Two Optimization-based Approaches to Portfolio Construction

Debarghya Das
dd367

Abstract

In our project, we explore two optimization-based techniques in portfolio construction. Firstly, we attempt to formalize a model that relates the volatility index of a stock to the stock prices in a unique way. We model an investor's risk aversion as a lognormal distribution, and establish a continuous one-to-one relationship between the volatility index and the risk aversion parameter in a typical Markowitz model. This shall henceforth be referred to as the Predictive Risk Aversion Model (PRAM).

Secondly, we attempt to use a binary Support Vector Machine to predict the direction of the movement of stock prices. We experiment with various kernels, and fine tune our parameters to optimize our results. We use distance of the test data to the hyperplane as a confidence metric, and thereby try to construct a portfolio from it. This shall henceforth be known as the Confidence-based Support Vector Machine model (CSVM).

1 Predictive Risk Aversion Model

1.1 Motivation

Several factors motivate this model:

1. **Poor Predictive Power** The simplicity of the Markowitz model doesn't accurately allow us to predict the direction of future returns of a given stock. The arithmetic mean of the last i returns on a stock isn't very indicative of future returns.

2. **Arbitrary Constant Parameters** The various formulations of the Markowitz model involves different arbitrary user-defined parameters. The commonly used risk aversion parameter δ in the objective function

$$\max_x \ \mu^T x - \frac{1}{2}\delta(x^T V x)$$

is kept throughout an investment period. This is quite unreasonable.

3. **Unrealistic ability to purchase fractional stocks** A major known critique of the Markowitz model is the allowance to purchase non-integral amounts of stock [5].

1.2 Introduction

We attempt to improve upon the pertinent inaccuracies of the Markowitz Model as follows:

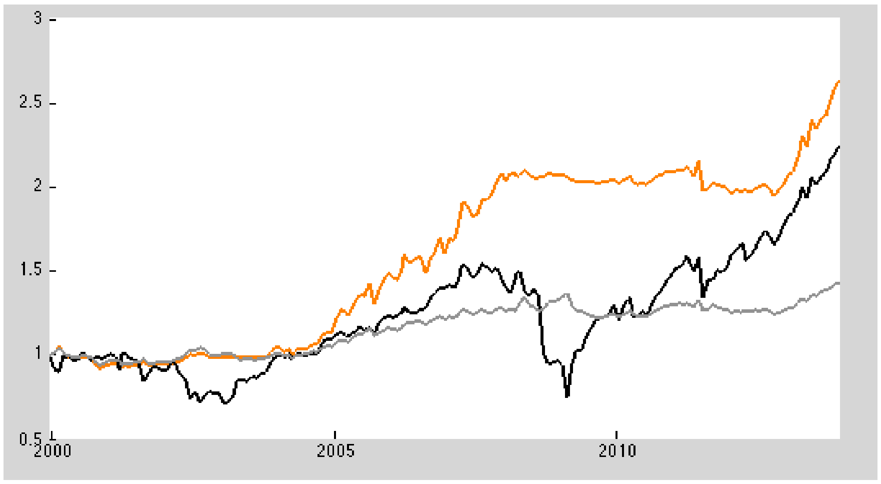
1. **Using ^VIX for greater predictive power** VIX is the Chicago Board Options Exchange (CBOE) volatility index for the S&P 500. It is commonly used in literature [9] [15] as one of the most accurate predictors of risk in the market.

2. **Modeling prior volatilities as a Gamma distribution** On account of its vast application in Econometrics [4] [13] and it's flexibility, it is particularly good fit in our particular model. It outperformed the LogNormal and Weibull distribution on various metrics.

3. **Modeling investor risk tolerance as a LogNormal distribution** Markowitz sets hard parameters for risk tolerance of an investor which is neither feasible or functional. Modeling risk tolerance as a distribution with a particular mean μ and variance σ^2 vastly generalizes the scope of the model.

4. **Establishing a continuous mapping from volatility to risk tolerance** Some previous works have dealt with using volatility to establish risk tolerance, but we bring forward a unique one-to-one mapping which makes the most sense for an investor. The LogNormal model pulls risk tolerance to 0 on high volatility, and pushes risk tolerance to ∞ on abnormally low volatility.

Figure 1: *Comparison of the 3 trading models starting with $w_{2000} = 1$. The orange line represents the unconstrained PRAM model with $\sigma_\mu = 0.015$ and $\sigma_\sigma = 1$. $w_{2014} = 2.57$ The black line represents a simple baseline with wealth equally distributed to all assets. $w_{2014} = 2.45$ The grey line represents the original Markowitz model with $\sigma = 0.015$.* $w_{2014} = 1.43$



5. **Add reasonable constraints for further realistic approximation for the small investor** We add several constraints such as *restrictions on shorting* stocks, *purchasing only integral amounts of stock* and *restricting drastic portfolio fluctuations*.

1.3 Description

1.3.1 Theory

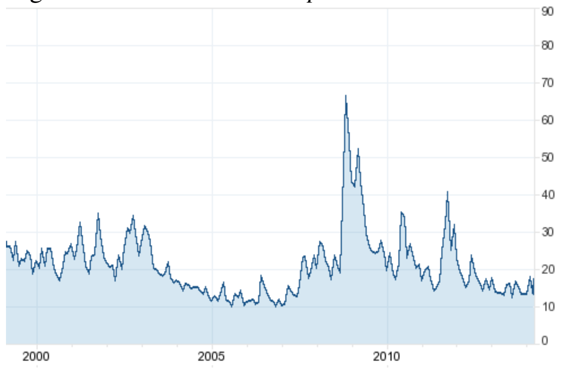
Firstly, we define what **ˆVIX** really is:

VIX is a trademarked ticker symbol for the Chicago Board Options Exchange Market Volatility Index, a popular measure of the implied volatility of S&P 500 index options. Often referred to as the fear index or the fear gauge, it represents one measure of the market's expectation of stock market volatility over the next 30 day period.

