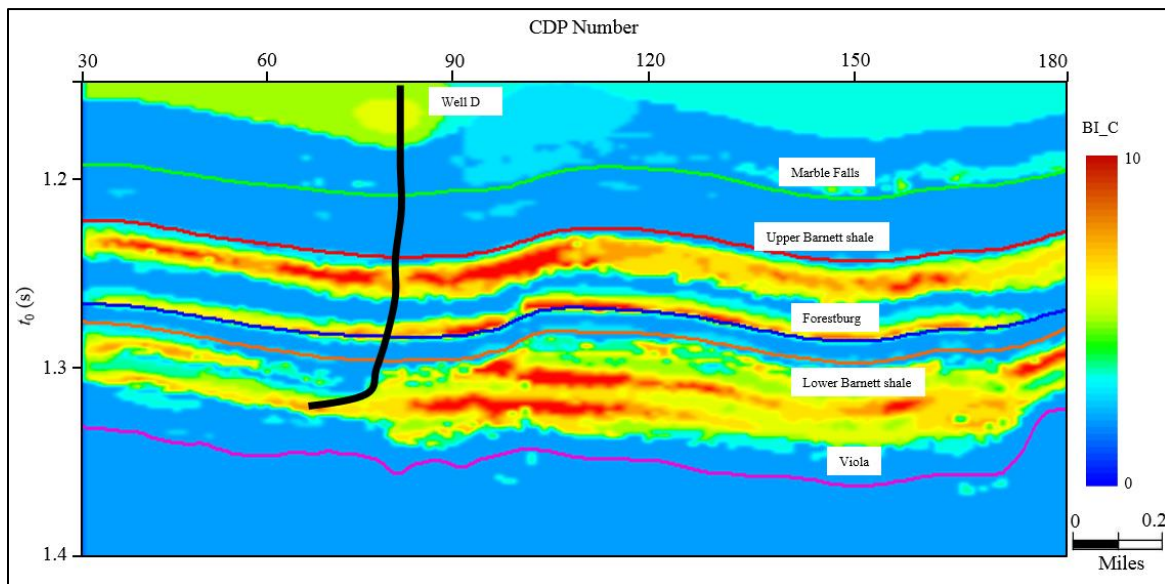


Figure 15. PSVM classification for brittleness index (Zhang et al., 2015).

## References

- Balabin, R. M. and E. I. Lomakina, 2011, Support vector machine regression (SVR/LS-SVM)—an alternative to neural networks (ANN) for analytical chemistry? Comparison of nonlinear methods on near infrared (NIR) spectroscopy data: *Analyst*, **136**, 1703-1712.
- Bennett, K. P. and A. Demiriz, 1999, Semi-supervised support vector machines: Advances in Neural Information Processing Systems 11: Proceedings of the 1998 Conference, 368-374.
- Fung, G. and O. L. Mangasarian, 2001, Proximal support vector machine classifiers: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM 2001, 77-86.
- Fung, G. M. and O. L. Mangasarian, 2005, Multicategory proximal support vector machine classifiers: *Machine Learning*, **59**, 77-97.
- Mangasarian, O. L. and E. W. Wild, 2006, Multisurface proximal support vector machine classification via generalized eigenvalues: *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **28**, 69-74.
- Shawe-Taylor, J. and N. Cristianini, 2004, Kernel methods for pattern analysis: Cambridge University Press, New York, United States.
- Verma, A. K., T. N. Singh and S. Maheshwar, 2014, Comparative study of intelligent prediction models for pressure wave velocity: *Journal of Geosciences and Geomatics*, **2**, 130-138.

## Volumetric\_Classification: Program **psvm3d**

Wong, W. and S. Hsu, 2006, Application of SVM and ANN for image retrieval: European Journal of Operational Research, **173**, 938-950.

Zhang, B., T. Zhao, X. Jin, and K. J. Marfurt, 2015, Brittleness evaluation of resource plays by integrating petrophysical and seismic data analysis: Interpretation, **3**, T81–T92.