Deutsche Bank
Markets Research

Asia

Quantitative Strategy
Quantiles

Date
18 October 2017

End-to-end portfolio construction

Vincent Zoonekynd
vincent.zoonekynd@db.com

Khoi LeBinh
khoi.lebinh@db.com

Jiazi Tang, CFA
jiazi.tang@db.com

Hemant Sambatur
hemant.sambatur@db.com

Elita Lai
elita.lai@db.com

North America: +1 212 250 8983
Europe: +44 20 754 71684
Asia: +852 2203 6990

2-step processes are often suboptimal

Portfolio construction is often a 2-step process: first model future returns, then compute the optimal portfolio – can we devise a 1-step, end-to-end portfolio construction process?

Meaningful objectives

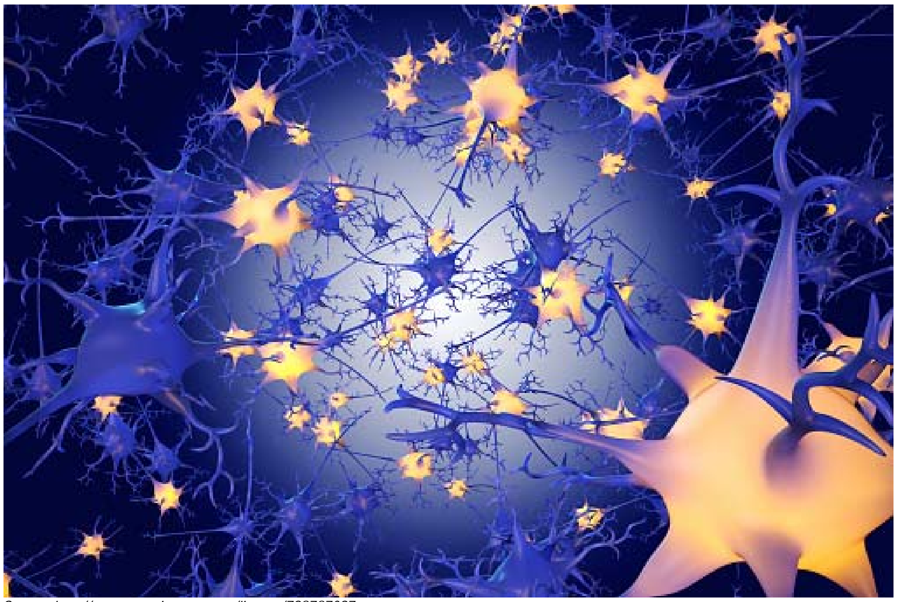
Portfolio optimization often focuses on easy-to-compute, single-period performance measures: instead, can we optimize arbitrary performance measures, closer to what investors care about?

Deep learning and differentiable portfolio construction

The deep learning hype promises to optimize any differentiable objective: can we design a differentiable portfolio construction procedure to leverage those tools and achieve end-to-end portfolio construction?

Regularization

We will answer those questions in the affirmative. We will also show how regularization techniques (mini-batches, dropout, ensembling, transaction costs) can further boost the performance of the resulting strategy and combine those ideas (linear model with sign constraints, estimated by Tensorflow to maximize the after-transaction-costs information ratio, regularized with mini-batches and ensembling) into a strategy outperforming the constrained regression of our *Machine Learning in Finance* report.



Source: http://www.gettyimages.com/license/738787037

Table Of Contents

A letter to our readers

OH: "I don't know why it's always a quadratic loss function.
I've never known anything to have a quadratic loss."
J.D. Cook (@DataSciFact)

The deep learning hype we have been listening to for a few years claims that computers can now learn from raw data, with no need for practitioners to provide intermediate steps. Those ideas have been successfully applied to computer vision, speech, video games, self-driving cars, robots, etc. – new areas of applications emerge every day.

What about Finance?

While the limited volume of data and non-stationarity issues call for extreme caution, some ideas and tools from the deep learning world can be fruitfully applied to Finance. In particular, they can address the following perennial problems.

- When optimizing a portfolio, we often limit ourselves to very simple objectives, such as maximizing returns, minimizing volatility or a combination of those. While there are other interesting objectives (e.g., risk parity and its variants), each is considered as a separate, independent problem, calling for an ad hoc analysis and, often, bespoke optimization algorithms. Deep learning, in contrast, calls for more freedom and experimentation in the choice of the objective, and for all objectives to be treated in the same way.
- Portfolio construction is often performed as a succession of well-understood, theoretically-motivated steps – risk model estimation, return forecast, portfolio optimization, to name a few. While this is appealing and safe, multi-step procedures tend to be sub-optimal. In particular, each step has to make a compromise between the simplicity of the model and the complexity of the data – but the compromises of different steps may be at odds with one another. Combining those steps into a single "pipeline" and asking the computer to fine-tune them, together, could lead to some improvements.

This report has four main parts.

In the first part, we present and compare several procedures to build portfolios from our investment signals, by computing a linear combination of those signals and using it to for quintile portfolios or to compute portfolio weights by applying some non-linear function. Since we can freely choose the performance measure we want to optimize, we opt for the (after-transaction-costs) information ratio.

In the second part, we review and compare the different types of optimization algorithms.

In a third part, we explain how to put our model in a deep learning framework. It may seem overkill for such a simple model, but this opens new possibilities:

- We can now use all the "tricks" developed by the deep learning community to improve optimization results;
- Since those frameworks allow arbitrary complex computation graphs, we could add other steps to the portfolio construction procedure, e.g., portfolio optimization, risk model estimation or trade implementation – but we leave this for another report.

In a fourth part, we review and combine several regularization methods, in particular mini-batches, dropout, penalties (such as transaction costs), early stopping and ensembling.

Finally, we combine all those ideas (linear model with sign constraints, estimated by Tensorflow to maximize the after-transaction-costs information ratio, regularized with mini-batches and ensembling): the resulting model is similar to that from our previous machine learning report (a constrained regression), with a few differences in the investment factors used (in particular, the fast-changing, turnover-inducing factors are no longer there); the returns are similar, but the volatility and turnover are lower, leading to a much improved information ratio.

Yours sincerely,

Vincent, Khoi, Hemant, Jiazi, Elita & the Global Quantitative Strategy Team.

Deutsche Bank Asia Quantitative Strategy Team.

Screen

The following figure shows the largest stocks currently in the long and short portfolios of the strategy developed in this report. We used the Japanese market (S&P BMI) as an example.

Figure 1: Large caps (more than 10 billion USD) with the largest weights in our long and short portfolios as of 2017-10-10. We also show some of the investment factors entering the model (B/P, cash flow and EPS diffusion), rescaled to be between -100 and +100.

side	Id	Name	MCap	Sector	Weight (bp)	B/P	CF	Diffusion
long	8058–JP	Mitsubishi	36.8	Industrials	41	88	83	60
long	8053–JP	Sumitomo	17.5	Industrials	38	88	38	81
long	6501–JP	Hitachi	34.8	Information Technology	36	61	76	62
long	5411–JP	JFE Holdings	11.5	Materials	33	94	64	61
long	5401–JP	Nippon Steel & Sumitomo Metal	20.8	Materials	31	91	86	49
long	8031–JP	Mitsui	26.2	Industrials	30	91	83	62
long	6902–JP	DENSO CORPORATION	27.1	Consumer Discretionary	30	54	36	59
long	4005–JP	Sumitomo Chemical	10.2	Materials	23	56	73	81
long	8591–JP	ORIX	21.4	Financials	23	83		55
long	5020–JP	JXTG Holdings.	16.3	Energy	22	85	98	−46
short	8697–JP	Japan Exchange Group	10.0	Financials	−57	−88		−62
short	7309–JP	Shimano	11.0	Consumer Discretionary	−51	−84	−57	−88
short	6869–JP	Sysmex	10.1	Health Care	−49	−95	−76	−64
short	7733–JP	Olympus	11.4	Health Care	−42	−79	−35	−61
short	2503–JP	Kirin Holdings	22.0	Consumer Staples	−36	−82	22	49
short	2914–JP	Japan Tobacco	36.6	Consumer Staples	−36	−76	−21	−65
short	4324–JP	Dentsu	11.3	Consumer Discretionary	−36	48	−34	−73
short	4523–JP	Eisai	15.5	Health Care	−35	−81	−58	46
short	2269–JP	Meiji Holdings	12.0	Consumer Staples	−31	−78	−40	−68

Source: Factset, S&P, Thomson Reuters, IHS Markit, Deutsche Bank Quantitative Strategy Research

Introduction

First motivation: 2-step procedures tend to be sub-optimal

Portfolio construction is often a 2-step process – first estimate the returns distribution, then build a portfolio – but 2-step procedures tend to be sub-optimal.

Portfolio construction is often a two-step process [27]:

- First forecast stock [1] returns or, more precisely, describe the distribution of future returns;
- Then build a portfolio.

Furthermore, each of those steps requires us to choose a model and an optimization criterion.

For instance,

- The first step could be a Gaussian model, with the expected returns computed as a linear combination of "investment factors" (momentum, P/E, etc.), chosen to minimize the squared error between the forecasted and the realized returns, and with the variance matrix from a factor model.
- The criterion in the second step could be the information ratio.

But the first criterion looks arbitrary: wouldn't we get better results by minimizing the absolute (un-squared) error, or forecasting the rank of the stocks, or the sign of their returns, or something else? This first criterion considers all stocks equally, trying to forecast returns with the same precision for all of them – but once we have decided that a stock would not be in our portfolio, any increase in precision is wasted: since the capacity of the model is limited, it is much preferable to increase the precision for the remaining stocks instead.

The first step is not aware of the final goal.

Our goal is to build a portfolio or a strategy, not to have a list of forecasts, for all stocks, with the same level of precision for all of them.

$$\begin{tikzcd}\text{Data} \arrow[r, "\text{criterion}_1"] & \begin{array}{c}\text{Return}\\\text{forecasts}\end{array} \arrow[r, "\text{criterion}_2"] & \text{Portfolio}\end{tikzcd}$$

This is yet another example of a sub-optimal multi-step process: can we replace it with a one-step process?

$$\begin{tikzcd}\text{Data} \arrow[rr, "\text{optimize criterion}_2"] & & \text{Portfolio}\end{tikzcd}$$

[1] In this report, we use stocks as an example, but the methodology we will present is general and applies to other assets as well.

Why are we still doing that?

Currently available computing power and data now allows 1-step portfolio construction.

The traditional 2-step approach is still widely used because (in no particular order)

- It has been advocated by textbooks and relied on by practitioners for decades – it may not be optimal, but it is known to work well enough;
- It has good theoretical properties: least squares, maximum likelihood, modern portfolio theory, etc.;
- Computers used not to be powerful enough to do anything else.

But times have changed:

- Computers are now more powerful, and software is more readily available, allowing more freedom in the portfolio construction procedure;
- The availability of more data and methods to control overfitting makes those new tools relatively safe and allows us to explore data more thoroughly.

Second motivation: arbitrary performance measures

End-to-end portfolio construction lets you freely choose the objective function to reflect your return requirements, risk appetite and cost.

There are a few ways of combining several investment signals with different horizons, *while controlling for transaction costs*:

- Linear combination of two signals: (1-λ)×Slow + λ ×Fast;
- Intersection of the quintile portfolios [16]: buy stocks with high Fast and Slow scores – if there are more than two signals, you could buy stocks that are in the top quintile for at least k signals, for some value of k.
- Partial rebalancing [28, 37]:

$$w_{t+1} \leftarrow (1-\lambda)w_t + \lambda w_{t+1}^{\text{ideal}}$$

While there can be theoretical justifications for some of them, they look like heuristics: we hope they will work, perhaps after some tweaking, but remain half-convinced.

The end-to-end approach advocated in this report directly addresses this problem: since we can use any objective function, we can simply use one that takes transaction costs into account.

There are many measures of performance and risk investors may want to optimize [31], and many properties of the resulting portfolio they may want to enforce: risk parity, diversification, etc. [2]. If your portfolio construction framework allows you to specify an arbitrary objective function, you can ask for a portfolio with good returns, low volatility, low value-at-risk, low drawdown, well-diversified, and close to risk parity.

Third motivation: end-to-end differentiability

Complex portfolio construction pipelines, provided they are differentiable, can be estimated with deep learning toolboxes.

For a couple of years, a promising approach to many complicated machine learning problems has been to transform the problem to make it differentiable (differentiable data structures [20], differentiable memory, differentiable Turing machines [19], etc.) and use some gradient-based optimization algorithm (stochastic gradient descent or its many variants: preconditioning [9], momentum [24, 33], variance reduction [11, 23], natural gradient [29], etc.).

The different steps to be learnt are often arranged into a "pipeline", which can now be learnt end-to-end. The elements of this pipeline are surprisingly varied: besides the traditional neural activation functions (ReLU, tanh), we can find normalization layers, memory cells (LSTM, GRU), implicitly-defined functions, optimizations, machine learning algorithms (k-means), etc.

Can we do the same thing for portfolio construction?

Fourth motivation: functional (formulaic) portfolios

Our model will be a generalization of value-weighted portfolios, but with 100 investment factors instead of just one.

The idea of using a formula to compute the weights of a portfolio, from some signals about the stocks, is not new: *value-weighted portfolios* [1] are a simple example.

It can even sometimes be given a theoretical justification: for instance, *stochastic portfolio theory* [15, 39, 13, 14] studies what happens when the weights are proportional to some power of the capitalization.

Here, we apply this idea to a large number of signals instead of just one or two [34].

Structure of the study

We will first define the setup of this study:

- Define a few reference strategies: we will try to beat the long-short quintile portfolios, for an equal-weighted signal, and for the constrained linear regression model introduced in our previous report [41];
- List a few portfolio construction procedures: for instance, our usual long-short quintile portfolio, but we also want a few differentiable ones, to be able to use those trendy "deep learning" tools;
- Decide on the quantity to be optimized: we will settle on the information ratio.

Then, in a first part, we will compare various portfolio construction procedures ("formulaic portfolios") and select the most promising – it will turn out to be, disappointingly but unsurprisingly, a linear model.

In a second part, we will review the most common optimization algorithms – a few robust ones, such as differential evolution, and some relying on the smoothness of the objective function, such as gradient descent or Newton's method – and compare them. We will see that they give similar results, but that those using first or second order information tend to converge faster.

We will also see that some of the intermediate portfolios, before the algorithms have converged, even though sub-optimal in-sample, have decent and better out-of-sample performance than more optimized portfolios.

In a third part, we will estimate those models with stochastic gradient descent (SGD), in a deep learning framework (e.g., TensorFlow [18]), after ensuring everything is differentiable.

In a fourth part, we will show a few ways of regularizing the results, e.g., with dropout, penalties, ensembling or transaction costs.

Setup

Data

- S&P BMI Japan
- Monthly stock-level data: our usual 100 investment factors, uniformized
- In-sample period: 1990-01 to 2009-12
- Out-of-sample period: 2010-01-31 to 2017-07-31

Strategies to beat

We will present a strategy outperforming the constrained regression model of our previous report, in terms of (after-cost) information ratio.

We will compare the approach we suggest with the following strategies:

- Long-short quintile portfolios from an equal-weighted signal, using all the investment factors in our library;
- Long-short quintile portfolios from a constrained regression, estimated on an expanding window, as introduced in our report on *Machine learning in finance* [41].

To compare with long-only strategies, we will also consider the corresponding long-only (top quintile) portfolios.

While the constrained regression model performs well on paper, it is not optimal: it tries to provide the best possible return forecasts, for all stocks, regardless of whether they will end in the portfolio, and, to this end, makes compromises, increasing the error for one stock to reduce it for another – but those compromises may be at odds with our final goal, which is a portfolio or a strategy, not a list of individual forecasts.

In addition, there are concerns that the turnover may be too high and that the transaction costs could eat all the performance.

Objective

We will focus on two measures of performance:

- The information ratio (IR);
- The information ratio after transaction costs, as the transaction costs increase from 0bp to 100bp.

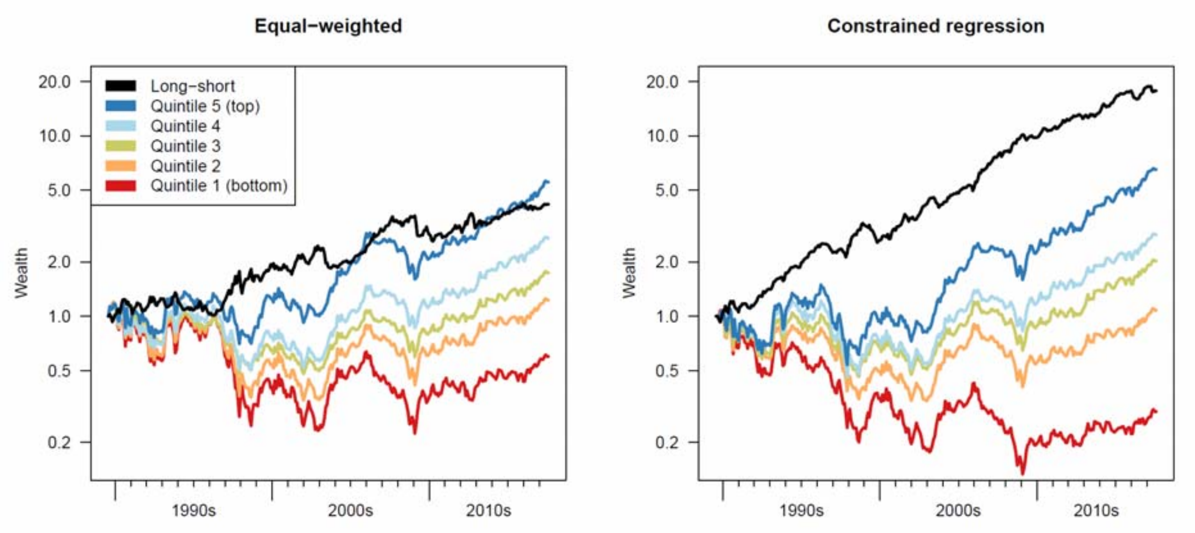
We will maximize the information ratio on the in-sample period, for a few portfolio construction recipes, using various optimization methods, tools and tricks, and check the performance of the optimal strategy on the out-of-sample period.

Of course, we could have considered many other performance ratios [42], or even objectives that are not performance-related, such as risk parity or diversification.

Limitations

We are looking for a *static* model: there will be no attempt at market timing – but, with care, the same ideas could be used to build a more dynamic model.

Figure 2: Performance of the reference strategies for the period 2010-present.
The equal-weighted strategy gives the same weights to all investment factors. The constrained regression is estimated on a expanding window. We only show the performance numbers for the period 2010-present, which will be the out-of-sample period for the models developed in this report.



