1 Topic

Description of the CVM (Core Vector Machine) and BVM (Ball Vector Machine) methods from the LIBCVM library (<u>http://c2inet.sce.ntu.edu.sg/ivor/cvm.html</u>).

The Support Vector Machines algorithms are well-known in the supervised learning domain. They are especially appropriate when we handle a dataset with a large number "p" of descriptors¹. But they are much less efficient when the number of instances "n" is very high. Indeed, a naive implementation is of complexity $O(n^3)$ for the calculation time and $O(n^2)$ for the storing of the values. In consequence, instead of the optimal solution, the learning algorithms often highlight the near-optimal solutions with a tractable computation time².

I recently discovered the CVM and BVM approaches. The idea of the authors is really clever: since only approximate best solutions can be highlighted, their approaches try to resolve an equivalent problem which is easier to handle - the minimum enclosing ball problem in computational geometry to detect the support vectors. So, we have a classifier which is as efficient as those obtained by the other SVM learning algorithms, but with an enhanced ability to process datasets with a large number of instances.

I found the papers really interesting. They are all the more interesting that all the tools enabling to reproduce the experiments are provided: the program and the datasets. So, all the results shown in the paper can be verified. It contrasts to too numerous papers where some authors flaunt tremendous results but we can never reproduce them.

The CVM and BVM methods are incorporated into the LIBCVM library. This is an extension of the LIBSVM (version 2.85), which is already included into Tanagra. The source code for LIBCVM being available, I compiled it as a DLL (Dynamic-link Library) and I included it also into Tanagra 1.4.44.

In this tutorial, we describe the behavior of the CVM and BVM supervised learning methods on the "Web" dataset available on the website of the authors. We compare the results and the computation time to those of the C-SVC algorithm based on the LIBSVM library.

2 "Web" dataset in the sparse format

The "<u>web</u> » database is initially subdivided in 2 files: "w8a.data" corresponds to the learning sample (49,749 instances); "w8a.test" is the test sample (14,951 instances). They are both in the sparse format, handled – among others – by the SVMLIGHT and the LIBSVM libraries.

Each row represents an instance of the dataset. The first value is the class membership. Then, we have a list of couples of values divided by the character ":". The first one is the variable number; the second one is the value associated to this variable. Thus, the value for the not referenced variables in a row is implicitly 0. This kind of representation enables to dramatically reduce the size of the data file without loss of information. Below, we observe the firsts instances of "w8a.data" data file.

¹ E.g. <u>http://data-mining-tutorials.blogspot.fr/2009/07/implementing-svm-on-large-dataset.html</u> (n=135, p=31,809).

² Paragraph based on the papers available on the LIBCVM website - <u>http://c2inet.sce.ntu.edu.sg/ivor/cvm.html</u>

📔 D:\Da	Mining\Databases_for_mining\benchmark_datasets\web\web\w8a.data - Notepad++	ΣX
Fichier	dition Recherche Affichage Encodage Langage Paramétrage Macro Exécution Compléments	
Docume	ts ?	Х
🕞 占	🗄 🐚 🕞 🕞 😂 🎸 🏠 🍋 🗢 🗲 🏙 🍖 🤏 👒 🖫 🖓 🎫 🔚 🎼 🖉 🗉 🗈	>>
😑 w8a.d	a 🖸	
1	1 41:1 54:1 117:1 250:1	
2	1 59:1 68:1 115:1	
3	1	
4	1	
5	1 41:1 54:1 55:1 106:1 117:1 149:1 171:1 206:1 207:1 217:1 222:1 298:1	
6	1	
7	1	Ŧ
length : 3	/8238 lines: 4975 Ln:1 Col:1 Sel:0 UNIX ANSI INS	

The first observation belongs to the class "-1". Then, we have V1 = 0 since this variable is not referenced; V2 = 0; ...; V41 = 1 because we observe "41:1"; V42 = 0; ...; V54 = 1; etc.

A data file with a large number of null values is common when it is obtained from the pretreatment of an unstructured document (e.g. text, image). This is a kind of compression. Except that the resulting file is readable with a simple text editor.

