Working Paper IS-97-005, Leonard N. Stern School of Business, New York University.
In: *Decision Technologies for Financial Engineering (Proceedings of the Fourth International Conference on Neural Networks in the Capital Markets, NNCM-96)*, pp. 3–22. Edited by A.S.Weigend, Y.S.Abu-Mostafa, and A.-P.N.Refenes. Singapore: World Scientific, 1997.
`http://www.stern.nyu.edu/~aweigend/Research/Papers/SharpeRatio`

NONLINEAR TRADING MODELS THROUGH SHARPE RATIO MAXIMIZATION

MARK CHOEY
Advanced Technology Group
Siemens Nixdorf Information Systems, Inc.,
200 Wheeler Road
Burlington, MA 01803
`choey@ix.netcom.com`

ANDREAS S. WEIGEND
Department of Information Systems
Leonard N. Stern School of Business, New York University
44 West Fourth Street, MEC 9-74
New York, NY 10012
`aweigend@stern.nyu.edu`
`www.stern.nyu.edu/~aweigend`

While many trading strategies are based on price prediction, traders in financial markets are typically interested in risk-adjusted performance such as the Sharpe Ratio, rather than price predictions themselves. This paper introduces an approach which generates a nonlinear strategy that explicitly maximizes the Sharpe Ratio. It is expressed as a neural network model whose output is the position size between a risky and a risk-free asset. The iterative parameter update rules are derived and compared to alternative approaches. The resulting trading strategy is evaluated and analyzed on both computer-generated data and real world data (DAX, the daily German equity index). Trading based on Sharpe Ratio maximization compares favorably to both profit optimization and probability matching (through cross-entropy optimization). The results show that the goal of optimizing out-of-sample risk-adjusted profit can be achieved with this nonlinear approach.

1 Introduction

1.1 Risk-adjusted Performance

A trader ideally would like to have very high profit (return on investment) with very low risk (variability of return). However, there is a fundamental relationship between profit and risk: the higher the expected profit, the greater the risk and similarly, the lower the expected profit, the lower the risk. This paper focuses on the tradeoff between profit and risk (as opposed to profit or risk alone). One way of representing this tradeoff is to combine profit and risk by taking the expected profit and dividing by the risk measure. This single quantity is called the *reward-to-risk* ratio or *risk-adjusted* performance. There are several risk-adjusted measures based on different notions of risk.

Examples are the *Sharpe Ratio* (the focus of this paper), the *Jensen Alpha*, and the *Treynor Ratio* (Hammer, 1991).

1.2 The Sharpe Ratio

The Sharpe Ratio (SR) takes the mean of the excess profit or excess return and divides it by the standard deviation of the excess return (Sharpe, 1970; Sharpe, 1994). The excess return is defined as the rate of return on an asset minus the return available on a baseline asset. The baseline asset is typically a short-term risk-free asset such as the three-month U.S. Treasury Bill. SR expresses the excess return in units of its standard deviation as

$$\mathrm{SR} = \frac{\text{average excess return}}{\text{standard deviation of excess return}} \quad . \tag{1}$$

One important implication of using only the first and second moments of the excess returns is that positive returns and negative returns are treated identically—large positive and negative returns of the same magnitude have the same effect on the risk measure.

SR scales as $\sqrt{T}$ with the time period over which it is measured, where T is the number of periods SR is calculated over. It is common to *annualize* SR which allows for the fair comparison of different training strategies even if the time scales of the strategies are not comparable.

2 Trading Strategy based on the Sharpe Ratio

Traditionally, SR is used to *evaluate* the performance of a trading strategy. In technical trading, strategies are often optimized by minimizing the sum-squared error (SSE) (or a more robust measure such as log(cosh(.)), Zimmermann & Weigend (1997)) of point predictions, and subsequently evaluated with SR or other risk-adjusted measures. Optimizing SSE while evaluating on SR will often lead to results with an optimal out-of-sample SSE, but a sub-optimal SR. A recent example of such a mismatch between the function used in training and the ultimate goal is the paper by Pi & Rögnvaldsson (1996). Here, a network is trained to predict a price, but evaluated on the SR which includes an ad-hoc translation of the price prediction to a trading strategy.

Here we take SR seriously and do not try to predict prices, instead *directly* use SR as our objective function. We assume that there are two asset classes, risky and risk-free, and on period i, the optimal position size or *allocation* for the risky asset is selected to maximize the SR. The allocation, α, takes on continuous values between $[-1, 1]$. If α is positive, a *long* position is taken, and if α is negative, a *short* position is taken. A short position means that the risky asset is borrowed by the trader and sold to an investor with the hope that the asset will fall in price, allowing the trader to repay the borrowed asset at a lower price and make a profit. If $\alpha = 0.5$, 50% of available cash are used to purchase (take a long position) the risky asset; if $\alpha = -0.5$, 50% of the total cash reserves are set aside as collateral for the short position.

3 Architecture

The trading strategy is implemented by using a neural network with multiple inputs, multiple hidden units, and one output. The inputs depend on the problem and are explained below in the specific examples (see Section 6 and 7). The hidden units are tanh sigmoids allowing re–representation of the inputs for the task. The output is also a (-1,1) tanh unit; its activation value, α_i, represents the allocation on day i.

4 Objective Functions

This section discusses the iterative weight updates for the SR objective function and compares it to alternative objectives, focusing on the comparison to the updates for profit maximization. For clarity, we derive the update rule for a single weight, w, to the output unit. As usual, hidden layers require the chain rule for their weight updates, see e.g. Rumelhart *et al.* (1996). The output of the network is the allocation, α_i, for period i.

4.1 Update rule for SR Maximization

For simplicity of language, we assume that we have daily data. Define p_i to be the price of the asset on day i, and x_i to be the *relative price returns* on day i

$$x_i = \ln(p_i) - \ln(p_{i-1}) \approx \frac{p_i - p_{i-1}}{p_i} \quad . \tag{2}$$

Additionally, define the asset *returns* (that includes the position size α_i) to be

$$\xi_i = \alpha_i x_i \quad . \tag{3}$$

The asset return is equal to the position size multiplied by the relative price return of the risky asset on day i. The average daily return, μ, is given by

$$\mu = \frac{1}{N} \sum_{i=1}^{N} \xi_i \tag{4}$$

and the average daily excess return, μ_e, is

$$\mu_e = \mu - k \tag{5}$$

where k is the daily risk-free rate of return and is assumed to be constant over time. The standard deviation of the excess return is given by

$$\sigma = \left\{ \frac{1}{N} \sum_{i=1}^{N} (\xi_i - \mu_e)^2 \right\}^{1/2} \quad . \tag{6}$$

Now, the annualized SR is defined as

$$\mathrm{SR}_T = \sqrt{T}\, \frac{\mu_e}{\sigma} \tag{7}$$

T is the number of trading days per year and we assume that $T = 253$.

