Technical University of Crete, Greece

Department of Electronic and Computer Engineering

Online Machine Learning Algorithms For Currency Exchange Prediction

Eleftherios Soulas

Chania, February 2013

Acknowledgements

This dissertation would never have materialized without the contribution of many individuals to whom I have the pleasure of expressing my appreciation and acknowledgement.

First of all I don't think there are enough words to express my gratitude and love to my parents Christos and Angelica as well as my sister Athena. None of this would have been possible if they hadn't there been for me and I can't picture myself without their guidance and support.

Next, I would like to thank my advisor, professor Minos Garofalakis, for the trust he showed in me and our discussions about my future steps.

Next I would like to thank my friend and mentor, Professor Dennis Shasha as he was the main reason I got involved in this project, my constant inspiration and unconditional help throughout the whole time I spent in New York.

Special thanks to Timir Karia, for providing us with high frequency stock market data and all of his support through emails and Skype calls. A big thanks to Alex Rubinstein who did the "dirty work", organizing the foreign exchange data so that we could focus on the algorithm.

Thanks to my dear friend and pedagogue Alastair Fisher, who kindly amended and refined my thesis.

I would also like to thank my great friends who were there at every step of the odyssey: Alimpertis Manos, Drakatou Maria-Louiza, Kofinas Nick, Papagrigoriou Stelios, Pavlakis Nick and Perros Ioakeim.

Abstract

Recently, a lot of interesting work has been done in the area of applying Machine Learning Algorithms for analyzing price patterns and predicting stock prices and index changes. Most stock traders nowadays depend on Intelligent Trading Systems which help them in predicting prices based on various situations and conditions, thereby helping them in making instantaneous investment decisions.
This thesis describes methods for two problems:

How to find highly correlated pairs of securities over the last recent time period (e.g. over the last hour) in a sliding window fashion. The base model used for this is Statstream.

How to predict foreign exchange rate changes in an online fashion based on machine learning algorithms. This dissertation explains the algorithms and discusses various metrics of accuracy. It finalizes by simulating a real-life trading enviroment and concludes that the new algorithm proposed can be used for an intelligent trader that would predict the stock price and buy a stock before the price rises, or sell it before its value declines.

Though it is very hard to replace the expertise that an experienced trader has gained, an accurate prediction algorithm can directly result into high profits for investment firms, indicating a direct relationship between the accuracy of the prediction algorithm and the profit made from using the algorithm.

Contents

List of Figures

LIST OF FIGURES

List of Tables

Chapter 1

Introduction

In information technology (branch of engineering dealing with the use of computers and telecommunications equipment to store, retrieve, transmit and manipulate data), Big Data is a term applied to voluminous unstructured and structured data sets, which, because of their size, cannot be reasonably stored in typical databases for easy processing and managing.

Data sources are everywhere, from Web 2.0 and user-generated content to large scientific experiments, from social networks to wireless sensor networks . Figure 1.1 contains an interesting info-graphic on how much data are being generated within 60 seconds by many famous web services.

Typical examples of massive data sources are the following:

- There are roughly 50,000 securities trading in the United States, and up to 100,000 quotes and trades (ticks) are generated per second. These applications share several special streaming characteristics and demonstrate that the query processing is different from that in the conventional static or slow updating database system.

- Astronomers have long recognized the importance of a ”digital observatory” that would support the data needs of researchers across the globe - the Sloan Digital Sky Survey is perhaps the most well known of these projects. Looking into the future, the Large Synoptic Survey Telescope (LSST) is a wide-field instrument that is capable of observing the entire sky every few days. When the telescope comes online around 2015 in Chile, its 3.2 Gigapixel primary camera will produce approximately half a Petabyte of archive images every month.

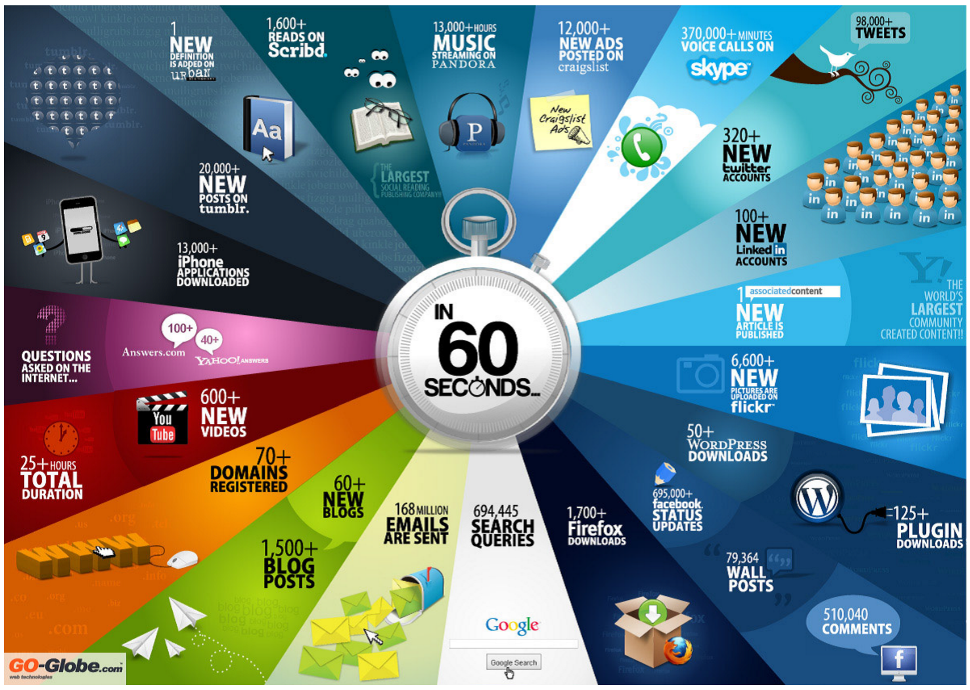

IN 60 SECONDS
1 NEW DEFINITION IS ADDED ON urban
1,600+ READS ON Scribd
13,000+ HOURS MUSIC STREAMING ON PANDORA
12,000+ NEW ADS POSTED ON craigslist
370,000+ MINUTES VOICE CALLS ON skype
98,000+ TWEETS
20,000+ NEW POSTS ON tumblr.
THE LARGEST SOCIAL READING PUBLISHING COMPANY
320+ NEW twitter ACCOUNTS
13,000+ iPhone APPLICATIONS DOWNLOADED
100+ NEW LinkedIn ACCOUNTS
1 NEW ARTICLE IS PUBLISHED associatedcontent
THE WORLD'S LARGEST COMMUNITY CREATED CONTENT!!
QUESTIONS ASKED ON THE INTERNET...
100+ Answers.com
40+ YAHOO! ANSWERS
6,600+ NEW PICTURES ARE UPLOADED ON flickr
600+ NEW VIDEOS
25+ HOURS TOTAL DURATION
70+ DOMAINS REGISTERED
50+ WORDPRESS DOWNLOADS
60+ NEW BLOGS
168 MILLION EMAILS ARE SENT
694,445 SEARCH QUERIES
1,700+ Firefox DOWNLOADS
695,000+ facebook STATUS UPDATES
125+ PLUGIN DOWNLOADS
1,500+ BLOG POSTS
79,364 WALL POSTS
510,040 COMMENTS
GO-Globe.com

Figure 1.1: Data being generated from various web sources within 1 minute.

- The Large Hadron Collider (LHC) near Geneva is the worlds largest particle accelerator, designed to probe the mysteries of the universe, including the fundamental nature of matter, by recreating conditions shortly following the Big Bang. Experiments at the LHC produce 30 Petabytes - 30 million Gigabytes - of data every year, which has to be stored, backed up, and made available to more than 8,500 scientists around the globe.

- In mission operations for NASA"s MISR Satellite, spatial samples are acquired every 275 meters. Over a period of 7 minutes, a 360 km wide swath of Earth comes into view from cameras pointed in 9 different directions.Terabyte data are generated every day.

- In telecommunications, the AT&T long distance data stream consists of approximately 300 million records per day from 100 million customers.

- The advent of next-generation DNA sequencing technology has created a deluge of sequence data that needs to be stored, organized, and delivered to scientists for

> further study. The European Bioinformatics Institute (EBI), which hosts a central repository of sequence data called EMBL-bank, has increased storage capacity from 2.5 Petabytes in 2008 to 5 Petabytes in 2009. Scientists are predicting that, in the not-so-distant future, sequencing an individual"s genome will be no more complex than getting a blood test today - ushering a new era of personalized medicine, where interventions can be specifically targeted for an individual.

One of the greatest challenges for 21st-century science is how we respond to this new era of data-intensive science. This is already been hailed as the "Fourth Paradigm" of science[2], one that is currently beyond experimental and theoretical research, or for the matter computer simulations of natural phenomena.

To this end, institutions and organizations of all size are turning to people who are capable of translating this treasure of data into valuable and tangible knowledge, establishing the profession of data scientists as one of the most sought-after professions of our times.

For this reason online response is desirable in many of those applications. Our intent is to build a foundational library of primitives to perform online or near online multi-stream information extraction on thousands or even millions of time series. Besides their immediate use, such primitives could provide a first level analysis of time series for online clustering and data mining systems.

In this project we mainly deal with Time Series Data. A time series is a series of numerical measurements related through time, $T = (t_1, y(t_1)), (t_2, y(t_2)), ..., (t_n, y(t_n))$ (see Figure 1.2).

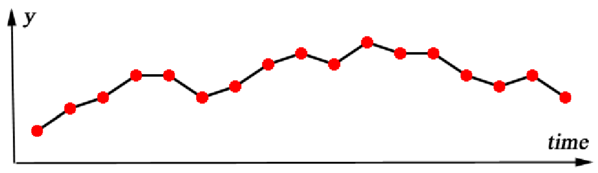


Figure 1.2: Example of time series data.

Time series are a very common form for collected data as companies and analysts are often concerned with discovering patterns in time such that they may be capable of predicting future patterns. Examples of time series include stock prices, periodic temperature readings, and other measurements made over time.

1.1 Thesis Outline

The first part of this thesis is mainly an extensive study on fast algorithms for time series manipulation, dimensionality reduction techniques and correlation discovery among vast amount of streams. All of these are investigated through Statstream. Chapter 2 describes Statstream system and the extensions we enhanced.

In Chapter 3 we present the problem and our main motivation. In Chapter 4 we adduce the fundamental knowledge one should have to familiarize with the concepts of machine learning and investment options. This knowledge is important for the following sections.

Chapter 5 presents our algorithm and explains our framework, Learnstream, which as far as we know is the first system capable of online machine learning in a streaming manor.

In Chapter 6 we adduce the experimental results based on three datasets (two foreign exchange data sets and one electrical consumption measurements). We simulate a trading environment and aver that our predictions on price movement can used to gain substantial profit.

Finally, in Chapter 7 we discuss the results of this thesis and present the combination of Statstream and Learnstream into a new system. Its name is StatLearn and is the first system capable of finding high correlated pairs and determining linear relationships among them in real time.

Chapter 2

Statstream Renewed

2.1 Background Work

2.1.1 Streaming frameworks

In many real life applications the data set under process is not a constant source of input, but rather a vast stream of continuously updated data. Often those data arrive at random rates and in no specific order. An example of such an application might be the continuous updates in large social networks and varied appliances found in finance, telecommunications, physics, biology, astronomy, etc.

Consequently there has been a pronounced interest in streaming databases that can handle queries on such type of data. Specifically, there is an ongoing research on a new type of Data Stream Management Systems (DBMS) which will be able to support continuous queries and real time updates, as traditional DBMS cannot hold continuously arriving data [3]. There are several streaming database systems that have been constructed such as Aurora[4], MAIDS[5], Niagara [6], Telegraph [7].

2.1.2 DFT Based Statstream

Zhu and Shasha[8] proposed a strategy for efficaciously and seamlessly computing sliding window pairwise correlations among thousands of streams. Statstream is responsible for finding all the stream pairs, with correlation above a threshold provided by the user, inside a sliding window. Sliding windows (sw) begin at the multiples of another entity called basic window (bw). Zhu and Shasha's algorithm finds all highly correlated pairs in a sliding window, assuming that the data inside that window can be well modeled by its first few Fourier coefficients.

They use DFT as a means of reducing the dimension of data. First few coefficients of DFT transformation are used to compute the "distance" between time series. Those time series pairs whose distance in coefficient space is smaller than a certain threshold are picked out for the second-stage verification. The above operations are performed within each sliding window and the framework can be used for both synchronous and asynchronous correlation discovery.

2.1.3 Sketch Based Statstream

Zhao and Shasha at [9], [10], extended the DFT [11] based statstream even more and presented a new sketch based data correlation strategy. With comparable efficiency, the new algorithm can handle more data types. This new statstream is a randomized algorithm. The whole algorithm is based on the Johnson-Lindenstrauss (JL) Lemma which states that high dimensional data points can be mapped to a low dimensional space while preserving the norm within a factor. *Statstream 2.0*[12] computes a synopsis vector for each time series; this is a random mapping from the original data space to a manageable low dimension.

The JL lemma proves that the approximation of the original distance may achieve sufficient accuracy with high probability as long as the synopsis size is larger than a given bound. This synopsis is used to filter the non-correlated time series pairs. In his doctoral thesis, Zhao prefers a set of strategies to determine a reliable bound for the requisite synopsis size as well as other system parameters.

After the synopsis filtering, the correlation will be verified using the full time series data on the pairs surviving the filter. Only those pairs passing the preliminary verification test will be reported as highly correlated.

2.2 Statstream

2.2.1 What's the goal of Statstream

To find correlation over windows from the same or different streams on synchronous and asynchronous (a.k.a. lagged) variations:

- ***Synchronous correlation:*** Given N_s streams, a start time t_{start}, and a window size w, find, for each time window W of size w, all pairs of streams $S1$ and $S2$ such that $S1$ during time window W is highly correlated (over 0.95 typically) with $S2$ during the same time window.

  Possible time windows are $[t_{start} \ldots start + w - 1], [t_{start+1} \ldots start + w], \ldots$ where t_{start} is some start time.

- ***Asynchronous correlation:*** Allow shifts in time. That is, given N_s streams and a window size w, find all time windows $W1$ and $W2$ where $|W1| = |W2| = w$ and all pairs of streams $S1$ and $S2$ such that $S1$ during $W1$ is highly correlated with $S2$ during $W2$.

2.2.2 Cooperative or Uncooperative?

For our problem we are interested in calculating all the pairwise correlations of Ns streams in a sliding window of size sw. Doing this naively requires $O(sw * N_s^2)$ time. One imagines though, that for applications with a vast number of streams, this time is extremely limited. Thus, our task is to study efficient ways and algorithms to speed up this somewhat intractable process. At this point we must distinguish the kinds of time series so as to understand why different optimization methods should be used. For this application we categorize the time series as cooperative or uncooperative.

- ***Cooperative:*** The time series that exhibit a substantial degree of regularity.

  There are several data reduction techniques that can be used in order to compress long time series in few coefficients as Fourier Transform methods [13], [14],[15], Wavelet Transforms [16], [17], Singular Value Decompositions [18], Piecewise Constant Approximations [19].

- ***Uncooperative:*** Time series with no such periodical regularities. An example can be the returns of a stock quote as the increase or decrease between two consecutive time-points do not necessarily follow any certain patterns.

One can see the contrast based on the number of Fourier coefficients in Figure 2.1

For this uncooperative time of time series, sketch-based approaches [20] can be used. The reason that we may want to use random projections of the time series onto random vectors is to exploit the bounds provided by the Johnson-Lindenstrauss lemma [21]. This approach will be further addressed in the following sections.

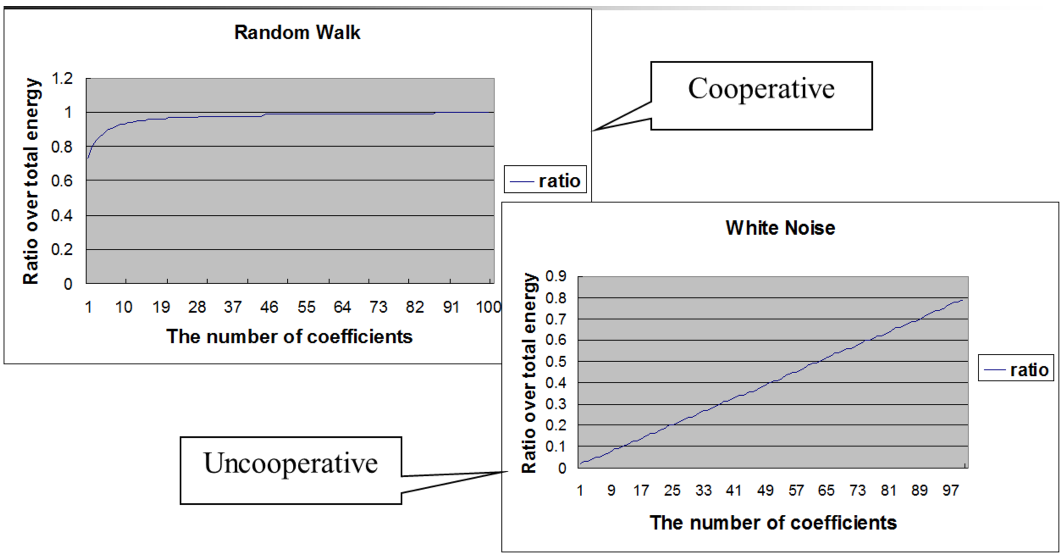


Figure 2.1: Cooperative vs Uncooperative Data.

