INFO 7374 **Yizhen Zhao**
Fall, 2018

Lecture 2 Machine Learning in Market Making

Algorithm Trading Basics

Market Making

1. **Market makers** are agents who stand ready to buy and sell securities in the financial markets. The rest of the market participants are therefore always guaranteed counterparty for their transactions.
2. Traditional market makers are usually under contractual arrangements with the stock exchange and are incentivized to achieve benchmark quoting requirements. Nowadays, **High Frequency Trading (HFT)** firms play the role of market makers by creating bid-ask spreads, churning mostly low priced, high volume stocks (typical favorites for HFT) many times in a single day.
3. Prior to the Volcker Rule (July 21, 2015), many investment banks had segments dedicated to HFT. Post-Volcker, no commercial banks can have proprietary trading desks or any such hedge fund investments. All major banks have shut down their HFT shops.
4. Nowadays, the HFT world still has players ranging from small firms to medium sized companies and big players. A few names from the industry (in no particular order) are Chopper Trading, Virtu Financial, Jump Trading, Jane Street, etc.
5. HFT firms make money from **two sources**: (1) proprietary trading and (2) getting paid for providing liquidity by Electronic Communications Networks (ECNs) and some exchanges.

Algorithm Trading

1. **Algorithmic trading** is a method of executing a large order (too large to fill all at once) using automated pre-programmed trading instructions accounting for variables such as time, price, and volume to send small slices of the order (child orders) out to the market over time.
2. Any strategy for algorithmic trading requires an identified opportunity that is profitable in terms of improved earnings or cost reduction. The following are **common trading strategies** used in algorithm-trading:

- **Trend-following Strategies:** The most common algorithmic trading strategies follow trends in moving averages, oscillators, price momentum and related technical indicators. These strategies do not involve making any predictions or price forecasts. Trades are initiated based on the occurrence of desirable trends.
- **Relative Value Arbitrage:** Buying a dual-listed stock or bond at a lower price in one market and simultaneously selling it at a higher price in another market offers the price differential

as risk-free profit or arbitrage. The same operation can also be done for stocks vs. futures instruments, as price differentials do exist from time to time.

- **Mean Reversion Strategy:** Mean reversion strategy is based on the idea that the high and low prices of an asset are a temporary phenomenon that revert to their mean value (average value) periodically. Identifying and defining a price range and implementing an algorithm based on that allows trades to be placed automatically when the price of asset breaks in and out of its defined range.
$$Range = Midprice \pm Spread$$
Here, $midprice$ doesn't have to be the statistical mean, but an approximation of the price trend. It can be a simple moving average (SMA), an exponential moving average (EMA) or an OLS estimate. It can also be a latent time-series estimated using $Filtering$ method. $Spread$ is an estimation about the range in which the price will move in the next time period.It is usually a function of market volatility and can incorporate any information that may impact the volatility, such as volume and news.
Note: Leverage Effect refers to that market volatility increases in response to bad news, but falls in response to good news.

- **Volume Weighted Average Price (VWAP):** Volume weighted average price strategy breaks up a large order and releases dynamically determined smaller chunks of the order to the market using stock-specific historical volume profiles. The aim is to execute the order close to the Volume Weighted Average Price (VWAP):
$$VWAP = \frac{\sum Number\ of\ Shares\ Bought\ \times Share\ Price}{Total\ Shares\ Bought}$$
If the price of a buy order is lower than the VWAP, or if the price of a sell order is higher than VWAP, it is a good trade.

- **Time Weighted Average Price (TWAP):** Time weighted average price strategy breaks up a large order and releases dynamically determined smaller chunks of the order to the market using evenly divided time slots between a start and end time. The aim is to execute the order close to the average price between the start and end times, thereby minimizing market impact.

- **Percentage of Volume (POV):** Until the trade order is fully filled, this algorithm continues sending partial orders, according to the defined participation ratio and according to the volume traded in the markets. The related "steps strategy" sends orders at a user-defined percentage of market volumes and increases or decreases this participation rate when the stock price reaches user-defined levels.

- **Statistical Arbitrage:** is a computationally intensive approach to algorithmically trading financial market assets such as equities and commodities. It involves the simultaneous

buying and selling of security portfolios according to predefined or adaptive statistical models. Statistical arbitrage techniques are modern variations of the classic Cointegration-based ***Pairs Trading Strategy:***
Pairs Trading is a market-neutral trading strategy that matches a long position with a short position in a pair of highly correlated instruments such as two stocks, exchange-traded funds (ETFs), currencies, commodities or options.

Parameter Estimation Methods

- Method of Moments (GMM)
- Maximum Likelihood Estimation (MLE)
- Bayesian Estimation

Choice of Quantitative Models

- Reduced Form Models
- Structure Models

Construction of Test Statistics

The Holy Trinity of Tests: Wald, LR and LM.

Suppose that θ is a parameter, $l(\theta)$ is the log-likelihood function and

$$\tilde{\theta} = argmax_{\theta} l(\theta)$$

is the MLE. We want to test $\theta = \theta_0$ against $\theta \neq \theta_0$.

- Wald: compare $\tilde{\theta}$ and θ_0.
- LR: compare $l(\tilde{\theta})$ and $l(\theta_0)$.
- LM: See how close $l'(\theta_0)$ is to zero.

Predictive Accuracy

Diebold Mariano Test

Out-of-sample root mean square prediction error (RMSPE) is a natural metric for the quality of point forecasts. Given two competing forecasts, we can work out their out-of-sample RMSPEs in recursive or rolling-window schemes. Let $\hat{u}_{1t}$ and $\hat{u}_{2t}$ denote the two prediction errors. The idea of the Diebold-Mariano test is to apply a $t - test$ to the series $z_t = u_{1t}^2 - u_{2t}^2$ and see if the mean is zero or not. Concretely, take the test statistic

$$DM\ test\ stat = \frac{\sum_{t=1}^{T}(\hat{u}_{1t}^2 - \hat{u}_{2t}^2)}{\hat{\sigma}/\sqrt{T}}$$

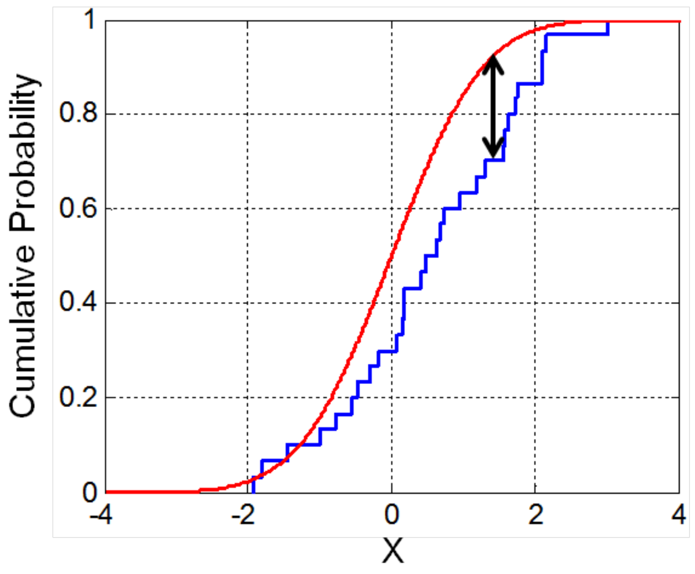
where T is the number of time periods for the out-of-sample forecast comparison and $\hat{\sigma}^2$ is the sample variance of $\hat{u}_{1t}^2 - \hat{u}_{2t}^2$. This is simply a $t - statistic$ testing the hypothesis that

$$E(u_{1t}^2) = E(u_{2t}^2)$$

and it has a standard normal null limiting distribution. This all works well for "non-nested" forecast comparisons, that is where the neither model is nested in the other.