Quantitatively, the VIX is the percentage expected annualized change in the next 30 days. High values of VIX are correlated with drastic market shifts and low values of VIX indicate low market shifts. This is what makes VIX an adequate risk indicator. Figure 2 shows the variation of VIX over our backtesting period. We see clear spikes during the 2008 recession and 2010 Eurozone crisis.

Previous works have shown arbitrary binary distinctions between values of VIX and choices of risk parameter. We attempt to model the distribution of values of VIX as a *Gamma* distribution. This gave us the best results amongst other purely positive distributions

Figure 2: *The ˆVIX over the period we backtested in.*



such as *Weibull* and *logNormal*. Further, we chose *Gamma* for its wide use in Econometrics as well it's high flexibility. Figure 3 shows how VIX values fits well with the Gamma distribution.

To establish a one-to-one mapping between VIX and our risk threshold, we chose to model the investor's risk as a certain distribution centered at a given mean μ. Initially, we chose a *Normal* distribution, but this was not particularly suitable at modeling the variance bound of a portfolio. This is because Normal distributions:

1. Would have to be clamped at 0.

2. During strong indications of positive return, fail to reach high enough values.

The most natural distribution of **investor risk** seemed to be a *LogNormal* distribution. It rids us of both these issues - maintaining both a consistently positive distribution with large risk tolerances at low VIX indices.

Figure 4 shows our LogNormal distribution.

Figure 3: *Approximating the distribution of ˆVIX to a* $\Gamma(k,\theta)$. *The complete range of values approximates to* $k = 7.2822$ *and* $\theta = 0.0081$

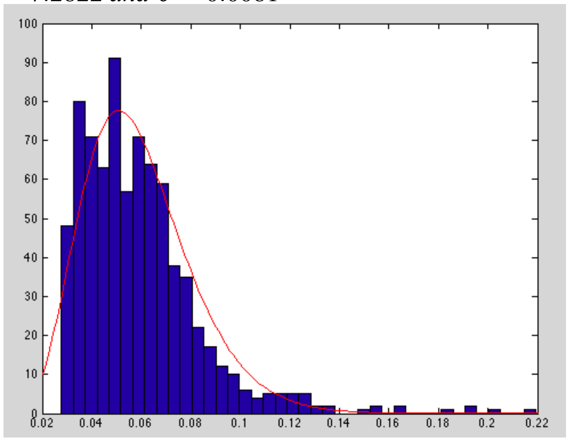
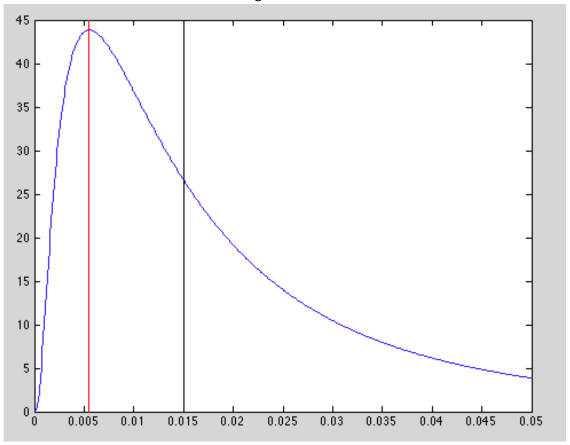


Figure 4: *The distribution of* σ, *the risk tolerance metric, in our PRAM model. The black line indicates the mean* $\sigma_\mu = 0.015$ *and the red line indicates the mode of* 0.055. *The standard deviation* $\sigma_\sigma = 1$.



1.3.2 Algorithm

Here is the relevant snippet of code for our sigma approximation:

```
samples = (number_of_samples:-1:1);
weights = (1 - rate_of_decay).^samples;
weights = weights ./ sum(weights);
vix_start = a_dates(i) - number_of_samples + 1
vix_i = vix(vix_start:a_dates(i));
sigma = weights*vix_i;
params = gamfit(vix(1:a_dates(i)));
risk_prob = gamcdf(sigma,params(1),params(2));
sigma = logninv(1-risk_prob, sigma_mu, sigma_sig);
```

1.3.3 Mathematics

Mathematically, since volatility v_{ix} is a measure of annualized percentage change for the next 30 days, we must scale the value appropriately for it to have any significance. For a trading horizon h, this is given by:

$$v = \frac{v_{ix}}{100\sqrt{52/h}}$$

. At iteration M, we weight the last N samples of **ˆVIX** $\gamma_M = [v_{M-N}, v_{M-N+1}, ..., v_i]$ with weights $W = [w_1, w_2, ..., w_N]$ where $\sum_{i=1}^{N} w_i = 1$ and $w_i = (1-r_w)^i$ for some weekly rate of decay r_w such that $r_w < 1$.

We use an annualized rate of decay $r_a = 0.5$, the relation between the two being $r_w = r_a^{1/52}$.