2.2.3 Sketch Based Statstream For Financial Data

Unfortunately, many applications generate uncooperative time series.

Stock market returns $= \frac{today's\_price - yesterday's\_price}{yesterday's\_price}$ are "white noise-like". That is, there is almost no correlation from one time point to the next. For collections of time series that do not concentrate power in the first few Fourier/Wavelet coefficients, which we term uncooperative, we use the sketch based version of Statstream 2.0 [12] are implemented.

In the following sections we aim to present a few of its features and the basic ideas of its functionality.

Statstream framework partitions time as following:

- ***timepoint:*** The smallest unit of time over which the system collects data, for example a second.
- ***basic window:*** A consecutive subsequence of timepoints over which the system maintains a compressed representation, for example two minutes.
- ***sliding window:*** A user-defined consecutive subsequence of basic windows over which the user wants statistics, for example an hour. This way the user can submit queries for highly correlated stream pairs which were above a certain threshold for the past sliding window, for example an hour.

Figure 2.2 depicts a graphical representation of time perception in statstream.

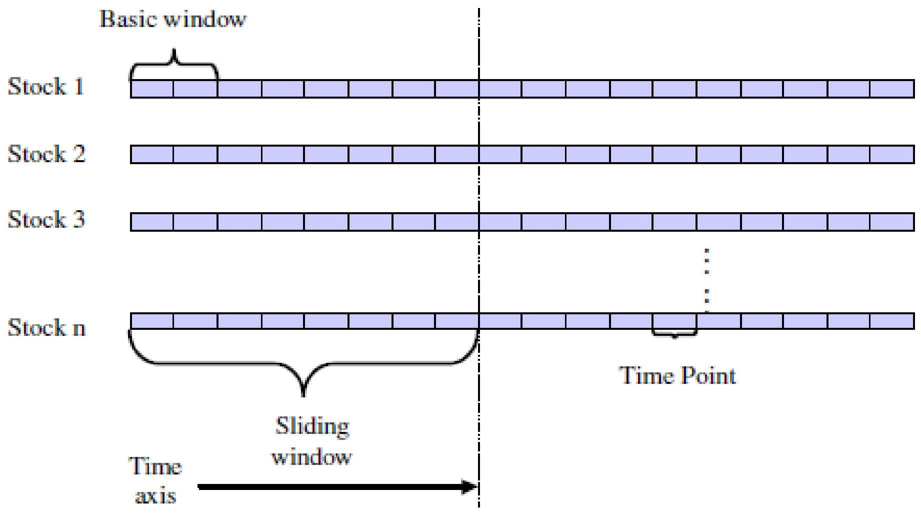


Figure 2.2: Time partitioning in statstream.

The choice of *sw* and *bw* depends on the application under consideration, because *bw* is the delay before results are reported.

For instance, a trader may ask which pairs of stock returns have been correlated with a value of over 0.9 for the last three hours and want the correlation information reported every 30 seconds.

2.2.4 Advatanges of basic window

The use of a basic window (bw) yields various benefits.

User queries need not be delayed more than the basic window time. For example, if a user asks for correlations for the $[t_i, t_{i+sw}]$ sliding window he will be informed about the results at $[t_{i+bw}, t_{i+sw+bw}]$. Assuming that the user defined $bw \ll sw$ the system has near online response rates

Moreover, due to the fact that stream digests are based on the basic window, the system allows the computation of correlations over windows of arbitrary size (chosen up front) with high accuracy.

2.2.5 Intuition & Guarantees Of The Sketch Approach

As pointed out above, the Fourier Transform behaves very poorly for "white noise" style data. For such data, sketches are invoked.

The sketch approach, as developed by Kushikvitz et al. [22], Indyk et al. [23], and Achlioptas [20], provides a very elegant bound approach: with high probability a random mapping taking points in R^m to points in $(R^d)^{2b+1}$ (the (2b+1)-fold cross-product of R^d with itself) approximately preserves distances (with higher fidelity the larger b is).

Given a point $x \in R^m$, we compute its dot product with d random vectors $r_i \in 1, -1^m$. The first random projection of x is given by $y1 = (x * r_1, x * r_2, ..., x * r_d)$.

We compute $2b$ more such random projections $y1, ..., y_{2b+1}$. If w is another point in R^m and $z1, ..., z_{2b+1}$ are its projections using dot products with the same random vectors then the median of $\parallel y_1 - z_1 \parallel, \parallel y_2 - z_2 \parallel, ..., \parallel y_{2b+1} - z_{2b+1} \parallel$ is a good estimate of $\parallel y - z \parallel$. It lies within a $\theta(1/d)$ factor of $\parallel y - z \parallel$ with probability $1 - (1/2)^b$.

2.2.6 Sketch Implementation In Statstream

It is too costly to build sketches for every new sliding window arriving. This is why Zhao and Shasha followed the approach of structured random vectors.

The apparently oxymoron idea is to form each structured random vector r from the concatenation of $nb = \frac{sw}{bw}$ random vectors $r = s_1, s_2, ..., s_{nb}$ where each s_i has length bw. Furthermore, each s_i is either u or $-u$, and u is a random vector in $\{-1, +1\}^{bw}$.

This choice is determined by another random binary k-vector b: if $b_i = 1 =¿\ s_i = u$ and if $b_i = 0 =¿\ s_i = -u$. The structured approach leads to an asymptotic performance of $O(nb)$ integer additions and $O(logbw)$ floating point operations per datum and per random vector. There is a 30 to 40 factor improvement in runtime over the naive method.

In order to compute the dot products with structured random vectors, we first compute dot products with the random vector u. For each random vector r of length equal to the sliding window length $sw = nb * bw$, the dot product with each successive length sw chunk of the stream[1] is computed to form the sketch.

2.2.6.1 Example

The theory above may be somewhat difficult to digest at first, thus so it will be instructive to present a very simple example.

As we mentioned a random vector r_{bw} is constructed as follows:

$$r_{bw} = (r_0, r_1, ..., r_{bw})$$

where

$$\{r_i\} = \begin{cases} +1, & \text{with probability } \frac{1}{2} \\ +1, & \text{with probability } \frac{1}{2} \end{cases}$$

To form a random vector of length sw, another random vector b is constructed of length $nb = \frac{sw}{bw}$. We call the second vector the control vector.

$$b = (b_0, b_1, ..., b_{nb})$$

where

$$\{b_i\} = \begin{cases} +1, & \text{with probability } \frac{1}{2} \\ +1, & \text{with probability } \frac{1}{2} \end{cases}$$

The random vector r for a sliding window is then built as follows[2]:

$$r = (r_{bw} * b_0, r_{bw} * b_1, ..., r_{bw} * b_{nb})$$

[1]successive chunks being one timepoint apart and bw being the length of a basic window

[2]This reduction of randomness does not noticeably diminish the accuracy of the sketch estimation

Now lets define the following example. Let's assume we are given a time series $X = (x1, x2, ...)$, sliding window of size $sw = 12$, and a basic window of size $bw = 4$. If the random vector within a basic window is $r_{bw} = (1, 1, -1, 1)$, the control vector $b = (1, -1, 1)$, the random vector for a sliding window will be $r_{sw} = (1, 1, -1, 1, -1, -1, 1, -1, 1, 1, -1, 1)$.

So now we will form the sketches out of the two streams (of sw size).

$$X^1_{sw} = (x1, x2, x3, x4, x5, x6, x7, x8, x9, x10, x11, x12 \tag{2.1}$$

$$X^5_{sw} = (x5, x6, x7, x8, x9, x10, x11, x12, x13, x14, x15, x16) \tag{2.2}$$

Sketch for X^1:

$$\begin{aligned} X^1_{sk} &= r_{sw} * X^1_{sw} \\ &= b_1 * r_{bw} * (x_1, x_2, x_3, x_4) + b_2 * r_{bw} * (x_5, x_6, x_7, x_8) + b_3 * r_{bw} * (x_9, x_10, x_11, x_12) \\ &= b * (r_{bw} * (x_1, x_2, x_3, x_4), r_{bw} * (x_5, x_6, x_7, x_8), r_{bw} * (x_9, x_10, x_11, x_12)) \end{aligned} \tag{2.3}$$

Sketch for X^5:

$$\begin{aligned} X^5_{sk} &= r_{sw} * X^5_{sw} \\ &= b_1 * r_{bw} * (x_5, x_6, x_7, x_8) + b_2 * r_{bw} * (x_9, x_10, x_11, x_12) + b_3 * r_{bw} * (x_13, x_14, x_15, x_16) \\ &= b * (r_{bw} * (x_5, x_6, x_7, x_8), r_{bw} * (x_9, x_10, x_11, x_12), r_{bw} * (x_13, x_14, x_15, x_16)) \end{aligned} \tag{2.4}$$

Now we will proceed with the comparison of the sketches so as to discover higly correlated pairs.

2.2.7 Sketch Vector Partitioning

In this implementation we have 60[1] random vectors to which each window is compared and the sketch vector is the vector of the dot products to those random vectors. In Figure 2.3 we can see graphically the procedure described above. As Zhu and Shasha showed [8], multi-dimensional search structures don't work well for more than 4 dimensions in practice.

[1]This is a parameter that can be modified

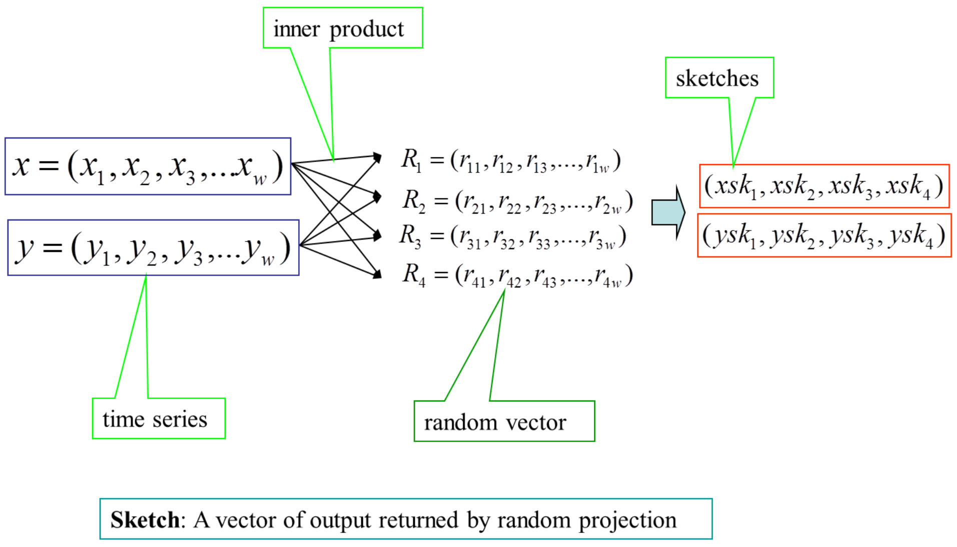

inner product
sketches
$x = (x_1, x_2, x_3, \ldots x_w)$
$y = (y_1, y_2, y_3, \ldots y_w)$
$R_1 = (r_{11}, r_{12}, r_{13}, \ldots, r_{1w})$
$R_2 = (r_{21}, r_{22}, r_{23}, \ldots, r_{2w})$
$R_3 = (r_{31}, r_{32}, r_{33}, \ldots, r_{3w})$
$R_4 = (r_{41}, r_{42}, r_{43}, \ldots, r_{4w})$
$(xsk_1, xsk_2, xsk_3, xsk_4)$
$(ysk_1, ysk_2, ysk_3, ysk_4)$
time series
random vector
Sketch: A vector of output returned by random projection

Figure 2.3: Sketch by random projection .

2.2.8 Grid Structure

To determine the closeness of the vectors we partition each sketch vector into sub vectors and build data structures for the sub vectors.

Given the idea of partitioning sketch vectors, the problem comes down to how we can combine the results of the different partitions.

For example, if each sketch vector is of length 40, we might partition each into ten groups of size four. Consequently, this would yield ten data structures. Then the adjacent results of pairs, are combined from each data structure to determine an overall set of candidate correlated windows.

We use correlation and distance more or less interchangeably because one can be computed from the other once the data has been normalized. Specifically, Pearson correlation is related to Euclidean distance as follows:

$$D^2(\hat{x}, \hat{y}) = 2(1 - corr(x, y))$$

[1]

[1] $\hat{x}$ and $\hat{y}$ are obtained from the raw time series by computing $\hat{x} = x - avg(x)var(x)$

So the algorithmic framework described above, and presented in the papers related to statstream [11], [9], [24], behaves as follows.Suppose we are seeking points within some distance d in the original time series space.

- Partition each sketch vector s of size N into groups of some size g.
- The i_{th} group of each sketch vector s is placed in the i_{th} grid structure of dimension g (In Figure 2.5 g = 2).
- If two sketch vectors s_1 and s_2 are within distance $c * d$[1] in more than a fraction f of the groups, then the corresponding windows are candidate-highly-correlated windows and will accordingly be checked exactly.

In figure 2.4 you can see the how the sketches are compared and the grid filtering outputs the candidate highly correlated pairs.

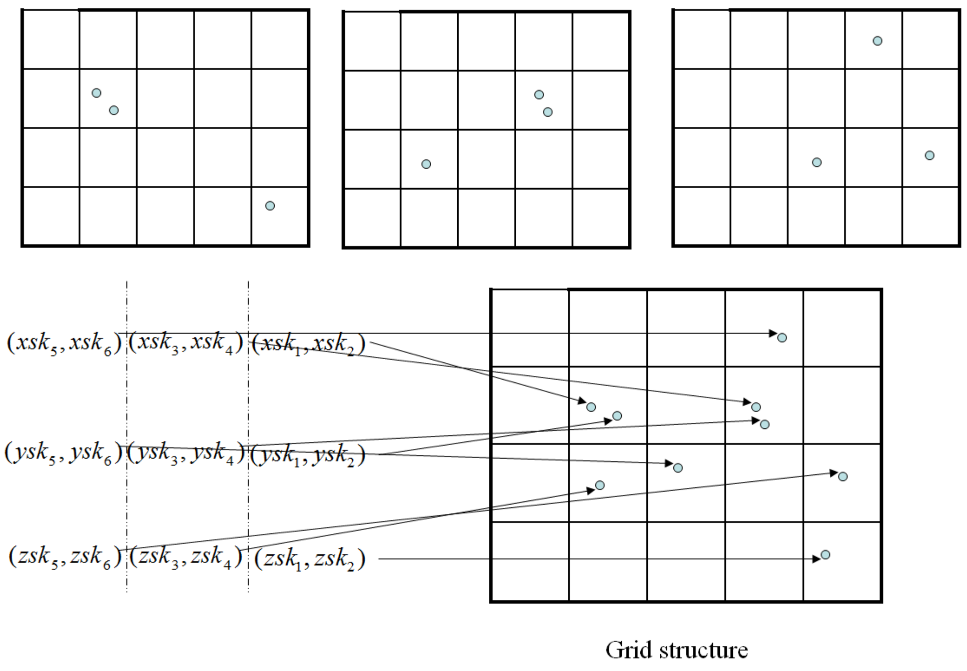

(xsk_5, xsk_6) (xsk_3, xsk_4) (xsk_1, xsk_2)
(ysk_5, ysk_6) (ysk_3, ysk_4) (ysk_1, ysk_2)
(zsk_5, zsk_6) (zsk_3, zsk_4) (zsk_1, zsk_2)
Grid structure

Figure 2.4: Grid structure.

In order to derive a better intuition about the grid structure of Figure 2.5, an analysis of its components is desirable. Assume a set of data points in a 2D space, where a 2-dimensional orthogonal regular grid is super-imposed on this space. In practice the

[1] c is a user defined parameter

indexed space is bounded. Let the spacing of the grid be a. The indexing space, a 2-dimensional square with diameter A and partitioned into $[\frac{A}{a}]^2$ small cells. Each cell is a 2-dimensional square with side length a. All the cells are stored in a 2-dimensional array in main memory.

