

2002s-49

Cost Functions and Model Combination for VaR-based Asset Allocation using Neural Networks

Nicolas Chapados, Yoshua Bengio

Série Scientifique
Scientific Series



CIRANO
Centre interuniversitaire de recherche
en analyse des organisations

Montréal
Mai 2002

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

- École des Hautes Études Commerciales
- École Polytechnique de Montréal
- Université Concordia
- Université de Montréal
- Université du Québec à Montréal
- Université Laval
- Université McGill
- Ministère des Finances du Québec
- MRST
- Alcan inc.
- AXA Canada
- Banque du Canada
- Banque Laurentienne du Canada
- Banque Nationale du Canada
- Banque Royale du Canada
- Bell Canada
- Bombardier
- Bourse de Montréal
- Développement des ressources humaines Canada (DRHC)
- Fédération des caisses Desjardins du Québec
- Hydro-Québec
- Industrie Canada
- Pratt & Whitney Canada Inc.
- Raymond Chabot Grant Thornton
- Ville de Montréal

© 2002 Nicolas Chapados et Yoshua Bengio. Tous droits réservés. *All rights reserved.* Reproduction partielle permise avec citation du document source, incluant la notice ©.
Short sections may be quoted without explicit permission, if full credit, including © notice, is given to the source.

Les cahiers de la série scientifique (CS) visent à rendre accessibles des résultats de recherche effectuée au CIRANO afin de susciter échanges et commentaires. Ces cahiers sont écrits dans le style des publications scientifiques. Les idées et les opinions émises sont sous l'unique responsabilité des auteurs et ne représentent pas nécessairement les positions du CIRANO ou de ses partenaires.

This paper presents research carried out at CIRANO and aims at encouraging discussion and comment. The observations and viewpoints expressed are the sole responsibility of the authors. They do not necessarily represent positions of CIRANO or its partners.

Cost Functions and Model Combination for VaR-based Asset Allocation using Neural Networks

Nicolas Chapados^{*} and Yoshua Bengio[†]

Résumé / Abstract

Nous introduisons un cadre d'allocation d'actifs basé sur le contrôle actif de la valeur à risque d'un portefeuille. À l'intérieur de ce cadre, nous comparons deux paradigmes pour faire cette allocation à l'aide de réseaux de neurones. Le premier paradigme utilise le réseau de neurones pour faire une prédiction sur le comportement de l'actif, en conjonction avec un allocateur traditionnel de moyenne-variance pour la construction du portefeuille. Le deuxième paradigme utilise le réseau pour faire directement les décisions d'allocation du portefeuille. Nous considérons une méthode qui accomplit une sélection de variable douce sur les entrées, et nous montrons sa très grande utilité. Nous utilisons également des méthodes de combinaison de modèles (comité) pour choisir systématiquement les hyper-paramètres pendant l'entraînement. Finalement, nous montrons que les comités utilisant les deux paradigmes surpassent de façon significative les performances d'un banc d'essai du marché.

We introduce an asset-allocation framework based on the active control of the value-at-risk of the portfolio. Within this framework, we compare two paradigms for making the allocation using neural networks. The first one uses the network to make a forecast of asset behavior, in conjunction with a traditional mean-variance allocator for constructing the portfolio. The second paradigm uses the network to directly make the portfolio allocation decisions. We consider a method for performing soft input variable selection, and show its considerable utility. We use model combination (committee) methods to systematize the choice of hyperparameters during training. We show that committees using both paradigms are significantly outperforming the benchmark market performance.

Mots clés : Valeur à risque, allocation d'actif, critère de performance financière, combinaison de modèles, réseau de neurones récurrents multi-couches.

Keywords: *Value-at-risk, asset allocation, financial performance criterion, model combination, recurrent multilayer neural networks*

^{*} CIRANO and Département d'informatique et recherche opérationnelle, Université de Montréal, Montréal, Québec, Canada, H3C 3J7. Email: chapados@iro.umontreal.ca

[†] CIRANO and Département d'informatique et recherche opérationnelle, Université de Montréal, Montréal, Québec, Canada, H3C 3J7. Tel: +1 (514) 343-6804, email: bengioy@iro.umontreal.ca

I. INTRODUCTION

In finance applications, the idea of training learning algorithms according to the criterion of interest (such as profit) rather than a generic prediction criterion, has gained interest in recent years. In asset-allocation tasks, this has been applied to training neural networks to directly maximize a Sharpe Ratio or other risk-adjusted profit measures [1–3].

One such risk measure that has recently received considerable attention is the *value-at-risk* (VaR) of the portfolio, which determines the maximum amount (usually measured in e.g. \$) that the portfolio can lose over a certain period, with a given probability.

Although the VaR has been mostly used to estimate the risk incurred by a portfolio [4], it can also be used to actively control the asset allocation task. Recent applications of the VaR have focused on extending the classical Markowitz mean–variance allocation framework into a mean–VaR version; that is, to find an efficient set of portfolios such that, for a given VaR level, the expected portfolio return is maximized [5, 6].

In this paper, we investigate training a neural network according to a learning criterion that seeks to maximize profit under a VaR constraint, while taking into account transaction costs. One can view this process as enabling the network to *directly learn* the mean–VaR efficient frontier, and use it for making asset allocation decisions; we call this approach the **decision model**. We compare this model to a more traditional one (which we call the **forecasting model**), that uses a neural network to first make a forecast of asset returns, followed by a classical mean–variance portfolio selection and VaR constraint application.

II. VALUE AT RISK

A. Assets and Portfolios

In this paper, we consider only the discrete-time scenario, where one *period* (e.g. a week) elapses between times $t - 1$ and t , for $t \geq 0$ an integer. By convention, the t -th period is between times $t - 1$ and t .

We consider a set of N assets that constitute the basis of our portfolios. Let \mathbf{R}_t be the random vector of simple asset returns obtained between times $t - 1$ and t . We shall denote a specific realization of the returns process—each time made clear according to context—by $\{\mathbf{r}_t\}$.

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

$$\mathbf{x}_t = (x_{1t}, x_{2t}, \dots, x_{Nt})', \quad (1)$$

where $x_{it} \in \mathbb{R}$ and $-\infty < x_{it} < \infty$.

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

The amounts x_{it} are chosen causally: they are a function of the information set available at time t , which we denote by \mathcal{I}_t . These amounts do not necessarily sum to one; they represent the net position (in e.g. \$) taken in each asset. Short positions (negative x_{it}) are allowed.

The **total return** of the portfolio \mathbf{x}_{t-1} during the period t is given by $R_t = \mathbf{x}_{t-1}' \mathbf{R}_t$.

B. Defining Value at Risk

Definition 2: The **value-at-risk** (VaR) with probability α of the portfolio \mathbf{x}_{t-1} over period t is the value $V_t \geq 0$ such that:

$$\Pr[\mathbf{R}_t' \mathbf{x}_{t-1} < -V_t \mid \mathcal{I}_{t-1}] = 1 - \alpha. \quad (2)$$

The VaR of a portfolio can be viewed as the maximal loss that this portfolio can incur with a given probability α , for a given period of time. The VaR reduces the risk to a single figure: the maximum amount V_t that the portfolio can lose over one period, with probability α .

C. The Normal Approximation

The value at risk V_t of a portfolio \mathbf{x}_{t-1} is not a quantity that we can generally measure, for its definition (2) assumes a complete knowledge of the conditional distribution of returns over period t . To enable calculations of the VaR, we have to rely on a *model* of the conditional distribution; the model that we consider is to approximate the conditional distribution of returns by a normal distribution.¹

¹This model supposes that the portfolio return can be reasonably well approximated by a normal distribution, which is the case for, e.g., stock portfolios on relatively long horizons; however, this approximation loses its validity for many types of derivative securities, including options, or short-horizon portfolios. Furthermore, several financial assets are known to exhibit “fat-tailed” distributions, in which cases the normal approximation may underestimate the true value at risk of a portfolio. The asset return distribution can of course be estimated from empirical data, using kernel methods [7] or neural networks [8], albeit at the cost of making further VaR analysis more complex. The results that we present in this paper rely on the normal approximation, since we are dealing with long-horizon (monthly) stock portfolios.

C.1 One-Asset Portfolio

Let us for the moment consider a single asset, and assume that its return distribution over period t , conditional on \mathcal{I}_{t-1} , is

$$R_t \sim \mathcal{N}(\mu_t, \sigma_t^2), \quad \sigma_t^2 > 0, \quad (3)$$

which is equivalent to

$$\Pr[R_t < r_t \mid \mathcal{I}_{t-1}] = \Phi\left(\frac{r_t - \mu_t}{\sigma_t}\right), \quad (4)$$

where $\Phi(\cdot)$ is the cumulative distribution function of the standardized normal distribution, and μ_t and σ_t^2 are respectively the mean and variance of the conditional return distribution.

According to this model, we compute the α -level VaR as follows: let x_{t-1} be the (fixed) position taken in the asset at time $t-1$. We choose $r_t = \sigma_t \Phi^{-1}(1-\alpha) + \mu_t$ that we substitute in the above equation, to obtain

$$\Pr[R_t < \sigma_t \Phi^{-1}(1-\alpha) + \mu_t \mid \mathcal{I}_{t-1}] = 1 - \alpha, \quad (5)$$

whence

$$\Pr[R_t x_{t-1} < (\sigma_t \Phi^{-1}(1-\alpha) + \mu_t) x_{t-1} \mid \mathcal{I}_{t-1}] = 1 - \alpha, \quad (6)$$

and, comparing eq. (2) and (6),

$$\begin{aligned} V_t &= -(\sigma_t \Phi^{-1}(1-\alpha) + \mu_t) x_{t-1} \\ &= (\sigma_t \Phi^{-1}(\alpha) - \mu_t) x_{t-1}, \end{aligned} \quad (7)$$

using the fact that $\Phi^{-1}(1-\alpha) = -\Phi^{-1}(\alpha)$ from the symmetry of the normal distribution.

C.2 Estimating V_t

Let $\hat{\mu}_t$ and $\hat{\sigma}_t$ be estimators of the parameters of the return distribution, computed using information \mathcal{I}_{t-1} . (We discuss below the choice of estimators.) An estimator of V_t is given by:

$$\hat{V}_t = (\hat{\sigma}_t \Phi^{-1}(\alpha) - \hat{\mu}_t) x_{t-1}. \quad (8)$$

If $\hat{\mu}_t$ and $\hat{\sigma}_t$ are unbiased, \hat{V}_t is also obviously unbiased.

C.3 N -Asset Portfolio

The previous model can be extended straightforwardly to the N -asset case. Let the conditional distribution of returns be

$$\mathbf{R}_t \sim \mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\Gamma}_t), \quad (9)$$

where $\boldsymbol{\mu}_t$ is the vector of mean returns, and $\boldsymbol{\Gamma}_t$ is the covariance matrix of returns (which we assume is positive-definite). Let \mathbf{x}_{t-1} the fixed positions taken in the assets at time $t-1$. We find the α -level VaR of the portfolio for period t to be

$$V_t = \Phi^{-1}(\alpha) \sqrt{\mathbf{x}'_{t-1} \boldsymbol{\Gamma}_t \mathbf{x}_{t-1}} - \boldsymbol{\mu}'_t \mathbf{x}_{t-1}. \quad (10)$$

In some circumstances (especially when we consider short-horizon stock returns), we can approximate the mean asset returns by zero. Letting $\boldsymbol{\mu}_t = \mathbf{0}$, we can simplify the above equation to

$$V_t = \Phi^{-1}(\alpha) \sqrt{\mathbf{x}'_{t-1} \boldsymbol{\Gamma}_t \mathbf{x}_{t-1}}. \quad (11)$$

We can estimate V_t in the N -asset case by substituting estimators for the parameters in the above equations. First, for the general case,

$$\hat{V}_t = \Phi^{-1}(\alpha) \sqrt{\mathbf{x}'_{t-1} \hat{\boldsymbol{\Gamma}}_t \mathbf{x}_{t-1}} - \hat{\boldsymbol{\mu}}'_t \mathbf{x}_{t-1}, \quad (12)$$

and when the mean asset returns are zero,

$$\hat{V}_t = \Phi^{-1}(\alpha) \sqrt{\mathbf{x}'_{t-1} \hat{\boldsymbol{\Gamma}}_t \mathbf{x}_{t-1}}. \quad (13)$$

D. The VaR as an Investment Framework

The above discussion of the VaR took the “passive” viewpoint of estimating the VaR of an existing portfolio. We can also use the VaR in an alternative way to actively control the risk incurred by the portfolio. The asset-allocation framework that we introduce to this effect is as follows:

- ① At each time-step t , a *target VaR* \tilde{V}_{t+1} is set (for example by the portfolio manager). The goal of our strategy is to construct a portfolio \mathbf{x}_t having this target VaR.
- ② We consult a decision system, such as a neural network, to obtain *allocation recommendations* for the set of N assets. These recommendations take the form of a vector \mathbf{y}_t , which gives the *relative weightings* of the assets in the portfolio; we impose no constraint (e.g. positivity or sum-to-one) on the y_{it} .
- ③ The recommendation vector \mathbf{y}_t is *rescaled* by a constant factor (see below) in order to produce a vector \mathbf{x}_t of final positions (in dollars) to take in each asset at time t . This rescaling is performed such that the estimator $\hat{V}_{t+1|t}$ (computed given the information set \mathcal{I}_t) of the portfolio VaR over period $t+1$ is equal to the target VaR, \tilde{V}_{t+1} .
- ④ Borrow the amount $\sum_{i=1}^N x_{it}$ at the risk-free rate r_{0t} and invest it at time t in the portfolio \mathbf{x}_t for exactly one period. At the end of the period, evaluate the profit or loss (using a performance measure explained shortly.)