Equal−weighted

Portfolio	CumulativeReturn	CAGR (%)	AnnVol (%)	IR	tstat	Skew	Kurtosis	HitRatio (%)	MaxDD (%)	VaR95 (%)	ES95 (%)
LS	49.1	5.5	10.1	0.54	1.6	−0.92	1.9	58	−17.1	−4.9	−6.7
5	160.7	13.6	12.6	1.08	3.0	0.02	−0.1	68	−9.3	−4.8	−6.2
4	133.2	11.9	12.3	0.97	2.7	0.02	0.1	64	−10.4	−4.9	−6.4
3	110.7	10.4	12.2	0.85	2.4	−0.08	0.2	59	−9.2	−4.6	−6.9
2	93.3	9.2	13.8	0.66	1.9	−0.02	0.4	54	−13.2	−5.7	−8.2
1	63.0	6.7	15.9	0.42	1.3	0.12	0.2	49	−25.0	−6.2	−8.8

Constrained regression

Portfolio	CumulativeReturn	CAGR (%)	AnnVol (%)	IR	tstat	Skew	Kurtosis	HitRatio (%)	MaxDD (%)	VaR95 (%)	ES95 (%)
LS	80.0	8.1	7.3	1.11	3.0	−0.59	0.6	67	−10.0	−2.9	−4.2
5	177.2	14.5	12.5	1.16	3.2	−0.06	0.0	64	−10.2	−4.2	−6.4
4	133.5	11.9	12.7	0.94	2.6	−0.06	−0.2	60	−9.6	−5.1	−6.8
3	124.9	11.4	12.4	0.91	2.6	0.03	0.3	62	−10.1	−5.3	−6.9
2	90.6	9.0	12.9	0.69	2.0	0.03	0.3	57	−11.6	−5.5	−7.3
1	47.2	5.3	15.2	0.35	1.1	0.15	0.2	54	−18.1	−6.7	−8.3

Source: Factset, S&P, Thomson Reuters, IHS Markit, Deutsche Bank Quantitative Strategy Research

Models

In this section,

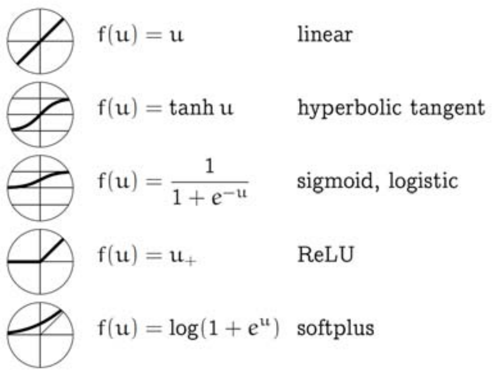
- We define a few portfolio construction recipes, such as the classical long-short quintile strategy, or linear weights;
- We optimize them using traditional optimization algorithms, namely
  - random search, to have a vague idea of how varied the portfolios generated by the recipe can be,
  - and Nelder-Mead, an optimization algorithm providing a good compromise between speed (slow, but not too much) and robustness (it can deal with non-differentiable functions), to explore more deeply the promising regions of the optimization landscape.

The best-performing portfolio construction recipe will turn out to be... linear.

Formulaic portfolios

The neural network community uses a few **transfer functions** to introduce non-linearities in neural networks (Figure 3). For instance, there are two S-shaped functions, the hyperbolic tangent, mapping the real line **R** to the interval (-1,+1), and the sigmoid (or logistic) function, mapping **R** to (0,1); the restricted linear unit (ReLU) only keeps the positive part of its input and zeroes out the negative part; the softplus is a smooth approximation of the ReLU function.

Figure 3: Transfer functions commonly used in neural networks.
Neural networks are just compositions of linear and non-linear functions, e.g., $x \mapsto Ax \mapsto f(Ax) \mapsto Bf(Ax) \mapsto g(Bf(Ax)) \mapsto$ ..., where the non-linear functions are chosen from a short list of "transfer functions".



$f(u) = u$	linear
$f(u) = \tanh u$	hyperbolic tangent
$f(u) = \dfrac{1}{1+e^{-u}}$	sigmoid, logistic
$f(u) = u_+$	ReLU
$f(u) = \log(1+e^{u})$	softplus

Source: Deutsche Bank Quantitative Strategy Research

Most of our models will compute a linear combination of our (uniformized) investment signals, to which we apply one of those non-linearities. After normalization, the result can be used as portfolio weights. Since the measures good at identifying out- and under-performing stocks may be different, we also consider separate signals for the long and short portfolios.

Our formulaic portfolios are obtained by applying a non-linearity (a transfer function) to a linear combination of investment factors – it can be seen as a 1-layer neural net.

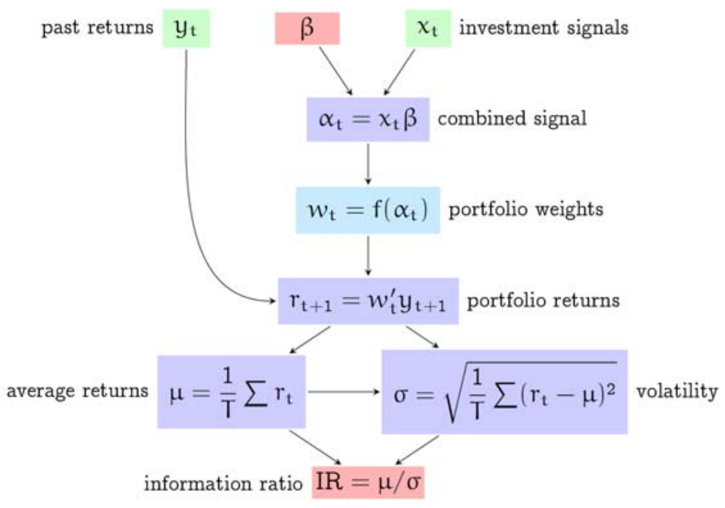
Here are the models we will consider, computing the portfolio weights w_i from investment signals x_i.

- Long-short quintile portfolio, using $\Sigma \beta_i x_i$ as a signal;
- Long-short quintile portfolio, using two signals, $\Sigma \beta_i x_i$ and $\Sigma \gamma_i x_i$ for the long and short parts;
- **Linear weights**: $w = w_+ - w_-$, where $w_\pm \propto (\Sigma \beta_i x_i)_\pm$ and $t_\pm = \text{Max}(\pm t, 0)$;
- Sigmoid weights: $w \propto \sigma(\alpha + \Sigma \beta_i x_i)$, where $\sigma(t) = 1/(1+e^{-t})$;
- ReLU weights: $w \propto (\Sigma \beta_i (x_i - \alpha_i))_+$
- Softplus weights: $w \propto \text{softplus}\ \Sigma \beta_i (x_i - \alpha_i)$, $\text{softplus}(t) = \log(1+e^t)$;
- Tanh weights: $w \propto \tanh(\alpha + \Sigma \beta_i x_i)$ (the long and short portfolios are normalized separately);
- Softmax weights: $w \propto \exp(\alpha + \Sigma \beta_i x_i)$
- Long-short sigmoid (resp. ReLU, softmax): separate long and short portfolios, each with sigmoid (resp. ReLU, softmax) weights;
- The long leg of the long-short portfolios above.

As in our previous report, **all the models are constrained**, i.e., the sign of each β_i is imposed from what we know and expect from each factor. This ensures the model remains **interpretable** and **prevents overfitting**.

These are static models: we are not trying to time the market.

Figure 4: A simple portfolio construction procedure. The inputs and outputs are in red, the data in green, the inner nodes in blue.
Starting with investment factors xi (given, for each stock) and model parameters β (to be estimated), we first compute a linear combination of those factors, $\alpha_t = x_t \beta$, then convert this combined signal into portfolio weights $w_t = f(\alpha_t)$ by rescaling and/or applying one of the transfer functions. Once we have the portfolio, we can compute its returns $r_{t+1} = w'_t y_{t+1}$, for all t, and finally the information ratio. The models examined in this report can be described by this directed acyclic graph (DAG).


past returns y_t
β
x_t investment signals
$\alpha_t = x_t\beta$ combined signal
$w_t = f(\alpha_t)$ portfolio weights
$r_{t+1} = w'_t y_{t+1}$ portfolio returns
average returns $\mu = \frac{1}{T}\sum r_t$
$\sigma = \sqrt{\frac{1}{T}\sum(r_t-\mu)^2}$ volatility
information ratio $IR = \mu/\sigma$

Source: Deutsche Bank Quantitative Strategy Research

Traditional optimization algorithms: random search and Nelder-Mead

Nelder-Mead is a good first choice for an optimization algorithm, provided we have a good starting point and let the algorithm run for long enough.

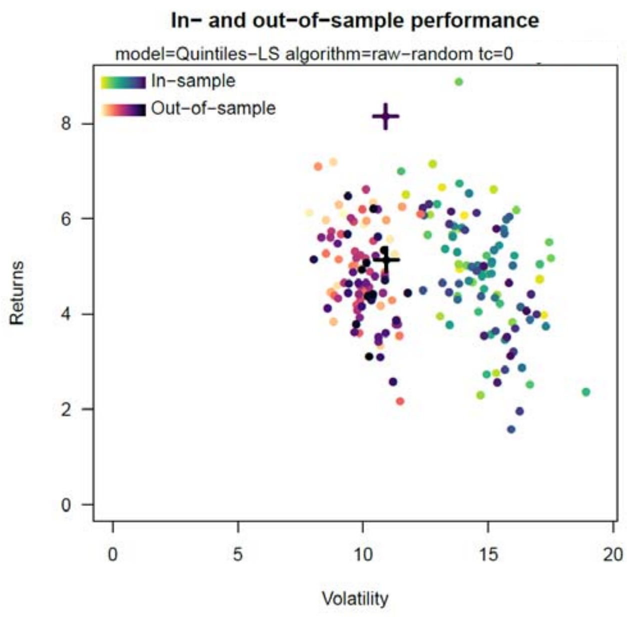
To fix the ideas, let us first consider the long-short quintile portfolio strategies. We want to find the model parameters (β, in Figure 4) maximizing the information ratio – it is an optimization problem. The simplest optimization "algorithm" is to take parameters at random, a very large number of times, and keep the set of parameters with the best performance (*random search*): this is what we have done on Figure 5, which shows long-short quintile portfolios built from linear combinations of our investment signals, with random coefficients. The best of 500 of those random strategies has already a decent performance. We also notice, as expected, that the out-of-sample performance is not as good as the in-sample one: the returns are lower. (The volatility is also lower, but that is only because the in-sample contained a few crises.)

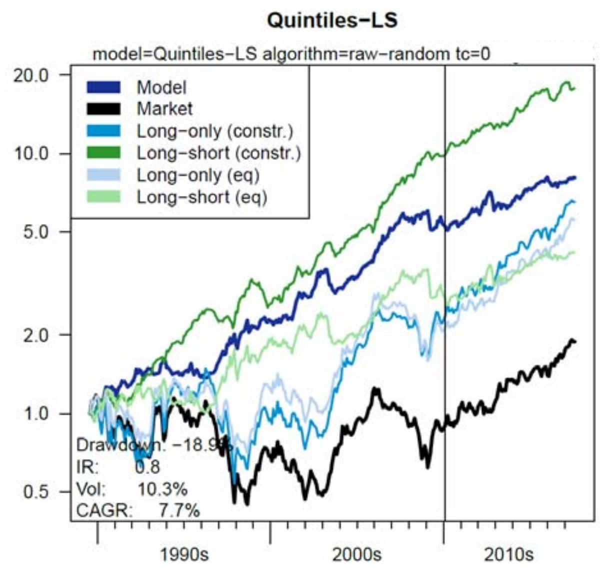
Figure 6 shows the portfolios examined by the *Nelder-Mead algorithm*, run for 500 steps. It is an iterative optimization algorithm, which starts with a sub-optimal solution and progressively improves it, step by step. There are two problems. First, the portfolio performance hardly moves – since the starting point is random, and therefore far from optimal, this suggests the algorithm has not converged. Second, the starting point seems to play a big role. Figure 7 is similar, but starts at the best of 500 strategies; in addition, Figure 8 lets the optimization run for 10,000 steps – the performance improves on that of the random strategy.

Figure 9 shows, more precisely, the effect of the starting point, and suggests to run the same optimization several times, with different starting points. Reassuringly, the best in-sample strategy is also the best out-of-sample one (but that is not guaranteed, of course).

In case we want a wider array of investment strategies, instead of just the maximum information one, we can maximize $\mu - \lambda \sigma^2$, for several values of λ, instead of IR=μ / σ. Figure 10 shows the resulting efficient frontier, and how it shrinks out of sample.

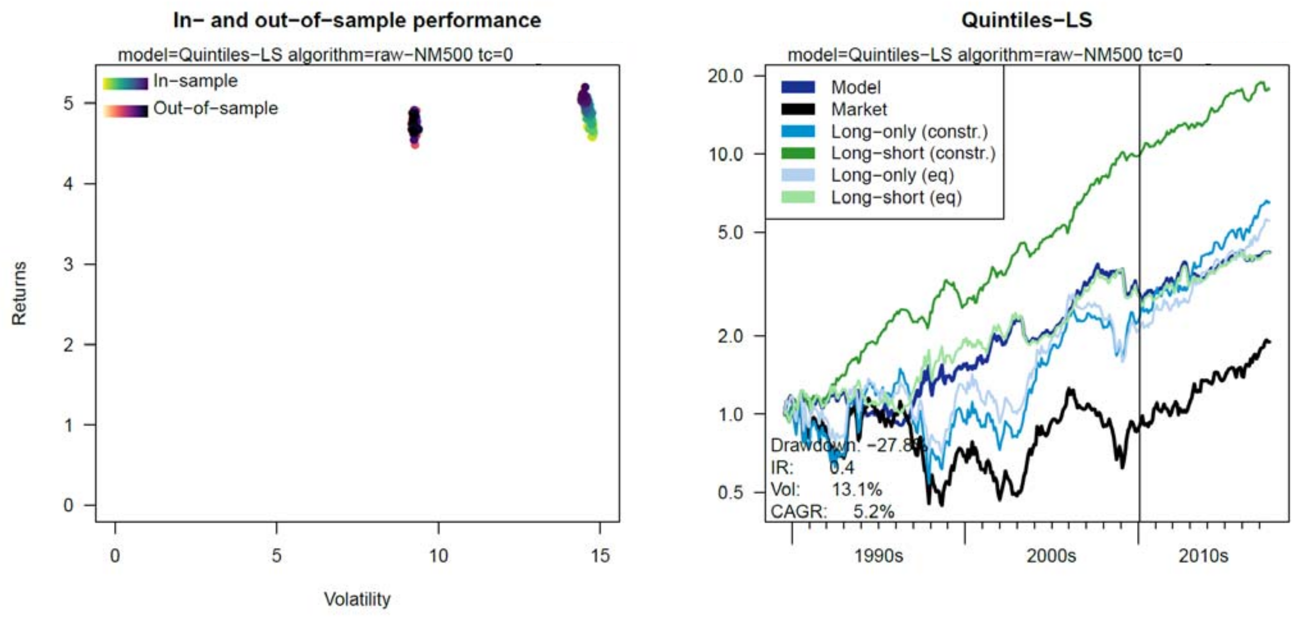
Figure 5: Random models. We consider 500 random models, i.e., linear combinations of the investment signals with random coefficients (but constrained signs) and pick the best in-sample of those 500. The out-of-sample performance is not stellar, but it is positive. The crosses indicate the in- and out-of-sample performance of the best in-sample strategy.





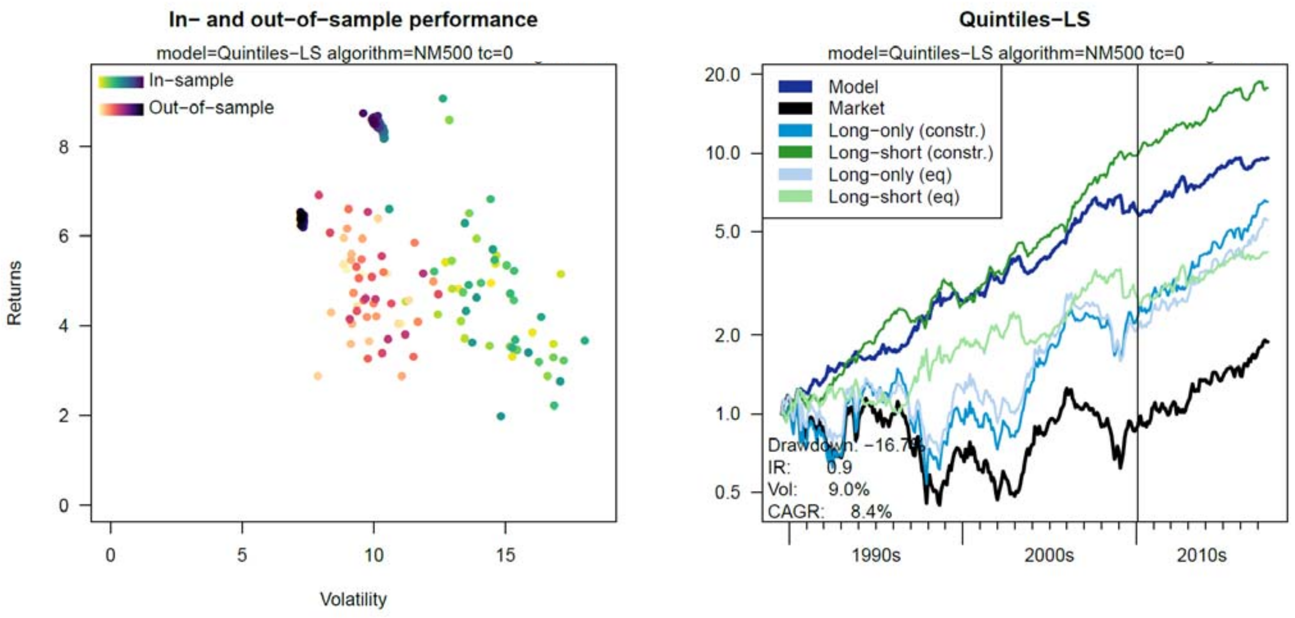
Source: Factset, S&P, Thomson Reuters, IHS Markit, Deutsche Bank Quantitative Strategy Research

Figure 6: Traditional optimization algorithms (500 steps). From a random starting point, we follow the Nelder-Mead algorithm for 500 steps. The algorithm has not explored as wide a variety of strategies as the random search, and has hardly moved from its starting point. We need to better choose the starting point and let the algorithm run longer.