Kolmogorov–Smirnov Test

The Kolmogorov–Smirnov statistic quantifies a distance between the empirical distribution function of the sample and the cumulative distribution function of the reference distribution. The null distribution of this statistic is calculated under the null hypothesis that the sample is drawn from the reference distribution (in the one-sample case). The distributions considered under the null hypothesis are continuous distributions but are otherwise unrestricted.



Note: Illustration of the Kolmogorov–Smirnov statistic. Red line is CDF, blue line is an ECDF (empirical CDF), and the black arrow is the K–S statistic.

Amisano Giacomini Test

The measure of accuracy for a particular density forecast uses ***Predictive Likelihood***, which is the sum of logarithmic predictive densities over the out-of-sample forecasting periods:

$$g(Y_t^*|x,\theta,T) = \sum_{t=1}^{T} \log f(Y_t^*|x,\theta,T)$$

$f(Y_t^*|x,\theta,T)$is the predictive density of Y_t^* implied by model at date t.

To decide whether one density forecast outperforms the other, one can use the difference-in-predictive-likelihood test proposed by Amisano and Giacomini(2007). The test pair wisely compares the predictive accuracy of one model with the other. The advantage ofone density forecast over another is measured by the difference of the logarithmic predictive densities in every period:

$$\Delta L(Y_t^*) = \log[g(Y_t^*|x,\theta,T)] - \log[p(Y_t^*|x,\theta,T)]$$

The test statistic takes the form of a t-statistic:

$$AG - test\ stat = \frac{\Delta\bar{L}(Y_t^*)}{\hat{\sigma}/\sqrt{T-k}}$$

where $\Delta\bar{L}(Y_t^*)$ is the sample average of $\Delta L(Y_t^*)$, and $\hat{\sigma}^2$ is the sample variance of $\Delta L(Y_t^*)$.

Review on Time-Series Econometrics

The simplest time series model is an AR(1) - the *Ornstein–Uhlenbeck process*

$$y_t = \alpha y_{t-1} + u_t$$

In the case $|\alpha| < 1$, we have

$$T^{\frac{1}{2}}(\hat{\alpha} - \alpha) \underset{d}{\rightarrow} N(0, 1-\alpha^2)$$

But this breaks down in the case $\alpha = 1$, which is a random walk. The knife-edge case where α is exactly equal to one arguably isn't that interesting per se. More importantly, this result doesn't work well unless the sample size is enormous if α is close to, but less than 1.

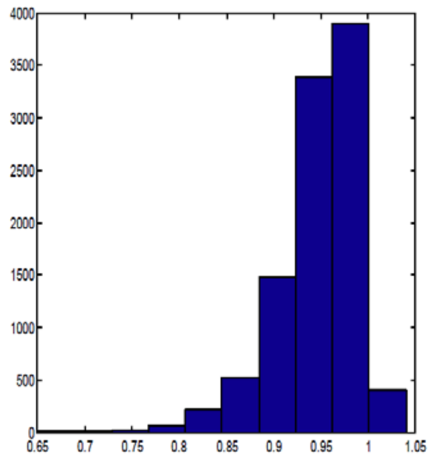
Stationary /Non-stationary Time Series

- A time series is a **random walk** if $y_t = y_{t-1} + u_t$ where u_t is iid.
- A time series is a **martingale** if $E(y_t) = y_{t-1}$.
- A time series is (weakly) **stationary** if its first two moments exist and do not change over time.
- A time series is **invertible** if it can be written as an autoregression.
- A time series is I(0) if it is both stationary and invertible. A time series is **I(d)** if its' d-th difference are I(0).
- If a time series is I(1), it is said to have a **unit root**.
- An **ARIMA(p,d,q)** model is a time series the d-th differences of which form a stationary and invertible ARMA(p,q) model.
- Suppose that x_t and y_t are two unrelated random walks. In a regression of one on the other, the coefficient is likely to be significant and the R-squared is likely to be high. But there is in fact no relation between the series. It is called a **spurious regression**.
- The fact that two time series have unit roots, does not mean that a relationship between them is a spurious regression. It is also possible that they are **cointegrated**.

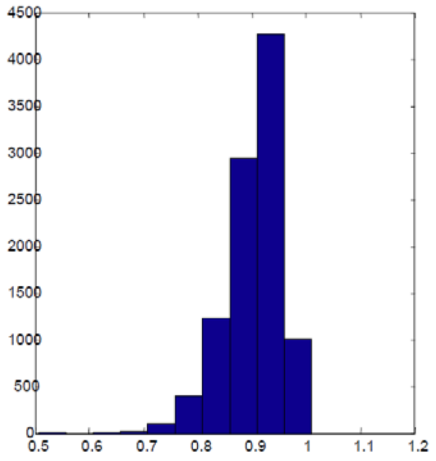
Unit Root

When $\alpha \to 1$, the simulated distribution of OLS estimator of α when $T = 100$.

$\alpha = 1$



$\alpha = 0.95$



Both are skewed to the left.
Here, normal distribution doesn't work and for this and many non-standard problems in econometrics, we need to introduce new tools --- Brownian motion.

