2003s-23

Stochastic Gradient Descent on a Portfolio Management Training Criterion Using the IPA Gradient Estimator

Christian Dorion, Yoshua Bengio

Série Scientifique
Scientific Series

Montréal
Mai 2003

CIRANO
Centre interuniversitaire de recherche
en analyse des organisations

CIRANO

Le CIRANO est un organisme sans but lucratif constitué en vertu de la Loi des compagnies du Québec. Le financement de son infrastructure et de ses activités de recherche provient des cotisations de ses organisations-membres, d'une subvention d'infrastructure du ministère de la Recherche, de la Science et de la Technologie, de même que des subventions et mandats obtenus par ses équipes de recherche.

CIRANO is a private non-profit organization incorporated under the Québec Companies Act. Its infrastructure and research activities are funded through fees paid by member organizations, an infrastructure grant from the Ministère de la Recherche, de la Science et de la Technologie, and grants and research mandates obtained by its research teams.

Les organisations-partenaires / The Partner Organizations

ISSN 1198-8177

Stochastic Gradient Descent on a Portfolio Management Training Criterion Using the IPA Gradient Estimator

Christian Dorion[*], *Yoshua Bengio*[†]

Résumé / *Abstract*

Dans cet article, nous jetons les bases pour l'apprentissage d'une stratégie de gestion d'un portefeuille de biens, de natures variées, et ne s'appuyant sur aucune supposition quant aux distributions des données financières. Ce modèle, basé sur l'utilisation d'un réseau de neurones, tente de capturer les tendances du marché. De plus, le modèle permet l'introduction d'un bruit stochastique au niveau des prix prévus par le réseau afin d'éviter les maxima locaux dans l'espace de décision. Dans ces conditions, nous démontrons que notre stratégie d'investissement suit un processus de décision markovien qui est presque sûrement lipchitzien en ses paramètres. Ainsi, l'estimateur du gradient IPA, obtenu ici par la méthode classique de rétropropagation, peut être utilisé pour approcher, par une descente de gradient, un maximum local de notre critère d'apprentissage, le Sharpe ratio.

Mots clés : Apprentissage, gestion de portefeuille, estimateur IPA, Sharpe ratio.

In this paper, we set the basis for learning a multitype assets portfolio management technique relying on no assumptions over the distributions of the financial data. The neural network based model tries to capture patterns in the evolution of the market. Furthermore, the model allows a stochastic perturbation in the asset pricing from the network to avoid local maxima in the decision space. Under those settings, we prove that our investment decision is a Markovian decision process which is Lipschitz continuous almost surely in its parameters. Therefore, the IPA gradient estimator, obtained here by the classical backpropagation algorithm, can be used in a gradient descent procedure to converge to a local maximum of our learning criterion, the Sharpe ratio.

Keywords: *Learning, portfolio management, IPA estimator, Sharpe ratio.*

* Département d'informatique et recherche opérationnelle, Université de Montréal, Québec, Canada, H3C 3J7. Email: dorionc@iro.umontreal.ca.

† CIRANO and Département d'informatique et recherche opérationnelle, Université de Montréal, Montréal, Québec, Canada, H3C 3J7, tel.: (514) 343-6804. Email: bengioy@iro.umontreal.ca.

1 Introduction

Adaptive systems such as neural networks proved to be a reliable tool in portfolio management (Weigend, Abu-Mostafa, and Refenes 1997). A classical role of neural networks is to predict chronological series (Weigend and Gershenfeld 1993).

In the past few years, theoretical arguments suggested that directly optimizing the financial criterion of interest should yield better performance, according to that same criterion, than optimizing an intermediate prediction criterion such as the often used mean squared error (Bengio 1997). Consequently, some experiments were led to confirm or invalidate those hypothesis.

In particular, (Chapados 2000) conducted a thorough comparison between the *forecasting* and the *decision* alternatives, performed within the *value-at-risk* control framework, and concluded that the forecasting model produces results statistically significantly better than those of the decision model. However, it was also remarked that for some configurations of the hyper-parameters, the model of decision provides the same performance as the forecasting model.

On the other hand, the decision model doesn't rely on the same idealist parametric assumptions on the data. In a portfolio combining stocks and options, this characteristic may be a major advantage of the new approach on the traditional models. In this work we try to optimize the Sharpe Ratio criterion with a stochastic gradient descent, known for being resistant to local minimums.

The main innovation in our framework is that our model let the Sharpe Ratio be a random variable which *takes the future in consideration* and that it is meant to introduce stochastic simulation concepts that, we hope, will help to get the model more resistant.

2 Suggested Model

The problem considered here is the management of assets in a *portfolio* . For simplicity, we will only consider the discrete time scenario, in which *a period* (e.g. a day or a month) is elapsed between times t and $t+1$, $t \in \mathbb{N}$. We refer to period t as being the period between times $t-1$ an t.