To update the weight with gradient ascent, we now need to calculate the gradient by taking the partial derivative of the annualized SR [a] with respect to the weight, w

$$\frac{\partial \mathrm{SR}_T}{\partial w} = \frac{\sqrt{T}}{\sigma^2} \left(\sigma \frac{\partial \mu_e}{\partial w} - \mu_e \frac{\partial \sigma}{\partial w} \right) \quad . \tag{8}$$

Next, we take the partial derivative of μ_e

$$\frac{\partial \mu_e}{\partial w} = \frac{\partial \mu}{\partial w} = \frac{1}{N} \sum_{i=1}^{N} x_i \frac{\partial \alpha_i}{\partial w} \tag{9}$$

where

$$\frac{\partial \alpha_i}{\partial w} = \frac{\partial \alpha_i}{\partial \gamma_i} \frac{\partial \gamma_i}{\partial w} \quad . \tag{10}$$

γ is the "net-input" to the output unit (i.e., the network input multiplied by w) and

$$\frac{\partial \alpha_i}{\partial \gamma_i} = 1 - \alpha_i^2 \tag{11}$$

$$\frac{\partial \alpha_i}{\partial w} = (1 - \alpha_i^2)\gamma_i \tag{12}$$

since $\partial \gamma_i / \partial w = \gamma_i$. The partial derivative of σ with respect to w can be calculated by taking the partial derivative of the variance of the excess returns,

[a] Note that in this case taking the partial derivative of $\log(\mathrm{SR}) = \log(\mu_e) - \log(\sigma)$ is mathematically not possible since μ_e can be negative.

σ^2, and using the relation $\partial\sigma/\partial w = (1/2\sigma)(\partial\sigma^2/\partial w)$ to arrive at $\partial\sigma/\partial w$. Taking the partial derivative of σ^2 we get

$$\frac{\partial\sigma^2}{\partial w} = \frac{2}{N}\left[\sum_{i=1}^{N}(\xi_i - \mu_e)\left(x_i\frac{\partial\alpha_i}{\partial w} - \frac{\partial\mu_e}{\partial w}\right)\right]$$

$$= \frac{2}{N}\left[\sum_{i=1}^{N}\xi_i x_i\frac{\partial\alpha_i}{\partial w} - \sum_{i=1}^{N}\xi_i\frac{\partial\mu_e}{\partial w} - \sum_{i=1}^{N}\mu_e x_i\frac{\partial\alpha_i}{\partial w} + \sum_{i=1}^{N}\mu_e\frac{\partial\mu_e}{\partial w}\right] \quad . \tag{13}$$

From Eq. 9, the last two terms cancel

$$\frac{\partial\sigma^2}{\partial w} = \frac{2}{N}\left[\sum_{i=1}^{N}\xi_i\left(x_i\frac{\partial\alpha_i}{\partial w} - \frac{1}{N}\sum_{j=1}^{N}x_j\frac{\partial\alpha_j}{\partial w}\right)\right] \quad , \tag{14}$$

arriving at $\partial\sigma/\partial w$

$$\frac{\partial\sigma}{\partial w} = \frac{1}{\sigma N}\left[\sum_{i=1}^{N}\xi_i\left(x_i\frac{\partial\alpha_i}{\partial w} - \frac{1}{N}\sum_{j=1}^{N}x_j\frac{\partial\alpha_j}{\partial w}\right)\right] \quad . \tag{15}$$

Substituting $\partial\sigma/\partial w$ (Eq. 15) and $\partial\mu_e/\partial w$ (Eq. 9) into $\partial\mathrm{SR}_T/\partial w$ (Eq. 8)

$$\frac{\partial\mathrm{SR}_T}{\partial w} = \frac{\sqrt{T}}{\sigma N}\left[\sum_{i=1}^{N}x_i\frac{\partial\alpha_i}{\partial w} - \frac{\mu_e}{\sigma^2}\sum_{i=1}^{N}\xi_i x_i\frac{\partial\alpha_i}{\partial w} + \frac{\mu_e}{N\sigma^2}\sum_{i=1}^{N}\xi_i\sum_{j=1}^{N}x_j\frac{\partial\alpha_j}{\partial w}\right] \quad . \tag{16}$$

Interchanging the order of summation on the right side and factoring out $x_i\partial\alpha_i/\partial w$, we get

$$\frac{\partial\mathrm{SR}_T}{\partial w} = \frac{\sqrt{T}}{\sigma N}\sum_{i=1}^{N}\left[1 - \frac{\mu_e}{\sigma^2}\left(\xi_i - \frac{1}{N}\sum_{j=1}^{N}\xi_j\right)\right]x_i\frac{\partial\alpha_i}{\partial w} \quad . \tag{17}$$

With $\mu = \frac{1}{N}\sum_{j=1}^{N}\xi_j$ (Eq. 4) and $\mu = \mu_e + k$ we can define

$$\kappa_i = \frac{\sqrt{T}}{\sigma}\left[1 - \frac{\mu_e}{\sigma}\frac{(\xi_i - (\mu_e + k))}{\sigma}\right] \tag{18}$$

and the SR update rule can be written in its final form as

$$\frac{\partial\mathrm{SR}_T}{\partial w} = \frac{1}{N}\sum_{i=1}^{N}\kappa_i x_i\frac{\partial\alpha_i}{\partial w} \quad . \tag{19}$$

4.2 Comparison to Other Objectives

There are several objective functions we could use to train the network including: SSE, typically used for prediction; cross-entropy, which gives the probability of x_i being positive or negative; and profit, which maximizes the return of the trading strategy.

SSE is given by

$$\sum_{i=1}^{N}(y_i - x_i)^2 \tag{20}$$

where y_i is the prediction and x_i is the price return for day i. SSE is fundamentally different from SR because SSE only tries to minimize prediction errors using pattern-by-pattern supervised learning. A trading strategy based on SSE optimization does not necessarily lead to an optimal SR.

Cross-entropy is given by

$$-\sum_{i=1}^{N}[d_i \ln y_i + (1-d_i)\ln(1-y_i)] \tag{21}$$

where $d_i \in \{0,1\}$. Equation 21 is the familiar cross-entropy of two classes (Buntine & Weigend, 1991; Rumelhart *et al.*, 1996). In order to create a trading strategy, we substitute $\frac{1}{2}(1+t_i)$ for d_i and $\frac{1}{2}(1+\alpha_i)$ for y_i and obtain

$$-\sum_{i=1}^{N}\left[\frac{1+t_i}{2}\ln\frac{1+\alpha_i}{2} + \frac{1-t_i}{2}\ln\frac{1-\alpha_i}{2}\right] \tag{22}$$

where $t_i = \text{sign}(x_i)$ is the target class (i.e., positive or negative price return) and α_i becomes the allocation. Cross-entropy is used as objective function when the goal is to obain the probability of an event (e.g., of the return being positive). It falls into the class of pattern-by-pattern supervised learning. It does not take risk into account.