In such a main memory grid structure, we can compute the cell to which a particular point belongs. Let us use $(c1, c2)$ to denote a cell that is the c_1^{th} in the first dimension and the c_2^{th}in the second dimension. A point p with coordinates $x1, x2$ is within the cell $(\lfloor \frac{x1}{a}, \frac{x2}{a} \rfloor)$. We say that point p is mapped to that cell. This can be easily extended to k-dimensions.

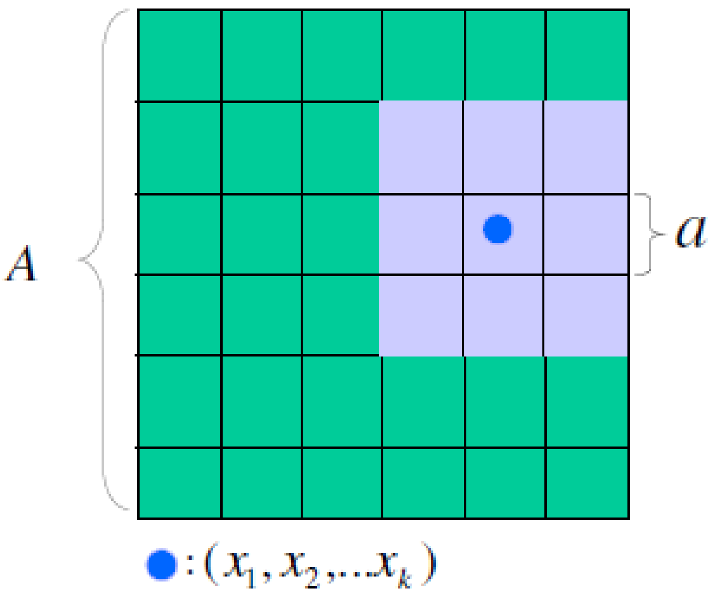


Figure 2.5: Grid structure.

2.2.9 Resutls

Sketches work much better than the SVD method, thus better than Fourier methods, for uncooperative data. Figure 2.6, compares the distances of the Fourier and sketch approximations for 1,000 pairs of 256 timepoint windows with a basic window size of length 32. Here sketch size=30 and SVD coefficient number=30. As one can see, the sketch distances are closer to the real distance

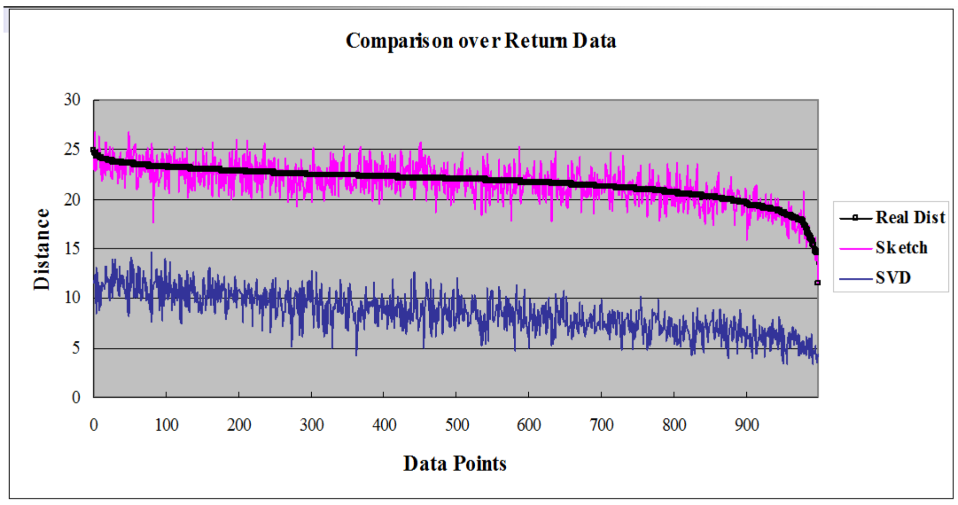


Figure 2.6: Sketch by random projection .

Moreover one can see that the graph between the sketch and the real distance Statstream-related papers provides many more guarantees based on bootstraping and experimental results, that for the sake of simplicity, will not be further analysed in this thesis.

2.3 Statstream Renewed

Our contribution to this part of the project was mostly to refurbish and refine Statstream in an elegant way.

2.3.1 Translation To Python

At first we translated Statstream from the language *K*[25] to *Python*. K is an extremely fast language. It is usually faster than the corresponding C program, e.g. present value calculation and almost always faster than Sybase or Oracle. Kx language, trusted by the world's premier financial institutions, stores massive data volumes and makes real-time decisions feasible.

Although *K* is a very efficient language, it nevertheless has a meager extant market (typically around 1000 programmers). Above all its syntax and way of use are not trivial per-se, something which may discourage programmers to adopt it.

On the other hand, *Python* is a very well known and highly developed language with a huge community. There are many packages and libraries for parallel programming, linear algebra operations and so on, which make the life of the programmers easier.

For these reasons we thought it would be a good idea to build a framework in a well known language in order to motivate others to use it and initiate their projects based on it.

This translation operation demanded substantial understanding of the algorithmic ideas of Statstream and a good knowledge of *K*'s commands, as its code can get very complex in a very few lines.

2.3.2 Debugging

Reviewing the code of statstream in K we noticed some inconsistencies.

The sketches are created based on the regularized time series. The standard way to regularize a given dataset is to subtract the *mean* and divide by the *standard deviation(s.t.d.*. When you subtract something from a set of numbers you shift the mean. When you shift the mean by the entire mean, the new mean will be 0. Finally, when you divide all the numbers by the s.t.d. you will normalize them so that the new spread has a deviation of 1.

In file *sketch˙statstream.k*, the time series are normalized in this line of code:

```
temp:(sketch.dotprods[listindex;‘bwu]-\:(sumu*sketch.sum%sw))%sketch.std
```

The misconception lies in the definition of standard deviations(std):

```
sketch.std:_sqrt ((sketch.sum2)- (_sqr sketch.sum)%sw)
```

Close inspection of the above code shows that this quantity is not really the standard deviation but the standard error. You can see the relationship between standard deviation and standard error from the below formula:

$$SE_{\hat{x}} = \frac{std}{\sqrt{n}}$$

where s is the sample standard deviation (i.e., the sample-based estimate of the standard deviation of the population),
and n is the size (number of observations) of the sample.

The Standard Error is the term used in statistics to refer to an estimate of that standard deviation, derived from a particular sample used to compute the estimate.

Although, there is no obvious reason on why we may want to divide the series by the standard error and not the standard deviation as then the spread of our dataset won't be 1.

2.3.2.1 Division By Zero

As we deciphered each line of Statstream we came across a minor bug. Specifically, in cases when the streams are dense and the consecutive values are too close to each other, there is a division by zero error. The reason for this bug is that before sketches are formed, the normal time series are firstly normalized by subtracting the mean and dividing by the standard deviation. If all the values are the same, standard deviation becomes zero. And, of course, division by zero is something best left to the philosophers.

We worked our way around by changing the normalization procedure and taking into account that dense arrays, like the financial dataset described in chapter 6.4.3, may be used.

2.3.3 Graphical User Interfaces

Statstream was a console application, something that could be obscure to the basic user who wants to change parameters and interact with the program in a more convenient way, without any knowledge of K or any other programming language.

In order to make the system more user-friendly we implemented a graphical user interface in Python. The program initializes and prompts for the user to select the input/output files and define values for all the parameters of Statstream. In this way he does not need to bother about having the same input names or generating the output in a predefined place, as was the case in the last version. This can be seen in Figure 2.7

When Statstream finishes processing the data, it reports the correlated pairs , in a *.csv* file format as well as in a graphical user interface screen.

As shown in Figure 2.8 there is a mesh that represents all the connections that pairwise connections between the streams. On the bottom of the screen there is a slider which is used to observe all the time points where sliding windows occur. Every dot in the grid, means that for this particular time frame the specific streams that cross are highly

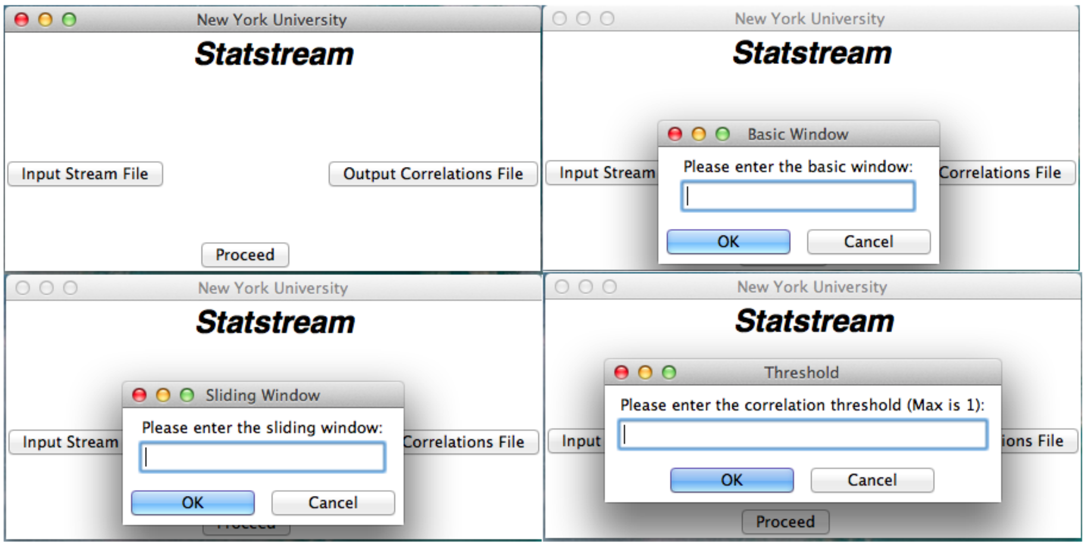


Figure 2.7: Initialization Gui.

correlated. If you mouse over the red dot, it is highlighted as green, and you can see the correlation coefficient value.

All in all this familiarization with Statstream's theoretical background provided extensive comprehension of state-of-the-art techniques for streaming databases, data reduction techniques and approximate algorithms using sketches. Moreover, at the end of this dissertation an explanation is proffered as to how StatStream and LearnStream, were combined to compose a new platform called StatLearn.

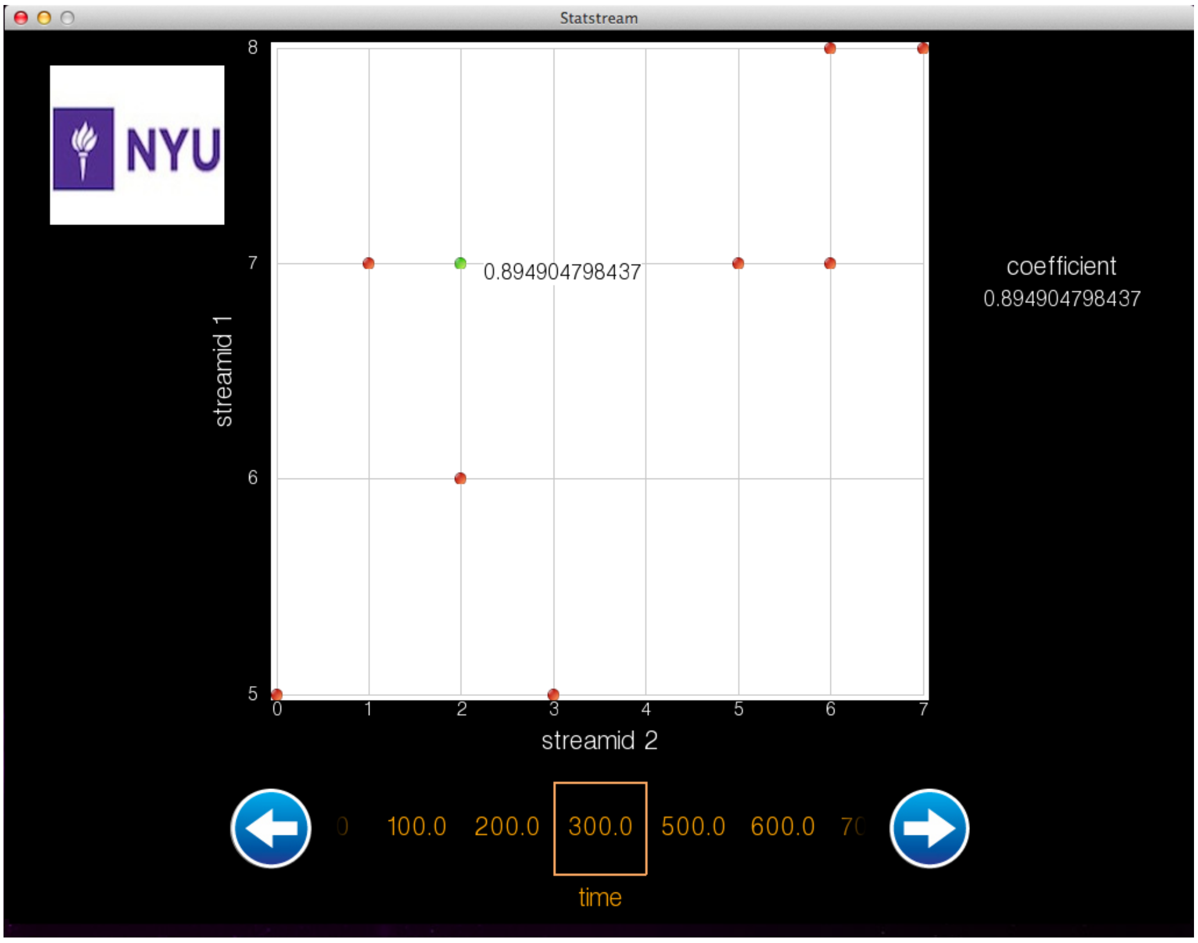


Figure 2.8: High Correlated Pairs GUI.

Chapter 3

Online Machine Learning

3.1 Introduction

Machine learning (ML) and related methods have produced some of the financial industry's most consistently profitable proprietary trading strategies during the past 20 years. With markets, trade execution and financial decision making becoming more automated and competitive, practitioners increasingly recognize the need for ML.

Moreover, the recent development of high frequency databases, i.e. a dataset containing tick-by-tick data on trades and/or orders, allows for empirical investigations of a wide range of products in the financial markets. In this way the advent of high frequency data bases contribute to shedding new light on model estimations and on econometric methods of market micro-structure.

Financial topics include high frequency data, trading strategies, execution models, forecasting, volatility, extreme events, credit risk, portfolio management, yield curve estimation, option pricing, and selection of indicators, models, and equilibria.

3.2 Motivation

The main motivation of our project was to differ from trivial econometric analysis and present a novel perspective as to how financial time series may be studied. Correlation financial analysts are mostly interested in linear relationships between stocks, for example cointegration. We want to examine more advanced and perhaps, even arguably,abstruse models such as 4*DJI -35*SNP500 = 2.4*NASDAQ.

Additionally, we aim to present a new method for trading strategies based on predictions of the price movement. We conclude our analysis by simulating a simplistic model of a trading environment. The trading strategy we follow is the most basic we could think of, meaning that the predictions based on our new algorithm manage to deliver substantial profits even if the trading part of the algorithm is not necessarily the most sophisticated. For the most part we based our analyses on forex data but we also experimented with more than financial datasets.

Apart from the financial focus of this essay, our overriding goal was to deliver software capable of being used in a variety of more generalized machine learning and streaming applications in multifarious environments .

Chapter 4

Background

4.1 Investment Options

4.1.1 Stocks

The stock can be bought in the primary or secondary market. When the company issues the stock for the first time, also called public issue, anyone can subscribe to this issue. There may be some restrictions regarding the number, the mode of payment and place where the applications can be submitted. The company issues a prospectus, a detailed document giving complete information regarding the company including the time frame for the project, utilization of the funds, future prospects, and risks perceived by the management and so on. The prospective applicants are advised to study this document carefully. The public issue is kept open for a few days, enabling the interested persons to apply. After all the applications have been received the shares are issued within the stipulated time. The company may also invite applications in case of substantial expansion, although such public issues are few and far between.

The other market is the secondary market. Huge transactions take place in this market every working day. Here the existing shareholders sell their shares to buyers. To purchase or sell shares in this market a person has to register himself with a broker or a broker house, authorized to operate in this market; the shares are quoted on a stock exchange and this information is widely available. The intimation to purchase or sell (quantity and price) has to be confirmed to the broker. Some brokerage commission has to be paid. After the necessary formalities, which may take at most a few days, the transaction is completed. The shares are kept in a depository and the details are given to the account holder periodically. The advantage of the secondary market is that the past performance of the company is available for study.

