Support Tensor Machine for Financial Forecasting

Giuseppe G. Calvi, Vladimir Lucic, Danilo P. Mandic

Imperial College London

E-mails: {giuseppe.calvi15, d.mandic}@imperial.ac.uk
vladimir.lucic@macquarie.com

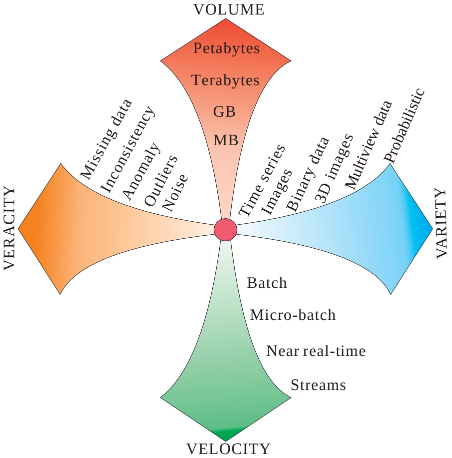
May 17, 2019

Outline

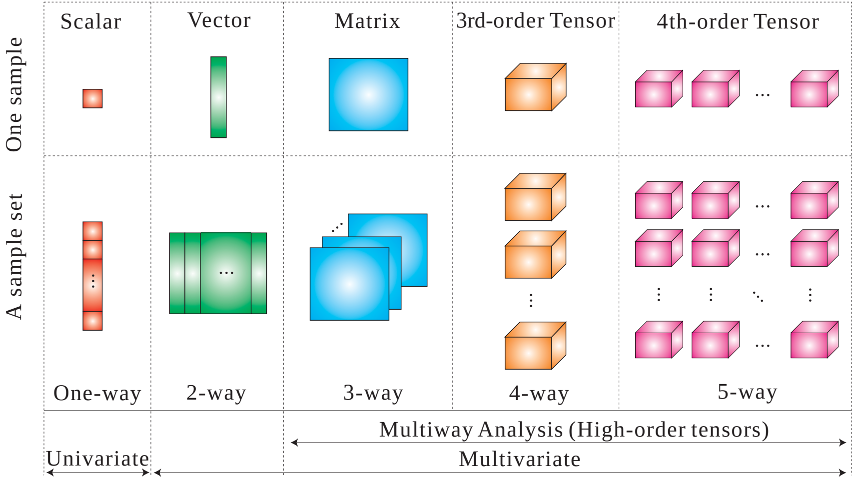
1. Modern-Day Data
   - 4V's of Big Data
   - Types of data
2. Support Vector Machine (SVM)
   - Recap
   - SVM in Finance
3. From Support Vector Machine to Support Tensor Machine (STM)
   - Motivation
   - Least-Squares STM
4. Predicting the Direction of Price Movement of the S&P 500
   - Problem Setup
   - Simulation Results
5. Conclusions

The 4V's of Big Data

- Multidimensional, multi-modal complex datasets
- These are characterized not only by size
- High Volume $\Longrightarrow$ need for scalable algorithms
- High Velocity $\Longrightarrow$ processing data in real-time
- High Veracity $\Longrightarrow$ dealing with noisy or incomplete data
- High Variety $\Longrightarrow$ integration across different kinds of data



Types of Data: From Scalars to Tensors



Support Vector Machines – One Slide Recap

- Support Vector Machines are inherently binary classifiers that operate on vector inputs
- That is, given a dataset of M elements $\{\mathbf{x}_m, t_m\}$, $m = 1, \ldots, M$, where each $\mathbf{x}_m \in \mathbb{R}^N$ with label $t_m \in \{+1, -1\}$, the SVM traces a hyperplane which best separates the data into two classes, one per label. Unseen datapoints are classified according to which side of the hyperplane they are mapped to
- The SVM achieves this by finding weight and bias parameters, $\mathbf{w}$, b which maximise the margin from the hyperplane to the data
- The SVM optimization problem is given by