The weighted **ˆVIX** $\bar{\gamma}_M = \gamma_M W'$ is then computed. We then proceed to estimate the parameters of the Γ_M distribution for **ˆVIX** at time i using the Maximum Likelihood Estimator:

$$\hat{k}_{mle}, \hat{\theta}_{mle} \subseteq \{\arg \max_{k,\theta \in \Theta} \hat{l}(k,\theta | v_1, v_2, ..., v_M)\}$$

For a Gamma distribution, the explicit form of the MLE is:

$$\hat{\theta} = \frac{1}{kM} \sum_{i=1}^{M} v_i$$

k, in Gamma distributions do not have a closed form solution, but can be approximated as follows:

$$s = \ln\left(\frac{1}{M}\sum_{i=1}^{M} v_i\right) - \frac{1}{M}\sum_{i=1}^{M} \ln(v_i)$$

$$k \approx \frac{3 - s + \sqrt{(s-3)^2 + 24s}}{12s}$$

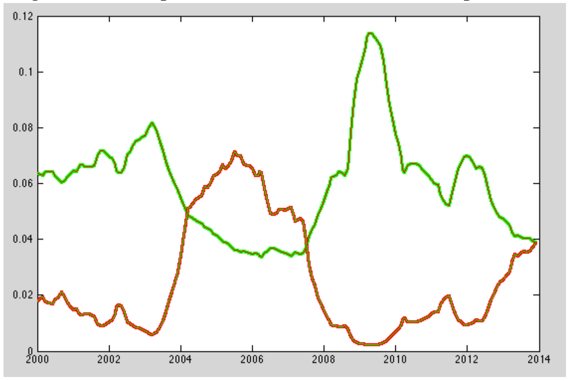
Using these values to compute $\Gamma(k,\theta)_M$ and proceed to find the cdf $P(\gamma_i \leq \hat{\gamma})$ or $\Phi_{k,\theta}(\hat{\gamma})$.

This value $\Phi_{\Gamma(k,\theta))_M}(\hat{\gamma})$ is an indicator to *how* volatile the index is compared to the norm. Risk tolerance is inversely related to risk. Therefore, the corresponding risk tolerance for timestep M is given by computing the inverse of the 1 minus the computed cdf of risk as follows:

$$\sigma_M = \Phi^{-1}_{\ln\mathcal{N}(\sigma_\mu, \sigma_\sigma^2)}(1 - \Phi_{\Gamma(k,\theta))_M}(\hat{\gamma}))$$

Essentially, this creates a one-to-one mapping between volatility and sigma for a particular timestep. Note that σ_μ is the investor's risk tolerance mean and $\sigma_\sigma = 1$ is the standard deviation of the investor's risk tolerance. Figure 5 shows the change of σ_M with time in relation to the change of weighted VIX $\hat{\gamma}$.

Figure 5: *Comparison of Risk Tolerance σ and VIX v. The green line represents v and the red line represents σ.*



1.3.4 Additional Small Investor / Reality Constraints

In addition to our primary goal, we added several constraints to our optimization model to more realistically emulate reality [1] and/or to emulate trading from the perspective of a small-time investor. [12] These include:

1. **Disallow shorting** Given a portfolio x of stocks from $1,2,...N$, where x_i is the portion of wealth invested in asset i, and $\sum x_i = 1$, we enforce the constraint $\min_{1\leq i\leq N} x_i \geq 0$. Positive portfolio weights semantically indicates the inability to take short positions.

2. **Allow only integral stock purchases** Given a portfolio x of stocks from $1,2,...N$, where x_i is the portion of wealth invested in asset i, and $\sum x_i = 1$, and assets A_i priced at $p(A_i)$, and wealth w, we enforce

$$\frac{wx_i}{p(A_i)} \bmod 1 = 0 \ \ \forall 1 \leq i \leq N$$

   This is semantically equivalent to the real life enforcement that you can only be integral amounts of a given stock.

3. **Restricting drastic portfolio modifications - Intertia Constraints** Although unnecessary given our optimization problem, we decided to experiment with certain restrictions on portfolio modifications at each timestep. Given a portfolio vector at timestep M x_M and at timestep $M+1$ x_{M+1}, we enforce

$$\|x_{M+1} - x_M\|_1 \leq 0.3$$

   This represents the L_1 norm of the portfolio change. Intuitively, it means that an investor cannot reallocate more than 30% of his assets at one timestep.
   Another arbitrary constraint we enforced was:

$$| x0_{M+1} - x0_M | \leq 0.2$$

   This intuitively means that the amount an investor has allocated in risk-free assets cannot change more than 20% in one iteration.
   We dub these seemingly arbitrary constraints **Inertia constraints**. The reason they are included is to preserve inertia in our portfolio. At each timestep, solving an optimization problem can be locally misdirected given a series of optimizations before it. These constraints preserve the solutions of the optimizations at previous timesteps.

1.3.5 Additional Legacy Constraints

Several other constraints that were left unchanged from the Markowitz are the following:

1. **Horizon** The interval at which trading occurs, in weeks. $h = 4$

2. **Sample Frequency** The interval at which stock prices are sampled, in weeks. $s_f = 1$

3. **Number of Samples** The number of samples used to weight the volatility and stock prices to project mean. $n_s = 50$

4. **Rate of Decay** The rate of decay of the weighting, annualized. $r_a = 0.5$

5. **Start Date** The week at which the backtest begins. We use this number to allow adequate data from the beginning of our backtest. $s_d = 55$

6. **End Date / Number of Rebalances** The number of times the portfolio rebalances in our backtest. We usually set this to the last date possible to allow a full backtest.