It should be noted that this framework differs from a conventional investment setting in that the profits generated during one period *are not reinvested* during the next. All that we are seeking to achieve is to construct, for each period, a portfolio \mathbf{x}_t that matches a given target VaR \tilde{V}_{t+1} . We assume that it is always possible to borrow at the risk-free rate to carry out the investment.

We mention that a framework similar to this one is used by at least one major Canadian bank for parts of its short-term asset management.

E. Rescaling Equations

Our use of the VaR as an investment framework is based on the observation that a portfolio with a given target VaR \tilde{V}_{t+1} can be constructed by homogeneously multiplying the recommendations vector \mathbf{y}_t (which does not obey any VaR constraint) by a constant:

$$\mathbf{x}_t = \beta_t \mathbf{y}_t, \quad (14)$$

where $\beta_t \geq 0$ is a scalar. To simplify the calculation of β_t , we make the assumption that the asset returns over period $t+1$, follow a zero-mean normal distribution, conditional on \mathcal{I}_t :

$$\mathbf{R}_{t+1} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Gamma}_{t+1}), \quad (15)$$

with $\mathbf{\Gamma}_{t+1}$ positive-definite. Then, given a (fixed) recommendations vector \mathbf{y}_t , $\|\mathbf{y}_t\| > 0$, the rescaling factor is given by

$$\beta_t = \frac{\tilde{V}_{t+1}}{\Phi^{-1}(\alpha) \sqrt{\mathbf{y}_t' \mathbf{\Gamma}_{t+1} \mathbf{y}_t}}. \quad (16)$$

It can be verified directly by substitution into eq. (11) that the VaR of the portfolio \mathbf{x}_t given by eq. (14) is indeed the target VaR \tilde{V}_{t+1} .

E.1 Estimating β_t

In practice, we have to replace the $\mathbf{\Gamma}_{t+1}$ in the above equation by an estimator. We can estimate the rescaling factor simply as follows:

$$\hat{\beta}_t = \frac{\tilde{V}_{t+1}}{\Phi^{-1}(\alpha) \sqrt{\mathbf{y}_t' \hat{\mathbf{\Gamma}}_{t+1} \mathbf{y}_t}}. \quad (17)$$

Unfortunately, even if $\hat{\mathbf{\Gamma}}_{t+1}$ is unbiased, $\hat{\beta}_t$ is biased in finite samples (because, in general for a random variable $X > 0$, $E[1/X] \neq 1/E[X]$). However, the samples that we use are of sufficient size for the bias to be negligible. [9] provides a proof that $\hat{\beta}_t$ is asymptotically unbiased, and proposes another (slightly more complicated) estimator that is unbiased in finite samples under certain assumptions.

F. The VaR as a Performance Measure

The VaR of a portfolio can also be used as the risk measure to evaluate the performance of a portfolio. The performance measure that we consider for a fixed strategy S is a simple average of the VaR-corrected net profit generated during each period (see e.g. [4], for similar formulations):

$$W^S = \frac{1}{T} \sum_{t=1}^T W_t^S, \quad (18)$$

where W_t^S is the (random) net profit produced by strategy S over period t (between times $t - 1$ and t), computed as follows (we give the equation for W_{t+1}^S to simplify the notation):

$$W_{t+1}^S = \frac{(\mathbf{R}_{t+1} - \iota r_{0t})' \mathbf{x}_t^S + \text{loss}_t}{V_{t+1}}, \quad (19)$$

The numerator computes the excess return of the portfolio for the period (over the borrowing costs at the risk-free rate r_{0t}), and accounts for the transaction costs incurred for establishing the position \mathbf{x}_t from \mathbf{x}_{t-1} , as described below.

We note that it is necessary to normalize W_{t+1}^S by the VaR V_t —the risk measure—, since eq. (16) clearly shows that a dollar profit as large as desired can be achieved by making V_t sufficiently large.

F.1 Estimating W^S and W_t^S

To estimate the quantities W^S and W_t^S , we substitute for $\{\mathbf{R}_t\}$ the realized returns $\{\mathbf{r}_t\}$, and we use the target VaR \tilde{V}_t as an estimator of the portfolio VaR V_t :

$$\hat{W}^S = \frac{1}{T} \sum_{t=1}^T \hat{W}_t^S \quad (20)$$

$$\hat{W}_{t+1}^S = \frac{(\mathbf{r}_{t+1} - \iota r_{0t})' \mathbf{x}_t^S + \text{loss}_t}{\tilde{V}_{t+1}}. \quad (21)$$

As for $\hat{\beta}_t$, we ignore the finite-sample bias of these estimators, for it is of little significance for the sample sizes that we use in practice.

Examining eq. (20), it should be obvious that this performance measure is equivalent to the well-known Sharpe Ratio [10] for symmetric return distributions (within a multiplicative factor), with the exception that it uses the *ex ante* volatility (VaR) rather than the *ex post* volatility as the risk measure.

F.2 Transaction Costs

Transaction costs are modeled by a simple multiplicative loss:

$$\text{loss}_t = -\mathbf{c}' |\mathbf{x}_t - \tilde{\mathbf{x}}_t| \quad (22)$$

where $\mathbf{c} = (c_1, \dots, c_N)'$, c_i the relative loss associated with a change in position (in dollars) in asset i , and $\tilde{\mathbf{x}}_t$ the portfolio positions in each asset *immediately before* that the transaction is performed at time t . This position is different from \mathbf{x}_{t-1} because of the asset returns generated during period t :

$$\tilde{x}_{it} = (r_{it} + 1) x_{i(t-1)}. \quad (23)$$

In our experiments, the transaction costs were set uniformly to 0.1%.

G. Volatility Estimation

As eq. (16) shows, the covariance matrix $\mathbf{\Gamma}_t$ plays a fundamental role in computing the value at risk of a portfolio (under the normal approximation). It is therefore of extreme importance to make use of a good estimator for this covariance matrix.

For this purpose, we used an exponentially-weighted moving average (EWMA) estimator, of the kind put forward by RiskMetrics [11]. Given an estimator of the covariance matrix at time $t - 1$, a new estimate is computed by

$$\hat{\mathbf{\Gamma}}_t = \lambda \hat{\mathbf{\Gamma}}_{t-1} + (1 - \lambda)(\mathbf{r}_t \mathbf{r}_t'), \quad (24)$$

where \mathbf{r}_t is the vector of asset returns over period t and λ is a *decay factor* that controls the speed at which observations are “absorbed” by the estimator. We used the value recommended by RiskMetrics for monthly data, $\lambda = 0.97$.

III. NEURAL NETWORKS FOR PORTFOLIO MANAGEMENT

The use of adaptive decision systems, such as neural networks, to implement asset-allocation systems is not new. Most applications of them fall into two categories: (i) using the neural net as a forecasting model, in conjunction with an allocation scheme (such as mean–variance allocation) to make the final decision; and (ii) using the neural net to directly make the asset allocation decisions. We start by setting some notation related to our use of neural networks, and we then consider these two approaches in the context of portfolio selection subject to VaR constraints.

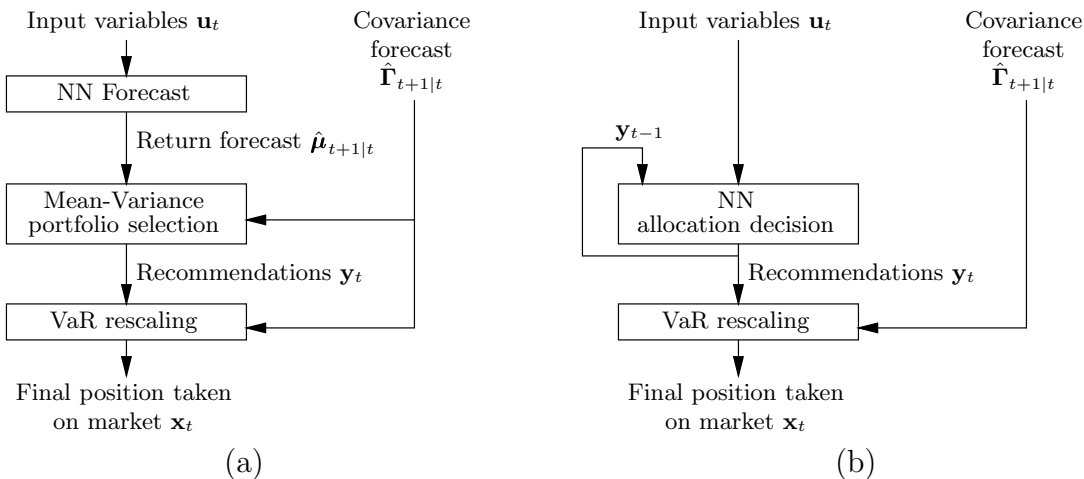


Fig. 1. The forecasting (a) and decision (b) paradigms for using neural networks (NN) in asset allocation.

A. Neural Networks

We consider a specific type of neural network, the multi-layer perceptron (MLP) with one hidden Tanh layer (with H hidden units), and a linear output layer. We denote by $f : \mathbb{R}^M \mapsto \mathbb{R}^N$ the vectorial function represented by the MLP. Let $\mathbf{x} (\in \mathbb{R}^M)$ be an input vector; the function is computed by the MLP as follows:

$$f(\mathbf{x}; \boldsymbol{\theta}) = \mathbf{A}_2 \tanh(\mathbf{A}_1 \mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2. \quad (25)$$

The adjustable parameters of the network are: \mathbf{A}_1 , an $H \times M$ matrix; \mathbf{b}_1 an H -element vector; \mathbf{A}_2 an $N \times H$ matrix; and \mathbf{b}_2 an N -element vector. We denote by $\boldsymbol{\theta}$ the vector of all parameters:

$$\boldsymbol{\theta} = \langle \mathbf{A}_1, \mathbf{A}_2, \mathbf{b}_1, \mathbf{b}_2 \rangle.$$

A.1 Network Training

The parameters $\boldsymbol{\theta}$ are found by training the network to minimize a cost function, which depends, as we shall see below, on the type of model—forecasting or decision—that we are using. In our implementation, the optimization is carried out using a conjugate gradient descent algorithm [12]. The gradient of the parameters with respect to the cost function is computed using the standard backpropagation algorithm [13] for multi-layer perceptrons.

B. Forecasting Model

The forecasting model centers around a general procedure whose objective is to find an “optimal” allocation of assets, one which maximizes the expected value of a utility function (fixed *a priori*, and specific to each investor), given a probability distribution of asset returns.

The use of the neural network within the forecasting model is illustrated in figure 1a. The network is used to make forecasts of asset returns in the next time period, $\hat{\boldsymbol{\mu}}_{t+1|t}$, given explanatory variables \mathbf{u}_t , which are described in section V-A (these variables are determined causally, i.e. they are a function of \mathcal{I}_t .)

B.1 Maximization of Expected Utility

We assume that an investor associates a utility function $U(\mathbf{r}_{t+1}, \mathbf{w}_t)$ with the performance of his/her investment in the portfolio \mathbf{w}_t over period $t + 1$. (For the remainder of this section, we suppose, without loss of generality, that the net capital in a portfolio has been factored out of the equations; we use \mathbf{w}_t to denote a portfolio whose elements sum to one.)

The problem of (myopic) utility maximization consists, at each time-step t , in finding the portfolio \mathbf{w}_t that maximizes the expected utility obtained at $t + 1$, given the information available at time t :

$$\mathbf{w}_t^* = \arg \max_{\mathbf{w}_t} \mathbb{E}[U(\mathbf{R}_{t+1}, \mathbf{w}_t) | \mathcal{I}_t]. \quad (26)$$

This procedure is called myopic because we only seek to maximize the expected utility over the next period, and not over the entire sequence of periods until some end-of-times.