Definition 1 *A* **portfolio** $\mathbf{x}_t$ *defined with respect to a set of* N *assets is the vector of possessed quantity for each asset at a time* t *given:*

$$\mathbf{x}_t = (x_t^{(0)}, x_t^{(1)}, \ldots, x_t^{(N)})',$$

where $x_t^{(i)} \in \mathbb{R}$ *and* $-\infty < x_t^{(i)} < \infty$.

(We use bold letters for vectors or matrices; the $'$ represents the transpose operation.)

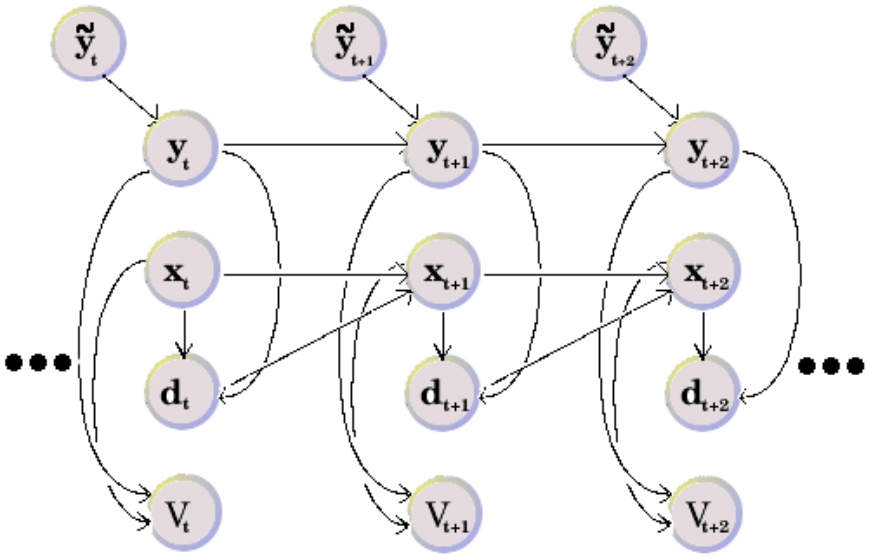

Figure 1: The Model

2.1 The Decision

Given an initial portfolio, we will *take some allocation decisions*, changing our *relative position* in the market. To do so, we assume there exists random process $\left\{\tilde{Y}_t\right\}_{t=0}^{\infty}$ describing the evolution of the market with filtration $\left\{\tilde{\mathcal{F}}_t : t \geq 0\right\}$[1]. By letting $\{Y_t\}_{t=0}^{\infty}$ be a random process such that $(\mathbf{x}_k, \tilde{\mathcal{F}}_k) \subset \mathcal{F}_t, \forall k \leq t$, we have

$$\mathrm{P}\left(Y_t \middle| \mathcal{F}_k, k = 0, \ldots, t-1\right) = \mathrm{P}\left(Y_t \middle| \mathcal{F}_{t-1}\right) \tag{1}$$

so that $\{Y_t\}$ is a *Markov process* (Billingsley 1995) on $(\Omega, \mathbb{F}, \mathbb{P})$[2]. At a given time t, $\mathbf{y}_t$ will denote a realization of Y_t.

We will consider our decision to be a stochastic function

$$\mathbf{d}_t(\mathbf{x}_t, \mathbf{y}_t) \sim P_\theta\left(\cdot \middle| \mathbf{x}_t, \mathbf{y}_t\right), \tag{2}$$

where θ are the parameters[3] of a neural network and $\mathbf{y}_t$ the information provided to it. It's important to note that even if our decision is such that $\mathbf{x}_{t+1} = \mathbf{x}_t$, the market itself evolved and, therefore, our *relative position* in the market changed.

Lemma 1 *If* $\{Y_t\}$ *is Markovian,* $\{(\mathbf{x}_t, Y_t)\}$ *is a Markovian Decision Process (MDP)*

In this work, the decision is based on the asset pricing

$$\mathbf{p}_t(\mathbf{x}_t, \mathbf{y}_t) = \boldsymbol{\mu}(\theta, \mathbf{x}_t, \mathbf{y}_t) + Z * \boldsymbol{\sigma}(\theta, \mathbf{x}_t, \mathbf{y}_t) \tag{3}$$

where $\boldsymbol{\mu}(\theta, \mathbf{x}_t, \mathbf{y}_t)$ and $\boldsymbol{\sigma}(\theta, \mathbf{x}_t, \mathbf{y}_t)$ are the outputs of the neural network (see Fig. 2) and Z is a random number from the standard normal distribution. We consider the $\mu_t^{(i)}(\theta, \mathbf{x}_t, \mathbf{y}_t)$ output as the expectation of the i^{th} asset price a time $t+1$, given $\mathcal{F}_t$, and $\sigma_t^{(i)}(\theta, \mathbf{x}_t, \mathbf{y}_t)$[4] as it uncertainty. The reader must note that this "gaussian-like" decision is simply a way to introduce noise on the decision. It's only meant to help us to explore the decision space more deeply and has no relation with any model on data.