The objective based on profit maximization we call the *profit objective* is measured by the average daily return, μ (Eq. 4). Like SSE and cross-entropy, the profit objective does not explicitly consider risk. Its update rule is the partial derivative of μ (Eq. 9, repeated here)

$$\frac{\partial\mu}{\partial w} = \frac{1}{N}\sum_{i=1}^{N} x_i \frac{\partial\alpha_i}{\partial w} \quad . \tag{23}$$

Comparing the SR update rule (Eq. 19) and the profit objective update rule (Eq. 23), we notice that they are identical with the exception of κ_i. For this reason, we call κ the *profit objective weight.*

The profit objective weight, κ_i (Eq. 18), controls (on a daily basis) the network's emphasis between maximizing profit and minimizing risk. This is completely determined by sign(κ_i): a positive κ_i causes all of the weights to be modified in the same direction (not the same magnitude) as if training on the profit objective. Conversely, a negative κ_i causes all of the weights to be modified in the opposite direction. Note that the two update rules modify the weights identically if $\kappa_i = 1$.

Analyzing Eq. 18, we can see when the SR update rule maximizes profit and when it minimizes risk. The SR update rule maximizes profit (i.e., κ_i is positive) when the inequality,

$$\frac{\mu_e}{\sigma}\frac{(\xi_i - (\mu_e + k))}{\sigma} < 1 \quad , \tag{24}$$

is satisfied and minimizes risk when it is not satisfied. Eq. 24 consists of two factors: the daily SR (μ_e/σ) and a quantity similar to a test statistic, $(\xi_i - (\mu_e + k))/\sigma$. Focusing on the case when $\mu_e/\sigma > 0$, the SR update rule maximizes profit for day i when:

1. an individual pattern is below the average profit ($\xi_i < \mu$) or

2. the return on day i does not deviate from the average profit by more than $(\xi_i - (\mu_e + k))/\sigma < (\mu_e/\sigma)^{-1}$.

The interpretation of the first point is that while overall, the trading strategy is outperforming the risk-free asset (since $\mu_e/\sigma > 0$), the individual return, ξ_i, is below the average profit, μ. For this pattern, the SR update rule will move in a direction, by adjusting the weights, that will increase ξ_i (i.e., increasing profit) which brings the return closer to μ. Conversely, if ξ_i is above μ, the SR update rule moves in a direction that decreases ξ_i (i.e., decreasing profit, but also decreasing risk). This has the effect of drawing the individual patterns towards μ, forcing the daily returns to be as consistent as possible which, in general, also decreases σ.

The second point is responsible for the continuous increase of μ during the training process. Assuming the daily SR to be 1, ($\mu_e/\sigma = 1$), if the deviation of ξ_i from μ is within one σ, the SR update rule will increase ξ_i (because Eq. 24 will be satisfied). If the deviation is not within one σ, the SR update rule decreases ξ_i. The effect is to increase those ξ_i that are close to μ and to decrease those ξ_i not close to μ. The degree of closeness is determined by $(\mu_e/\sigma)^{-1}$. The larger the SR, the closer ξ_i needs to be to μ in order to satisfy the inequality (Eq. 24).

5 Search

Gradient ascent is performed on the SR to update the network parameters. The SR objective function is a *global* (requires all patterns to compute SR) cost function, but the updates are made *locally* or on a per-pattern basis. Therefore, for the optimal calculation of $\partial \mathrm{SR}_T/\partial w$, we recompute $\{\alpha_i\}$, μ_e, and σ after each pattern since $\{\alpha_i\}$ changes with every update. Thus, μ_e and σ also change every update.

The search procedure used is summarized in the following steps:

1. Calculate the set of allocations, $\{\alpha_i\}$, in response to all patterns by making a feed-forward pass through the network.
2. Calculate μ_e and σ (Eq. 5 and Eq. 6 respectively) using all patterns.
3. Choose a pattern i at random and calculate ξ_i (Eq. 3).
4. Calculate $\partial \mathrm{SR}_T/\partial w$ (Eq. 19) using μ_e, σ, and ξ_i.
5. Update the network weights (Eq. 25).
6. Repeat using the next pattern.

The weights are updated with

$$w = w + \eta \Delta w \tag{25}$$

where

$$\Delta w = \frac{\partial \mathrm{SR}_T}{\partial w} = \kappa_i x_i (1 - \alpha_i^2) \gamma_i \quad . \tag{26}$$

During the course of training we expect σ to typically decrease since we are maximizing the SR. However, in revisiting Eq. 18, we see that the $1/\sigma$ term outside of the brackets contributes to *increasing* the size of Δw as training progresses. This has an effect that is opposite to that of simulated annealing. One variation we tried was to remove this term, which helps to stabilize the magnitude of weight changes as learning progresses.

The computational cost of the weight updates can be reduced by using block-mode training which accumulates gradients for a block of patterns *before* updating the weights. Additionally, rather than computing μ_e and σ for all the patterns, (i.e., after feed-forward passes through the entire training set to produce $\{\alpha_i\}$), a stochastic approximation of μ_e and σ can be employed.

6 Computer-generated Data Experiments

The trading strategy's two key variables are the price return, x, and the allocation, α. The network only has control of the variable α. It is interesting to see the optimal value of α if given *perfect* knowledge of x (i.e., the future values of x are known exactly). The optimal α in this experiment should give us the *best possible* SR we could ever obtain since we have a perfect estimate of the price return for tomorrow, i.e., the data is *noise-free.* The noise-free price return data is our first computer-generated data experiment.

In real-world situations however, we do not have perfect knowledge of x, we can only expect to have an estimate of x that includes error bars or confidence intervals of that estimate. In this more realistic scenario, the framework of determining the expected value (m_i) and the standard deviation (s_i) of a target price return (Nix & Weigend, 1995) could be utilized to provide the network with two separate inputs. The problem is then broken down into two stages: the first is predicting m_i and s_i, and the second is determining α_i based on the predicted m_i and s_i. In this case, what are the values of α as a function of m_i and s_i? Intuitively, the best SR achievable cannot be as large as the value obtained in the noise-free experiment and we would expect α to decrease as s_i increases. The noisy price return data is our second experiment.

The two computer-generated data experiments will help us answer two fundamental questions: 1) the theoretical allocation and 2) the relationship of the allocation and the noise level in the data.