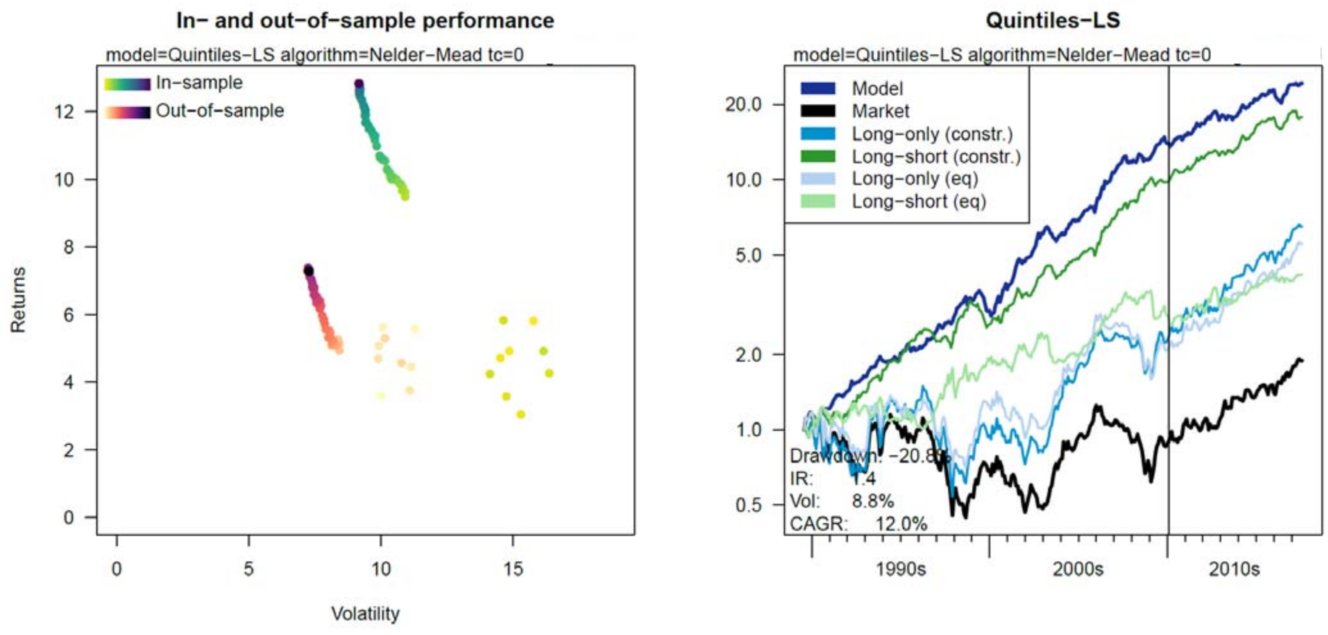
Source: Factset, S&P, Thomson Reuters, IHS Markit, Deutsche Bank Quantitative Strategy Research

Figure 7: Traditional optimization algorithms (500 steps, starting at the best point among 500 random models). To quickly explore a large variety of strategies, we use random search to find a good starting point for the Nelder-Mead algorithm.



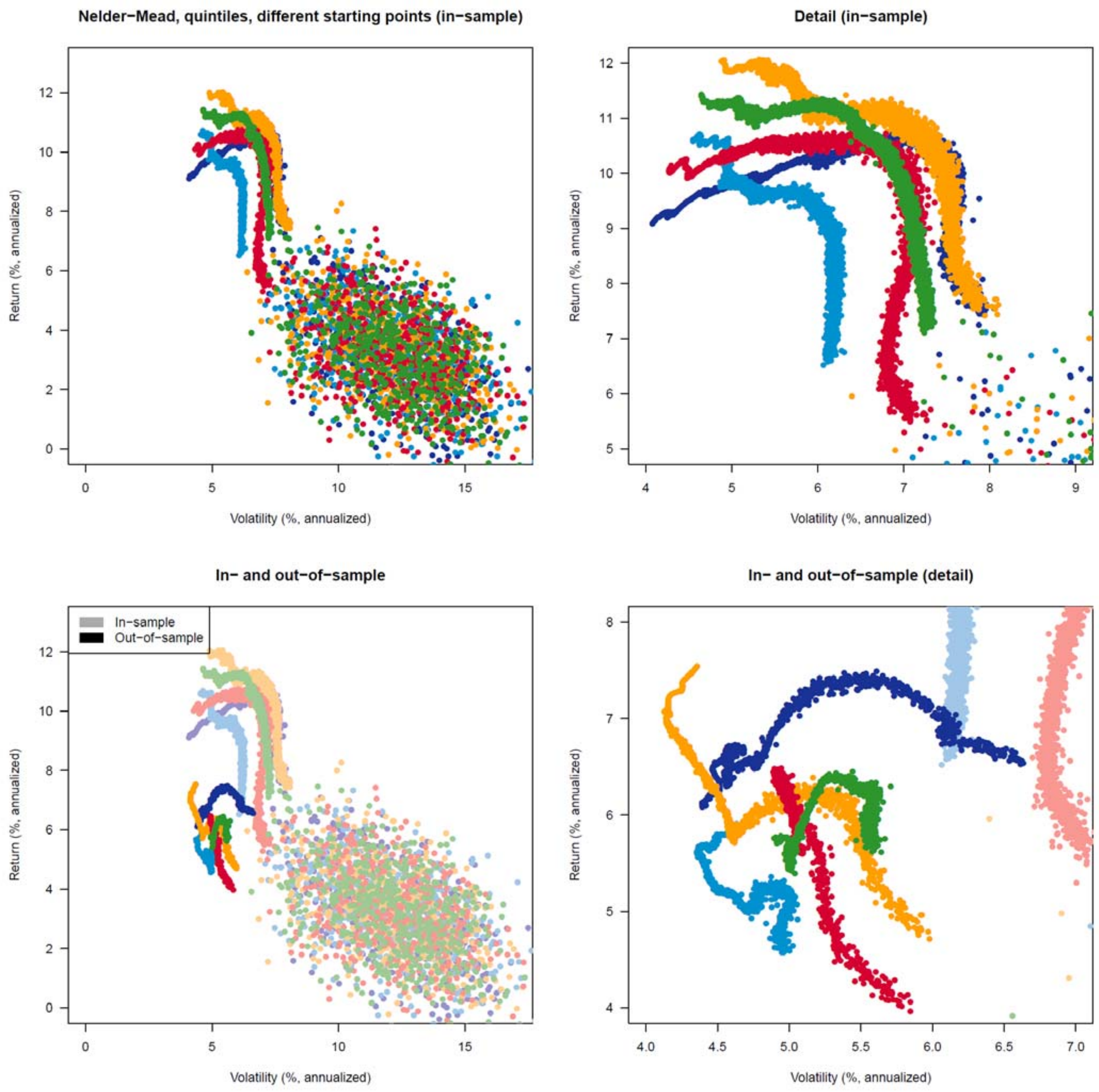
Source: Factset, S&P, Thomson Reuters, IHS Markit, Deutsche Bank Quantitative Strategy Research

Figure 8: Traditional optimization algorithms (10,000 steps). Using a good starting point (the best of 500 random models) and letting the algorithm run longer gives performance comparable to that of the constrained regression.



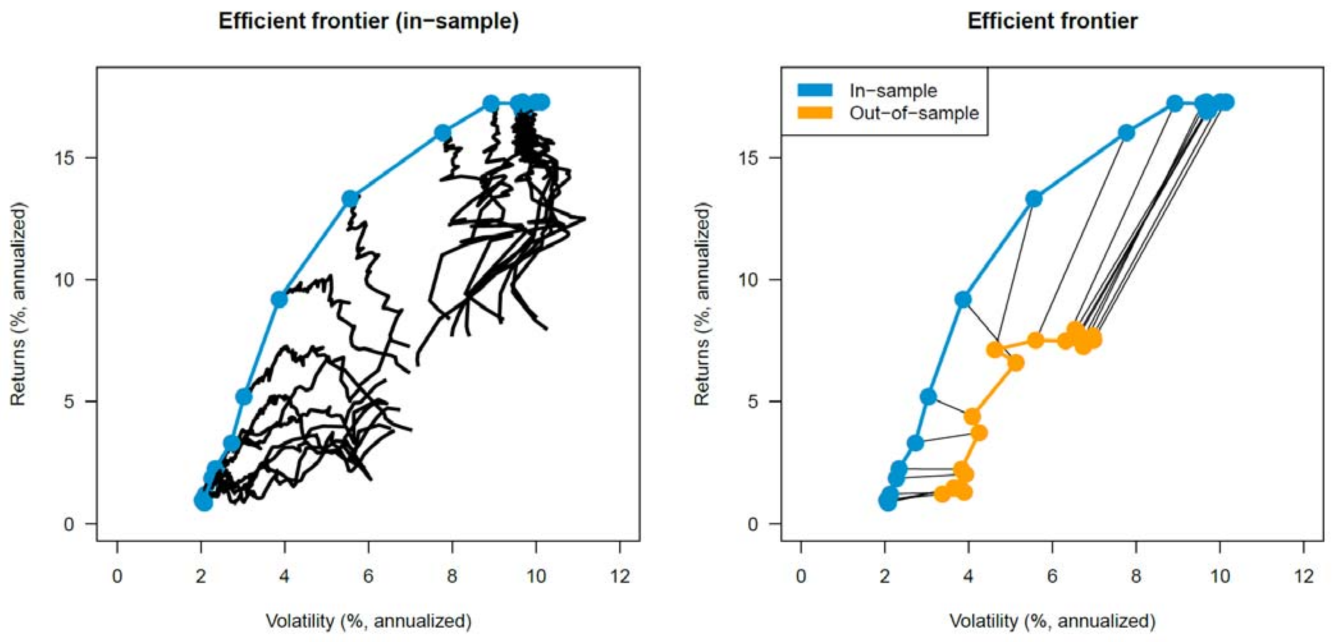
Source: Factset, S&P, Thomson Reuters, IHS Markit, Deutsche Bank Quantitative Strategy Research

Figure 9: Different starting points (Nelder-Mead). As before, we use a random search for 500 steps and use the best (in-sample) model as a starting point for the Nelder-Mead algorithm, which we let run for 10,000 steps. We did that 5 times (in 5 different colours). The cloud of random points corresponds to the initial random search, and the snake-like trajectories to the Nelder-Mean algorithm. The final strategies give different in-sample risk-return compromises with comparable information ratios. The out-of-sample performance shows more differences but each run shows a steady improvement in either return or volatility, and sometimes both.



Source: Factset, S&P, Thomson Reuters, IHS Markit, Deutsche Bank Quantitative Strategy Research

Figure 10: Efficient frontier (linear model). Instead of maximizing the information ratio IR=μ / σ we have maximized $\mu - \lambda \sigma^2$, for several values of λ. The black lines on the left are the steps in the Nelder-Mead algorithm, and the blue dots the performance of the final strategy. On the right, we show the corresponding out-of-sample points: as expected, the performance drops a lot – high-return strategies see their return decrease, while low-volatility strategies see their volatility increase.



Source: Factset, S&P, Thomson Reuters, IHS Markit, Deutsche Bank Quantitative Strategy Research

Performance

The best model, among our formulaic portfolios, is linear.

The following table summarizes the performance of the various models examined. The linear model performs better than the others.

Many of the functions popular with neural networks give very poor results.

- The *restricted linear unit* (ReLU), $x \mapsto x_+ = \mathrm{Max}(x, 0)$, leads to "dead neurons": once the output is zero, the gradient is also zero, and the weight can no longer change. Gradient-free methods relying on local changes to the candidate solution are also affected. This is not a big problem with neural networks, because they are very redundant but, with the models we consider, it means that once a factor has dropped off the model, it can never come back.

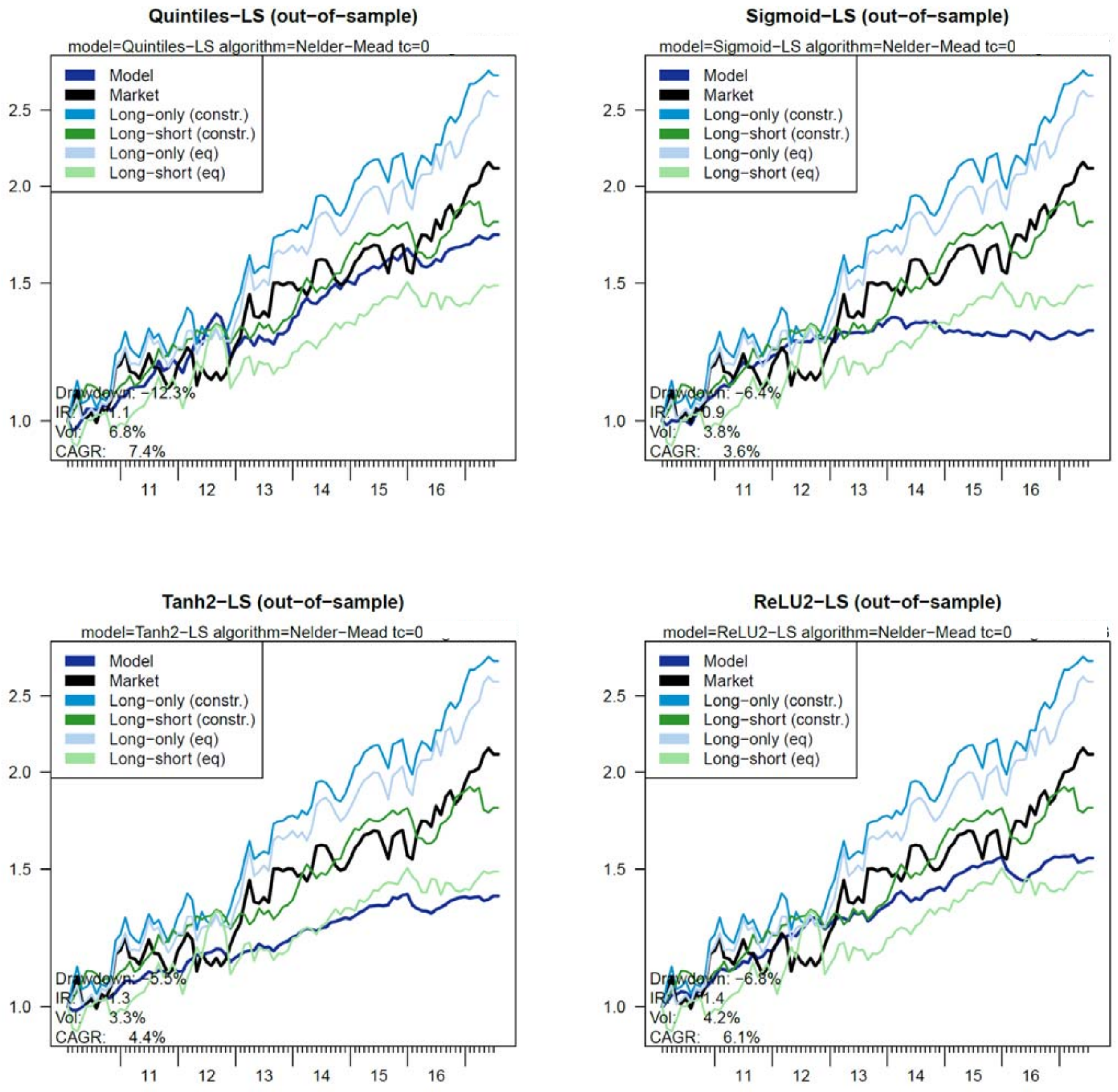
  The *softplus* function alleviates the problem, to some extent.
- Similarly, the *sigmoid* and *hyperbolic tangent* functions can lead to "saturated neurons": once the output is sufficiently close to the maximum or minimum (±1 for the hyperbolic tangent, 0 and 1 for the sigmoid), the gradient is zero, and the weight can no longer change. The problem is worse for the sigmoid.
- With the *softmax*, the presence of the exponential leads to very concentrated and, therefore, risky portfolios.

Figure 11: Performance of the various strategies tested, on the period 2010-present. The linear model performs best.

	Long-only			Long-short		
	μ	σ	IR	μ	σ	IR
Equal-weighted	14%	13%	1.1	6%	10%	0.5
Constrained	14%	13%	1.2	8%	7%	1.1
Quintiles	14%	12%	1.1	7%	8%	1.0
Quintiles 2				7%	8%	1.0
Linear	**16%**	**13%**	**1.2**	**6%**	**5%**	**1.4**
Tanh	14%	13%	1.2	4%	3%	1.3
Sigmoid	15%	13%	1.1	3%	4%	0.9
ReLU	11%	25%	0.4	6%	4%	1.4
Softplus	8%	9%	0.9	3%	3%	1.1
Softmax	7%	28%	0.2	-16%	30%	<0

Source: Factset, S&P, Thomson Reuters, IHS Markit, Deutsche Bank Quantitative Strategy Research

Figure 12: Performance of some of the long-short strategies tested (dark blue) compared with the reference strategy (dark green). The volatility of the optimized strategies tends to be lower than that of the reference



Source: Factset, S&P, Thomson Reuters, IHS Markit, Deutsche Bank Quantitative Strategy Research

Optimization algorithms

In this section, we review the main optimization algorithms – derivative-free (e.g., Nelder-Mead), gradient-based (gradient descent), Hessian-based (Newton) and stochastic (differential evolution) – and we compare them on our portfolio construction problem:

- They give very similar results but visit different regions of the optimization landscape;
- Some require a very large number of iterations: gradient- or Hessian-based algorithms, which can exploit the smoothness of the objective function, tend to be faster.

Technical interlude

Optimization algorithms can be classified from the amount of information about the objective function they use. For instance:

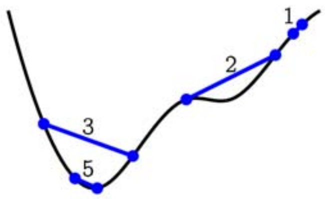
Some optimization algorithms make no assumption on the objective function, while others assume it is differentiable and use its derivatives.

- *Derivative-free methods* use the function itself, often by estimating it at a point and at nearby points, to infer the direction of in which a local extremum may lie. Those methods are reasonably fast, and relatively robust: for instance, they still work with non-continuous functions, such as the quintile portfolio construction.

  For example, the **Nelder-Mead** algorithm evaluates the function at n+1 points, forming a (triangular) "ball" (a *simplex*), and lets that ball "roll" downhill, inflating or deflating it to control its speed.

Derivative-free methods, such as Nelder-Mead, are robust but slow.

Figure 13: Nelder-Mead optimization in dimension 1. The algorithm moves a "simplex" (in dimension 1, it is a segment) towards the minimum, flipping, dilating or shrinking it at each step. (The steps represented here are not consecutive: only one point of the simplex changes at each iteration.)



Source: Deutsche Bank Quantitative Strategy Research

- *Gradient methods* use the first derivative of the function to estimate the direction of the nearest extremum: for instance, gradient descent moves in the direction of the gradient, $x \leftarrow x - \alpha f'(x)$, for some step size α. *Conjugate gradient* is a commonly-used variant.
- *Hessian-based methods* use the first two derivatives: approximating the objective function as a quadratic function gives an estimate, not only of the direction of the extremum, but also of its distance. While in

Derivative-based methods (gradient descent, L-BFGS), when they apply, are faster.

low dimensions it is reasonable to compute the Hessian (**Newton's method** approximates the objective function as a quadratic function and moves to its extremum: $x \leftarrow x - f''(x)^{-1} f'(x)$, in higher dimensions, it is too large to compute or even store: quasi-Newton algorithms such as BFGS, or its low-memory variant **L-BFGS**, use an approximation. The truncated Newton (or Hessian-free) method computes the inverse of the Hessian with an iterative algorithm but stops it after just one iteration.

Further algorithms use randomness in various ways:

The deep learning community has designed a very large number of gradient descent variants; Adam is currently one of the most popular.

- **Stochastic gradient descent** (SGD): if the objective function is a large sum, $f(x)=\Sigma f_i(x)$, (each term usually corresponding to an observation), the gradient of a random subset of this sum (corresponding to a mini-batch) gives an unbiased estimate of the full gradient.

  There are many improvements of stochastic gradient descent (Figure 16):

  - *Variance-reduced stochastic gradient* (SVRG, SAG, SAGA) keeps a seldom-updated estimate of the gradient, $f'(x^*)$, and adjusts it with the gradient from the current observation or mini-batch, e.g., $f'(x) \approx f'(x^*) + f_i'(x) - f_i'(x^*)$
  - *Momentum* is a gradient descent variant that does not move along the current gradient, but along a weighted sum of the previous gradients; this helps avoid plateaus.
  - *Adagrad* and *RMSprop* are variants of stochastic gradient descent, with the gradient normalized by its average norm.
  - **Adam** combines Momentum and RMSprop; it is one of the most popular stochastic gradient descent variant.

