Novel Support Vector Machines for Diverse Learning Paradigms

Gabriella Melki, PhD. Candidate, Virginia Commonwealth University

Advisor: Dr. Alberto Cano
Assistant Professor
Virginia Commonwealth University

Advisor: Dr. Sebastián Ventura
Professor
University of Córdoba

VCU

Motivation

- **Large-scale learning**
  - Memory capacity
  - Computational time

- Extension to **diverse paradigms**:
  - Multiple-Target Learning
  - Multiple-Instance Learning
  - Data Stream Learning

- Our contributions aim to deal with these drawbacks by developing novel learning algorithm for **Support Vector Machines**.

Multiple-Target Learning

- **The goal:** build a model which predicts ***multiple*** outputs, or ***targets***, simultaneously.

$X \in \Re^{n \times d}$			Y_1	…	Y_m	
$x_{1,1}$	…	$x_{1,d}$	0.3466	…	2.9832	*Training Samples*
$x_{2,1}$	…	$x_{2,d}$	0.5493	…	4.9036	
…	…	…	…	…	…	
$x_{n,1}$	…	$x_{n,d}$	0.923	…	2.6783	
$\hat{x}_{n+1,1}$	…	$\hat{x}_{n+1,d}$	?	…	?	*Unknown Samples*
$\hat{x}_{n+2,1}$	…	$\hat{x}_{n+2,d}$	?	…	?	
Inputs			Outputs			

Multiple-Instance Learning

- **The goal:** build a model which, given an unseen ***Bag*** containing multiple instances, will predict the bag's label.

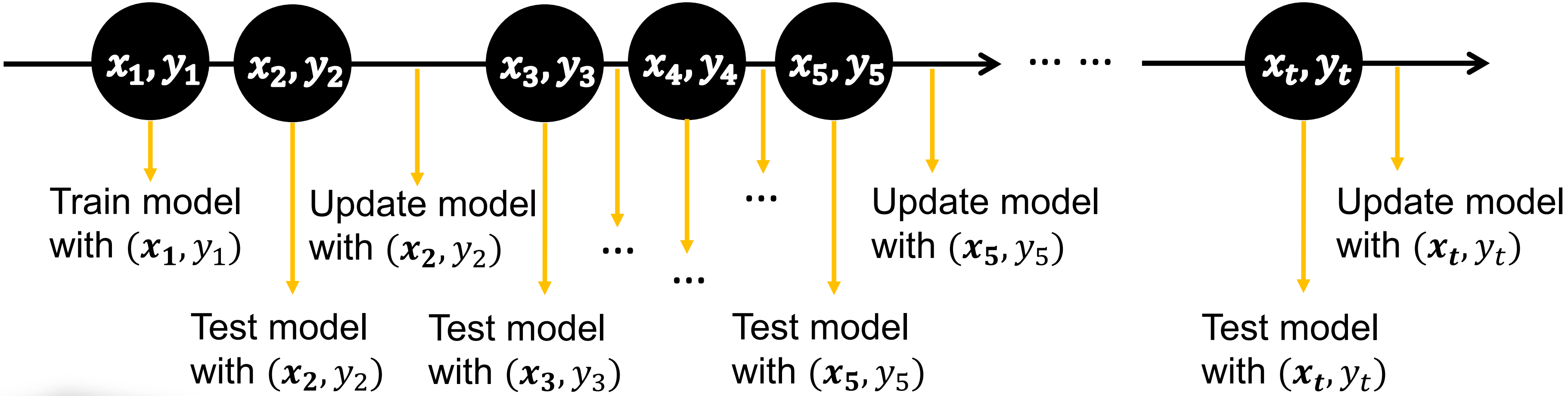
Training
Bags of instances

Unknown
Bag of instances

$\boldsymbol{B}$	$\boldsymbol{X} \in \mathbb{R}^{\boldsymbol{m} \times \boldsymbol{d}}$			$\boldsymbol{Y} \in \{\mathbf{1}, -\mathbf{1}\}^{\boldsymbol{n}}$
$\boldsymbol{B}_1$	$\boldsymbol{x}_{1,1}$	…	$\boldsymbol{x}_{1,d}$	$+1$
	$\boldsymbol{x}_{2,1}$	…	$\boldsymbol{x}_{2,d}$	
	$\boldsymbol{x}_{3,1}$	…	$\boldsymbol{x}_{3,d}$	
$\boldsymbol{B}_2$	$\boldsymbol{x}_{4,1}$	…	$\boldsymbol{x}_{4,d}$	-1
	$\boldsymbol{x}_{5,1}$	…	$\boldsymbol{x}_{5,d}$	
…	…	…	…	…
$\boldsymbol{B}_n$	$\boldsymbol{x}_{m,1}$	…	$\boldsymbol{x}_{m,d}$	$+1$
$\widehat{\boldsymbol{B}}_{n+1}$	$\widehat{\boldsymbol{x}}_{m+1,1}$	…	$\widehat{\boldsymbol{x}}_{m+1,d}$	?
	$\widehat{\boldsymbol{x}}_{m+2,1}$	…	$\widehat{\boldsymbol{x}}_{m+2,d}$	

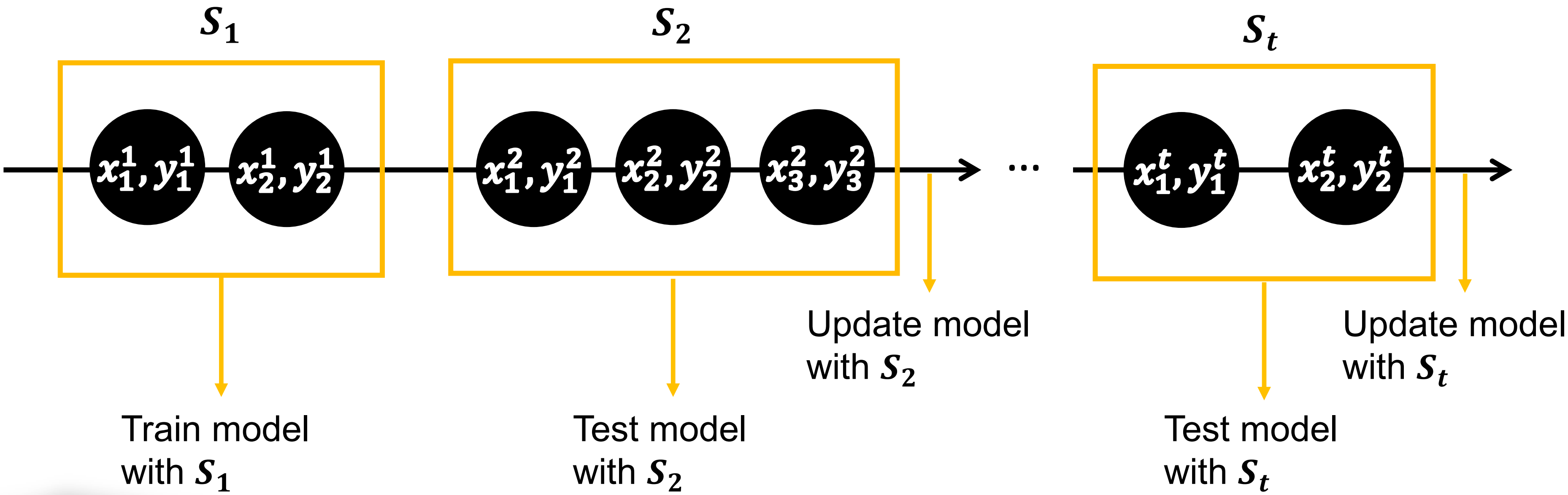
Data Stream Learning

- **The goal:** build a model from ***continuously*** and possibly ***drifting*** arriving data, which predicts the output of unseen arriving ***samples***.



Data Stream Learning

- **The goal:** build a model from ***continuously*** and possibly ***drifting*** arriving data, which predicts the output of unseen arriving ***batches***.



Contributions

- **Multi-Target** support vector regressors using maximum correlation chains:
  - *Support Vector Regressor* **(SVR)**
  - *Ensemble of Randomly Chained SVRs* **(SVRRC)**
  - *SVR using a single Maximally Correlated Chain* **(SVRCC)**
- **Multi-Instance** support vector machine bag-level formulation and algorithm:
  - *Multi-Instance Representative SVM* **(MIRSVM)**
- **Online**, i.e. stochastic, SVM with novel stopping criterion & update
  - *OnLine Learning Algorithm using Worst-Violators* **(OLLAWV)**
- **Data Stream** support vector machines for batched streams
  - *OnLine Learning Algorithm - List 2* **(OLLA-L2)** & **OLLAWV**

Agenda

Background

- Support Vector Machines
- NN & SVMs: Form vs. Norm
- SVM Solvers

Contributions

- Multi-Target SVR using Max Correlation Chains
- Multi-Instance SVM using Bag Representatives
- Novel OnLine SVM using Worst-Violators
- OLLAWV for Batched Data Streams

Conclusions

Future Work

- Background
- Algorithm
- Experimental Environment
- Results & Statistical Analysis
- Conclusions

Support Vector Machines (SVM)

- SVMs have been developed in order to target modern machine learning problems:
  - **learning from high dimensional & sparse data.**

- This has been achieved by introducing novel cost function.

- Well, what is the novel cost function and why is it so attractive?

NN & SVMs: Form vs. Norm

Classic Approach by NN

- **Form:** sum of weighted basis functions
- **Norm:**
  - Minimize **sum of errors squared** in output space
  - Model parameters **predefined** before training
  - Estimation error remains **fixed**
  - Training error is **minimized**

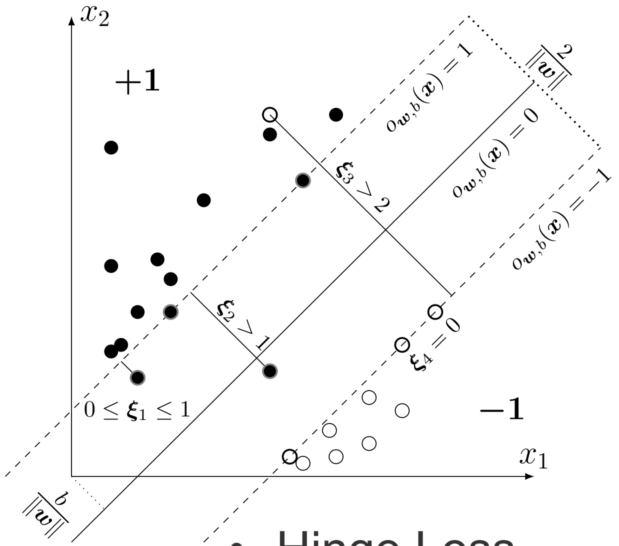
Support Vector Machines

- **Form:** sum of weighted basis functions
- **Norm:**
  - Maximize **margin** in input space (minimize $||w||^2$)
  - Model parameters **not predefined** & depend on data
  - Estimation error is **minimized**
  - Training error is **fixed**

Support Vector Machines

$$\min_{(\boldsymbol{w},b)\in\mathcal{H}_o\times\mathbb{R}} R \;=\; \frac{1}{2}||\boldsymbol{w}||^2 + C\sum_{i=1}^{n} L(y_i,\, o_{(w,b)}(\boldsymbol{x}_i))$$

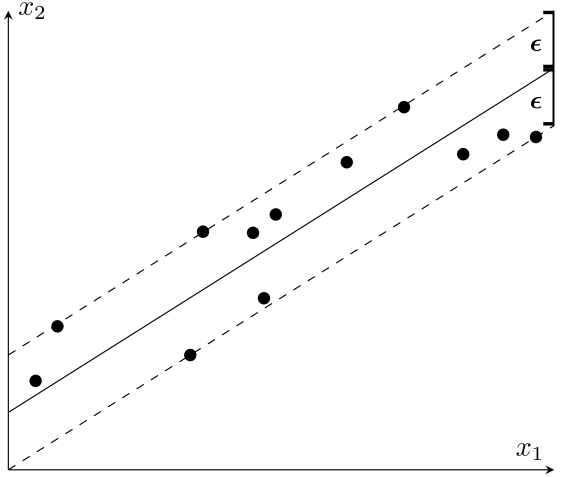
Classification



- Hinge Loss

$$L(y_i,\, o_{(w,b)}(\boldsymbol{x}_i)) = \max\left\{0, 1 - y_i o_{(w,b)}(\boldsymbol{x}_i)\right\}$$

Regression



- Epsilon-Insensitivity Loss

$$L(y_i,\, o_{(w,b)}(\boldsymbol{x}_i)) = \begin{cases} 0 & if |y_i - o_{(w,b)}(\boldsymbol{x_i})| \leq \epsilon \\ |y_i - o_{(w,b)}(\boldsymbol{x_i})| - \epsilon & \text{otherwise.} \end{cases}$$

Solving the SVM Problem

SVM Optimization			
Method	**Description**	**Memory**	**Big Data**
Interior Point	Direct precise solution	At least $O(n^2)$ memory	❌
Active Set	Constraint-based	$O(N_{FSV}^2)$ (free support vectors)	⚠️
Working Set	Iterative approximate solution	$O(n+q^2)$, q is the size of working set	✅

Multi-Target SVR using Maximum Correlation Chains

G. Melki et al. “Multi-target support vector regression via correlation regressor chains”. **Information Sciences**, vol. 415, pp. 53–69, 2017

Multi-Target Regression

Approach	Methods	Targets' Relationships Exploited	Run Time	Accuracy
Problem Transformation	Single Target	❌	⚠️	✅
	MTR Stacking	⚠️	⚠️	⚠️
	Regressor Chains	⚠️	⚠️	⚠️
Algorithm Adaptation	Statistical methods	✅	⚠️	⚠️
	Kernel methods	✅	✅	⚠️
	MTR Trees	✅	✅	✅

