Forecasting GOOGL Price Direction Using Support Vector Machines

Peter Kokalov

1. Introduction:

This goal is this paper is to explore the predictability of daily movements in the financial market using various binary classification methods, with a specific focus on the Support Vector Machine. I evaluate the accuracy of the forecasting using accuracy, sensitivity, and specificity metrics on NASDAQ:GOOGL. Additionally, backtesting a trading strategy using the long-short signals created from the model produced interesting results examined below.

Along with being shrouded by intensive and unexpected noise, financial markets tend to exhibit non-stationarity and are thus difficult to model in a linear fashion. This makes the process of signal generation difficult. However, practitioners and researchers are beginning to uncover significant dynamic factors that can be used as parameters for various statistical and "machine-learning" algorithms capable of producing high degrees of accuracy, although sometimes at the expense of interpretability. This paper focuses on the use of three classification algorithms: The Support Vector Machine (SVM), Quadratic Discriminant Analysis (QDA), and Logistic Regression, with the goal being: given an observation x_i of p features at time t_i, predict whether Google's share price at time t_{i+1} will be higher or lower than it is at time t_i,.

2. Preprocessing and Feature Input:

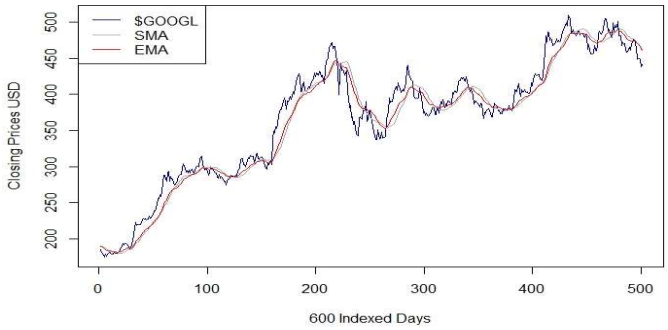
The number of features included during preprocessing was large and consisted of Google's closing price, all stocks in the S&P 500 Index, five lagged returns of Google given as

$$\frac{x_t - x_{t-i}}{x_{t-i}}$$

where x_t represents the share price of Google at day t and i represents the look-back window, along with 18 technical indicators, for example the Simple Moving Average (SMA) with n = 20 as

$$\frac{\sum_{i=1}^{n} x_i}{n}$$

Preprocessing and data exploration was done in Python using AlphaVantage's API (alphavantage.co). Intra-day data from 2004-10-14 to 2018-01-17 was collected, stored into a Pandas DataFrame object and exported as a csv file to R.



Given the high-dimensional feature space (446 input features) of likely correlated variables, a Principal Component Analysis (PCA) was used on our input matrix. PCA allows us to summarize our variables into a smaller number of representative variables that collectively explain most of the variability in the original set.

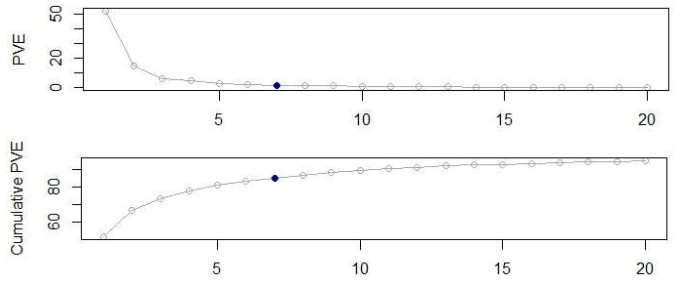
The first principal component in our set of p features $X_1, X_2, \ldots, X_p$ is the linear combination of the features

$$Z_1 = \emptyset_{11}X_1 + \emptyset_{21}X_2 + \ldots + \emptyset_{p1}X_p$$

that has the largest variance, subject to $\sum_{j=1}^{p} \emptyset_{j1}^2 = 1$, as otherwise we could blow up the loadings arbitrarily large. With the variables in our input matrix centered to have mean zero, PCA solves the optimization problem

$$\underset{\emptyset_{11},\ldots,\emptyset_{p1}}{\text{maximize}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \left(\sum_{j=1}^{p} \emptyset_{j1} x_{ik} \right)^2 \right\} \text{ subject to } \sum_{j=1}^{p} \emptyset_{j1}^2 = 1.$$

Since related batches of equities often move in tandem, the dimensionality of our original feature space was reduced to 7 predictor variables (the first 7 principal components) explaining ~90% of space variance.