  There are other ways to add randomness to stochastic gradient descent: for instance, dropout sets half the parameters to zero, at random, when evaluating the gradient, thereby ensuring some redundancy in what the weights are capturing.

- *Simulated annealing*: move in a random direction, with a higher probability for directions that improve the objective function. This is (by far) one of the slowest optimization algorithms: it is valuable for combinatorial optimization, but not recommended for continuous variables.

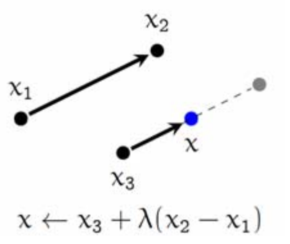
While those methods usually look for a local optimum only, population-based algorithms such as differential evolution, though slower, can find a global one even when there are many nearby local extrema.

- *Population-based* optimization algorithms do not only keep track of one candidate solution, as gradient-based methods do, but a whole "population" of them. Those methods are slow but robust: in particular, they can deal with objective functions with many local extrema.

  For instance, **differential evolution** keeps a pool of solutions, combines them three at a time to generate new candidates, keeping the best ones, to progressively improve the population.

  Particle swarm optimization uses a similar idea, modifying each candidate in the population according to its fitness and velocity and the fitness of its neighbours.

Figure 14: Differential evolution creates new solutions by combining 3 individuals from the current population, using two of them, x_1 and x_2, to define a direction, and moving the third one, x_3 in that direction, by a small amount.

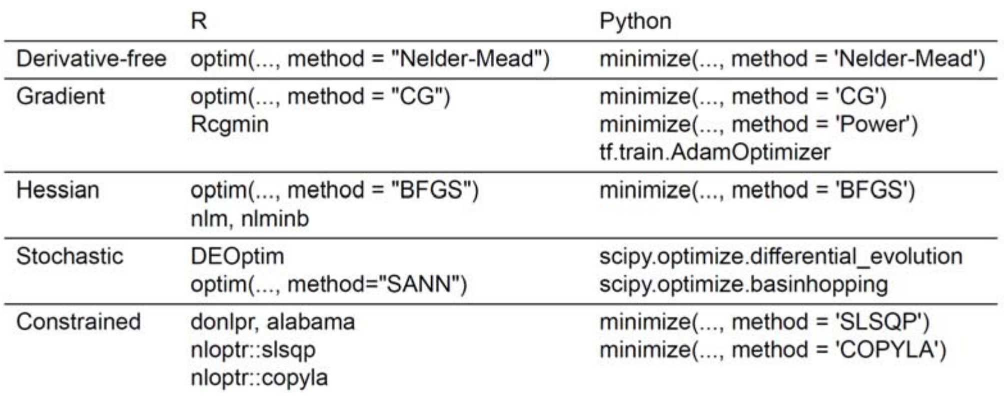


Source: Deutsche Bank Quantitative Strategy Research

- **Bayesian optimization** [6, 36] goes one step further: it keeps all the candidates it has seen and uses them to build an increasingly precise picture of the optimization landscape (e.g., with Gaussian processes) and decide where to evaluate the objective function next (e.g., maximizing the expected improvement). Bayesian optimization can deal with time-consuming and noisy functions, but does not scale well with dimension.

This list is not exhaustive and does not cover constrained optimization, cone programming [4, 26, 38, 25], robust optimization [8], combinatorial optimization or multi-objective optimization [10].

Figure 15: Some optimization algorithms available in R and Python



	R	Python
Derivative-free	optim(..., method = "Nelder-Mead")	minimize(..., method = 'Nelder-Mead')

Gradient	optim(..., method = "CG")
	Rcgmin
		minimize(..., method = 'CG')
		minimize(..., method = 'Power')
		tf.train.AdamOptimizer

Hessian	optim(..., method = "BFGS") nlm, nlminb	minimize(..., method = 'BFGS')
Stochastic	DEOptim optim(..., method="SANN")	scipy.optimize.differential_evolution scipy.optimize.basinhopping

Constrained	donlpr, alabama
	nloptr::slsqp
	nloptr::copyla
		minimize(..., method = 'SLSQP')
		minimize(..., method = 'COPYLA')

Source: Deutsche Bank Quantitative Strategy Research

Figure 16: Gradient descent variants. This is still an active area of research: recent developments include variance reduction, proximal methods, manifold optimization and information geometry – not all those methods have escaped academia yet: it may take a few years before they become available (and well documented) in your preferred programming language.

Gradient descent	$x \leftarrow x - \alpha \nabla f$ in $\mathbb{R}^n$
Gradient descent	$x \leftarrow \exp_x(-\alpha \nabla f)$ on a Riemannian manifold, e.g., positive definite matrices
Newton	$x \leftarrow x - \alpha (\nabla^2 f)^{-1} \nabla f$ If f is not convex, take the absolute values of the eigenvalues of the Hessian $\nabla^2 f$
Natural gradient	$x \leftarrow x - \alpha I^{-1} \nabla f$ where I is the Fisher information matrix, *i.e.*, the Riemannian metric on some statistical manifold
Stochastic gradient descent	$x \leftarrow x - \alpha \Delta x$
Momentum	$v \leftarrow \mu v - \alpha \Delta x$ $x \leftarrow x + v$
Nesterov momentum (NAG)	Evaluate the gradient after the momentum step
Adagrad	$N \leftarrow N + (\Delta x)^2$ $x \leftarrow x - \alpha \Delta x / \sqrt{N}$
RMSprop	Idem, with an exponential moving average
Adam	Momentum + RMSprop
Variance reduced SGD	$\nabla f(x) \approx \nabla f(x^*) - \nabla f_i(x^*) + \nabla f_i(x)$
Projected gradient	$x \leftarrow \mathrm{pr}_C(x - \alpha \nabla f)$ for constrained regression, $x^* = \operatorname*{Argmin}_{x \in C} f(x)$
Proximal gradient	$x \leftarrow \mathrm{prox}_h(x - \alpha \nabla f)$ for penalized optimization, $x^* = \mathrm{Argmin}(f + h)$

The proximal operator generalizes the projection:

$$\mathrm{prox}_h z = \operatorname*{Argmin}_{\theta} \tfrac{1}{2} \|z - \theta\|_2^2 + h(\theta)$$

$$\mathrm{prox}_{I_C} z = \mathrm{pr}_C$$

$$\mathrm{prox}_{\lambda \|\cdot\|_1} z = S_\lambda(z) =$$

$$= \text{element-wise soft-thresholding}$$

Source: Factset, S&P, Thomson Reuters, IHS Markit, Deutsche Bank Quantitative Strategy Research

Which optimization algorithm should we use?

Try several optimization algorithms, check that they give consistent results, and pick the fastest one – it will often be a gradient- or Hessian-based one.

Let us now examine the effect, if any, of the optimization algorithm used (Figure 18).

There is a stark contrast between single-path methods and population-based ones: the former take a relatively direct path towards a local extremum, while the latter move very slowly, as a dispersed herd, leaving no area unexplored, towards a set of global extrema (the extremum may not be unique).

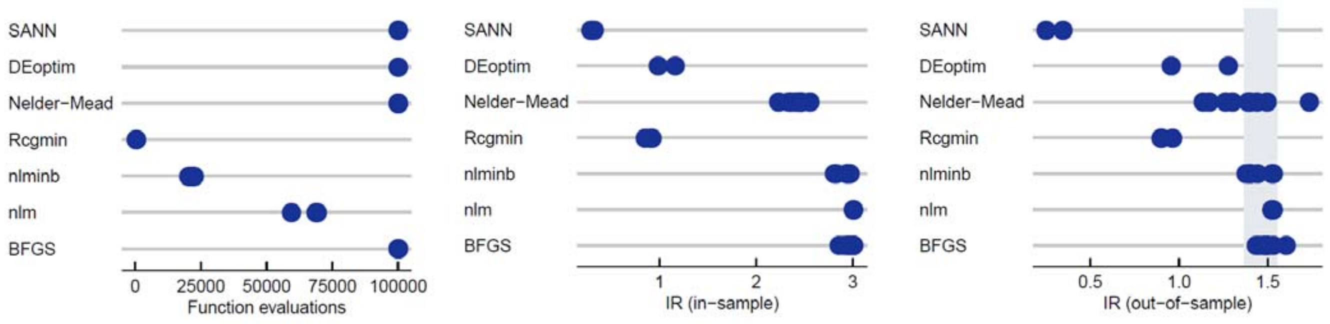
Higher-order methods (nlminb is a quasi-Newton method, newuoa is a trust-region quadratic approximation), even using approximate derivatives, seem to converge faster, at least for smooth objectives – with the quintile portfolio construction, they would correctly notice that the objective is piecewise constant and immediately stop.

Those higher-order methods do not seem very sensitive to the starting point (Figure 19).

Stochastic optimization algorithms are much slower. Differential evolution makes good progress towards the solution, but would require much more time to reach a solution of the same quality as the single-path methods.

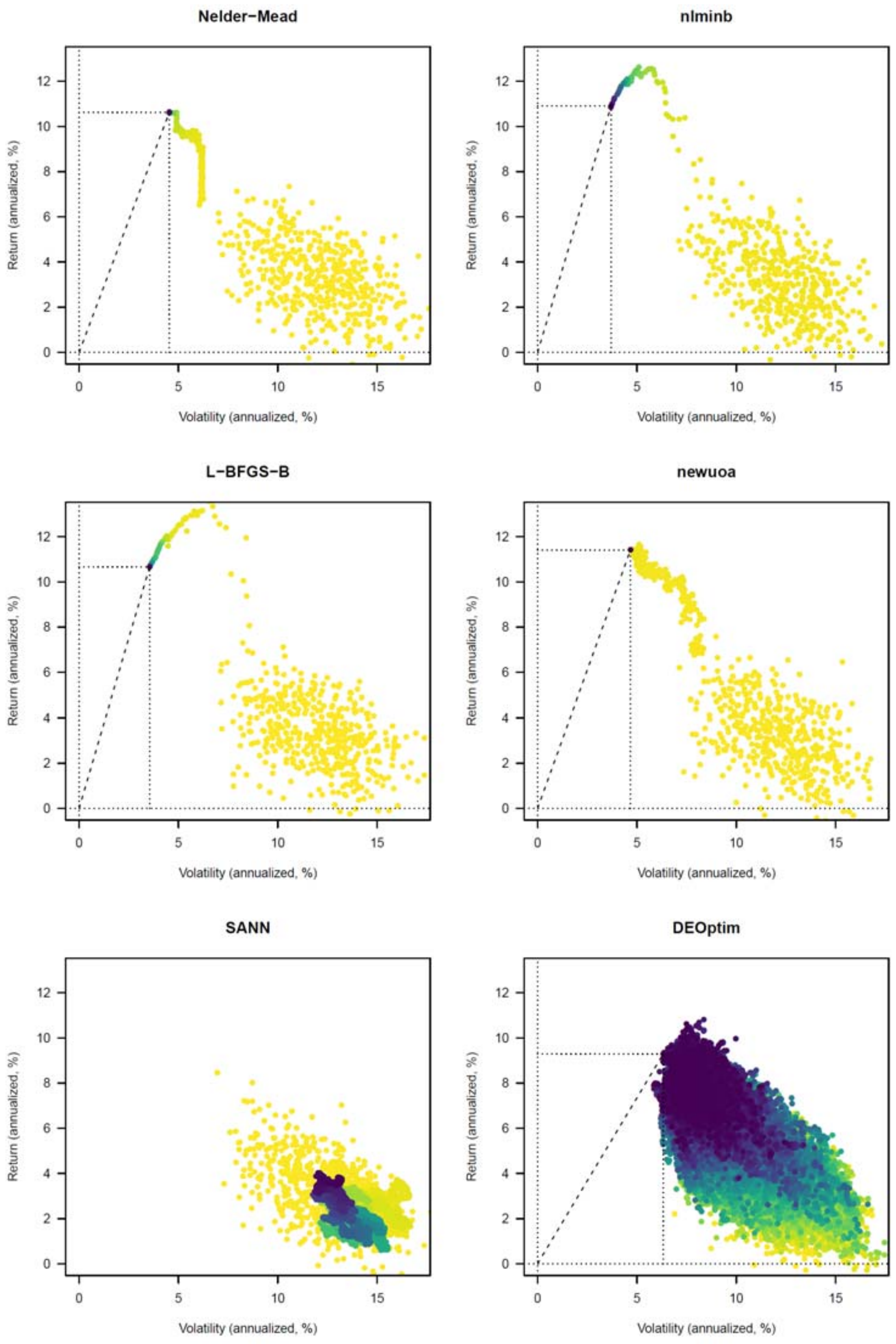
The nature of simulated annealing is clearly visible on the path: it is (literally) a random walk – since it is slightly biased towards better portfolios, it should eventually give a good portfolio, but that will take a very, very long time – much more than we are willing to wait. However, all those optimization algorithms give strategies with a similar performance (Figure 17) – simulated annealing and differential evolution require much more iterations than we have allowed them, and the conjugate gradient stalled and stopped too early.

Figure 17: In- and out-of-sample performance of the linear strategy after n function evaluations (capped at 100,000). Derivative-free methods (Nelder-Mead) need more iterations to converge: there are significant differences between runs. On this problem, the first-order method we tried (conjugate gradient) stopped too early. Second-order methods give more consistent results, and are often faster.



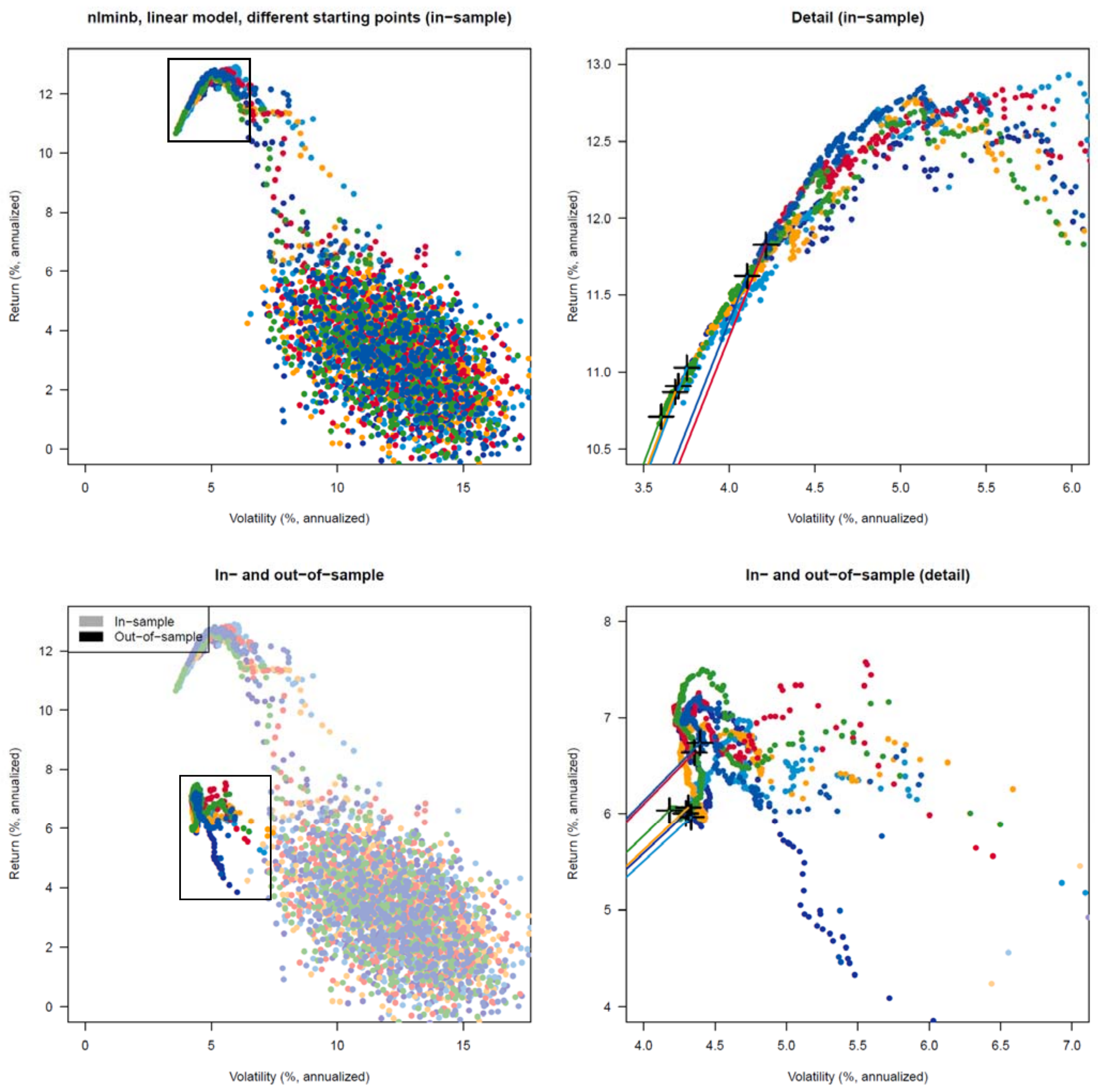
Source: Factset, S&P, Thomson Reuters, IHS Markit, Deutsche Bank Quantitative Strategy Research

Figure 18: Various optimization algorithms maximizing the (in-sample) information ratio (IR) of the linear strategy. As earlier, we first use a random search (the cloud of points in the bottom right of each plot) to find a good starting point. They end up with comparable strategies, but take different path (at different speeds): some progressively improve both returns and volatility, while others quickly find a high-returns strategy and then progressively increase the IR. Differential evolution (DEOptim) would require more iterations to give a decent result – our objective function is too "nice" for this algorithm to have an edge. As usually for continuous problems, simulated annealing (SANN) is even slower.



Source: Factset, S&P, Thomson Reuters, IHS Markit, Deutsche Bank Quantitative Strategy Research

Figure 19: Different starting points (nlminb). Some algorithms are less sensitive to the starting point: while the Nelder-Mead runs (Figure 9) had very different trajectories, the nlminb ones are much more similar. The straight lines represent the information ratio.



Source: Factset, S&P, Thomson Reuters, IHS Markit, Deutsche Bank Quantitative Strategy Research

Deep learning frameworks

In this section, we explain what "deep learning" frameworks are, and how we can use them for our portfolio construction problem, even though it does not even remotely look like a deep neural network:

A (deep) neural network is a function defined by composing (many) simple functions, such as linear transformations and sigmoids.

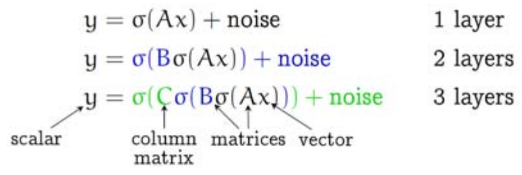
- Those frameworks allow us to describe the model and the objective function as a *computation graph*;
- To optimize the objective function, they rely on *back-propagation* or, more generally, *automatic differentiation*, to effectively (and exactly) compute gradients;
- The optimization itself is carried out by some variant of gradient descent (Figure 16)

Deep Learning

Neural networks are models obtained by composing simple functions, usually linear functions $x \mapsto Ax$ for some matrix A, and "non-linearities" (often, a sigmoid function, as in Figure 3).