While investing in stocks it is necessary to remember that liquidity is low. Only funds not likely to be needed urgently should be invested. It is absolutely essential to study the background and the past performance of the company. The performance should be compared with the performance of the competitors. To minimize the risks, it is advisable to have diversified stocks. One must devote time to study the trends and the market movement of stocks. Stock markets these days follow a global trend. Investors monitor not only NYSE & NASDAQ but also FTSE, NIKKEI, HANG SENG as well as DAX and CAC.

4.1.2 Bonds

A bond is defined as a long-term promissory note with stipulated interest supported by a consideration or under seal secured by a mortgage. Bonds hold the promise of stipulated interest on a long-term basis. There is a guarantee for the performance. Those issued by the Governments are also termed securities. The issuing Government or Federal Government, in the case of issue by State Government or Local Authority, guarantees the payment.

Companies issue debentures. These may be secured by a charge on specific assets of the company. To ensure proper compliance of the regulations and proper upkeep and maintenance of the assets, a trust is formed or trusties are appointed. Even debt instruments issued by companies are covered under the broad term BOND for the purpose of investments. It is compulsory for such companies to get a rating from the recognized Rating Agencies. This helps in estimating the repaying capacity of the company. Triple A (AAA)is the highest rating. The interest on a bond can be fixed for the entire period, or it can be floating.

The floating rate will be linked to either the bank rate or some other independently determined rate such as LIBOR. In general, the safety of the investment and the regular income from the interest are assured. The market price of the bonds does not fluctuate widely only on the market. This ensures liquidity.

A bond-holder is a secured creditor. He has legal rights to sue the company in case of default. Bonds maintain a balance of safety, yield and liquidity. The returns in investments from bonds over a period of time are likely to yield lower returns than the stock market. In several financial applications, such as stock trading, past data hold much information, which, if studied can produce meaningful results. The whole story lies in the ways one utilizes the research for techniques and algorithms and so exploit those data, using them to derive more appropriate future decisions and/or explaining the past.

4.1.3 Mutual Funds

An individual investor who wishes to invest in stock but has limited money. On the other hand, the different stocks being traded in the stock market are quite large. When an opportunity arises to purchase some stock, he may not have the liquidity necessary capital. He may not be able to study the trends in stock market. He may not be able to analyse the movement of prices in the stock market. It may be difficult for him to visualize the future prospects of different categories of industries. He may not be able to analyse the performance of individual companies and the changes in their management. In short very few persons can have the time, knowledge and skills to take the best advantage of opportunities that arise in the stock market. Mutual funds are basically investment companies, which continuously sell and buy stock. Anyone can participate in its activities by investing in the mutual fund. The investment company, usually a trust, manages the total capital available to a mutual fund. All the stock owned, by this company, valued at the market price, is the net asset, value or NAV. This amount divided by the total number of units issue, will be the NAV per unit. The Mutual Fund Company continuously sells the units and repurchases its units on a daily basis by announcing NAV daily. The Mutual Fund Company will buy the units from the investor at his option at any time at the NAV. For managing the fund, the company will charge some commission called load. This can be charged either at the time of selling or at the time of repurchase. It can be seen that by investing in a mutual fund one can get the benefit from the broader market and the expertise of the professional management. The fund manager of AMC observes the stock market all the time, trying to get the best yield for the investors. Mutual funds state specific investment objectives in their prospectus. The main type or objectives are growth, balanced income, and industry specific funds. Growth funds possess diversified portfolios of common stocks in the hope a portfolio of stocks, and bonds. This achieves both capital gains and dividend along with interest income. Income funds concentrate heavily on high interest and high dividend yielding securities. Industry specific funds invest in portfolios of selected industries. This appeals to investors who are extremely optimistic about the prospects of these few industries. One should be willing to assume the risks associated with such a concentration of investment. As happened in information technology bad performance results in huge losses. Sometimes the same company may have a family of mutual funds. The investors may be allowed to shift from a fund with one objective to a fund with a different objective for a fee.

4.1.4 FOREX

The following investment option is of major importance as it was the central focus of this project.

Each country has its own currency. Whenever one currency is exchanged with another it is a foreign exchange or Forex transaction. The foreign exchange market has experienced many changes since its inception. For many years the United States and its allies participated in a system under the Bretton Woods Agreement. Foreign exchange rates were tied to the amount of gold reserves belonging to the nation. However, in the summer of 1971, President Nixon took the United States off the gold standard. After this many countries followed and there was no relation between the gold reserves and the exchange rates. Floating exchange rates came into existence.

Today supply and demand for a particular currency, or its relative value, is the driving factor in determining exchange rates. The fall of communism and the dramatic growth of the Asian and Latin American economies have decreased obstacles and increased opportunities for investors in foreign exchange. Increasing trade and foreign investment have made the economies of all nations more and more interrelated. Economic figures are regularly reported around the world. Inflation, unemployment levels, unexpected news, such as natural disasters or political instability, alters the desirability of holding a particular currency. This influences the international supply and demand for that currency.

The U.S. dollar, therefore, fluctuates constantly against currencies of the rest of the world. Forex investments are investments in a currency other than that of your own country. If you have U.S. dollars your investment is in dollars. If you have British

Pounds your investment is in Pounds. If you desire to visit a foreign country and you know the approximate amount of money you will spend then you have the option of either taking your own currency to that country or exchanging the same when you visit that country. You also have the option of exchanging the currency in your own country and keeping the currency of the foreign country with you well before you visit that country. e.g. You are to visit Japan but you are at present in New York. You can change the U.S. dollars into Japanese Yen before you leave. This is a foreign exchange investment. You would do it if you think the Yen is going to become stronger. i.e. In future you will get less yen for dollars.

Forex investments are essentially trading in the future via options. An option gives you the right, but not an obligation, to buy or sell a specific amount or foreign exchange at a specified price within a specified period. Options are termed either call or put. A call gives the holder the right to buy the foreign currency or Forex at a specified price. The put gives the right to sell Forex at a specified price. Depending on the actual market price when you exercise the option you will gain/lose the difference between the specified price and the market price.

The question lies as to how people can trade in Forex Market and what underlying risks should be taken into account. Trading in the Forex market is executed through the brokerage houses that deal in foreign exchange. You can trade by options in the Forex market; although the Forex market may go up or go down. If you expect the Forex market to go up you will invest in calls e.g. the value of 1 us dollar today is 48 Indian Rupees. If you expect the Forex market will change to 1 us dollar equal to 55 Indian Rupees in four months then you can go invest call option; agreeing to buy 10 U.S. dollar at the rate of 50 Indian Rupees / dollar at any time during the next six months. Whenever the actual market price is above 50 you can exercise the option. You can actually buy 10 U.S. dollar by paying only 50 Rs. Per dollar i.e. by paying only Rs 500. But the actual value being 540 Rs. Your gain is Rs. 40.Similarly, if you think the market is going to be Rs. 40 per dollar, you can invest in a put option. In this case you will be able to sell the dollars in Forex market at the agreed price i.e. Rs. 48/dollar though the actual market price is less i.e. Only 40 Rs/dollar.

The gain an investor makes will depend on the actual difference in the market price and the options price. However, there will be some fee/commission levied. This would be the cost of transaction, and result in a reduction of the gain, to some extent.

The foreign exchange market is the largest financial market in the world. Traditionally the foreign exchange market has only been available to banks, money managers and large financial institutions. Over the years, however, these institutions, including the U.S.Federal Reserve Bank, have realized large gains via currency trading. This growing market is now linked to a worldwide network of currency dealers, including banks, central banks, brokers, and customers such as importers and exporters. Today the foreign exchange market offers opportunities for profit not only to banks and institutions, but also to individual investors. More accurately it can be noted that large markets involve trading of $1,500 billion every day. Market participants anticipate the direction of currency prices generate the bulk of currency activity.

In general, the value of currency vs. other currency (i.e. Exchange rate or foreign exchange rate) is a reflection of the condition of that country's economy with respect to the other major economies. George Soros, (in)famously, took a massive position against the British Pound in 1992 and virtually forced the U.K.government out of the semi-fixed exchange rate mechanism with its EU partners. He made a fortune out of this transaction. You can lose money also. The quantum fund set up by George Soros produced remarkable annual compound returns of 30% between 1969 and 1987.

Nevertheless Forex trading presents many computational challenges that may be applicable for state of the art computer science techniques.

4.1.5 Why do people invest in the Forex Market?

People invest in Forex markets because of the considerable opportunities they offer. An individual investor can realize huge profit potential by buying or selling a particular currency against the U.S. dollar or any other major currency.

Investors can generate profits whether a currency is rising or falling. Buying one currency, which is expected to rise, against another currency can do this. Or you may sell one currency, which is expected to fall, against another currency.

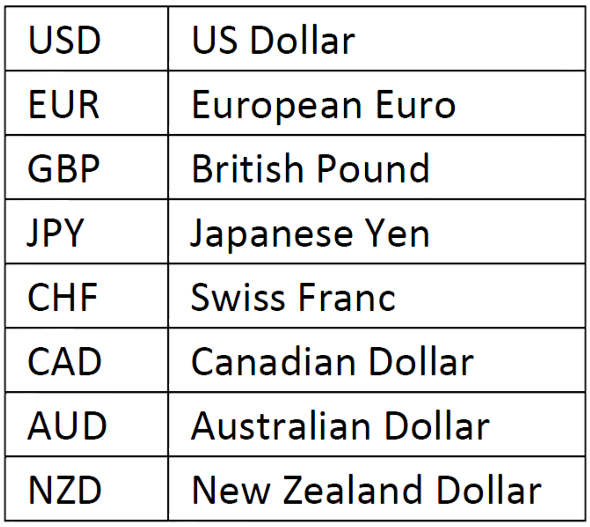
Taking a long position means buying a currency at one price and aiming to sell it later at a higher price. A short position is one in which the investor sells a currency that he hopes will fall and aims to buy it back later at a lower price.

Depending on the risks that an investor is prepared to take the gains can be substantial. The style of George Soros was to take big, often interlinked speculative position using lots of leverage. It was possible to produce a 10% gain in the net worth of the fund that he was managing by means of a 1% move in the YEN.

4.1.6 Bid/Ask Price

The forex market has bid and ask prices that are constantly changing. As explained investors trade foreign currency units that can be as small as a thousand units or as large as a billion units.

The eight most traded currencies of the FOREX Market are shown in Figure 4.1



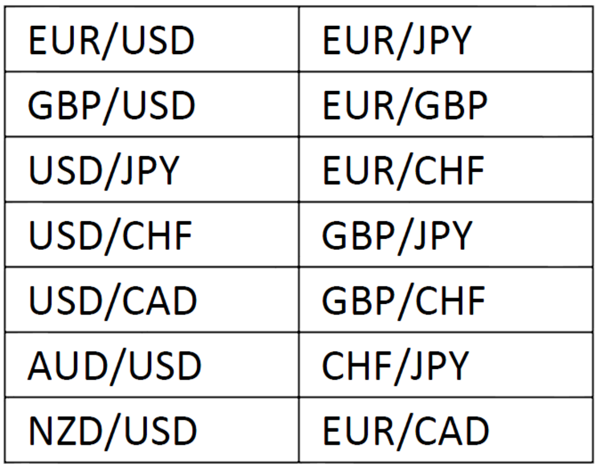
USD	US Dollar
EUR	European Euro
GBP	British Pound
JPY	Japanese Yen
CHF	Swiss Franc
CAD	Canadian Dollar
AUD	Australian Dollar
NZD	New Zealand Dollar

Figure 4.1: Most traded currencies.

Forex trading is always done in pairs, since any trade involves the simultaneous buying of a currency and selling of another currency. The trading revolves around 14 main currency pairs. These pairs are shown in Figure 4.2

4.1.7 Bid Price

The bid price is the price at which the dealer firm would be best able to buy the foreign currency units. This, as mentioned, will always be lower than the quote price. Because the dealer firm is buying the foreign currency units, the investor would be selling at the bid price.



EUR/USD	EUR/JPY
GBP/USD	EUR/GBP
USD/JPY	EUR/CHF
USD/CHF	GBP/JPY
USD/CAD	GBP/CHF
AUD/USD	CHF/JPY
NZD/USD	EUR/CAD

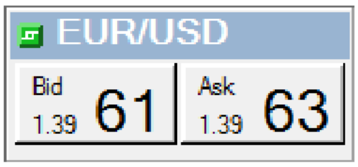
Figure 4.2: Currency Pairs.

4.1.8 Ask Price

Also known as the offer price, the ask price is the price at which the dealer firm would be best able to sell the foreign currency units. This will always be greater than the quote price. Depending on the dealer firm, there could be an ask price that is close to the quote price or far away. This same concept applies for the bid price.

In the case of the ask price, if the dealer firm is selling the foreign currency units, this means that the investor would be buying at the ask price.

An example of bid/ask prices is shown in Figure 4.3



EUR/USD

Bid 1.39 61

Ask 1.39 63

Figure 4.3: Ask/Bid Prices.

4.1.9 Bid-Ask Spread

The bid-ask spread is the width of the buying and selling prices. If the market is in a so-called fast market, the prices will fluctuate in a way that will make the bid and ask spread widen on the side of the asking price or on the side of the bid price. This depends on which way the foreign currency market is heading. If it is a fast market and heading down, the bid prices will be much lower than the quote price. The opposite is also true for the ask price in an up market.

In other words, thinking of it as simple subtraction, it is the difference between the highest a buyer is willing to bid and the lowest a seller is willing to ask.

For example, we can use the euro and dollar currency pair, the EUR/USD pair. The quote currency in this case would be the U.S. dollar, while the euro is the base currency. That means the quote for the bid and the ask would be in terms of U.S. dollars. If the dealer firm has a bid-ask spread of three pips and the current quote is 1.3008, we can say that the bid will be below the current price and the ask will be higher than the current price. The bid may be at 1.3007, and the quote currency ask may be at 1.3010. This leaves the 3 pips difference between them. The following explains what the bid price is, what the ask price is and what the bid-ask spread is.(from Financial Web [26])

The terms of bid and ask prices are of great importance as they will be used in later chapters. 1

4.2 Machine Learning

ML themes include reinforcement learning, optimization methods, recurrent and state space models, on-line algorithms, evolutionary computing, kernel methods, Bayesian estimation, wavelets, neural nets, SVMs, boosting, and multi-agent simulation.

For this project we were mainly interested in on-line linear regression algorithms which will be further discussed later on.

4.2.1 Supervised and Unsupervised Learning

There are two discernible categories of machine learning: supervised and unsupervised learning. In supervised learning, the system receives a dataset as an input and uses it to make decisions and classifications, from which it infers a mathematical function that maps every element in the input to the output.

On the contrary unsupervised learning is a pure learning process where the system must classify and make decisions based only on the sequence of its past actions via trial and error. What makes unsupervised learning techniques really complex is that such a system can set up hypotheses, no human can figure out. One would also run the risk of obtaining a hypothesis too complex or specific to aid researchers (as per Occam's razor).

For this project we were mainly interested in supervised learning and specifically gradient descent type of algorithms which are proved to be adequate for many machine learning applications. This section aims to present a brief introduction to the various forms of batch and stochastic gradient descent. Much research and novelty work has been developed in this area, especially by Bottou et al. [27],[28],[29],[30] as well as from Yann LeCun et al. [31], [32] This will be the theoretical foundation for the algorithms proposed in the later chapters.

4.2.2 Gradient Descent

There are two main variations of gradient based algorithms.

- Batch algorithms avoid this issue by completely optimizing the cost function defined on a set of training examples. On the other hand, such algorithms cannot process as many examples because they must iterate the training set several times over to achieve the optimum. As datasets grow to practically infinite sizes, we argue that online algorithms outperform learning algorithms that operate by repetitively sweeping over a training set.

- Online algorithms operate by repetitively drawing a fresh random example and adjusting the parameters on the basis of this single example only. Online algorithms can quickly process a large number of examples. On the other hand, they usually are not able to fully optimize the cost function defined on these examples.

In order to get an intuition about the gradient descent algorithms lets first consider a simple example. Given a pair of (x, y) composed of an arbitrary input x and a scalar y, we consider a *loss function* $\ell(y, \hat{y})$ that represents the cost of predicting $\hat{y}$, based on x, when the actual answer is y. If $\mathcal{F}$ is a family of functions $f_w(x)$ weighted by a vector w, we seek the function $f \in \mathcal{F}$ that minimizes the loss $Q(z, w) = \ell(f_w(x), y)$, averaged on all the examples. Although we would like to average over the unknown distribution $dP(z)$ we must often settle for computing the average on a sample $z_1 ... z_n$

Let

$$E_n(f) = \frac{1}{n}\sum_{i=1}^{n} \ell(f(x_i), y_i)$$

be the *empirical risk* which measures the training performance and

$$E(f) = \int \ell(f(x), y) dP(z)$$

be the *expected risk* that measures the performance on future samples.