To make our decision, we also need to consider $m_t^{(i)} \in \mathcal{F}_t$ the price of the i^{th} asset at time t, $C_0/2$ the additive cost and C_1 the multiplicative cost on transactions. Basically, we want to buy if $p_t^{(i)} - m_t^{(i)} > 0$[5], else we sell.

[1] Given a sample space Ω, a class $\mathcal{F}$ of subsets of Ω is called a *field* if $\Omega \subset \mathcal{F}$, $A \in \mathcal{F} \Rightarrow A^c \in \mathcal{F}$ and $A, B \in \mathcal{F} \Rightarrow A \cup B \in \mathcal{F}$. $\mathcal{F}\,(X)$, X a r.v., can be seen as the information needed for X to be *measurable* (Billingsley 1995)

[2] $\mathbb{F} = \{\mathcal{F}_t : t \geq 0\}$, the natural filtration of the $\{Y_t\}$ process. $\mathbb{P}$ a probability measure.

[3] Actually, θ is a *concatenation* of the **1)** inputs to hidden weights, **2)** hidden to outputs weights, **3)** bias on the hidden layer, **4)** bias on the output layer and **5)** two hardness parameters.

[4] In fact, $\sigma_t^{(i)}(\theta, \mathbf{x}_t, \mathbf{y}_t) = sigmoid(2^n doutput)$, to ensure positivity. The $\sigma_t^{(i)}(\theta, \mathbf{x}_t, \mathbf{y}_t)$ value is therefore bounded to 1, which is an adequate boundary financially speaking.

[5] From now on until the end of the decision analysis, we will hide the dependencies on θ, $\mathbf{x}_t$ and $\mathbf{y}_t$.

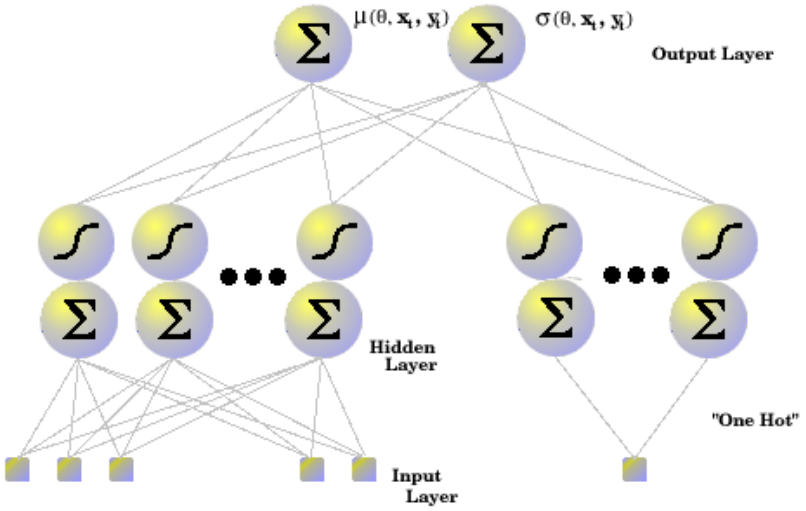

Figure 2: The Neural Network Topology

Definition 2 *The* **direction** *of our decision is*

$$\phi_t^{(i)} = tanh\left(\alpha(p_t^{(i)} - m_t^{(i)})\right)$$

where α *is an hardness (free) parameter*[6].

a continuous and differentiable function with $\lim_{\alpha\to\infty} \phi_t^{(i)} = sign(p_t^{(i)} - m_t^{(i)})$. Taking a decision in direction $\phi_t^{(i)}$, we hope for a valorization of our portfolio proportional to[7]

$$\Delta_t^{(i)} = p_t^{(i)}(1 - C_1\phi_t^{(i)}) - m_t^{(i)}(1 + C_1\phi_t^{(i)}) \tag{4}$$

Now, let γ be the capital level of the investor[8]. Then, we could trade a quantity

$$\tilde{d}_t^{(i)} = \gamma * \Delta_t^{(i)} \tag{5}$$

but the strategy is valid only if the expected benefits are greater than the cost of the transactions, that is if $\tilde{d}_t^{(i)}\Delta_t^{(i)} - C_0 > 0$, so our decision is

[6] Learned by the neural network.

[7] considering that we would have to trade the asset again to really benefit from the valorization.