Although using dim-reduction for classification is great in terms of computational cost, it comes with a different price - the interpretability of the model suffers because we are unable to decipher which of the original features were used to explain the largest proportions of variance in our principal components.

3. Modelling:

The data was trained on roughly the first 80% of observations and tested on the last 20%. The classification results are below ordered as SVM, Log-Reg, and QDA.

	Positive	Negative	Total
Positive	*TP* = 180	*FN* = 133	313
Negative	*FP* = 165	*TN* = 159	324
Total	345	292	*N*

The accuracy of the SVM was 53.22%, the sensitivity at 57.51%, and the specificity at 49.07%.

	Positive	Negative	Total
Positive	*TP* = 229	*FN* = 174	313
Negative	*FP* = 116	*TN* = 118	324
Total	345	292	*N*

The accuracy of the Logistic Regression was 54.47%, the sensitivity at 56.82%, and the specificity at 50.43%.

	Positive	Negative	Total
Positive	*TP* = 149	*FN* = 104	313
Negative	*FP* = 196	*TN* = 188	324
Total	345	292	*N*

The accuracy of the QDA was 52.90%, the sensitivity at 58.89%, and the specificity at 48.96%. Although Logistic Regression obtained the highest accuracy, even by a small margin, the uniqueness of the Support Vector Machine played consequence to some very interesting results during the backtest of the classifiers.

4. The Support Vector Machine:

The Support Vector Machine is a supervised learning algorithm most often characterized by the capacity control of the decision boundary, it's structural risk minimization (SRM), and it's use of kernels. The cool thing about SRM is that rather than seeking to minimize training error, Support Vector Machines attempt to minimize an upper bound on the generalization error and are hence often impervious to overfitting.

The Support Vector Machine uses an approach based on the concept of creating a separating hyperplane between classes - the assumption being that the hyperplane which not only separates the points, but also maximizes the distance between them,

Forecasting GOOGL Price Direction Using Support Vector Machines

Peter Kokalov

will generalize best. It is an extension to the support vector classifier given as the more intuitive optimization problem

$$\underset{\beta_0,\beta_{1,\ldots},\beta_p,\epsilon_{1,\ldots},\epsilon_n}{\text{maximize}}\ M \text{ subject to } \sum_{j=1}^{p}\beta_j^2 = 1,$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) \geq M(1-\epsilon_i),$$

$$\epsilon_i \geq 0, \sum_{i=1}^{n}\epsilon_i \leq C,$$

where C is a nonnegative tuning parameter and M is the width of the margin. This highlights the goal of maximizing margin while keeping observations on the right side of hyperplane, to some extent. This extent is determined by C which is the cost parameter bounding the severity of margin violations ϵ_i. In other words, C controls our bias-variance trade-off.

Hinting again at the inherent structural risk minimization of the Support Vector Machine, we can observe that only the support vectors, which are those observations that lie directly on, or on the wrong side of the margin, have any impact on our margin M.

Now, to define the Support Vector machine, we can use the more formal definition. Let's define the classifier as:

$$w^T x + b = 0$$

where the assignment of new observation z to $\{-1,1\}$ is given as

$$h_w(z) = \text{sign}(w^T x + b),$$

where w are the weights of each classifier, and b is the bias which effectively removes the constraint that the normal w to the hyperplane needing to pass the origin of the feature space. As a combined equation with labelled observations (x_i, y_i), we the have the constraint

$$y_i \cdot (w^T x_i + b) \geq 1$$

To maximize margin, we need to minimize

$$\min_{w,b,\varepsilon}\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\varepsilon_i,$$

