# Volatility Forecasting using SVM

Project for CS229 Machine Learning

Jeremy Andre Alfred Wechselberger Shanbin Zhao
FinMath EE MSE

## Introduction

Financial time series forecasting is one of the most challenging applications of modern time series analysis. Financial time series are inherently noisy, nonstationary and deterministically chaotic. These characteristics suggest that there is no complete information that could be obtained from the past behavior of financial markets to fully capture the dependency between the future price and that of the past.

There are two main categories in financial time series forecasting: univariate analysis and multivariate analysis. In multivariate analysis, any indicator, whether it is related to the output directly or not, can be incorporated as the input variable, while in univariate analysis, the input variables are restricted to the time series being forecasted. A general univariate model that is commonly used is based on the AutoRegressive Integrated Moving Average (ARIMA) method. Compared to other multivariate models, the performance of ARIMA is not satisfactory because this model is parametric, and additionally, it is developed on the assumption that the time series being forecasted are linear and stationary. These constraints are not consistent with the characteristics of financial time series. Therefore, Artificial Neural Network (ANN) assisted multivariate analysis has become a dominant and popular tool in recent years. The prediction performance is greatly improved by the use of a neural network. A neural network is more effective in describing the dynamics of non-stationary time series due to its unique non-parametric, nonassumable, noise-tolerant and adaptive properties.

However, a critical issue concerning neural networks is the over-fitting problem. It can be attributed to the fact that a neural network captures not only useful information contained in the given data, but also unwanted noise. This usually leads to a large generalization error. Unlike most of the traditional learning machines that adopt the Empirical Risk Minimization Principle, SVMs implement the Structural Risk Minimization Principle, which seeks to minimize an upper bound of the generalization error rather than minimize the training error. This will result in better generalization than conventional techniques.

## Different Market Volatilities

Here is a brief description of the differents types of volatilities:

**Realized volatility** This is a statistical measure of the *noise* in the markets on $n$ days. If we denote the price times series of a stock as $S_0, S_1, S_2, \ldots, S_n$, realized volatility is given by:

$$\sigma_r = \sqrt{\frac{\sum_{i=1}^{n} \left(\ln \frac{S_i}{S_{i-1}}\right) 2}{n} \times 252}$$

**Implied volatility** This is the key parameter used to price vanilla options (Call and Put) using the Black-Scholes formula. It reflects the market expectations about the realized volatility. At the end of the life of the option, since both sides are usually neutral w.r.t. the stock price (*delta hedged*), the comparison of the realized vol VS the initial implied vol will determine if the option was exchanged at a too expensive or too cheap price.

Our idea is to forecast implied volatility, since it is a human factor and therefore more likely to show patterns that a SVM-based algorithm would capture.

## Classification Approach

To simplify our first approach and still keep it useful for a trader, we turn the forecasting into a simpler classification problem ($+1/-1$ output). Based on the last daily observations of implied volatility $\sigma_t$, and the time left to expiry, we build a simple binary output:

$$Y_t = \begin{cases} +1 \text{ if } \sigma_{t+1} > \sigma_t \\ -1 \text{ if } \sigma_{t+1} < \sigma_t \end{cases}$$

After downloading data from Option Metrics, and implementing an SMO algorithm from [3] we run the following algorithm:

1. To find the best parameters $\varepsilon$ and $C$, we use optimization function `fminbnd` and `fminsearch`, to optimize short cycle SMO optimization. We also implemented an optimization step for the kernel parameter, but did not have enough time to use it.

2. Then we run a new SMO with more iterations to get the coefficients $\alpha$, the intercept $b$ and the prediction $F$.

3. Finally, we compute the error rate as:

$$\text{Error} = \frac{\#\text{ of differences between F and Y}}{\#\text{ of observations}}$$

The results were quite poor, and this method couldn't really anticipate changes in volatility. Our error rates were around 50%. So we decided to use SVM regression, with the modified SMO for regression from [3].