The expected utility can be expressed in the form of an integral:

$$\mathbb{E}[U(\mathbf{R}_{t+1}, \mathbf{w}_t) | \mathcal{I}_t] = \int_{\mathbf{R}_{t+1}} P_{t+1|t}(\mathbf{r}) U(\mathbf{r}, \mathbf{w}_t) d\mathbf{r}, \quad (27)$$

where $P_{t+1|t}(\cdot)$ is the probability density function of the asset returns, \mathbf{R}_{t+1} , given the information available at time t .

B.2 Quadratic Utility

Some “simple” utility functions admit analytical solutions for the expected utility (27). To derive the mean–variance allocation equations, we shall postulate that investors are governed by a quadratic utility of the form

$$U(\mathbf{R}_{t+1}, \mathbf{w}_t) = \mathbf{R}'_{t+1} \mathbf{w}_t - \lambda (\mathbf{w}'_t (\mathbf{R}_{t+1} - \boldsymbol{\mu}_{t+1}))^2. \quad (28)$$

The parameter $\lambda > 0$ represents the risk aversion of the investor; more risk-averse investors will choose higher λ 's.

Assuming the first and second moment of the conditional distribution of asset returns exist, and writing them $\boldsymbol{\mu}_{t+1}$ and $\boldsymbol{\Gamma}_{t+1}$ respectively (with $\boldsymbol{\Gamma}_{t+1}$ positive-definite), eq. (28) can be integrated out analytically to give the expected quadratic utility:

$$\mathbb{E}[U(\mathbf{R}_{t+1}, \mathbf{w}_t) | \mathcal{I}_t] = \boldsymbol{\mu}'_{t+1} \mathbf{w}_t - \lambda \mathbf{w}'_t \boldsymbol{\Gamma}_{t+1} \mathbf{w}_t. \quad (29)$$

Substituting estimators available at time t , we obtain an estimator of the expected utility at time $t + 1$:

$$\hat{U}_{t+1}(\mathbf{w}_t) = \hat{\boldsymbol{\mu}}'_{t+1|t} \mathbf{w}_t - \lambda \mathbf{w}'_t \hat{\boldsymbol{\Gamma}}_{t+1|t} \mathbf{w}_t. \quad (30)$$

(We abuse slightly the notation here by denoting by \hat{U} the estimator of *expected* utility.)

B.3 Mean–variance allocation

We now derive, under quadratic utility, the portfolio allocation equation. We seek a vector of “optimal” weights \mathbf{w}_t^* that will yield the maximum expected utility at time $t + 1$, given the information at time t .

Note that we can derive an analytical solution to this problem because we allow the weights to be negative as well as positive; the only constraint that we impose on the weights is that they sum to one (all the capital is invested). In contrast, the classical Markowitz formulation [14] further imposes the *positivity* of the weights; this makes the optimization problem tractable only by computational methods, such as quadratic programming.

We start by forming the lagrangian incorporating the sum-to-one constraint to eq. (29), observing that maximizing this equation is equivalent to minimizing its negative ($\boldsymbol{\nu}$ is the vector $(1, \dots, 1)'$):

$$\mathcal{L}(\mathbf{w}_t, \alpha) = -\boldsymbol{\mu}'_{t+1} \mathbf{w}_t + \lambda \mathbf{w}'_t \boldsymbol{\Gamma}_{t+1} \mathbf{w}_t + \alpha (\mathbf{w}'_t \boldsymbol{\nu} - 1). \quad (31)$$

After differentiating this equation and a bit of algebra, we find:

$$\mathbf{w}_t^* = \frac{1}{\lambda} \boldsymbol{\Gamma}_{t+1}^{-1} \left(\boldsymbol{\mu}_{t+1} - \frac{\boldsymbol{\nu}' \boldsymbol{\Gamma}_{t+1}^{-1} \boldsymbol{\mu}_{t+1} - \lambda}{\boldsymbol{\nu}' \boldsymbol{\Gamma}_{t+1}^{-1} \boldsymbol{\nu}} \boldsymbol{\nu} \right). \quad (32)$$

In practical use, we have to substitute estimators available at time t for the parameters $\boldsymbol{\mu}_{t+1}$ and $\boldsymbol{\Gamma}_{t+1}$ in this equation.

To recapitulate, the “optimal” weight vector \mathbf{w}_t^* constitutes the *recommendations vector* \mathbf{y}_t output by the mean–variance allocation module in figure 1a.

B.4 MLP Training Cost Function

As illustrated in figure 1a, the role played by the neural network in the forecasting model is to produce estimates of the mean asset returns over the next period. This use of a neural net is all-the-more classical, and hence the training procedure brings no surprise.

The network is trained to minimize the prediction error of the realized asset returns, using a quadratic loss function:

$$C_F(\boldsymbol{\theta}) = \frac{1}{T} \sum_{t=1}^T \|f(\mathbf{u}_t; \boldsymbol{\theta}) - \mathbf{r}_{t+1}\|^2 + C_{\text{WD}}(\boldsymbol{\theta}) + C_{\text{ID}}(\boldsymbol{\theta}) \quad (33)$$

where $\|\cdot\|$ is the Euclidian distance, and $f(\cdot; \boldsymbol{\theta})$ is the function computed by the MLP, given the parameter vector $\boldsymbol{\theta}$. The $C_{\text{WD}}(\boldsymbol{\theta})$ and $C_{\text{ID}}(\boldsymbol{\theta})$ terms serve regularization purposes; they are described in section IV.

As explained above, the network is trained to minimize this cost function using a conjugate gradient optimizer, with gradient information computed using the standard backpropagation algorithm for MLPs.

C. Decision Model

Within the decision model, in contrast with the forecasting model introduced previously, the neural network directly yields the allocation recommendations \mathbf{y}_t from explanatory variables \mathbf{u}_t (figure 1b).

We introduce the possibility for the network to be *recurrent*, taking as input the recommendations emitted during the previous time step. This enables, in theory, the network to make decisions that would not lead to excess trading, to minimize transaction costs.

C.1 Justifying The Model

Before explaining the technical machinery necessary for training the recurrent neural network in the decision model, we provide a brief explanation as to why such a network would be attractive. We note immediately that, as a downside for the model, the steps required to produce a decision are not as “transparent” as they are for the forecasting model: everything happens inside the “black box” of the neural network. However, from a pragmatic standpoint, the following reasons lead us to believe that the model’s potential is at least worthy of investigation:

- The probability density estimation problem—which must be solved in one way or another by the forecasting model—is intrinsically a difficult problem in high dimension. The decision model does not require an explicit solution to this problem (although some function of the density is learned implicitly by the model).
- The decision model does not need to explicitly postulate a utility function that admits a simple mathematical treatment, but which may not correspond to the needs of the investor. The choice of this utility function is important, for it directly leads to the allocation decisions within the forecasting model. However, we already know, without deep analysis, that quadratic utility does not constitute the “true” utility of an investor, for the sole reasons that it treats good news just as negatively as bad news (because both lead to high variance), and does not consider transaction costs. Furthermore, the utility function of the forecasting model is not the final financial criterion (18) on which it is ultimately evaluated. In contrast, the decision model directly maximizes this criterion.

C.2 Training Cost Function

The network is trained to directly minimize the (negative of the) financial performance evaluation criterion (18):

$$C_D(\boldsymbol{\theta}) = -\frac{1}{T} \sum_{t=1}^T W_t + C_{WD}(\boldsymbol{\theta}) + C_{ID}(\boldsymbol{\theta}) + C_{\text{norm}}. \quad (34)$$

The terms $C_{WD}(\boldsymbol{\theta})$ and $C_{ID}(\boldsymbol{\theta})$, which are the same as in the forecasting model cost function, are described in section IV. The new term C_{norm} induces a preference on the norm of the solutions produced by the neural network; its nature is explained shortly.

The effect of this cost function is to have the network learn to maximize the profit returned by a VaR-constrained portfolio.

C.3 Training the MLP

The training procedure for the MLP is quite more complex for the decision model than it is for the forecasting model: the feedback loop, which provides as inputs to the network the recommendations \mathbf{y}_{t-1} produced for the preceding time step, induces a recurrence which must be accounted for. This feedback loop is required for the following reasons:

- The transaction costs introduce a coupling between two successive time steps: the decision made at time t has an impact on both the transaction costs incurred at t and at $t+1$. This coupling induces in turn a gradient with respect to the positions \mathbf{x}_t coming from the positions \mathbf{x}_{t+1} , and this information can be of use during training. We explain these dependencies more deeply in the following section.
- In addition, knowing the decision made during the preceding time step can enable the network to learn a strategy that minimizes the transaction costs: given a choice between two equally profitable positions at time t , the network can minimize the transaction costs by choosing that closer to the position taken at time $t-1$; for this reason, providing \mathbf{y}_{t-1} as input can be useful. Unfortunately, this ideal of minimizing costs can never be reached perfectly, because our current process of rescaling the positions at each time step for reaching the target VaR is always performed *unconditionally*, i.e. oblivious to the previous positions.

C.4 Backpropagation Equations

We now introduce the backpropagation equations. We note that these equations shall be, for a short moment, slightly incomplete: we present in the following section a regularization condition that ensures the existence of local minima of the cost function.

The backpropagation equations are obtained in the usual way, by traversing the flow graph of the allocation system, unfolded through time, and by accumulating all the contributions to the gradient at a node. Figure 2 illustrates this

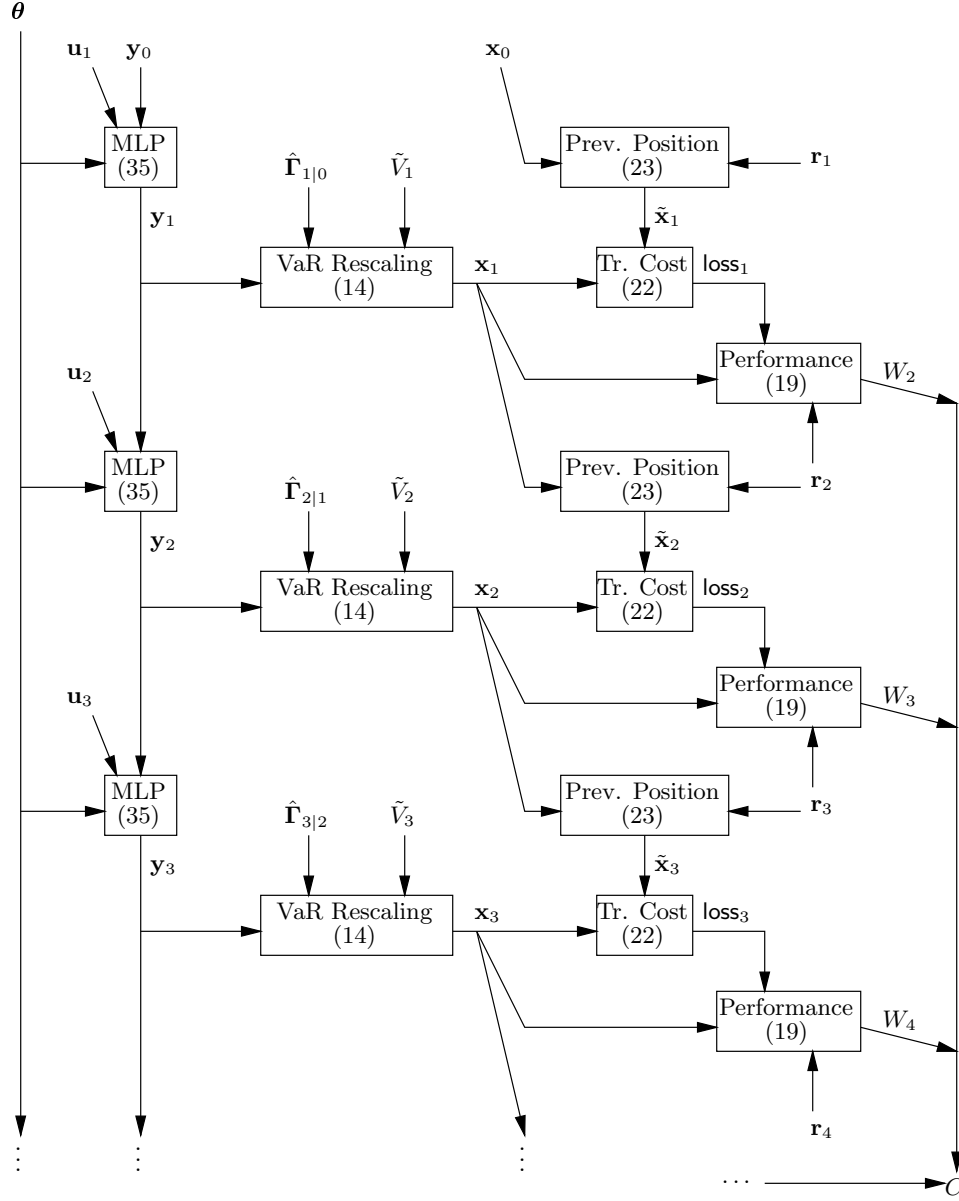


Fig. 2. Flow graph of the steps implemented by the decision model, unfolded through time. The backpropagation equations are obtained by traversing the graph in the reverse direction of the arrows. The numbers in parentheses refer to the equations (in the main text) used for computing each value.

graph, unfolded for the first few time steps. Following the backpropagation-through-time (BPTT) algorithm [13], we compute the gradient by going back in time, starting from the last time step T until the first one.