✅ - *good performance*
⚠️ - *dependent on # of targets & their correlations (if exist)*
❌ - *poor performance*

G. Melki et al. "Multi-target support vector regression via correlation regressor chains". **Information Sciences**, vol. 415, pp. 53–69, 2017

Base-Line Multi-Target SVR

$$\mathcal{D}_1 : [\boldsymbol{X}][\boldsymbol{Y}_1] \longrightarrow h_1 : \mathcal{D}_1 \to \hat{\boldsymbol{Y}}_{\mathbf{1}}$$

$$\mathcal{D}_2 : [\boldsymbol{X}][\boldsymbol{Y}_2] \longrightarrow h_2 : \mathcal{D}_2 \to \hat{\boldsymbol{Y}}_{\mathbf{2}}$$

$$\mathcal{D} : [\boldsymbol{X}][\boldsymbol{Y}] \qquad \vdots$$

$$\mathcal{D}_m : [\boldsymbol{X}][\boldsymbol{Y}_m] \longrightarrow h_m : \mathcal{D}_m \to \hat{\boldsymbol{Y}}_{\boldsymbol{m}}$$

Multi-Target Support Vector Regression (SVR)

Input: Training dataset $\mathcal{D}$

Output: ST models $h_j, j = 1, \ldots, m$

1: **for** $j = 1$ to m **do**

2: $\quad \mathcal{D}_j = \{\boldsymbol{X}, \boldsymbol{Y}_j\}$ ▷ Get ST data

3: $\quad h_j : \boldsymbol{X} \to \mathbb{R}$ ▷ Build ST model for the j^{th} target

4: **end for**

G. Melki et al. "Multi-target support vector regression via correlation regressor chains". **Information Sciences**, vol. 415, pp. 53–69, 2017

SVR with Random Chains (SVRRC)

$$\mathcal{D} : [\boldsymbol{X}][\boldsymbol{Y}_1 \boldsymbol{Y}_2 \boldsymbol{Y}_3]$$

$$\begin{array}{cccc}
[1,2,3] & [1,3,2] & \dots & [3,2,1] \\
\downarrow & \downarrow & & \downarrow \\
h_1 : [\boldsymbol{X}] \rightarrow \hat{\boldsymbol{Y}}_\mathbf{1} & h_1 : [\boldsymbol{X}] \rightarrow \hat{\boldsymbol{Y}}_\mathbf{1} & \dots & h_1 : [\boldsymbol{X}] \rightarrow \hat{\boldsymbol{Y}}_\mathbf{1} \\
\downarrow & \downarrow & & \downarrow \\
h_2 : [\boldsymbol{X}\boldsymbol{Y}_\mathbf{1}] \rightarrow \hat{\boldsymbol{Y}}_\mathbf{2} & h_2 : [\boldsymbol{X}\boldsymbol{Y}_\mathbf{1}] \rightarrow \hat{\boldsymbol{Y}}_\mathbf{3} & \dots & h_2 : [\boldsymbol{X}\boldsymbol{Y}_\mathbf{3}] \rightarrow \hat{\boldsymbol{Y}}_\mathbf{2} \\
\downarrow & \downarrow & & \downarrow \\
h_3 : [\boldsymbol{X}\boldsymbol{Y}_\mathbf{1}\boldsymbol{Y}_\mathbf{2}] \rightarrow \hat{\boldsymbol{Y}}_\mathbf{3} & h_3 : [\boldsymbol{X}\boldsymbol{Y}_\mathbf{1}\boldsymbol{Y}_\mathbf{3}] \rightarrow \hat{\boldsymbol{Y}}_\mathbf{2} & \dots & h_3 : [\boldsymbol{X}\boldsymbol{Y}_\mathbf{3}\boldsymbol{Y}_\mathbf{2}] \rightarrow \hat{\boldsymbol{Y}}_\mathbf{1}
\end{array}$$

Multi-Target SVR with Random-Chains (SVRRC)

Input: Training dataset $\mathcal{D}$, c random chains $\mathcal{C}$
Output: An ensemble of chained models $h_\mathcal{C}$

for each $\boldsymbol{C} \in \mathcal{C}$ **do** ▷ For each random chain
    $h_{\boldsymbol{C}} \leftarrow$ `buildChainedModel`$(\mathcal{D}, \boldsymbol{C})$ ▷ Build a chained model for chain $\boldsymbol{C}$
end for

SVR max-Correlation Chain (SVRCC)

$$\mathcal{D} : [\boldsymbol{X}][\boldsymbol{Y}_1 \boldsymbol{Y}_2 \boldsymbol{Y}_3] \xrightarrow[\frac{\mathbf{E}[(Y_i - \mu_i)(Y_j - \mu_j)]}{\sqrt{\mathbf{E}[(Y_i - \mu_i)(Y_i - \mu_i)]\mathbf{E}[(Y_j - \mu_j)(Y_j - \mu_j)]}}]{\textit{generate maximum correlation chain}} [1, 2, 3]$$

$$h_1 : [\boldsymbol{X}] \rightarrow \hat{\boldsymbol{Y}}_{\mathbf{1}} \longrightarrow h_2 : [\boldsymbol{X}\boldsymbol{Y}_{\mathbf{1}}] \rightarrow \hat{\boldsymbol{Y}}_{\mathbf{2}} \longrightarrow h_3 : [\boldsymbol{X}\boldsymbol{Y}_{\mathbf{1}}\boldsymbol{Y}_{\mathbf{2}}] \rightarrow \hat{\boldsymbol{Y}}_{\mathbf{3}}$$

Multi-Target SVR with max-Correlation Chain (SVRCC)

1: $\mathbf{P} = corrcoef(\boldsymbol{Y})$ ▷ Find correlation coefficient matrix for target variables
2: $\boldsymbol{C} = \sum_{i=1}^{n} \mathbf{P}_{ij}, \forall j = 1, \ldots, m$ ▷ Sum rows of the correlation matrix
3: $\boldsymbol{C} = \texttt{sort}\,(\boldsymbol{C}, \textbf{decreasing})$ ▷ Sort sums in decreasing order
4: $h_{\boldsymbol{C}} = \texttt{buildChainedModel}(\mathcal{D}, \boldsymbol{C})$ ▷ Build a max-correlation chained model

G. Melki et al. “Multi-target support vector regression via correlation regressor chains”. **Information Sciences**, vol. 415, pp. 53–69, 2017

Experimental Environment

SVM Hyper-Parameters	$C \in \{1, 10, 100\}$ $\gamma \in \{1^{-9}, 1^{-7}, 1^{-5}, 1^{-3}, 1^{-1}, 1, 5, 10\}$ $\epsilon \in \{0.01, 0.1, 0.2\}$
Average Correlation Coefficient (aCC ↑)	$\frac{1}{m}\sum_{j=1}^{m} \frac{\sum_{l=1}^{\mathcal{N}_{test}} (y_j^{(l)} - \bar{y}_j)(\hat{y}_j^{(l)} - \bar{\hat{y}}_j)}{\sqrt{\sum_{l=1}^{\mathcal{N}_{test}} (y_j^{(l)} - \bar{y}_j)^2 \sum_{l=1}^{\mathcal{N}_{test}} (\hat{y}_j^{(l)} - \bar{\hat{y}}_j)^2}}$
Mean Squared Error (MSE ↓)	$\frac{1}{m}\sum_{j=1}^{m} \frac{1}{\mathcal{N}_{test}} \sum_{l=1}^{\mathcal{N}_{test}} (y_j^{(l)} - \hat{y}_j^{(l)})^2$
Average Root Mean Squared Error (aRMSE ↓)	$\frac{1}{m}\sum_{j=1}^{m} \sqrt{\frac{\sum_{l=1}^{\mathcal{N}_{test}} (y_j^{(l)} - \hat{y}_j^{(l)})^2}{\mathcal{N}_{test}}}$
Average Relative Root Mean Squared Error (aRRMSE ↓)	$\frac{1}{m}\sum_{j=1}^{m} \sqrt{\frac{\sum_{l=1}^{\mathcal{N}_{test}} (y_j^{(l)} - \hat{y}_j^{(l)})^2}{\sum_{l=1}^{\mathcal{N}_{test}} (y_j^{(l)} - \bar{y}_j)^2}}$

Multi-Target (MT) Regression datasets

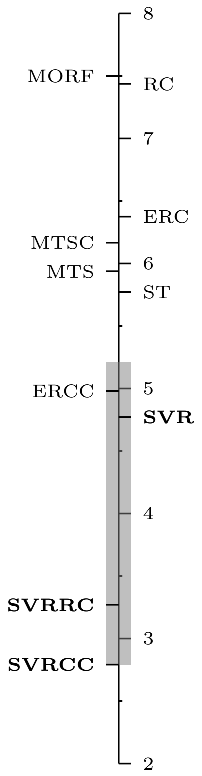
Dataset	Samples (n)	Attributes (d)	Targets (m)
EDM	145	16	2
Enb	768	8	2
Jura	359	11	7
Osales	639	413	12
Scpf	1137	23	3
Slump	103	7	3
Solar Flare 1	323	10	3
Solar Flare 2	1,066	10	3
Water Quality	1,060	16	14
OES97	323	263	16
OES10	403	298	16
ATP1d	201	411	6
ATP7d	188	411	6
Andro	49	30	6
Wisconsin Cancer	198	34	2
Stock	950	10	3
California Housing	20,640	7	2
Puma8NH	8,192	8	3
Puma32H	8,192	32	6
Friedman	500	25	6
Polymer	41	10	4
M5SPEC	80	700	3
MP5SPEC	80	700	3
MP6SPEC	80	700	3

G. Melki et al. “Multi-target support vector regression via correlation regressor chains”. **Information Sciences**, vol. 415, pp. 53–69, 2017

Results

Average Relative Root Mean Square Error (aRRMSE) for MT regressors

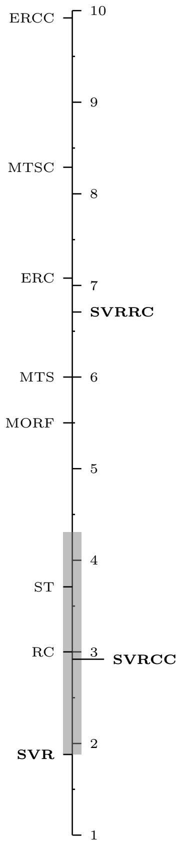
Datasets	MORF	ST	MTS	MTSC	RC	ERC	ERCC	SVR	SVRRC	SVRCC
Slump	0.6939	0.6886	0.6690	0.6938	0.7019	0.7022	0.6886	0.5765	**0.5545**	0.5560
Polymer	0.6159	0.5971	0.5778	0.6493	0.6270	0.6544	0.6131	0.5573	0.5253	**0.5116**
Andro	0.5097	0.5979	0.5155	0.5633	0.5924	0.5885	0.5666	0.4856	0.4651	**0.4455**
EDM	0.7337	0.7442	0.7413	0.7446	0.7449	0.7452	0.7443	0.7058	0.7070	**0.6978**
Solar Flare 1	1.3046	1.1357	1.1168	1.0758	0.9951	1.0457	1.0887	0.9917	0.9455	**0.9320**
Jura	0.5969	0.5874	0.5906	0.5892	0.5910	0.5896	0.5880	0.5952	**0.5764**	0.5885
Enb	0.1210	0.1165	0.1231	0.1211	0.1268	0.1250	0.1139	0.0977	0.0910	**0.0899**
Solar Flare 2	1.4167	1.1503	**0.9483**	1.0840	1.0092	1.0522	1.0928	1.0385	1.0253	1.0298
Wisconsin Cancer	0.9413	0.9314	0.9308	0.9336	**0.9305**	0.9313	0.9323	0.9555	0.9483	0.9427
California Housing	0.6611	0.6447	0.6974	0.6630	0.7131	0.6690	0.6146	0.6130	0.5945	**0.5852**
Stock	0.1653	0.1844	0.1787	0.1803	0.1802	0.1789	0.1752	0.1364	**0.1337**	0.1388
SCPF	0.8273	0.8348	0.8436	0.8308	0.8263	0.8105	0.8290	0.8164	0.8037	**0.8013**
Puma8NH	0.7858	0.8142	0.8118	0.8311	0.8199	0.8205	0.8207	**0.7655**	0.7744	0.7676
Friedman	0.9394	0.9214	0.9231	0.9210	0.9231	0.9209	0.9204	0.9218	0.9208	**0.9196**
Puma32H	0.9406	**0.8713**	0.8727	0.8791	0.8752	0.8729	0.8740	0.9364	0.9367	0.9319
Water Quality	**0.8994**	0.9085	0.9109	0.9093	0.9121	0.9097	0.9057	0.9343	0.9310	0.9045
M5SPEC	0.5910	0.5523	0.5974	0.5671	0.5552	0.5542	0.5558	0.2951	0.2935	**0.2925**
MP5SPEC	0.5522	0.5120	0.5683	0.5133	0.5145	0.5143	0.5119	0.2484	**0.2323**	0.2358
MP6SPEC	0.5553	0.5152	0.5686	0.5119	0.5198	0.5187	0.5109	0.2850	0.2669	**0.2623**
ATP7d	0.5563	0.5308	**0.5141**	0.5142	0.5558	0.5397	0.5182	0.5455	0.5371	0.5342
OES97	0.5490	0.5230	0.5229	0.5217	0.5239	0.5237	0.5222	0.4641	**0.4618**	0.4635
Osales	0.7596	0.7471	**0.7086**	0.7268	0.8318	0.7258	0.7101	0.7924	0.7924	0.7811
ATP1d	0.4173	0.3732	0.3733	0.3712	0.3790	**0.3696**	0.3721	0.3773	0.3707	0.3775
OES10	0.4518	0.4174	0.4176	0.4171	0.4178	0.4180	0.4166	0.3570	0.3555	**0.3538**
Average	0.6910	0.6625	0.6551	0.6589	0.6611	0.6575	0.6536	0.6039	0.5935	**0.5893**
Ranks	7.5000	5.7708	5.9375	6.1667	7.4375	6.3750	4.9792	4.7708	3.2708	**2.7917**



