Gradient Descent for Sharpe Ratio Optimization

Joseph Nunez
Seminar in Differential Geometry
Harvey Mudd College
March 7, 2018

Abstract

A classical problem in portfolio management is the optimization of the Sharpe ratio: the ratio of a portfolio's returns to its level of risk. We develop a Sharpe ratio optimization algorithm which uses gradient descent of the Kullback-Leibler divergence to determine the proportions with which to allocate a portfolio to optimize the Sharpe ratio. We backtest our algorithm on a universe of 25 stocks with data from Quandl.com, and we observed a 21.49% increase in portfolio value over the last 150 trading days, a period during which the Dow Jones industrial Average increased by 15.80%. This initial result is very promising, and invites further study, including extending the problem to include the possibility of going short in stocks as well as long.

1 Introduction

A common goal of investors is to achieve the highest possible rate of return on invests while exposing themselves to as little risk as possible. When evaluating investments based on this paradigm, it is useful to consider the Sharpe ratio, which the ratio of an investment's return (or expected return) over the risk associated with that investment. I will propose a strategy for maximizing the Sharpe ratio of a portfolio to be distributed across n assets using gradient descent. This strategy will rely upon conventional quantitative measures of risk and expected returns from finance and portfolio theory, then use geometric methods to determine the optimal portfolio configuration according to those measures.

The gradient descent algorithm we produce relies upon a number of approaches taken from portfolio theory about how to measure the expected return and risk of a portfolio of stocks. In particular, we will need to define what a portfolio is in the context of our algorithm, how we measure a portfolio's expected returns, and how we determine the risk level of a portfolio. The proof of our algorithm also relies on results from information geometry, so some conclusions from information geometry will be included as well that will be discussed in this section as well.

1.1 Portfolio Representation

A portfolio of n stocks denoted $a_1, \ldots, a_n$ can be represented as a sequence of weights $\pi_1, \ldots, \pi_n$, where

$$\sum_{i=1}^{n} \pi_i = 1; \quad 0 \leq pi_i \leq 1 \; \forall i. \tag{1}$$

Each weight π_i corresponds to the proportion of the portfolio allocated to a_i. The set of all points satisfying these constraints, points in $\mathbb{R}^n$ whose coordinates sum to 1 and are individually bounded by 0 and 1, is an n-dimensional simplex, a hyperplane of $\mathbb{R}^n$ with some

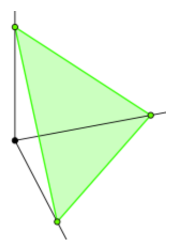
Figure 1: A simplex in $\mathbb{R}^3$ [8]

special properties. In particular, simplexes are a focus of information geometry. Therefore, we can use an n-dimensional simplex to represent all possible portfolios of n stocks

1.2 Portfolio Returns

It has been empirically observed in the field of finance that asset prices follow a log normal distribution. That is, the logarithmic continuous growth rates of asset prices roughly fit a normal distribution, though in practice the distributions of returns tend to have thinner sides and fatter tails than normal distributions owing to the existence of news events such as earning reports that cause large, sudden shifts in asset prices. Figure 2 visually shows how well asset returns fits a normal distribution, in this case using the past year's daily return for Apple stock.

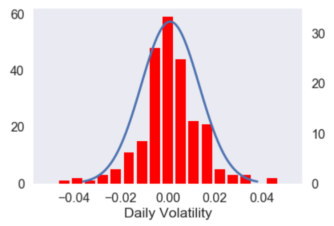

Figure 2: The distribution of the log daily returns of AAPL overlaid with a normal distribution with matching standard deviation and mean.

To compute the log continuous growth rates for an asset, we will look at the daily close of each asset a_i's price (the value of the asset at 4:00 PM ET, when markets close), divide it by the previous day's close, then take the logarithm of that ratio. Once we have done so for every day in our sample (we use one year's worth of data for ever trading day), we take the mean, which gives us our expected daily return over the long run μ_i. The standard deviation of the log returns gives us σ_i, which will act as our proxy for risk.

Once we have the expected return μ_i for each asset a_i in our portfolio, the expected returns of the overall portfolio is simply a weighted sum of the returns of the individual assets

$$\mu_\pi = \sum_{i=1}^{n} \pi_i \mu_i. \tag{2}$$

The risk level of portfolio is slightly less straightforward, owing to the fact that the returns of different assets are correlated with one another. As such it is necessary to use the covariance between pairs of assets when calculating the risk of the portfolio overall. Let R_i be the distribution of returns corresponding to asset a_i. Using the statistical formula for the risk of correlated distributions, we get

$$\sigma_\pi^2 = \sum_{i,j=1}^{n} \pi_i \pi_j Cov(R_i, R_j), \tag{3}$$

where $Cov(R_i, R_j)$ is the covariance of the returns for assets i and j [1].