**SVM Regression**

Suppose we are given a training set $\{(x_1, y_1), \ldots, (x_m, y_m)\} \subset \mathbb{R}^d \times \mathbb{R}$, where $x_i's$ are the regressors and $y_i's$ are the observations. In "$\epsilon - insensitive$" measure, our goal is find a function $f(x)$ that has at most $\epsilon$ deviation from the actually obtained target $y_i's$ for all the training data, and at the same time, is as flat as possible. In other words, we do not care about errors as long as they are less than $\epsilon$, but will penalize it otherwise. Consider the linear function

$$f(x) =< w, x > +b \tag{1}$$

with $w \in \mathbb{R}^d, b \in \mathbb{R}$. Flatness in the case of (1) means that one seeks small $\|w\|_2 2$. Formally we can write this problem as a convex optimization problem by requiring:

$$min \frac{1}{2}\|w\|^2$$

$$s.t. \begin{cases} y_i - < w, x_i > -b & \leq \varepsilon \\ < w, x_i > +b - y_i & \leq \varepsilon \end{cases} \tag{2}$$

The tacit assumption in (2) was that such a function $f$ actually exists that approximates all pairs $(x_i, y_i)$ with $\varepsilon$ precision, or in other words, that the convex optimization problem is feasible. Sometimes, however, this may not be the case, or we also may want to allow for some errors in order to gain flatness. Analogously to the "soft margin" loss function, one can introduce slack variables $\xi_i, \xi_i^*$ to cope with otherwise infeasible constraints of the optimization problem. Hence we arrive at the formulation

$$min \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}(\xi_i + \xi_i^*)$$

$$s.t. \begin{cases} y_i - < w, x_i > -b & \leq \varepsilon + \xi_i \\ < w, x_i > +b - y_i & \leq \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* & \geq 0 \end{cases} \tag{3}$$

The constant $C > 0$ determines the tradeoff between the flatness of $f$ and the amount up to which deviations larger than $\varepsilon$ are tolerated. The formulation above corresponds to dealing with a so called $\varepsilon$-insensitive loss function $|\xi|_\varepsilon$ described by

$$\begin{cases} 0 & if |\xi| \leq \varepsilon \\ |\xi| - \varepsilon & \text{otherwise} \end{cases} \tag{4}$$

**Volatility Forecasting Using Implied Volatilities**

The problem where we apply the SVM regression algorithm is autoregressive time series, therefore the formula looks like

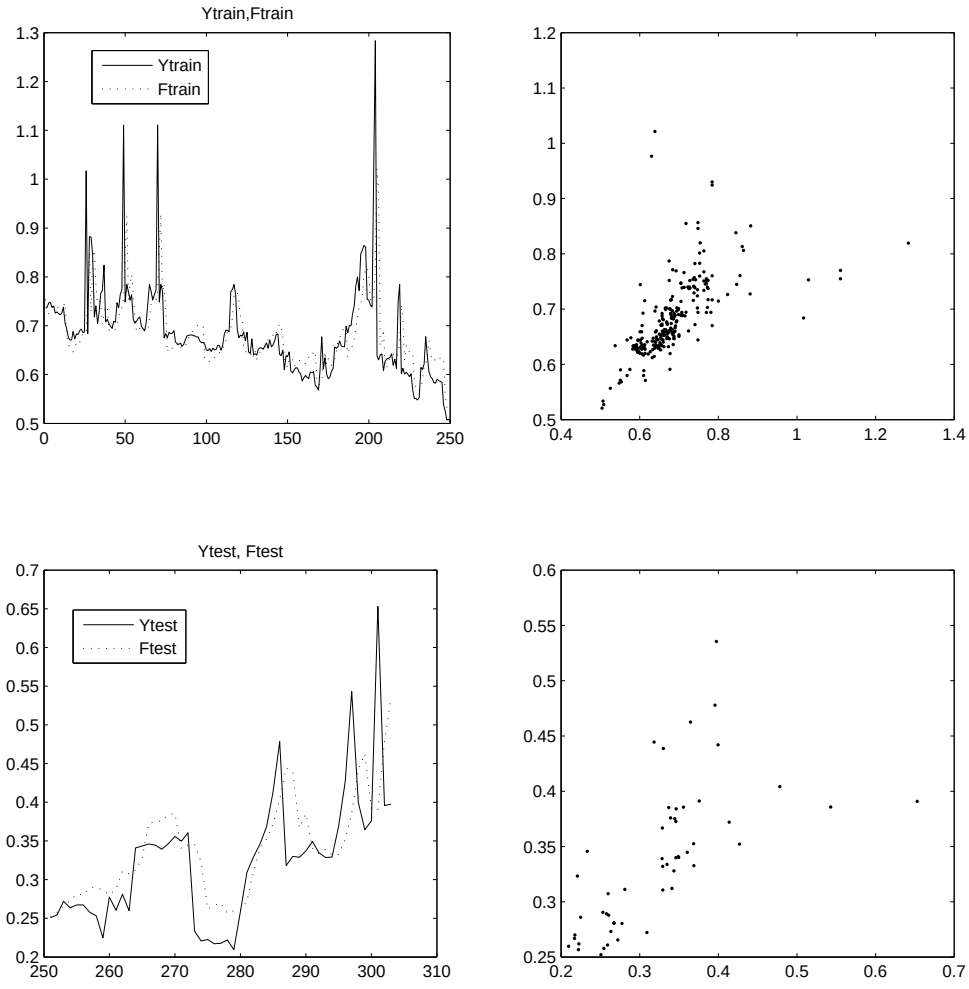
$$\sigma_i = \sum_{j=1}^{l} \beta_j \sigma_{i-j} + \epsilon_i \tag{5}$$