Results

Run Time (seconds) for MT regressors

Datasets	MORF	ST	MTS	MTSC	RC	ERC	ERCC	SVR	SVRRC	SVRCC
Slump	38.1	2.6	9.9	15.9	1.8	11.1	50.5	**0.6**	1.9	0.7
Polymer	7.6	2.7	9.1	15.5	1.9	14.9	80.5	**0.5**	2.6	**0.5**
Andro	25.7	4.4	15.0	34.2	3.4	33.2	197.9	**1.1**	6.2	**1.1**
EDM	24.8	2.8	9.4	18.1	2.1	5.8	19.0	**0.9**	1.0	**0.9**
Solar Flare 1	34.1	3.5	13.6	26.7	2.7	17.7	86.9	**2.3**	9.3	2.6
Jura	64.3	7.9	31.8	74.3	6.4	43.5	254.2	**4.7**	18.7	5.3
Enb	71.4	6.6	26.1	63.6	**5.4**	15.6	69.6	11.3	17.7	15.9
Solar Flare 2	55.4	7.4	30.7	68.0	**6.3**	42.9	241.5	9.4	53.5	15.6
Wisconsin Cancer	51.4	6.1	21.9	53.7	4.9	14.8	61.6	**2.0**	2.4	**2.0**
California Housing	93.0	9.7	34.8	75.9	**8.2**	21.3	102.0	15.8	25.2	23.6
Stock	93.7	11.7	46.8	96.7	**11.0**	75.4	427.3	18.5	90.5	26.3
SCPF	66.3	19.3	65.9	176.3	**15.0**	104.2	734.2	32.8	162.8	48.8
Puma8NH	130.4	29.7	106.7	288.6	**27.9**	201.6	1227.7	94.1	516.6	177.1
Friedman	79.5	27.0	81.2	258.3	25.0	273.7	2871.6	**12.3**	322.3	18.8
Puma32H	93.9	68.1	181.0	635.0	87.7	667.9	6087.0	**32.2**	1018.7	53.1
Water Quality	108.4	**93.1**	262.1	912.3	127.2	925.4	10993.3	110.2	2567.9	189.5
M5SPEC	89.8	68.9	166.3	604.6	73.7	262.3	3132.1	**39.2**	546.7	45.1
MP5SPEC	84.5	94.6	221.2	888.3	91.5	557.0	6864.1	**49.3**	1132.1	58.4
MP6SPEC	90.3	93.4	212.6	871.0	89.1	557.6	6761.3	**47.2**	1227.1	58.5
ATP7d	**70.5**	262.6	452.1	2319.8	242.1	1779.2	24373.8	80.0	1897.4	136.5
OES97	**83.4**	485.3	1146.6	4928.9	499.8	5315.0	58072.1	148.2	3759.1	342.6
Osales	**92.0**	1094.8	2340.7	8322.2	986.5	11361.2	122265.3	437.0	4830.1	843.6
ATP1d	**70.7**	272.9	476.5	2568.9	261.9	2138.9	26768.9	95.0	2127.8	174.4
OES10	**90.0**	738.9	1633.6	6682.9	688.5	7150.8	83533.1	229.1	5419.4	577.1
Average	71.2	142.2	316.5	1250.0	136.2	1316.3	14803.2	**61.4**	1073.2	117.4
Ranks	5.5	3.71	6.0	8.29	3.0	7.08	9.92	**1.88**	6.71	2.92



Multi-Target Regression Conclusions

- **SVR –** Results show SVR's superior performance as a **base-line model**, rather than regression trees, as used in MORF.

- **SVRRC –** Results highlighted the importance **exploiting relationships** among the target variables during training.

- **SVRCC –** Results show it's superiority:
  - Ranked the **first** in all quality metrics & **second** best in terms of run time.
  - SVRCC performed **better** than SVR & SVRRC
  - This shows that the targets' **maximum correlation does positively contribute toward model training**

G. Melki et al. "Multi-target support vector regression via correlation regressor chains". **Information Sciences**, vol. 415, pp. 53–69, 2017

Multi-Instance SVM using Bag Representatives

G. Melki et al. “MIRSVM: Multi-Instance Support Vector Machine with Bag Representatives”. **Pattern Recognition**, vol. 79, 228-241, 2018

Multi-Instance Classification

Paradigms

Instance Space
- Collective assumption
- Standard MI assumption

Bag Space
- Distance between bags

Embedded Space
- Vocabulary based
- Not Vocabulary based

✅ bags contain large **number** of instances

❌ bags of different **classes** contain similar instances

❌ bags contain large **number** of instances

✅ bags of different **classes** contain similar instances

Multi-Instance Classification

Paradigms

- **Instance Space**
  - Collective assumption
  - Standard MI assumption
- **Bag Space**
  - Distance between bags
- **Embedded Space**
  - Vocabulary based
  - Not Vocabulary based

✅ bags contain large **number** of instances
❌ bags of different **classes** contain similar instances

❌ bags contain large **number** of instances

✅ bags of different **classes** contain similar instances

Multi-Instance Classification

Paradigms

Instance Space

Collective assumption

Standard MI assumption

✅ bags contain large **number** of instances

❌ bags of different classes contain similar instances

$$Y_I = \begin{cases} +1 & \text{if } \exists\, y_i = +1, \forall i \in I \\ -1 & \text{otherwise.} \end{cases}$$

- Inspired by MI-SVM (Andrews et. al.)
  - Mixed-integer QP problem
  - Solved heuristically using **positive** bag-representatives

✅ When positive bags contain a **sparse number** of positive instances

❌ **Class imbalance** with representatives

G. Melki et al. “MIRSVM: Multi-Instance Support Vector Machine with Bag Representatives”. **Pattern Recognition**, vol. 79, 228-241, 2018

MIRSVM: Formulations

Primal

$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \frac{1}{2}||\boldsymbol{w}||^2 + C\sum_I \xi_I,$$

$$\text{s.t. } Y_I(\langle \boldsymbol{w}, \boldsymbol{x}_{s_I}\rangle + b) \geq 1 - \xi_I, \ \forall I \in \{1,\ldots,n\},$$

$$\xi_I \geq 0, \ \forall I \in \{1,\ldots,n\},$$

$$s_I = \underset{i\in I}{\text{argmax}}(\langle \boldsymbol{w}, \ \boldsymbol{x}_i\rangle + b), \ \forall I \in \{1,\ldots,n\}.$$

- $\boldsymbol{S}$ vector of bag representative indices
- $\boldsymbol{x}_{\boldsymbol{S}_{\boldsymbol{I}}}$ representative instance of bag $\boldsymbol{B}_{\boldsymbol{I}}$

Dual

$$\max_{\boldsymbol{\alpha}} \sum_I \alpha_I - \frac{1}{2}\sum_I \sum_{K\in I} \alpha_I\alpha_K Y_I Y_K \mathcal{K}(\boldsymbol{x}_{s_I}, \boldsymbol{x}_{s_K})$$

$$\text{s.t.} \sum_I \alpha_I Y_I = 0,$$

$$0 \leq \alpha_I \leq C, \ \forall I \in \{1,\ldots,n\},$$

$$s_I = \underset{i\in I}{\text{argmax}}(\boldsymbol{o}_I), \ \forall I \in \{1,\ldots,n\}.$$

- $\boldsymbol{o}_{\boldsymbol{I}} \in \mathbb{R}^{|\boldsymbol{B}_{\boldsymbol{I}}|}$ output of bag $\boldsymbol{B}_{\boldsymbol{I}}$

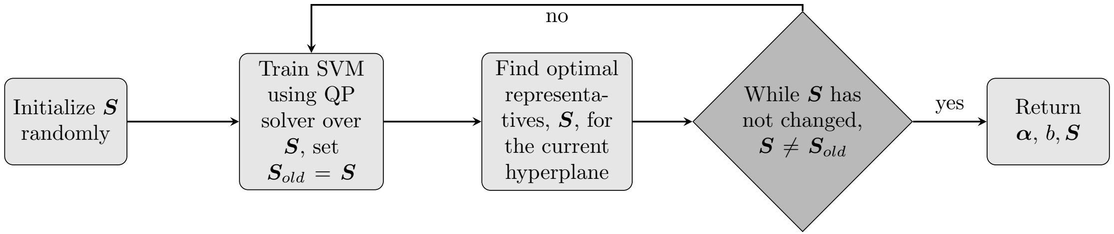
  $$\boldsymbol{o}_I = \mathcal{K}(\mathcal{B}_I, \boldsymbol{X}_S) * (\boldsymbol{\alpha} \cdot \boldsymbol{Y}_S) + b$$

- $b \in \mathbb{R}$ bias term

  $$b = \frac{1}{n_{\boldsymbol{sv}}}\sum_{\boldsymbol{sv}} Y_{\boldsymbol{sv}} - \mathcal{K}(\boldsymbol{X}_{\boldsymbol{sv}}, \boldsymbol{X}_{\boldsymbol{sv}}) * (\boldsymbol{\alpha}_{\boldsymbol{sv}} \cdot Y_{\boldsymbol{sv}})$$

G. Melki et al. “MIRSVM: Multi-Instance Support Vector Machine with Bag Representatives”. **Pattern Recognition**, vol. 79, 228-241, 2018

MIRSVM: Algorithm

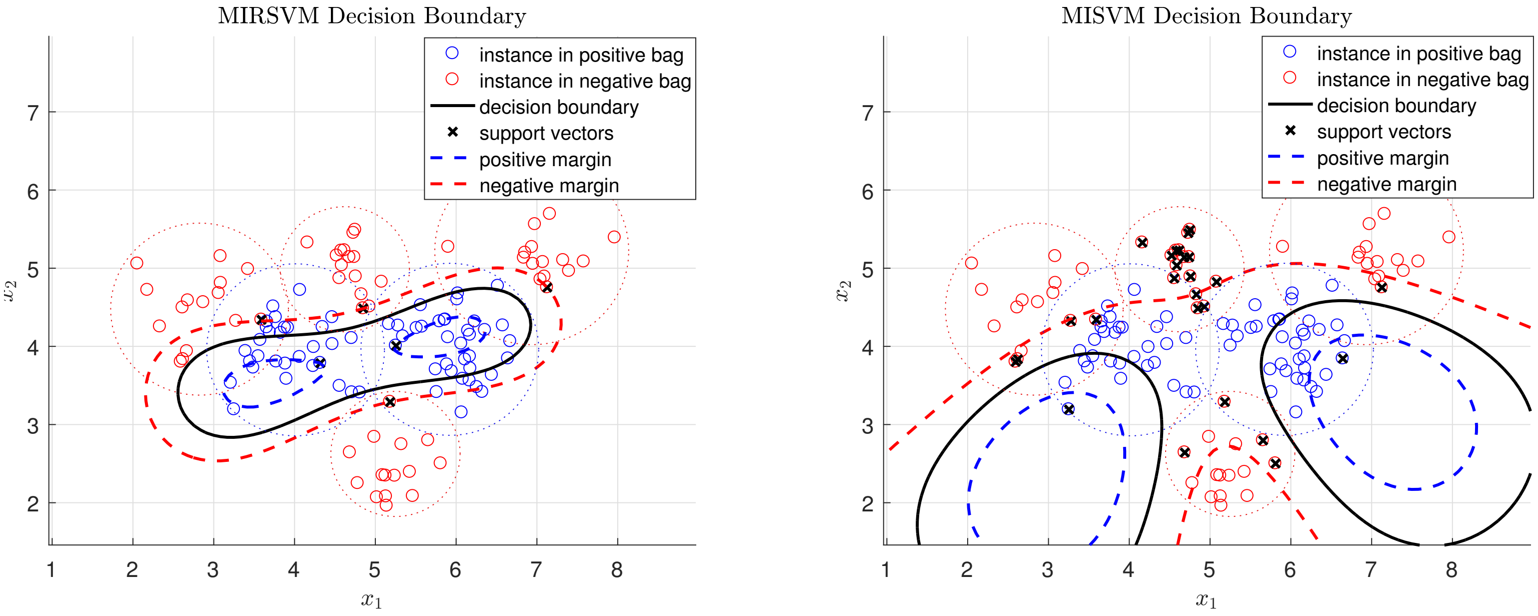


Multi-Instance Representative SVM (MIRSVM)