2 Portfolio Return Distributions

Using our expected return and risk measures, we see that every portfolio can be mapped to a normal distribution. It is therefore useful to have a measure of difference between normal distributions. We will use the Kullback-Leibler divergence, defined over continuous probability distributions as

$$D_{KL}(p, q) = \int_{-\infty}^{\infty} p(x) \log \frac{p(x)}{q(x)} dx.$$

Let p and q be Gaussian distributions with respective means μ_1 and μ_2 and standard deviations σ_1 and σ_2. We wish to find a closed form for the KL-divergence. First, we can separate the logarithm to get

$$D_{KL}(p, q) = \int_{-\infty}^{\infty} p(x) \log p(x) dx - \int_{-\infty}^{\infty} p(x) \log q(x) dx.$$

Recall that the equation of a Guassian is

$$p(x) = \frac{1}{\sqrt{2\pi\sigma_1^2}} e^{-\frac{(x-\mu_1)^2}{2\sigma^2}}$$

Thus the first term becomes

$$\int_{-\infty}^{\infty} p(x) \log \frac{1}{\sqrt{2\pi\sigma_1^2}} e^{-\frac{(x-\mu_1)^2}{2\sigma_1^2}} dx$$

Then the logarithm further simplifies to

$$\frac{1}{2}\log 2\pi\sigma_1^2 \int_{-\infty}^{\infty} p(x)\frac{(x-\mu_1)^2}{2\sigma_1^2} dx$$

This separates into three integrals when the square gets expanded

$$\frac{1}{2}\log 2\pi\sigma_1^2 \frac{\int p(x)x^2 dx - \int p(x)2\mu_1 x dx + \int p(x)\mu_1^2 dx}{2\sigma_1^2}$$

The third integral becomes μ_1^2 since Guassians have area 1. The first two integrals become the expectations of x^2 and $2\mu_1 x$. Since variance of x is equal the expectation of x^2 minus the square of the expectation of x, the expectation of x^2 is equal to $\sigma_1^2 + \mu_1^2$. The expectation of x is simply μ_1. This yields

$$\frac{1}{2}\log 2\pi\sigma_1^2 + \frac{\sigma_1^2 + \mu_1^2 - 2\mu_1^2 + \mu_1^2}{2\sigma_1^2} = \frac{1}{2}\log 2\pi\sigma_2^2 + \frac{1}{2}$$

Which factors to become $\frac{1}{2}(1 + \log 2\pi\sigma_1^2)$.

Similarly, the second term can be rewritten as

$$\int_{-\infty}^{\infty} p(x) \log \frac{1}{\sqrt{2\pi\sigma_2^2}} e^{-\frac{(x-\mu_2)^2}{2\sigma_2^2}} dx$$

$$\log\sqrt{2\pi\sigma_2^2} - \int_{-\infty}^{\infty} p(x)\frac{(x-\mu_2)^2}{2\sigma_2^2} dx$$

This separates into three integrals when the square gets expanded

$$\frac{1}{2}\log 2\pi\sigma_1^2 + \frac{\int p(x)x^2 dx - \int p(x)2\mu_2 x dx + \int p(x)\mu_2^2 dx}{2\sigma^2}$$

The third integral becomes μ_2^2 since Guassians have area 1. The first two integrals become the expectations of x^2 and $2\mu_2 x$. Since variance of x is equal the expectation of x^2 minus the square of the expectation of x, the expectation of x^2 is equal to $\sigma_1^2 + \mu_1^2$. The expectation of x is simply μ_1. This yields

$$\frac{1}{2}\log 2\pi\sigma_2^2 + \frac{\sigma_1^2 + \mu_1^2 - 2\mu_1\mu_2 + \mu_2^2}{2\sigma_2^2} = \frac{1}{2}\log 2\pi\sigma_2^2 + \frac{\sigma_1^2 + (\mu_1 - \mu_2)^2}{2\sigma_2^2}$$

Combining these expressions, we get

$$\frac{1}{2}\log 2\pi\sigma_2^2 + \frac{\sigma_1^2 + (\mu_1 - \mu_2)^2}{2\sigma_2^2} - \frac{1}{2}(1 + \log 2\pi\sigma_1^2)$$

We then combing the logarithms to get $\frac{1}{2}\log\frac{2\pi\sigma_2^2}{2\pi\sigma_1^2} = \frac{1}{2}\log\frac{\sigma_2^2}{\sigma_1^2} = \frac{\sigma_2}{\sigma_1}$, which yields

$$KL(p,q) = \log\frac{\sigma_2}{\sigma_1} + \frac{\sigma_1^2 + (\mu_1 - \mu_2)^2}{2\sigma_2^2} - \frac{1}{2}. \tag{4}$$

This will serve as the closed form to be used in the implementation of the algorithm [3].