$$\text{subject to } y_i \cdot (w^T x_i + b) \geq 1 - \varepsilon_i, \forall i \in [l],$$

$$\varepsilon_i \geq 0, \forall i \in [l]$$

So, what's so special about the Support Vector Machine? In an enlarged feature space such as the one we're using to classify GOOGL's price direction, the solution to our hyperplane is often inherently non-linear along with the fact that computing in high dimensions can become infeasible. This is why we use kernels. By keeping the dot product of observations constant through higher dimensions, a kernel can map the original finite-dimensional space onto a much higher dimensional space in attempt to make the separation of classes easier, i.e. more linear. In other words, we can find a linearly separable boundary by searching through a higher dimension, then map the same solution back to the original non-linear space, giving us our decision boundary!

In this project, we use a Gaussian Radial kernel which takes form

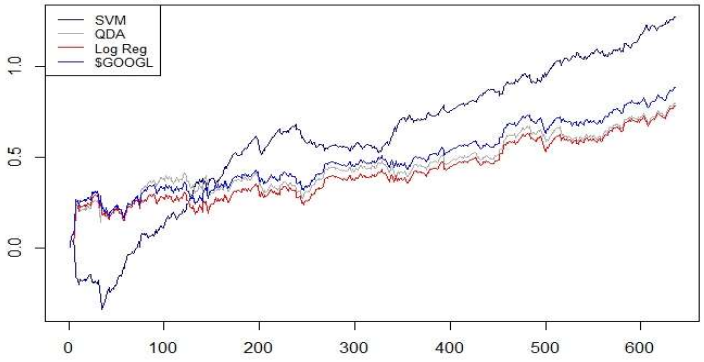
$$K(x_i, x_{i'}) = e^{\left(-\gamma\sum_{j=1}^{p}(x_{ij}-x_{i'j})^2\right)}$$

where γ is a positive constant. The radial kernel works as follows: Given a test observation $x^* = (x_1^* \ldots x_p^*)^T$, if x^* is far from a training observation x_i in terms of Euclidian distance, then $\sum_{j=1}^{p}(x_j^* - x_{ij})^2$ will be large, and thus $K(x_i, x_{i'})$ will be small. Thus, the kernel has local behavior and only nearby observations have an affect on our classifier.

In the modelling, our optimal cost parameter C was 1 and parameter gamma γ, was set to 0.1.

5. Backtesting:

Using the signals created from the classifiers, I ran a long-short strategy during the validation period of July 2015 to January 2018. The algorithm was constructed as follows: If output signal equals +1, buy GOOGL at market open, elseif output signal equals -1, short GOOGL at market open. Theoretical settlement occurred daily at closing price and transaction fees were ignored. The returns of each algorithms against one another are displayed in the diagram below.



As seen in the plot, the SVM maintained short positions during large positive spikes in the first few months of trading and underwent drastic drawdowns. The Logistic Regression and QDA followed GOOGL's price movement well throughout the entire period but ended up slightly underperforming on the backtest, with QDA ending slightly up on Log Reg. Although the SVM displayed the worst drawdown, it led competitors in its absolute return and Sharpe ratio. A comparison between being long GOOGL and the SVM Long-Short portfolio is displayed below in the table.

	Absolute Return	Sharpe	Max Drawdown
SVM Long-Short	127.54%	2.24	39.31%
GOOGL Long	79.87%	1.39	14.04%

6. References:

[1] En.wikipedia.org. (2017). *Support Vector Machine* [online]. Available at: https://en.wikipedia.org/wiki/Support_vector_machine.

[2] En.wikipedia.org. (2017). *Kernel Method* [online]. Available at: https://en.wikipedia.org/wiki/Kernel_method.

[3] Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani, *An Introduction to Statistical Learning with Applications in R.* Springer Texts in Statistics, 2013.

[4] https://jeremykun.com (2017). *Formulating the Support Vector Machine Optimization problem.* [online] Available at: https://jeremykun.com/2017/06/05/formulating-the-support-vector-machine-optimization-problem/