7. **Transaction Cost** The fraction of wealth the investor must relinquish when making a trade. We retain the value 1.5%. $t_c = 0.015$

1.4 Data

For our backtest, we obtained fresh data from Yahoo! Finance. Our data extends from 1st Jan, 1999 to 1st Jan, 2014 - a period of 15 years. This is the extent that is covered both by **ˆVIX** as well as all the SPDR ETFs. The **SPDR** Exchange Traded Funds is a family of 9 funds traded in the United States, Europe and Asia-Pacific. This data consisted of 782 observations. We also used 30-day Treasury bills as our risk free rate.

1.5 Experiment and Results

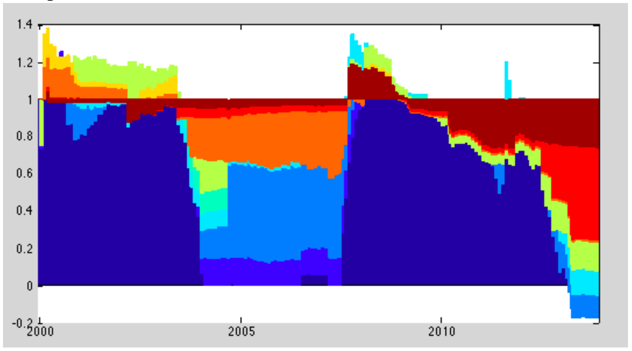
Although we performed various experiments in order to fine-tune our parameters, there were 2 backtests that are particular important - one without our realistic constraints, and one with. Both of them successfully beat our 2 baselines - a uniformly distributed portfolio in the SPDR ETFs and the initial Markowitz model baseline with $\sigma = 0.015$.

1.5.1 Without Constraints

The performance of our unconstrained portfolio can be seen in Figure 1 and Figure 9. In Figure 6, we include a chart showing the variation in the allocation to the portfolio from Figure 1. It demonstrates the high allocation of wealth in risk-free assets during high risk periods such as 2008 and 2010. In Figure 7, we show the returns of the various assets with time in comparison to our portfolio. Figure 8, however, is probably the most demonstrative of the benefit of PRAM over an ordinary baseline and Markowitz. We recall that the

2000 - 2014	Old M'witz	PRAM	Baseline
Annualized Return	2.59%	6.98%	5.83%
Standard Deviation	1.54%	2.47%	5.06%
Mean Return	0.21%	0.58%	0.58%
Total Return	1.4311	2.6297	2.212

Figure 6: *The change in the allocation of the portfolio with time for our unconstrained portfolio. We see high allocation in risk-free assets - in navy blue during high risk periods.*



only constraints/parameters used for this backtest were $\sigma_\mu = 0.015$, $\sigma_\sigma = 1$, and $t_c = 0.015$, besides the common settings mentioned in 1.3.5.

1.5.2 With Small Investor / Realistic Constraints

For our small investor constraint simulation, we include the constraints in 1.3.4. We start with an initial wealth of

Figure 7: *The returns for our optimal portfolio in orange. The lines in other colors represent the returns from the 9 ETFs.*

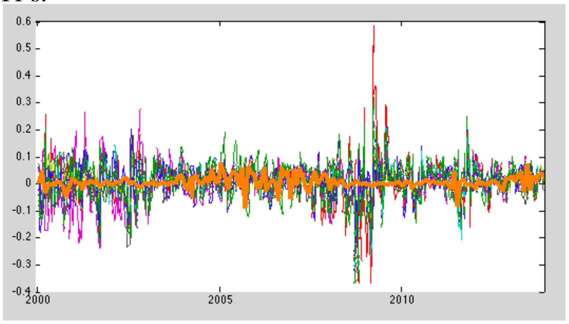


Figure 8: *The returns of the baseline portfolio (in blue) and the PRAM (in red) shown as a normal distribution. Although both models return the same average return, as Markowitz is a bad predictor of return anyway, PRAM's reduction of over 2x makes a huge difference in the long run.*

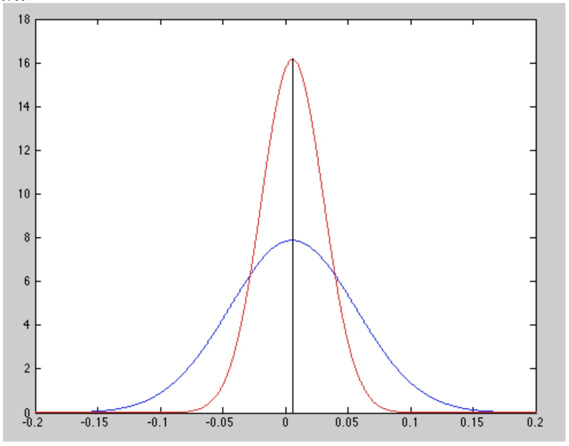
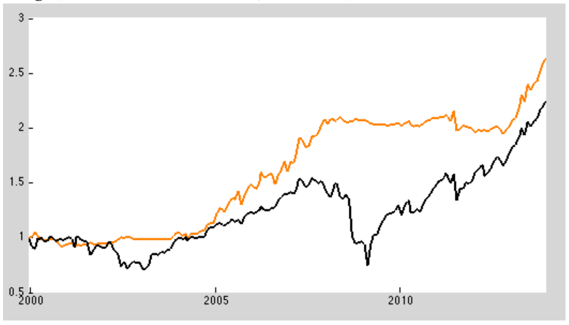


Figure 9: *A comparison of performance of the PRAM (in orange) and the baseline (in black).*



$5000. We dub this constrainted PRAM (cPRAM). This performs a little worse than PRAM on all parameters, as expected, but still achieves the same results overall in comparison to the 2 baseline models.