3 Information Geometry

Information geometry is the study of the manifold of probability distributions, called the statistic manifold, or S, and was developed primarily by Shun-ichi Amari. Notably, when dealing with probability distributions with n disjoint outcomes, the set of all such distributions is an n-dimensional simplex, since each outcome must have a probability between 0 and 1 and the cumulative probability of all outcomes must be 1.

In his 1992 paper 'Information Geometry of Boltzmann Machines', Amari proved that Boltmann machines, recurrent neural networks which learn to reproduce a probability distribution, will always converge to a global optimum approximation of the target distribution. Boltzmann machines have a set of connection weights which represent the probability of a certain node being 0 or 1. For a machine with n nodes, there are 2^n possible outcomes, so the set of possible weights is a $(2^n - 1)$-dimensional simplex. To find the best approximation of a target distribution within this space, a Boltzmann machine performs stochastic gradient descent of the KL-divergence between the distribution it is realizing and the target distribution. In proving that Boltzmann machines will always converge to the global best approximation, Amari demonstrated that there is a dually affine connection between the error surface of the KL-divergence over a finite-dimensional simplex [2].

In the context of gradient descent, the error surface is defined by the map

$$D : S \to \mathbb{R};$$

$$s \mapsto D(p(s), y),$$

where S is the space of possible weights, D is a distance function, s is a set of weights in S, $p(s)$ is the prediction generated based on the weights s, and y is the target trying to being realized. Amari demonstrated that the error surface is smooth and lacks local minima when it the weight space is a finite dimensional simplex. Therefore, by traveling down the gradient of the error surface, an algorithm is guaranteed to reach the global minimum of the error surface.

Therefore, because our weight space for our portfolio configurations is a finite-dimensional simplex, we want to construct our problem in the context of minimizing the KL-divergence between the distribution of the log returns of our portfolio and some target distribution. If we do so, Amari's conclusions allow us to guarantee that using gradient descent will find the optimal portfolio configuration.

4 Methods

The methods section consists of both the gradient descent algorithm itself and the methods used to backtest the algorithm's performance. The gradient descent algorithm is an alteration of stochastic gradient descent algorithms owing to the fact that rather than having a large set of sample inputs and target outputs, we instead have one target distribution and a single set of input stocks. As a result, rather than stochastically estimating the gradient, we actually compute the gradient with respect to the portfolio weights. My backtesting approach rather simply applies the algorithm every day to compute new weights, invests based on that, and records the resulting change in portfolio value at the end of each day.

4.1 Gradient Descent Algorithm

To perform gradient descent over an error surface, at each point, we need to calculate the gradient with respect to the weights, then take a small step in the direction of most negative gradient. We do this until the error stops changing with successive steps, indicating a gradient of the zero vector, which indicates a minimum. Since the error surface has no global minima, this minimum must represent the portfolio weights with the least error.

To implement this algorithm, we must compute the gradient of the KL-divergence with respect to each portfolio weight π_i. Let p be the distribution of log returns generated according to the portfolio weights and let q be the target distribution. Let μ_1 and μ_2 be the respective means μ_1 and μ_2 and σ_1 and σ_2 be the respective standard deviations of p and q. Then we have

$$\frac{\partial}{\partial \pi_i} D_{KL}(p, q) = \frac{\partial}{\partial \pi_i} \log \frac{\sigma_2}{\sigma_1} + \frac{\sigma_1^2 + (\mu_1 - \mu_2)^2}{2\sigma_2^2} - \frac{1}{2}.$$