SVM

$$\arg\min_{\mathbf{w},b} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{m=1}^{M}\xi_m$$

$$\text{s.t. } t_m(\mathbf{w}^T\phi(\mathbf{x}_m) + b) \geq 1 - \xi_m$$

$$\xi_m \geq 0, \ m = 1, \ldots, M$$

Least Squares SVM (LS-SVM)

$$\arg\min_{\mathbf{w},b} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{m=1}^{M}e_m^2$$

$$\text{s.t. } t_m(\mathbf{w}^T\phi(\mathbf{x}_m) + b) = 1 - e_m$$

$$m = 1, \ldots, M$$

SVM in Finance

- Ease of their interpretability and richness of underlying physical meaning
- Most methods operate based on the concept of **Empirical Risk Minimization** (ERM), which means that the parameters of the model are selected in order to fit the **existing sample**
- SVM instead follows the **Structural Risk Minimization** (SRM) principle, which balances the model's complexity against its success at fitting the training data
- Hence SVM is less prone to overfitting than other ERM-based techniques
- SVM is **robust** and **computationally efficient**

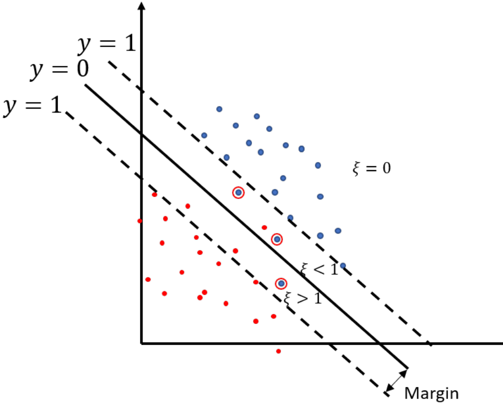


Figure: 2-D representation of SVM

Motivation for Tensor Extension

- Tensors have shown that the exploitation of structural information can lead to better expressive powers of algorithms
- SVM operates on vectors: what if its inputs are tensors?
  - Higher structural information in data $\implies$ inherently more features
  - Hence, given the right tensorization, performance may be superior to that of SVM
- SVM has been extended to its tensor version, namely the Support Tensor Machine (STM)
- The STM operates directly on tensor-valued inputs,
- STM has already shown to bring improvements in applications such as real world image and video processing
- Here, we apply it for the first time in a financial context

Support Tensor Machine (STM)

- Consider a dataset of M elements, each of which is an N-th order tensor $\underline{\mathbf{X}}_m \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$
- The STM operates on the pairs $\{\underline{\mathbf{X}}_m, t_m\}$ and assigns the binary label/target $\{+1, -1\}$ by solving the following for each mode-n

STM

$$\underset{\mathbf{w}_n, b}{\arg\min}\ \frac{1}{2}||\mathbf{w}_n||^2 \prod_{1\leq i\leq N}^{i\neq n} (||\mathbf{w}_i||^2) + C\sum_{m=1}^{M} \xi_m$$

$$\text{s.t.} \quad t_m\left(\mathbf{w}_n^T(\underline{\mathbf{X}}_m \bar{\times}_{i\neq n}\mathbf{w}_i) + b\right) \geq 1 - \xi_m$$

$$\xi_m > 0, \quad m = 1, 2, \ldots, M$$

Least Squares STM (LS-STM)

$$\underset{\mathbf{w}_n, b}{\arg\min}\ \frac{1}{2}||\mathbf{w}_n||^2 \prod_{1\leq i\leq N}^{i\neq n} (||\mathbf{w}_i||^2) + \frac{\gamma}{2}\epsilon^T\epsilon$$

$$\text{s.t.} \quad t_m\left(\mathbf{w}_n^T(\underline{\mathbf{X}}_m \bar{\times}_{i\neq n}\mathbf{w}_i) + b\right) = 1 - \epsilon_m$$

$$m = 1, \ldots, M$$

- A label is assigned to a new datapoint $\underline{\mathbf{X}}_*$, by

$$t_* = \text{sign}\left(\underline{\mathbf{X}}_* \times_1 \mathbf{w}_1 \times_2 \cdots \times_N \mathbf{w}_N + b\right)$$

LS-STM Algorithm

Algorithm 1. Least-Squares Support Tensor Machine (LS-STM)

Input: Dataset $\{\underline{\mathbf{X}}_m, t_m\}, m = 1, \ldots, M$, with $\underline{\mathbf{X}}_m \in \mathbb{R}^{I_1 \times \cdots \times I_N}$
Output: Set of weights $\{\mathbf{w}_n\}, n = 1, \ldots, N$ and bias b.