	Old M'witz	cPRAM	Baseline
Annualized Return	2.59%	6.58%	5.83%
Standard Deviation	1.54%	2.33%	5.06%
Mean Return	0.21%	0.53%	0.58%
Total Return	1.4311	2.411	2.212

Figure 10: *A comparison of performance of the cPRAM (in orange) and the baseline (in black).*

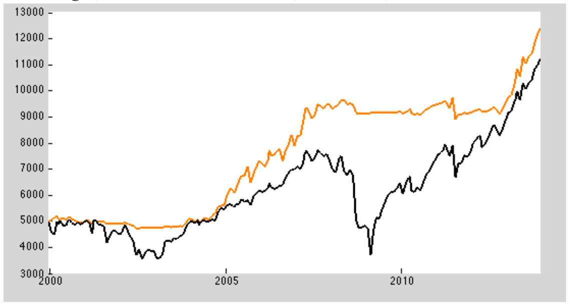
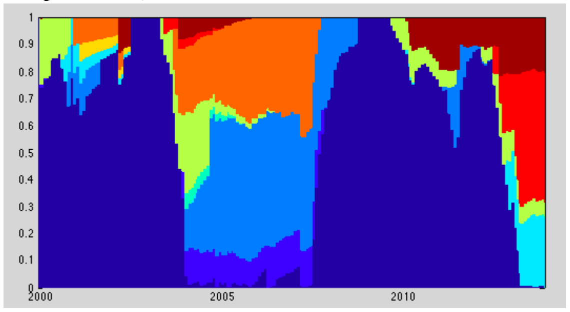


Figure 11: *The change in the allocation of the portfolio of cPRAM with time for our unconstrained portfolio. Notice how the allocations sum to 1 at each timestep (no short positions).*



1.6 Conclusion

A quick summary of the achievements of our model are:

1. Although widely stated in literature [8], we confirm the volatility index as a suitable predictor for stock market prices.

2. We build a generalized scalable model that relates **volatility** and **risk tolerance**. Our model was *untouched* by the 2008 recession.

3. We **reduce risk by over 2x** from a standard baseline.

4. We **increase overall return by 4x** from the standard Markowitz model.

5. We enforce **realistic constraints** on our model and demonstrate that it has little overall effect.

6. We enforce **small investor constraints** on our model and demonstrate that it has little overall effect.

1.7 Future Work

Although there are infinite scopes for future work, we narrow it down to some crucial ones:

1. **Modeling Return Better** While PRAM focusses on *reducing risk*, it doesn't focus on *increasing return* as much. Experimenting with various return predictors would be interesting.

2. **Big Data** We narrowed our dataset down to tiny pieces for the purpose of this project, but it would be interesting to see how this model could scale on a much bigger dataset of stock prices.

3. **Different Trade frequency** It would be interesting to do further research on how this model would play with lower or higher horizons.

2 Confidence-based Support Vector Machine

2.1 Motivation

Most common modern portfolio theory methods rely on some kind of feature extraction in order to determine the direction of movement of stock prices. However, using **Support Vector Machines** [2], we attempt to formalize a completely novel way of looking at this problem. Here are some reasons we decided to use an SVM:

1. Instead of trying to gauge what features are indicators of future stock prices, we attempt to use a Support Vector Machine to *learn* the pattern so it can predict the direction of market movement.

2. Because SVMs are *agnostic of the distribution of its input data*, it is particularly useful for our purposes.

2.2 Introduction

Support Vector Machines are a supervised machine learning method broadly used as a binary classifier (although it can be extended to be a multi-class classifier). SVMs are first trained with a bunch of input vectors and their respective output. It then uses a kernel function to transform the space of the input, and proceeds to find the maximum margin hyperplane that separates the binary classes.
After training, the vectors to be classified are similarly transformed to the appropriate space using the kernel function. The SVM simply figures out which side of the hyperplane the vector lies on in order to spit out a prediction for the vector's class. The vectors which form the hyperplane's maximum margin are referred to as *support vectors*. Figure 12 demonstrates the functionality of a simple SVM.
Several new ideas we'd like to introduce with our cSVM model include the following:

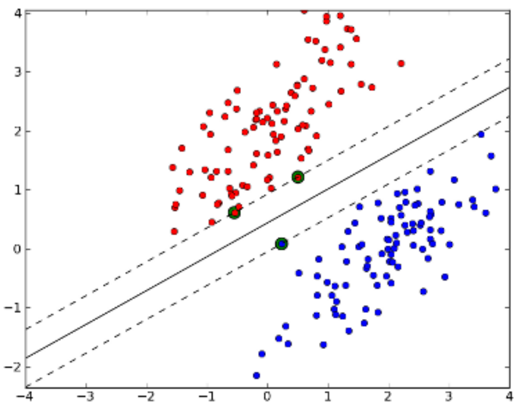
1. **Efficient Binary classification of direction of stock market movement** The first and foremost utility of our SVM is training on vast amounts of data to classify the future direction of the stock market. For SVMs, this involves a lot of parameter estimation, kernel searching, and more.

2. **Using a function of the confidence of the classification to compute portfolio weights** The binary nature of an SVM makes it difficult for us to actually compute a portfolio from it's predictions. We therefore use *distance to the hyperplane* as a criteria for estimating the confidence of a classification. We proceed to use a function of this confidence to assemble reasonable portfolio weights.