Now take our expressions from section 1.2 for the return and risk of a portfolio, which are given by

$$\mu_\pi = \sum_{i=1}^{n} \pi_i \mu_i,$$

$$\sigma_\pi^2 = \sum_{i,j=1}^{n} \pi_i \pi_j Cov(R_i, R_j),$$

and plug them into the divergence formula. To avoid confusion, we'll now use σ and μ for the standard deviation and mean of the target distribution:

$$\frac{\partial}{\partial \pi_i} D_{KL}(p,q) = \frac{\partial}{\partial \pi_i}(\log \sigma - \log \sqrt{\sum_{i,j=1}^{n} \pi_i \pi_j Cov(R_i, R_j)}$$

$$+ \frac{\sum_{i,j=1}^{n} \pi_i \pi_j Cov(R_i, R_j) + (\sum_{j=1}^{n} \pi_j \mu_j - \mu)^2}{2\sigma^2} - \frac{1}{2})$$

Now we'll compute the partial derivative of each term. Note that σ and μ are constant with respect to π_i since the target does not change depending on the weights. The $\log \sigma$ and $\frac{1}{2}$ are constants, so they become zero in the partial derivative. The other log term becomes

$$-\frac{1}{\sqrt{\sum_{i,j=1}^{n} \pi_i \pi_j Cov(R_i, R_j)}} \frac{\partial}{\partial \pi_i} \sqrt{\sum_{i,j=1}^{n} \pi_i \pi_j Cov(R_i, R_j)}$$

$$= -\frac{1}{\sqrt{\sum_{i,j=1}^{n} \pi_i \pi_j Cov(R_i, R_j)}} \frac{1}{2\sqrt{\sum_{i,j=1}^{n} \pi_i \pi_j Cov(R_i, R_j)}} \frac{\partial}{\partial \pi_i} \sum_{i,j=1}^{n} \pi_i \pi_j Cov(R_i, R_j)$$

$$= -\frac{\sum_{j=1, j\neq i}^{n} \pi_j Cov(R_i, R_j) + 2\pi_i}{2\sum_{i,j=1}^{n} \pi_i \pi_j Cov(R_i, R_j)}.$$

Notice that the $2\pi_i$ is a result of there being a single π_1^2 in the summation and $Cov(R_i, R_i) = 1$. The next term should be split up into two fractions:

$$\frac{\sum_{i,j=1}^{n} \pi_i \pi_j Cov(R_i, R_j)}{2\sigma^2} + \frac{(\sum_{j=1}^{n} \pi_j \mu_j - \mu)^2}{2\sigma^2}$$

The first fraction gets treated just like the previous summation and becomes

$$\frac{\sum_{j=1, j\neq i}^{n} \pi_j Cov(R_i, R_j) + 2\pi_i}{2\sigma^2}.$$

The second fraction becomes

$$\frac{2(\sum_{j=1}^{n} \pi_j \mu_j - \mu)}{2\sigma^2} \frac{\partial}{\partial \pi_i} (\sum_{j=1}^{n} \pi_j \mu_j - \mu)$$

$$= \frac{\mu_i (\sum_{j=1}^{n} \pi_j \mu_j - \mu)}{\sigma^2}.$$

Taken all together, we get that the partial derivative of the Kullback-Leibler divergence with respect to π_i is

$$-\frac{\sum_{j=1, j\neq i}^{n} \pi_j Cov(R_i, R_j) + 2\pi_i}{2\sum_{i,j=1}^{n} \pi_i \pi_j Cov(R_i, R_j)} + \frac{\sum_{j=1, j\neq i}^{n} \pi_j Cov(R_i, R_j) + 2\pi_i}{2\sigma^2} + \frac{\mu_i (\sum_{j=1}^{n} \pi_j \mu_j - \mu)}{\sigma^2}$$

It's straightforward to precompute the covariance matrix and expected returns for all of the assets' returns ahead of time, so everything in this expression is in terms of readily available information, so this can be computed easily inside of an algorithm.

To perform gradient descent according to this rule, during each update step (also called training epoch), the weights are updated according to the following rule:

$$\pi_{t=k+1} = \pi_{t=k} - \epsilon \nabla [D_{KL}(p(\pi_{t=k}), q)] D_{KL}(p(\pi_{t=k}), q),$$

Where $\pi_{t=k}$ is the portfolio weight vector after k update iterations, p is the function that turns the portfolio weights into a distribution of returns, q is target distribution, D_{KL} is the Kullback-Leibler divergence, and ∇ is the gradient operator with respect to all of the π_is.

We will also use a variable learning rate, which is a mechanism that gradient descent algorithms use to help them converge faster. The variable learning rate strategy we used is to increase ϵ by a factor of 1.05 whenever the most recent update caused the error to decrease and to decrease ϵ by a factor of 0.7 whenever the most recent update caused the error to increase by a factor of 1.04 or greater. My implementation of the algorithm used 800 training epochs and an initial learning rate of $\epsilon = 0.001$. These values were chosen through trial and error to get the algorithm to quickly and consistently converge.