where the $\sigma_i$'s are the implied volatility data and $\epsilon_i$'s are the noises. We use Apple Inc. option data, and set $l = 5$. The daily implied volatility is downloaded from Bloomberg, and the training set is 300 points. In the simulation, we add in the time to maturity parameter as an additional feature. Therefore the formula changes slightly

$$\sigma_i = \sum_{j=1}^{l} \beta_j \sigma_{i-j} + \beta_0 (T - t_i) + \epsilon_i \tag{6}$$

The following figure shows the results obtained with a training sets of 250 dates and a test of 50 remaining dates. Prediction $F$ is unfortunately often late compared to the actual value $Y$, but it can sometimes anticipate an upside or downside move.

Correlation between $F$ and $Y$ looks good for small values, but spreads out with larger volatilities.



**Volatility Forecasting Using Online Learning**

An online version of the SVM regression model has been put forth by several researchers [10, 4, 5]. The online version differs from the standard ("batch") version in that adding one more point to the training set does not require retraining the entire model. Due to the non-stationary nature of financial time series, the most recent training data gives the most information about future values [10,8]. The online version of the SVM reduces the computational burden of updating the regression model as each new data point is revealed.

Volatility clustering is a well known phenomenan in financial time series and it is the basis of our regression approach. The time scales over which this autocorrelation exists, however, tend to be smaller than one day [6,7]. Figure below shows the spectrum of the DM-$ 5-minute volatilities [7]. Most of the information about the time signal exists at frequencies within a day. This shows that prediction for times within the same day might be more accurate than predication for times in another day. An online implementation of SVM regression is required to lessen the burden of updating the model for each new point.

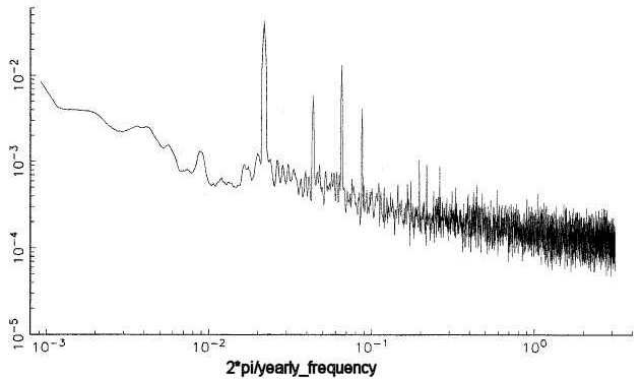



Figure 1: Spectrum for the five-minute absolute returns on an open position of the Deutschemark-U.S. Dollar spot exchange rate market from October 1, 1992 through September 29, 1993