Those linear and non-linear transformations can be stacked into several layers.

Figure 20: Increasingly deeper models

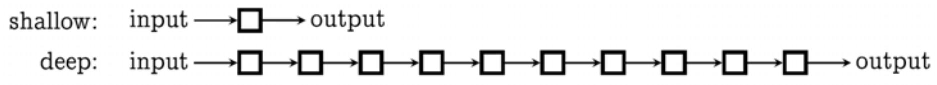


Source: Deutsche Bank Quantitative Strategy Research

Deep learning refers to machine learning on neural networks with many such layers (10 or more, as in Figure 21).

We will not use deep learning per se (indeed, our model is actually linear), but we will use tools and ideas from deep learning.

Figure 21: Shallow versus deep networks



Source: Deutsche Bank Quantitative Strategy Research

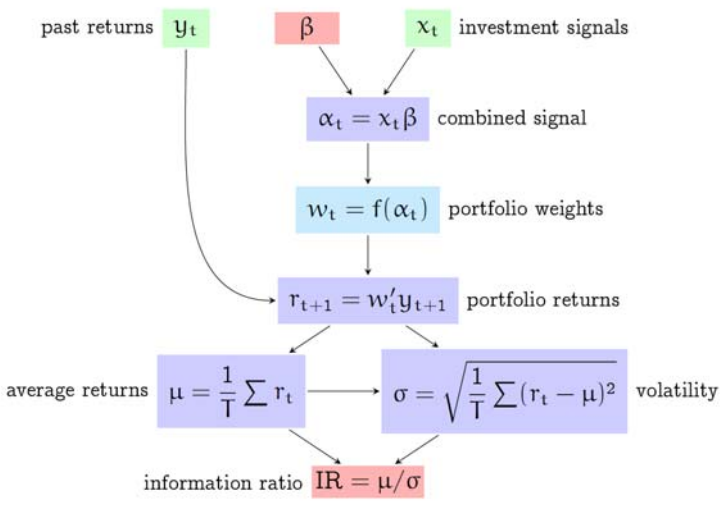
Computation graph

Deep learning frameworks represent the model to be estimated as a "computation graph".

For a few years, neural networks, and frameworks to fit and use them, have been touted as a magical solution to many problems. Let us try to use them for portfolio construction, to see how much value they actually add.

Those frameworks (we will use Tensorflow,[2] but everything we will say applies to the others as well [18,32]) let you describe your computations as a "**computation graph**", and provide a few predefined nodes such as convolutional layer, fully-connected layer, batch renormalization layer, LSTM or GRU memory cells, etc. While they are mostly used for neural networks, nothing prevents us from using them with other models: indeed, the model we will look at (Figure 22) is mostly linear.

Figure 22: A simple portfolio construction procedure.

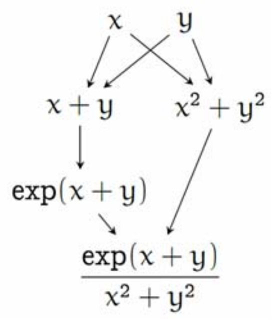

past returns y_t
β
x_t investment signals
$\alpha_t = x_t\beta$ combined signal
$w_t = f(\alpha_t)$ portfolio weights
$r_{t+1} = w_t' y_{t+1}$ portfolio returns
average returns $\mu = \frac{1}{T}\sum r_t$
$\sigma = \sqrt{\frac{1}{T}\sum(r_t-\mu)^2}$ volatility
information ratio $IR = \mu/\sigma$

Source: Deutsche Bank Quantitative Strategy Research

Figure 23 or Figure 24 show the computation graph for a simple formula, which could be part of lengthier computations. Figure 22 shows the computation graph for the portfolio construction procedure examined in this report; Figure 25 shows how we could add a portfolio optimization step.

Figure 23: A sequence of assignments and computations, and the corresponding graph

$$a \leftarrow x+y$$
$$b \leftarrow x^2+y^2$$
$$c \leftarrow \exp(a)$$
$$d \leftarrow c/b$$



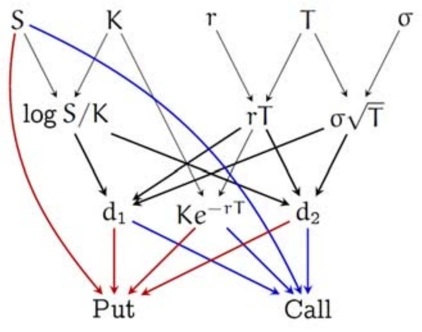
Source: Deutsche Bank Quantitative Strategy Research

[2] While we usually do our computations in R, part of this report was done with Python – Tensorflow can also be used from R, but the Python documentation is more extensive.

Representing the computations as a graph, rather than a sequence of lines of code, does not seem to add anything. However, it makes it easier to efficiently and accurately compute derivatives.

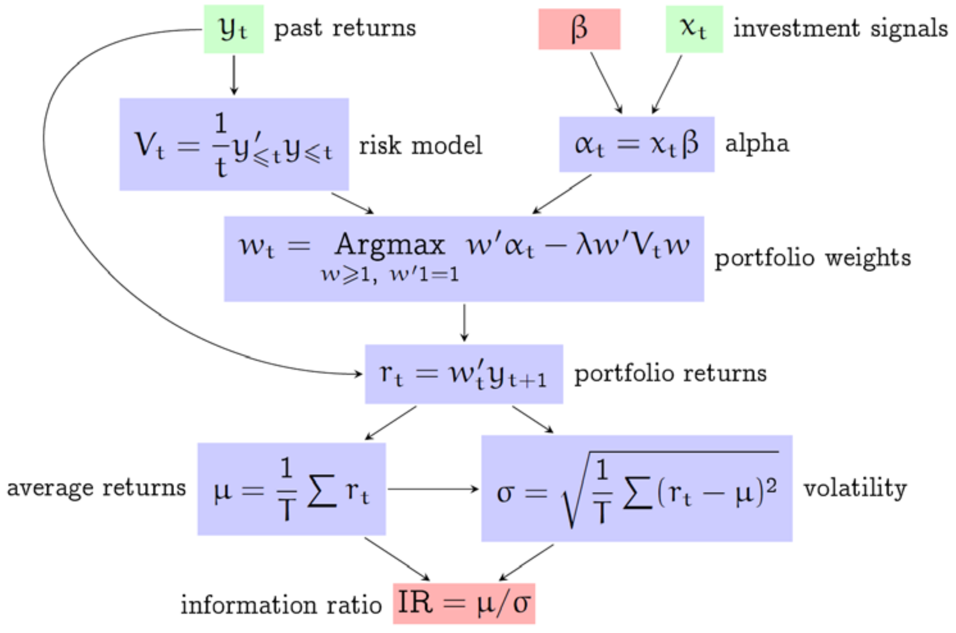
In turn, this allows us to use optimization algorithms leveraging first, or even second derivatives, which tend to be significantly faster than derivative-free algorithms such as Nelder-Mead.

Figure 24: Computation graph for a simple computation (the Black-Scholes formula). The graph may look messy but, as long as it is a directed acyclic graph (DAG), the computer can efficiently use it to compute partial derivatives (in this example, Greeks).



Source: Deutsche Bank Quantitative Strategy Research

Figure 25: Computation graph for a more realistic portfolio construction procedure, including a risk model and an optimization. Compare with Figure 4.


y_t past returns
β
x_t investment signals
$V_t = \frac{1}{t} y'_{\leqslant t} y_{\leqslant t}$ risk model
$\alpha_t = x_t \beta$ alpha
$w_t = \underset{w \geqslant 1,\ w'1=1}{\text{Argmax}}\ w'\alpha_t - \lambda w' V_t w$ portfolio weights
$r_t = w'_t y_{t+1}$ portfolio returns
average returns $\mu = \frac{1}{T} \sum r_t$
$\sigma = \sqrt{\frac{1}{T} \sum (r_t - \mu)^2}$ volatility
information ratio $IR = \mu/\sigma$

Source: Deutsche Bank Quantitative Strategy Research

Technical interlude: how to derive a function

There are many ways of computing the gradient of the objective function, needed for gradient descent optimization. Automatic differentiation is gaining popularity.

While using the gradient of the objective function helps the optimization (and the gradient is actually required by deep learning toolkits), computing this gradient can pose a few problems:

- Approximate gradient computations are imprecise and, in dimension n, requires at least n more function evaluations, which will not be used for anything else – this looks a bit wasteful;
- Deriving a closed formula for the gradient by hand is cumbersome and error-prone;
- Deriving a closed formula with a computer algebra system is also problematic: you would have to convert the code computing the objective function into a formula, which is not straightforward if the code is separated into several functions – and the problem gets even worse if the code contains control structures such as conditional or loops.

However, some efficient ways of computing derivatives or gradients have recently become more widely known. Let us review them, starting with the most classical (and most problematic) ones.

- The traditional way of computing a numeric derivative, with **finite differences**,

$$f'(x) \approx \frac{f(x+h)-f(x)}{h} \quad \text{or} \quad f'(x) \approx \frac{f(x+h)-f(x-h)}{2h},$$

  for h small ($h=10^{-6}$ is a common, relatively safe choice), is imprecise, and one has to be very careful not to choose h too small: since the numerator is the difference of two very close numbers, it has a much lower precision than each of them – for instance, if they both have 10 significant digits but share 5 of those digits, their difference only has 5 significant digits left. Fig 26 shows this phenomenon.

- Most functions we want to differentiate are not just differentiable but *analytic* (i.e., they are the sum of their Taylor series in the neighbourhood of any point): they can also be defined for **complex numbers**. We can therefore approximate the derivative as

$$f'(x) \approx \operatorname{Re} \frac{f(x+ih)-f(x)}{ih},$$

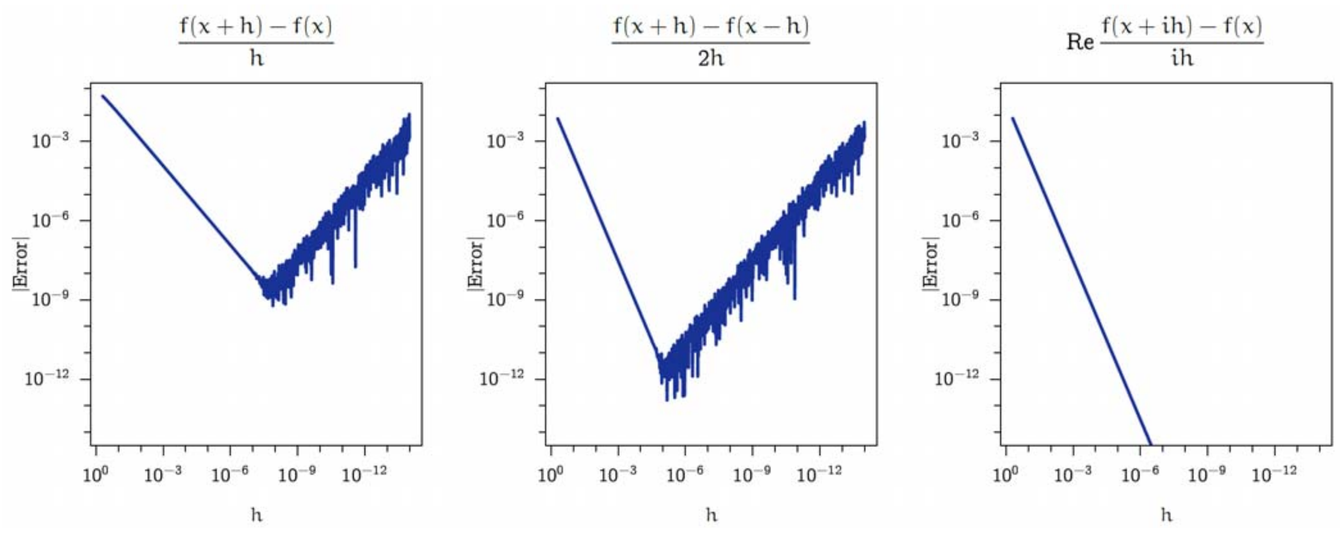
  for $x, h \in \mathbf{R}$, and h small. Contrary to the previous situation, we can let h become very small ($h = 10^{-50}$ is perfectly reasonable) without loss of precision.

- Those analytic functions can also be generalized to **dual numbers**: in the same way the complex numbers are defined from the real numbers by adding a new element, i, such that $i^2 = -1$, the dual numbers are defined from the real numbers by adding an element, ε, such that $\varepsilon^2 = 0$ (a "nilpotent infinitesimal"). The derivative can then be computed from the Taylor expansion

$$f(x + \varepsilon) = f(x) + \varepsilon f'(x).$$

Contrary to the previous methods, this is not an approximation. With programming languages with template functions (C++, Julia, etc.), this is surprisingly easy to implement.

Figure 26: Error of the finite difference approximations of f'(1) for f(x) = sin(x)/x: one-sided, central, and complex. For the one-sided and central approximations, as the step h becomes smaller (from left to right), the error progressively decreases but, at some point, rounding errors start to accumulate and the error increases. The central approximation converges faster (the error is O(h^2), while that of the 1-sided formula is O(h) – higher order schemes would converge even faster); the complex formula gives exact results (up to one or two ULP[3]) for h ≤ 10^{-8}.



Source: Deutsche Bank Quantitative Strategy Research

- It is also possible to differentiate a function defined, not by a formula, but by a computer program. It is relatively easy when the program is just a sequence of assignments, as in Figure 23: we can use the chain rule, which is simply the product of the derivatives at each step of the computation (Figure 27).

  For functions of several variables, this is actually a matrix product. We can choose the order in which we evaluate it: for instance, right-to-left, i.e., the order in which we evaluate the function itself (forward mode), or left to right (backward mode, Figure 28). The result is the same, but the size of the intermediate results may vary a lot: forward mode is preferable if the dimension of the input space is lower, and backward mode if it is larger.

  Forward and backward differentiation can be mixed together and with other ways of computing derivatives: for instance, if a step in the computations requires to solve an equation, or to minimize some quantity, the gradients can be computed with the implicit function theorem and/or the first order condition [22].

  Surprisingly, this still works for more complicated programs, with loops, conditionals and recursion. When evaluating the function, it suffices to keep track of all the assignments and computations actually performed: for instance, if the body of a loop is executed 10 times, we store the corresponding instructions 10 times; if a branch of

__Automatic differentiation__ (AD) can compute the gradient of any (differentiable) function defined, not just by a formula, but by a computer program, even if it contains control structures such as loops or conditionals.

[3] "Unit in the last place": difference between two consecutive floating point numbers.

a conditional is executed and the other ignored, we store the former and forget the latter. Since the resulting execution trace is a sequence of assignments, with no control structures left, it can be differentiated as before.

Several software packages now offer **automatic differentiation** (AD), e.g., `madness` in R [30], `autograd` in Python [22], and all the deep learning frameworks (TensorFlow, PyTorch, etc.).[4]

All those methods can be generalized to higher-order derivatives (e.g., with hypercomplex numbers, hyperdual numbers), but the size of the Hessian often makes them unwieldy.

Figure 27: The chain rule to differentiate a function. If the x_i's are vectors, the partial derivatives are matrices, and the product of matrices can be computed in any order, usually left-to-right (backward differentiation) or right-to-left (forward differentiation).

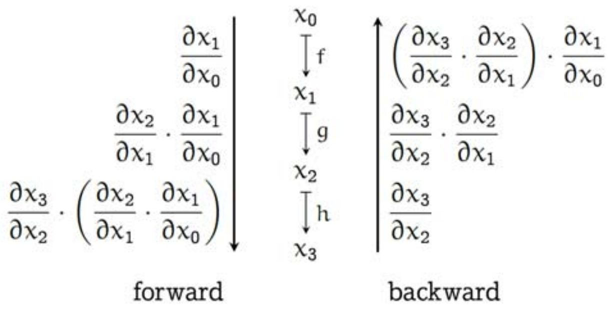
$$\begin{tikzcd} x_0 \arrow[r, mapsto, "f"] & x_1 \arrow[r, mapsto, "g"] & x_2 \arrow[r, mapsto, "h"] & x_3 \end{tikzcd}$$

$$h\big[g\big(f(x_0)\big)\big]' = h'\big[g\big(f(x_0)\big)\big] \cdot g'[f(x_0)] \cdot f'(x_0)$$

$$\frac{\partial x_3}{\partial x_0} = \frac{\partial x_3}{\partial x_2} \cdot \frac{\partial x_2}{\partial x_1} \cdot \frac{\partial x_1}{\partial x_0}$$

Source: Deutsche Bank Quantitative Strategy Research

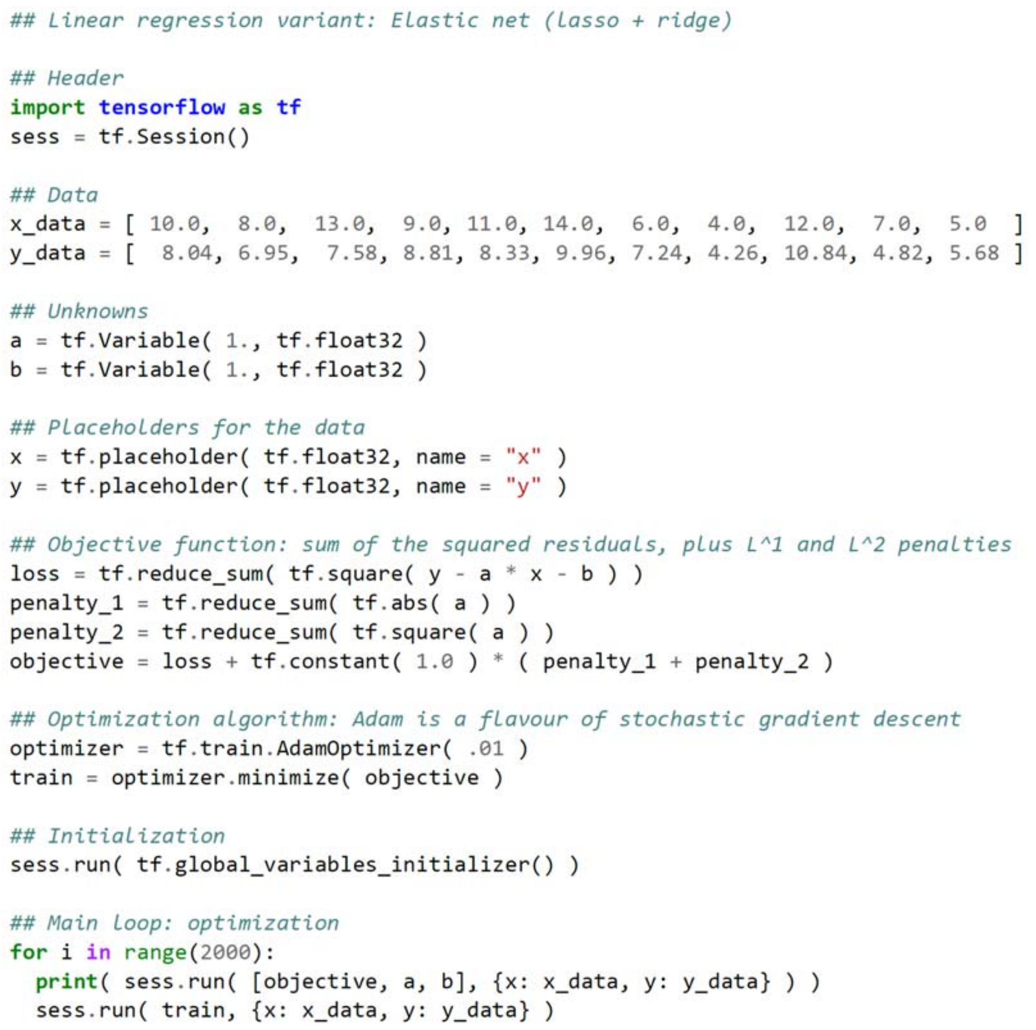
Figure 28: Forward and backward differentiation compute the same product of matrices of partial derivatives, but in a different order. The result is the same, but the size of the intermediate results, and therefore the time and space needed to do the computations, differs. For forward differentiation, the intermediate results are $\partial x_i = \partial$ input, while for backward differentiation, they are ∂ output$= \partial x_i$. Backward differentiation (the "backpropagation" of neural network lore) is more economical if the input is high-dimensional and the output low-dimensional.