Up to this point, we have discussed the gradient descent algorithm in terms of approaching a target distribution rather than optimizing the Sharpe ratio of the portfolio. Since gradient descent requires a target to approach, this is a necessity, but it means that we must choose the target distribution such that minimizing the divergence from the target distribution maximizes the Sharpe ratio. Fortunately, this choice is relatively straightforward: choose a distribution with an unachievably high Sharpe ratio, but also close enough to the achievable distributions that the divergence is meaningfully different for different achievable distributions. As the algorithm seeks to get closer to this distribution, the portfolio's Sharpe ratio will increase. For our target's growth rate, we choose the maximum growth rate among the individual stocks because the portfolio cannot grow faster than its fastest-growing element. For our target risk level, we don't know what the true minimum risk is since that depends upon the covariances, but we know that we can minimize the $\pi_i \pi_j$ component of each term in the summation by evenly distributing our portfolio across all of the assets. To be extra cautious and ensure we don't overestimate the minimum risk, we'll also divide this risk by a factor of 10. Now we have μ and σ for our target distribution.

4.2 Backtesting

I implemented by backtesting script in Python 3 in a Juptyer Notebook. Unfortunately, I cannot make a functioning version of this code publicly available, as it requires a Quandl API key to run, and Quandl's terms of service forbid sharing one's API key. The stocks I considered investing in were 25 stocks chosen from the Dow Jones Industrial Average (DJIA). The five DJIA stocks I did not consider were excluded because my data source, Quandl.com, did not have enough of their historical pricing data freely available in order for me to include them in my testing[1].

For each of the past 150 trading days[2,3], I ran the gradient descent algorithm on the previous year's worth of data (251 trading days) to obtain the portfolio weights with the best Sharpe ratio.

I then calculated the returns that a portfolio with the determined weights would have had on that day and applied that rate of return to a portfolio with an initial value of $10,000. For the sake of comparison, I also tracked the performance of the Dow Jones Industrial Average over that time period as a baseline, and applied that growth rate to a portfolio with an initial value of $10,000.

5 Results

In Figure 3 on the following page, the orange line represents the performance of $10,000 invested in a DJIA index or mutual fund, and the blue line represents the performance of $10,000 invested according to the algorithm.

Over the course of 150 trading days, the gradient descent algorithm resulted in a growth rate of 21.49% compared with a growth of only 15.80% for the market as a whole. Interestingly, the period between trading days 130 and 140 saw a rapid decline in the investments made by our algorithm, though this is unsurprising as it coincided with a very sharp decline in the stock market (including the single largest point loss in the DJIA in a single day). The particular reasons for this crash are still unclear, but many speculate that it was caused by broad fears that stocks in general were overpriced, which lead to selling off stocks across all sectors, which meant that even historically uncorrelated stocks behaved similarly. Our risk estimates considered such an event highly unlikely based on our covariance values, so the portfolio was poorly configured for such an event. However, our portfolio

[1] The stocks I excluded were 3M (MMM), Apple (AAPL), DowDuPont (DWDP), Intel (INTC), and Travelers Companies (TRV).

[2] Trading days are Mondays through Fridays excluding holidays.

[3] The last trading day I used in my testing was February 28, 2018.