Brownian Motion

The stochastic process $B(t)$ is a **Brownian motion** if

1. $B(0) = 0$
2. $B(t) - B(s) \sim N(0, \sigma^2(t-s))$ for any $t > s$
3. If $t_1 < t_2 < t_3 < t_4$ then
   $B(t_2) - B(t_1)$ is independent of $B(t_4) - B(t_3)$

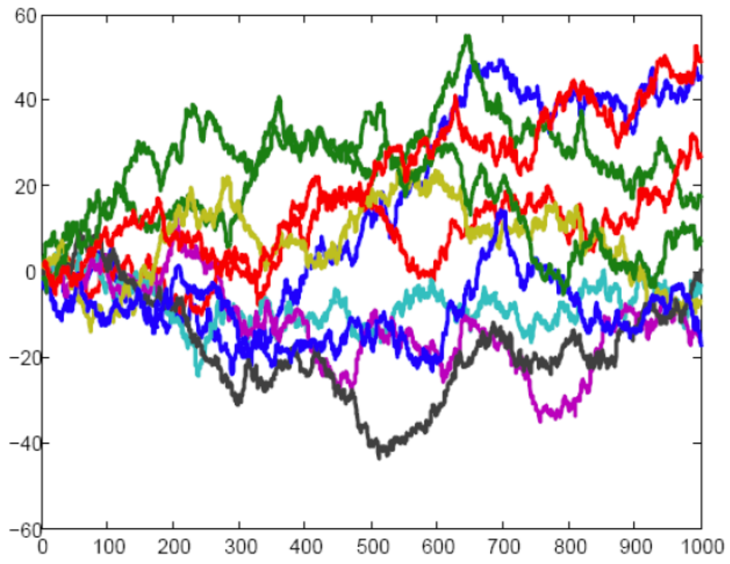
Functional Central Limit Theorem with Non-i.i.d. Errors

Suppose that $\varepsilon_1, \varepsilon_2, \cdots \varepsilon_T$ are stationary with mean 0 and (average) zero-frequency spectral density ω^2 satisfying suitable conditions. Let $S_t = \sum_{s=1}^{t} \varepsilon_s$. Define the function

$$S_T(r) = \frac{1}{T^{\frac{1}{2}}\omega} S_{Tr}$$

Then $S_T(r) => B(r)$.

Example: Ten Brownian Motions



Each time-series you see is actually one of the many possible realizations.

Continuous Mapping Theorem

Suppose that $X_T \underset{d}{\rightarrow} X$ (uniformly in r, if applicable). Then $f(X_T) \underset{d}{\rightarrow} f(X)$ where $f(\cdot)$ is any continuous function.

Combining these pieces,

$$T(\hat{\alpha} - 1) \underset{d}{\rightarrow} \frac{\frac{1}{2}\{B(1)^2 - 1\}}{\int_0^1 B(r)^2 dr}$$

Unit Root Tests

1. Augmented Dicky-Fuller Test (1979)
Null: Unit Root, $\sqrt{\mathrm{T}}(\hat{\rho}_{\mathrm{T}} - 1) \overset{\mathrm{p}}{\rightarrow} 0$

$$y_t = y_{t-1} + \varepsilon_t$$

Alternatives:
(i) No Drift

$$y_t = \rho y_{t-1} + \varepsilon_t$$

(ii) With Drift

$$y_t = c + \rho y_{t-1} + \varepsilon_t$$

(iii) With Drift and Time Trend

$$y_t = c + \delta t + \rho y_{t-1} + \varepsilon_t$$

Notes: Failure to reject the null (insignificant test stat or to say, large p -value) indicates that the time series y_{t} is a unit root process.

2. Philips-Perron Test (1988)
Null: Unit Root, $\sqrt{\mathrm{T}}(\hat{\rho}_{\mathrm{T}} - 1) \overset{\mathrm{p}}{\rightarrow} 0$

$$y_t = y_{t-1} + \varepsilon_t$$

Alternatives:
(i) No Drift

$$y_t = \rho y_{t-1} + \varepsilon_t$$

(ii) With Drift

$$y_t = c + \rho y_{t-1} + \varepsilon_t$$

(iii) With Drift and Time Trend

$$y_t = c + \delta t + \rho y_{t-1} + \varepsilon_t$$

Notes: Failure to reject the null (insignificant test stat or to say, large p -value) indicates that the time series y_{t} is a unit root process.

3. Kwiatkowski, Philips, Schmidt, and Shin (KPSS) Test (1992)
Null: Trend Stationary

$$y_t = c + \delta t + u_{1t}$$

Alternatives:
(i) no deterministic trend: $y_t = c_t + u_{1t}$ and $c_t = c_{t-1} + u_{2t}$
(ii) with trend: $y_t = c_t + \delta t + u_{1t}$ and $c_t = c_{t-1} + u_{2t}$
where u_{1t} is a stationary process. u_{2t} is an independent and identically distributed process with mean 0 and variance σ^2.
The null says $\sigma^2 = 0$, which implies the random walk term c_t is constant. The alternative says $\sigma^2 > 0$, which indicates that the above unit root is a random walk. Rejection of the null indicates y_t is a non-stationary process.

Cointegration

Definition: Two non-stationary time series x_t and y_t are said to be cointegrated, if they are both I(1) but if there exists some linear combination $u_t = y_t - \beta x_t$, for $0 < k < \infty$, that is I(0).

We can rewrite the definition of cointegration as $y_t = \beta x_t + u_t$ where the regressor is I(1) and the error term is I(0). This model has intriguing statistical properties:

- OLS is super-consistent (meaning $T(\hat{\beta} - \beta)$ converges to a distribution, that is a function of Brownian motions).
- If x_t is strictly exogeneous (independent of the error at all leads and lags), then t- and F-statistics associated with OLS have their usual normal and χ^2 limiting distributions.
- If x_t is not strictly exogeneous, there are estimators other than OLS such that t- and F-statistics have normal and χ^2 limiting distributions. A popular choice is dynamic OLS which estimates the relationship
$$y_t = \beta x_t + d(L)\Delta x_t + u_t$$
where $d(L)$ is a two-sided polynimial. Another choice is the maximum likelihood estimator proposed by Soren Johansen.

Paris Trading

- Pair Selection：Use β from CAPM
- Introduce CAPM
$$R^e_{i,t} = \alpha + \beta \cdot R^e_{M,t} + u_{i,t}$$
- Benchmark Pair：Exxon Mobil Corporation (XOM) and Chevron Corporation (CVX)
- Test two things：(1) the difference between two β from CAPMis at most 0.15.

(2) whether the two stocks are cointegrated.

- Reason to choose β: If the ratio β_i/β_j for two separate securities, i and j is close to one, then we expect them to be affected by market movements in the same fashion, a condition that favors pairs-trading compatibility.
- Test Cointegration:
  1. Engle-Granger two-step cointegration test.
  2. Cointegration vector:
     (a) Trace Test (see Johansen (1988))
     (b) Maximum Eigen Value Test (see Johansen and Juselius' (1990)).

Procedure

- Regression: $\log(P_1) = \beta_{coint} \log(P_2)$, β_{coint} is the cointegration ratio. We constrain the intercept to 0 since if pair is cointegrated, then we expect 0 returns on one asset to predict 0 returns on the other.
- Spread $S_t = \log(P_1) - \beta_{coint} \log(P_2)$.
- Test spread of pair for stationarity using an Augmented-Dickey Fuller (ADF) Test, which tests the null hypothesis that a process has a unit root (is not stationary). If the pair is cointegrated, then the spread should be stationary.

