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 σ_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 ε 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 α, 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 ϵ 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 ϵ, 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 ε 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 ξ_i, ξ_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 ε are tolerated. The formulation above corresponds to dealing with a so called ε-insensitive loss function $|\xi|_\varepsilon$ described by

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