3. **Kernel Function Estimation** We attempt to figure out a kernel function along with it's parameters which maximize the accuracy of the SVM.

4. **Determination of SVM Paramters** We have yet to figure out what input data we'd like to train the cSVM with - stock prices, returns, normalized prices, etc.

Figure 12: *A binary SVM which classifies input vectors of size 2 with no kernel using the maximum margin hyperplane. The circled vectors are the support vectors.*



2.3 Description

We first mathematically formulate an SVM. Given a data set for training (x_i, y_i), for $i = 1, 2, ..., N$, with $x_i \in \mathbb{R}^n$ and $y_i \in \{-1, +1\}$, we map these input vectors into a higher dimensional kernel Hilbert space where a linear hyperplane suffices for binary classification. The hyperplane can be stated as $f(x) = \mathbf{w}\phi(x) + b$, where $\phi(.)$ is the mapping function to a different space, b is the bias. Hence, the objective function we try to minimize is

$$C \sum_{i=1}^{n} l(y_i, f(x_i)) + \frac{1}{2} \|\mathbf{w}\|^2$$

where $l(y_i, f(x_i))$ is the loss function. As we do not intend to use a *hard margin* which allows for absolutely no training error, but use *soft margin* instead, we reformulize our optimization as

$$\min_{w,b,\xi} C \sum_{i=1}^{n} \psi_e(\xi_i) + \frac{1}{2} \|\mathbf{w}\|^2$$

subject to

$$y_i(w\phi(x_i) + b) \geq 1 \ \forall i$$

and

$$\xi_i \geq 0 \ \forall i$$

where

$$\psi_e(\xi) = \begin{cases} \frac{\xi^2}{4\varepsilon} \text{ if} \xi \in [0, 2\varepsilon] \\ \xi - \varepsilon \text{ if} \in [2\varepsilon, +\infty] \end{cases}$$

The Lagrangian for the *primal* formulation is:

$$L(w,b,\xi) = C\sum_{i=1}^{n} l(y_i, f(x_i)) + \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \gamma_i \xi_i$$
$$- \sum_{i=1}^{n} \alpha_i (y_i(w\phi(\xi_i) + b) - 1 + \xi_i)$$

Applying the *Karush-Kuhn-Tucker Conditions*, and reformulating the problem as a *dual* problem as maximization in terms of dual variables α_i and γ_i.

$$\max_{\alpha,\gamma} R(\alpha,\gamma) = \sum_{i=1}^{n} \alpha_i - \frac{\varepsilon}{C}\sum_{i=1}^{n} (\alpha_i + \gamma_i)^2$$
$$-\frac{1}{2}\sum_{i=1}^{n}\sum_{i=1}^{n} \alpha_i \cdot y_i \cdot \alpha_j \cdot y_j \cdot \langle \phi(x_i) \cdot \phi(x_j) \rangle$$

subject to

$$\alpha_i \geq 0 \ , \ \gamma_i \geq 0 \ , \ 0 \leq \alpha_i + \gamma_i \leq C \ , \ \forall i \sum_{i=1}^{n} \alpha_i \cdot y_i = 0$$

Setting $\gamma_i = 0$ clearly maximizes this equation as $R(\alpha, 0) \geq R(\alpha, \gamma)$ and using the Kernel function to replace the dot product of the space transformation functions allows to rewrite the maximization as a minimization

$$\min_{\alpha} \frac{1}{2}\sum_{i=1}^{n}\sum_{i=1}^{n} \alpha_i \cdot y_i \cdot \alpha_j \cdot y_j \cdot K(x_i, x_j) - \sum_{i=1}^{n} \alpha_i + \frac{\varepsilon}{C}\sum_{i=1}^{n} \alpha_i^2$$

subject to

$$0 \leq \alpha_i \leq C \ , \ \sum_{i=1}^{n} \alpha_i \cdot y_i = 0$$

Clearly, by using $\hat{\alpha} = [y_1\alpha_1, y_2\alpha_2, ..., y_n\alpha_n]$, $P = [-y_1, -y_2, ..., -y_n]^T$, and $Q = K + \Lambda$ where Λ is a diagonal matrix such that $\Lambda_{ii} = \frac{2\varepsilon}{C_i}$ and K is the Kernel matrix with $K_{ij} = K(x_i, x_j)$. This reduces the equation to the general form:

$$\min_{\hat{\alpha}} \frac{1}{2}\hat{\alpha}^T Q \hat{\alpha} + P^T \hat{\alpha}$$

subject to $0 \leq \hat{\alpha} \leq C_i \forall y_i = +1$ and $-C_i \leq \hat{\alpha} \leq 0 \forall y_i = -1$ and $\sum_{i=1}^{n} \hat{\alpha}_i = 0$. Equations of this form can be solved by general **Quadradic Programming** techniques as well as **Sequential Minimization Optimization** techniques. For our particular purposes, we stick to the **Least Squares** which has been shown in literature [11] to produce the best results.
The parameters we impose on our cSVM formulation as follows:

1. $(\mathbf{x_i}.\mathbf{y_i})$ The input data we use for our svm is a vector of length n of the last n returns. At a given timestep M, $x_i = [r_{M-n+1}, r_{M-n+2}, ..., r_M]$ and

   $$y_i = \begin{cases} +1 \text{ if } r_{M+1} \geq r_M \\ -1 \text{ if } r_{M+1} < r_M \end{cases}$$

   Our SVM automatically scales and normalizes x_i.