In order to generate the price return data for each day i, we draw x_i from a Gaussian random variable with parameters $\mu = m_i$ and $\sigma = s_i$ where m_i and s_i are themselves drawn from random variables. The mean on day i, m_i, is drawn from a random variable that is uniformly distributed over $(-\sqrt{3}, \sqrt{3})$ for both experiments. The standard deviation on day i, s_i, is set to 0 for the first experiment and is drawn from a random variable uniformly distributed over $(0, 2\sqrt{3})$ for the second experiment.

For both experiments, a 10-tanh hidden unit feed-forward network is trained on three different objective functions: SR, profit, and cross-entropy, and evaluated on four measures: SR, annual profit, annual standard deviation, and cross-entropy. This gives 12 combinations to analyze. 2000 "price returns" are generated and split into four even sets of 500 points for the training set and for each of the three test sets.

6.1 Noise-free data

Fig. 1 shows the training curves for the training and test sets vs. iterations (an iteration is defined as a weight update of a *single* pattern). The training curves

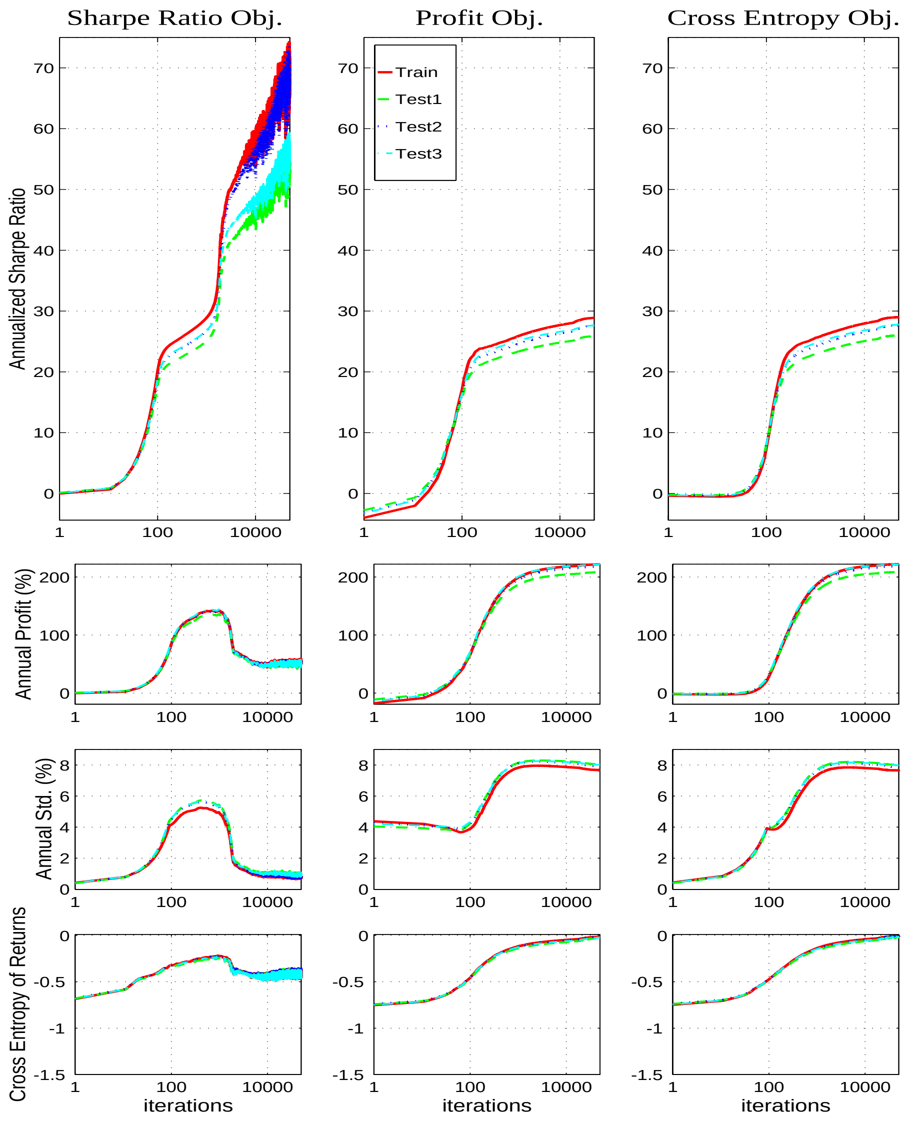

Figure 1: Training curves for the noise-free experiment. Each column of plots show results of *training* on each objective while each row of plots show the *evaluation* of each objective. The SR objective has a much lower annual profit as well as standard deviation (risk-reward tradeoff). In this experiment, the profit and cross-entropy objectives have similar training curves because the optimal α which maximizes their respective objectives are identical.

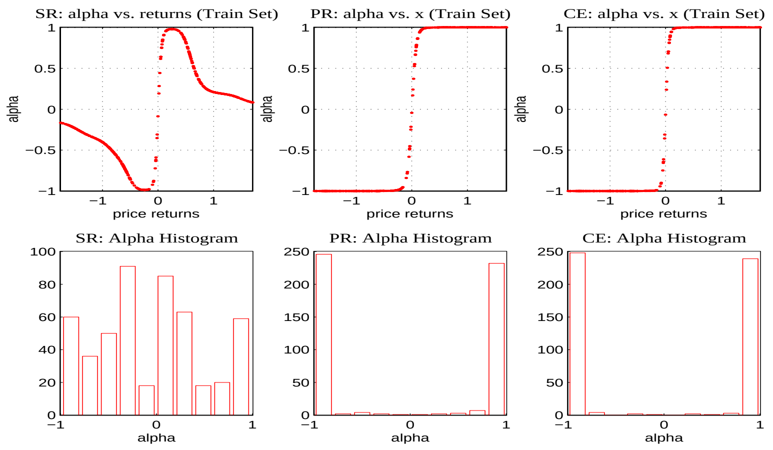

Figure 2: Noise-free data experiment: scatter plots of α vs. x and histograms of α for all objective functions. The top three plots from left to right are the plots for α vs. the price returns, x, for the SR, profit, and cross-entropy objectives while the bottom row of plots are the corresponding histograms of α. The top left plot shows that for the SR objective, abs(α) is a decreasing function of abs(x). The profit and cross-entropy have identical α vs. x plots because of the noise-free data.

for the SR objective are averaged over 10 patterns for better presentation in the plots. The training curves show that the SR objective's SR increases steadily, while its annual profit and standard deviation fluctuates. The profit and cross-entropy objectives' final values for all four evaluation measures are almost identical. Note that for the profit and cross-entropy objectives, the annual profit and annual standard deviation are much higher than the SR's corresponding values. This illustrates the SR objective's inclination to tradeoff large profit for smaller risk. Additionally, note that the cross-entropy obtains its maximum towards the end of training.