[8] i.e is the the investor interested in investing thousands or millions of dollars, for example.

$$d_t^{(i)} = \tilde{d}_t^{(i)} \text{sigmoid}(\beta(\tilde{d}_t^{(i)}\Delta_t^{(i)} - C_0)) \tag{6}$$

where β is another hardness (free) parameter such that $\lim_{\beta\to\infty} d_t^{(i)} = \tilde{d}_t^{(i)}\mathcal{I}_{\{\tilde{d}_t^{(i)}\Delta_t^{(i)} - C_0 > 0\}}$

2.2 The Objective Function

Let $\{V_t\}$ be the random process on $(\Omega, \mathbb{F}, \mathbb{P})$ describing the value of the portfolio. At a given time t, V_t is a random variable that can be measured according to the information at time t, noted $\mathcal{F}_t$. An intuitive way to define the value is

Definition 3 *The* **value** *of a portfolio at time t is given by*

$$V_t = \sum_i x_t^{(i)} m_{t'}^{(i)}$$

where, given $t' \leq t$, $m_{t'}^{(i)}$ is last known[9] *value of the asset i.*

As we introduced above, we will train our learning algorithm according to the criterion of interest, in this experiment, the *Sharpe Ratio* over the returns $\{R_t\}$.

$$R_t = \frac{V_t - V_{t-1}}{V_{t-1}} \tag{7}$$

and we will consider an empirical[10] version of the Sharpe Ratio.

Definition 4 *The empirical* **Sharpe Ratio** *is defined by*

$$\mathcal{U}_\theta(\omega) = \frac{\overline{R}(\omega)}{s_{R(\omega)}}$$

where, $R_t(\omega)$ being a realization of R_t over the given period t, $\overline{R}(\omega)$ is the sample mean and $s_{R(\omega)}$ the sample standard deviation of the $R_t(\omega)$ over the trajectory ω.

Our goal is to maximize the expectation of that ratio, therefore

Definition 5 *Our* **objective function** *is*

$$J(\theta) \ = E[\mathcal{U}_\theta] = \int \mathcal{U}_\theta(\omega)\mathbb{P}(d\omega)$$

[9] There are days in the database where the values of some assets are not given. However, note that $d_t^{(i)}$=0 whenever $m_t^{(i)}$ is unknown, therefore $x_t^{(i)} = x_{t'}^{(i)}$ and the previous value is considered unchanged.

[10] Our Sharpe Ratio is, therefore a random variable.

3 The IPA Estimator

Obviously, our objective function is not available in closed analytical form, but we can consider each learning epoch as an independent trajectory and use the "sample average".

If we want to use a stochastic gradient descent algorithm (Bishop 1995), we need to know the *sensitivity* of $J(\theta)$ with respect to θ. That is

$$\frac{d}{d\theta}J(\theta) \ = \frac{d}{d\theta}\int \mathcal{U}_\theta(\omega)\mathbb{P}(d\omega) \tag{8}$$

and one may wonder if

$$\frac{d}{d\theta}\int \mathcal{U}_\theta(\omega)\mathbb{P}(d\omega) \stackrel{?}{=} \int \frac{d}{d\theta}\mathcal{U}_\theta(\omega)\mathbb{P}(d\omega) \tag{9}$$

that is if

$$\lim_{\Delta\theta\to 0}\frac{E[\mathcal{U}_{\theta+\Delta\theta}]-E[\mathcal{U}_\theta]}{\Delta\theta} \stackrel{?}{=} E\left[\lim_{\Delta\theta\to 0}\frac{\mathcal{U}_{\theta+\Delta\theta}-\mathcal{U}_\theta}{\Delta\theta}\right] \tag{10}$$

Equations 8 and 9 are known as the pathwise analysis approach regrouping IPA and SPA methods, among others. The pathwise approach for sensitivity analysis is particularly suitable in our application, because we do not assume knowledge of the underlying probability measure of the driving process $\{\tilde{y}_n\}$.

The IPA method is simply to take the estimator given by Eq. 9 whenever the context allows the application of the following theorem.

Theorem 1 *(***Dominated Convergence** *(Billingsley 1995)) Let* $\{\Psi_n, n\in\mathbb{N}\}$*, and let* Ψ *and each* Ψ_n *be a random variable on* $(\Omega,\mathbb{F},\mathbb{P})$*. If* $\Psi_n\to\Psi$ *a.s. and if there is a random variable* K *such that* $E[K]<\infty$ *and* $|\Psi_n|\leq K, \forall n\in\mathbb{N}$*, then* $\lim_{n\leftarrow\infty}E[\Psi_n]=E[\Psi]$*.*

If we want to apply it in our context, it may be useful to remind ourselves of the Lipschitz continuity property

Definition 6 *A random variable* $\Upsilon_\theta(\omega)$ *on* $(\Omega,\mathbb{F},\mathbb{P})$ *is said to be* **Lipschitz continuous a.s.** *in* θ *if there is a random variable* $K<\infty$ *with* $E[X]<\infty$ *such that* $\forall\omega$*,*

$$\sup_{\theta\in\Theta:\theta+\Delta\theta\in\Theta}\|\Upsilon_{\theta+\Delta\theta}(\omega)-\Upsilon_\theta(\omega)\|\leq K(\omega)\Delta\theta$$