Recall that we denote by $f(\cdot; \theta)$ the function computed by a MLP with parameter vector θ . In the decision model, the allocation recommendations \mathbf{y}_t are the direct product of the MLP:

$$\mathbf{y}_t = f(\mathbf{y}_{t-1}, \mathbf{u}_t; \theta), \quad (35)$$

where \mathbf{u}_t are explanatory variables considered useful to the allocation problem, which we can compute given the information set \mathcal{I}_t .

We shall consider a slightly simpler criterion C to minimize than eq. (34), one that does not include any regularization term. As we shall see below, incorporating those terms involves trivial modifications to the gradient computation. Our simplified criterion C (illustrated in the lower right-hand side of figure 2) is:

$$C = -\hat{W}. \quad (36)$$

From eq. (18), we account for the contribution brought to the criterion by the profit at each time step:

$$\frac{\partial C}{\partial \hat{W}_{t+1}} = -\frac{1}{T}. \quad (37)$$

Next, we make use of eq. (19), (22) and (23) to determine the contribution of transaction costs to the gradient:

$$\frac{\partial C}{\partial \text{loss}_t} = -\frac{1}{T \tilde{V}_t} \quad (38)$$

$$\frac{\partial \text{loss}_t}{\partial x_{it}} = -c_i \text{sign}(x_{it} - \tilde{x}_{it}) \quad (39)$$

$$\frac{\partial \text{loss}_t}{\partial \tilde{x}_{it}} = c_i \text{sign}(x_{it} - \tilde{x}_{it}) \quad (40)$$

$$\begin{aligned} \frac{\partial \text{loss}_{t+1}}{\partial x_{it}} &= c_i \text{sign}(x_{i(t+1)} - \tilde{x}_{i(t+1)}) \frac{\partial \tilde{x}_{i(t+1)}}{\partial x_{it}} \\ &= c_i \text{sign}(x_{i(t+1)} - \tilde{x}_{i(t+1)}) (1 + r_{i(t+1)}). \end{aligned} \quad (41)$$

From this point, again making use of eq. (19), we compute the contribution of x_{it} to the gradient, which comes from the two ‘‘paths’’ by which x_{it} affects C : a first direct contribution through the return between times t and $t + 1$; and a second indirect contribution through the transaction costs at $t + 1$:

$$\frac{\partial C}{\partial x_{it}} = \frac{\partial C}{\partial \hat{W}_{t+1}} \frac{\partial \hat{W}_{t+1}}{\partial x_{it}} + \frac{\partial C}{\partial \hat{W}_{t+2}} \frac{\partial \hat{W}_{t+2}}{\partial x_{it}}. \quad (42)$$

Because $\partial C / \partial \hat{W}_{t+1}$ is simply given by eq. (37), we use eq. (19) to compute

$$\frac{\partial C}{\partial \hat{W}_{t+1}} \frac{\partial \hat{W}_{t+1}}{\partial x_{it}} = -\frac{1}{T \tilde{V}_t} \left(r_{i(t+1)} - r_{0t} + \frac{\partial \text{loss}_t}{\partial x_{it}} \right), \quad (43)$$

whence,

$$\frac{\partial C}{\partial \hat{W}_{t+1}} \frac{\partial \hat{W}_{t+1}}{\partial x_{it}} = -\frac{1}{T \tilde{V}_t} (r_{i(t+1)} - r_{0t} - c_i \text{sign}(x_{it} - \tilde{x}_{it})). \quad (44)$$

In the same manner, we compute the contribution

$$\frac{\partial C}{\partial \hat{W}_{t+2}} \frac{\partial \hat{W}_{t+2}}{\partial x_{it}} = -\frac{1}{T \tilde{V}_{t+1}} \frac{\partial \text{loss}_{t+1}}{\partial x_{it}}, \quad (45)$$

which gives, after simplification,

$$\frac{\partial C}{\partial \hat{W}_{t+2}} \frac{\partial \hat{W}_{t+2}}{\partial x_{it}} = -\frac{1}{T \tilde{V}_{t+1}} (c_i \text{sign}(x_{i(t+1)} - \tilde{x}_{i(t+1)}) (1 + r_{i(t+1)})). \quad (46)$$

Finally, we add up the two previous equations to obtain

$$\frac{\partial C}{\partial x_{it}} = -\frac{1}{T \tilde{V}_t} (r_{i(t+1)} - r_{0t} - c_i \text{sign}(x_{it} - \tilde{x}_{it})) - \frac{1}{T \tilde{V}_{t+1}} (c_i \text{sign}(x_{i(t+1)} - \tilde{x}_{i(t+1)}) (1 + r_{i(t+1)})). \quad (47)$$

We are now in a position to compute the gradient with respect to the neural network outputs. Using eq. (14) and (16), we start by evaluating the effect of y_{it} on x_{it} :²

$$\frac{\partial x_{it}}{\partial y_{it}} = \frac{\tilde{V}_t}{\Phi^{-1}(\alpha) \left(\mathbf{y}'_t \hat{\Gamma}_{t+1|t} \mathbf{y}_t \right)^{\frac{1}{2}}} - \frac{y_{it} \tilde{V}_t \sum_{k=1}^N \hat{\gamma}_{ik(t+1)} y_{kt}}{\Phi^{-1}(\alpha) \left(\mathbf{y}'_t \hat{\Gamma}_{t+1|t} \mathbf{y}_t \right)^{\frac{3}{2}}}, \quad (48)$$

and for $i \neq j$,

$$\frac{\partial x_{it}}{\partial y_{jt}} = -\frac{y_{it} \tilde{V}_t \sum_{k=1}^N \hat{\gamma}_{jk(t+1)} y_{kt}}{\Phi^{-1}(\alpha) \left(\mathbf{y}'_t \hat{\Gamma}_{t+1|t} \mathbf{y}_t \right)^{\frac{3}{2}}}. \quad (49)$$

²To arrive at these equations, it is useful to recall that $\mathbf{y}' \Gamma \mathbf{y}$ can be written in the form of $\sum_k \sum_\ell \gamma_{k\ell} y_k y_\ell$, whence it easily follows that $\frac{\partial}{\partial y_i} \mathbf{y}' \Gamma \mathbf{y} = 2 \sum_k \gamma_{ik} y_k$.

(As previously noted, α is the desired level of the VaR and $\Phi^{-1}(\cdot)$ is the inverse cumulative distribution function of the standardized normal distribution.)

The complete gradient is given by

$$\frac{\partial C}{\partial y_{it}} = \sum_k \frac{\partial C}{\partial x_{kt}} \frac{\partial x_{kt}}{\partial y_{it}} + \frac{\partial C}{\partial f_{t+1}}, \quad (50)$$

where $\partial C/\partial f_{t+1}$ is the gradient with respect to the inputs of the neural network at time $t+1$, which is a usual by-product of the standard backpropagation algorithm.

C.5 Introducing a ‘‘Preferred Norm’’

The cost function (36) corresponding to the financial criterion (18) cannot reliably be used in its original form to train a neural network. The reason lies in the rescaling equations (14) and (16) that transform a recommendation vector \mathbf{y}_t into a VaR-constrained portfolio \mathbf{x}_t . Consider two recommendations $\mathbf{y}_t^{(1)}$ and $\mathbf{y}_t^{(2)}$ that differ only by a multiplicative factor $\delta > 0$:

$$\mathbf{y}_t^{(2)} = \delta \mathbf{y}_t^{(1)}.$$

As can easily be seen by substitution in the rescaling equations, the final portfolios obtained from those two (different) recommendations are identical! Put differently, two different recommendations that have the same direction but different lengths are rescaled into the same final portfolio.

This phenomenon is illustrated in figure 3, which shows the level curves of the cost function for a small allocation problem between two assets (stocks and bonds, in this case), as a function of the *recommendations* output by the network. We observe clearly that different recommendations in the same direction yield the same cost.

The direct consequence of this effect is that the optimization problem for training the parameters of the neural network is not well posed: two different sets of parameters yielding equal solutions (within a constant factor) will be judged as equivalent by the cost function. This problem can be expressed more precisely as follows: for nearly every parameter vector $\boldsymbol{\theta}$, there is a direction from that point that has (exactly) zero gradient, and hence there is no local minimum in that direction. We have observed empirically that this could lead to severe divergence problems when the network is trained with the usual gradient-based optimization algorithms such as conjugate gradient descent.

This problem suggests that we can introduce an *a priori* preference on the norm of the recommendations, using a modification to the cost function that is analogous to the ‘‘hints’’ mechanism that is sometimes used for incorporating *a priori* knowledge in neural network training [15]. This preference is introduced by way of a soft constraint, the regularization term C_{norm} appearing in eq. (34):

$$C_{\text{norm}} = \frac{\phi_{\text{norm}}}{2T} \sum_{t=1}^T \left(\sum_{i=1}^N y_{it}^2 - \rho^2 \right)^2. \quad (51)$$

Two parameters must be determined by the user: (i) ρ , which is the desired norm for the recommendations output by the neural network (in our experiments, it was arbitrarily set to $\rho^2 = 0.9$), and (ii) ϕ_{norm} , which controls the relative importance of the penalization in the total cost.

Figure 4 illustrates the cost function modified to incorporate this penalization (with $\rho^2 = 0.9$ and $\phi_{\text{norm}} = 0.1$). We now observe the clear presence of local minima in this function. The optimal solution is in the same direction as previously, but it is now encouraged to have a length ρ .

This penalization brings forth a small change to the backpropagation equations introduced previously: the term $\partial C/\partial y_{it}$, eq. (50), must be adjusted to become:

$$\frac{\partial C'}{\partial y_{it}} = \frac{\partial C}{\partial y_{it}} + \frac{\phi_{\text{norm}}}{T} \left(\sum_{j=1}^N y_{jt}^2 - \rho^2 \right) (2y_{it}). \quad (52)$$

C.6 Reference Portfolio

A second type of preference takes the form of a *preferred portfolio*: in some circumstances, we may know *a priori* what should be ‘‘good positions’’ to take, often because of regulatory constraints. For instance, a portfolio manager may be mandated to construct her portfolio such that it contains ‘‘approximately’’ 60% stocks and 40% bonds. This constraint, which results from policies on which the manager has no immediate control, constitutes the reference portfolio.

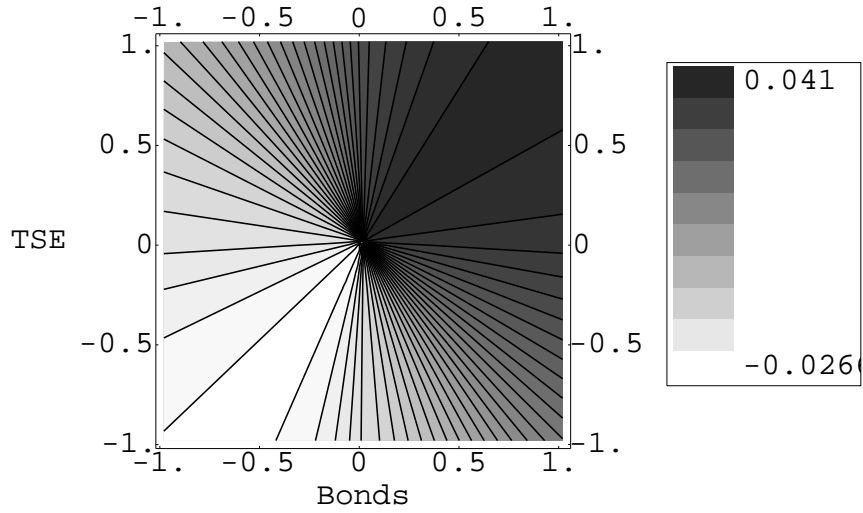


Fig. 3. Level curves of the non-regularized cost function for a two-asset allocation problem. The axes indicate the value of each component of a *recommendation*. There is no minimum point to this function, but rather a half-line of minimal cost, starting around the origin towards the bottom left. This is undesirable, since it may lead to numerical difficulties when optimizing the VaR criterion.