Strategy Design

Allocation Ratio	Calculation of Spread	Strategy Design
1:1 Dollar	$\log(P_1) - \log(P_2)$.	long one share of P_2, short one share of P_1
CAPM β	$\log(P_1) - \beta_1/\beta_2 \log(P_2)$.	long one share of P_2, short β_1/β_2 share of P_1
Cointegration β	$\log(P_1) - \beta_{coint} \log(P_2)$	long one share of P_2, short β_{coint} shares of P_1

1. For each time point in the time series, calculate the risk-adjusted spread between the two assets of the pair.
2. If the spread is above its historical mean, then we expect that stock 1 is overpriced and stock 2 is underpriced. Thus, we short-sell stock 1 and buy stock 2. On the other hand, if the spread is under its historical mean, we short-sell stock 2 and buy stock 1.
3. If the signal is less than the closing threshold, close any existing position in the pair.
4. If the signal is greater than the stop-loss threshold, we close the position.

Note: the open threshold is set to 1σ, the close threshold is set to 0.5σ, and the stop-loss threshold to 4σ.

Conditional Heteroskedasticity
It is very common for financial time series to exhibit bursts of volatility. Modeling this is important for forecasting and many other purposes. The original model was autoregressive conditional heteroskedasticity (ARCH) which specifies that

$$r_t = \mu + \sigma_t \varepsilon_t$$
$$\sigma_t^2 = \omega + \alpha(r_{t-1} - \mu)^2$$
$$\sigma_1^2 = \omega/(1-\alpha)$$

where ε_t is iid standard normal.

- The kurtosis of above process is
$$\frac{3(1-\alpha^2)}{(1-3\alpha^2)} > 3$$
The above model not only allows for burst of volatility, but it also accounts for fat tails.

- The estimation is fairly easy by maximum likelihood as the log-likelihood function is
$$-\frac{1}{2}\sum_{t=1}^{T}\log(\sigma_t^2) - \frac{1}{2}\sum_{t=1}^{T}\frac{r_t - \mu}{\sigma_t^2}$$

and can be numerically maximized with respect to the parameters α, μ and ω.

- The model has been extended in a great many ways. Three in particular are:
  1. GARCH: Generalized ARCH. A GARCH(p,q) model is
$$r_t = \mu + \sigma_t \varepsilon_t$$
$$\sigma_t^2 = \omega + \sum_{i=1}^{q}\alpha_i(r_{t-i} - \mu)^2 + \sum_{i=1}^{p}\beta_i\sigma_{t-i}^2$$
  2. GARCH in mean
$$r_t = \mu + \lambda\sigma_t + \sigma_t \varepsilon_t$$
$$\sigma_t^2 = \omega + \sum_{i=1}^{q}\alpha_i(r_{t-i} - \mu)^2 + \sum_{i=1}^{p}\beta_i\sigma_{t-i}^2$$
  3. Exponential GARCH
$$r_t = \mu + \sigma_t \varepsilon_t$$
$$log\sigma^2 = \omega + \alpha\log\sigma_{t-1}^2 + \beta[\theta\varepsilon_t + (|\varepsilon_t| - E(|\varepsilon_t|))]$$
  Since ε_t is standard normal, $E(|\varepsilon_t|) = \sqrt{2/\pi}$. This model is particularly useful for representing stock returns, because they not only show burst of volatility, but volatility tends to rise when returns are low. This model can capture a skewness effect of this sort.

- All above models can be estimated by maximum likelihood (MLE). Any number of extensions to this framework have been proposed. One can add in explanatory variables or have nonlinear or multivariate specifications. A general challenge is ensuring that the variances remain positive. Of course, any parameterization which gives negative variances will in turn have a likelihood of minus infinity.

Filtering Methods

The filtering problem is that there is unobserved variable (a "state" variable) that evolves by some law of motion and there are observed variables that are related to the unobserved state. It might sound an arcane problem, but as we have seen, it has an enormous number of important applications.

The basic filtering problem is a linear model in what is known as state space form. This is the model where we observe y_t while α_t is an unobserved state and

$$y_t = Z_t \alpha_t + \varepsilon_t$$
$$\alpha_t = T\alpha_{t-1} + \eta_t$$

where $\varepsilon_t \sim i.i.d.N(0,H)$ and $\eta_t \sim i.i.d.N(0,Q)$.

The Kalman Filter

The Kalman Filter allows inference to be done in the basic state space model. In this model, $\alpha_t|Y_s$ is normal; let $\alpha_{t|s}$ and $P_{t|s}$ denote its mean and variance. We then have

Updating Equations

$$\alpha_{t|t} = \alpha_{t|t-1} + P_{t|t-1} Z_t' F_t^{-1}(y_t - Z_t \alpha_{t|t-1})$$

$$P_{t|t} = P_{t|t-1} - P_{t|t-1} Z_t' F_t^{-1} Z_t P_{t|t-1})$$

where

$$F_t = Z_t P_{t|t-1} Z_t' + H$$

Prediction Equations

$$\alpha_{t+1|t} = T\alpha_{t|t}$$

$$P_{t+1|t} = TP_{t|t}T' + Q$$

if α_t is stationary, can initialize from the unconditional mean and variance-covariance matrix of α_t. The mean is just zero. The variance is $P_{t|t}$ which is the solution to the equation

$$P_{0|0} = TP_{0|0}T' + Q$$

the solution to which is

$$vec(P_{0|0}) = (I - T \otimes T)^{-1} vec(Q)$$

So, in Python, $P_{0|0}$is simply

$$reshape(inv(eye(n^2) - kron(T, T)) * reshape(Q, n^2, 1), n, n)$$

where n is the number of elements in the state vector α_t.

Then we iterate through the updating the predictive equations to get $\alpha_{t|t-1}$ and α_t. The Kalman filter has two potential purposes:
(i) estimation and
(ii) inference about the state vector.
For the first of these, we have log-likelihood:

$$l = \sum_{t=1}^{T} \log(f(y_t|Y_{t-1}))$$

$$y_t|Y_{t-1} \sim N(Z_t\alpha_{t|t-1}, Z_t P_{t|t-1} Z_t' + H) = N(Z_t\alpha_{t|t-1}, F_t)$$

$$l = -\frac{T}{2}\log(2\pi) - \frac{1}{2}\sum_{t=1}^{T} \log|F_t| - \frac{1}{2}\sum_{t=1}^{T} v_t' F_t^{-1} v_t$$

where $v_t = y_t - Z_t\alpha_{t|t-1}$.
For inference about the sate vector, we already have $\alpha_{t|t}$, the filtered estimates. But we might want $\alpha_{t|T}$, the "smoothed" estimates. These are obtained with one more set of recursions known as the Kalman smoother:

$$\alpha_{t|T} = \alpha_{t|t} + P_{t|t} T' P_{t+1|t}^{-1} (\alpha_{t+1|T} - T\alpha_{t|t})$$

$$P_{t|T} = P_{t|t} + P_{t|t} T'^{P_{t+1|t}^{-1}} (P_{t+1|T} - P_{t+1|t}) P_{t+1|t}^{-1} T P_{t|t})$$

The Kalman filter/smoother need values of the parameters. We can use numerical methods to find the parameter values that maximize the likelihood (MLE), and then plug these in to get filtered and smoothed estimates of the states, which are also of interest.