**Online Learning Algorithm**

Briefly, the online learning algorithm for SVM regression works by incrementing the weight of the new point iteratively until the KKT conditions are satisfied for the new point, while ensuring that the KKT conditions remain satisfied for all other points in the training set. The training data can be partitioned into three, non-overlapping sets: the Support set, the Error set, and the Remaining set (S-set, E-set, and R-set, respectively). Eq. 7, 8, and 9 show the definitions of the S, E, and R-sets, where $\theta_i$ is the weighting of the $i$th point in the training set [10]. On each iteration, the online learning algorithm keeps track of the movements of the training samples between these sets.

$$E = \{\ i \mid\ \text{abs}(\theta_i) = C\ \} \tag{7}$$

$$S = \{\ i \mid\ 0 < \text{abs}(\theta_i) < C\ \} \tag{8}$$

$$R = \{\ i \mid\ \theta_i = 0\ \} \tag{9}$$

We were able to obatin a Matlab v.7 implementation of an online SVM regression model from F. Parrella's personal website. The table below shows the results of an SVM regression model for two different values of $\epsilon$. As can be seen, there are very few E-set samples, which leads us to believe that changing C will have little effect on the generalization accuracy of the model. The table below also shows that changing $\epsilon$ has little effect on the RMS error of the model. Note the difference in "online" training time (i.e. the training of the points was done consecutively, however each point was individually "learned").

| Num Points | $\epsilon$ | C | Num E-set | Num S-set | Error (RMS) | Time |
|---|---|---|---|---|---|---|
| 200 | 0.05 | 10 | 1 | 56 | 0.500 | 10 min |
| 200 | 0.01 | 10 | 0 | 135 | 0.503 | 1 hr |

### Improvements on SVM Regression for Financial Time-series

There have been several articles on SVM regression using point-specific C and/or $\epsilon$ parameters [8]. In the traditional SVM regression problem we assign a constant model complexity term ($\epsilon$) and constant trade-off between minimizing model complexity and generalization error (C). By having point-specific C and $\epsilon$ values, the contribution of a point to the generalization error and model complexity can be changed on a point-by-point basis (i.e. more distant points might have a lower C). The online learning alogrithm can easily be extended to implement this feature in an efficient way.

### Conclusion

The recent rise of quantitative funds using automatic trading algorithms has developed an intensive research of typical patterns in the market, that would reduce risk and increase profits. Some of them are based on cutting-edge machine learning techniques.

Even if we did not reach a satisfying level of confidence in our predictions, wether binary or exact, we have reviewed many different aspects of machine learning that will certainly help us in our future jobs.

### Reference

[1] Lijuan Cao and Francis E.H. Tay, Neural Comput & Applic (2001)10:184C192

[2]Alex J. Smola and Bernhard Scholkopf, NeuroCOLT Technical Report Series, NC2-TR-1998-030

[3] Learning with Kernels, Bernhard Schlkopf and Alex Smola (MIT Press, Cambridge, MA, 2002)

[4] Mario Matrin, "On-line Support Vector Machines for Function Approximation," Technical Report; Software Department, University Politécnica de Catalunya (2002)

[5] Francisco Parrella, "Online Support Vector Regression," Thesis; University of Genoa, Italy (2007)

[6] Torden Anderson and Tim Bollerslev, "Intraday periodicity and volatility persistence in financial markets," Journal of Empirical Finance 4 (1997) 115-158

[7] Torden Anderson and Tim Bollerslev, "Heterogeneous Information Arrivals and Return Volatility Dynamics: Uncovering the Long-Run in High Frequency Returns," The Journal of Finance, Vol. LII, No. 3, (1997)

[8] Lijuan Cao and Francis E.H. Tay, "Support Vector Machine with Adaptive Parameters in Financial Time Series Forecasting," IEEE Transactions on Neural Network, Vol. 14, No. 6, (2003)

[9] "Online Support Vector Regression," Personal website, Created by Francisco Parella; Viewed on December 12, 2007 (http://onlinesvr.altervista.org/)

[10] Junshui Ma, James Theiler, and Simon Perkins, "Accurate On-line Support Vector Regression