Inspection of each of the plots of α with respect to x in Fig. 2 provides insight into the theoretical allocations of each of the objectives when the data is noise-free. The top row of scatter plots shows α plotted versus the price returns x for each of the objective functions and the bottom row of plots shows the histogram of α for each of the objective functions.

For the profit objective, it is no surprise that the optimal $\alpha_i = \text{sign}(x_i)$, because the way to make the most profit is by taking full positions (long or short) based on the sign of x_i. Likewise, for the cross-entropy objective, optimal $\alpha_i = \text{sign}(x_i)$, because cross-entropy maximization implies finding the probability x_i is positive or negative. Since the data is noise-free, the probability is always 1.0 and therefore the α_i equals sign(x_i).

However, for the SR objective, the result is not straight-forward:

- For $x > 0$, α is a decreasing function of x towards 0.
- For $x < 0$, α is an increasing function of x towards 0.

Therefore, the optimal abs(α) decreases as an increasing function of abs(x). This relationship can be seen because the standard deviation is symmetrical: it is indifferent to the sign of the return and reveals one of the intrinsic weaknesses of SR as a performance measure (Sharpe, 1970).

6.2 Noisy Data

In the second experiment, s_i is no longer constant, but now drawn randomly, for each pattern, from a uniform distribution $(0, 2\sqrt{3})$. Both m_i and s_i are given to the network as inputs. How do the values of the respective objective functions compare to the first experiment, and how does α vary with respect to m_i and s_i? Fig. 3 shows the training curves vs. iterations for the second experiment. Similar shapes in the training curves can be observed, however the levels of all objective function values are significantly reduced, attributable to the increased noise level in the price returns.

The top row of scatter plots in Fig. 4 shows α plotted versus x for each of the objective functions, while the bottom row shows the histogram of α for each of the objective functions. The noise in the data smears the solid shapes exhibited in the plots of α vs. x for all of the objectives when compared to their theoretical counterparts in the noise-free experiment. However, the SR objective still has the tendency to take on moderate position sizes compared to the other objectives and is indicative of its risk-minimization behavior. Although the histogram of the profit objective α is still relatively binary, it is now not strictly a function of sign(x), which is caused by the noisy estimates. Although not as extreme as the profit objective, the cross-entropy objective takes larger position sizes relative to the SR objective.

Fig. 5 is a surface and contour plot of the SR objective's optimal α vs. m and s. The trend of increasing s is clear: abs(α) decreases towards 0, indicating that the best action is to take very small positions in the risky asset if s_i is relatively large (i.e., the estimate of x is noisy).

Summarizing the two experiments with computer-generated data, the SR objective performs well at adjusting its position sizes to avoid risk when compared to the other objectives because it is encouraged to decrease α during periods of large abs(x) and large noise levels. For the noise-free case, α decreases as abs(x) increases: profit is sacrificed for lower σ. For the noisy case,

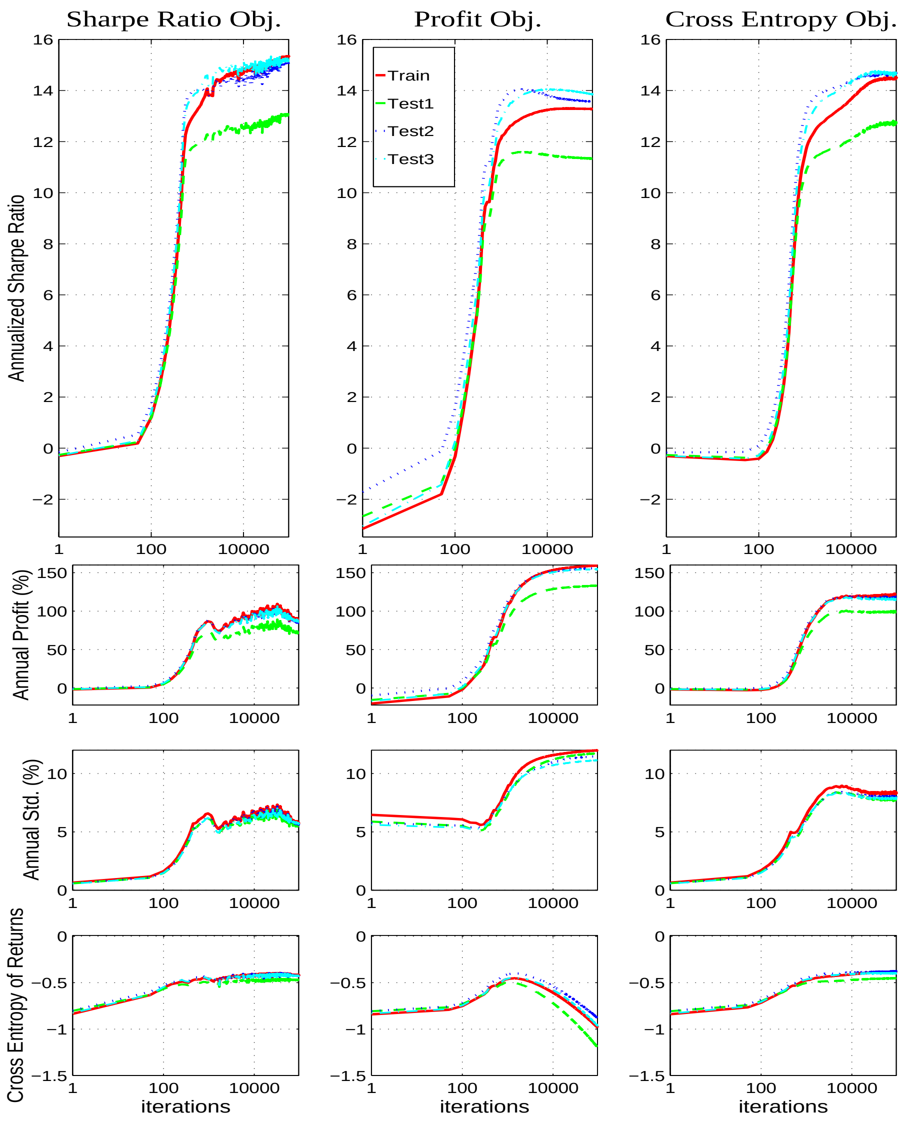

Figure 3: Training curves for the noisy computer generated data. Each column of plots shows results of *training* on each objective while each row of plots shows the *evaluation* of each objective. The SR objective has a higher SR than the profit objective and slightly higher than cross-entropy on all test sets at the end of training. As in the noise-free experiment, the SR objective has the lowest annual profit and standard deviation while the profit objective has the highest annual profit and standard deviation. One iteration denotes the presentation of a single pattern.