4.2.3 Batch Gradient Descent

In order to minimize *empirical risk* $E_n(f_w)$ the use of gradient descent is proposed [33] and each iteration of the algorithm updates the weights w on the basis of the gradient of $E_n(f_w)$ and

$$w_{t+1} = w_t - \eta \frac{1}{n} \sum_{i=1}^{n} \nabla Q((x_i, y_i), w_t), \tag{4.1}$$

where η is an adequately chosen gain. As stated in [34], [29], under sufficient regularity assumptions, learning gain η sufficiently small and the initial estimate w_0 close enough to the optimum, this algorithm achieves *linear convergence.*

Many optimization algorithms have been designed by replacing the scalar η with a positive definite matrix Γ_t that approaches the inverse of the Hessian of the cost at the optimum,

$$w_{t+1} = w_t - \eta \frac{1}{n} \sum_{i=1}^{n} \nabla Q((x_i, y_i), w_t)$$

This *second order gradient descent* is a variant of the well known Newton algorithm. Under the same regularity assumptions as stated above, second order gradient descent achieves *quadratic convergence.*

4.2.4 Stochastic Gradient Descent

The computational complexity of learning algorithms becomes the critical limiting factor when one envisions very large datasets. Our implementation advocates stochastic gradient algorithms for large scale data mining and machine learning problems and builds on top of them. This is why stochastic gradient descent is of crucial importance to this project as it constitutes the cornerstone of our algorithm.

Stochastic Gradient Descent is a simple yet very efficient approach to discriminative learning of linear classifiers under convex loss functions such as (linear) Support Vector

Machines and Logistic Regression. Even though SGD has been around in the machine learning community for a long time, it has received a considerable amount of attention just recently in the context of large-scale learning.

SGD has been successfully applied to large-scale and sparse machine learning problems often encountered in text classification and natural language processing. The advantages of Stochastic Gradient Descent are:

- Efficiency.
- Ease of implementation (lots of opportunities for code tuning).

The disadvantages of Stochastic Gradient Descent include:

- Requires a number of hyperparameters such as the regularization parameter and the number of iterations.
- Sensitive to feature scaling.

Stochastic Gradient Descent algorithm is a drastic simplification of the batch gradient descent. Instead of computing the gradient of $E_n(f_w)$ exactly, each iteration estimates this gradient on the basis of a single randomly picked example:

$$w_{t+1} = w_t - \eta_t \nabla Q((x_i, y_i), w_t) \tag{4.2}$$

This stochastic process depends on permutation (shuffling) of the training data before fitting the model. It is conjectured that 4.2 behaves and converges like 4.1 despite its simplified procedure.

Averaging this update over all possible choices of the training examples would restore the batch gradient descent algorithm. The online gradient descent simplification relies on the hope that the random noise introduced by this procedure will not perturbate the average behaviour of the algorithm. Significant empirical evidence substantiates this hope.

Since the stochastic algorithm does not need to remember which examples were visited during the previous iterations, it can process examples on the fly in a deployed system. In such a situation, the stochastic gradient descent directly optimizes the expected risk, since the examples are randomly drawn from the ground truth distribution.

Due to this fact *SGD* is prefectly applicable to the streaming environment we aim to develop in this project.

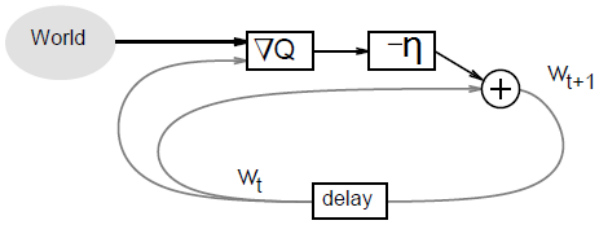


Figure 4.4: Online Gradient Descent. The parameters of the learning system are updated using information extracted from real world observations.

4.2.5 Scikit-Learn: *Machine Learning in Python*

As we were mainly interested in building on top of the algorithm, we considered using developed packages and libraries instead of writing the code from scratch. We selected *scikit-learn* [35] which is a Python module integrating classic machine learning algorithms in the tightly-knit scientific Python world (numpy, scipy, matplotlib). It aims to provide simple and efficient solutions to learning problems, accessible to everybody and reusable in various contexts: machine-learning as a versatile tool for science and engineering.

The SGD implementation of *scikit-learn* is influenced by the Stochastic Gradient SVM of Leon Bottou [36]. Similar to SvmSGD, the weight vector is represented as the product of a scalar and a vector which allows an efficient weight update in the case of L2 regularization. In the case of sparse feature vectors, the intercept is updated with a smaller learning rate (multiplied by 0.01) to account for the fact that it is updated more frequently. Training examples are picked up sequentially and the learning rate is lowered after each observed example. The learning rate schedule is adopted from Shalev-Shwartz et al. 2007 [37]. For multi-class classification, a "one versus all" approach is used. Scikit's implementation uses the truncated gradient algorithm proposed by Tsuruoka et al. 2009 for L1 regularization [38]

4.2.6 Mathematical formulation

This python implementation uses almost the same equations as the one presented in chapter 4.2.4 but with a few more additions.

As we explained before, given a set of training examples $(x_1, y_1), \ldots, (x_n, y_n)$ where $x_i \in \mathbf{R}^n$ and $y_i \in \{-1, 1\}$, our goal is to learn a linear scoring function $f(x) = w^T x + b$

with model parameters $w \in \mathbf{R}^m$ and intercept $b \in \mathbf{R}$. In order to make predictions, we simply look at the sign of $f(x)$. A common choice to find the model parameters is by minimizing the regularized training error given by:

$$E(w, b) = \sum_{i=1}^{n} L(y_i, f(x_i)) + \alpha R(w)$$

where L is a loss function that measures model (mis)fit and R is a regularization term (aka penalty) that penalizes model complexity; $\alpha > 0$ is a non-negative hyperparameter.

Stochastic gradient descent is an optimization method for unconstrained optimization problems. In contrast to batch gradient descent, SGD approximates the true gradient of $E(w, b)$ by considering a single training example at a time.

The python class we use implements a first-order SGD learning routine. The algorithm iterates over the training examples and for each example updates the model parameters according to the update rule given by

$$w_{t+1} = w_t - \eta(\alpha \frac{\partial R(w)}{\partial w} + \frac{\partial L(w^T x_i + b, y_i)}{\partial w})$$

where η is the learning rate which controls the step-size in the parameter space. The intercept b is updated similarly but without regularization.

The learning rate η can be either constant or gradually decaying.

For classification, the default learning rate schedule is given by

$$\eta^{(t)} = \frac{1}{\alpha(t_0 + t)}$$

where t is the time step [1], t_0 is determined based on a heuristic proposed by Leon Bottou such that the expected initial updates are comparable with the expected size of the weights [2].

For regression, the default learning rate schedule, inverse scaling is given by

$$\eta^{(t)} = \frac{eta_0}{t^{power\_t}}$$

where eta_0 and $power\_t$ are hyperparameters choosen by the user. We can still use eta_0 as a constant learning rate but it is not prefered.

[1]there are a total of $n_{samples} * epochs$ time steps

[2]this assuming that the norm of the training samples is approximately 1

When the model trains and fits the data, its parameters can be accessed through the members *coef* and *intercept*. These two values are of crucial importance as later on we discuss how we use them to give a ”warm start” to consecutive learning procedures.

4.2.7 Regularization Term

By far the most popular loss functions used for regression problems are based on Least Squares estimation, as explained in [38], [39]. They are alternately referred to as the minimizer of the residual sum of squared errors (RSS). The two most popular least squares choices for the regularization term R are:

- ***L1 norm:***

$$R(w) := \sum_{i=1}^{n} |w_i|$$

- ***L2 norm:***

$$R(w) := \frac{1}{2}\sum_{i=1}^{n} w_i^2,$$

Elastic Net:
A convex combination of L1 and L2 is called Elastic Net [40]

$$R(w) := \rho\frac{1}{2}\sum_{i=1}^{n} w_i^2 + (1-\rho)\sum_{i=1}^{n} |w_i|$$

The Elastic Net solves some deficiencies of the L1 penalty in the presence of highly correlated attributes. The parameter ρ has to be specified by the user.

4.2.8 Loss Function

The selection of the appropriate loss function depends critically on the application. It is a well investigated subject with many variations. There are several loss functions like:

- ***Hinge:*** (soft-margin) linear Support Vector Machine,
- ***Modified Huber:***[1] smoothed hinge loss,

[1]Note that, in principle, since they allow to create a probability model, log and modified huber, are more suitable for one-vs-all classification.

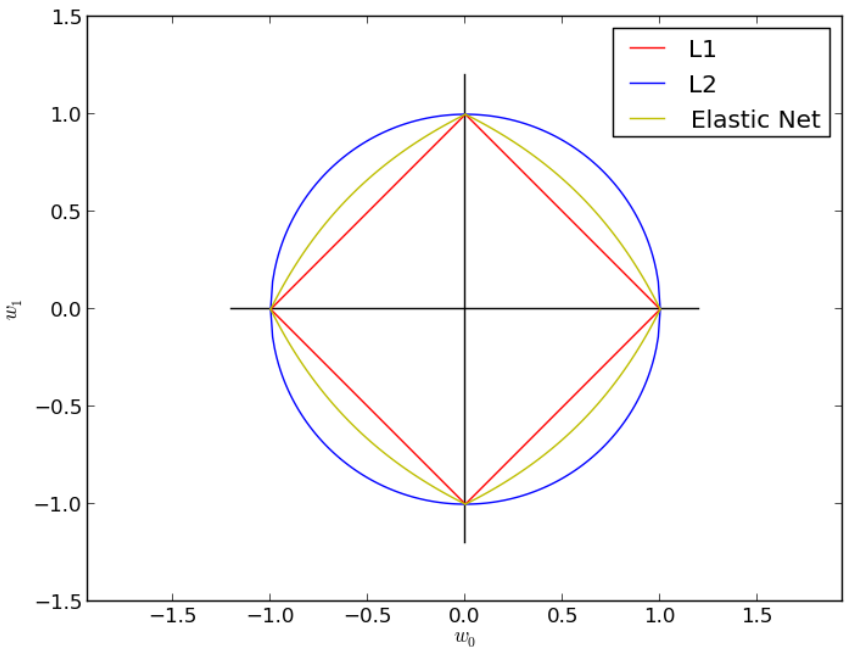


Figure 4.5: Contours of the three different regularization terms when R(w) = 1.

- ***Log:***[1] Logistic Regression,
- ***Squared Loss:***[1] Ordinary least squares,
- ***Huber:***[1] Huber loss for robust regression,
- ***Epsilon Insensitive:*** Linear Support Vector Regression,

The last three loss functions were the ones we mainly used to conduct our experiments in the test phase.

4.2.9 Complexity

The major advantage of SGD is its efficiency, which is basically linear in the number of training examples. If X is a matrix of size (n, p) training has a cost of $O(kn\bar{p})$, where k

[1]The Huber and epsilon-insensitive loss functions can be used for robust regression.

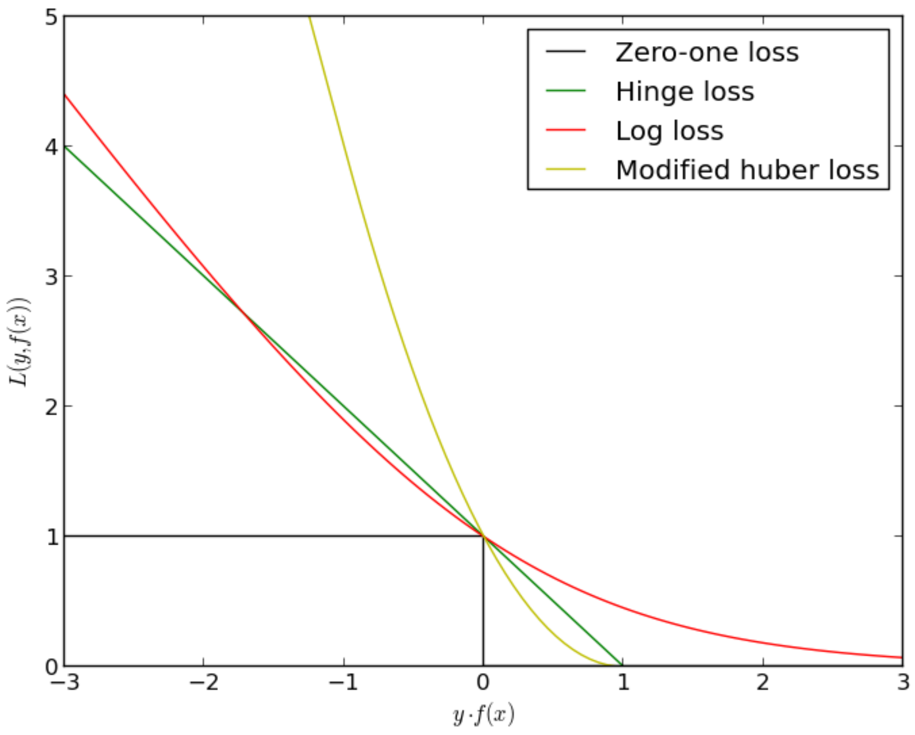


Figure 4.6: All of the above loss functions can be regarded as an upper bound on the misclassification error (Zero-one loss).

is the number of iterations (*epochs*) and $\bar{p}$ is the average number of non-zero attributes per sample.

Recent theoretical results, however, show that the runtime to get some desired optimization accuracy does not increase as the training set size increases, a fact that we exploited when designing our algorithm so as to properly select the training's set size.

4.2.10 Other Supervised Machine Learning Techniques

There many algorithms, and even more variations, when it comes to machine learning. Some state of the art techniques are among others: Decision trees, SVMs, Logistic Model Trees, Naive Bayes, etc. For the sake of simplicity we will not refer to them even further as there is a substantial extant research.

Chapter 5

Predicting The Future Of Financial Data In Real Time

5.1 Stock Prediction based on Textual Analysis of Financial News Articles

Our research initiated the effort to correlate and possibly combine the data we can obtain from two fields: the media and the stock markets.

Nowadays, a huge amount of valuable information related to the financial markets is available on the web. The majority of this information comes from Financial News Articles, Company Reports and Expert Recommendations (Blogs from valid sources can also act as a source of information) Most of this data is in a textual format as opposed to a numerical format which makes it hard to use. Thus the problem domain can now be viewed as one that involves Mining of Text Documents and Time Series Analysis concurrently.

As far as the news media part is concerned we aimed to exploit the knowledge we can obtain from social media, blogs, news feeds and all sorts of documents that can shape common public's opinion. In addition, historical stock market data (as NASDAQ quotes), are provided by Yahoo! Finance and other major financial sites. This whole approach was aimed at finding correlations between the price movements and the appearance of certain keywords in the documents. From this we could infer, with statistical proof and experiments, that the stock market movement is biased to the news and the direction of the public's view. Although this may be the instigation of interesting conclusions, the fact that the keyword extraction is very noisy was a major problem. Searching for top-k keywords, or at least a bag-of-words, in a vast amount of data may flatten our belief on what is important or not. There is interesting research in this area, especially by Christidis et. al. in [41] and Bollen et. al. [42], [43], [44].

Specifically Bollen et al. didn't just rely on text mining of what's written in social media but used those sources to determine a positive or negative emotion. They used this information later to discover correlations between the emotional changes of the public common and the stock market's movement. It was a very clever perspective and the findings very promising.

5.2 Our Approach

On the contrary, as we wanted to study financial time series under the prism of streaming analysis, and thus we followed a different path.

Empirical research in macroeconomics, as well as in financial economics, is largely based on time series. It has always been standard to view economic time series as realizations of stochastic processes. This approach allows the model builder to use statistical inference in constructing and testing equations that characterize relationships between economic variables.

Some of the most important reasons sets of high frequency data become available to researchers are based on the following factors:

- The low cost of data collection at the present,
- Wider and wider use of electronic technology in the financial markets,
- Increased ability and capacity to manage and analyse very large datasets.

The NYSE is the most extensively studied financial market, but its particular characteristic makes it difficult to generalize the results to other markets. In fact, the NYSE is essentially a hybrid market, combining batch and continuous trading, a dealing floor and an upper mechanism for arranging block traders, a limit order book and a designated monopoly specialist. These particular features do not allow the generalization of empirical findings on the NYSE and therefore new research is needed.