2. $\mathbf{K}(\mathbf{x_i}, \mathbf{x_j})$ We use the **radial basis function kernel** as it best fit our purposes as an SVM stock predictor.

   $$K(x_i, x_j) = \exp(-\frac{\|x_i - x_j\|_2^2}{2\sigma^2})$$

   With experimentation, we settled on $\sigma = 1$ as it gave us the best results. We experimented with **linear**, **quadratic**, **polynomial** and **multilayer perceptron** kernels as well. The quadratic and polynomial kernels were massive time sinks, and the linear and multilayer perceptron lead to suboptimal predictive performance.

3. **C** For our soft margin box constraint C we experimented with many values but a value of $C = 30$ in the end gave us decent results.

4. **Method to find hyperplane** The *Quadratic Programming* method was a huge time sink, and *SMO* turned about to slightly worse performing than *Least Squares*. Therefore, we chose *Least Squares* [14].

5. **Confidence to portfolio distribution** We compute the confidence of a certain classification c with a Kernel as follows:

   $$c = b + \sum_{n \in SV} \alpha_n y_n K(x_n, x)$$

   After experimenting with various models of using confidence as portfolio weights, we realized linear proportions gave us mediocre results. For a series of *confidences* for a set of N assets given by the vector $C = [c_1, c_2, ..., c_N]$ at iteration M, we compute the equivalent portfolio weights as follows:

   $$x_i = \max(4^{\frac{c_i}{\max C}} - 1, 0)$$

   followed by

   $$x_i \leftarrow \frac{x_i}{\sum_{i=1}^{N} x_i}$$

   The intuition behind this function includes disallowing of taking short positions on stocks, and normalizing but exponentially rewarding high confidence, while keeping the summation of the weights of the portfolio at 1.

We left our model in its early stages, and did not implement any regulatory measures on the trading. There were no transaction costs or portfolio adjustment constraints associated with our cSVM simulation.

2.4 Data

For our backtest, we used the same data as we did in the first part of this paper, extending from 1st Jan, 1999 to 1st Jan, 2014 - a period of 15 years.

2.5 Experiment and Results

The results from experimenting with the cSVM were quite interesting. We show 3 prominent experiments and their results.

2.5.1 Highly Optimized cSVM

This experiment uses a cSVM whose parameters were highly optimized for optimal performance. Everything from C to γ to the Kernel function $K(x_i, x_j)$, and all the other parameters mention in 2.3 were optimized for. Because of the large time it takes to train an SVM, we simply used the first 300 weeks of our data for training and the next 500 weeks for classification without further updating of our model.
We merely compare cSVM to the baseline and categorically avoid comparing it to PRAM or Markowitz from the earlier experiments as this experiment is intentionally devoid of transaction costs and it cannot be run for a longer period, because the training period will be sacrificed.
cSVM successfully predicted the direction of movement **63.03**% of the time. Figure 13 shows the performance of cSVM compared to a baseline and Figure 14 shows the change in portfolio allocation over time.

mid 2004 - 2014	cSVM	Baseline
Annualized Return	14.88%	7.68%
Standard Deviation	3.77%	3.06%
Mean Return	0.61%	0.34%
Total Return	3.6077	1.9824

2.5.2 Self-trained cSVM

As a sanity check, we ran a backtest using an SVM trained on the same stocks it was being tested on. We expected great results, and we weren't disappointed. The cSVM accurately predicted direction of stock movement **70**% of the time and returned **3100**% in 14 years. We hypothesize that the 70% could be much improved on and that our formulation of the relation between confidence and portfolio slightly threw off the numbers.

Figure 13: *A comparison of the performance of the highly optimized cSVM with a standard baseline. The red line, cSVM, returned 360% in 9.2 years while the baseline, in black, returned a mere 198%.*

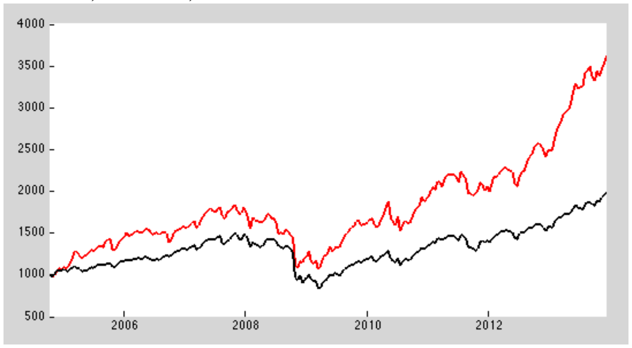


Figure 14: *The portfolio distribution during the cSVM backtest. Unsurprisingly, there is a wide degree of variation, no risk free positions. This is a side effect of having no transaction costs.*

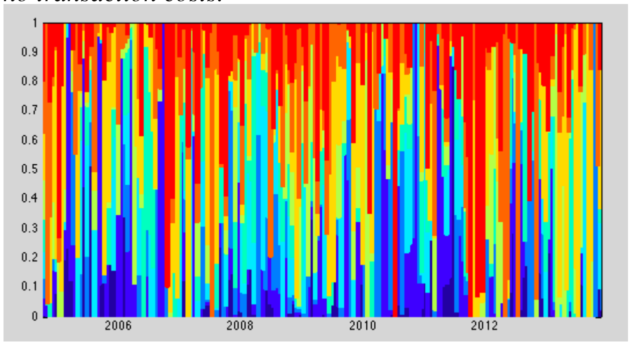


Figure 15: *The typically high returns of a trading model with cSVM trained on itself beforehand. The model produced 3100% returns over 14 years.*