Input: Training dataset $\mathcal{D}$, SVM Parameters C and σ
Output: SVM model parameters $\boldsymbol{\alpha}$ and b, Bag Representative IDs $\boldsymbol{S}$

for $I \in \{1, \ldots, n\}$ **do**
    $\boldsymbol{S}_I \leftarrow \text{rand}\,(|\mathcal{B}_I|, 1, 1)$ ▷ Assign each bag a random instance
end for
while $\boldsymbol{S} \neq \boldsymbol{S}_{old}$ **do**
    $\boldsymbol{S}_{old} \leftarrow \boldsymbol{S}$
    $\boldsymbol{X}_S \leftarrow \boldsymbol{X}(\boldsymbol{S}),\ \boldsymbol{Y}_S \leftarrow \boldsymbol{Y}(\boldsymbol{S})$ ▷ Initialize the representative dataset
    $\boldsymbol{G} \leftarrow (\boldsymbol{Y}_S \times \boldsymbol{Y}_S) \cdot \mathcal{K}\,(\boldsymbol{X}_S,\ \boldsymbol{X}_S,\ \sigma)$ ▷ Build Gram matrix
    $\boldsymbol{\alpha} \leftarrow \text{quadprog}\,(\boldsymbol{G}, -\mathbf{1}^n, \boldsymbol{Y}_S, \mathbf{0}^n, \mathbf{0}^n, \boldsymbol{C}^n)$ ▷ Solve QP Problem
    $\boldsymbol{sv} \leftarrow \text{find}\,(0 < \boldsymbol{\alpha} \leq C)$ ▷ Get the support vector indices
    $n_{\boldsymbol{sv}} \leftarrow \text{count}\,(0 < \boldsymbol{\alpha} \leq C)$ ▷ Get the number of support vectors
    $b \leftarrow \frac{1}{n_{\boldsymbol{sv}}} \sum_{i=1}^{n_{\boldsymbol{sv}}} (\boldsymbol{Y}_{\boldsymbol{sv}} - \boldsymbol{G}_{\boldsymbol{sv}} * (\boldsymbol{\alpha}_{\boldsymbol{sv}} \cdot \boldsymbol{Y}_{\boldsymbol{sv}}))$ ▷ Calculate the bias term
    for $I \in \{1, \ldots, n\}$ **do**
        $\boldsymbol{G}_I \leftarrow (Y_I \times \boldsymbol{Y}_S) \cdot \mathcal{K}\,(\mathcal{B}_I,\ \boldsymbol{X}_S,\ \sigma)$
        $\boldsymbol{S}_I \leftarrow \text{argmax}_{i \in I}\,(\boldsymbol{G}_I * \boldsymbol{\alpha} + b)$ ▷ Select optimal bag-representatives
    end for
end while

Comparison of MIRSVM and MISVM



G. Melki et al. "MIRSVM: Multi-Instance Support Vector Machine with Bag Representatives". **Pattern Recognition**, vol. 79, 228-241, 2018

Experimental Environment

SVM Hyper-Parameters	$C \in \{0.1, 1, 10, 100, 1000, 10000\}$ $\sigma \in \{0.1, 0.5, 1, 2, 5, 10\}$
Accuracy	$\frac{TP+TN}{n'}$
Precision	$\frac{TP}{TP+FP}$
Recall	$\frac{TP}{TP+FN}$
Cohen's Kappa Rate	$\frac{n' - \frac{(TP+FN)*(TP+FP)}{n'}}{1 - \frac{(TP+FN)*(TP+FP)}{n'}}$
Area Under ROC Curve	$\frac{1 + \frac{TP}{TP+FN} - \frac{FP}{FP+TN}}{2}$

Multi-Instance (MI) Classification datasets

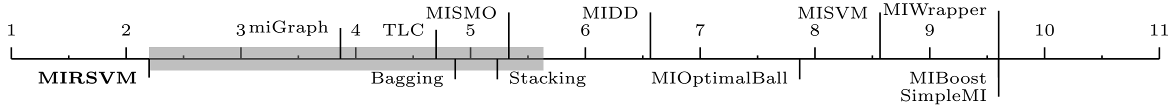
Dataset	Attributes	Positive Bags	Negative Bags	Total	Instances	Avg. Bag Size
Suramin	20	7	6	13	2898	222.92
EastWest	24	10	10	20	213	10.65
WestEast	24	10	10	20	213	10.65
Musk1	166	47	45	92	476	5.17
Musk2	166	39	62	101	6728	66.61
Webmining	5863	21	92	113	3423	30.29
Mutagenesis-atoms	10	125	63	188	1618	8.61
Mutagenesis-bonds	16	125	63	188	4081	21.71
Mutagenesis-chains	24	125	63	188	5424	28.85
TRX	8	25	168	193	26611	137.88
Elephant	230	100	100	200	1391	6.96
Fox	230	100	100	200	1320	6.60
Tiger	230	100	100	200	1188	5.94
Component	200	423	2707	3130	36894	11.79
Function	200	443	4799	5242	55536	10.59

G. Melki et al. "MIRSVM: Multi-Instance Support Vector Machine with Bag Representatives". **Pattern Recognition**, vol. 79, 228-241, 2018

Results

Accuracy for MI classifiers

Datasets	MIRSVM	miGraph	MIBoost	MIOptimalBall	MIDD	MIWrapper	MISMO	MISVM	SimpleMI	TLC	Bagging	Stacking
suramin	0.8000	**0.8462**	0.5000	0.7250	0.4250	0.5000	0.7250	0.5000	0.5000	0.6000	0.6650	0.4615
eastWest	**0.8000**	0.7000	0.5000	0.7250	0.6125	0.5000	0.7125	0.5625	0.5000	0.6000	0.6000	0.4500
westEast	0.7500	0.7500	0.5000	0.3750	0.4500	0.5000	0.7375	0.4125	0.5000	0.5625	**0.9649**	0.6375
musk1	**0.9022**	0.8152	0.5109	0.7717	0.8804	0.5109	0.7826	0.7609	0.5109	0.8587	0.8142	0.8587
musk2	0.8218	0.7426	0.6139	0.7723	0.7228	0.6139	0.7030	0.7129	0.6139	0.6238	**0.8756**	0.6733
webmining	0.8500	0.8142	0.8142	0.7699	0.8142	0.8142	0.8407	0.6903	0.8142	0.8142	**0.9358**	0.8053
trx	0.8860	0.8964	0.8705	**0.9016**	0.8808	0.8705	0.8705	0.8705	0.8705	0.8756	0.6450	0.8860
mutagenesis-atoms	0.7714	0.7606	0.6649	0.6436	0.7074	0.6649	0.6915	0.6649	0.6649	**0.7766**	**0.7766**	0.7606
mutagenesis-bonds	0.8252	0.7872	0.6649	0.6915	0.7713	0.6649	0.7979	0.6649	0.6649	0.8351	0.8351	**0.8564**
mutagenesis-chains	**0.8411**	0.7926	0.6649	0.6702	0.7766	0.6649	0.8351	0.6649	0.6649	0.8404	0.8404	0.8351
tiger	0.7750	0.7950	0.5000	0.5000	0.7100	0.5000	0.7200	0.7550	0.5000	0.6650	**0.8000**	0.7250
elephant	**0.8300**	**0.8300**	0.5000	0.5000	0.7900	0.5000	0.8100	0.8000	0.5000	0.8000	0.5625	0.8250
fox	0.6550	0.6300	0.5000	0.5000	0.5800	0.5000	0.5250	0.4750	0.5000	0.6450	**0.8587**	0.6500
component	**0.9366**	0.9153	0.8649	0.8696	0.8780	0.8649	0.8968	0.8703	0.8649	0.9358	0.6000	0.9355
function	0.9523	0.9405	0.9155	0.9138	0.9193	0.9155	0.9376	0.9195	0.9155	**0.9649**	0.6238	0.9647
Average	**0.8264**	0.8010	0.6390	0.6886	0.7279	0.6390	0.7724	0.6883	0.6390	0.7598	0.7598	0.7550
Rank	**2.2000**	3.8667	9.6000	7.8667	6.5667	9.6000	5.3333	8.5667	9.6000	4.7000	4.8667	5.2333



G. Melki et al. "MIRSVM: Multi-Instance Support Vector Machine with Bag Representatives". **Pattern Recognition**, vol. 79, 228-241, 2018

Results

Run Time (seconds) for MI classifiers

Datasets	MIRSVM	miGraph	MIBoost	MIOptimalBall	MIDD	MIWrapper	MISMO	MISVM	SimpleMI	TLC	Bagging	Stacking
suramin	**0.1**	19.7	8.8	30.5	7922.0	9.5	52.3	333.9	7.2	35.5	183.0	90.6
eastWest	**0.1**	3.0	5.5	9.4	217.1	6.3	14.8	21.4	5.8	15.4	15.4	15.2
westEast	**0.1**	2.8	6.5	7.8	79.7	6.5	14.7	99.5	6.0	16.6	12128.1	10.8
musk1	**0.4**	56.8	13.4	32.1	3542.6	20.6	89.7	198.4	11.1	93.0	86272.6	759.5
musk2	**2.3**	452.3	97.3	782.9	126016.8	208.3	1799.4	26093.5	16.1	1772.2	2229.3	16759.0
webmining	**300.6**	302.5	45745.4	60474.8	47601.4	68736.7	51923.6	105622.3	2685.9	86272.6	9861.5	592948.9
trx	61.8	2206.4	17.6	682.3	339110.5	19.3	8670.3	134622.1	**7.4**	2229.3	243.3	11927.9
mutagenesis-atoms	9.8	193.1	8.8	99.2	2623.0	8.0	55.0	53.5	**6.4**	44.0	44.0	153.9
mutagenesis-bonds	**8.3**	410.3	10.2	310.2	17538.7	12.3	457.4	2794.8	8.4	131.1	131.1	853.1
mutagenesis-chains	19.3	513.4	12.0	525.0	48982.7	14.8	2451.9	6637.4	**7.2**	224.4	224.4	1619.0
tiger	29.5	302.8	44.5	157.8	23220.5	56.2	208.0	608.8	**16.2**	183.0	212.1	1085.0
elephant	47.7	306.7	45.5	243.9	56456.2	69.7	232.1	1114.3	20.8	212.1	**16.6**	1462.2
fox	81.0	303.1	44.2	206.1	27773.8	66.0	369.6	891.5	**23.5**	243.3	93.0	1729.1
component	231.7	3091.0	572.5	228209.6	96263.9	1096.9	629366.4	37224.6	144.0	9861.5	**35.5**	79149.8
function	740.3	8162.7	935.5	768458.0	350124.7	1887.5	1052225.3	565026.4	**232.8**	12128.1	1772.2	185918.5
Average	**102.2**	1088.4	3171.2	70682.0	76498.2	4814.6	116528.7	58756.2	213.3	7564.1	7564.1	59632.2
Rank	2.3	6.2	3.1	7.2	11.1	4.3	8.5	10.1	**1.9**	7.2	6.5	9.7

G. Melki et al. "MIRSVM: Multi-Instance Support Vector Machine with Bag Representatives". **Pattern Recognition**, vol. 79, 228-241, 2018

Multi-Instance Classification Conclusions

- **MIRSVM** – Results show the better performance of MIRSVM, overall:
  - Outperformed MISVM & MISMO
    - Bag-representative selector: remedied **class imbalance** resulting in improved accuracy
    - Fast **convergence**
- Statistical analysis showed that:
  - **bag-level** based & **ensemble** learners had the best performance
  - **instance-level** based learners performed poorly in comparison or were deemed as strongly biased and unstable classifiers
- MIRSVM performs statistically better, neither compromising accuracy nor run time while displaying a robust performance across all datasets.

G. Melki et al. "MIRSVM: Multi-Instance Support Vector Machine with Bag Representatives". **Pattern Recognition**, vol. 79, 228-241, 2018

Online SVM using Worst-Violators

G. Melki et al. “OLLAWV: OnLine Learning Algorithm using Worst-Violators”. **Applied Soft Computing**, vol. 66, 384-393, 2018

Online, i.e. Stochastic, Learning

- Stochastic Gradient Descent
- Primal Optimization
- Similar to the perceptron algorithm, but non-linear
- Early stopping → control margin size

Advantages	Disadvantages
✅ Simple implementation	❌ No meaningful stopping criterion
✅ Computationally efficient when reaching predefined risk	❌ Computationally complex when number of samples too large

G. Melki et al. "OLLAWV: OnLine Learning Algorithm using Worst-Violators". **Applied Soft Computing**, vol. 66, 384-393, 2018

Online, i.e. Stochastic, Learning

- Rather than minimizing:

$$\min_{(\boldsymbol{w},b)\in\mathcal{H}_o\times\mathbb{R}} R \;=\; \frac{1}{2}||\boldsymbol{w}||^2 + C\cancel{\sum_{i=1}^{n}} L(y_i,\, o_{(w,b)}(\boldsymbol{x}_i)),$$

instead, the following is minimized:

$$\min_{(\boldsymbol{w},b)\in\mathcal{H}_o\times\mathbb{R}} R \;=\; \frac{1}{2}||\boldsymbol{w}||^2 + CL(y_j,\, o_{(w,b)}(\boldsymbol{x}_j)).$$

- The term *stochastic* stems from iterating over data point $\boldsymbol{x_j}$ to the data point $\boldsymbol{x_{j+1}}$ **randomly**, while **minimizing** $\boldsymbol{R}$, by updating the weight vector $\boldsymbol{w}$ just as the perceptron, i.e. using *gradient descent*

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta\frac{\partial R}{\partial \boldsymbol{w}}$$

G. Melki et al. "OLLAWV: OnLine Learning Algorithm using Worst-Violators". **Applied Soft Computing**, vol. 66, 384-393, 2018

OLLAWV: Formulation

- Primal SVM optimization problem:

$$\min_{\boldsymbol{w}\in\mathcal{H}_o\times\mathbb{R}} R \;=\; \frac{1}{2}||\boldsymbol{w}||^2 + C\sum_{i=1}^{n}\max\left\{0, 1 - y_i o_{(w)}(\boldsymbol{x}_i)\right\}$$

- Weight vector update:

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + \eta \begin{cases} C y_i \boldsymbol{x_i} - \boldsymbol{w} & y_i o_i < 1 \\ -\boldsymbol{w} & \text{otherwise} \end{cases}$$