1: Initialize randomly $\{\mathbf{w}_n\}, n = 1, \ldots, N$
2: **while** not converged or *max iterations* reached **do**
3: **for** $n = 1$ to N **do**
4: $\eta \leftarrow \prod_{i \neq n} ||\mathbf{w}_i||^2$
5: $\mathbf{x}_m = \underline{\mathbf{X}}_m \bar{\times}_{i \neq n} \mathbf{w}_i$
6: Find $\mathbf{w}_n$ by optimizing:

$$\min_{\mathbf{w}_n, b, \epsilon} \frac{\eta}{2} ||\mathbf{w}_n||^2 + \frac{C}{2} \epsilon^T \epsilon$$

$$\text{s.t.} \quad t_m(\mathbf{w}_n^T \mathbf{x}_m + b) = 1 - \epsilon_m, \quad m = 1, \ldots, M$$

7: **end for**
8: **end while**
9: **Return** $\{\mathbf{w}_n\}, n = 1, \ldots, N$ and b.

A Note on Kernel Usage and Probabilistic Interpretation

- Classical SVM often makes use of the so-called **kernel trick** $\Longrightarrow$ the weights are not computed explicitly, but implied by the kernel
- We employed an RBF (Gaussian) kernel
- However, LS-STM does require the actual weights $\mathbf{w}_n$, $n = 1, \dots, N$
- These were estimated in a least squares fashion by

$$\mathbf{w}_n = (\boldsymbol{\Omega}_n^T \boldsymbol{\Omega}_n)^{-1} \boldsymbol{\Omega}_n^T (\mathbf{y} - b)$$

  where $\boldsymbol{\Omega}_n = \left[\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_M\right]^T \in \mathbb{R}^{M \times I_n}$ and
  $y(\mathbf{x}^*) \approx y_i = \sum_{\substack{m=1 \\ m \neq i}}^{M} \alpha_m t_m k(\mathbf{x}_m, \mathbf{x}_*) + b$, $i = 1, \dots, M$ (details in the paper)

- The results were interpreted probabilistically by Platt scaling, which, for a test point $\mathbf{x}^*$, computes the probability

$$P(t_* = 1 | x_*) = \frac{1}{1 + \exp(Ay_* + B)}$$

  where A, B are learnt parameters

Application and Problem Setup

- The potential of STM still has to be explored within the study (i.e. prediction) of time series
- Being financial predictions particularly challenging, the LS-STM is employed to predict the daily direction of movement of price for the S&P 500 financial index, in the period ranging from January 2006 to January 2017
- Given daily closing prices, S_t, and daily trading volumes, V_t, the percentage returns, r_t, annualized return, R_A, annualized volatility, σ_A, annualized Sharpe ratio, SR_A, are defined as

$$r_t = \frac{S_t - S_{t-1}}{S_{t-1}}, \qquad v_t = \frac{V_t - V_{t-1}}{V_{t-1}}$$

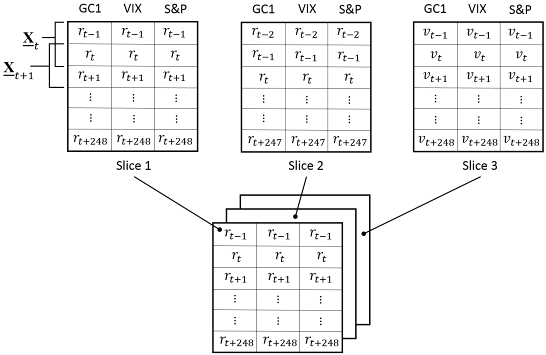
$$R_A = 250 \frac{1}{L} \sum_{t=1}^{L} r_t, \quad \sigma_A = \sqrt{250} \sqrt{\frac{1}{L-1} \sum_{t=1}^{L} (r_t - \bar{r})^2}$$

$$\mathrm{SR}_A = \frac{R_A}{\sigma_A}$$

where $\bar{r} = \frac{1}{L} \sum_{t=1}^{L} r_t$, and L is the number of samples.