Source: Deutsche Bank Quantitative Strategy Research

[4] Automatic differentiation has recently gained popularity in the quantitative finance community, in particular to compute Greeks, e.g., for assets priced with Monte Carlo simulations [21].

Figure 29: Deep learning frameworks are not limited to neural networks: here, we use TensorFlow for a penalized regression. (It is only an example, showing how straightforward it is: if you just want a penalized regression, there are better alternatives.)



```
## Linear regression variant: Elastic net (lasso + ridge)

## Header
import tensorflow as tf
sess = tf.Session()

## Data
x_data = [ 10.0,  8.0,  13.0,  9.0, 11.0, 14.0,  6.0,  4.0,  12.0,  7.0,  5.0  ]
y_data = [  8.04, 6.95,  7.58, 8.81, 8.33, 9.96, 7.24, 4.26, 10.84, 4.82, 5.68 ]

## Unknowns
a = tf.Variable( 1., tf.float32 )
b = tf.Variable( 1., tf.float32 )

## Placeholders for the data
x = tf.placeholder( tf.float32, name = "x" )
y = tf.placeholder( tf.float32, name = "y" )

## Objective function: sum of the squared residuals, plus L^1 and L^2 penalties
loss = tf.reduce_sum( tf.square( y - a * x - b ) )
penalty_1 = tf.reduce_sum( tf.abs( a ) )
penalty_2 = tf.reduce_sum( tf.square( a ) )
objective = loss + tf.constant( 1.0 ) * ( penalty_1 + penalty_2 )

## Optimization algorithm: Adam is a flavour of stochastic gradient descent
optimizer = tf.train.AdamOptimizer( .01 )
train = optimizer.minimize( objective )

## Initialization
sess.run( tf.global_variables_initializer() )

## Main loop: optimization
for i in range(2000):
  print( sess.run( [objective, a, b], {x: x_data, y: y_data} ) )
  sess.run( train, {x: x_data, y: y_data} )
```

Source: Deutsche Bank Quantitative Strategy Research

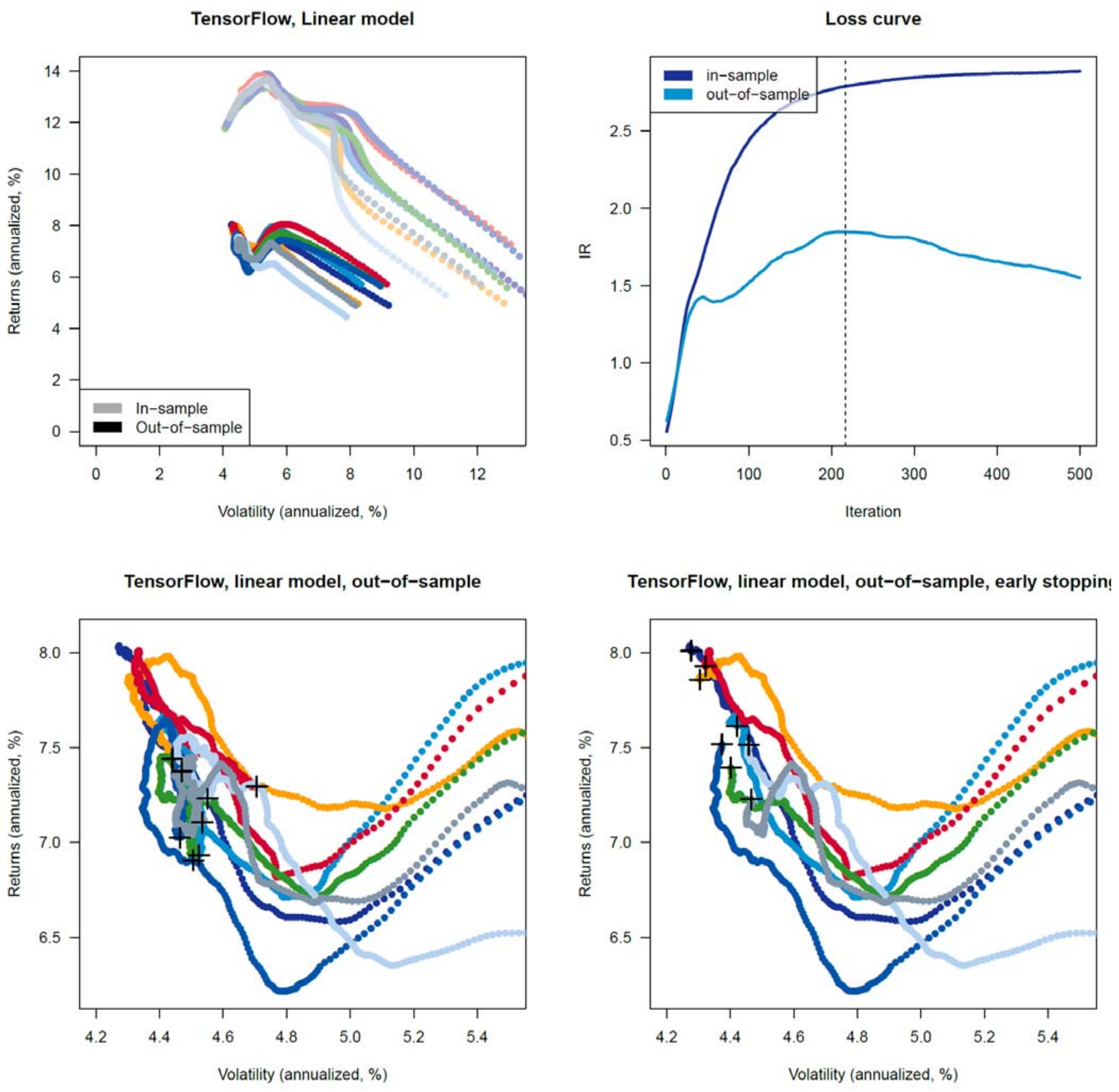
Gradient descent

Gradient descent often works best with a few "tricks" (regularization) from the deep learning world, such as early stopping.

Figure 30 shows the results of the portfolio construction procedure, optimized by gradient descent.

- The algorithm converges after 500 steps, versus 10,000 for Nelder-Mead; the steps are more time-consuming, though.
- Contrary to traditional optimization algorithms, the starting point does not seem to matter much: we started the optimization at a random portfolio, not the best among 500 random portfolios as before.
- While the in-sample performance steadily increases, the out-of-sample performance reaches a peak after approximately 200 steps – **early stopping** will improve performance.

Figure 30: Risk-return profile of strategies estimated by gradient descent. The trajectory of the optimization algorithm is similar to that of higher-order optimization algorithms examined in the previous section. The loss curve shows that the out-of-sample performance increases, reaches a peak after approximately 200 steps, and then decresases, while the in-sample performance continues to increase – the model starts to overfit the data. This is confirmed by the bottom two plots, in which the crosses indicate the final models: letting the algorithm run for too long deteriorates both returns and volatility.



Source: Factset, S&P, Thomson Reuters, IHS Markit, Deutsche Bank Quantitative Strategy Research

Regularization

In this last section, we review several ways of regularizing our optimization problem:

- The usual L^1 and L^2 penalties do not work well here: using them on the model parameters has almost no effect, because they are already constrained, and using them on the portfolio weights, because the weights are constrained to sum up to 1, can have a counter-intuitive effect;
- Business-motivated penalties, such as transaction costs, can improve the strategy, in particular by reducing the volatility;
- Deep learning regularization techniques, such as mini-batches (using only some of the observations at each step), dropout (using only some of the variables at each step), early stopping and ensembling also give promising results.

We then build a final model, by combining those ideas.

Penalties

Blindly adding penalties, because they are rumoured to be beneficial, does not always have the expected effect, in particular if the model is constrained.

It is customary to add various types of penalties to the objective function, to regularize the model (especially if there are a lot of parameters – it is not vital here) or to twist it in some direction. For instance, we could add:

- An L^1 penalty on the model parameters to reduce the number of variables used – this is not needed here, because we already have a sign constraint on them;
- An L^2 penalty on the model parameters to reduce their amplitude – ditto;
- An L^1 penalty on the stock weights, in the hope this would create a sparse portfolio, as L^1 penalties often do – this would not work, because the weights are derived values, not parameters and the model does not allow sparse portfolios;
- An L^2 penalty, on the stock weights, in the hope this would create a more concentrated portfolio – but because of the constraint that the weights be positive and sum to one, the squared L^2 norm is the Herfindahl index (a measure of concentration), and penalizing on it would actually have the opposite effect, reducing concentration and shrinking the portfolio towards the equal-weighted portfolio;
- A “negative L^2 penalty”, in the hope this would create a more concentrated portfolio.

The effect of those penalties was either imperceptible or disastrous.

Interpretable penalties can move the portfolio in some direction, e.g., lower risk, lower turnover, closer to risk parity, etc.

Interpretable penalties have a better chance of giving favourable results: one could try, for instance, to add a penalty for the difference between the risk contributions, to move the portfolio closer to a risk parity portfolio.

Technical interlude: mini-batches and dropout

Minibatches and dropout, i.e., only using part of the observations (stocks), and part of the variables (investment signals) at each step, can limit overfitting.

The computations we have just presented were using gradient descent, while "deep learning" is all about *stochastic* gradient descent.

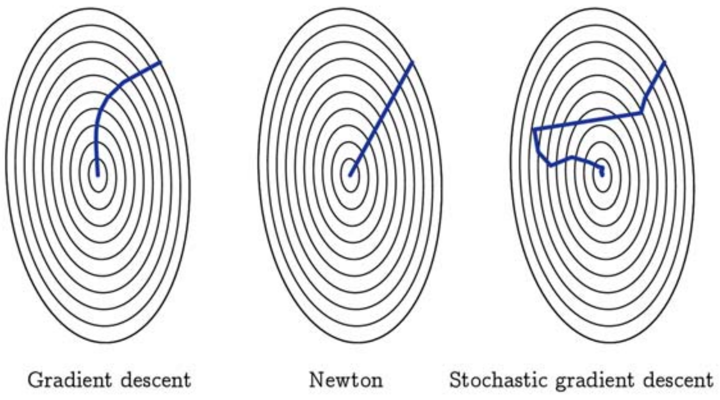
When minimizing[5] a function defined as a sum over the observations in the training set,

$$\mathrm{Loss}(\theta) = \sum_{i \in \mathrm{Training}} \mathrm{Loss}(\theta, i)$$

we can compute the gradient using a subset of the training data (a **mini-batch**) – a different subset each time.

This adds noise and may seem to slow down convergence, but it allows the algorithm to more easily escape plateaus of the loss function and reduces the influence of outliers.

Figure 31: Gradient descent variants.
The ellipses are the level curves of the function to minimize (here, a quadratic function).
Gradient descent moves by small steps in the direction of the gradient, i.e., orthogonally to the ellipses – the path is not straight.
Newton's method approximates the function to minimize with a quadratic function and directly jumps to its minimum.



Source: Deutsche Bank Quantitative Strategy Research

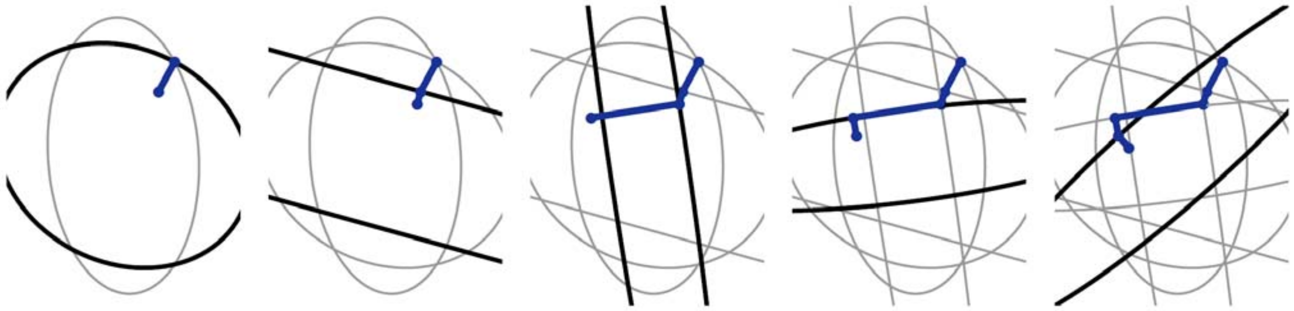
Even though our loss function is not additive, we can use the same idea and compute the information ratio of the strategy built on a random subset of stocks, different at each step in the optimization.

We could also have used random subsets of investment signals (**dropout**).

The idea of using a different, random subset ot the data, either some of the observations (mini-batch) or some of the variables (dropout), at each step of an iterative algorithm is not new: it is the main reason algorithms such as random forests or xgboost perform so well.

[5] For minimization problems, the objective function is often called "loss function". For instance, the loss function of the ordinary least squares problem is the sum of the squared residuals.

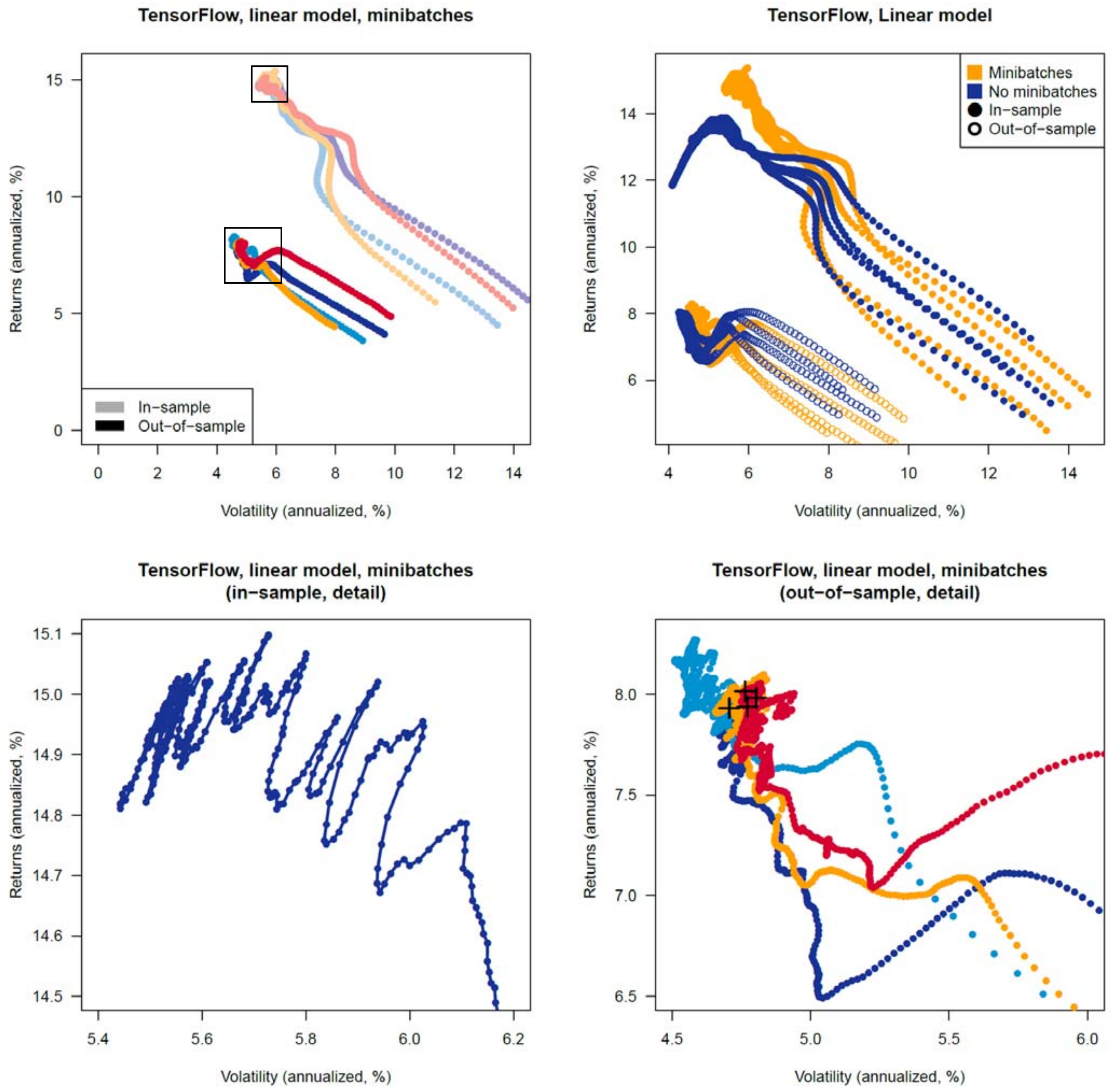
Figure 32: Stochastic gradient descent (SGD) decomposes the objective function as a sum, $\text{Loss}(\theta) = \Sigma_i \text{Loss}_i(\theta)$, and moves in the direction of a different gradient ∇Loss_i at each iteration – on average, the move is in the correct direction. The level set of the current term is in bold, that of the sum and those of the previous terms in grey.



Source: Deutsche Bank Quantitative Strategy Research

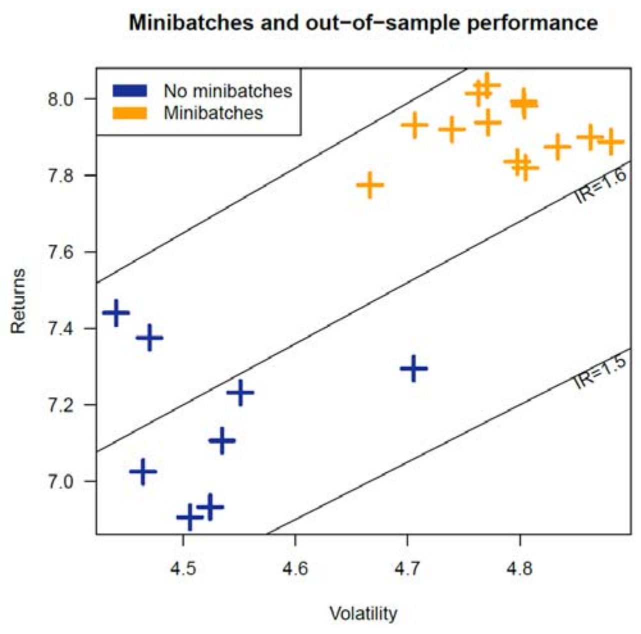
Figure 33 shows the performance of the strategies, in- and out-of-sample, as the optimization progresses: while without mini-batches the reduction in volatility is accompanied by a significant reduction in returns, for a marginal improvement in the information ratio (IR), with mini-batches, the returns remain stable as the volatility decreases. Figure 34 shows that early stopping is no longer necessary: mini-batches already limit overfitting.