The skeleton of this algorithmic framework is based on machine learning, and specifically on stochastic gradient descent. The salient alteration we try to realize is the incorporation state of the art machine learning techniques in an on-line streaming context. To the best of our knowledge this is the first attempt at an online machine learning system.

5.3 LearnStream

In this project we propose a different use of the stochastic gradient descent algorithm. We incorporate the idea of the sliding window from Stastream, as well as from [45], [46] and endeavour to create a platform capable of both stream processing and machine learning.

Thy name, LearnStream.

5.3.1 Sliding Window Stochastic Gradient Descent

To evaluate a classifier's performance, given by a machine learning scheme, either a special testing set or a cross validation technique may be employed. A test set contains pre-classified examples different from those in the training set, and is used only for evaluation, not for training.

If data are scarce, it is sensible to use cross validation in order not to waste any data, which could be useful in enhancing classifier performance; all data are used both for training the classifier and for testing its performance.

More examples do not necessarily mean better classifier performance. Even though the classifier becomes better in the training set it could actually perform worse on the test data. This is due to the over fitting of the classifier transfer function, so that it fits too tightly to the training data and the border between classes is jagged rather than smooth, unlike how it should usually be.

But why should over fitting necessarily be a problem? In the way we target our approach we exploit over fitting in order to get more accurate results.

As we explained in section 4.2, as every other classifiers, SGD has to be fitted with two arrays: an array X of size $[n_{samples}, n_{features}]$ holding the training samples, and an array y of size $n_{samples}$ holding the target values (class labels) for the training samples.

We train our regressor over a sliding window $[t_i, t_{i+sw}]$ and then use the linear model obtained to predict the value for t_{i+sw+1}. An important note is that in order for our predictions to have meaning we need to use the data up to time point t to predict values for $t + 1$.

For example, we monitor the ticks of the prices for three stocks $A[t_0, t_{sw}]$, $B[t_0, t_{sw}]$ and $C[t_0, t_{sw}]$ and we want to make predictions for the price of $C[t_{sw+1}]$. How do we do connect the values at t_{sw+1} with those in $[t_0, t_{sw}]$?.

The simple solution we came up with was to wait one time point in the beginning of the algorithm in order to comprehend what price we aim to predict and then use those "future" values as the target dataset. In other words all the stocks in the X will start from time point t_0... and the target stock in y from $t_1 ... t_n$. This way we can train the model based on the movement between the present and the future rates so as to make more accurate predicitions.

5.3.2 Iterative Tuning Of Sliding Window

It's palpable though that we expect different outcomes with different window sizes. If the algorithm trains 50 time points the linear model will be different from when it trains 5000.

We don't know apriori what sliding window will be capable of predicting the smallest error. Therefore an iterative tuning process is performed off-line to determine what would be a suitable interval for this particular dataset.

In particular, when we have a plethora of data at our disposal, we execute parallel versions of our algorithm with various window sizes. For each one we calculate the error of the so prediction (in terms of relativity to the correct beliefs) and then we pick the sliding window which minimizes this error. Our assumption is that for a specific dataset the variance of its values will not dramatically change in the future so the optimal window up to the point we examine, will be approximately the optimal window for that application.

5.3.3 Beyond Simple SGD

The reason stochastic gradient descent has not been utilized for online applications is that the average case needs a large amount of training samples in order to converge and, as a result, the prediction stage would have to wait indefinitely.

In our approach we attempted to differ even more from the norm and research innovative ways to attack this problem. To that end we created a hybrid algorithm of SGD with randomly selecting which samples of the dataset would be candidate for training the model.

For example if SGD was to train a sliding window of the following values: $1, 2, 3, 4, 5$ our algorithm randomly forms a new dataset which may have this form $3, 1, 4, 5, 5$ [1]

5.3.4 Exponential Random Picking

The random selection is based on an exponential distribution which selects values closer to more current time points, with higher probability. Figure 5.1 shows the probability of every time point in a sliding window, to be a candidate for the the new training dataset.

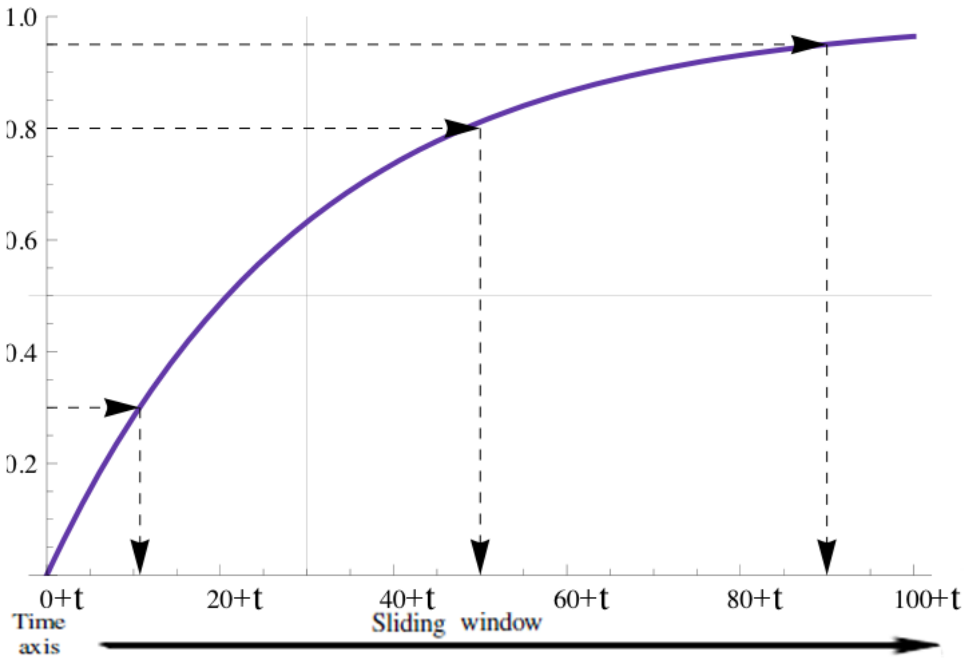


Figure 5.1: Exponential Selection.

It is easily understood that prior time points have less chance of being selected than more recent ones.

[1] we allow the use of selecting the same value more than once

This idea was nurtured by the fact that economists are keener on the present stock prices than on the historical prices. They do not want to forget the past entirely but they aim to place emphasis on the present as the balances may have changed concerning old statistics.

Although it may not be exactly rational to voluntarily discard random samples, we will see in later chapters that this modification was the reason for compelling performance comparing to the studied stochastic gradient descent algorithm, until present.

5.3.5 Warm Start

We may wish to focus more on recent data but that does not mean we should forget about the past. Nevertheless, how can we find the golden measure as to what is important to remember and what is not?

We tried several different approaches but none of them had the desired results. Based on the warm start idea of [47], we thought of initializing the $coef$ and $intercept$ weights, described in chapter 4.2.6, of each SGD's execution to a value different than zero (which is the normal thing for a different run of the algorithm).

Specifically we used the coefficients produced by the last execution in the previous sliding window. By doing this we affect the algorithm's initialization point and seemingly this can be either good or bad, as it can help us find the global minimum of the cost function or can lead us to miscalculation. As one may imagine though there are not any coefficients stored, in the very first execution, in which we just apply normal stochastic gradient descent.

Unexpectedly this method works surprisingly well; even better than the plain SGD and even more effectively than the approximate approach without the warm start. Additionally, it is a logical approach from the point of view that our data is sequential, hence, the warm start would possibly give a good head-start in contrast to the initialization from zero, where no further information is utilized.

5.3.6 Learning Rate

Despite the abundance of methods for learning from examples, there are only a few that can be used effectively for online learning. Classic batch training algorithms cannot straightforwardly handle non-stationary data. As described in [48],there exists the problem of ”catastrophic interference”, in which training with new examples interferes excessively with previously learned examples, leading to saturation and slow convergence. Variants of stochastic gradient descent methods are proposed in [49], which mainly implemented an adaptive learning rate for achieving online learning.

Although an adaptive learning rate is a good idea in many theoretical fields (for example game theory), there was no point in including it in our implementation as the sliding window approach we follow practically eliminates the need for a modification in the learning rate. The sliding window approach alters the very outcome of the results by itself and therefore a fixed *alpha* value can be sufficient to give us adequate results.

Chapter 6

Experimental Results

In this section, we appraise the experimental results of our work with real data and discuss the purity and the associated attributes of the algorithms we studied.

The way in which we conducted our experiments was by contrasting the three versions of Stochastic Gradient Descent, formulated by the ideas explained in the chapter 5.2:

- Plain Stochastic Gradient Descent.
  From here on we will refer to it as *plainSGD*

- Stochastic Gradient Descent with random sample picking (explained in chapter 5.3.4).
  From here on we will refer to it as *approxSGD*

- Stochastic Gradient Descent with random sample picking and warm start (explained in chapter 5.3.5).
  From here on we will refer to it as *approxWarmSGD*

6.1 Datasets

The experiments were mainly conducted by using three datasets. Two proprietary Forex datasets, with which we tested our trading strategy as well as one publicly available household electric power consumption dataset.

6.1.1 FOREX Data Sets

Our primary dataset was obtained by Capital K Partners, thanks to Timir Karia. It is composed of 28 different currency pairs with ticks aggregated every 100ms. There are in total 80 days of data, starting from 01/05/2012 until 09/08/2012. Every pair consists of 42 features (such as the best bid/ask price, best bid/ask volume, mid prices, bid/ask range, bid/ask slope etc.).

For the sake of simplicity we only used the bid/ask prices (they are described in section 4.1.6). The procedure we followed was to use all the pairs associated with EUR currency (i.e. EURCAD, EURAUD, EURGBP) to predict the price of EURUSD pair. In other words we form the training matrix X by using the prices from all the currency pairs and the target array y, by using the prices from the EURUSD pair, in a manner like that described in chapter 5.3.1.

The problem is that we could not predict both bid and ask prices simultaneously as they derive from different attributes[1]. There are two separate instances of our algorithm running in parallel: one associated with the bid price and one with the ask price. In this way we can combine the predictions and create a buying/selling signal for our trading system.

The secondary dataset on Forex data was obtained by a hedge fund from Singapoore. Its far more simplistic and it contains ticks aggregated every 5 minutes for a time period of around three months. Its only logical that 5 minutes is a large interval in which we omit a lot of information and we do not expect the consistent results generated by the first.

6.1.2 Individual Household Electric Power Consumption Data Set [1]

This dataset contains measurements of electric power consumption in one household with a one-minute sampling rate over a period of almost 4 years. Different electrical quantities and some sub-metering values are available. This dataset consist of roughly 2070000 values for 4 years (16/12/2006 - 26/11/2010).

Its features are:

[1]we predicted the $mid\,price = \frac{bid+ask}{2}$ but we couldn't use it effectively in our trading strategy later on.

1. ***Date:*** Date in format dd/mm/yyyy.
2. ***Time:*** time in format hh:mm:ss.
3. ***Global Active Power:*** Household global minute-averaged active power (in kilowatt).
4. ***Global Reactive Power:*** Household global minute-averaged reactive power (in kilowatt).
5. ***Voltage:*** Minute-averaged voltage (in volt).
6. ***Global Intensity:*** household global minute-averaged current intensity (in ampere).
7. ***Sub Metering 1:*** Energy sub-metering No. 1 (in watt-hour of active energy). It corresponds to the kitchen, containing mainly a dishwasher, an oven and a microwave (hot plates are not electric but gas powered).
8. ***Sub Metering 2:*** Energy sub-metering No. 2 (in watt-hour of active energy). It corresponds to the laundry room, containing a washing-machine, a tumble-drier, a refrigerator and a light.
9. ***Sub Metering 3:*** Energy sub-metering No. 3 (in watt-hour of active energy). It corresponds to an electric water-heater and an air-conditioner.

The reason we used a dataset so unrelated to the other two, which have an obvious financial similarity, was to cross check the predictive accuracy of our algorithm and aver that this work can be more than just a financial analysis.

We use attributes 1,2,4,5,6,7,8,9 to predict the global active power of the house for the next time-stamp, something which may help us optimize our electricity usage. The method is almost the same as the one we followed for the financial datasets but somewhat simpler, as we deal with only one predictor.

6.2 Metrics

In this section, we appraise the error metrics we used in order to compare the performance of the algorithms we tested.

6.2.0.1 Absolute/Relative Error

Experimental and measurement errors always create uncertainty in the final data. No measurement of a physical quantity can be entirely accurate. This problem can be solved by introducing the rules of significant figures. It is important to know, therefore, just how much the measured value is likely to deviate from the unknown, true, value of the quantity.

The *absolute error* in a measured quantity is the uncertainty in the quantity and has the same units as the quantity itself.

For example if you predict the price of a stock to be 1.3223$±0.00002$, the 0.00002$ is an absolute error.

The relative error (also called the fractional error) is obtained by dividing the absolute error in the quantity by the quantity itself.

$$relative\,error = \frac{absolute\,error}{value\,of\,thing\,measured}$$

or in terms common to Error Propagation

$$relative\,error = \frac{\Delta x}{x}$$

, where x is any variable.

The relative error is usually more significant than the absolute error. For example a \$1 error in the price of a cheap stock is probably more serious than a \$1 error in the quote of an expensive one. Relative errors are dimensionless. When reporting relative errors, it is usual to multiply the fractional error by 100 and report it as a percentage.

For example, two economists, Mr. John and Mr. Doe, predict the price of Aegis Value(AVALX) and Apple(AAPL) stocks. Mr. John predicts the price of $AVALX$ to be 11.23 ± 3$ and Mr. Doe the price of $AAPL$ to be 454.27 ± 3$.
Clearly, the overall accuracy of economist Mr. Doe's is much better than that of Mr. John's. The comparative accuracy of these measurements can be determined by looking at their relative errors. In our example:

$$relative\,error_{AVAX} = \frac{3\$}{11.23\$} \times 100 = 26.72\%$$

$$relative\ error_{AAPL} = \frac{3\$}{454.27\$} \times 100 = 0.66\%$$

Obivously, the relative error in the Apple's price is considerably smaller than the relative error in the Aegis's price, even though the amount of absolute error is the same in each case.

Relative Error was our baseline method to determine whether or not our predictions were meaningful. Scikit-learn[35] implements three more metrics for regression, which we used to further analyse our estimator and are presented below.

6.2.0.2 Mean Squared Error

In statistics, the mean squared error (MSE) of an estimator is one of many ways to quantify the difference between values implied by an estimator and the true values of the quantity being estimated. The mean squared error is perhaps the most important of the characterizations for an estimator. It captures the error that the estimator makes. MSE is a risk function, corresponding to the expected value of the squared error loss or quadratic loss. MSE measures the average of the squares of the "errors." The error is the amount by which the value implied by the estimator differs from the quantity to be estimated.

If $\hat{y}_i$ is the predicted value of the *i-th* sample and y_i is the corresponding true value, then the mean squared error (MSE) estimated over $n_{samples}$ is defined as:

$$MSE(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} (y_i - \hat{y}_i)^2$$

The best value is 0.0, higher values are worse.

6.2.0.3 Explained Variance Score

Explained variation measures the proportion to which a mathematical model accounts for the variation (dispersion) of a given data set. Often, variation is quantified as variance; then, the more specific term explained variance can be used.

If $\hat{y}$ is the estimated target output and y is the corresponding (correct) target output, then the explained variance is estimated as follow:

$$Explained\,Variance(y, \hat{y}) = 1 - \frac{Var\{y - \hat{y}\}}{Var\{y\}}$$

The best possible score is 1.0, lower values are worse.

6.2.0.4 R^2 Score, the coefficient of determination

The coefficient of determination, denoted R2, is used in the context of statistical models whose main purpose is the prediction of future outcomes on the basis of other related information.

R2 is most often seen as a number between 0 and 1.0, used to describe how well a regression line fits a set of data. An R2 near 1.0 indicates that a regression line fits the data well, while an R2 closer to 0 indicates a regression line does not fit the data very well. It is the proportion of variability in a data set that is accounted for by the statistical model. It provides a measure of how well *future outcomes* are likely to be predicted by the model.

If $\hat{y}_i$ is the predicted value of the *i-th* sample and y_i is the corresponding true value, then the score R^2 estimated over $n_{samples}$ is defined as:

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=0}^{n_{samples}-1} (y_i - \hat{y}_i)^2}{\sum_{i=0}^{n_{samples}-1} (y_i - \bar{y})^2}$$

where $\bar{y} = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} y_i$.

Best possible score is 1.0, lower values are worse.