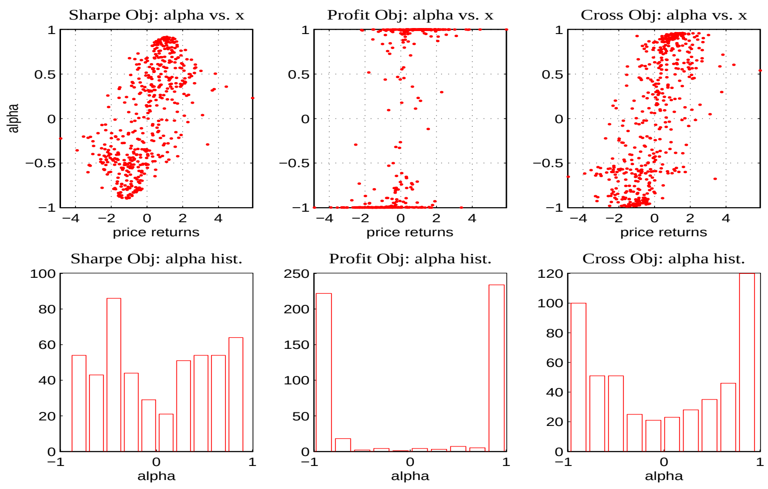

Figure 4: Noisy data experiment: α as a function of x and histograms of α for all objective functions. The top three scatter plots from left to right are the respective optimal α plotted vs. the price returns, x, for the SR, profit, and cross-entropy objectives. The bottom row of plots are the corresponding histograms. All plots of α vs. x have gone through significant changes and have become much more blurred when compared to the noise-free experiment. In the histograms, note that the profit and cross-entropy objectives still have the tendency to take binary position sizes as opposed to the SR objective which takes smaller position sizes to minimize risk.

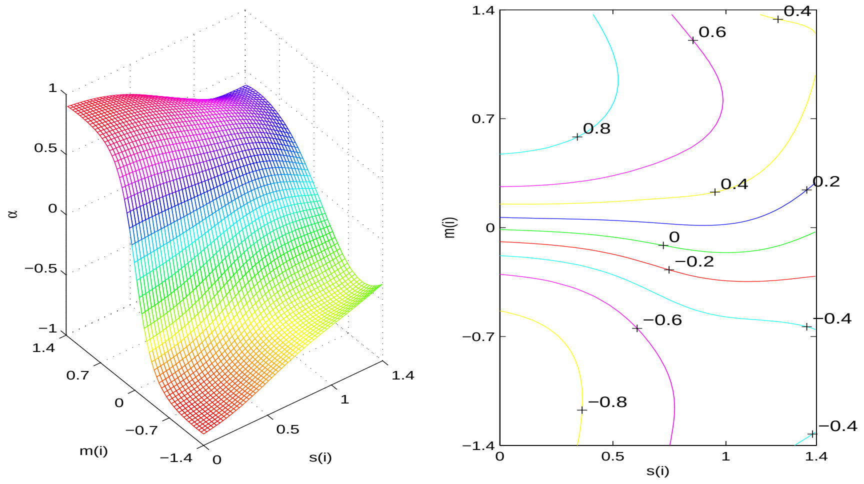

Figure 5: Noisy data experiment surface and contour plot of the SR's optimal α vs. m_i and s_i, the mean and standard deviation of the price returns, x_i. The main feature of the surface plot on the left is that for increasing s_i, abs(α) decreases: the more uncertain the estimate of x, the smaller the position size.

α decreases with increasing noise level: uncertainty in the value of x results in a decreasing position size.

7 Applications to German Stock Index Trading

As an application of our trading strategy to real world data, we use the German Stock Index DAX (Deutscher Aktien Index) from January 14, 1986 through June 29, 1996. Table 7 summarizes how the entire data set is split into a training and three test sets. Test set 2 is used as a tuning or cross-validation set to determine the point during training where the network starts to overfit on the training set.

Set	*Dates*	*Length*
Training set	November 29, 1988 to August 29, 1994	1500 days
Test set 1	December 15, 1987 to November 28, 1988	250 days
Test set 2 (tuning)	August 30, 1994 to August 14, 1995	250 days
Test set 3	August 15, 1995 to July 29, 1996	250 days

Table 2: Partitioning of the DAX data into a training set and three test sets. The dates and size of the data sets are given. Test set 2 is used as the tuning set.

We use 19 inputs: 5 inputs derived from the DAX Indicator index (30 stock index), 3 inputs derived from the DAX Composite (100-200 stocks), 2 inputs derived from the Morgan Stanley German Price Index, 2 inputs from the Dow-Jones Industrial Average, 2 inputs from the U.S. Dollar/Deutsche-Mark exchange rate, 1 input each for the 30-Year U.S. Bond, the price difference of Gold, the price difference of the Nikkei-225 index, and 2 inputs from the Morgan Stanley European Price Index. All networks have 15 tanh hidden units. We train networks with three objective functions: SR, profit, and cross-entropy. The networks are evaluated on four: SR, annual profit, annual standard deviation, and cross-entropy.

Fig. 6 shows the training curves for all objective functions. The three panels in the top row show the annualized SR for all objectives. The bottom set of nine plots shows the annual relative profit in percent, annual relative standard deviation in percent, and cross-entropy values for all objectives. The plots show that the SR objective overfits on the SR at around iteration 40,000. The SR objective is more prone to overfitting when compared to the other objectives.[b]

[b]The phenomenon that overfitting is particularly strong on the quantity that is optimized has been noted in other contexts before, e.g., Weigend (1994) discusses this in a classification context, and Weigend *et al.* (1995) give the example of a regression mixture model.