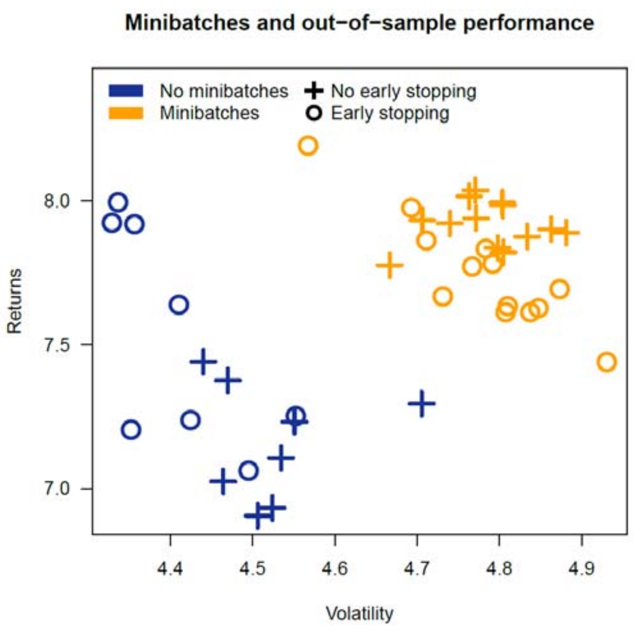
Figure 33: How mini-batches change the optimization results. While the in-sample trajectories without mini-batches first find a high-return strategy and then slowly tweak it, significantly reducing the returns for a very small gain in in-sample IR, with mini-batches, the returns remain high, and the need for early stopping is less obvious.

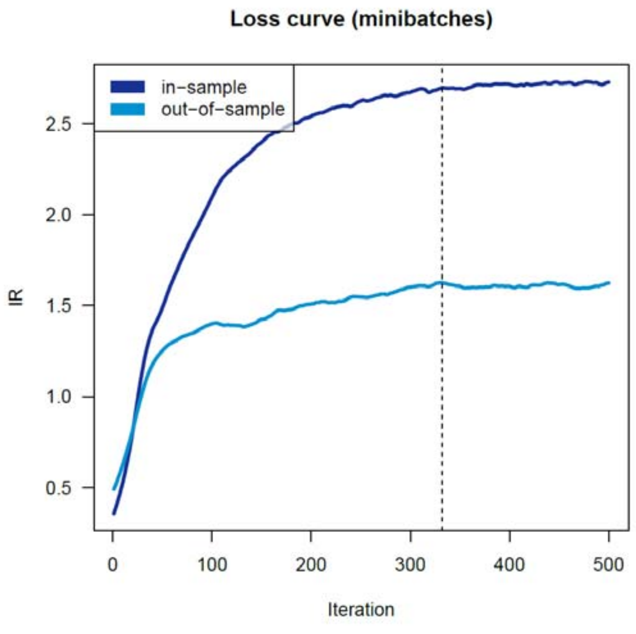


Source: Factset, S&P, Thomson Reuters, IHS Markit, Deutsche Bank Quantitative Strategy Research

Figure 34: How mini-batches change the optimization results. Mini-batches improve the out-of-sample returns and information ratio (but the volatility is slightly higher) and the performance of different runs is more consistent (top left). Early stopping improves the results without mini-batches (top right) by increasing the (out-of-sample) returns and reducing the volatility, but the only effect in the presence of mini-batches seems to make the results less consistent; this is confirmed by the loss curve (bottom).







Source: Factset, S&P, Thomson Reuters, IHS Markit, Deutsche Bank Quantitative Strategy Research

Example: accounting for transaction costs

As an application, we can devise an investment strategy that takes transaction costs into account: we simply use the same framework, but maximize the *after-transaction-cost information ratio*, for various levels of transaction costs.

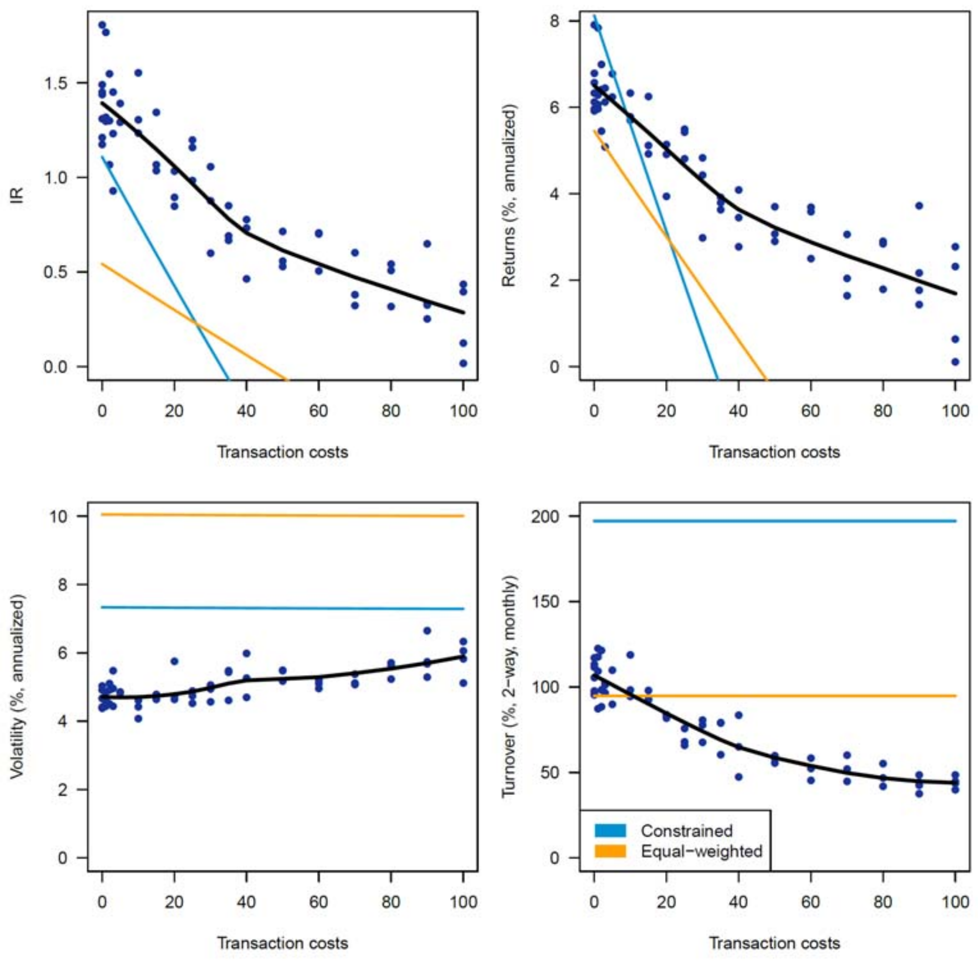
Transaction costs have a regularizing effect: they penalize high-turnover strategies.

Telling the computer the transaction costs are higher than they actually are increases the regularization.

Figure 35 shows that the best achievable information ratio decreases from 1.5 to 0.7 as the transaction costs increase from 0bp to 40bp. The volatility remains unchanged. The turnover decreases, but perhaps not as much as one could have hoped for.

Figure 42 shows how much the transaction costs matter in the selection of the strategy, as they increase.

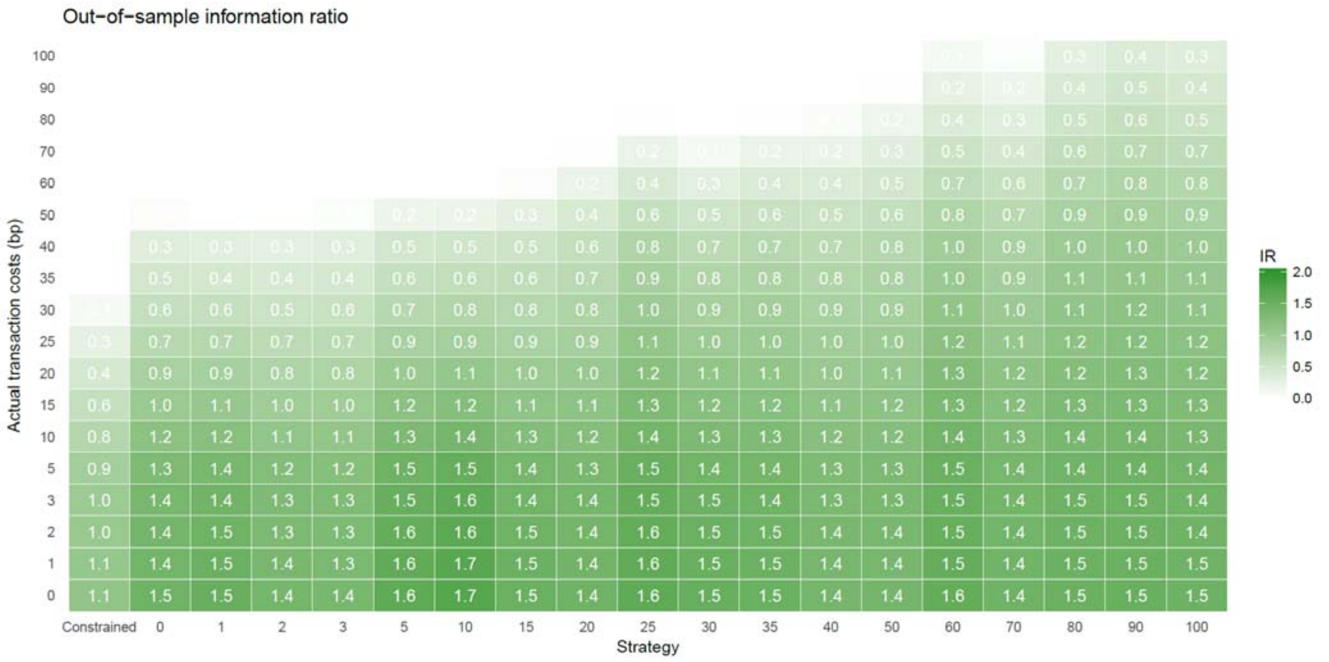
Figure 35: Performance of the long-short linear strategy for increasingly large transaction costs. For each value of the transaction costs, the optimization was run several times (blue dots); the black line is their average. Since the reference strategies do not change with transaction costs, they yield straight lines; in contrast, for the transaction-cost-aware strategies, the information ratio and the returns do not deteriorate linearly. The volatility is not significantly affected. The turnover decreases from 100% to 50%.



Source: Factset, S&P, Thomson Reuters, IHS Markit, Deutsche Bank Quantitative Strategy Research

The following figure shows that increasing the transaction costs during the optimization has a regularizing effect, and leads to better out-of-sample performance: for instance, the best out-of-sample performance if your transaction costs are 40bp is not that optimized for that level of transaction costs (it overfits the data), but those for 60bp to 100bp.

Figure 36: Out-of-sample information ratio of the various transaction-cost-aware strategies (horizontal axis) as the actual transaction costs (vertical axis) increase. Note that the transaction costs have a regularizing effect: if the actual transaction costs are 30bp (row), the best out-of-sample strategy is obtained by telling the algorithm that the transaction are much higher, between 60bp and 100bp.



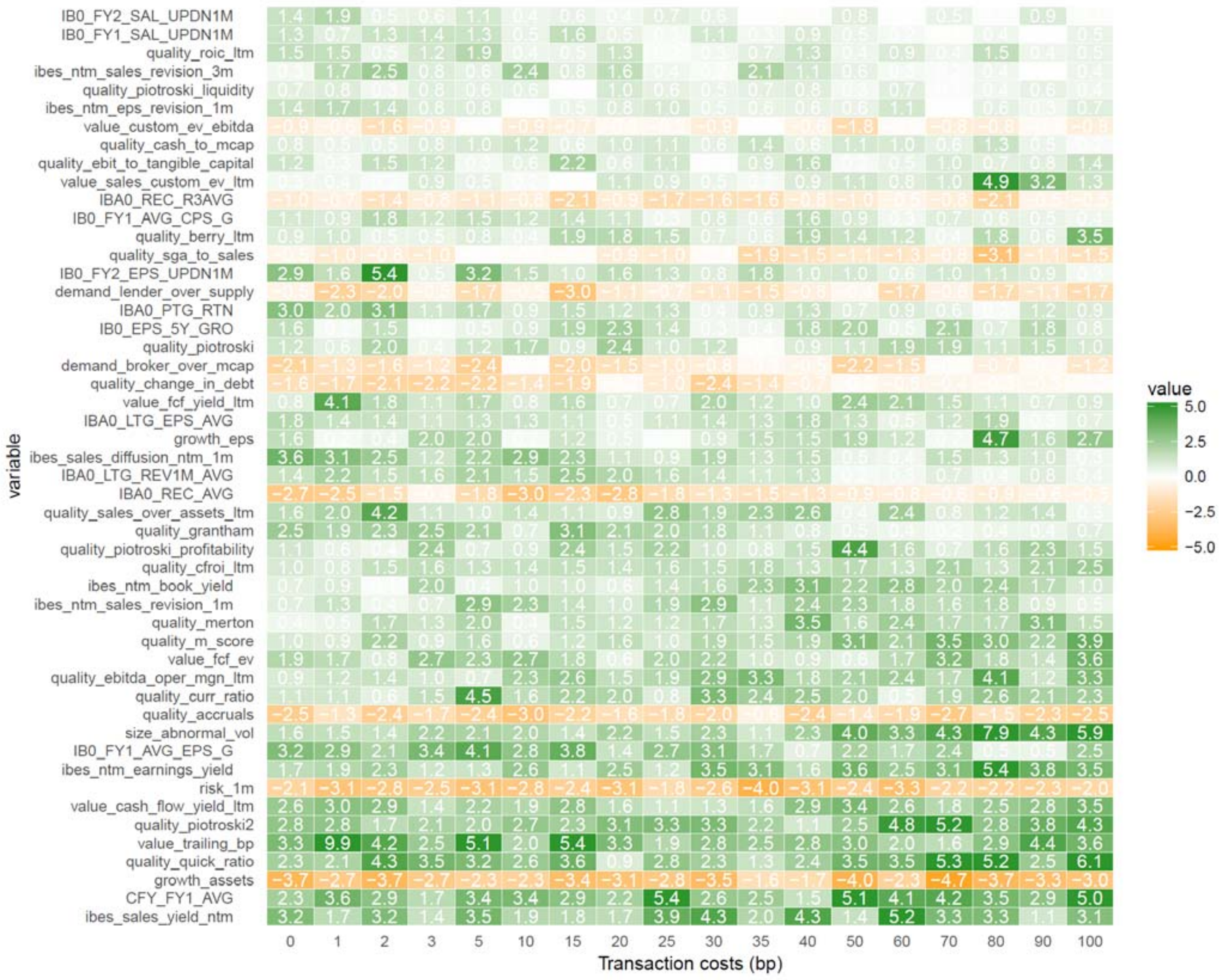
Actual transaction costs (bp)	Constrained	0	1	2	3	5	10	15	20	25	30	35	40	50	60	70	80	90	100
100															0.1	[illegible]	0.3	0.4	0.3
90															0.2	0.2	0.4	0.5	0.4
80													0.1	0.2	0.4	0.3	0.5	0.6	0.5
70										0.2	0.1	0.2	0.2	0.3	0.5	0.4	0.6	0.7	0.7
60								[illegible]	0.2	0.4	0.3	0.4	0.4	0.5	0.7	0.6	0.7	0.8	0.8
50		[illegible]			[illegible]	0.2	0.2	0.3	0.4	0.6	0.5	0.6	0.5	0.6	0.8	0.7	0.9	0.9	0.9
40		0.3	0.3	0.3	0.3	0.5	0.5	0.5	0.6	0.8	0.7	0.7	0.7	0.8	1.0	0.9	1.0	1.0	1.0
35		0.5	0.4	0.4	0.4	0.6	0.6	0.6	0.7	0.9	0.8	0.8	0.8	0.8	1.0	0.9	1.1	1.1	1.1
30	0.1	0.6	0.6	0.5	0.6	0.7	0.8	0.8	0.8	1.0	0.9	0.9	0.9	0.9	1.1	1.0	1.1	1.2	1.1
25	0.3	0.7	0.7	0.7	0.7	0.9	0.9	0.9	0.9	1.1	1.0	1.0	1.0	1.0	1.2	1.1	1.2	1.2	1.2
20	0.4	0.9	0.9	0.8	0.8	1.0	1.1	1.0	1.0	1.2	1.1	1.1	1.0	1.1	1.3	1.2	1.2	1.3	1.2
15	0.6	1.0	1.1	1.0	1.0	1.2	1.2	1.1	1.1	1.3	1.2	1.2	1.1	1.2	1.3	1.2	1.3	1.3	1.3
10	0.8	1.2	1.2	1.1	1.1	1.3	1.4	1.3	1.2	1.4	1.3	1.3	1.2	1.2	1.4	1.3	1.4	1.4	1.3
5	0.9	1.3	1.4	1.2	1.2	1.5	1.5	1.4	1.3	1.5	1.4	1.4	1.3	1.3	1.5	1.4	1.4	1.4	1.4
3	1.0	1.4	1.4	1.3	1.3	1.5	1.6	1.4	1.4	1.5	1.5	1.4	1.3	1.3	1.5	1.4	1.5	1.5	1.4
2	1.0	1.4	1.5	1.3	1.3	1.6	1.6	1.5	1.4	1.6	1.5	1.5	1.4	1.4	1.5	1.4	1.5	1.5	1.4
1	1.1	1.4	1.5	1.4	1.3	1.6	1.7	1.5	1.4	1.6	1.5	1.5	1.4	1.4	1.5	1.4	1.5	1.5	1.5
0	1.1	1.5	1.5	1.4	1.4	1.6	1.7	1.5	1.4	1.6	1.5	1.5	1.4	1.4	1.6	1.4	1.5	1.5	1.5

Source: Factset, S&P, Thomson Reuters, IHS Markit, Deutsche Bank Quantitative Strategy Research