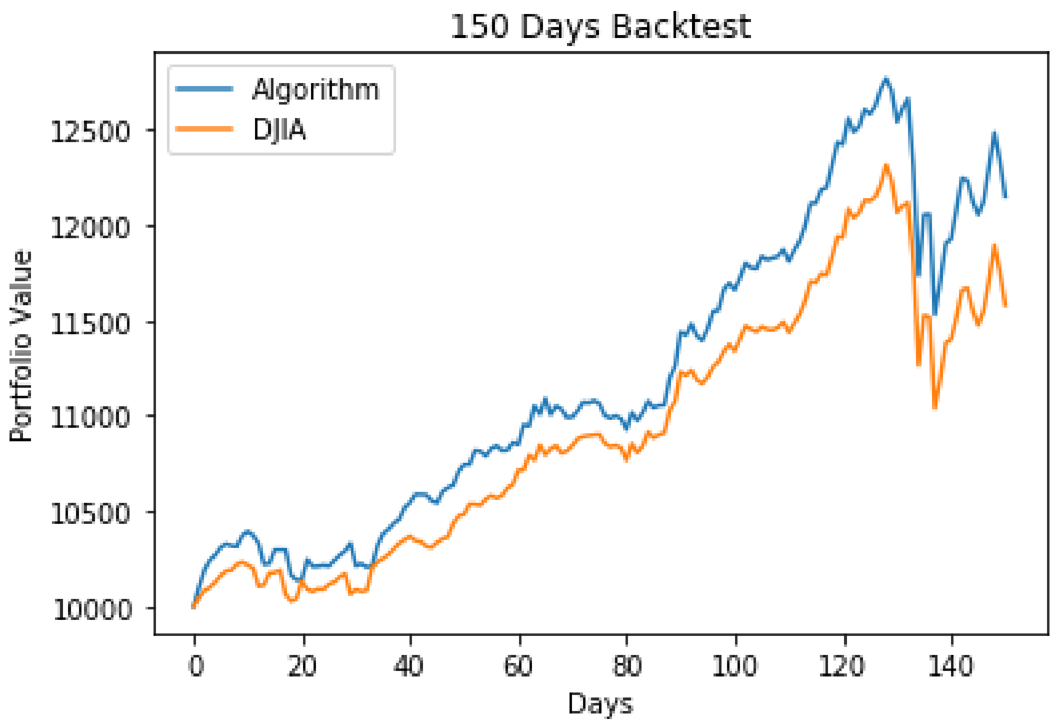

Figure 3: 150 trading days backtest of algorithm versus DJIA baseline

did not underperform the market during this time, indicating that our algorithm does not overexpose us to these events relative to the market as a whole.

More importantly, during the low volatility period that marks the beginning of the graph, we see that our algorithms gradually outperforms the market, as our optimization gives us a slight edge over the market when the stocks are performing as expected. When this is coupled with the fact that our algorithm does not face greater risk over the market, we see that our algorithm will gradually outpace the market over time.

6 Acknowledgments

Thank you to professors Gu, Evans, and Sweedyk for courses in which I developed the background knowledge of information geometry, gradient descent algorithms, and financial economics necessary to complete this project. Parts of the code I used during this project were adapted from code developed during these courses.

7 Future Work

This algorithm demonstrates the gradient descent along an error surface generated by the Kullback-Leibler divergence is a promising approach for tackling problems in portfolio theory. There are several directions in which to expand this algorithm.

First, we could expand the weight space beyond 0 and 1 to allow for the potential to go short in an asset and use the capital from the short sale to go extra long in another asset. This approach would require altering the function that converts the weights to a portfolio and calculates its distance from the target distribution. We would need to prove this new function to still produce an error surface with no local minima, and to recompute the gradient to account for this change.

Another extension would be to study the rebalancing frequency. Out of convenience with respect to data availability, I used daily returns, hence optimizing the Sharpe ratio over a 1-day horizon, so I rebalanced daily. It's quite possible that this is not the optimal frequency with respect to long-term growth, and insight into this problem could be found in Wong's thesis, which focused on rebalancing frequency [1].

There is also the possibility of exploring different risk metrics to use; I recently heard of a type of probability distribution called a Slash distribution, which has similar features to a normal distribution, but features the narrower sides and fatter tails which have been empirically observed in asset log growth rates.

References

[1] Wong, T. 2016, 'Geometry and Optimization of Relative Arbitrage', PhD Thesis, University of Michigan, Ann Arbor MI.

[2] Amari, S., Kurata, K., Nagaoka, H. 1992, 'Information Geometry of Boltzmann Machines', IEEE Transactions on Neural Networks, 260-271.

[3] 'KL divergence between two univariate Gaussians', Stats.stackexchange.com, https://stats.stackexchange.com/questions/7440/kl-divergence-between-two-univariate-gaussians

[4] 'NumPy Reference — NumPy v1.14 Manual', Docs.scipy.org, https://docs.scipy.org/doc/numpy/reference/

[5] 'Matplotlib 2.2.0 documentation', Matplotlib.org, https://matplotlib.org/#documentation

Data Sources:

[6] 'Quandl', Quandl.com, https://www.quandl.com/

[7] '^DJI Historical Prices | Dow Jones Industrial Average Stock - Yahoo Finance', Finance.yahoo.com, https://finance.yahoo.com/quote/%5EDJI/history?p=%5EDJI

Image Sources:

[8] 'Simplex', Wikipedia.org, https://en.wikipedia.org/wiki/Simplex