The reader may note that if $\Upsilon_\theta(\omega)$ is differentiable, the above is nothing much than

$$\frac{d}{d\theta}\Upsilon_\theta(\omega)\leq K(\omega) \tag{11}$$

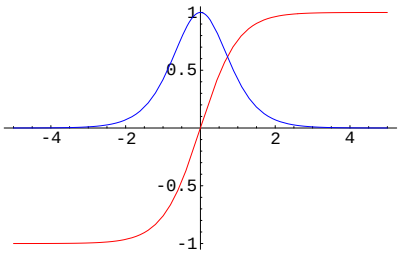

Figure 3: Tanh Function (in red) and its Derivative Sech^2 (in blue)

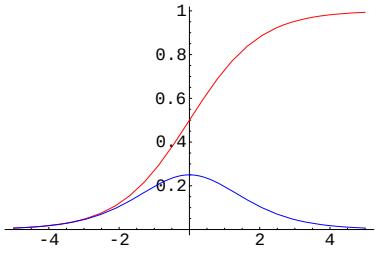

Figure 4: Sigmoid Function (in red) and its Derivative (in blue)

We'll now argue that it is verified under our assumptions. Indeed, we could first take a look to the neural network output functions of θ,

$$\mathbf{f}_k(\theta, \mathbf{x}_t, \mathbf{y}_t) = \sum_{j=1}^{n_H} \theta_{kj} tanh\left(\sum_{i=1}^{d} \theta_{ji} y_t^{(i)} + \theta_{j0}\right) + \theta_{k0} \tag{12}$$

where k=1 stands for $\boldsymbol{\mu}(\theta, \mathbf{x}_t, \mathbf{y}_t)$, k=2 for $\boldsymbol{\sigma}(\theta, \mathbf{x}_t, \mathbf{y}_t)$, n_H is the number of neurons in the hidden layer and d is the dimension of $\mathbf{y}_t$. It's easy to show that that the previous are Lipschitz continuous functions of θ. In our model, $\boldsymbol{\mu}(\theta, \mathbf{x}_t, \mathbf{y}_t)$ and $\boldsymbol{\sigma}(\theta, \mathbf{x}_t, \mathbf{y}_t)$ are then use in the computation of a decision function whose behavior with respect to θ mainly depends on the behavior of the tanh and sigmoid functions. Well, those functions, as can be seen on Figures 3 and 4, have bounded derivatives. Therefore, our decision function is Lipschitz continuous in θ. Finally, the Sharpe ratio is a simple composition of linear functions the decisions, so we can conclude, from the following lemma, that $\mathcal{U}_\theta$ is Lipschitz continuous almost surely in θ.

Lemma 2 *If Υ_θ is a Lipschitz continous a.s. random variable and $f : \mathbb{R} \to \mathbb{R}$ is a function such that $\exists f'(x) \leq b, \forall x$, then $f(\Upsilon_\theta)$ is a Lipschitz continous a.s. random variable.*

Since we have Lipschitz continuity a.s., the dominated convergence theorem applies and we may, therefore, perform a stochastic gradient descent and use backpropagation to evaluate the $\frac{d}{d\theta}\mathcal{U}_\theta$ IPA estimator.

It must be noted that stochastic approximation with non-biased estimators ensure, under certain conditions, that θ will converge to the set of stability points of the ODE implicit to the maximization. However, unless we have the concavity of of the cost function, which is clearly not the case here, those points are more likely to be local minimums than global.

4 Experimental Setup

At the moment, we have a database containing options and indexes prices (open, high, low and close prices) for the last 30 years or so. Given those prices, we try to extract relevant and predictive informations for each option and index. Once this preprocessing done, we train the network (see Fig. 2) according to to the Sharpe Ratio criterion.

4.1 Raw Data

The database can be seen as a matrix of trade data having the following form:
raw_data[row][column]
where

- There is one row for each (time, asset ticker) pair.
- The columns contain:
  - The time.
  - The asset ticker
  - The asset price
  - If the asset is an option:
    - The expiration date
    - The strike
    - The underlying asset ticker