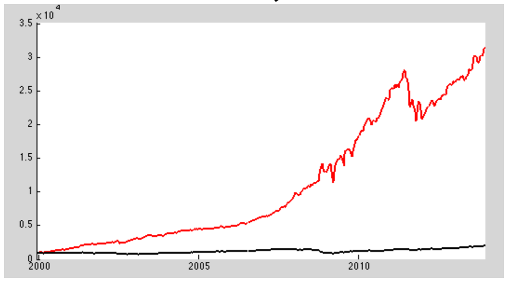
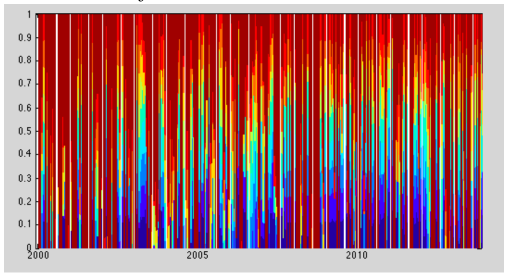


Figure 16: *The portfolio allocation variation of a self-trained cSVM. Note the high variation in portfolio with time due to lack of transaction costs.*



2.5.3 Cross-validation SVM

Cross-validation is often used method in Machine Learning literature [3] used to test the accuracy of an algorithm. To test the accuracy of our SVM, we performed a 9-way cross-validation on the 9 SPDR ETFs. Our mean correct predictions with $n = 36$ samples as input data were $\mu = 0.6481$ and $\sigma = 0.041$. We used scaled sample returns as our input data. The value is statistically significant and much greater than all baselines such as blind prediction of stocks falling or rising, random predictions, and brute momentum predictions by around $5 - 8\%$.

2.6 Conclusion

Some of the takeaways from our cSVM model were:

1. **Speed can be a factor** A real-time updating model was infeasible to backtest on as back test times exceed 10 - 20 minutes. Eventually we had to fall back on a single-time trained model.

2. **The importance of parameters** An SVM is completely made or broken by your choice of parameters. It is crucial to perform techniques like *grid search* [6] and *binary search* to tune the SVM for a particular purpose. Parameter choice also ranges to choice of the Kernel and more.

3. **Using confidence of SVMs for portfolio allocation is hard, but possible** There are many difficulties in using SVMs for portfolio allocation. We had not fine tuned methods to keep the portfolio inertial, and not change drastically over the course of a few weeks. Further, we found it difficult to establish suitable relations between confidence and portfolio allocation. Figuring out how much relative wealth to put in risk free assets was also non-obvious.

4. **Experimentation with alternate Kernels** We used the radial basis function because it has been widely used in literature [7] with SVMs for stock market prediction but we hypothesize a more thorough investigation into the infinite space of Kernels that obey Mercer conditions could perhaps reveal something more appropriate for our purposes.

5. **Experimenting with alternate inputs** For cSVM, we mostly experimented with inputs like past *N* normalized stock prices or returns. It would be interesting to experiment with long-term metrics like *P/E ratio*, *Beta value* and *EPS* or even measures like *VIX* to find more interesting patterns using SVM. There is literature [10] which studies the use of various metrics like this.

2.7 Future Work

Prediction of stock market data seems like a task ideal for Support Vector Machines, but we think the biggest issue is to extract key features as inputs that are more telling than merely the last *N* stock prices. For example, using things like *social media sentiments* or *volume variations* or even metrics like *EPS* as mentioned before would expose new ground for the use of SVMs in stock market predictions.

More recent literature has shown vast success in the use of another popular machine learning technique , *Artificial Neural Networks* with feed-forward propagation as a predictor of stock prices. Machine Learning is most certainly the future of stock predictions. It's the industry standard when it comes to high-frequency trading models run at proprietary trading firms already and has far-reaching potential.

References

[1] Harry Markowitz Bruce Jacobs, Kenneth Levy. Portfolio optimization with factors, scenarios, and realistic short positions. 2005.

[2] Vladimir Vapnik Corinna Cortes. Support vector networks. 1995.

[3] C.C. Taylor D. Michie, D. J. Spiegelhalter. Machine learning, neural and statistical classification. 1994.

[4] William H. Greene. A gamma-distributed stochastic frontier model. 1990.

[5] Sara Emanuelsson Hannes Marling. The markowitz portfolio theory. 2012.

[6] Young-Chan Lee Jae H. Min. Bankruptcy prediction using support vector machine with optimal choice of kernel function parameters. 2005.

[7] Kyoung jae Kim. Financial time series forecasting using support vector machines. 2003.

[8] Barbara Ostdiek Jeff Fleming and Robert E. Whaley. Predicting stock market volatility: A new measure. 2006.

[9] Andrew Gary Joy. A dynamic optimization model incorporating the vix index to predict future returns. 2010.

[10] F.E.H Tay L. J. Cao. Support vector machine with adaptive parameters in financial time series forecasting. 2003.

[11] Phichhang Ou. Prediction of stock market index movement by ten data mining techniques. 2009.

[12] Norbert Trautmann Philipp Baumann. Portfolio-optimization models for small investors. 2012.

[13] S. M. Duarte Queirs. On the emergence of a generalised gamma distribution. application to traded volume in financial markets. 2005.

[14] Ruhaidah Samsudin Shuhaida Ismaila, Ani Shabria. A hybrid model of self-organizing maps (som) and least square support vector machine (lssvm) for time-series forecasting. 2011.

[15] James Tobin. Liquidity preference as behavior towards risk. 1958.