For convenience, we have concatenated the two data files and we used a small program to transform the file in the dense format (standard tab separated attribute value description, "**w8a.txt**"). Thus, we have a data file with 64,700 rows and 301 columns (including the class attribute). We show here the firsts rows of the new data file.

w 🗐	Ba.txt	þff	ice (Calc											-	-		-	-					x
Eichi	er É <u>d</u> itio	n <u>A</u> f	ficha	age	Ins	erti	on	F <u>o</u> rn	nat	<u>O</u> ut	tils <u>D</u>	<u>)</u> onné	es S	ipina	Tana	agra	Fe <u>n</u> êt	tre A	id <u>e</u>				_ &	×
i 🖬	- 🖻 ۽	4 @		2	PDI		3 (0	ABC Ø	ABC) 🔏			• 🝰		• @	→	M	n, z ⊻iz \$	la	e [2	🔶 🖷	»
	Albany AMT 💽 10 💽 🙈 🏾 🖉 🖺 🗄 🗏 🗐 🛄 % 🐜 🕬 ∉ 🖻 📄 🔹																							
A1			•	f ix,	Σ	E =	=	clas	se															
	A	В	C	D	Ε	F	G	Н	Ι	J	K	L	м	Ν	0	Р	Q	R	S	Т	U	V	W	X
1	classe	vi v	2 v	З (/4	v5	v6	v7	v8	v9	v10	v11	v12	v13	v14	v15	v16	v17	v18	v19	v20	v21	v22 _	<mark>/23</mark> 🗐
2	neg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
3	neg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
4	neg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
5	neg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
6	neg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
7	neg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
8 ((neq	0 euille	0 1/-	• ₽/	0	0	0	0	0	0 ∢	0	0	0	0	0	0	0	0	0	0	0	0	0	ν. Ε
Feui	lle1/1			Par	défa	ut					S	TD	*		S	omm	e=0		Θ-		Ó		• 🕀 🗄	L00%

3 Core Vector Machine (CVM)

We do not describe the CVM approach in this section. This has been done in the Tsang and al. paper available online: Ivor W. Tsang, James T. Kwok, Pak-Ming Cheung. <u>Core vector machines: Fast SVM</u> <u>training on very large data sets.</u> Journal of Machine Learning Research, 6:363-392, 2005. According to the authors, the main consequence is that we can handle large datasets with similar results to the standard implementation of the SVM algorithms. To check this, we will compare the results with those of C-SVC from the LIBSVM library (section 5).

3.1 Importing the data file

Into Tanagra, we click on the FILE / NEW menu to create a new diagram. We select the "w8a.txt" data file and we start the data importation.

TANAGRA 1.4.44									
File Diagram Window H	łelp								
D New									
🗃 Open Data mir	ing diagram (empty)								
Save									
Save as									
Close	Choose your dataset and start download								
Exit									
	Diagram title : Default title								
	Data mining diagram file name :								
	:\DataMining\Databases_for_mining\benchmark_datasets\web\w8a.tdm								
	Dataset (*.txt,*.arff,*.xls) :								
	D:\DataMining\Databases_for_mining\benchmark_datasets\web\w8a.txt								
Data visualization									
Feature construction									
	Checking Missing Values OK Cancel Help								
PLS									
Spv learning assessment	Scoring Association								
Correlation scatterplot	🖉 Scatterplot 🔛 View dataset								
Export dataset	🔀 Scatterplot with label 🛛 🔛 View multiple scatterplot								

301 variables and 64,700 instances are imported. The class attribute is discrete; all the others are viewed as continuous even if they are binary. This last remark is not insignificant. This will determine the data transformation mode for the learning process.

TANAGRA 1.4.44 - [Datase	et (w8a.txt)]						
Tile Diagram Compo	nent Window	Help					- 8 ×
🗅 📽 🖪 👫							
Default til	tle		D (*
Dataset (w8a.txt)			Datase	et desc	ription		
			301 attribu				
			64700 exa	mple(s)			
			Attribute	Category	Informations		
			classe	Discrete	2 values		
			v1	Continue	-		
			v2 Continue -				-
							•
			Compone	nts			
Data visualization	Statist	ics	Nonp	arametric	statistics	Instance selection	
Feature construction	Feature se	lection		Regressi	on	Factorial analysis	
PLS	Cluster	ing		Spv learn	ing	Meta-spv learning	
Spv learning assessment	Scorir	ng		Associat	ion		
Correlation scatterplot	🖳 Export da	taset	1.	Scatterp	lot	🖉 Scatterplot wi	th label
•							4

3.2 Partition into learning and test samples

The learning sample corresponds to the first 49,749 instances. We partition the dataset using the SELECT FIRST EXAMPLES component (INSTANCE SELECTION tab).

TANAGRA 1.4.44 - [Selec	t first examples 1]		
Tile Diagram Comp	onent Window Help		- 8 ×
🗅 📽 🔛 🎎			
Default	title	Select first examples 1	*
Dataset (w8a.txt)		Select first instances	
	Parameters	Parameters	= =
	/iew	m: 49749	
Data visualization	Statistics		
Feature construction	Feature selection	OK Cancel Help	
PLS	Clustering		
Spv learning assessment	Scoring	Association	
Continuous select exam Discrete select example	es 🍂 Rule-based sele	· · · · · · · · · · · · · · · · · · ·	ied samplin
	III		 is.