Preprocessing

In a first preprocessing, the raw data are preprocessed to give us a new matrix structure

data[row][column]
where

- There is one row for each (time, **option** ticker) pair.
- The columns contain:
  - The time: t_1 (in days since the, chosen, January 2^{nd} 1990)
  - The expiration date: T
  - The price at t_1: ${c_1}$[11]
  - The price at $t_1 - 1$: ct_1
  - The geometric mean of the option between $t_1 - 2$ and $t_1 - 1$: ct_2
  - The geometric mean of the option between $t_1 - 5$ and $t_1 - 1$: ct_5
  - The geometric mean of the option between $t_1 - 10$ and $t_1 - 1$: ct_{10}
  - The geometric mean of the option between $t_1 - 20$ and $t_1 - 1$: ct_{20}
  - The strike: K
  - The price of the underlying stock at t_1: s_1
  - The geometric mean of the underlying stock between $t_1 - 2$ and $t_1 - 1$: st_2
  - The geometric mean of the underlying stock between $t_1 - 5$ and $t_1 - 1$: st_5
  - The geometric mean of the underlying stock between $t_1 - 10$ and $t_1 - 1$: st_{10}
  - The geometric mean of the underlying stock between $t_1 - 20$ and $t_1 - 1$: st_{20}
  - The price of the underlying stock at T: s_T
  - The interest rate at t_1: r_1
  - The underlying stock ticker
  - The option ticker

and then processed back to get preprocessed form

[11] which is, in the terminology of section 2.1, $m_t^{(i)}$

preprocessed_data[row][column]
where

- There is one row for each (time, **option** ticker) pair.
- The columns contain:
  - t_1
  - T-t1
  - r_1
  - $\log(c_1/K)$
  - $\log(s_1/K)$
  - $\log(c_1/ct_1)$
  - $\log(c_1/ct_2)$
  - $\log(c_1/ct_5)$
  - $\log(c_1/ct_{10})$
  - $\log(c_1/ct_{20})$
  - $\log(s_1/st_1)$
  - $\log(s_1/st_2)$
  - $\log(s_1/st_5)$
  - $\log(s_1/st_{10})$
  - $\log(s_1/st_{20})$
  - st5: Will be used as scale factor (see section 2.1)
  - c_1
  - If option is a call, $\max(0,(s_1 - K))$, else (put), $\max(0,(K - s_1))$: A lower bound on option price
  - The underlying stock ticker
  - The option ticker

The second preprocessing is meant to get the informations more predictive than the first. Before they can be used, data must be scanned to extract some general informations such as:

- The number of different stocks underlying to the options
- The number of different options in the database
- The columnwise mean and standard deviation of the data matrix

where the mean and standard deviation are used to normalize the columns that will be inputs of the neural network.

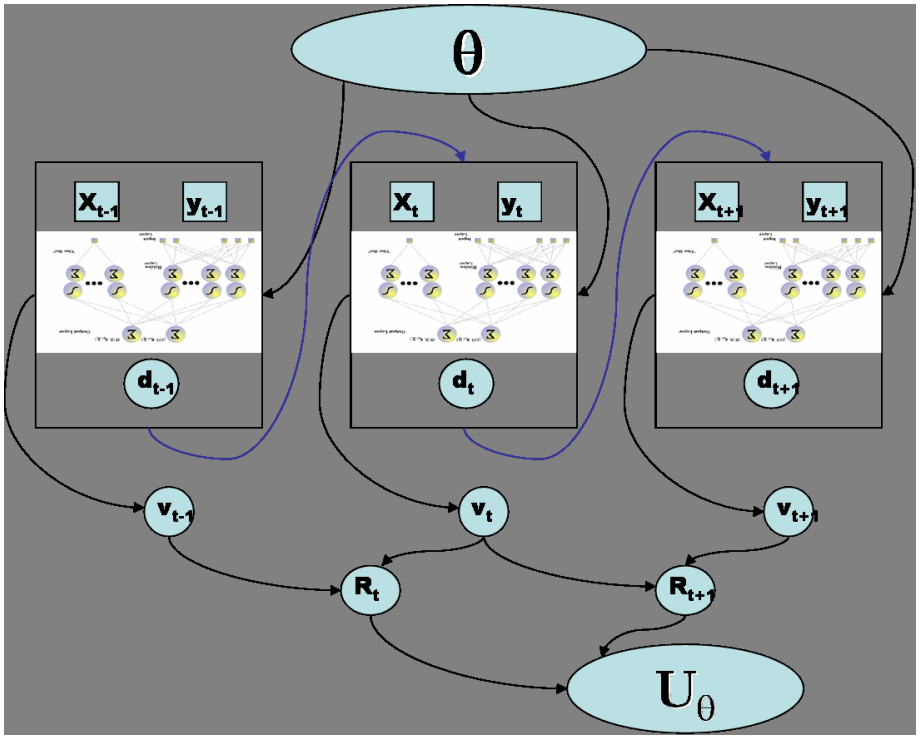

Figure 5: The Implementation (see Figures 1 and 2 for explicit dependencies)

Finally, the data matrix is split in three sub-matrices keeping the first 70% of the data for the train, letting the following 15% for the validation and the other 15% for the test phase.