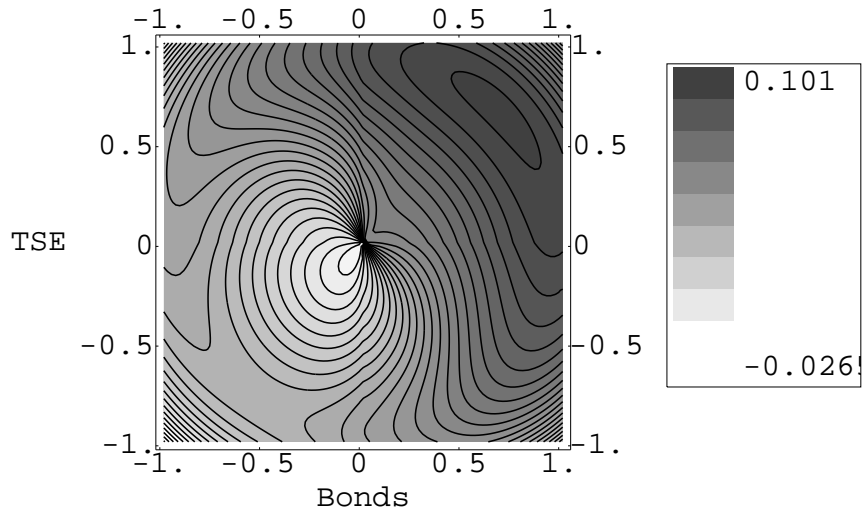


Fig. 4. Level curves of the regularized cost function for the two-asset problem. The “preferred” norm of the recommendations has been fixed to $\rho^2 = 0.9$. In contrast to fig. 3, a minimum can clearly be seen a bit to the left and below the origin (i.e. along the minimum half-line of fig.3). This regularized cost function yields a better-behaved optimization process.

We shall denote this reference portfolio by ψ_t . The cost function eq. (34) is modified to replace the C_{norm} term by a $C_{\text{ref. port.}}$ term that penalizes the squared-distance between the network output and the reference portfolio:

$$C_{\text{ref. port.}} = \frac{1}{T} \sum_{t=1}^T \text{penalty}_{\text{ref. port.}}(\mathbf{y}_t) \quad (53)$$

$$\text{penalty}_{\text{ref. port.}}(\mathbf{y}_t) = \frac{\phi_{\text{ref. port.}}}{2} \|\mathbf{y}_t - \psi_t\|^2, \quad (54)$$

with $\|\cdot\|$ the Euclidian distance.

With this change, the backpropagation equations are simple to adjust; we add a contribution to $\partial C/\partial y_{it}$, eq. (50), which becomes:

$$\frac{\partial C_{\text{ref. port.}}}{\partial y_{it}} = \frac{\partial C}{\partial y_{it}} + \frac{\phi_{\text{ref. port.}}}{T} (y_{it} - \psi_{it}). \quad (55)$$

In our experiments with the TSE 300 sectors (see section V), we favored this reference-portfolio penalization over the preferred-norm penalization. Our reference portfolio was chosen to be the market weight of each sector with respect to the complete TSE index; the $\phi_{\text{ref. port.}}$ hyper-parameter was set to a constant 0.1.

IV. REGULARIZATION, HYPERPARAMETER SELECTION, AND MODEL COMBINATION

Regularization techniques are used to specify *a priori* preferences on the network weights; they are useful to control network capacity to help prevent overfitting. In our experiments, we made use of two such methods, weight decay and input decay (in addition, for the decision model, to the norm preference covered previously.)

A. Weight Decay

Weight decay is a classic regularization procedure that imposes a penalty to the squared norm of all network weights:

$$C_{\text{WD}}(\boldsymbol{\theta}) = \frac{\phi_{\text{WD}}}{2} \sum_k \theta_k^2, \quad (56)$$

where the summation is performed over all the elements of the parameter vector $\boldsymbol{\theta}$ (in our experiments, the biases, e.g. \mathbf{b}_1 and \mathbf{b}_2 in eq. (25), were omitted); ϕ_{WD} is a hyperparameter (usually determined through trial-and-error, but not in our case as we shall see shortly) that controls the importance of C_{WD} in the total cost.

The effect of weight decay is to encourage the network weights to have smaller magnitudes; it reduces the learning capacity of the network. Empirically, it often yields improved generalization performance when the number of training examples is relatively small [16]. Its disadvantage is that it does not take into account the function to learn: it applies without discrimination to every weight.

B. Input Decay

Input decay is a method for performing “soft” variable selection during the regular training of the neural network. Contrarily to combinatorial methods such as branch-and-bound and forward or backward selection, we do not seek a “good set” of inputs to provide to the network; we provide them all. The network will automatically penalize the network connections coming from the inputs that turn out not to be important.

Input decay works by imposing a penalty to the squared-norm of the weights linking a particular network input to all hidden units. Let $\theta_{jh}^{(1)}$ the network weight (located on the first layer of the MLP) linking input j to hidden unit h ; the squared-norm of the weights from input j is:

$$C_{\text{ID}}^{(j)}(\boldsymbol{\theta}) = \sum_{h=1}^H \left(\theta_{jh}^{(1)} \right)^2, \quad (57)$$

where H is the number of hidden units in the network. The weights that are part of $C_{\text{ID}}^{(j)}(\boldsymbol{\theta})$ are illustrated in figure 5.

The complete contribution $C_{\text{ID}}(\boldsymbol{\theta})$ to the cost function is obtained by a non-linear combination of the $C_{\text{ID}}^{(j)}$:

$$C_{\text{ID}}(\boldsymbol{\theta}) = \phi_{\text{ID}} \sum_j \frac{C_{\text{ID}}^{(j)}}{\eta + C_{\text{ID}}^{(j)}(\boldsymbol{\theta})}, \quad (58)$$

The behavior of the function $x^2/(\eta + x^2)$ is shown in figure 6. Intuitively, this function acts as follows: if the weights emanating from input j are small, the network must absorb a high marginal cost (locally quadratic) in order to increase the weights; the net effect, in this case, is to bring those weights closer to zero. On the other hand, if the

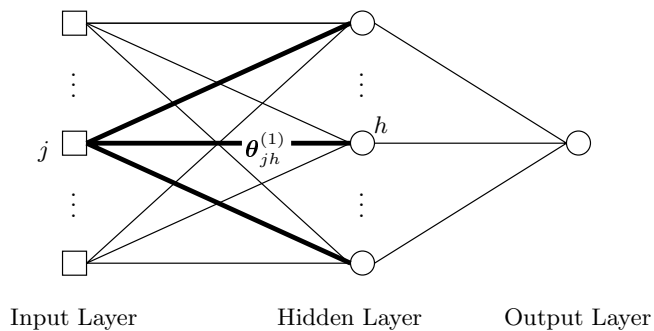


Fig. 5. Soft variable selection: illustration of the network weights affected by the input decay penalty term $C_{\text{ID}}^{(j)}(\boldsymbol{\theta})$, for an input j in a one-hidden-layer MLP (thick lines).

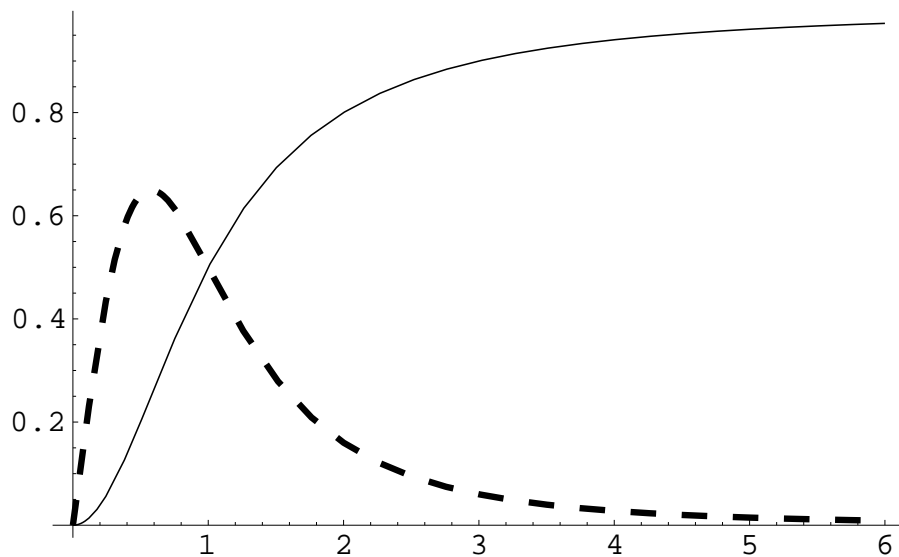


Fig. 6. Soft variable selection: shape of the penalty function $x^2/(\eta + x^2)$ (solid), and its first derivative (dashed), for $\eta = 1$.

weights associated with that input have become large enough, the penalty incurred by the network turns into a constant independent of the value of the weights; those are then free to be adjusted as appropriate. The parameter η acts as a threshold that determines the point beyond which the penalty becomes constant.

Input decay is similar to the *weight elimination* procedure [17] sometimes applied for training neural networks, with the difference that input decay applies in a collective way to the weights associated with a given input.

C. Model Combination

The capacity-control methods described above leave open the question of selecting good values for the hyperparameters ϕ_{WD} and ϕ_{ID} . These parameters are normally chosen such as to minimize the error on a validation set, separate from the testing set. However, we found desirable to completely avoid using a validation set, primarily because of the limited size of our data sets. Since we are not in a position to *choose* the best set of hyperparameters, we used model combination methods to altogether avoid having to make a choice.

We use model combination as follows. We have M underlying models, sharing the same basic MLP topology (number of hidden units) but varying in the hyperparameters. Each model m implements a function $f_{mt}(\cdot)$.³ We construct a *committee* whose decision is a convex combination of the underlying decisions:

$$\mathbf{y}_t^{\text{com}} = \sum_{m=1}^M w_{mt} f_{mt}(\mathbf{u}_t), \quad (59)$$

with \mathbf{u}_t the vector of explanatory variables, and $w_{mt} \geq 0$, $\sum_m w_{mt} = 1$. The weight given to each model depends on

³Because of the retrainings brought forth by the sequential validation procedure described in section IV-D, the function realized by a member of the committee has a time dependency.

the combination method; intuitively, models that have “worked well” in the past should be given greater weight. We consider three such combination methods: hardmax, softmax, and exponentiated gradient.

C.1 Hardmax

The simplest combination method is to choose, at time t , the model that yielded the *best generalization performance* (out-of-sample) for all (available) preceding time steps. We assume that a generalization performance result is available for all time steps from $G + 1$ until $t - 1$ (where t is the current time step).⁴

Let $\hat{W}_m(\tau)$ the (generalization) financial performance returned during period τ by the m -th member of the committee. Let m_t^* the “best model” until time $t - 1$:

$$m_t^* = \arg \max_m \sum_{\tau=G+1}^{t-1} \hat{W}_m(\tau). \quad (60)$$

The weight given at time t to the m -th member of the committee by the hardmax combination method is:

$$w_{mt}^{\text{hardmax}} = \begin{cases} 1 & \text{if } m = m_t^*, \\ 0 & \text{otherwise.} \end{cases} \quad (61)$$

C.2 Softmax

The softmax method is a simple modification of the previous one. It consists in combining the average past generalization performances using the softmax function. Using the same notation as previously, let \bar{W}_{mt} be the average financial performance obtained by the m -th committee member until time $t - 1$:

$$\bar{W}_{mt} = \frac{1}{t - G - 1} \sum_{\tau=G+1}^{t-1} \hat{W}_m(\tau). \quad (62)$$

The weight given at time t to the m -th member of the committee by the softmax combination method is:

$$w_{mt}^{\text{softmax}} = \frac{\exp(\bar{W}_{mt})}{\sum_{k=1}^M \exp(\bar{W}_{kt})}. \quad (63)$$

C.3 Exponentiated Gradient

We used the fixed-share version [18] of the exponentiated gradient algorithm [19]. This method uses an exponential update of the weights, followed by a redistribution step that prevents any of the weights from becoming too large. First, raw weights are computed from the “loss” (19) incurred in the previous time step:

$$\tilde{w}_{mt} = w_{m(t-1)} e^{\delta W_t(f_{m(t-1)})}. \quad (64)$$

Next, a proportional share of the weights is taken and redistributed uniformly (a form of taxation) to produce new weights:

$$\begin{aligned} \text{pool}_t &= \sum_{m=1}^M \tilde{w}_{mt} \\ w_{mt}^{\text{exp. grad.}} &= (1 - \alpha) \tilde{w}_{mt} + \frac{1}{M - 1} (\text{pool}_t - \alpha \tilde{w}_{mt}). \end{aligned} \quad (65)$$

The parameters δ and α control, respectively, the convergence rate and the minimum value of a weight. Some experimentation on the initial training set revealed that $\delta = 0.3$, $\alpha = 0.01$ yielded reasonable behavior, but these values were not tuned extensively.