6.2.0.5 Precision/Recall

$$Precision = \frac{true\,positive}{true\,positive + false\,positive} \times 100$$

and

$$Recall = \frac{true\,positive}{true\,positive + false\,negative} \times 100$$

are two well known evaluation methods but unfortunately could not apply in the task of regression. As mentioned above, we are trying to make predictions about the bid/ask price of a currency pair. Even if our forecast is extremely accurate we wouldn't be able to guess the exact price as the decimal points extend even to ten digits. As of this, precision and recall would be always zero.

6.3 Parameters Tuning

As surmised from the previous chapters, we have 5 distinct parameters in our algorithm. In order to determine the appropriate value for every attribute and for every dataset, we isolated them and through permutations, we modified only the parameter under consideration, so as to attain the optimal solution.

- ***Learning Rate*** (α)5.3.6 $\in R$
  Learning rate can be any real value thus it is not feasible to test every combination. We narrowed the problem down to determine how many decimal points will α have. As a matter of fact, we came to the realization that a value of two zeros after decimal point performs much better than a larger or a smaller learning rate. To that end we set $\alpha = 0.0015$

- ***Regularization Term***4.2.7 $\in \{L1,\ L2,\ Elastic\ Net\}$
  $L1$ and $L2$ have almost the same results but for most of the cases $L2$ delivers slightly more accuracy.

- ***Loss Function***4.2.8 $\in \{Squared\ Loss, Huber, Epsilon\ Insensitive\}$
  Huber Loss and Squared Loss were much better on the average than Epsilon Insensitive Loss. Huber was slightly better than Squared loss for the *approxWarmSGD* for every dataset, while for the other two algorithms it was mainly dependent on the dataset.

  In sum, the improvement or regression, in terms of relative error and the other methods stated, was sufficiently minor to avoid further examination.

- ***Sliding Window(SW)***5.3.1 $\in N$
  It is only logical that if you train more data then the results would be more accurate. Nevertheless, while the sliding window increases, time complexity increases as well, which is something we do not want as time execution is always important.
  Moreover, an optimal sliding window for all the datasets could not exist. All in all we need a value small enough, in order not to increase the time sgd needs to converge, as well as a value which will minimize the error based on each particular dataset.

The iterative tuning procedure followed in order to figure out the proper sw is explained in section 5.3.2.
The results we obtained helped us set the

$$SW_{capital\ K\ Forex\ Dataset} = 72$$

$$SW_{singapore\ Forex\ Dataset} = 100$$

$$SW_{Electrical\ Consumption\ Dataset} = 155$$

- ***Approximation Time Points***5.3.4 $\in SW_{size}$
  This is probably the most interesting feature of our approximation algorithm. If the sliding window, on which plainSGD uses all the samples to train the model, is SW then let the size of the dataset used from *approxSGD* and *approxWarmSGD* be *approxSW*.

  Again neither in this case can we define a global standard for how many random samples of those extant in the normal sliding window we should use. It seems though that if we set $Size_{approxSW} = Size_{SW} - 1$ we produce a prediction with the smallest relative and mean squared error. We contrast this approach with setting $Size_{approxSW} = Size_{SW}/2$, in order to see how our algorithm behaves when we decrease the number of time points examined.

6.4 Experiments

6.4.1 Electric Power Consumption Data Set

In the table below you see the results of the electric power consumption in terms of the metrics referred above. We present results both for $SW - 1$ and $\frac{SW}{2}$ approximation points.

Table 6.1: Electrical Consumption Data *SW-1 Approx. Points*

Metric	**plainSGD**	**approxSGD**	**approxWarmSGD**
Average Relative Error	46.5268326356%	35.1390031025%	12.3209254031%

Continued on next page

Table 6.1 – *Continued from previous page*

Metric	plainSGD	approxSGD	approxWarmSGD
Mean Squared Error	1.20014251329	0.836286330532	0.15540327753
Explained Variance	0.556287783866	0.654150947989	0.913688374021
R^2 Score	0.333425379143	0.535515792892	0.913687016623

Table 6.2: Electrical Consumption Data *SW/2 Approx. Points*

Metric	plainSGD	approxSGD	approxWarmSGD
Average Relative Error	46.5268326356%	43.9709146395%	12.3961653422%
Mean Squared Error	1.20014251329	1.33460563753	0.154441109978
Explained Variance	0.556287783866	0.550474129003	0.914222866339
R^2 Score	0.333425379143	0.258742826809	0.914221416883

We can see that our method (*approxWarmSGD*) is better than the other two in all four metrics. The remarkable thing though is that even though we decrease the number of time points examined by half, there is almost no loss in the accuracy of the results.

6.4.2 Singapore Hedge Fund FOREX Data Set

Likewise, we present the tables for the Singapore FOREX Data set. The error metrics were almost the same for both bid and task predictions so we show only one of them

Table 6.3: Singapoore FOREX Data *SW-1 Approx. Points*

Metric	plainSGD	approxSGD	approxWarmSGD
	Bid / Ask	Bid / Ask	Bid / Ask
Average Relative Error	0.1366274551%	0.133887583%	0.2294997927%
Mean Squared Error	5.79361776e-06	5.552743418e-06	1.547903339e-05
Explained Variance	0.9990655232	0.9991046995	0.9974962369
R^2 Score	0.9990535129	0.9990928638	0.9974712337

Table 6.4: Singapoore FOREX Data *SW/2 Approx. Points*

Metric	**plainSGD**	**approxSGD**	**approxWarmSGD**
	Bid / Ask	Bid / Ask	Bid / Ask
Average Relative Error	0.1366274551%	0.09281447603%	0.1551851906%
Mean Squared Error	5.79361778e-06	3.045854724e-06	7.963935037e-06
Explained Variance	0.9990655232	0.9995082637	0.9987048952
R^2 Score	0.9990535129	0.9995024072	0.9986989542

As we can see for this particular dataset *approxSGD* without the warm start performs better. This is probably caused because the values are infrequent (ticks every 5 minutes).

In other words imagine the following example: we pick a point somewhere in the globe and then we start to turn the globe. Then we monitor only a specific frame(like a sliding window) and not the whole globe in order to determine our position every time. If we turn the globe slowly, the knowledge of the last point observed could help us monitor more efficiently where we will end up. In contrast, if we turn the globe fast, the perception of the last point observed will probably confuse our belief as the changes are so vast we could not use this information efficiently.

6.4.3 Capital K FOREX Data Set

This brings us to the cornerstone of our data sets, the one on which we based our theoretical study to produce a trading strategy that could potentially generate substantial profit.

As stated above, we have 80 days of data each one of which consists of 860 thousand ticks approximately. Each date is a separate dataset for which we have different results. In the table below you can see the average values from all the dates.

Table 6.5: Capital K Forex Data, *SW-1 Approx. Points*

Metric	**plainSGD**	**approxSGD**	**approxWarmSGD**
	Bid / Ask	Bid / Ask	Bid / Ask
Average Relative Error	0.01167%/0.0117%	0.0116%/0.0117%	1.502e-03%/1.554e-03%
Mean Squared Error	2.201e-08/2.223e-08	2.175e-08/2.191e-08	1.287e-09/1.372e-09
Explained Variance	0.999	0.999	0.999
R^2 Score	0.9812/0.9828	0.9814/0.9831	0.9989/0.9989

Table 6.6: Capital K Forex Data, *SW/2 Approx. Points*

Metric	**plainSGD**	**approxSGD**	**approxWarmSGD**
	Bid / Ask	Bid / Ask	Bid / Ask
Average Relative Error	0.01167%/0.0117%	0.0314%/0.0313%	1.701e-03%/1.751e-03%
Mean Squared Error	2.201e-08/2.223e-08	1.538e-07/1.517e-07	1.514e-09/1.602e-09
Explained Variance	0.999	0.999	0.998
R^2 Score	0.9812/0.9828	0.8687/0.8832	0.9987/0.9987

It is palpable that *approxWarmSGD* is superior in all metrics. The results even if we use $\frac{SW}{2}$ time points are approximately the same and the relative error is remarkably low (~0.0015%).

6.5 Trading Strategy

Most empirical studies with high-frequency data look at the time series of volatility, trading volume and spreads. Several researchers argue that all these time series follow a *U-shaped* or a *J-shaped* pattern, i.e. the highest point of these variables occurs at the opening of the trading day and they fall to lower levels during the midday period, and then rise again towards the close. The behaviour of these variables is not easy to explain theoretically using the basic models related to threefold types of agents: the informed trader, the non informed trader and the market maker.

The introduction of a distinction between discretionary and non-discretionary uninformed traders partially overcome this difficulty. If the uninformed or liquidity traders can discretionary choose the time of their trades, then they can congregate in the periods when trading costs are low.

Some other models go further in explaining the positive relation between volatility and spread, indicating that more volatility is associated with the revelation of more information, and thus the market becomes more uncertain and spreads widen [50], [51]

6.5.1 Spread Based Trading

In contrast to the above, the approach we followed was again a very basic one, aimed at showing that if we rely on the learning algorithm we can make a constant positive profit even if our trading technique does not revolve around macroeconomic theories and models.

Our strategy relies mostly on the spread of the bid and ask prices for consecutive time points. At this point we are not dealing with volatility; we only trade one unit of EUR/USD per time. It is obvious though that in the case that you trade one unit and get a profit of 0.002$, it is the same as if you trade 1 million units and get a profit of 2000$. Of course if you try to perform such a trade tracking volatility, you must check first that the amount of stocks you wish to buy is available.

Specifically we have the two following cases where we get a buy/sell signal.

- If $bid\_prediction_{t+1} > ask_t$ then buy at time t and sell at $t+1$
  If the bid price at time point $t+1$ is larger than the ask price at time t, it means that we will get a positive profit if we buy something that will be sold later for a higher price, than the price we bought it.

- If $ask\_prediction_{t+1} < bid_t$ then sell at time t (supposing you have 1 unit available) and rebuy it at $t+1$ to make things even.
  If the ask price at time point $t+1$ is smaller than the bid price at time t, it means that if we sell something now and rebuy it later we will gain a profit by exploiting this spread.

As is easily understood, though, if the prediction is not accurate we end up buying something that we have to sell for less or selling something and then need to buy it for a higher price. At this point we do not care much about this and even if we are about to lose money we still execute the transaction.

Table 6.7 shows the trading results for every day and the total profit in the end. The first column (*goldenMethodProfit*), is the maximum profit we could gain if we predicted the exact value for the next time point. We "hacked" the trading algorithm by reading the actual price for the next time point and deciding to buy, sell or do nothing. In this way we can never make a wrong decision and maximize our profit. Unfortunately, we can not perform this strategy in real life, yet..

Table 6.7: Capital K FOREX Dataset Trading Results

Date	**goldenMethodProfit**	**plainSGD**	**approxSGD**	**approxWarmSGD**
2012/05/01	0.0239371 $	-0.0342849 $	-0.0269991 $	-0.000753200000002 $
2012/05/02	0.0761673 $	-0.0291822 $	-0.0175658 $	0.00277210000001 $
2012/05/03	0.0340376 $	-0.0488826 $	-0.0392353 $	-0.0016023 $
2012/05/04	0.1588481 $	0.0274162 $	0.0511831 $	0.0477925 $
2012/05/06	0.616827 $	0.5445014 $	0.5684931 $	0.5750704 $
2012/05/07	0.0641958 $	0.0161211 $	0.0215621 $	0.00400079999999 $
2012/05/08	0.0405388 $	0.00381669999999 $	0.00674389999999 $	-0.0017237 $
2012/05/09	0.089591 $	0.0253449 $	0.0266849 $	0.0023354 $
2012/05/10	0.0623359000001 $	-0.0273292 $	-0.0186682 $	-0.0015336 $
2012/05/11	0.0568553 $	0.0016687 $	0.0058607 $	0.000805299999997 $
2012/05/13	0.00609079999999 $	-0.0541837 $	-0.0421192 $	-0.000129300000001 $
2012/05/14	0.0702268 $	0.0133073 $	0.0181479 $	-0.000316600000005 $
2012/05/15	0.221093 $	0.0479608 $	0.0784407 $	0.0801666 $
2012/05/16	0.0533726 $	-0.0497275 $	-0.0294674 $	-0.00666 $
2012/05/17	0.0476873 $	-0.0156991 $	-0.00569449999998 $	0.0122044 $
2012/05/18	0.0625968 $	-0.0339329 $	-0.0171566 $	-0.00485040000001 $
2012/05/20	0.000837800000001 $	-0.0306222 $	-0.0220222 $	-0.006 $
2012/05/21	0.0275625 $	0.0112561 $	0.0120186 $	-0.0011088 $
2012/05/22	0.0213184 $	-0.0146806 $	-0.0121239 $	-2.13000000002e^{-05} $
2012/05/25	0.0287465 $	-0.0380112 $	-0.0271716 $	-0.000464800000001 $
2012/05/27	0.00906550000004 $	-0.00280859999998 $	0.00326730000002 $	0.00204670000001 $
2012/05/28	0.0098993 $	-0.0252188 $	-0.0232469 $	-0.0007262 $
2012/05/29	0.0752841 $	-0.0209611 $	-0.00200590000001 $	0.0056653 $
2012/05/31	0.0482915 $	-0.0511841 $	-0.0312588 $	0.00608160000001 $
2012/06/01	0.3012095 $	0.0108270999999 $	0.0537842 $	0.0870463 $

Continued on next page

Table 6.7 – *Continued from previous page*

Date	goldenMethodProfit	plainSGD	approxSGD	approxWarmSGD
2012/06/03	0.0031913 $	-0.0326401 $	-0.0230999 $	0 $
2012/06/04	0.0302697 $	-0.00652450000001 $	-0.00154900000001 $	-0.000264900000003 $
2012/06/05	0.0337185 $	-0.0566453 $	-0.0463728 $	-0.0026809 $
2012/06/06	0.1704434 $	0.000312599999958 $	0.0245615 $	0.0551334 $
2012/06/12	0.0159152 $	-0.0018495 $	-0.0022344 $	-0.0003722 $
2012/06/13	0.0279101 $	-0.020317 $	-0.0135648 $	0.00335750000001 $
2012/06/14	0.0200004 $	-0.0257194 $	-0.0146346 $	-0.0013187 $
2012/06/15	0.0307023 $	-0.0091405 $	-0.0005642 $	0.000256499999997 $
2012/06/17	0.000726200000001 $	-0.015285 $	-0.007924 $	-0.00023 $
2012/06/18	0.0663894999999 $	-0.0067474 $	-0.0041927 $	-0.0027532 $
2012/06/19	0.0407958 $	-0.0320735 $	-0.0163544 $	-0.0030473 $
2012/06/20	0.44752 $	0.1504543 $	0.1960734 $	0.3364993 $
2012/06/21	0.0622062999999 $	-0.0545687000001 $	-0.0378072000001 $	0.0123713999999 $
2012/06/22	0.0141182 $	-0.0321446 $	-0.0313011 $	-0.0006107 $
2012/06/24	$7.34999999994e^{-05}$ $	-0.4442654 $	-0.3293038 $	-0.04096 $
2012/06/25	0.0091785 $	-0.0430264 $	-0.0358235 $	0 $
2012/06/26	0.0389425 $	-0.0314376 $	-0.0156091 $	0.00335050000001 $
2012/06/27	0.0181894 $	-0.0221411 $	-0.0136355 $	0.0021179 $
2012/06/28	0.0455772 $	-0.0557239 $	-0.0366159 $	-0.0013171 $
2012/06/29	0.0906558 $	-0.0145692 $	0.0014315 $	0.0019478 $
2012/07/01	0.000680899999999 $	-0.000330600000002 $	-0.000240600000001 $	0 $
2012/07/02	0.0775112000001 $	0.0452053000001 $	0.0510538000001 $	0.0553430000001 $
2012/07/03	0.0058701 $	-0.0830429 $	-0.0662699 $	0 $
2012/07/04	0.00817519999999 $	-0.0146892 $	-0.0096125 $	$-6.61000000006e^{-05}$ $
2012/07/05	0.0513885 $	-0.0822023 $	-0.059481 $	-0.0063864 $
2012/07/06	0.0273257 $	-0.0662285 $	-0.0511703 $	-0.0009453 $
2012/07/08	0.0020017 $	0 $	0 $	0 $
2012/07/09	0.00954100000001 $	-0.0520138 $	-0.043469 $	-0.0004271 $
2012/07/10	0.00594 $	-0.00046 $	-0.0007281 $	-0.0002872 $
2012/07/11	0.0164148 $	-0.0272327 $	-0.0189462 $	-0.0023874 $
2012/07/12	0.0290472 $	-0.00936000000001 $	-0.000822500000012 $	0.00806419999999 $
2012/07/13	0.0067269 $	-0.0519178 $	-0.0361188 $	-0.0003176 $
2012/07/15	$1.91e^{-05}$ $	-0.0423109 $	-0.02321 $	0 $
2012/07/16	0.0119677 $	-0.027802 $	-0.0227166 $	-0.0005198 $
2012/07/17	0.0287962 $	-0.0681972 $	-0.044853 $	0.000664999999996 $
2012/07/18	0.0110806 $	-0.0069917 $	-0.0040249 $	-0.0001258 $
2012/07/19	0.028801 $	-0.062382 $	-0.0365526 $	0.0017298 $