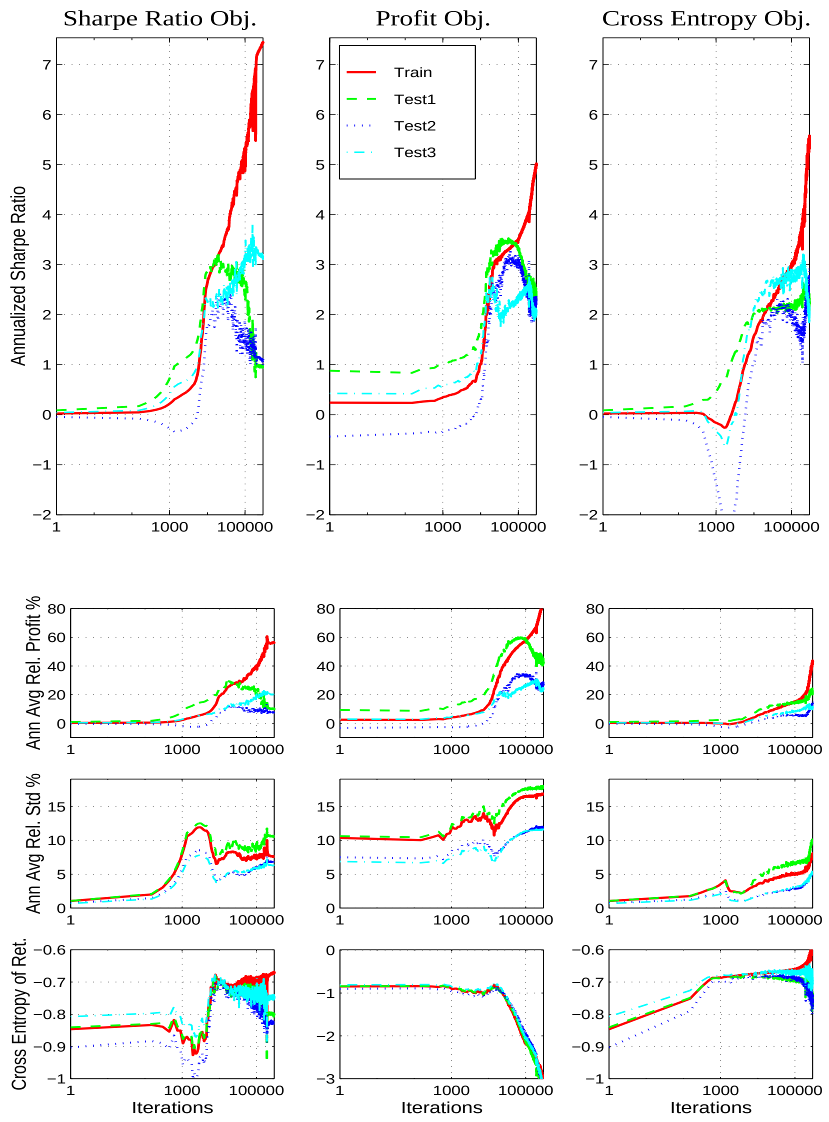

Figure 6: DAX training curves: each column of plots show results of *training* on each objective while each row of plots show the *evaluation* of each objective. The three objectives obtain fairly high SR's on test sets, but all achieve it differently. The SR objective has moderate profit and standard deviation while the profit objective has higher profit as well as standard deviation. The cross-entropy objective has the smallest profit and standard deviation of the three objectives.

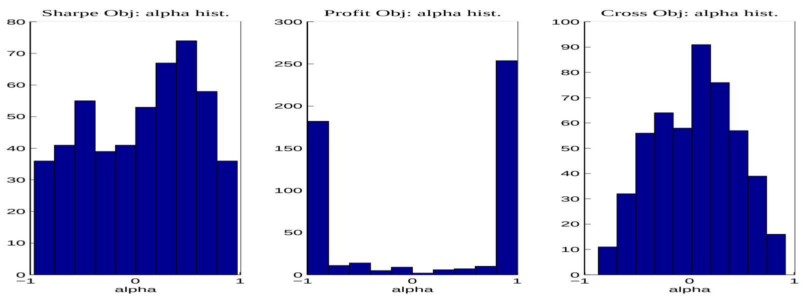

Figure 7: DAX histograms of α for all objective functions. The SR objective has the tendency to position sizes that are more evenly distributed compared to the profit and cross-entropy objectives. The profit objective for the most part takes full positions sizes which has as a result large profit as well as a large standard deviation of returns. The cross-entropy objective takes small position sizes which is indicative of the noisy data.

The SR of the test sets for all three objectives is similar, but achieved in different ways. The SR objective has a moderate profit and moderate standard deviation. The profit objective has the highest profit and highest standard deviation. Finally, the cross-entropy has the lowest profit and lowest standard deviation.

Fig. 7 illustrates the distribution of position sizes for each objective function. When compared to the profit objective, the SR objective takes moderate positions, which shows the effects risk minimization has on the position size. The cross-entropy objective has the tendency to take small position sizes which results in small profit and standard deviation.

Fig. 8 shows the profit and loss curves of all objective functions along with the simple buy-and-hold strategy. We use the network at iteration 40,000 for the SR objective, 192,000 for the profit objective, and 40,000 for the cross-entropy objective. The stopping points were determined by monitoring the validation set of their respective objectives. The plot shows that the SR objective obtains moderate profit when compared to the other two objectives which is as expected. Clearly, the profit objective should have the highest profit. However, the profit objective pays for high profit with the highest standard deviation. Not surprisingly, the SR objective achieves moderate profit and modest standard deviation while the cross-entropy has the lowest profit and standard deviation.

The profit and loss curves were computed without transaction costs. In trading, however, transactions costs can play an important role. With the common assumption of proportionality to the absolute value of the change in position size, i.e.,