An extensive analysis of this combination method, including bounds on the generalization error, is provided by [18].

⁴We shall see in section IV-D that this out-of-sample performance is available, for all time steps beyond an initial training set, by using the sequential validation procedure described in that section.

D. Performance Estimation for Sequential Decision Problems

Cross-validation is a performance-evaluation method commonly used when the total size of the data set is relatively small, *provided* that the data contains no temporal structure, i.e. the observations can be freely permuted. Since this is obviously not the case for our current asset-allocation problem, ordinary cross-validation is not applicable.

To obtain low-variance performance estimates, we use a variation on cross-validation called *sequential validation* that preserves the temporal structure of the data. Although a formal definition of the method can be given (e.g. [9]), an intuitive description is as follows:

- ① An *initial training set* is defined, starting from the first available time step and extending until a predefined time G (included). A model of a given topology \mathcal{M} (fixing the number of hidden units, and the value of the hyperparameters) is trained on this initial data.
- ② The model is tested on the P observations in the data set that *follow* after the end of the training set. The test result for each time step is computed using the financial performance criterion, eq. (19). These test results are saved aside.
- ③ The P test observations used in step ② are added to the training set, and a model with the same topology \mathcal{M} is retrained using the new training set.
- ④ Steps ② and ③ are performed until the data set is exhausted.
- ⑤ The final performance estimate for the model with topology \mathcal{M} for the entire data set is obtained by averaging the test results for all time steps saved in step ② (*cf.* eq. (18)).

We observe that for every time step beyond G (the end of the initial training set), a generalization (out-of-sample) performance result is available for a given time step, even though the data for this time step might eventually become part of a later training set.

The “progression” factor P in the size of the training set is a free parameter of the method. If non-stationarities are suspected in the data set, P should be chosen as small as possible; the obvious downside is the greatly increased computational requirement incurred with a small P . In our experiments, we attempted to strike a compromise by setting $P = 12$, which corresponds to retraining every year for monthly data.

Finally, we note that the method of sequential validation owes its simplicity to the fact that the model combination algorithms described above (which can be viewed as performing a kind of model selection) operate strictly on *in-sample* data, and make use of out-of-sample data solely to calculate an unbiased estimate of the generalization error. Alternatively, model selection or combination can be performed after the fact, by choosing the model(s) that performed the best on *test* data; when such a choice is made, it is advisable to make use of a procedure proposed by White [20] to test whether the chosen models might have been biased by data snooping effects.

V. EXPERIMENTAL RESULTS AND ANALYSIS

A. Overall Setting

Our experiments consisted in allocating among the 14 sectors (subindices) of the Toronto Stock Exchange TSE 300 index. Each sector represents an important segment of the Canadian economy. Our benchmark market performance is the complete TSE 300 index. (To make the comparisons meaningful, the market portfolio is also subjected to VaR constraints). We used monthly data ranging from January 1971 until July 1996 (no missing values). Our “risk-free” interest rate is that of the short-term (90-day) Canadian government T-bills.

To obtain a performance estimate for each model, we used the sequential validation procedure, by first training on 120 months and thereafter retraining every 12 months, each time testing on the 12 months following the last training point.

A.1 Inputs and Preprocessing

The input variables \mathbf{u}_t provided to the neural networks consisted of:

- 3 series of 14 moving average returns (short-, mid-, and long-term MA depths).
- 2 series of 14 return volatilities (computed using exponential averages with a short-term and long-term decay).
- 5 series, each corresponding to the “instantaneous” average over the 14 sectors of the above series.

The resulting 75 inputs are then normalized to zero-mean and unit-variance before being provided to the networks.

A.2 Experimental Plan

The experiments that we performed are divided into two distinct parts.

The first set of experiments is designed to understand the impact of the model type (and hence of the cost function used to train the neural network), of network topology and of capacity-control hyperparameters on the financial performance criterion. In this set, we consider:

- **Model type.** We compare (i) the decision model without network recurrence, (ii) the decision model with recurrence, (iii) the forecasting model without recurrence.

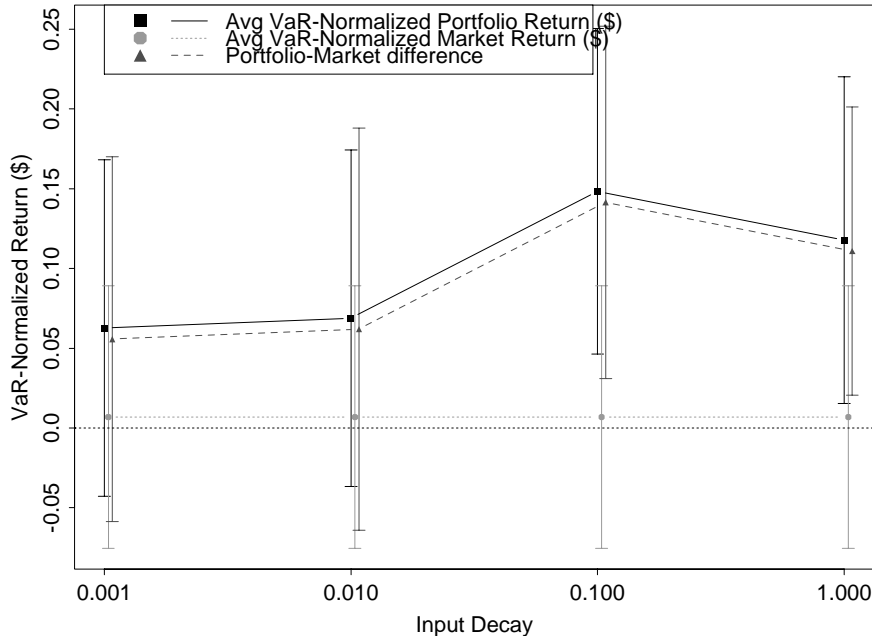


Fig. 7. Effect of Input Decay on the financial performance obtained by an MLP in an asset-allocation task (solid). The (constant) benchmark market performance is given (dotted), along with the MLP–market difference (dashed). The error bars represent 95% confidence intervals. We note that the use of input decay can significantly improve performance.

- **Network topology.** For each model type, we evaluate the effect of the number of hidden units, from the set $NH \in \{2, 5, 10\}$.
- **Capacity control.** For each of the above cases, we evaluate the effects of the weight decay and input decay penalizations. Since we do not know *a priori* what are good settings for the hyperparameters, we train several networks, one for each combination of $\phi_{WD} \in \{10^{-3}, 10^{-2}, 10^{-1}, 10^0\}$ and $\phi_{ID} \in \{10^{-3}, 10^{-2}, 10^{-1}, 10^0\}$.

The second set of experiments verifies the usefulness of the model combination methods. We construct committees that combine, for a given type of model, MLP’s with the same number of hidden units but that vary in the setting of the hyperparameters controlling weight and input decay (ϕ_{WD} and ϕ_{ID}). We evaluate the relative effectiveness of the combination methods, along with the overall performance of a committee compared with that of its underlying models.

B. Results with Single Models

We now analyze the generalization (out-of-sample) performance obtained by all single models on the financial performance criterion. In all the results that follow, we reserve the term “significant” to denote statistical significance at the 0.05 level.

Detailed performance results for the individual models is presented elsewhere [9]. Comparing each model to the benchmark market performance⁵ we observe that several of the single models are yielding net returns that are significantly better than the market.

Figure 7 shows the impact of input decay on a cross-section of the experiments (in this case, the forecasting model with 5 hidden units, and constant $WD = 1.0$.) At each level of the input decay factor, the average performance (square markers) is given with a 95% confidence interval; the benchmark market performance (round markers) and the difference between the model and the benchmark (triangular markers) are also plotted.

B.1 ANOVA Results for Single Models

We further compared the single models using a formal analysis of variance (ANOVA) to detect the systematic impact of a certain factors. The ANOVA (e.g. [21]) is a well-known statistical procedure used to test the effect of several experimental factors (each factor taking several discrete levels) on a continuous measured quantity, in our case, a financial performance measure. The null hypothesis being tested is that the mean performance measure is identical for all levels of the factors under consideration.

⁵This comparison is performed using a paired *t*-test to obtain reasonable-size confidence intervals on the differences. The basic assumptions of the *t*-test—normality and independence of the observations—were quite well fulfilled in our results.

TABLE I

ANOVA results for the **decision model without recurrence**, showing the effect of single factors (number of hidden units (NH), weight decay (WD) and input decay (ID)) along with second- and third-order interactions between these factors. Bold-starred entries are statistically significant at the 5% level. The input decay and number of hidden units factors are significant.

	Degrees of freedom	Sum of squares	F -value	$\Pr(F)$
<i>ID</i>	3	3.146	2.937	0.032*
<i>WD</i>	3	0.015	0.014	0.998
<i>NH</i>	2	3.458	4.841	0.008*
<i>ID : WD</i>	9	0.158	0.049	0.999
<i>ID : NH</i>	6	1.483	0.692	0.656
<i>WD : NH</i>	6	0.114	0.053	0.999
<i>ID : WD : NH</i>	18	0.589	0.092	0.999
Residuals	8928	3188.357		

TABLE II

ANOVA results for the **decision model with recurrence**. The same remarks as table I apply. The input decay and number of hidden units factors are significant.

	Degrees of freedom	Sum of squares	F -value	$\Pr(F)$
<i>ID</i>	3	8.482	8.649	0.000*
<i>WD</i>	3	0.111	0.114	0.952
<i>NH</i>	2	2.969	4.542	0.012*
<i>ID : WD</i>	9	0.392	0.133	0.999
<i>ID : NH</i>	6	1.505	0.767	0.596
<i>WD : NH</i>	6	0.286	0.146	0.990
<i>ID : WD : NH</i>	18	0.336	0.057	1.000
Residuals	8928	2918.357		

The results are given in tables I, II, and III, respectively for the decision model without and with recurrence, and for the forecasting model. We make the following observations:

- For all the model types, the input decay factor has a very significant impact.
- The number of hidden units is significant for the decision models (both with and without recurrence) but is not significant for the forecasting model.
- Weight decay is never significant.
- Higher-order interactions (of second and third order) between the factors are never significant.

B.2 Comparisons Between Models

We also tried to detect performance differences attributable to the model type (decision with or without recurrence, forecasting).

As table IV shows, an ANOVA using the model type as the only factor reveals that we must forcibly reject the null hypothesis that all model types are equivalent. To better understand the performance differences, we performed pairwise comparisons between models.

Table V shows the difference between all model pairs (averaging over all the possible values of the other factors, i.e. the number of hidden units and the value of the hyperparameters controlling weight decay and input decay). The table

TABLE III

ANOVA results for the **forecasting model without recurrence**. The same remarks as table I apply. The input decay factor is significant.

	Degrees of freedom	Sum of squares	F -value	$\Pr(F)$
<i>ID</i>	3	5.483	3.617	0.013*
<i>WD</i>	3	0.068	0.044	0.988
<i>NH</i>	2	0.384	0.380	0.684
<i>ID : WD</i>	9	0.172	0.038	1.000
<i>ID : NH</i>	6	3.684	1.215	0.295
<i>WD : NH</i>	6	0.207	0.068	0.999
<i>ID : WD : NH</i>	18	0.977	0.107	1.000
Residuals	8928	4511.072		

TABLE IV

ANOVA results on the model types: the choice of model type is significant.

	Degrees of freedom	Sum of squares	F -valeur	$\Pr(F)$
Model type	2	4.19	5.298083	0.005*
Residuals	26925	10651.81		

TABLE V

Pairwise comparisons between model types, averaging over the performance at all levels of the other factors (nb. of hidden units, weight and input decay). ‘D’=decision model; ‘F’=forecasting model.

Model x	Model y	sample $x - y$	t -value	DoF	$\Pr(t)$
D w/ rec.	D w/o rec.	0.002	0.371	8975	0.710
F w/o rec.	D w/o rec.	0.027	3.457	8975	0.001*
F w/o rec.	D w/ rec.	0.025	3.275	8975	0.001*

shows that, at the “highest level,” the forecasting model performs significantly better than either decision model.