Example: Use Kalman Filter for Pairs Trading
The pairs-trading strategy could be applied to a two ETFs that both track the performance of varying duration US Treasury bonds. They are:

- **TLT** - iShares 20+ Year Treasury Bond ETF
- **IEI** - iShares 3-7 Year Treasury Bond ETF

The goal is to build a mean-reverting strategy from this pair of ETFs.

The synthetic "spread" between TLT and IEI is the time series that we are actually interested in longing or shorting. The Kalman Filter is used to dynamically track the hedging ratio between the two in order to keep the spread stationary (and hence mean reverting).

Trading Rule: we go "long the spread" if the forecast error drops below the negative one standard deviation of the spread. Respectively we can go "short the spread" if the forecast error exceeds one positive standard deviation of the spread. The exit rules are simply the opposite of the entry rules.

The dynamic hedge ratio is represented by one component of the hidden state vector at time t, which we will denote as θ_t. This is the "beta" slope value that is from linear regression:

$$R_{i,t}^{e} = \alpha + \beta \cdot R_{M,t}^{e} + u_{i,t}$$

Longing the spread here means purchasing (longing) N units of TLT and selling (shorting) $\theta_t^0 N$, which must take the "floor" value integer. The latter is necessary as we must transact a whole number of units of the ETFs. "Shorting the spread" is the opposite of this. N controls the overall size of the position.

e_t represents the *forecast error* or *residual error* of the prediction at time t, while Q_t represents the variance of this prediction at time t.

For completeness, the rules are specified here:

1. $e_t < -\sqrt{Q_t}$ - Long the spread: Go long N shares of TLT and go short $\theta_t^0 N$ units of IEI
2. $e_t \geq -\sqrt{Q_t}$ - Exit long: Close all long positions of TLT and IEI
3. $e_t > \sqrt{Q_t}$ - Short the spread: Go short N shares of TLT and go long $\theta_t^0 N$ units of IEI
4. $e_t \leq \sqrt{Q_t}$ - Exit short: Close all short positions of TLT and IEI

The role of the Kalman filter is to help us calculate θ_t, as well e_t and Q_t. θ_t represents the vector of the intercept and slope values in the linear regression between TLT and IEI at time t. It is estimated by the Kalman filter. The forecast error/residual $e_t = y_t - \hat{y}_t$ is the difference between the predicted value of TLT *today* and the Kalman filter's estimate of TLT *today*. Q_t is the variance of the predictions and hence $\sqrt{Q_t}$ is the standard deviation of the prediction.

Hamilton Switching Model

This is an important nonlinear filtering model

$$y_t = \alpha + \beta S_t + \varepsilon_t$$

where S_t is a Markov switching process.

$$P(S_t = 1 | S_{t-1} = 1) = p$$
$$P(S_t = 0 | S_{t-1} = 0) = q$$

$$f(y_t|Y_{t-1}) = N(\alpha,\sigma^2)P(S_t = 0|Y_{t-1}) + N(\alpha+\beta,\sigma^2)P(S_t = 1|Y_{t-1})$$
$$= \frac{1}{(2\pi\sigma^2)^{\frac{1}{2}}}\exp\left(-\frac{(y_t-\alpha)^2}{2\sigma^2}\right)P(S_t = 0|Y_{t-1})$$
$$+ \frac{1}{(2\pi\sigma^2)^{\frac{1}{2}}}\exp\left(-\frac{(y_t-\alpha-\beta)^2}{2\sigma^2}\right)P(S_t = 1|Y_{t-1})$$
a mixture of normals.

<u>Updating equations (from Bayes Theorem)</u>

$$P(S_t = 0|Y_t) = \frac{1}{(2\pi\sigma^2)^{\frac{1}{2}}}\exp\left(-\frac{(y_t-\alpha)^2}{2\sigma^2}\right)P(S_t = 0|Y_{t-1})/f(y_t|Y_{t-1})$$

$$P(S_t = 1|Y_t) = \frac{1}{(2\pi\sigma^2)^{\frac{1}{2}}}\exp\left(-\frac{(y_t-\alpha-\beta)^2}{2\sigma^2}\right)P(S_t = 1|Y_{t-1})/f(y_t|Y_{t-1})$$

<u>Prediction Equations</u>

$$P(S_t = 1|Y_{t-1}) = pP(S_{t-1} = 1|Y_{t-1}) + (1-q)P(S_t = 0|Y_{t-1})$$
$$P(S_t = 0|Y_{t-1}) = (1-p)P(S_{t-1} = 1|Y_{t-1}) + qP(S_{t-1} = 0|Y_{t-1})$$

<u>Starting iterations</u>

Only need $P(S_1 = 1|Y_0) = P(S_1 = 1)$, the unconditional probability. We know that
$$P(S_1 = 1) = \frac{1-p}{2-p-q}$$
and this allows us to start the recursions.

The model is useful for fitting business cycles (Hamilton (1989)). A few last comments:
(i) In practice, we would replace ε_t by an AR process.
(ii) Also, it is possible to let p and q depend on variables at time $t-1$.
(iii) There is also a smoother that can be run backwards to get state probabilities conditional on the whole sample. This uses the recursions
$$P(S_t = 1|Y_T) = \frac{P(S_{t+1} = 0|Y_T)P(S_t = 1|Y_t)P(S_{t+1} = 0|S_t = 1)}{P(S_{t+1} = 0|Y_t\ )}$$
$$+ \frac{P(S_{t+1} = 1|Y_T)P(S_t = 1|Y_t)P(S_{t+1} = 1|S_t = 1)}{P(S_{t+1} = 1|Y_t\ )}$$
for $t = T-1, T-2, \ldots$ starting from $P(S_T = 1|Y_T)$.

Particle Filtering

Particle Filtering is a numeric device that is useful for filtering in a very general context (nonlinear and non-Gaussian). But it is based on simulation. There are many versions of the particle filter. Here are the steps for a simple illustrative implementation.