Simulation Results

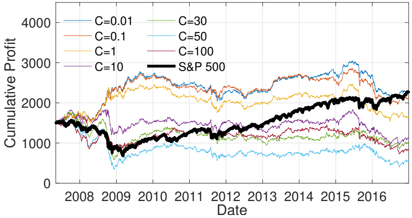
- As predictors the VIX, the S&P 500, the GC1 and the S&P 500 itself are used
- The VIX indicates the implied volatility of the S&P 500, while the GC1 is a gold commodity
- Sliding windows of $L = 250$ samples: 249 for training, one for testing



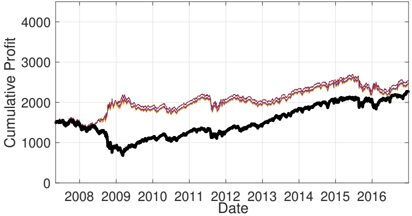
SVM vs LS-STM Performance

Table: SVM vs. LS-STM performance for varying values of C. Accuracies $>$ 50% within this financial context are valid results.

	$C = 0.01$			$C = 0.1$			$C = 1$			$C = 10$			$C = 30$			$C = 50$			$C = 100$		
	%	SR_A	σ_A	%	SR_A	σ_A	%	SR_A	σ_A	%	SR_A	σ_A	%	SR_A	σ_A	%	SR_A	σ_A	%	SR_A	σ_A
SVM	51.90	0.44	**0.13**	51.44	0.37	**0.13**	51.25	0.16	**0.14**	51.11	-0.06	0.19	50.88	-0.05	0.25	50.65	-0.14	0.37	50.42	-0.17	0.25
LS-STM	**52.32**	**0.46**	0.14	**52.37**	**0.46**	0.14	**52.32**	**0.45**	0.15	**52.41**	**0.46**	**0.14**	**52.41**	**0.49**	**0.14**	**52.41**	**0.49**	**0.14**	**52.41**	**0.49**	**0.14**



(a) Profit based on SVM



(b) Profit based on LS-STM

Figure: Cumulative profits generated using SVM and LS-STM based strategies, for varying values of C.

Conclusions

- We have applied the Support Tensor Machine (STM), in particular its least-squares formulation (LS-STM), to the problem of financial classification
- This has been achieved by adequately tensorizing the input data from the VIX, the GC1 gold commodity, and the S&P 500 itself
- We devised a method to allow for kernel usage, based on a least squares approximation of the weights, and interpreted the results probabilistically via Platt scaling
- LS-STM considerably performed better than SVM in all metrics, and exhibited better stability than its vector counterpart, owing to the superior structural information in tensors

New Software: Higher Order Tensors ToolBOX (HOTTBOX)

Our Python toolbox for multilinear algebra: `github.com/hottbox/hottbox`

Documentation: `hottbox.github.io`

Tutorials: `github.com/hottbox/hottbox-tutorials`

Selected references

- C. Cortes and V. Vapnik, "Support-vector networks", *Machine Learning*, vol. 20, no. 3 pp. 273-297, 1995.
- F. E. Tay and L. Cao, "Application of support vector machines in financial time series forecasting", *Omega*, vol. 29, no. 4, pp. 309–317, 2001.
- D. Tao, X. Li, X. Wu, and S. J. Maybank, "Supervised tensor learning", *Knowledge and Information Systems*, vol. 13, pp. 1–42, 2007
- A. Cichocki, A.H. Phan, Q. Zhao, N. Lee, I. Oseledets, M. Sugiyama, and D. P. Mandic, "Tensor networks for dimensionality reduction and large-scale optimization. Part 2: Applications and future perspectives," *Foundations and Trends in Machine Learning*, vol. 9, no. 6, pp. 431-673, 2017.
- J. Platt *et al.*, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods", *Advances in Large Margin Classifiers*, vol. 10, no. 3, pp. 61–74, 1999.