However, the conclusions made by averaging over all values of the hyperparameters lose their validity when we consider a smaller set of hyperparameters values. Table VI shows the performance difference obtained by restricting the models under consideration to the set trained with input decay $\phi_{ID} \in \{0.001, 0.01\}$ (and averaging over the other factors). For this subset of models, the decision model **with** recurrence is significantly better than either the decision model **without** recurrence or the forecasting model.

From these results, we cannot draw definitive conclusions on the relative merit of the decision model versus the forecasting model, other than the decision model is considerably more sensitive than is the forecasting model to the settings of the hyperparameters used to train the neural network. This should intuitively make sense since much more is expected from the neural network within the decision model.

C. Results with Model Combination

The raw results obtained by the combination methods are given in tables VII, VIII, and IX, respectively for the decision models without and with recurrence, and the forecasting model. Each table gives the generalization financial performance obtained by a committee constructed by combining MLP’s with the same number of hidden units, but trained with different values of the hyperparameters controlling weight decay and input decay (all combinations of $\phi_{WD} \in \{10^{-3}, 10^{-2}, 10^{-1}, 10^0\}$ and $\phi_{ID} \in \{10^{-3}, 10^{-2}, 10^{-1}, 10^0\}$.) Each result is given with a standard error derived from the t distribution, along with the difference in performance with respect to the market benchmark (whose standard error is derived from the t distribution using paired differences.)

A graph summarizing the results for the exponentiated gradient combination method appears in figure 8. Similar graphs are obtained for the other combination methods.

By way of illustration, figure 9 shows the (out-of-sample) behavior of one of the committees. The top part of the figure plots the monthly positions taken in each of the 14 assets. The middle part plots the monthly returns generated by the committee and, for comparison, by the market benchmark; the monthly value-at-risk, set in all our experiments to 1\$, is also illustrated, as an experimental indication that is is not traversed too often (the monthly return of either the committee or the market should not go below the $-1\$$ mark more than 5% of the times). Finally, the bottom part gives the net cumulative returns yielded by the committee and the market benchmark.

TABLE VI

Pairwise comparisons between model types, for the subset of experiments at input decay $\phi_{ID} \in \{0.001, 0.01\}$, and averaging over the performance of the other factors. ‘D’=decision model; ‘F’=forecasting model.

Model x	Model y	sample $x - y$	t -value	DoF	$\Pr(t)$
D w/ rec.	D w/o rec.	0.026	3.164	4487	0.002*
F w/o rec.	D w/o rec.	0.005	0.419	4487	0.676
F w/o rec.	D w/ rec.	-0.022	-2.001	4487	0.046*

TABLE VII

Results for three model combination methods, applied to the **decision model without recurrence**. NH refers to the number of hidden units. The average net market return for the period under consideration is 0.009 (standard error=0.042). Bold-starred entries are statistically significant at the 5% level.

NH	Exponentiated Gradient				Softmax				Hardmax			
	Avg. Return		Diff. w/ market		Avg. Return		Diff. w/ market		Avg. Return		Diff. w/ market	
2	0.023	(0.044)	0.014	(0.035)	0.043	(0.043)	0.034	(0.033)	0.012	(0.045)	0.003	(0.059)
5	0.107	(0.041) *	0.099	(0.056)	0.099	(0.041) *	0.090	(0.053)	0.126	(0.041) *	0.117	(0.061)
10	0.090	(0.045) *	0.081	(0.060)	0.089	(0.045) *	0.080	(0.060)	0.083	(0.046)	0.074	(0.057)

TABLE VIII

Results for three model combination methods, applied to the **decision model with recurrence**. The same remarks as in table VII apply. Many of those committees significantly beat the market.

NH	Exponentiated Gradient				Softmax				Hardmax			
	Avg. Return		Diff. w/ market		Avg. Return		Diff. w/ market		Avg. Return		Diff. w/ market	
2	0.072	(0.043)	0.063	(0.038)	0.087	(0.042) *	0.078	(0.036) *	0.050	(0.039)	0.041	(0.052)
5	0.138	(0.041) *	0.129	(0.051) *	0.132	(0.041) *	0.123	(0.047) *	0.124	(0.041) *	0.116	(0.054) *
10	0.090	(0.042) *	0.081	(0.056)	0.084	(0.043) *	0.076	(0.056)	0.106	(0.042) *	0.097	(0.057)

TABLE IX

Results for three model combination methods, applied to the **forecasting model without recurrence**. The same remarks as in table VII apply. Many of those committees significantly beat the market.

NH	Exponentiated Gradient				Softmax				Hardmax			
	Avg. Return		Diff. w/ market		Avg. Return		Diff. w/ market		Avg. Return		Diff. w/ market	
2	0.127	(0.052) *	0.119	(0.048) *	0.137	(0.052) *	0.128	(0.049) *	0.031	(0.050)	0.022	(0.048)
5	0.156	(0.055) *	0.147	(0.053) *	0.138	(0.053) *	0.129	(0.054) *	0.130	(0.054) *	0.121	(0.050) *
10	0.113	(0.052) *	0.104	(0.058)	0.120	(0.051) *	0.111	(0.057)	0.040	(0.052)	0.032	(0.058)

TABLE X

ANOVA results for the exponentiated gradient committees. The factors are the model type (noted M : decision without or with recurrence; forecasting) and the number of hidden units (noted NH), along with the interaction between the two. No factor can be singled out as “the most important”.

	DoF	Sum of squares	F -value	$Pr(F)$
M	2	0.959	1.215	0.297
NH	2	1.006	1.274	0.280
$M : NH$	4	0.346	0.219	0.928
Residuals	1665	657.217		

TABLE XI

ANOVA results comparing the model combination method (noted C : hardmax; softmax; exp. gradient), the model type (noted M , as before), the number of hidden units (noted NH), along with higher-order interactions between these factors. The number of hidden units is the only significant factor.

	DoF	Sum of squares	F -value	$Pr(F)$
C	2	0.673	0.862	0.422
M	2	1.094	1.401	0.246
NH	2	3.365	4.309	0.013 *
$C : M$	4	0.905	0.579	0.678
$C : NH$	4	0.546	0.350	0.844
$M : NH$	4	0.674	0.431	0.786
$C : M : NH$	8	0.302	0.097	0.999
Residuals	4995	1950.545		

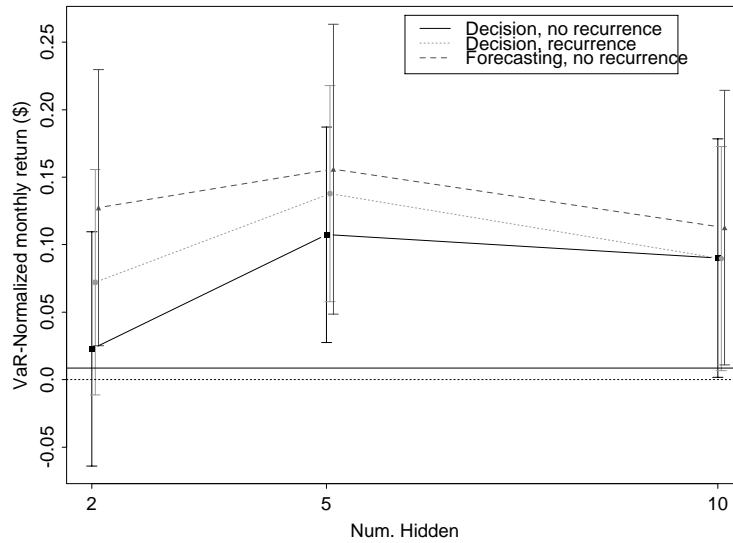


Fig. 8. Out-of-sample performance of committees (made with exponentiated gradient) for three types of models. The market performance is the solid horizontal line just above zero. The error bars denote 95% confidence intervals. We note that the forecasting committee is slightly but not significantly better than the others.

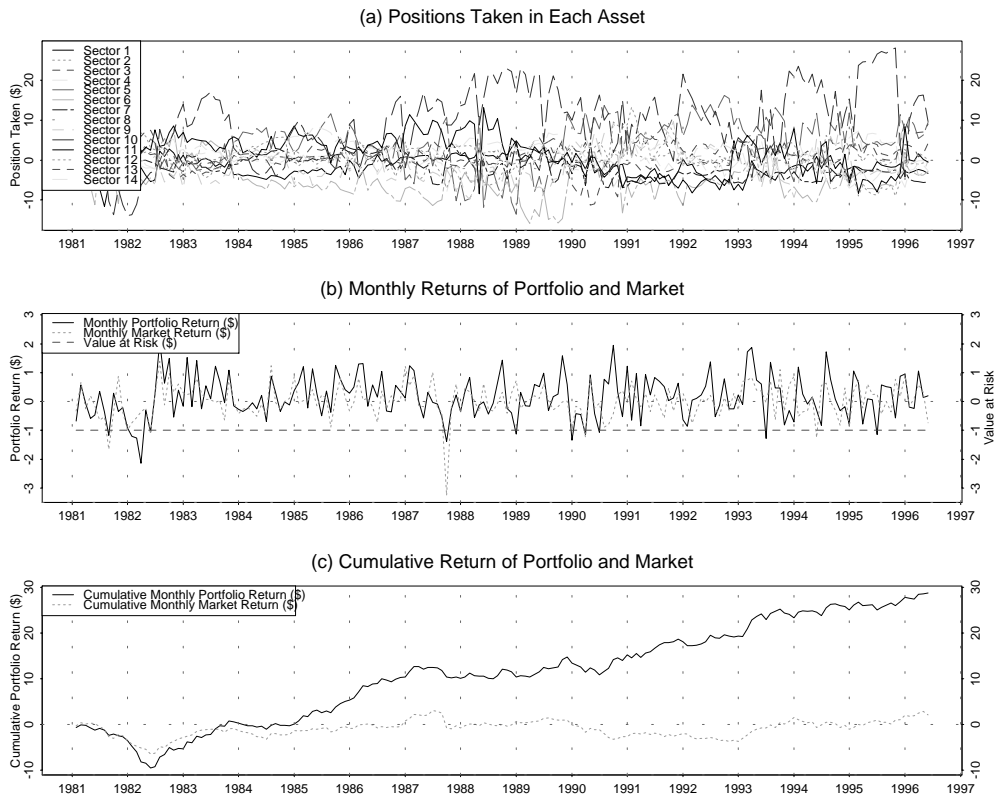


Fig. 9. Out-of-sample behavior of the (exponentiated gradient) committee built upon the forecasting model with 5 hidden units. (a) Monthly positions (in \$) taken in each asset. (b) Monthly return, along with the 95%-VaR (set to 1\$); we note that the risks taken are approximately as expected, from the small number of crossings of the $-1\$$ horizontal line. (c) Cumulative return: the decisions would have been very profitable.

TABLE XII

Analysis of the performance difference between the exponentiated gradient committees and the best underlying model that is part of each committee. We observe that the committees are never significantly worse than the best model they contain.

	Model	Underlying $\langle WD, ID \rangle$	Perf. difference	t -value	DoF	p -value
Decision w/o recur.	NH=2	$10^{-1}, 10^{-1}$	-0.034	-0.77	185	0.43
	NH=5	$10^0, 10^{-3}$	-0.033	-1.65	185	0.10
	NH=10	$10^{-3}, 10^{-1}$	-0.019	-0.82	185	0.41
Decision w/ recur.	NH=2	$10^0, 10^{-3}$	-0.024	-0.71	185	0.47
	NH=5	$10^{-3}, 10^{-2}$	-0.001	-0.05	185	0.95
	NH=10	$10^{-3}, 10^{-3}$	-0.024	-1.59	185	0.11
Forecast w/o recur.	NH=2	$10^{-2}, 10^{-3}$	0.005	0.16	185	0.86
	NH=5	$10^{-2}, 10^{-1}$	-0.007	-0.22	185	0.82
	NH=10	$10^{-3}, 10^{-1}$	-0.015	-0.46	185	0.64

C.1 ANOVA Results for Committees

Tables X and XI formally analyze the impact of the model combination methods.

Restricting ourselves to the exponentiated gradient committees, we first note (table X) that *no factor*, either the model type or the number of hidden units, has a statistically significant effect on the performance of the committees.

Secondly, when we contrast all the combination methods taken together, we note that the number of hidden units has an overall significant effect. This appears to be attributable to the relative weakness of the ‘hardmax’ combination method, even though no direct statistical evidence can confirm this conjecture. The other combination methods—softmax and exponentiated gradient—are found to be statistically equivalent in our results.

C.2 Comparing a Committee with its Underlying Models

We now compare the models formed by the committees (restricting ourselves to the exponentiated gradient combination method) against the performance of their *best underlying* model, and the *average performance* of their underlying models, for all model types and number of hidden units.