We add the component into the diagram. We activate the PARAMETERS contextual menu. We set "m = 49749". We validate the choice and we click on the VIEW menu.

TANAGRA 1.4.44 - [Select	first examples 1]		
📅 File Diagram Compo	onent Window Help		_ <i>B</i> >
🗅 📽 🔛 👪			
Default t	itle	Select first e	examples 1
⊡ 🛄 Dataset (w8a.txt)		Parame	eters
Select first exa	bc	mple size : 49749	
Par	ameters		=
Exe	cute	Resu	
Vie	w 49	749 selected examples from 6470	
		mputation time : 0 ms.	
	Cr	eated at 18/05/2012 11:25:54	*
		mponents	
Data visualization	Statistics	Nonparametric statistics	Instance selection
Feature construction	Feature selection	Regression	Factorial analysis
PLS	Clustering	Spv learning	Meta-spv learning
Spv learning assessment	Scoring	Association	
🖉 Continuous select exam	iples 🌂 Recover examples	🖉 Sampling	💉 Stratified samplin
🖉 Discrete select example	es 🛛 🦧 Rule-based selectio	n 💦 🗡 Select first exam	mples
•			•
			د

Tanagra confirms the selection of the first observations.

3.3 Set the parameters of learning process

We insert the DEFINE STATUS component to indicate the role of the variables: CLASSE is the target attribute; the others are the input ones "V1...V300".



Then, we add the CVM (SPV LEARNING tab). We set the parameters by clicking on the SUPERVISED PARAMETERS menu.



Into the **METHOD** tab, we specify the settings of the learning process³:

cv	M parameters		
]	
	Method	Transformation	Optimization
		Kernel type :	Radial basis function
	Degree of ker	nel function :	1
		Gamma :	1 🔶
		Coef 0 :	0
	Penality cost (Complexity) :	1 🗲
	Max # of c	ore vectors :	50000 ←
			Compute probabilities
	M	ore informations a	about parameters
			OK Cancel Help

- Kernel Type: RBF (Radial Basis Function, option : -t 2)
- Gamma: 1 (-g 1)
- Cost: 1 (-c 1)
- Max number of core vectors: 50000 (-f 50000)

Into the **TRANSFORMATION** tab, we set the eventual data transformation used. **This option is specific to Tanagra**. By default, Tanagra uses the following NORMALIZATION for each input attribute:

$$z = 2 \times \frac{x - x_{min}}{x_{max} - x_{min}} - 1$$

So, Z varies between [-1; +1]. This transformation is important when the variables are expressed in different units. The variances are not the same. The variables which have higher amplitude may influence excessively the calculations. For our dataset, we have only binary input attributes 0/1. This normalization is not really required. This is the reason for which we select the **NONE** option.

CVM parameters	-
Method T	ransformation Optimization
	Data transformation None
	OK Cancel Help

³ See <u>http://c2inet.sce.ntu.edu.sg/ivor/cvm.html</u> to understand these settings (Section **How to Use**).

In the **OPTIMIZATION** tab, we set the options which influence the quality of the optimization (and which influence also the calculation times!):

CVM p	arameters		_	
	Method	Transformation	Optimization	
	Epsilo	n for Tolerance : Vse shrinkin	(b.0001 C	~
			ОК	Cancel Help

- Epsilon: 0.0001 (-e 0.0001)
- Use shrinking for optimization (-h 1).

We validate and we click on the VIEW menu to run the learning phase.

3.4 Training phase and model evaluation



The resubstitution error rate is 0.55% (244 + 31 = 275 misclassified instances on 49,749). There are 3,843 support vectors.

SVM characteristics Characteristic Value # classes 2 # support vectors 3843 # support vectors for each class # sv. for neg 2592 # sv. for pos 1251 Computation time : 148123 ms. Created at 18/05/2012 11:59:47

The calculation time of the learning phase is 148 seconds. But it can be divided in two parts, the detail for each step is provided into the log file (debugfile.txt): 65 seconds are dedicated to the creation of the model, this is really fast for a SVM like algorithm considering the size of our dataset; 83 seconds correspond to the classification of the instances into the learning sample *AND* the test sample. This second step is needed for the calculation of the error rate on the learning sample, but it is also need for the calculation of the test error rate. Its execution time is not negligible. It will be all the more important that

the number of support vectors is high.



To obtain the test error rate, we add again the DEFINE STATUS component. We set CLASSE as TARGET, and the prediction column PRED_SPVINSTANCE_1 as INPUT.



Then, we insert the TEST component (SPV LEARNING ASSESSMENT tab) into the diagram. By default, it computes the confusion matrix on the unselected instances into the branch i.e. the test sample.