- Representer Theorem $\boldsymbol{w} = \sum_{i=1}^{n} \alpha_i \phi(\boldsymbol{x_i})$:

$$\sum_{i=1}^{n}\alpha_i\phi(\boldsymbol{x}_i) \leftarrow \sum_{i=1}^{n}\alpha_i\phi(\boldsymbol{x}_i) + \eta \begin{cases} C y_i \phi(\boldsymbol{x_i}) - \sum_{i=1}^{n}\alpha_i\phi(\boldsymbol{x}_i) & y_i o_i < 1 \\ -\sum_{i=1}^{n}\alpha_i\phi(\boldsymbol{x}_i) & \text{otherwise} \end{cases}$$

- Stochastic update:

$$\forall i : \alpha_i \leftarrow \alpha_i + \eta \begin{cases} (C y_i - \alpha_i) & y_i o_i < 1 \\ (-\alpha_i) & \text{otherwise} \end{cases}$$

G. Melki et al. “OLLAWV: OnLine Learning Algorithm using Worst-Violators”. **Applied Soft Computing**, vol. 66, 384-393, 2018

OLLAWV: Formulation

- Primal SVM optimization problem:

$$\min_{\boldsymbol{w} \in \mathcal{H}_o \times \mathbb{R}} R = \frac{1}{2}||\boldsymbol{w}||^2 + C \sum_{i=1}^{n} \max\left\{0, 1 - y_i o_{(w)}(\boldsymbol{x}_i)\right\}$$

- Weight vector update:

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + \eta \begin{cases} C y_i \boldsymbol{x_i} - \boldsymbol{w} & y_i o_i < 1 \\ -\boldsymbol{w} & \text{otherwise} \end{cases}$$

- Representer Theorem $\boldsymbol{w} = \sum_{i=1}^{n} \alpha_i \phi(\boldsymbol{x_i})$:

$$\sum_{i=1}^{n} \alpha_i \phi(\boldsymbol{x}_i) \leftarrow \sum_{i=1}^{n} \alpha_i \phi(\boldsymbol{x}_i) + \eta \begin{cases} C y_i \phi(\boldsymbol{x_i}) - \sum_{i=1}^{n} \alpha_i \phi(\boldsymbol{x}_i) & y_i o_i < 1 \\ -\sum_{i=1}^{n} \alpha_i \phi(\boldsymbol{x}_i) & \text{otherwise} \end{cases}$$

- Stochastic update:

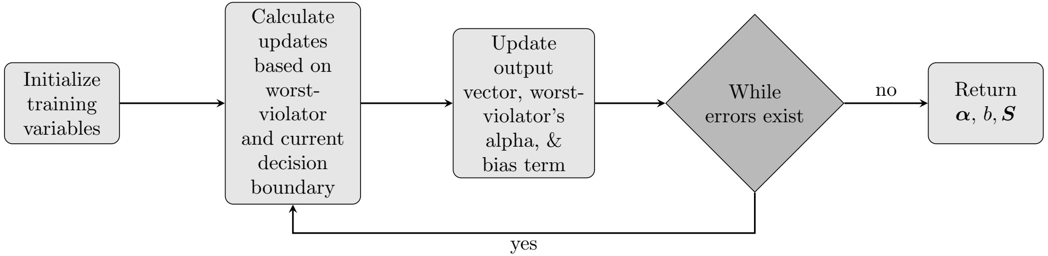
$$\forall i : \alpha_i \leftarrow \alpha_i + \eta \begin{cases} (C y_i - \alpha_i) & y_i o_i < 1 \\ (-\alpha_i) & \text{otherwise} \end{cases}$$

- Updates worst-violating sample **only**
- Worst-violator updated **once**

$$\Lambda \leftarrow C y_i$$
$$\alpha_i \leftarrow \alpha_i + \eta \Lambda$$
$$b \leftarrow b + \eta \Lambda$$

G. Melki et al. “OLLAWV: OnLine Learning Algorithm using Worst-Violators”. **Applied Soft Computing**, vol. 66, 384-393, 2018

OLLAWV: Algorithm



OnLine Learning Algorithm using Worst-Violators (OLLAWV)

Input: $\mathcal{D}$, C, γ, β, M

Output: $\boldsymbol{\alpha}$, b, $\boldsymbol{S}$

$\boldsymbol{\alpha} \leftarrow \mathbf{0}$, $b \leftarrow 0$, $\boldsymbol{S} \leftarrow \mathbf{0}$ ▷ Initialize OLLAWV model parameters

$\boldsymbol{o} \leftarrow \mathbf{0}$, $t \leftarrow 0$ ▷ Initialize the output vector and iteration counter

$wv \leftarrow 0$, $yo \leftarrow y_{wv} * \boldsymbol{o}_{wv}$ ▷ Initialize hinge loss error and worst-violator index

while $yo < M$ **do**

  $t \leftarrow t + 1$

  $\eta \leftarrow 2/\sqrt{t}$ ▷ Learning rate

  $\Lambda \leftarrow \eta * C * y_{wv}$ ▷ Calculate hinge loss update

  $B \leftarrow (\Lambda * \beta)/n$ ▷ Calculate bias update

  $\boldsymbol{o} \leftarrow \boldsymbol{o} + \Lambda * \mathcal{K}(\boldsymbol{x}_{\neg \boldsymbol{S}}, \boldsymbol{x}_{wv}, \gamma) + B$ ▷ Update output vector

  $\boldsymbol{\alpha}_{wv} \leftarrow \boldsymbol{\alpha}_{wv} + \Lambda$ ▷ Update worst-violator's alpha value

  $b \leftarrow b + B$ ▷ Update bias term

  $\boldsymbol{S}_t \leftarrow wv$ ▷ Save index of worst-violator

  $[yo,\, wv] \leftarrow \min\limits_{wv \in \{\neg \boldsymbol{S}\}} \{y_{wv} \cdot o_{wv}\}$ ▷ Find the worst-violator

end while

VCU

G. Melki et al. "OLLAWV: OnLine Learning Algorithm using Worst-Violators". **Applied Soft Computing**, vol. 66, 384-393, 2018

Experimental Environment

Algorithm Hyperparameters

Algorithm	Parameters
SVM	Penalty: $C \in \{4^n\}, n = \{-2, \dots, 5\}$ RBF Kernel: $\gamma \in \{4^n\}, n = \{-5, \dots, 2\}$
k-NN	Number of neighbors: $k \in \{1, 3, 5, 7\}$
J48	Pruning: {True, False} Pruning Confidence: {0.1, 0.25, 0.5}
JRip	Pruning: {True, False}
Naïve Bayes	Use kernel estimation: {True, False}
Logistic	Log-likelihood: $\{1e^{-7}, 1e^{-8}, 1e^{-9}\}$

- Nested Cross-Validation (5 inner and 5 outer folds)
- Metrics: Accuracy, Run Time, Support Vectors

Classification Datasets

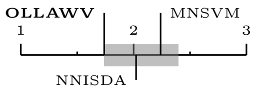
Dataset	# Samples	# Attributes	# Classes
small datasets			
iris	150	4	3
teach	151	5	3
wine	178	13	3
cancer	198	32	2
sonar	208	60	2
glass	214	9	6
vote	232	16	2
heart	270	13	2
dermatology	366	33	6
prokaryotic	997	20	3
eukaryotic	2,427	20	4
medium datasets			
optdigits	5,620	64	10
satimage	6,435	36	6
usps	9,298	256	10
pendigits	10,992	16	10
reuters	11,069	8,315	2
letter	20,000	16	26
large datasets			
adult	48,842	123	2
w3a	49,749	300	2
shuttle	58,000	7	7
web (w8a)	64,700	300	2
ijcnn1	141,691	22	2
intrusion	5,209,460	127	2

G. Melki et al. "OLLAWV: OnLine Learning Algorithm using Worst-Violators". **Applied Soft Computing**, vol. 66, 384-393, 2018

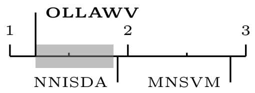
SVM Comparison Results

Comparison of OLLAWV vs. NNISDA and MNSVM

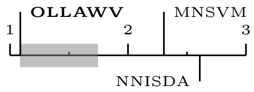
Dataset	Accuracy (%)			Run Time (s)			Support Vectors (%)		
	OLLAWV	NNISDA	MNSVM	OLLAWV	NNISDA	MNSVM	OLLAWV	NNISDA	MNSVM
small datasets									
iris	**97.33**	94.00	96.67	**0.05**	0.27	3.57	**13.50**	40.20	29.80
teach	52.32	52.31	**52.95**	**0.12**	0.44	8.85	**69.19**	99.80	87.40
wine	**98.87**	96.60	96.60	**0.28**	0.43	4.84	**15.02**	44.40	48.60
cancer	80.36	**81.86**	81.38	**0.49**	0.85	4.46	**42.79**	83.80	89.60
sonar	**92.32**	89.48	87.57	**0.59**	0.98	3.03	**31.26**	73.00	66.00
glass	**72.41**	67.81	69.30	**0.46**	1.01	11.94	**62.84**	90.80	87.60
vote	**96.54**	96.11	93.99	**0.26**	0.46	1.49	**13.36**	33.20	34.00
heart	82.22	**83.33**	**83.33**	**0.50**	0.91	6.45	**37.69**	73.00	82.00
dermatology	97.82	**98.36**	**98.36**	**1.62**	2.47	11.68	**36.94**	59.00	59.80
prokaryotic	88.96	88.86	**88.97**	**6.09**	10.64	50.86	**29.01**	51.20	49.00
eukaryotic	77.38	79.56	**81.21**	61.95	**49.16**	342.76	**54.11**	76.40	72.60
medium datasets									
optdigits	99.11	99.29	**99.31**	**411**	528	787	**28.64**	31.60	30.60
satimage	91.66	**92.39**	92.35	1,334	**687**	1,094	**20.72**	45.00	44.80
usps	97.49	98.05	**98.24**	10,214	**5,245**	7,777	**11.22**	29.40	28.00
pendigits	99.56	**99.62**	99.61	**723**	909	1,500	**10.27**	17.60	16.60
reuters	98.03	**98.08**	97.99	**954**	1,368	1,657	**8.770**	18.20	18.60
letter	96.99	99.11	**99.13**	**5,259**	12,009	26,551	**43.56**	57.60	56.60
large datasets									
adult	84.75	85.07	**85.13**	**21,025**	72,552	123,067	**34.66**	56.00	56.60
w3a	**98.86**	98.82	98.82	**6,532**	15,951	24,562	**3.270**	14.60	12.40
shuttle	99.77	99.83	**99.87**	**2,833**	7,420	45,062	**2.010**	6.00	16.40
web	98.94	**99.00**	99.00	**12,067**	30,583	38,040	**4.320**	13.20	10.80
ijcnn1	98.31	99.34	**99.41**	**162,587**	296,917	370,144	16.36	11.00	**7.600**
intrusion	**99.77**	99.67	99.66	**2,402,804**	4,646,810	3,772,113	**0.780**	2.000	1.700
Average	**91.29**	91.15	91.25	**114,209**	221,350	191,861	**25.66**	44.65	43.79
Ranks	**1.739**	2.022	2.239	**1.217**	1.913	2.869	**1.087**	2.609	2.304



Accuracy (%)



Run Time (s)

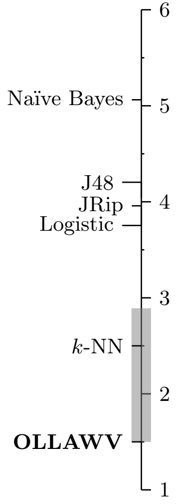


Support Vectors (%)

Non-SVM Comparison Results

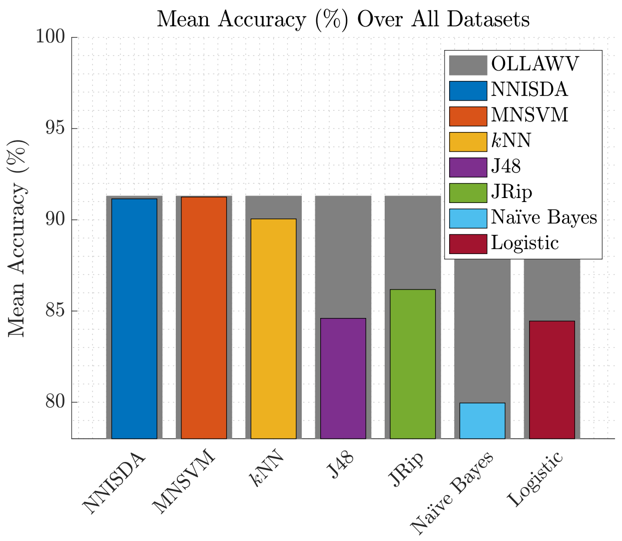
Accuracy (%) for Non-SVM Methods vs. OLLAWV