Table XII indicates which of the respective underlying models yielded the best performance (*ex post*) for each committee, and tabulates the average difference between the performance of the committee (noted x) and the performance of that best underlying (noted y). Even though a committee suffers in general from a slight performance degradation with respect to its best underlying model, this difference is, in no circumstance, statistically significant. (Furthermore, we note that the best underlying model can never directly be used by itself, since its performance can only be evaluated after the facts.)

Table XIII gives the results of the average performance of the underlying models (noted y) and compares it with the performance of the committee itself (noted x). We note that the committee performance is significantly better in four cases out of nine, and “quasi-significantly” better in two other cases. We observe that comparing a committee to the average performance of its underlying models is equivalent to randomly picking one of the underlyings.

We can conclude from these results that, contrarily to their human equivalents, model committees can be significantly more intelligent than one of their members picked randomly, and can never be (according to our results) significantly worse than the *best* of their members.

VI. CONCLUSION

We demonstrated the viability of directly training a (possibly recurrent) neural network according to a VaR-adjusted profit criterion for making asset-allocation decisions within a VaR-control framework. The performance results are comparable to those obtained with a forecasting model used jointly with classical mean–variance portfolio selection.

We showed the importance of the input decay regularizer as a soft input selection procedure, in the case where networks contain a large number of inputs.

Finally, we noted an effective use of committee methods to systematize the choice of hyperparameters during neural network training. While many of their underlying models are underperformers, we found that several of our committees (both of the forecasting and the decision types) are nevertheless significantly outperforming the benchmark market index.

TABLE XIII

Analysis of the performance difference between the exponentiated gradient committees and the arithmetic mean of the performance of the models that are part of each committee (equivalent to average performance obtained by randomly picking a model from the committee).

For the decision model with recurrence and the forecasting model, we see that the committees frequently significantly outperform the random choice of one of their members.

Model		Average of underlyings	Perf. Difference	t -value	DoF	p -value
Decision w/o recur.	NH=2	0.033 (0.033)	-0.012	-0.539	185	0.591
	NH=5	0.078 (0.034)	0.026	1.595	185	0.112
	NH=10	0.070 (0.040)	0.016	1.834	185	0.068
Decision w/ recur.	NH=2	0.043 (0.030)	0.025	1.106	185	0.270
	NH=5	0.087 (0.032)	0.049	3.156	185	0.002 *
	NH=10	0.057 (0.039)	0.032	2.653	185	0.009 *
Forecast w/o recur.	NH=2	0.095 (0.042)	0.030	2.008	185	0.046 *
	NH=5	0.089 (0.040)	0.065	3.471	185	0.001 *
	NH=10	0.079 (0.040)	0.031	1.902	185	0.059

REFERENCES

- [1] Y. Bengio, "Training a neural network with a financial criterion rather than a prediction criterion," In Weigend et al. [22].
- [2] M. Choey and A. S. Weigend, "Nonlinear trading models through sharpe ratio maximization," In Weigend et al. [22].
- [3] J. Moody and L. Wu, "Optimization of trading systems and portfolios," In Weigend et al. [22].
- [4] P. Jorion, *Value at Risk: The New Benchmark for Controlling Market Risk*, Irwin McGraw-Hill, 1997.
- [5] A. A. Gaivoronski and G. Pflug, "Finding optimal portfolios with constraints on value at risk," in *Proceedings III Stockholm Seminar on Risk Behavior and Risk Management*, Stockholm, 1999.
- [6] R. A.J. Pownall, R. Huisman, and Kees G. Koedijk, "Asset allocation in a value-at-risk framework," in *Proceedings of the European Finance Association Conference*, Helsinki, Finland, 1999.
- [7] B.W. Silverman, *Density Estimation for Statistics and Data Analysis*, Chapman & Hall, London, 1986.
- [8] Malik Magdon-Ismael and Amir Atiya, "Neural networks for density estimation," in *Advances in Neural Information Processing Systems*, 1998.
- [9] N. Chapados, "Optimization criteria for learning algorithms in portfolio selection," M.S. thesis, Universit de Montral, Montral, Canada, January 2000.
- [10] W. F. Sharpe, "The sharpe ratio," *The Journal of Portfolio Management*, vol. 21, no. 1, pp. 49–58, 1994.
- [11] RiskMetrics, "Technical document, fourth edition," Tech. Rep., J.P. Morgan, New York, NY, 1996, <http://www.riskmetrics.com>.
- [12] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes in C : The Art of Scientific Computing*, Cambridge University Press, Cambridge, second edition, 1992.
- [13] D.E. Rumelhart, G.E. Hinton, and R.J. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing*, D.E. Rumelhart and J.L. McClelland, Eds., vol. 1, chapter 8, pp. 318–362. MIT Press, Cambridge, 1986.
- [14] H. M. Markowitz, *Portfolio Selection: Efficient Diversification of Investments*, John Wiley & Sons, 1959.
- [15] Yaser S. Abu-Mostafa, "Hints," *Neural Computation*, vol. 7, no. 4, pp. 639–671, July 1995.
- [16] G.E. Hinton, "Learning translation invariant in massively parallel networks," in *Proceedings of PARLE Conference on Parallel Architectures and Languages Europe*, J.W. de Bakker, A.J. Nijman, and P.C. Treleaven, Eds., Berlin, 1987, pp. 1–13, Springer-Verlag.
- [17] A. S. Weigend, D. E. Rumelhart, and B. A. Huberman, "Back-propagation, weight-elimination and time series prediction," in *Connectionist Models: Proceedings of the 1990 Summer School*, D.S Touretzky, J. L. Elman, T. J. Sejnowski, and G. E. Hinton, Eds., San Mateo, CA, 1991, Morgan Kaufmann.
- [18] M. Herbster and M. K. Warmuth, "Tracking the best expert," *Machine Learning*, vol. 32, no. 2, 1998.
- [19] J. Kivinen and M. K. Warmuth, "Additive versus exponentiated gradient updates for linear prediction," *Information and Computation*, vol. 132, no. 1, pp. 1–64, 1997.
- [20] Halbert White, "A reality check for data snooping," *Econometrica*, 2000, To be published.
- [21] H. Scheffé, *The Analysis of Variance*, John Wiley & Sons, New York, NY, 1959.
- [22] A. S. Weigend, Y. Abu-Mostafa, and A.-P. Refenes, Eds., *Decision Technologies for Financial Engineering: Proceedings of the Fourth International Conference on Neural Networks in the Capital Markets (NNCM '96)*. World Scientific Publishing, 1997.

Liste des publications au CIRANO*

Série Scientifique / *Scientific Series* (ISSN 1198-8177)

- 2002s-49 Cost Functions and Model Combination for VaR-based Asset Allocation using Neural Networks / N. Chapados et Y. Bengio
- 2002s-48 Experiments on the Application of IOHMMs to Model Financial Returns Series / Y. Bengio, V.-P. Lauzon et R. Ducharme
- 2002s-47 Valorisation d'Options par Optimisation du Sharpe Ratio / Y. Bengio, R. Ducharme, O. Bardou et N. Chapados
- 2002s-46 Incorporating Second-Order Functional Knowledge for Better Option Pricing / C. Dugas, Y. Bengio, F. Bélisle, C. Nadeau et R. Garcia
- 2002s-45 Étude du biais dans le Prix des Options / C. Dugas et Y. Bengio
- 2002s-44 Régularisation du Prix des Options : Stacking / O. Bardou et Y. Bengio
- 2002s-43 Monotonicity and Bounds for Cost Shares under the Path Serial Rule / Michel Truchon et Cyril Tétéjédo
- 2002s-42 Maximal Decompositions of Cost Games into Specific and Joint Costs / Michel Moreaux et Michel Truchon
- 2002s-41 Maximum Likelihood and the Bootstrap for Nonlinear Dynamic Models / Sílvia Gonçalves, Halbert White
- 2002s-40 Selective Penalization Of Polluters: An Inf-Convolution Approach / Ngo Van Long et Antoine Soubeyran
- 2002s-39 On the Mediational Role of Feelings of Self-Determination in the Workplace: Further Evidence and Generalization / Marc R. Blais et Nathalie M. Brière
- 2002s-38 The Interaction Between Global Task Motivation and the Motivational Function of Events on Self-Regulation: Is Sauce for the Goose, Sauce for the Gander? / Marc R. Blais et Ursula Hess
- 2002s-37 Static Versus Dynamic Structural Models of Depression: The Case of the CES-D / Andrea S. Riddle, Marc R. Blais et Ursula Hess
- 2002s-36 A Multi-Group Investigation of the CES-D's Measurement Structure Across Adolescents, Young Adults and Middle-Aged Adults / Andrea S. Riddle, Marc R. Blais et Ursula Hess
- 2002s-35 Comparative Advantage, Learning, and Sectoral Wage Determination / Robert Gibbons, Lawrence F. Katz, Thomas Lemieux et Daniel Parent
- 2002s-34 European Economic Integration and the Labour Compact, 1850-1913 / Michael Huberman et Wayne Lewchuk
- 2002s-33 Which Volatility Model for Option Valuation? / Peter Christoffersen et Kris Jacobs
- 2002s-32 Production Technology, Information Technology, and Vertical Integration under Asymmetric Information / Gamal Atallah

* Consultez la liste complète des publications du CIRANO et les publications elles-mêmes sur notre site Internet :

- 2002s-31 Dynamique Motivationnelle de l'Épuisement et du Bien-être chez des Enseignants Africains / Manon Levesque, Marc R. Blais, Ursula Hess
- 2002s-30 Motivation, Comportements Organisationnels Discrétionnaires et Bien-être en Milieu Africain : Quand le Devoir Oblige / Manon Levesque, Marc R. Blais et Ursula Hess
- 2002s-29 Tax Incentives and Fertility in Canada: Permanent vs. Transitory Effects / Daniel Parent et Ling Wang
- 2002s-28 The Causal Effect of High School Employment on Educational Attainment in Canada / Daniel Parent
- 2002s-27 Employer-Supported Training in Canada and Its Impact on Mobility and Wages / Daniel Parent
- 2002s-26 Restructuring and Economic Performance: The Experience of the Tunisian Economy / Sofiane Ghali and Pierre Mohnen
- 2002s-25 What Type of Enterprise Forges Close Links With Universities and Government Labs? Evidence From CIS 2 / Pierre Mohnen et Cathy Hoareau
- 2002s-24 Environmental Performance of Canadian Pulp and Paper Plants : Why Some Do Well and Others Do Not ? / Julie Doonan, Paul Lanoie et Benoit Laplante
- 2002s-23 A Rule-driven Approach for Defining the Behavior of Negotiating Software Agents / Morad Benyoucef, Hakim Alj, Kim Levy et Rudolf K. Keller
- 2002s-22 Occupational Gender Segregation and Women's Wages in Canada: An Historical Perspective / Nicole M. Fortin et Michael Huberman
- 2002s-21 Information Content of Volatility Forecasts at Medium-term Horizons / John W. Galbraith et Turgut Kisinbay
- 2002s-20 Earnings Dispersion, Risk Aversion and Education / Christian Belzil et Jörgen Hansen
- 2002s-19 Unobserved Ability and the Return to Schooling / Christian Belzil et Jörgen Hansen
- 2002s-18 Auditing Policies and Information Systems in Principal-Agent Analysis / Marie-Cécile Fagart et Bernard Sinclair-Desgagné
- 2002s-17 The Choice of Instruments for Environmental Policy: Liability or Regulation? / Marcel Boyer, Donatella Porrini
- 2002s-16 Asymmetric Information and Product Differentiation / Marcel Boyer, Philippe Mahenc et Michel Moreaux
- 2002s-15 Entry Preventing Locations Under Incomplete Information / Marcel Boyer, Philippe Mahenc et Michel Moreaux
- 2002s-14 On the Relationship Between Financial Status and Investment in Technological Flexibility / Marcel Boyer, Armel Jacques et Michel Moreaux
- 2002s-13 Modeling the Choice Between Regulation and Liability in Terms of Social Welfare / Marcel Boyer et Donatella Porrini
- 2002s-12 Observation, Flexibilité et Structures Technologiques des Industries / Marcel Boyer, Armel Jacques et Michel Moreaux
- 2002s-11 Idiosyncratic Consumption Risk and the Cross-Section of Asset Returns / Kris Jacobs et Kevin Q. Wang
- 2002s-10 The Demand for the Arts / Louis Lévy-Garboua et Claude Montmarquette
- 2002s-09 Relative Wealth, Status Seeking, and Catching Up / Ngo Van Long, Koji Shimomura