1. Generate n draws from the steady state distribution of α_0.
2. For each of the draws in (1), use the transition equation to get draws from the distribution of α_1. Call these draws $\{\tilde{\alpha}_{1,i}\}^{n}_{i=1}$. This gives the density of α_1 conditional on Y_0.
3. For each of the draws in (2), compute $q_1^i = p(y_1|\tilde{\alpha}_{1,i})$. Normalize these probabilities so that $\sum_{i=1}^{n} q_1^i = 1$.
4. Resample with replacement from $\{\tilde{\alpha}_{1,i}\}^{n}_{i=1}$ with probability q_1^i of selecting $\tilde{\alpha}_{1,i}$. Call these new draws $\{\tilde{\alpha}_{1,i}\}^{n}_{i=1}$. This gives the density of α_1 conditional on Y_1.
5. Repeat step 2-4 cycling through the whole sample.
   The density of y_t conditional on Y_{t-1} can be approximated by
   $$p(y_t|Y_{t-1}) = \frac{1}{n}\sum_{i=1}^{n} p(y_t|\tilde{\alpha}_{1,i})$$
   and the likelihood is $L = \prod_{t=1}^{T} p(y_t|Y_{t-1})$.

Particle Filter could be estimated using simulated Maximum Likelihood method or Markov Chain Monte Carlo (MCMC).

Both Hamilton Switching Model and the MCMC for Particle Filtering can be estimated by MLE.

Maximum Likelihood Estimation

Say $X_1, X_2, \cdots, X_n$ is $i.i.d.$ from a density $f(x,\theta)$ where θ is a $k \times 1$ vector of parameters. The joint probability density of the data is $\prod_{i=1}^{n} f(X_i,\theta)$.

Idea of maximum likelihood estimation. View this as a function of θ called the likelihood function:

$$L(\theta) = \prod_{i=1}^{n} f(X_i,\theta)$$

The MLE is the value of θ that maximizes the likelihood function:

$$\hat{\theta}_{MLE} = argmax_{\theta} L(\theta)$$

Because it is easier to work with sums than products, we generally write the MLE as

$$\hat{\theta}_{MLE} = argmax_{\theta} L(\theta)$$

where $l(\theta) = \log L(\theta) = \prod_{i=1}^{n} \log f(X_i,\theta)$

Random Forest

Ensemble Methods

Decision Trees (DTs) are a supervised learning technique that predicts values of *responses* by learning decision rules derived from *features*. They can be used in both a **regression** and a **classification** context. For this reason they are sometimes also referred to as Classification and Regression Trees (CART).

Their main disadvantage lies in the fact that they are often uncompetitive with other supervised techniques such as support vector machines or deep neural networks in terms of prediction accuracy.

However they can become extremely competitive when used in an *ensemble* method such as with bootstrap aggregation (**"bagging"**), **Random Forests** or **boosting**.

In quantitative finance ensembles of DT/CART models are used in forecasting, either future asset prices/directions or liquidity of certain instruments.

Bootstrap

Bootstrapping is a statistical resampling technique that involves random sampling of a dataset *with replacement*. It is often used as a means of quantifying the uncertainty associated with a machine learning model. The idea is to repeatedly sample data with replacement from the original training set in order to produce multiple separate training sets. These are then used to allow "meta-learner" or "ensemble" methods to *reduce the variance of their predictions*, thus greatly improving their predictive performance.

Two of the following ensemble techniques–bagging and random forests–make heavy use of bootstrapping techniques, and they will now be discussed.

Boosting

Another general machine learning ensemble method is known as *boosting*. Boosting differs somewhat from bagging as it does not involve bootstrap sampling. Instead models are generated sequentially and iteratively, meaning that it is necessary to have information about model i before iteration i + 1 is produced.

The idea is to *iteratively learn* weak machine learning models on a continually-updated training data set and then add them together to produce a final, strong learning model. This differs from bagging, which simply *averages the models* on separate bootstrapped samples.

Random Forest

Step 1 Bootstrap: split the sample $t = 1, \cdots, T$ into the first K observations (Training) and the rest $T - K$ observations (Test). Sample with replacement from the first K observations to create B sub-samples

Step 2 Cross Validation: for each sub sample, apply all prediction methods in your consideration; use the rest $T - K$ observations to calculate the root-mean squared errors (RMSE) for each model.

Step 3 Aggregation: For each sub-sample, keep only the model with the highest precision (smallest root-mean squared errors). For the prediction made by each of these models $\hat{y}$, take the average to form your prediction $\tilde{y}$.

Step 4 Prediction: compare your prediction with the actual open price. If $\tilde{y} > y_t$, your model predicts the asset price will increase, you take a long position. If $\tilde{y} < y_t$, your model predicts the asset price will decrease, you may create a short position.

A sample program of using Python `sklearn.ensemble` is enclosed as follows:

```
import sklearn
from sklearn.ensemble import (RandomForestRegressor)
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split


# Use the training-testing split with 70% of data in the
# training data with the remaining 30% of data in the testing
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=random_state)

# Pre-create the arrays which will contain the MSE for
# each particular ensemble method
estimators = np.zeros(axis_step)
rf_mse = np.zeros(axis_step)

# Estimate the Random Forest MSE over the full number
# of estimators, across a step size ("step_factor")
for i in range(0, axis_step):
    print("Random Forest Estimator: %d of %d..." % (
        step_factor*(i+1), n_estimators)
    )
    rf = RandomForestRegressor(
        n_estimators=step_factor*(i+1),
        n_jobs=n_jobs,
        random_state=random_state
    )
    rf.fit(X_train, y_train)
    mse = mean_squared_error(y_test, rf.predict(X_test))
    estimators[i] = step_factor*(i+1)
    rf_mse[i] = mse
```

Support Vector Machine

Support vector machine (SVM) analysis is one of the best "out of the box" supervised classification techniques. It is an important tool for both the quantitative trading researcher and data scientist. It was first identified by Vladimir Vapnik and his colleagues in 1992. SVM regression is considered a nonparametric technique because it relies on kernel functions.

Support Vector Machine is mostly used to carry out natural language document classification, for the purposes of sentiment analysis and, ultimately, automated trade filter or signal generation. This lecture will make use of Support Vector Machines (SVM) to classify text documents into mutually exclusive groups.

Linear SVM Regression: Primal Formula

Suppose we have a set of training data where x_i is a multivariate set of N observations with observed response values y_i
To find the linear function

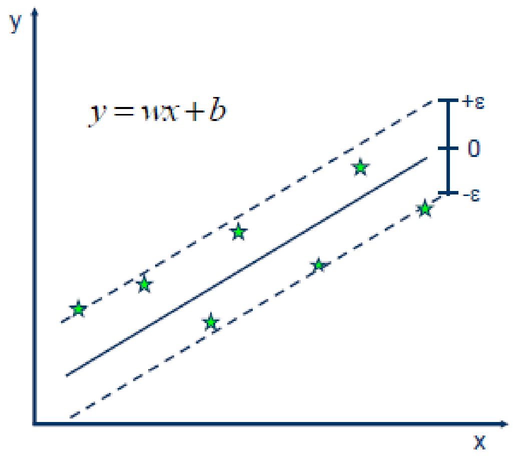
$$f(x) = x'w + b$$

and ensure that it is as flat as possible, find *f*(*x*) with the minimal norm value (*w′w*). This is formulated as a convex optimization problem to minimize

$$J(w) = \frac{1}{2}w'w$$

subject to all residuals having a value less than ε; or, in equation form:

$$\forall i: |y_i - (wx_i + b)| \leq \varepsilon.$$



- Solution:

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

- Constraints:

$$y_i - wx_i - b \leq \varepsilon$$
$$wx_i + b - y_i \leq \varepsilon$$

It is possible that no such function *f*(*x*) exists to satisfy these constraints for all points. To deal with otherwise infeasible constraints, introduce slack variables ξ_i and ξ_i^* for each point. This approach is similar to the “soft margin” concept in SVM classification, because the slack variables allow regression errors to exist up to the value of ξ_i and ξ_i^*, yet still satisfy the required conditions.