Dataset	OLLAWV	k-NN	J48	JRip	Naïve Bayes	Logistic
small datasets						
iris	**97.33 ± 1.49**	96.00 ± 3.65	94.00 ± 2.79	90.67 ± 4.35	96.00 ± 2.79	97.33 ± 2.79
teach	52.32 ± 3.46	**59.64 ± 2.89**	49.72 ± 7.58	56.75 ± 9.60	53.75 ± 6.46	51.77 ± 6.68
wine	**98.87 ± 1.54**	97.73 ± 3.72	90.43 ± 5.83	93.24 ± 3.27	96.60 ± 3.14	96.05 ± 2.58
cancer	**80.36 ± 5.80**	77.32 ± 6.93	73.81 ± 8.57	73.78 ± 5.81	67.73 ± 5.07	77.32 ± 7.78
sonar	**92.32 ± 3.11**	88.99 ± 4.59	76.16 ± 10.6	75.18 ± 6.77	73.69 ± 7.65	75.18 ± 7.31
glass	**72.41 ± 2.28**	67.73 ± 5.91	65.06 ± 5.51	65.59 ± 9.66	49.46 ± 5.19	62.04 ± 5.75
vote	**96.54 ± 1.87**	92.26 ± 3.19	95.70 ± 2.12	96.54 ± 2.45	92.24 ± 3.24	93.54 ± 2.59
heart	82.22 ± 2.93	79.63 ± 5.71	78.52 ± 2.81	80.74 ± 4.06	**84.44 ± 4.46**	83.33 ± 3.93
dermatology	**97.82 ± 0.05**	96.18 ± 1.78	94.52 ± 2.21	91.27 ± 5.08	97.28 ± 1.64	96.98 ± 2.28
prokaryotic	**88.96 ± 2.14**	87.96 ± 3.01	78.54 ± 1.62	79.13 ± 2.78	62.38 ± 3.54	87.57 ± 2.56
eukaryotic	77.38 ± 1.96	**81.42 ± 2.06**	65.27 ± 2.92	66.42 ± 3.47	39.27 ± 3.43	69.55 ± 1.34
medium datasets						
optdigits	**99.11 ± 0.38**	98.74 ± 0.39	90.87 ± 1.09	91.28 ± 0.40	92.42 ± 0.75	95.05 ± 0.91
satimage	**91.66 ± 0.80**	90.38 ± 0.72	85.64 ± 1.21	85.33 ± 0.77	85.41 ± 0.92	88.14 ± 1.11
usps	**97.49 ± 0.22**	97.04 ± 0.47	88.73 ± 0.46	89.20 ± 1.00	79.45 ± 0.59	91.88 ± 0.65
pendigits	**99.56 ± 0.12**	99.33 ± 0.17	96.24 ± 0.31	96.34 ± 0.41	88.34 ± 0.65	95.59 ± 0.18
reuters	**98.03 ± 0.22**	97.15 ± 0.43	96.90 ± 0.32	97.18 ± 0.44	93.52 ± 0.02	69.54 ± 0.28
letter	**96.99 ± 0.21**	95.71 ± 0.19	87.34 ± 0.68	87.02 ± 0.66	74.12 ± 0.97	77.45 ± 0.16
large datasets						
adult	**84.75 ± 0.26**	83.85 ± 0.28	84.38 ± 0.28	83.73 ± 0.17	80.57 ± 0.09	82.46 ± 0.14
w3a	**98.86 ± 0.04**	98.60 ± 0.06	98.71 ± 0.05	98.41 ± 0.10	96.71 ± 0.20	98.61 ± 0.12
shuttle	99.77 ± 0.03	99.93 ± 0.03	**99.97 ± 0.02**	99.96 ± 0.02	98.57 ± 0.24	96.83 ± 0.12
web	**98.94 ± 0.05**	98.89 ± 0.06	98.79 ± 0.09	98.50 ± 0.13	96.71 ± 0.21	98.70 ± 0.08
ijcnn1	98.31 ± 0.07	**98.48 ± 0.04**	98.40 ± 0.09	98.11 ± 0.10	90.69 ± 0.26	92.29 ± 0.16
intrusion	**99.77 ± 0.02**	88.20 ± 1.06	58.01 ± 26.6	87.66 ± 3.79	49.75 ± 30.7	65.15 ± 15.7
Average	**91.29 ± 1.26**	90.05 ± 2.06	84.60 ± 3.64	86.18 ± 2.84	79.96 ± 3.58	84.45 ± 2.83
Ranks	**1.500**	2.500	4.041	3.958	5.063	3.938

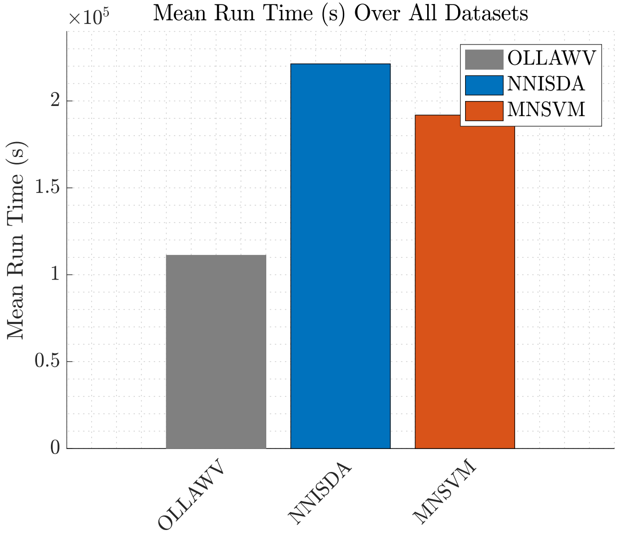


OLLAWV Conclusions

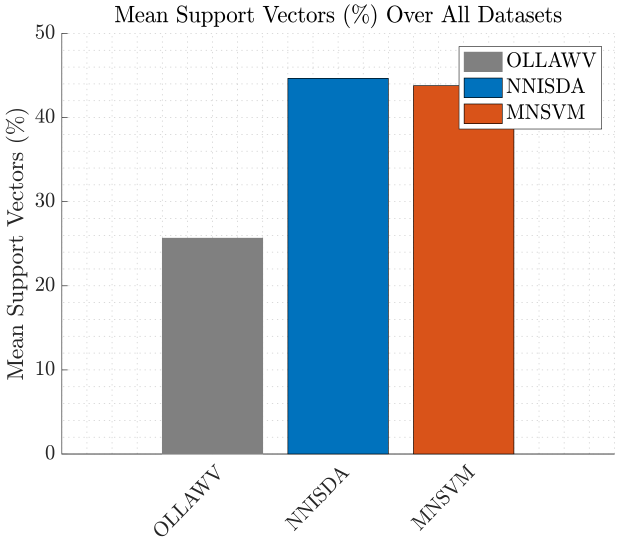
Competitive **accuracy** with SVM and non-SVM methods.



Significant decrease in **run time**.



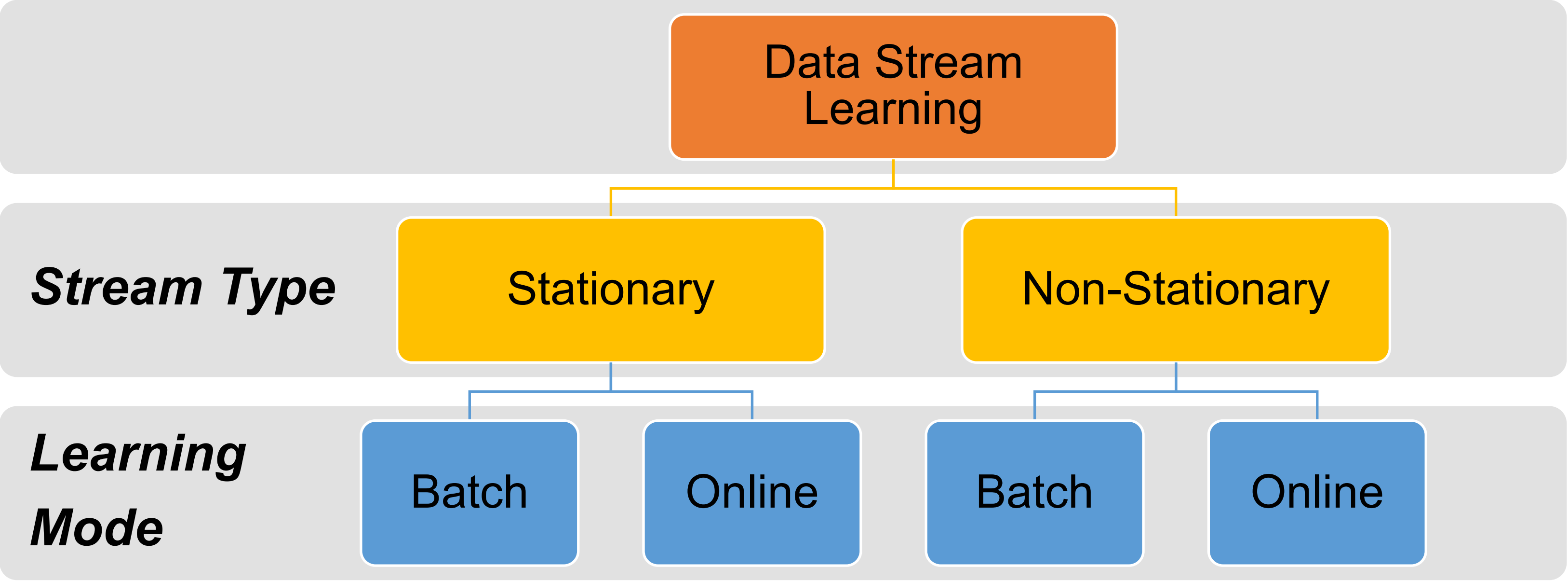
Significant decrease in **model complexity**.



G. Melki et al. "OLLAWV: OnLine Learning Algorithm using Worst-Violators". **Applied Soft Computing**, vol. 66, 384-393, 2018

OLLAWV for Batched Data Streams

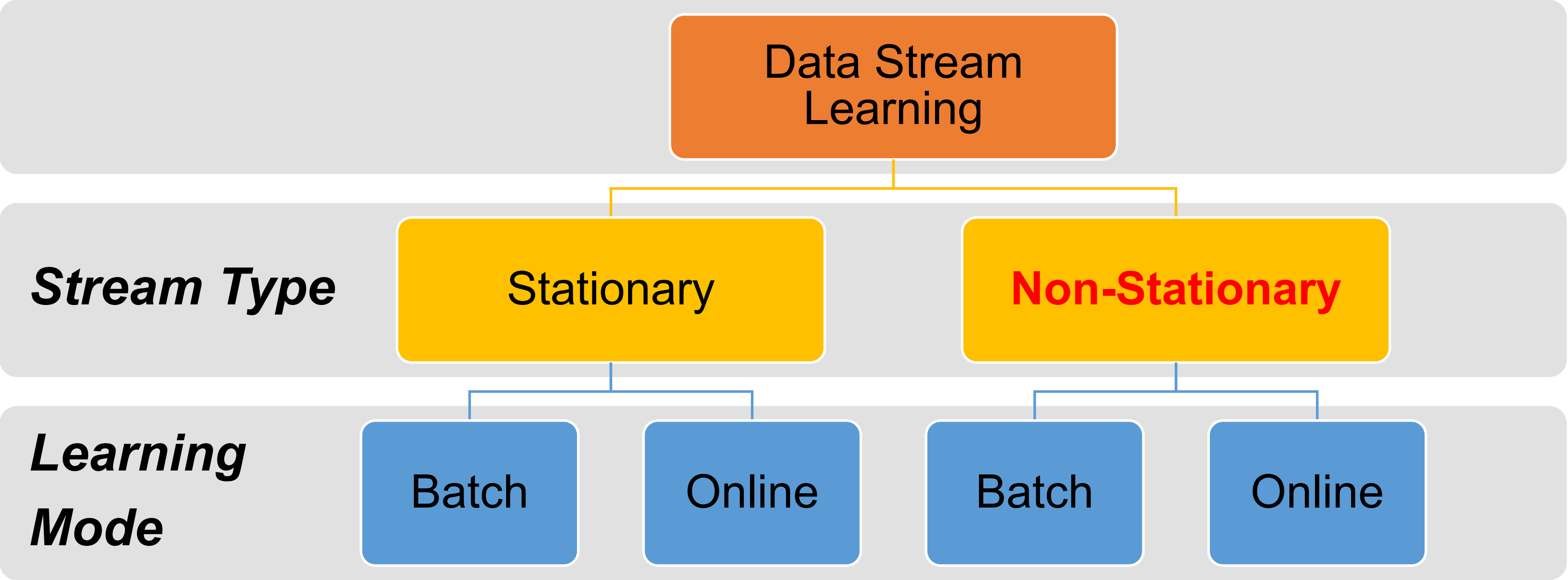
Data Stream Classification



Requirements:

- Incremental Processing
- Limited Time
- Limited Memory
- Changes in Data

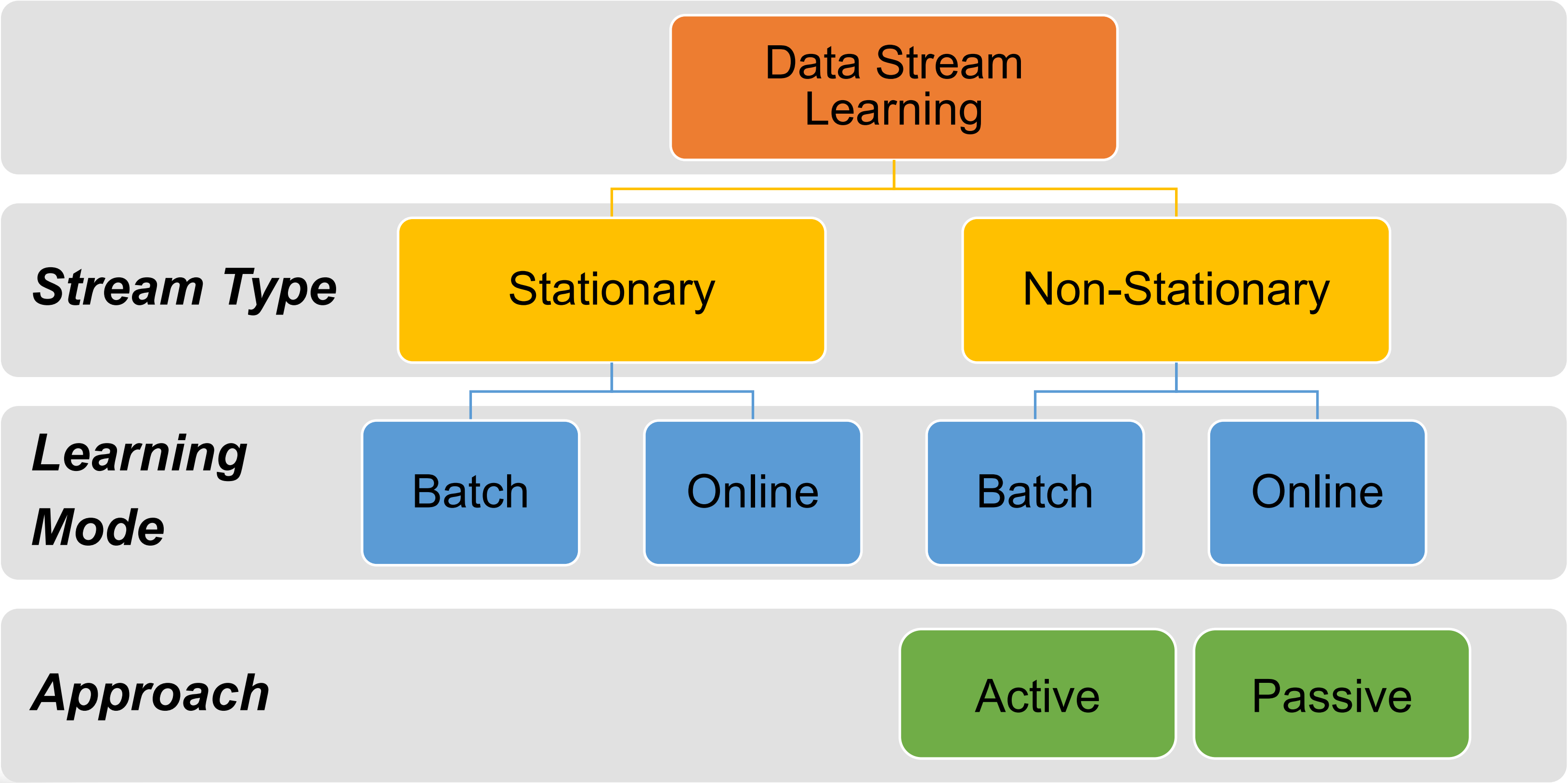
Data Stream Classification



Requirements:

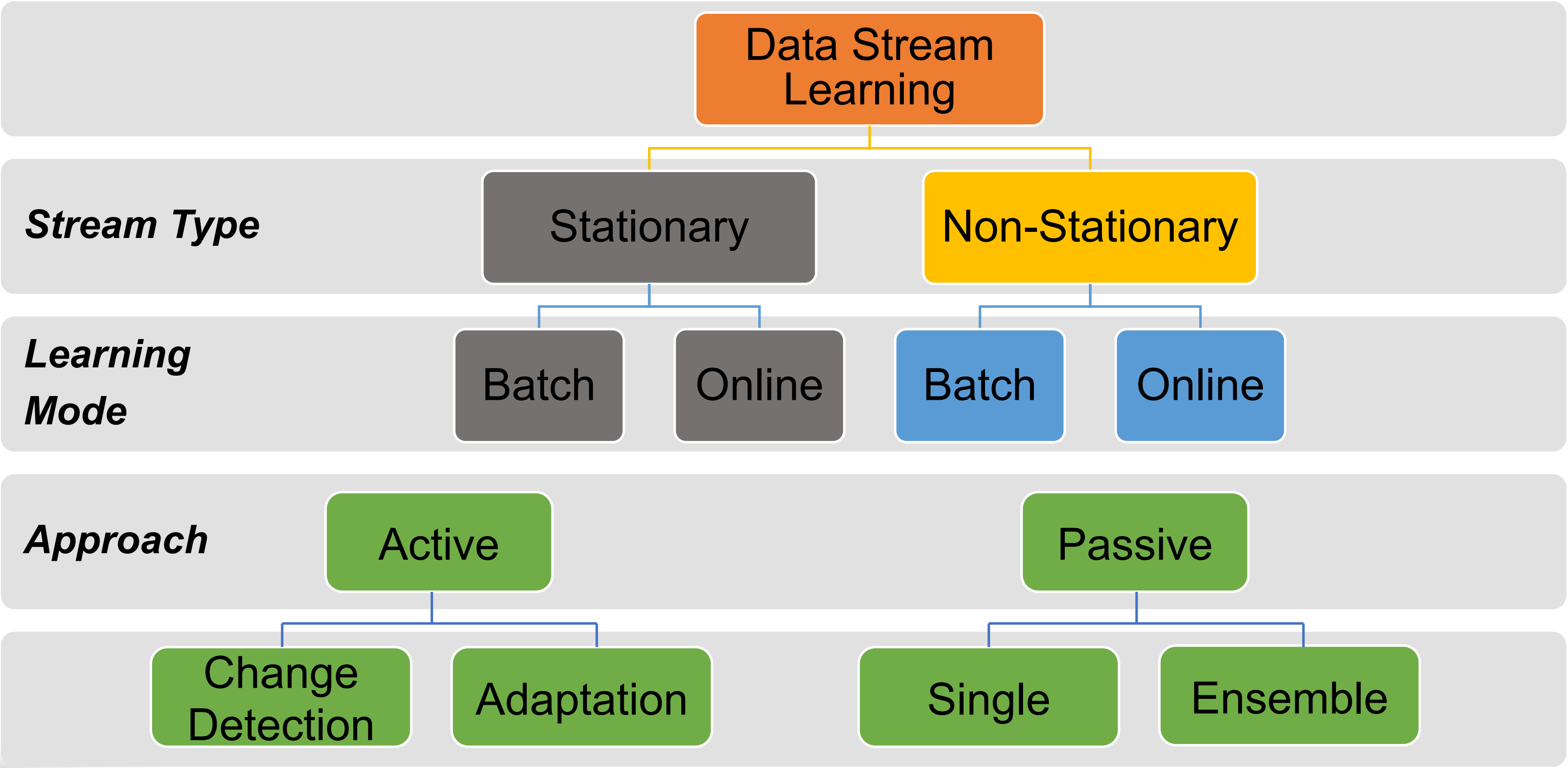
- Incremental Processing
- Limited Time
- Limited Memory
- **Changes in Data**

Data Stream Classification



VCU

Data Stream Classification



OnLine Learning Algorithm – List 2

OnLine Learning Algorithm - List 2 (OLLA-L2)

Input: $\boldsymbol{X}, \boldsymbol{Y}, \beta, n, e$

Output: $\boldsymbol{\alpha}$, b, $\boldsymbol{S}$

$\boldsymbol{\alpha} \leftarrow \mathbf{0}, b \leftarrow 0, \boldsymbol{S} \leftarrow \mathbf{0}$, $\boldsymbol{o} \leftarrow \mathbf{0}$, $i \leftarrow 0$ ▷ Initialize model and algorithm parameters

for $t = 1, \ldots, n * e$ **do**

  $\eta \leftarrow 2/\sqrt{t}$ ▷ Learning rate in function of time

  if $y_i\, o_i \leq 1$ **then** ▷ Check if current sample is a violator

    Calculate Λ *and* P ▷ Calculate update parameters

    $\boldsymbol{S} \leftarrow [\boldsymbol{S} \cup i]$ ▷ Save index of current violator

    $\alpha_i \leftarrow \alpha_i + (\Lambda - \mathrm{P})$ ▷ Update violator's alpha value

    $b \leftarrow b + (\Lambda - \mathrm{P})\beta$ ▷ Update bias term

  end if

  $i \leftarrow i + 1$ ▷ Get new sample

  if $i = n$ **then** ▷ If the sample index exceeds the number of samples

    $i = 0$ ▷ Reset sample index

  end if

  $o_i \leftarrow \boldsymbol{K}(\boldsymbol{x}_i, \boldsymbol{x}_{\boldsymbol{S}})\,\boldsymbol{\alpha}_{\boldsymbol{S}} + b$ ▷ Calculate the new sample's output value

end for

Comparison: OLLAWV vs. OLLA List 2

Comparison of OLLAWV vs. OLLA-L2

Dataset	Accuracy (%)		Run Time (s)	
	OLLA-L2	OLLAWV	OLLA-L2	OLLAWV
RBFNoDrift	93.07	**94.21**	**0.0238**	0.0329
HyperplaneSlow	87.40	**90.09**	**0.0261**	0.0353
HyperplaneFaster	87.40	**89.51**	**0.0256**	0.0263
STAGGERGeneratorF1	**100.0**	**100.0**	0.0034	**0.0021**
HyperplaneFaster02	87.41	**89.49**	**0.0257**	0.0268
MixedGeneratorBT	92.45	**98.00**	**0.0108**	0.0205
MixedGeneratorBF	92.55	**98.03**	**0.0107**	0.0299
SineGeneratorF1BF	97.37	**97.79**	**0.0091**	0.0122
SineGeneratorF2BF	97.37	**97.79**	**0.0091**	0.0121
STAGGERGeneratorF1BF	**100.0**	**100.0**	0.0035	**0.0021**
STAGGERGeneratorF2BF	**100.0**	**100.0**	0.0039	**0.0022**
HyperplaneFasterAN0	87.40	**89.51**	**0.0255**	0.0263
HyperplaneFasterAN5	87.29	**89.29**	**0.0258**	0.0264
SEASuddenAN0	84.01	**87.80**	0.0494	**0.0208**
SEASuddenAN05	83.69	**87.53**	0.0494	**0.0284**
Average	91.83	**93.94**	**0.0201**	0.0203
Rank	1.900	**1.100**	**1.3333**	1.6667

Experimental Environment

Base Streamed Datasets & Generators

Dataset	# Samples	# Attributes	# Classes
Static			
Shuttle	57,999	10	7
Census	299,284	42	2
CovType	581,012	55	7
Generators			
RandomRBFGenerator	1,000,000	10	2
LEDGenerator	1,000,000	2	10
HyperplaneGenerator	1,000,000	10	2
WaveformGenerator	1,000,000	40	3
STAGGERGenerator	1,000,000	3	2
MixedGenerator	1,000,000	4	2
SineGenerator	1,000,000	2	2
SEAGenerator	1,000,000	2	2

Algorithms

Algorithm	Description
HOT	Hoeffding Option Tree
AdaHOT	Adaptive Hoeffding Option Tree
NB	Naïve Bayes
k-NNPAW	k-NN with Probabilistic Adaptive Windows
DDM	Drift Detection Method with HOT
VFDR	Very Fast Decision Rules
VFDR-NB	VFDR with Naïve Bayes
SAE2	Social Adaptive Ensemble 2
Learn.NSE	Learn++ for Non-Stationary Environments
DWM	Dynamic Weighted Majority
DACC	Dynamic Adaptation to Concept Changes
OCBoost	Online Coordinate Boosting

- Permutations of the base generators & datasets:
  - **Drifting** mechanisms (slow, sudden)
  - Attribute **noise** introduced
- Interleaved **Test-Then-Train** evaluation

Results

Accuracy (%) for Data Stream Classifiers

Dataset	OLLAWV	HOT	AdaHOT	NB	k-NNPAW	DDM	VFDR	VFDR-NB	SAE2	LearnNSE	DWM	DACC	OCBoost
CovType	**90.28**	85.34	86.22	60.04	87.89	59.56	60.32	75.58	76.21	69.97	71.96	61.70	71.29
Census	93.76	94.70	**94.74**	87.02	93.65	91.85	93.65	84.06	90.13	84.14	91.40	90.37	93.47
Shuttle	**99.67**	98.18	98.52	90.02	99.26	98.49	88.40	96.06	90.24	93.79	89.91	92.03	74.21
RBFNoDrift	**94.21**	92.94	92.96	71.99	93.75	92.48	77.53	81.71	89.16	70.28	70.43	65.01	92.08
LEDNoDrift	73.83	73.85	73.84	**73.94**	65.83	73.64	41.16	73.75	67.60	67.84	71.15	48.27	17.44
HyperplaneSlow	**90.09**	82.10	82.42	77.69	84.03	81.57	68.88	85.19	82.67	86.20	88.06	80.66	85.78
HyperplaneFaster	**89.51**	82.72	85.34	77.23	84.27	84.33	78.63	85.18	83.01	86.50	86.76	81.15	87.80
RBFGradualRecurring	98.41	94.63	94.44	58.33	**98.43**	93.47	60.08	86.16	88.48	72.87	74.96	61.91	49.62
RBFBlips	**99.07**	95.67	95.60	60.83	98.94	94.92	66.90	88.35	89.04	77.53	79.98	68.27	47.46
WaveformGenerator	83.94	82.99	**84.14**	80.41	80.13	83.61	63.88	75.84	80.18	80.19	78.39	73.59	55.05
STAGGERGeneratorF1	**100.0**	99.99	99.99	**100.0**	**100.0**	**100.0**	99.91	**100.0**	95.04	89.82	**100.0**	99.96	**100.0**
HyperplaneFaster02	**89.49**	82.77	85.36	77.25	84.27	87.64	78.89	85.11	83.04	86.51	86.76	81.23	87.59
RBFGradualRecurringv2	**97.18**	93.29	93.00	57.47	95.73	93.19	57.96	80.71	84.45	62.41	63.79	49.42	48.88
MixedGeneratorBT	98.00	99.11	**99.32**	91.93	97.67	99.11	83.12	93.28	93.16	90.91	91.19	89.16	98.98
MixedGeneratorBF	98.03	99.18	**99.36**	92.04	97.59	99.20	89.96	94.30	93.41	90.76	91.46	88.61	98.94
RandomRBFGeneratorC4A25	**99.12**	97.43	97.14	81.59	98.69	96.89	72.33	89.03	90.61	78.23	79.67	63.02	52.16
RandomRBFGeneratorC4A50	**99.74**	99.16	99.14	91.90	99.17	98.99	80.63	95.63	92.00	86.03	90.45	73.78	50.66
SineGeneratorF1BF	97.79	**99.75**	99.73	93.55	95.54	99.66	94.83	95.90	94.51	92.55	93.30	92.20	99.51
SineGeneratoF2BF	97.79	**99.75**	99.73	93.55	95.41	99.66	95.26	96.10	94.57	92.55	93.34	92.20	99.48
STAGGERGeneratorF1BF	**100.0**	99.99	99.99	**100.0**	**100.0**	**100.0**	99.91	**100.0**	95.04	89.82	**100.0**	99.96	**100.0**
STAGGERGeneratorF2BF	**100.0**	99.98	99.98	**100.0**	**100.0**	99.98	99.87	**100.0**	95.02	44.41	**100.0**	**100.0**	0.61
HyperplaneFasterAN5	**89.29**	82.69	85.24	87.38	84.19	88.74	79.14	84.89	82.92	86.41	86.67	81.12	87.38
SEASuddenAN0	87.80	84.92	85.18	88.23	87.22	**88.97**	81.56	85.17	85.11	85.77	86.93	83.73	88.23
SEASuddenAN05	87.53	84.57	84.82	87.53	86.96	**88.28**	81.55	85.11	84.57	85.63	86.54	83.53	87.53
Average	**93.94**	91.90	92.34	82.50	92.03	91.43	78.93	88.21	87.51	81.30	85.55	79.20	73.92
Rank	**2.52**	5.42	4.54	8.19	4.8542	4.63	10.96	6.77	8.46	9.04	7.27	10.90	7.46