Continued on next page

Table 6.7 – *Continued from previous page*

Date	goldenMethodProfit	plainSGD	approxSGD	approxWarmSGD
2012/07/20	0.0542126 $	-0.0070123 $	0.0121815 $	0.0173619 $
2012/07/22	0.000979899999999 $	-0.0967228 $	-0.0575202 $	0 $
2012/07/23	0.0198586 $	-0.0115944 $	-0.010476 $	-0.000934000000001 $
2012/07/24	0.0824358999999 $	-0.0155094000001 $	0.00836189999994 $	0.0312996 $
2012/07/25	0.1288265 $	0.0223879 $	0.0381827 $	0.0423119 $
2012/07/26	0.0470796 $	-0.0193145 $	-0.0121306 $	-0.0037587 $
2012/07/27	0.0880507 $	-0.00642770000001 $	0.00408049999999 $	-0.00492140000001 $
2012/07/29	$8.78999999996e^{-05}$ $	-0.07629 $	-0.05687 $	-0.031028 $
2012/07/30	0.00886399999999 $	-0.0069527 $	-0.0040943 $	-0.0004847 $
2012/07/31	0.0392846 $	-0.0117898 $	0.0002319 $	0.0014711 $
2012/08/01	0.0635613 $	-0.0692096 $	-0.0506329 $	-0.0046502 $
2012/08/02	0.8581657 $	0.4955924 $	0.5432271 $	0.4788215 $
2012/08/03	0.0473335 $	-0.059746 $	-0.0366579 $	0.0026634 $
2012/08/05	0.0032218 $	-0.0559779 $	-0.0324349 $	-0.0001001 $
2012/08/06	0.0125088 $	-0.0398368 $	-0.0284713 $	-0.0005952 $
2012/08/07	0.0070426 $	-0.1341935 $	-0.1111658 $	0 $
2012/08/08	0.011554 $	-0.0293867 $	-0.0184645 $	$-8.999999999e^{-05}$ $
2012/08/09	0.0214832 $	-0.0484038 $	-0.0326047 $	0.0014498 $
Total	5.1889791$	-1.2490645 $	-0,0894891 $	1.7545607 $

As we see approxWarmSGD is the only method that in the end has a positive profit. The astonishing thing is that even in the days when you “bleed” through loss the algorithm somehow manages to minimize it and restrain it at about $\frac{1}{10}$ of the loss of the other two methods.

One can see the above results graphically represented in Figure 6.1

For the Singapore FOREX Dataset the results were not so encouraging though, as you can see in table 6.8.

Table 6.8: Singapoore FOREX Dataset Trading Results

goldenMethodProfit	plainSGD	approxSGD	approxWarmSGD
11.810265$	-5.259669$	-5.209989$	-0.14704$

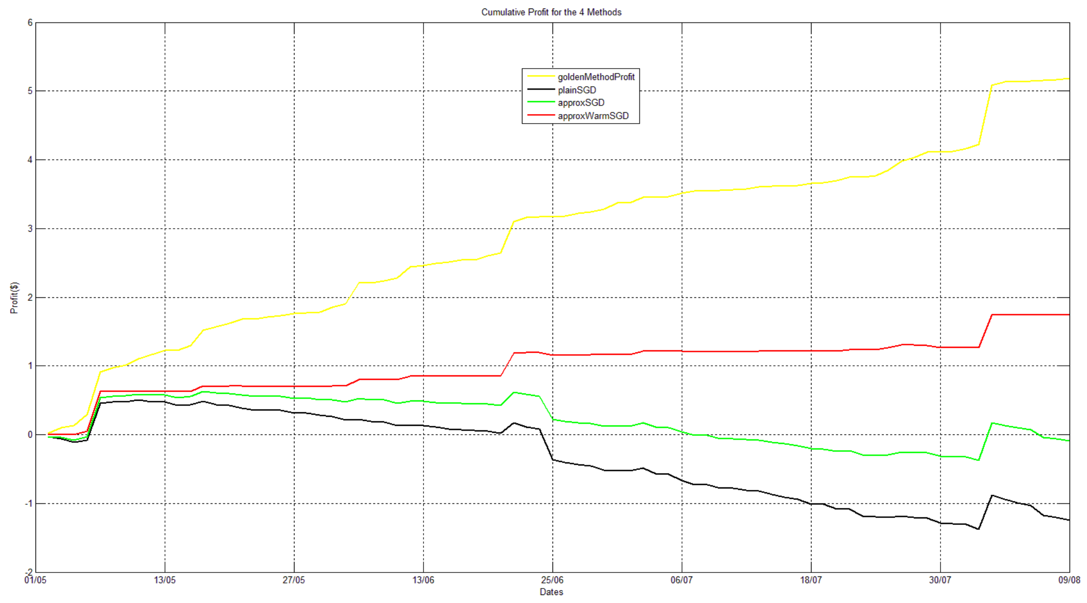


Figure 6.1: Profit graph for every trading day.

As stated in section 6.1.1 and 1, it is very difficult to capture the essence of the currency movement by using ticks aggregated every 5 minutes. At least, we can state that our algorithm could sustains a loss 37,14 times less than the other two algorithms.

6.5.2 Other Methods We Tried

We thought of other techniques as well that take into account long term trading. Those methods though need further examination as problems could occur for instance in the case that we do not sell if the prediction is not right and just hold the amount purchased until we find a more profitable choice, we might end up stacking up our inventory and keep losing money as we do not sell anything.

Although it is not easy to develop such a strategy, we believe that we can generate much better results if we design a method that tries to minimize loss and not blindly buys and sells.

Chapter 7

Conclusion

In this dissertation we examined fast algorithms for time series data streams and concentrated on appraising two problems: finding correlations in an online sliding window manor and predicting the future of financial time series based on an online approximation algorithm we created. We tested our results with data collected from the stock market and publicly available data sets.

7.1 StatLearn

Based on the knowledge we gained from Statstream 2.2 and Learnstream5.3, we present a new platform capable of finding highly correlated pairs in real time and then find linear relationships between them. Its name is **StatLearn**.

7.2 Future Work

The work in this thesis can be used as the basis for several future research directions, some of which are listed below.

7.2.0.1 Trading Strategy Improvement

The most important improvement would be to have an improved, sophisticated trading strategy which would take into account all the previous history and decide when to buy or sell not only based on the prediction but in another learning algorithm.

Outlier Detection

Except for finding correlations and linear relationships we would like to find outliers and other aberration detection in an online mode as well.

References

[1] Frank, A., Asuncion, A.: UCI machine learning repository. http://archive.ics.uci.edu/ml (2010) vii, 50

[2] Hey, T., Tansley, S., Tolle, K., eds.: The Fourth Paradigm: Data-Intensive Scientific Discovery. Microsoft Research, Redmond, Washington (2009) 3

[3] Babcock, B., Babu, S., Datar, M., Motwani, R., Widom, J.: Models and issues in data stream systems. In: Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems. PODS '02, New York, NY, USA, ACM (2002) 1–16 5

[4] Carney, D., Çetintemel, U., Cherniack, M., Convey, C., Lee, S., Seidman, G., Stonebraker, M., Tatbul, N., Zdonik, S.: Monitoring streams: a new class of data management applications. In: Proceedings of the 28th international conference on Very Large Data Bases. VLDB '02, VLDB Endowment (2002) 215–226 5

[5] Cai, Y.D., Clutter, D., Pape, G., Han, J., Welge, M., Auvil, L.: Maids: Mining alarming incidents from data streams 5

[6] Naughton, J., Dewitt, D., Maier, D., Aboulnaga, A., Chen, J., Galanis, L., Kang, J., Krishnamurthy, R., Luo, Q., Prakash, N., Ramamurthy, R., Shanmugasundaram, J., Tian, F., Tufte, K., Viglas, S.: The niagara internet query system. IEEE Data Engineering Bulletin **24** (2001) 27–33 5

[7] Chandrasekaran, S., Cooper, O., Deshpande, A., Franklin, M.J., Hellerstein, J.M., Hong, W., Krishnamurthy, S., Madden, S., Raman, V., Reiss, F., Shah, M.: Telegraphcq: Continuous dataflow processing for an uncertan world (2003) 5

[8] Shasha, D., Zhu, Y.: High Performance Discovery In Time Series: Techniques And Case Studies (Monographs in Computer Science). SpringerVerlag (2004) 5, 12

[9] Cole, R., Shasha, D., Zhao, X.: Fast window correlations over uncooperative time series. In: Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining. KDD '05, New York, NY, USA, ACM (2005) 743–749 6, 14

[10] Zhao, X.: High performance algorithms for multiple streaming time series. PhD thesis, New York, NY, USA (2006) AAI3205697. 6

[11] Zhu, Y., Shasha, D.: Statstream: statistical monitoring of thousands of data streams in real time. In: Proceedings of the 28th international conference on Very Large Data Bases. VLDB '02, VLDB Endowment (2002) 358–369 6, 14

[12] Shasha, D.: Statstream. `http://cs.nyu.edu/shasha/papers/statstream.html` (November 2005) 6, 8

[13] Agrawal, R., Faloutsos, C., Swami, A.N.: Efficient similarity search in sequence databases. In: Proceedings of the 4th International Conference on Foundations of Data Organization and Algorithms. FODO '93, London, UK, UK, Springer-Verlag (1993) 69–84 7

[14] Faloutsos, C., Ranganathan, M., Manolopoulos, Y.: Fast subsequence matching in time-series databases. Technical report, College Park, MD, USA (1993) 7

[15] Rafiei, D., Mendelzon, A.: Similarity-based queries for time series data. SIGMOD Rec. **26**(2) (June 1997) 13–25 7

[16] : Efficient time series matching by wavelets. In: Proceedings of the 15th International Conference on Data Engineering. ICDE '99, Washington, DC, USA, IEEE Computer Society (1999) 126– 7

[17] Gilbert, A.C., Kotidis, Y., Muthukrishnan, S., Strauss, M.J.: Surfing wavelets on streams: One-pass summaries for approximate aggregate queries. In: In VLDB. (2001) 79–88 7

[18] Korn, F., Jagadish, H.V., Faloutsos, C.: Efficiently supporting ad hoc queries in large datasets of time sequences. In: In SIGMOD. (1997) 289–300 7

[19] Keogh, E., Chakrabarti, K., Pazzani, M., Mehrotra, S.: Dimensionality reduction for fast similarity search in large time series databases. JOURNAL OF KNOWLEDGE AND INFORMATION SYSTEMS **3** (2000) 263–286 7

[20] Achlioptas, D.: Database-friendly random projections, ACM Press (2001) 274–281 8, 10

[21] Johnson, W.B., Lindenstrauss, J.: Extensions of lipschitz mappings into a hilbert space. (1982) 8

[22] Kushilevitz, E., Ostrovsky, R., Rabani, Y.: Efficient search for approximate nearest neighbor in high dimensional spaces. In: Proceedings of the thirtieth annual ACM symposium on Theory of computing. STOC '98, New York, NY, USA, ACM (1998) 614–623 10

[23] Indyk, P.: Stable distributions, pseudorandom generators, embeddings, and data stream computation. J. ACM **53**(3) (May 2006) 307–323 10

[24] Shasha, D.: Statstream pictorial architecture. `http://cs.nyu.edu/shasha/papers/statstream/architecture.html` (November 2005) 14

[25] : Kx language. `http://www.kx.com/index.php` 16

[26] : Financial web. `http://www.finweb.com/investing/forex-bid-and-ask-price.html` 31

[27] Bottou, L.: Stochastic learning. In Bousquet, O., von Luxburg, U., eds.: Advanced Lectures on Machine Learning. Lecture Notes in Artificial Intelligence, LNAI 3176. Springer Verlag, Berlin (2004) 146–168 32

[28] Bottou, L., LeCun, Y.: Large scale online learning. In Thrun, S., Saul, L., Schölkopf, B., eds.: Advances in Neural Information Processing Systems 16. MIT Press, Cambridge, MA (2004) 32

[29] Bottou, L.: Large-scale machine learning with stochastic gradient descent. In Lechevallier, Y., Saporta, G., eds.: Proceedings of the 19th International Conference on Computational Statistics (COMPSTAT'2010), Paris, France, Springer (August 2010) 177–187 32, 33

[30] Gasso, G., Pappaioannou, A., Spivak, M., Bottou, L.: Batch and online learning algorithms for nonconvex Neyman-Pearson classification. ACM Transaction on Intelligent System and Technologies **2**(3) (2011) 32

[31] Raiko, T., Valpola, H., LeCun, Y.: Deep learning made easier by linear transformations in perceptrons. Journal of Machine Learning Research - Proceedings Track **22** (2012) 924–932 32

[32] LeCun, Y., Haffner, P., Bottou, L., Bengio, Y.: Object recognition with gradient-based learning. In: Shape, Contour and Grouping in Computer Vision. (1999) 319– 32

[33] Rumelhart, D.E., McClelland, J.L., PDP Research Group, C., eds.: Parallel distributed processing: explorations in the microstructure of cognition, vol. 1: foundations. MIT Press, Cambridge, MA, USA (1986) 33

[34] Dennis, J., Schnabel, R.: Numerical methods for unconstrained optimization and nonlinear equations. Prentice-Hall series in computational mathematics 33

[35] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12** (2011) 2825–2830 35, 53

[36] Bottou, L.: Stochastic gradient descent (version 2). `http://leon.bottou.org/projects/sgd` (September 2012) 35

[37] Singer, Y., Srebro, N.: Pegasos: Primal estimated sub-gradient solver for svm. In: In ICML. (2007) 807–814 35

[38] Tsuruoka, Y., Tsujii, J., Ananiadou, S.: Stochastic gradient descent training for l1-regularized log-linear models with cumulative penalty. In: Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1 - Volume 1. ACL '09, Stroudsburg, PA, USA, Association for Computational Linguistics (2009) 477–485 35, 37

[39] Schmidt, M.: Least squares optimization with l1-norm regularization (2005) 37

[40] Zou, H., Hastie, T.: Regularization and variable selection via the elastic net. Journal of the Royal Statistical Society, Series B **67** (2005) 301–320 37

[41] Ruiz, E.J., Hristidis, V., Castillo, C., Gionis, A., Jaimes, A.: Correlating financial time series with micro-blogging activity. In: Proceedings of the fifth ACM international conference on Web search and data mining. WSDM '12, New York, NY, USA, ACM (2012) 513–522 42

[42] Bollen, J., Mao, H.: Twitter mood as a stock market predictor. IEEE Computer **44**(10) (2011) 91–94 42

[43] Bollen, J., Mao, H., Pepe, A.: Modeling public mood and emotion: Twitter sentiment and socio-economic phenomena. In: ICWSM. (2011) 42

[44] Bollen, J., Mao, H., Zeng, X.J.: Twitter mood predicts the stock market. CoRR **abs/1010.3003** (2010) 42

[45] Kontaki, M., Papadopoulos, A.N.: Efficient similarity search in streaming time sequences. In: Proceedings of the 16th International Conference on Scientific and Statistical Database Management. SSDBM '04, Washington, DC, USA, IEEE Computer Society (2004) 63– 43

[46] Buchgraber, T., Shutin, D., Poor, H.V.: A sliding-window online fast variational sparse bayesian learning algorithm. In: ICASSP. (2011) 2128–2131 43

[47] Velazco, M., Oliveira, A., Lyra, C.: Neural networks give a warm start to linear optimization problems. In: Neural Networks, 2002. IJCNN '02. Proceedings of the 2002 International Joint Conference on. Volume 2. (2002) 1871 –1876 46

[48] Sutton, R.S., Whitehead, S.D.: Online learning with random representations. In: In Proceedings of the Tenth International Conference on Machine Learning, Morgan Kaufmann (1993) 314–321 46

[49] Saad, D., ed.: On-line learning in neural networks. Cambridge University Press, New York, NY, USA (1998) 47

[50] Chang, C.C., Chen, S.S., Chou, R., Hsin, C.W.: Intraday return spillovers and its variations across trading sessions. Review of Quantitative Finance and Accounting **36** (2011) 355–390 60

[51] Lee, C.M.C., Mucklow, B., Ready, M.J.: Spreads, depths, and the impact of earnings information: An intraday analysis. Review of Financial Studies **6**(2) (1993) 345–74 60