4.2 The Implementation

The implementation is done with *C++* and uses the powerful tools of the *PLearn*[12] library, in particular the *Var* class[13]. Basically, the Var hierarchy provides an easy and quite efficient way to establish dependencies between component of a complicated mathematical model. Furthermore, it provides tools to easily propagate to the modification to a source variable to all of its descendant and backpropagate the gradient in the other way.

With those Vars as building blocks, we construct a Var graph from θ to the Sharpe Ratio variable, for each of the training, validation and testing phase. As can be seen on the Fig. 5, there is only one *node* θ on which depends all *time blocks*. As mentioned before, θ is simply a *concatenation* of the **1)** inputs to hidden weights, **2)** hidden to outputs weights, **3)** bias on the hidden layer, **4)** bias on the output layer and **5)** two hardness parameters.

Each of those time blocks consists in a Decision Network, which implements mathematical dependencies of section. Given an info variable $\mathbf{y}_t$ and the current portfolio $\mathbf{x}_t$, it computes a decision d_t from the outputs of a neural network. The values of $\mathbf{y}_t$ and $\mathbf{x}_t$ also allows us to compute the value v_t according to Def. 3. The block $t+1$ depends on the block t in the fact that $\mathbf{x}_{t+1}$ is a binary variable of $\mathbf{x}_t$ and $\mathbf{d}_t(\mathbf{x}_t, \mathbf{y}_t)$, in fact $\mathbf{x}_{t+1} = \mathbf{x}_{t+1} + \mathbf{d}_t(\mathbf{x}_t, \mathbf{y}_t)$.

Once the time dependencies are build, the following statistics can be computed:

[12] http://plearn.sourceforge.net/

[13] http://plearn.sourceforge.net/LibraryReference/html/Variable_h-source.html

$$R_t = \frac{V_t - V_{t-1}}{V_{t-1}} \tag{13}$$

$$\overline{R}(\omega) = \frac{1}{N}\sum_{t=1}^{N} R_t;\ s_{R(\omega)} = \left[\left(\frac{1}{N}\sum_{t=1}^{N} {R_t}^2\right) - \overline{R}(\omega)^2\right]^{1/2} \tag{14}$$

$$\mathcal{U}_\theta(\omega) = \frac{\overline{R}(\omega)}{s_{R(\omega)}} \tag{15}$$

The reader may wonder why the settings involve an instantiation of neural network for each $\mathbf{y}_t$ in the dataset, which approach is quite unusual. The thing is that the objective function depends on values of the portfolio computed at each time step and the backpropagation on those variables enforce the need to keep the network in memory. This presently causes a problem which will be discussed in §5.2.

5 Results and discussion

Figure 6 shows the results obtained with different learning rates, in terms of improvement in the Sharpe Ratio (vertical axis). The horizontal axis is the number of training epochs.

5.1 Gradient Descent

We explored stochastic gradient descent and observed that it increases the Sharpe ratio value very slowly even with the best found learning rate. Therefore, we are currently exploring the conjugate gradient descent and waiting for results. At the moment, it seems to get caught very rapidly in local minimums and the code must be deeply reviewed to find a way to avoid it. A simple omission in the propagation path could cause the function to vary abnormally (or not to vary).

However, once those practical issues solved, theoretical issues hold. That is, both algorithm proceed by first finding the propagation path from θ to $\mathcal{U}_\theta$ and then build the IPA estimator $\frac{d}{d\theta}\mathcal{U}_\theta$ of the gradient (chain rule on the propagation path). Then we can update the gradient:

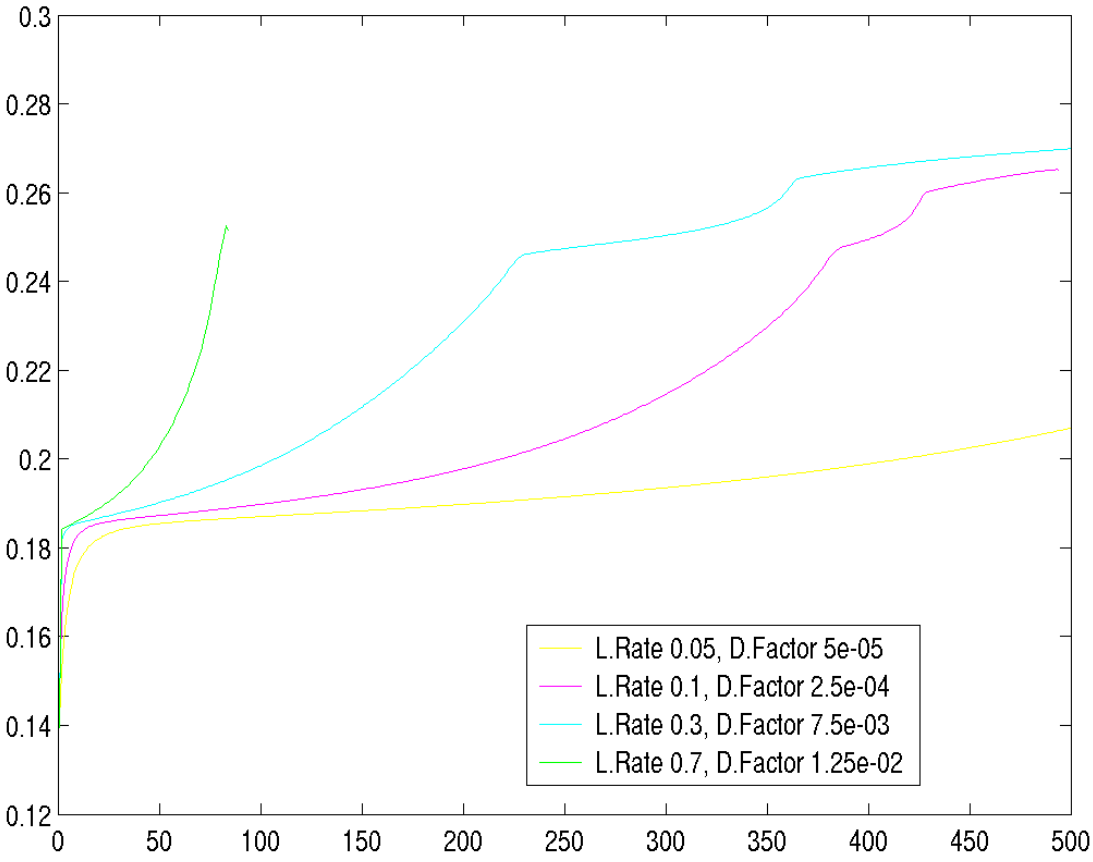

Figure 6: First Learning Rate Selection With 100 Hidden Units and 10^{-6} as Weight Decay

$$\theta \leftarrow \theta + \eta \frac{d}{d\theta} \mathcal{U}_\theta \tag{16}$$

and hope to converge to

$$\theta^* \ s.t. \ E\left[\mathcal{U}_{\theta^*}\right] = \max_\theta E\left[\mathcal{U}_\theta\right] \tag{17}$$

where η may be the learning rate of the stochastic gradient descent or the step found by the conjugate gradient algorithm.

5.2 The Memory Limit

One of the major problems in that model is the cost of the time dependencies. Indeed, giving a brief look at the implementation Fig. 5, one can see that each time block needs the results of the previous one in entry. The thing is that, θ being common to all the networks, all blocks must be kept in memory to propagate their values and to be backpropagated on. Obviously, that construction increases rapidly the need of RAM memory with the number of entry.

It is well know that predictions on options related to few underlying assets are less likely to provide good results than predictions on a large set of spots. Under the current settings, we are not able to train on more than 1164 options related to 5 underlying stocks since the 1G of RAM of the machines are exceeded with more data. Previous models, without time dependencies, could easily train over 60 different stocks underlying nearly 50000 options.

6 Conclusion

On the whole, the project as produced no concrete results, but as it is part of a master project, the effort wasn't vain and the work will continue. In particular, the costful time dependencies may be reengineered. At first glance, a solution could be to implement a *time windows* structure, that is to break to dependencies after a certain time. There would be some adjustment to do on our way to perform the gradient descent, but this is one of the avenue we will explore in the next months.

Another way to reduce the RAM needed can be to develop specific *Var* tools for the project. The reader should remind himself that the current implementation uses the existing tools of the *PLearn* library. The thing is that those tools were not designed specifically for our problem and we must often use a combination of many of those tools to perform some operation on the data. Writing specific Var could help reduce the need in memory, but that hypothesis has to be confirmed.

Finally, a challenging avenue we would also like to explore in the next months is the possibility to initialize the weights (θ) of the neural networks with values learned on a simpler model which already gave interesting results.

References

Bengio, Y. (1997). Using a Financial Training Criterion Rather than a Prediction Criterion. *International Journal of Neural Systems*.

Billingsley, P. (1995). *Probability and Measure*. New York: Wiley.

Bishop, C. (1995). *Neural Networks for Pattern Recognition*. London, UK: Oxford University Press.

Chapados, N. (2000). Critères d'optimisation d'algorithmes d'apprentissage en gestion de portefeuille. Master's thesis, Département d'informatique et de recherche opérationnelle, Université de Montréal, Montréal, Canada.

Weigend, A. and N. Gershenfeld (Eds.) (1993). *Time Series Prediction: Forecasting the future and understanding the past*. Addison-Wesley.

Weigend, A. S., Y. Abu-Mostafa, and A.-P. Refenes (Eds.) (1997). *Decision Technologies for Financial Engineering: Prooceedings of the Fourth International Conference on Neural Networks in the Capital Markets (NNCM '96)*. World Scientific Publishing.