Results

Training Time (seconds) for Data Stream Classifiers

Dataset	OLLAWV	HOT	AdaHOT	NB	*k*-NNPAW	DDM	VFDR	VFDR-NB	SAE2	LearnNSE	DWM	DACC	OCBoost
CovType	0.0451	0.0765	0.0909	**0.0009**	0.0387	0.0577	0.0371	0.0445	0.1121	6.2806	0.0419	0.0357	0.0984
Census	0.0168	0.1040	0.1162	**0.0007**	0.0386	0.0345	0.0735	0.0458	0.0363	0.9368	0.0156	0.0175	0.0543
Shuttle	0.0103	0.0276	0.0298	**0.0004**	0.0386	0.0069	0.0108	0.0069	0.0217	0.2692	0.0104	0.0186	0.0581
RBFNoDrift	0.0329	0.0174	0.0247	**0.0002**	0.0383	0.0828	0.4267	0.3172	0.0491	3.4029	0.0104	0.0137	0.0402
LEDNoDrift	0.0697	0.0373	0.0649	**0.0003**	0.0389	0.0245	0.0097	0.0082	0.0694	5.4734	0.0175	0.0290	0.0547
HyperplaneSlow	0.0353	0.0094	0.0142	**0.0002**	0.0389	0.1405	0.1871	0.3006	0.0473	3.3734	0.0091	0.0142	0.0389
HyperplaneFaster	0.0263	0.0295	0.0416	**0.0002**	0.0384	0.1231	0.3016	0.3784	0.0424	3.2991	0.0098	0.0143	0.0376
RBFGradualRecurring	0.0456	0.0316	0.0369	**0.0003**	0.0389	0.0391	1.1284	0.9295	0.1004	11.7980	0.0335	0.0474	0.1070
RBFBlips	0.0396	0.0243	0.0299	**0.0003**	0.0384	0.0278	0.7587	0.8342	0.0948	11.9934	0.0342	0.0473	0.1050
WaveformGenerator	0.0394	0.0751	0.0965	**0.0006**	0.0390	0.1847	0.9670	0.9271	0.1715	17.9675	0.0511	0.0743	0.1318
STAGGERGeneratorF1	0.0021	0.0006	0.0007	**0.0001**	0.0389	0.0005	0.0013	0.0007	0.0018	0.5517	0.0003	0.0020	0.0113
HyperplaneFaster02	0.0268	0.0302	0.0465	**0.0002**	0.0388	0.0260	0.3192	0.3010	0.0456	3.2893	0.0095	0.0141	0.0372
RBFGradualRecurringv2	0.0545	0.0169	0.0203	**0.0003**	0.0383	0.0859	0.7622	0.7634	0.0552	12.0529	0.0367	0.0464	0.1053
MixedGeneratorBT	0.0205	0.0027	0.0026	**0.0001**	0.0336	0.0071	1.7640	1.5534	0.0086	0.9761	0.0024	0.0052	0.0204
MixedGeneratorBF	0.0299	0.0024	0.0024	**0.0001**	0.0334	0.0065	1.1035	1.0924	0.0093	0.8971	0.0024	0.0058	0.0209
RandomRBFGeneratorC4A25	0.0200	0.0441	0.0407	**0.0003**	0.0358	0.0730	1.7831	1.6927	0.1245	13.9876	0.0395	0.0719	0.1580
RandomRBFGeneratorC4A50	0.0157	0.0333	0.0337	**0.0007**	0.0350	0.0909	2.8802	2.5559	0.1917	27.8456	0.0752	0.1432	0.2974
SineGeneratorF1BF	0.0122	0.0056	0.0056	**0.0001**	0.0334	0.0073	1.9301	2.9169	0.0127	1.3768	0.0034	0.0071	0.0235
SineGeneratoF2BF	0.0121	0.0051	0.0053	**0.0001**	0.0314	0.0077	5.0598	5.9477	0.0117	1.3996	0.0027	0.0074	0.0236
STAGGERGeneratorF1BF	0.0021	0.0006	0.0006	**0.0001**	0.0323	0.0005	0.0006	0.0005	0.0015	0.6175	0.0003	0.0022	0.0143
STAGGERGeneratorF2BF	0.0022	0.0010	0.0009	**0.0001**	0.0340	0.0006	0.0005	0.0005	0.0017	0.6752	0.0003	0.0020	0.0139
HyperplaneFasterAN5	0.0264	0.0200	0.0200	0.0440	0.0269	0.0115	0.2298	0.2417	0.0312	2.5014	**0.0080**	0.0147	0.0740
SEASuddenAN0	0.0208	0.0051	0.0052	0.0268	0.0312	0.0099	0.0370	0.0373	0.0176	1.3545	**0.0032**	0.0059	0.0335
SEASuddenAN05	0.0284	0.0055	0.0055	0.0257	0.0309	0.0095	0.0321	0.0334	0.0167	1.3365	**0.0032**	0.0061	0.0326
Average	0.0264	0.0252	0.0307	**0.0043**	0.0359	0.0441	0.8252	0.8721	0.0531	5.5690	0.0175	0.0269	0.0663
Rank	6.5833	4.7917	5.7917	**1.8750**	7.7500	6.0833	9.8333	9.5833	8.2917	12.6667	2.7917	5.5417	9.4167

VCU

Data Stream Classification Conclusions

OLLA-L2	OLLAWV

✅ OnLine (i.e. stochastic) learning algorithms
✅ Accurate & Produce sparse models

OLLA-L2	OLLAWV
✅ simple, efficient update	❌ kernel calculation bottleneck
❌ no stopping criterion	✅ implicit stopping criterion
❌ less accurate than OLLAWV	✅ worst-violator update

Experimental study showed OLLAWV's merit as a base classifier:

- Batch data stream setting
- Slower drifting streams

Conclusions & Future Work

Conclusions

- **Multiple-Target Regression -** 3 unique approaches:
  - **SVR:** baseline problem transformation

    ✅ *fast, accurate* ❌ *targets' correlations*

  - **SVRRC:** ensemble of random chains using SVR

    ✅ *targets' correlations somewhat exploited* ❌ *slow, random*

  - **SVRCC:** maximally correlated chained model

    ✅ *correlations exploited, faster run times* ❌ *uncorrelated targets*

- **SVRCC** provided the best results among the contributions and the methods compared.

G. Melki et al. "Multi-target support vector regression via correlation regressor chains". **Information Sciences**, vol. 415, pp. 53–69, 2017.

Conclusions

- **Multiple-Instance Classification**
  - **MIRSVM:** novel MI bag-level formulation and algorithm, with a bag-representative selector

✅ *remedied class imbalance, fast convergence*

❌ *issues when limited number of bags, QP solver*

  - **Bag-level** based and **ensemble learners** had the best performance
  - **Instance-level** based learners deemed as strongly biased and unstable classifiers

G. Melki et al. "MIRSVM: Multi-Instance Support Vector Machine with Bag Representatives". **Pattern Recognition**, vol. 79, 228-241, 2018.

Conclusions

- **Traditional Classification**
  - **OLLAWV:** novel online (stochastic) algorithm for the L1SVM problem + stopping criterion

✅ *~2x **faster** than contemporary SVM solvers*

✅ *~1.7x **smaller** (less complex) final model*

✅ ***faster** & **less complex** model without **sacrificing accuracy***

  - OLLAWV performed the best against 5 **non-SVM** algorithms

G. Melki et al. "OLLAWV: OnLine Learning Algorithm using Worst-Violators". **Applied Soft Computing**, vol. 66, 384-393, 2018

Conclusions

- **Data Stream Classification**
  - **OLLA-L2:** online algorithm with change detector

✅ *fast update mechanism*
❌ *no stopping criterion*
❌ *less accurate than OLLAWV*

  - **OLLAWV:** stochastic base algorithm (batched data streams)

✅ *accuracy*
✅ *stopping criterion*
⚠️ *runtime*
❌ *does not accommodate abrupt drifting streams*

Future Work

- **OLLAWV for the Multi-target Learning & Multi-Instance Learning paradigms:**
  - Implement the existing contributions using OLLAWV
  - Extend OLLAWV to the regression case
  - Investigate OLLAWV's performance under the *Multi-Instance regression* and *Multi-Label classification* settings

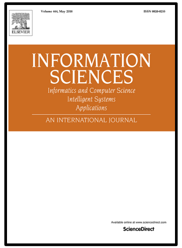
Future Work

- **OLLAWV for the Multi-target Learning & Multi-Instance Learning paradigms:**
  - Implement the existing contributions using OLLAWV
  - Extend OLLAWV to the regression case
  - Investigate OLLAWV's performance under the *Multi-Instance regression* and *Multi-Label classification* settings
- **OLLAWV & OLLA-L2 for the online Data Stream Learning paradigm:**
  - Investigate stopping criteria for OLLA-L2
  - Explore merging favorable properties of OLLA-L2 and OLLAWV
  - Add functionality for handling non-stationary streams

Future Work

- **OLLAWV for the Multi-target Learning & Multi-Instance Learning paradigms:**
  - Implement the existing contributions using OLLAWV
  - Extend OLLAWV to the regression case
  - Investigate OLLAWV's performance under the *Multi-Instance regression* and *Multi-Label classification* settings
- **OLLAWV & OLLA-L2 for the online Data Stream Learning paradigm:**
  - Investigate stopping criteria for OLLA-L2
  - Explore merging favorable properties of OLLA-L2 and OLLAWV
  - Add functionality for handling non-stationary streams
- **Parallelized/Distributed version of OLLAWV**

Publications



[1] **G. Melki**, A. Cano, V. Kecman, and S. Ventura. “*Multi-target support vector regression via correlation regressor chains*”. Information Sciences, vol. 415-416, pp. 53–69, 2017.

Impact Factor: **4.305**, Quartile: **Q1**

[2] **G. Melki**, A. Cano, and S. Ventura. “*MIRSVM: Multi-Instance Support Vector Machine with Bag Representatives*”. Pattern Recognition, vol. 79, pp. 228-241, 2018.

Impact Factor: **3.962**, Quartile: **Q1**

[3] **G. Melki**, A. Cano, V. Kecman, and S. Ventura. “*OLLAWV: OnLine Learning Algorithm using Worst-Violators*”. Applied Soft Computing, vol. 66, pp. 384-393, 2018.

Impact Factor: **3.907**, Quartile: **Q1**

Vita

- **G. Melki**, A. Cano, V. Kecman, and S. Ventura. "*OLLAWV: OnLine Learning Algorithm using Worst-Violators*". Applied Soft Computing, vol. 66, pp. 384-393, 2018.
- **G. Melki**, A. Cano, and S. Ventura. "*MIRSVM: Multi-Instance Support Vector Machine with Bag Representatives*". Pattern Recognition, vol. 79, pp. 228-241, 2018.
- **G. Melki**, A. Cano, V. Kecman, and S. Ventura. "*Multi-target support vector regression via correlation regressor chains*". Information Sciences, vol. 415-416, pp. 53–69, 2017.
- **G. Melki**, *Fast Online Training of L1 Support Vector Machines*, Master's Thesis, Virginia Commonwealth University, 2016.
- **G. Melki**, V. Kecman, *Speeding Up Online Training of L1 Support Vector Machines*, In: Proceedings of the IEEE SoutheastCon, 2016, pp. 1-6, 2016.
- V. Kecman, **G. Melki**, *Fast Online Algorithm for SVMs*, In: Proceedings of the IEEE SoutheastCon, 2016, pp. 1-6, 2016.
- V. Kecman, L. Zigiç, **G. Melki**, *Models and Algorithms for Support Vector Machines: Direct L2 SVM*, Seminar at Max Planck Institute for Intelligent Systems, Empirical Inference, Tübigen, Germany, 2015.