Including slack variables leads to the objective function, also known as the primal formula

$$J(w) = \frac{1}{2}w'w + C\sum_{i=1}^{N}(\xi_i + \xi_i^*)$$

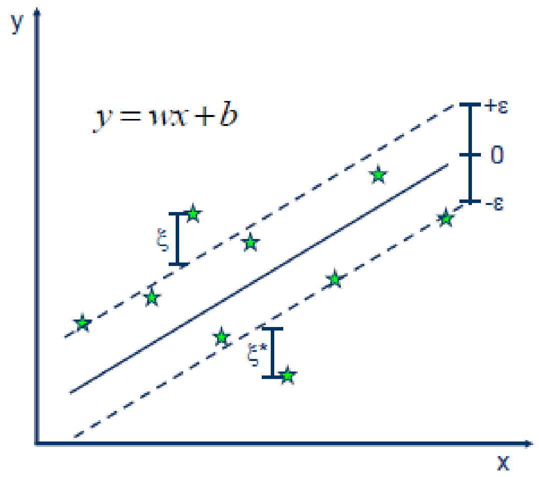
subject to

$$\forall i: y_i - (x_i'w + b) \le \varepsilon + \xi_i.$$

$$\forall i: (x_i'w + b) - y_i \le \varepsilon + \xi_i^*.$$

$$\forall i: \xi_i \ge 0$$

$$\forall i: \xi_i^* \ge 0$$



• Minimize:

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

• Constraints:

$$y_i - wx_i - b \le \varepsilon + \xi_i$$

$$wx_i + b - y_i \le \varepsilon + \xi_i^*$$

$$\xi_i, \xi_i^* \ge 0$$

The constant C is the box constraint, a positive numeric value that controls the penalty imposed on observations that lie outside the epsilon margin (ε) and helps to prevent overfitting (regularization). This value determines the trade-off between the flatness of *f*(*x*) and the amount up to which deviations larger than ε are tolerated.

- The smaller the value of C, the more sensitive the algorithm is to the training data (higher variance and lower bias).
- The larger the value of C, the less sensitive the algorithm is to the training data (lower variance and higher bias).

The linear ε-insensitive loss function ignores errors that are within ε distance of the observed value by treating them as equal to zero. The loss is measured based on the distance between observed value y and the ε boundary. This is formally described by

$$L_\varepsilon = \begin{cases} 0 & if\ |y - f(x)| \le \varepsilon \\ |y - f(x)| - \varepsilon & otherwise \end{cases}$$

Linear SVM Regression: Dual Formula

The optimization problem previously described is computationally simpler to solve in its Lagrange dual formulation. The solution to the dual problem provides a lower bound to the solution of the primal (minimization) problem. The optimal values of the primal and dual problems need not be equal, and the difference is called the “duality gap.” But when the problem is convex and satisfies a constraint qualification condition, the value of the optimal solution to the primal problem is given by the solution of the dual problem.

To obtain the dual formula, construct a Lagrangian function from the primal function by introducing nonnegative multipliersa_i anda_i^* for each observation x_i . This leads to the dual formula, where we minimize

$$L(a) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}(a_i - a_i^*)(a_j - a_j^*)x_i{}'x_j + \varepsilon\sum_{i=1}^{N}(a_i + a_i^*) + \sum_{i=1}^{N} y_i(a_i^* - a_i)$$

subject to the constraints

$$\sum_{i=1}^{N} a_i - a_i^* = 0$$

$$\forall i: 0 \le a_i \le C$$

$$\forall i: 0 \le a_i^* \le C$$

The $w$parameter can be completely described as a linear combination of the training observations using the equation

$$w = \sum_{i=1}^{N}(a_i - a_i^*)x_i$$

The function used to predict new values depends only on the support vectors:

$$f(x) = \sum_{n=1}^{N} (a_i - a_i^*)(x_i' x) + b.$$

The Karush-Kuhn-Tucker (KKT) complementarity conditions are optimization constraints required to obtain optimal solutions. For linear SVM regression, these conditions are

$$\forall i: a_i(\varepsilon + \xi_i - y_i + x_i' w + b) = 0$$

$$\forall i: a_i(\varepsilon + \xi_i + y_i - x_i' w - b) = 0$$

$$\forall i: \xi_i(C - a_i) = 0$$

$$\forall i: \xi_i^*(C - a_i^*) = 0$$

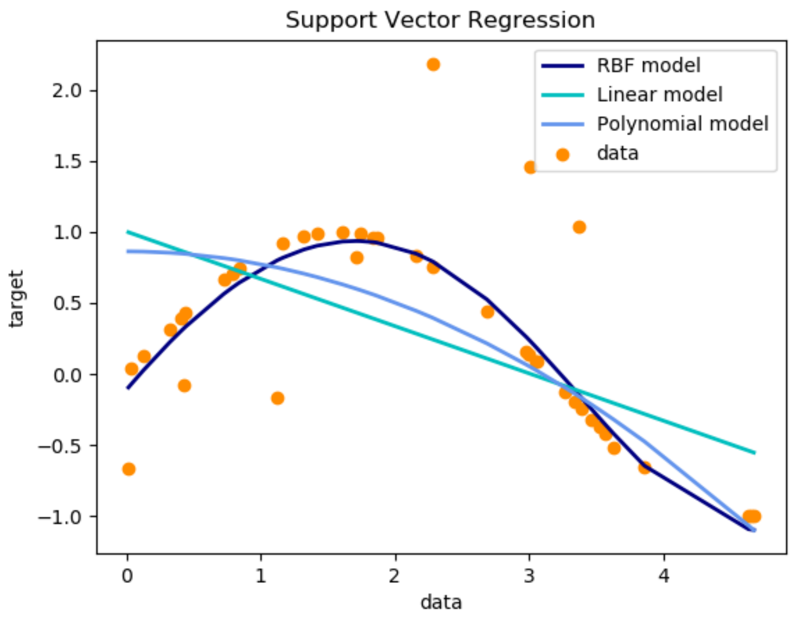
These conditions indicate that all observations strictly inside the epsilon tube have Lagrange multipliers$a_i = 0$ and$a_i^* = 0$. If either a_i or $a_i^*$is not zero, then the corresponding observation is called a ***support vector***.