The test error rate is 0.57%, with 85 (74 +11) misclassified among 14,951 instances.

4 Ball Vector Machine

BVM is a variant of CVM. The reference article is: I. W. Tsang, A. Kocsor, J. T. Kwok. <u>Simpler core</u> <u>vector machines with enclosing balls</u>. *Proceedings of the Twenty-Fourth International Conference on* Machine Learning (ICML), Corvallis, Oregon, USA, June 2007.



Roughly speaking, the minimization requirement during the search of the enclosing ball is relaxed. The calculation time should be reduced during the learning process. But the classification performance (accuracy) should not be affected.

We add the BVM component (SPV LEARNING tab) into the diagram. We set exactly the same settings as the CVM component above. We leave the default values for the additional parameters.

The calculation time is 84 seconds (27 for the model construction, 57 for the classification of all the instances). We obtain 2,710 support vectors. The resubstitution error rate is 0.56%.



The test error rate is 0.59% (88 misclassified among 14,951 instances).



5 Comparison with C-SVC [LIBSVM]

5.1 The C-SVC approach from the LIBSVM library

CVM and BVM are expected to provide results comparable to those of standard SVM implementation, while being able to handle large databases. To verify this, we apply the C-SVC from the state-of-the-art LIBSVM library on the same dataset.

We add the C-SVC component (SPV LEARNING tab) into the diagram. We set the same settings as for CVM and BVM.



We launch the calculations.

Characteristic	Value	Classifier performances							
# classes	2		Error rate 0.0061 🧲						
# support vectors	33057		Values pr	ediction	Confusion matrix				
# support vectors for	each class	Va	lue Recal	1-Precision		neg	pos	Sum	
# sv. for neg	31784		eg 0.9994		neg	48243	27	48270	
# sv. for pos	1273		os 0.8147		pos	274	1205	1479	
Computation time : 21					Sum	48517	1232	49749	

We obtain 33,057 support vectors in 2,125 seconds (\approx 35 minutes!). More specifically, the time for the model construction is 1,560 seconds; it is 564 seconds for the classification of the instances. The resubstitution error rate is 0.61%.

TANAGRA 1.4.44 - [Test 3] 📅 File Diagram Component Window Help _ 8 × 📽 🖪 🔛 ٦. Default title Test : Dataset (w8a.txt) 🗄 💉 Select first examples 1 Evaluation set : unselected examples 🖃 🚹 Define status 1 Supervised Learning 1 (CVM) 🖻 🏠 Define status 2 pred_SpvInstance_3 P Test 1 Error rate 0.0061 Supervised Learning 2 (BVM) Values prediction Confusion matrix Define status 3 P Test 2 **T**-Precision Value Rec pos Sum Supervised Learning 3 (C-SVC) 14489 neg 0.9994 0.0057 8 14497 neg 🗄 🚰 Define status 4 pos 0.8172 0.0211 371 454 pos 83 P Test 3 14572 14951 379 Sum Components Data visualization Statistics Nonparametric statistics Instance selection Feature construction Feature selection Regression Factorial analysis PLS Clustering Spv learning Meta-spv learning Spv learning assessment Scoring Association 1 Binary logistic regression ås C4.5 C-RT CS-MC4 🔣 BVM 🔀 C-PLS CS-CRT 🖄 C-SVC 4

The test error rate is 0.61%. This is similar to those of CVM and BVM.

5.2 Comparison

To compare the methods, we summarize the key results in the following table:

	Model cha	racteristics	Calculation time (sec.)			
Method	# support vector	Test error rate	Model	Classification		
			construction	(64,700 cases)		
CVM [LIBCVM]	3843	0.57 %	60	80		
BVM [LIBCVM]	2710	0.59 %	27	57		
C-SVC [LIBSVM]	33057	0.61 %	1560	564 ⁴		

With the same settings, and the same accuracy than C-SVC, the calculation time is divided by 27 for CVM; and by 57 for BVM. It seems also that these approaches can handle very large datasets (e.g. the KDD-99 database with n = 4,898,431 instances in the authors' experiments).

<u>Note</u>: To be honest, the differences are not always as spectacular. By changing the settings (e.g. gamma = 0, which is equivalent to set gamma = 1 / number of descriptors), we can obtain slightly less accurate classifiers, but the calculations times become very similar.

⁴ The calculation time is strongly penalized by the large number of support vectors. Speaking generally, fastness in classification phase depends on the number of supports vectors.

6 Conclusion

The CVM and BVM algorithms from the LIBCVM library seem particularly efficient. They are a very credible alternative to conventional implementations of the SVM algorithms in the treatment of large databases (with many instances). These methods are incorporated in Tanagra 1.4.44 (and later).