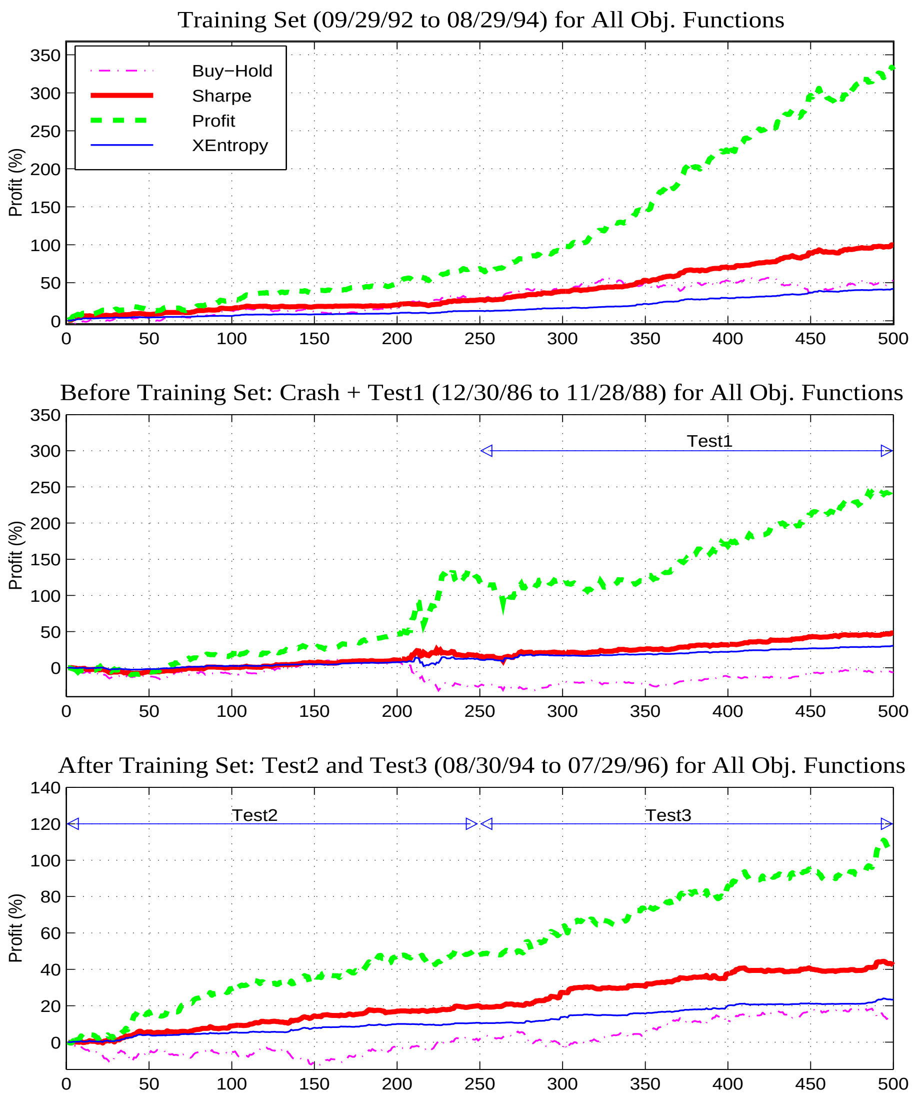

Figure 8: DAX P&L curves for the training and three test sets with dates as given in the plot titles. The network was evaluated at the overfitting point which was determined by test set 2. The center panel includes the US stock market crash of 1987 as the first 250 points for the profit evaluation. As expected, the profit objective does well in profit while the SR objective has moderate profit and the cross-entropy has very little profit above the simple buy-and-hold strategy.

$$\text{transaction cost} \propto \sum_{i=2}^{N} |\alpha_i - \alpha_{i-1}| \quad , \tag{27}$$

we find that the strategy generated by training on profit has significantly higher transaction costs than the strategy generated by training on SR, while the transaction costs for the network trained on cross-entropy is somewhat smaller:

$$\begin{aligned} \text{transaction cost (profit)} &\approx 1.8 * \text{transaction cost (SR)} \\ \text{transaction cost (cross-entropy)} &\approx 0.8 * \text{transaction cost (SR)} \quad . \end{aligned}$$

Revisiting Fig. 8, this will mean that the dashed line (profit) gets pulled down about 1.8 times as much as the thick solid line (SR). The thin solid line (cross-entropy) will be pulled down about 80 per cent as much. Only the dash-dotted line (buy-and-hold) remains unchanged, since there are no transactions.

8 Conclusion

We have presented a method for SR maximization using an iterative update rule and developed a trading strategy based on this method. The trading strategy is based on taking positions between two asset classes: risky and risk-free. We implemented an iterative update (gradient ascent) for a neural network architecture and tested the method on computer-generated and real-world data. The computer-generated data for the noise-free experiment shows how SR may not be an ideal risk-adjusted measure. The computer-generated data for the noisy experiment gives insight into how α is optimized in the presence of noise. The real-world data application to German Stock Index trading shows good results for the SR objective when compared to both the profit objective and cross-entropy, in terms of annual profit as well as standard deviation of returns. The technique of SR optimization is shown to be a feasible approach to explicitly consider both profit (average excess return) and risk (standard deviation of excess return) in a trading strategy. However, using standard deviation as the risk measure has the necessary limitation that positive returns are penalized. This problem can be addressed by using an asymmetrical risk-measure such as semideviation (Markowitz, 1962; Harlow, 1991) in trading applications.

Acknowledgments

We wish to thank Alan Yamamura for discussions in 1994, Jaewoo Kang for his contributions to the initial stages of this work in 1995, as well as Michelle Hsu, Tom McCabe, Ralph Neuneier, Shanming Shi, and Hans Georg Zimmermann. This work was supported by the National Science Foundation (ECS-9309786).

References

Buntine, W. L. and A. S. Weigend. 1991. Bayesian back-propagation. *Complex Systems* **5**, 603–643.

Hammer, D. A. 1991. *Dynamic Asset Allocation: Strategies for the Stock, Bond, and Money Markets.* John Wiley and Sons, New York.

Harlow, W. V. 1991. Asset allocation in a downside-risk framework. *Financial Analysts Journal* **47**, 28–40.

Markowitz, H. M. 1962. *Portfolio Selection: Efficient Diversification of Investments.* John Wiley and Sons, New York.

Nix, D. A. and A. S. Weigend. 1995. Learning local error bars for nonlinear regression. In *Advances in Neural Information Processing Systems 7 (NIPS*94)*, G. Tesauro, D. S. Touretzky and T. K. Leen (eds), pp. 488–496. MIT Press, Cambridge, MA.

Pi, H. and T. S. Rögnvaldsson. 1996. A neural network approach to futures trading. In *Neural Networks in Financial Engineering (Proceedings of the Third International Conference on Neural Networks in the Capital Markets, NNCM-95)*, A.-P. N. Refenes, Y. Abu-Mostafa, J. Moody and A. Weigend (eds), pp. 17–25, Singapore. World Scientific.

Rumelhart, D. E., R. Durbin, R. Golden and Y. Chauvin. 1996. Backpropagation: The basic theory. In *Mathematical Perspectives on Neural Networks*, P. Smolensky, M. C. Mozer and D. E. Rumelhart (eds), pp. 533–566. Lawrence Erlbaum Associates, Hillsdale, NJ.

Sharpe, W. F. 1970. *Portfolio Theory and Capital Markets.* McGraw-Hill, New York.

Sharpe, W. F. 1994. The Sharpe Ratio. *Journal of Portfolio Management* **21**, 49–58.

Weigend, A. S. 1994. On overfitting and the effective number of hidden units. In *Proceedings of the 1993 Connectionist Models Summer School*, M. C. Mozer, P. Smolensky, D. S. Touretzky, J. L. Elman and A. S. Weigend (eds), pp. 335–342, Hillsdale, NJ. Lawrence Erlbaum Associates.

Weigend, A. S., M. Mangeas and A. N. Srivastava. 1995. Nonlinear gated experts for time series: Discovering regimes and avoiding overfitting. *International Journal of Neural Systems* **6**, 373–399.

Zimmermann, H. G. and A. S. Weigend. 1997. Representing dynamical systems in feed-forward networks: A six layer architecture. In *Decision Technologies for Financial Engineering (Proceedings of the Fourth International Conference on Neural Networks in the Capital Markets, NNCM-96)*, A. S. Weigend, Y. S. Abu-Mostafa and A.-P. N. Refenes (eds), pp. 289–306, Singapore. World Scientific.