Nonlinear SVM Regression: Primal Formula

Some regression problems cannot adequately be described using a linear model. In such a case, the Lagrange dual formulation allows the previously-described technique to be extended to nonlinear functions.

Obtain a nonlinear SVM regression model by replacing the dot product $x_1'x_2$ with a nonlinear kernel function $G(x_1,x_2) = <\varphi(x_1),\varphi(x_2)>$, where $\varphi(x)$ is a transformation that maps x to a high-dimensional space.

- Linear: $G(x_j, x_k) = x_j' x_k$
- Polynomial: $G(x_j, x_k) = (1 + x_j' x_k)^q$, where $q = 1, 2, 3, \cdots$
- Gaussian: $G(x_j, x_k) = \exp(-||x_j - x_k||^2)$
- Radial Basis Function (RBF): $G(x_j, x_k) = \exp(-\frac{1}{2\sigma^2}||x_j - x_k||^2)$



Toy example of 1D regression using linear, polynomial and RBF kernels.

<u>Data Preparation for SVM</u>

SVM usually requires you to normalize the data inputs.

- *Numerical Inputs:* SVM assumes that your inputs are numeric. If you have categorical inputs you may need to covert them to binary dummy variables (one variable for each category).
- *Binary Classification:* Basic SVM as described in this post is intended for binary (two-class) classification problems. Although, extensions have been developed for regression and multi-class classification.

<u>Nonlinear SVM Regression: Dual Formula</u>

The dual formula for nonlinear SVM regression replaces the inner product of the predictors $(x_j' x_k)$with the corresponding element of the Gram matrix $G(x_j, x_k)$.

Nonlinear SVM regression finds the coefficients that minimize

$$L(a) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}(a_i - a_i^*)(a_j - a_j^*)G(x_i, x_j) + \varepsilon\sum_{i=1}^{N}(a_i + a_i^*) + \sum_{i=1}^{N} y_i(a_i^* - a_i)$$

subject to

$$\sum_{i=1}^{N} a_i - a_i^* = 0$$

$$\forall i: 0 \leq a_i \leq C$$

$$\forall i: 0 \leq a_i^* \leq C$$

The function used to predict new values is equal to

$$f(x) = \sum_{n=1}^{N} (a_i - a_i^*) G(x_i, x) + b.$$

The Karush-Kuhn-Tucker (KKT) complementarity conditions are

$$\forall i: a_i(\varepsilon + \xi_i - y_i + f(x_i)) = 0$$

$$\forall i: a_i(\varepsilon + \xi_i + y_i - f(x_i)) = 0$$

$$\forall i: \xi_i(C - a_i) = 0$$

$$\forall i: \xi_i^*(C - a_i^*) = 0$$

Solving the SVM Regression Optimization Problem
The minimization problem can be expressed in standard quadratic programming form and solved using common quadratic programming techniques. However, it can be computationally expensive to use quadratic programming algorithms, especially since the Gram matrix may be too large to be stored in memory. Using a decomposition method instead can speed up the computation and avoid running out of memory.

Decomposition methods (also called *chunking and working set methods*) separate all observations into two disjoint sets: the working set and the remaining set. A decomposition method modifies only the elements in the working set in each iteration. Therefore, only some columns of the Gram matrix are needed in each iteration, which reduces the amount of storage needed for each iteration.

Sequential minimal optimization (SMO) is the most popular approach for solving SVM problems. SMO performs a series of two-point optimizations. In each iteration, a working set of two points are chosen based on a selection rule that uses second-order information. Then the Lagrange multipliers for this working set are solved analytically.

Example: Natural Language Processing

We will see an example of how to use SVM to transform news into trading signals.

Neural Networks

Assume $s_t^{n_1,n_2}$ is the trading signals generated from the short n_1 and the long n_2 moving averages. Under general regularity conditions, a sufficiently complex single hidden layer feed-forward network can approximate any number of a class of functions to any desired degree of accuracy.

The Linear Test Regression

$$r_t = \alpha + \sum_{i=1}^{p} \beta_i r_{t-i} + \sum_{i=1}^{p} \eta_i s_{t-i}^{n_1,n_2} + \epsilon_t \ \epsilon_t \sim ID(0, \sigma_t^2)$$

Single Layer Feed-forward Network Model

$$r_t = \alpha + \sum_{i=1}^{p} \beta_i r_{t-i} + \sum_{j=1}^{d} \eta_j G(\alpha_j + \sum_{i=1}^{p} \gamma_i s_{t-i}^{n_1,n_2}) + \epsilon_t \ \epsilon_t \sim ID(0, \sigma_t^2)$$

where G is the **activation function** which is chosen to be a sigmoidal function:

$$G(x) = \frac{1}{1 + e^{-\alpha x}}$$

Single Layer Feed-forward Network Model with lagged returns alone:

$$r_t = \alpha + \sum_{i=1}^{p} \beta_i r_{t-i} + \sum_{j=1}^{d} \beta_j G(\alpha_j + \sum_{i=1}^{p} \gamma_i r_{t-i}) + \epsilon_t \ \epsilon_t \sim ID(0, \sigma_t^2)$$

d is the total types of signals you want to enclose in the prediction.
p is the total numbers of lags you choose to enclose in the information set of prediction.

Stochastic Gradient Descent

The third machine learning algorithm we decided to test on our data was Linear Stochastic GradientDescent (SGD). For example, to train our linear model on our data, one can iteratively fit one hundred linear models on 4830 data points in 15 day rolling window. The first 4000 of these points were part of our training set and the other 830 points were part of our validation set.

SGD uses gradient descent to find the minimum or maximum of a given function. In our case, we sought to minimize the log loss function on our data. The log loss function is a classification loss function used as an evaluation metric. Specifically, we may try to classify the signals as +1 (price going up), 0 (price staying neutral), or -1 (price going down), the log loss function quantifies the accuracy of our classifier by penalizing the false classifications our linear model makes. Using SGD allows us to select the linear model that minimizes the number of incorrect predictions we make via the log loss function. The algorithm works as follows:

1. Choose an initial vector of parameters ω and learning rate η
2. Repeat until shuffle examples in the training set.
3. For $i = 1,2,\cdots,n$ do:

$$\omega \coloneqq \omega - \eta \Delta Q_i(\omega)$$

In your code, you may imported the linear SGD algorithm from the scikitlearn library in python. Please note the following:

- ω is a vector of zeros of size n
- n is the length of our training set (in above example $\omega = 4000$)
- The stochastic gradient descent process is repeated 500 times to find the minimum of the logloss objective function
- $Q_i(\omega)$ is the log loss objective function (the gradient of this function is taken in the formula)
- η the learning rate is internally optimized by the scikitlearn library