Interpretability, capacity

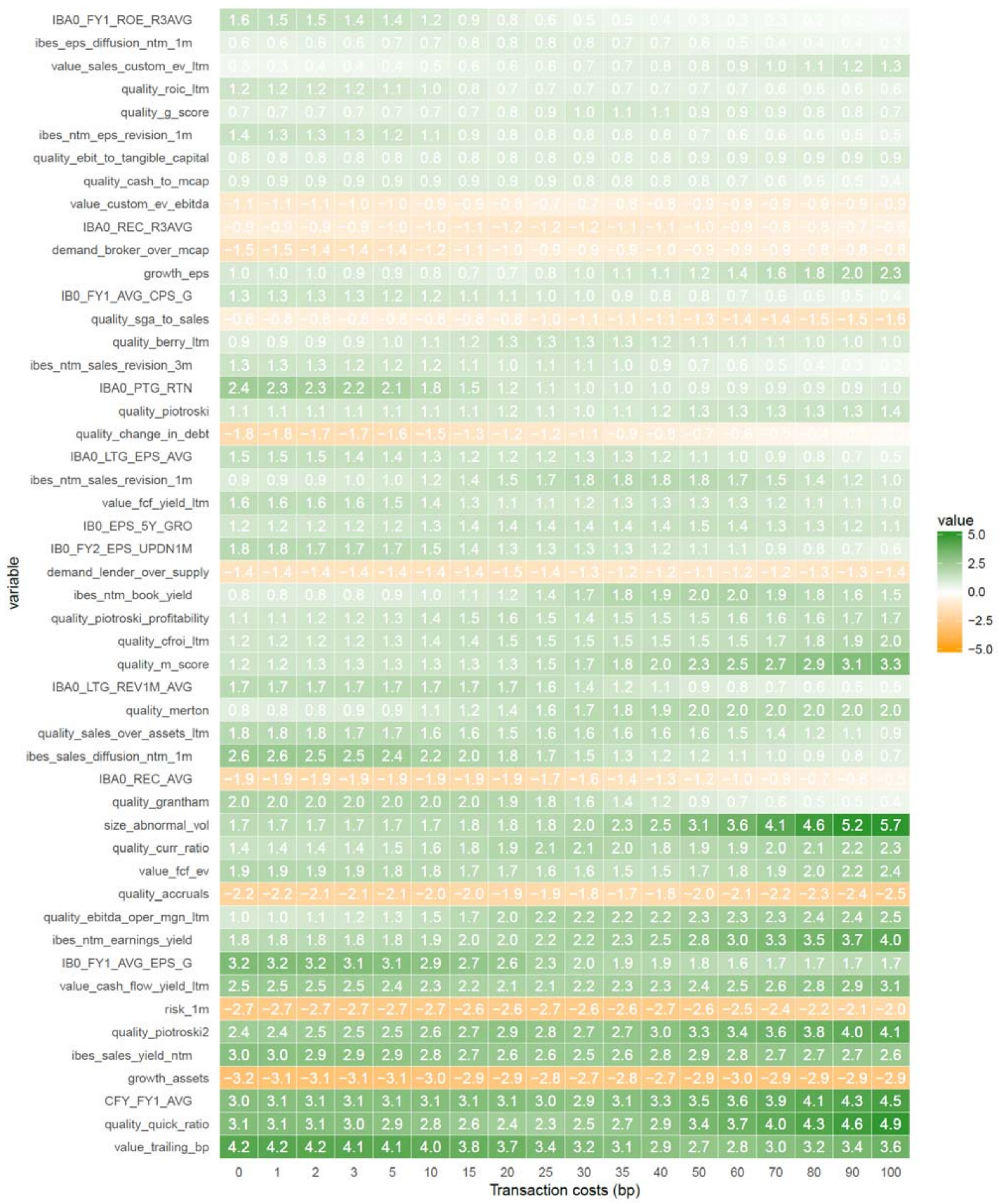
Let us now examine how similar those portfolios are, as we increase the transaction costs. Figure 37 shows the model coefficients, but they are too noisy to clearly show how the strategies change. Figure 38 shows the smoothed model coefficients: we can more clearly see which factors see their weight increase or decrease as the transaction costs increase.

Figure 37: Model coefficients (for the most important variables). While it is possible to see which factors are the most important, it is less clear how their importance changes as the transaction costs increase.



Source: Factset, S&P, Thomson Reuters, IHS Markit, Deutsche Bank Quantitative Strategy Research

Figure 38: Model coefficients, smoothed (for the most important variables). As the transaction costs increase, slow-changing factors (in particular, many quality signals) see their importance increase, while fast-changing ones see it decrease.



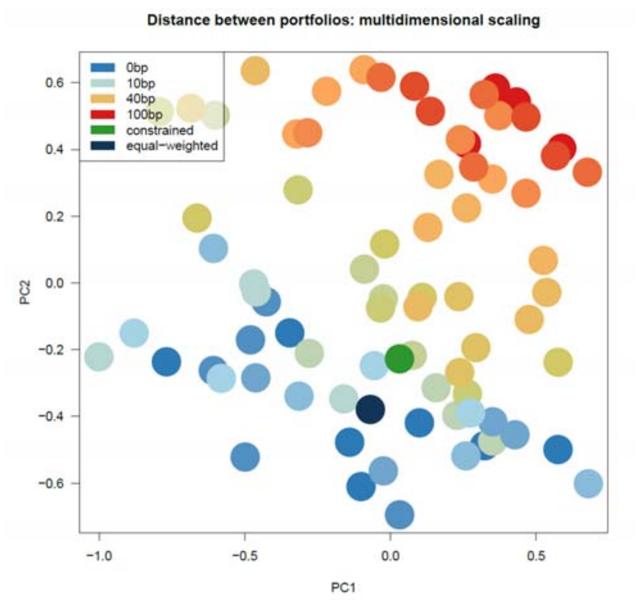
variable	0	1	2	3	5	10	15	20	25	30	35	40	50	60	70	80	90	100
IBA0_FY1_ROE_R3AVG	1.6	1.5	1.5	1.4	1.4	1.2	0.9	0.8	0.6	0.5	0.5	0.4	0.3	0.3	0.3	0.2	0.2	0.2
ibes_eps_diffusion_ntm_1m	0.6	0.6	0.6	0.6	0.7	0.7	0.8	0.8	0.8	0.8	0.7	0.7	0.6	0.5	0.4	0.4	0.4	0.3
value_sales_custom_ev_ltm	0.3	0.3	0.4	0.4	0.4	0.5	0.6	0.6	0.6	0.7	0.7	0.8	0.8	0.9	1.0	1.1	1.2	1.3
quality_roic_ltm	1.2	1.2	1.2	1.2	1.1	1.0	0.8	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.6	0.6	0.6	0.6
quality_g_score	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.8	0.9	1.0	1.1	1.1	0.9	0.9	0.9	0.8	0.8	0.7
ibes_ntm_eps_revision_1m	1.4	1.3	1.3	1.3	1.2	1.1	0.9	0.8	0.8	0.8	0.8	0.8	0.7	0.6	0.6	0.6	0.5	0.5
quality_ebit_to_tangible_capital	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.9	0.8	0.8	0.9	0.9	0.9	0.9	0.9	0.9
quality_cash_to_mcap	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.8	0.8	0.8	0.8	0.7	0.6	0.6	0.5	0.4
value_custom_ev_ebitda	−1.1	−1.1	−1.1	−1.0	−1.0	−0.9	−0.9	−0.8	−0.7	−0.7	−0.8	−0.8	−0.9	−0.9	−0.9	−0.9	−0.9	−0.9
IBA0_REC_R3AVG	−0.9	−0.9	−0.8	−0.9	−1.0	−1.0	−1.1	−1.2	−1.2	−1.2	−1.1	−1.1	−1.0	−0.9	−0.8	−0.8	−0.7	−0.6
demand_broker_over_mcap	−1.5	−1.5	−1.4	−1.4	−1.4	−1.2	−1.1	−1.0	−0.9	−0.9	−0.9	−1.0	−0.9	−0.9	−0.9	−0.8	−0.8	−0.8
growth_eps	1.0	1.0	1.0	0.9	0.9	0.8	0.7	0.7	0.8	1.0	1.1	1.1	1.2	1.4	1.6	1.8	2.0	2.3
IB0_FY1_AVG_CPS_G	1.3	1.3	1.3	1.3	1.2	1.2	1.1	1.1	1.0	1.0	0.9	0.8	0.8	0.7	0.6	0.6	0.5	0.4
quality_sga_to_sales	−0.8	−0.8	−0.8	−0.8	−0.8	−0.8	−0.8	−0.8	−1.0	−1.1	−1.1	−1.1	−1.3	−1.4	−1.4	−1.5	−1.5	−1.6
quality_berry_ltm	0.9	0.9	0.9	0.9	1.0	1.1	1.2	1.3	1.3	1.3	1.3	1.2	1.1	1.1	1.1	1.0	1.0	1.0
ibes_ntm_sales_revision_3m	1.3	1.3	1.3	1.2	1.2	1.2	1.1	1.0	1.1	1.1	1.0	0.9	0.7	0.6	0.5	0.4	0.3	0.2
IBA0_PTG_RTN	2.4	2.3	2.3	2.2	2.1	1.8	1.5	1.2	1.1	1.0	1.0	1.0	0.9	0.9	0.9	0.9	0.9	1.0
quality_piotroski	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.2	1.1	1.0	1.1	1.2	1.3	1.3	1.3	1.3	1.3	1.4
quality_change_in_debt	−1.8	−1.8	−1.7	−1.7	−1.6	−1.5	−1.3	−1.2	−1.2	−1.1	−0.9	−0.8	−0.7	−0.6	−0.5	−0.4	[illegible]	[illegible]
IBA0_LTG_EPS_AVG	1.5	1.5	1.5	1.4	1.4	1.3	1.2	1.2	1.2	1.3	1.3	1.2	1.1	1.0	0.9	0.8	0.7	0.5
ibes_ntm_sales_revision_1m	0.9	0.9	0.9	1.0	1.0	1.2	1.4	1.5	1.7	1.8	1.8	1.8	1.8	1.7	1.5	1.4	1.2	1.0
value_fcf_yield_ltm	1.6	1.6	1.6	1.6	1.5	1.4	1.3	1.1	1.1	1.2	1.3	1.3	1.3	1.3	1.2	1.1	1.1	1.0
IB0_EPS_5Y_GRO	1.2	1.2	1.2	1.2	1.2	1.3	1.4	1.4	1.4	1.4	1.4	1.4	1.5	1.4	1.3	1.3	1.2	1.1
IB0_FY2_EPS_UPDN1M	1.8	1.8	1.7	1.7	1.7	1.5	1.4	1.3	1.3	1.3	1.3	1.2	1.1	1.1	0.9	0.8	0.7	0.6
demand_lender_over_supply	−1.4	−1.4	−1.4	−1.4	−1.4	−1.4	−1.4	−1.5	−1.4	−1.3	−1.2	−1.2	−1.1	−1.2	−1.2	−1.3	−1.3	−1.4
ibes_ntm_book_yield	0.8	0.8	0.8	0.8	0.9	1.0	1.1	1.2	1.4	1.7	1.8	1.9	2.0	2.0	1.9	1.8	1.6	1.5
quality_piotroski_profitability	1.1	1.1	1.2	1.2	1.3	1.4	1.5	1.6	1.5	1.4	1.5	1.5	1.5	1.6	1.6	1.6	1.7	1.7
quality_cfroi_ltm	1.2	1.2	1.2	1.2	1.3	1.4	1.4	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.7	1.8	1.9	2.0
quality_m_score	1.2	1.2	1.3	1.3	1.3	1.3	1.3	1.3	1.5	1.7	1.8	2.0	2.3	2.5	2.7	2.9	3.1	3.3
IBA0_LTG_REV1M_AVG	1.7	1.7	1.7	1.7	1.7	1.7	1.7	1.7	1.6	1.4	1.2	1.1	0.9	0.8	0.7	0.6	0.5	0.5
quality_merton	0.8	0.8	0.8	0.9	0.9	1.1	1.2	1.4	1.6	1.7	1.8	1.9	2.0	2.0	2.0	2.0	2.0	2.0
quality_sales_over_assets_ltm	1.8	1.8	1.8	1.7	1.7	1.6	1.6	1.5	1.6	1.6	1.6	1.6	1.6	1.5	1.4	1.2	1.1	0.9
ibes_sales_diffusion_ntm_1m	2.6	2.6	2.5	2.5	2.4	2.2	2.0	1.8	1.7	1.5	1.3	1.2	1.2	1.1	1.0	0.9	0.8	0.7
IBA0_REC_AVG	−1.9	−1.9	−1.9	−1.9	−1.9	−1.9	−1.9	−1.9	−1.7	−1.6	−1.4	−1.3	−1.2	−1.0	−0.9	−0.7	−0.6	−0.5
quality_grantham	2.0	2.0	2.0	2.0	2.0	2.0	2.0	1.9	1.8	1.6	1.4	1.2	0.9	0.7	0.6	0.5	0.5	0.4
size_abnormal_vol	1.7	1.7	1.7	1.7	1.7	1.7	1.8	1.8	1.8	2.0	2.3	2.5	3.1	3.6	4.1	4.6	5.2	5.7
quality_curr_ratio	1.4	1.4	1.4	1.4	1.5	1.6	1.8	1.9	2.1	2.1	2.0	1.8	1.9	1.9	2.0	2.1	2.2	2.3
value_fcf_ev	1.9	1.9	1.9	1.9	1.8	1.8	1.7	1.7	1.6	1.6	1.5	1.5	1.7	1.8	1.9	2.0	2.2	2.4
quality_accruals	−2.2	−2.2	−2.1	−2.1	−2.1	−2.0	−2.0	−1.9	−1.9	−1.8	−1.7	−1.8	−2.0	−2.1	−2.2	−2.3	−2.4	−2.5
quality_ebitda_oper_mgn_ltm	1.0	1.0	1.1	1.2	1.3	1.5	1.7	2.0	2.2	2.2	2.2	2.2	2.3	2.3	2.3	2.4	2.4	2.5
ibes_ntm_earnings_yield	1.8	1.8	1.8	1.8	1.8	1.9	2.0	2.0	2.2	2.2	2.3	2.5	2.8	3.0	3.3	3.5	3.7	4.0
IB0_FY1_AVG_EPS_G	3.2	3.2	3.2	3.1	3.1	2.9	2.7	2.6	2.3	2.0	1.9	1.9	1.8	1.6	1.7	1.7	1.7	1.7
value_cash_flow_yield_ltm	2.5	2.5	2.5	2.5	2.4	2.3	2.2	2.1	2.1	2.2	2.3	2.3	2.4	2.5	2.6	2.8	2.9	3.1
risk_1m	−2.7	−2.7	−2.7	−2.7	−2.7	−2.7	−2.6	−2.6	−2.7	−2.6	−2.6	−2.7	−2.6	−2.5	−2.4	−2.2	−2.1	−2.0
quality_piotroski2	2.4	2.4	2.5	2.5	2.5	2.6	2.7	2.9	2.8	2.7	2.7	3.0	3.3	3.4	3.6	3.8	4.0	4.1
ibes_sales_yield_ntm	3.0	3.0	2.9	2.9	2.9	2.8	2.7	2.6	2.6	2.5	2.6	2.8	2.9	2.8	2.7	2.7	2.7	2.6
growth_assets	−3.2	−3.1	−3.1	−3.1	−3.1	−3.0	−2.9	−2.9	−2.8	−2.7	−2.8	−2.7	−2.9	−3.0	−2.9	−2.9	−2.9	−2.9
CFY_FY1_AVG	3.0	3.1	3.1	3.1	3.1	3.1	3.1	3.1	3.0	2.9	3.1	3.3	3.5	3.6	3.9	4.1	4.3	4.5
quality_quick_ratio	3.1	3.1	3.1	3.0	2.9	2.8	2.6	2.4	2.3	2.5	2.7	2.9	3.4	3.7	4.0	4.3	4.6	4.9
value_trailing_bp	4.2	4.2	4.2	4.1	4.1	4.0	3.8	3.7	3.4	3.2	3.1	2.9	2.7	2.8	3.0	3.2	3.4	3.6

Source: Factset, S&P, Thomson Reuters, IHS Markit, Deutsche Bank Quantitative Strategy Research

Figure 39 shows the distance between the portfolios obtained as the transaction costs increase:

- There is around 75% overlap between any two high-turnover portfolios;
- There is around 90% overlap between any two low-turnover portfolios;
- There is around 75% overlap between high- and low-turnover portfolios.

Figure 39: Distance between the portfolios. The Euclidean distance between the dots is comparable to the L1 distance (2-way turnover) between the portfolios. While the difference between high- and low-turnover portfolios is significant, it is small. The low-turnover portfolios are more similar to one another than the high-turnover ones.



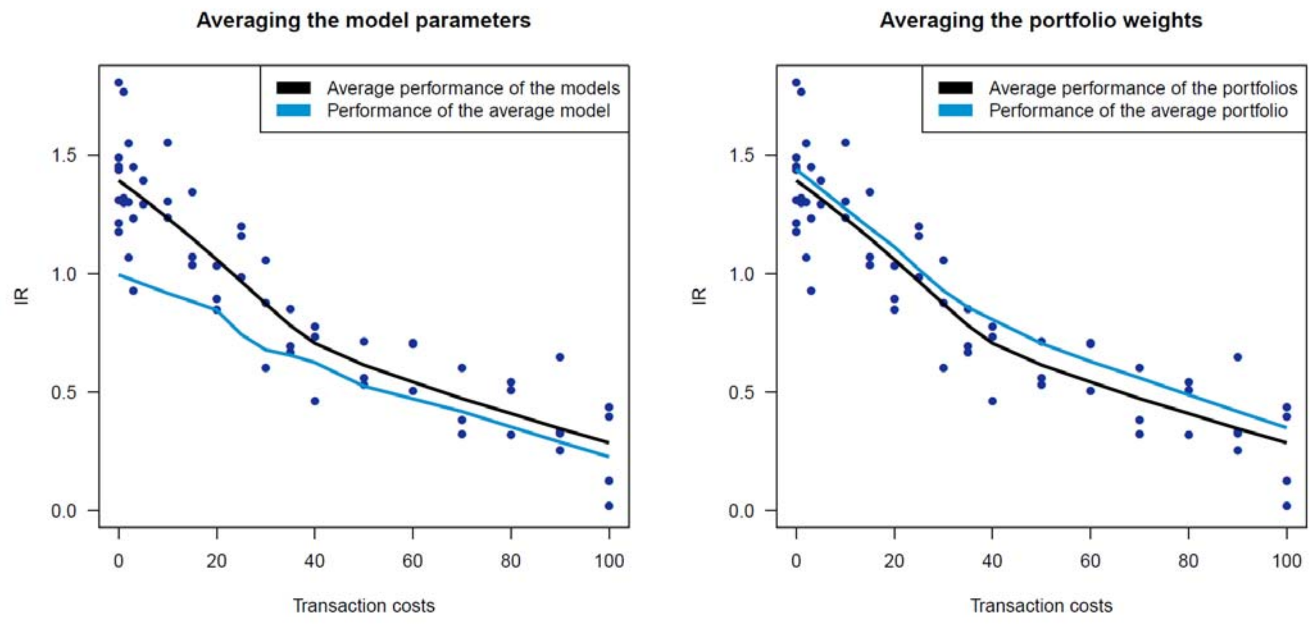
Source: Factset, S&P, Thomson Reuters, IHS Markit, Deutsche Bank Quantitative Strategy Research

Ensembling

Since we have several strategies, we can try to combine them – hopefully, this will give more stable results. The following figure shows that averaging the model parameters degrades performance, while averaging the portfolios weights (*ensembling*) gives a slight improvement.

Ensembling, i.e., averaging the outputs of several models, often improves performance.

Figure 40: Averaging model parameters or portfolio weights. Averaging the model parameters (β in Figure 4) degrades performance – it is usually a bad idea. Ensembling, i.e., averaging the portfolio weights (w in Figure 4) improves performance, but not by much – since the model is mostly linear (the only non-linear part is the rescaling of the weights), only a small improvement could be expected.



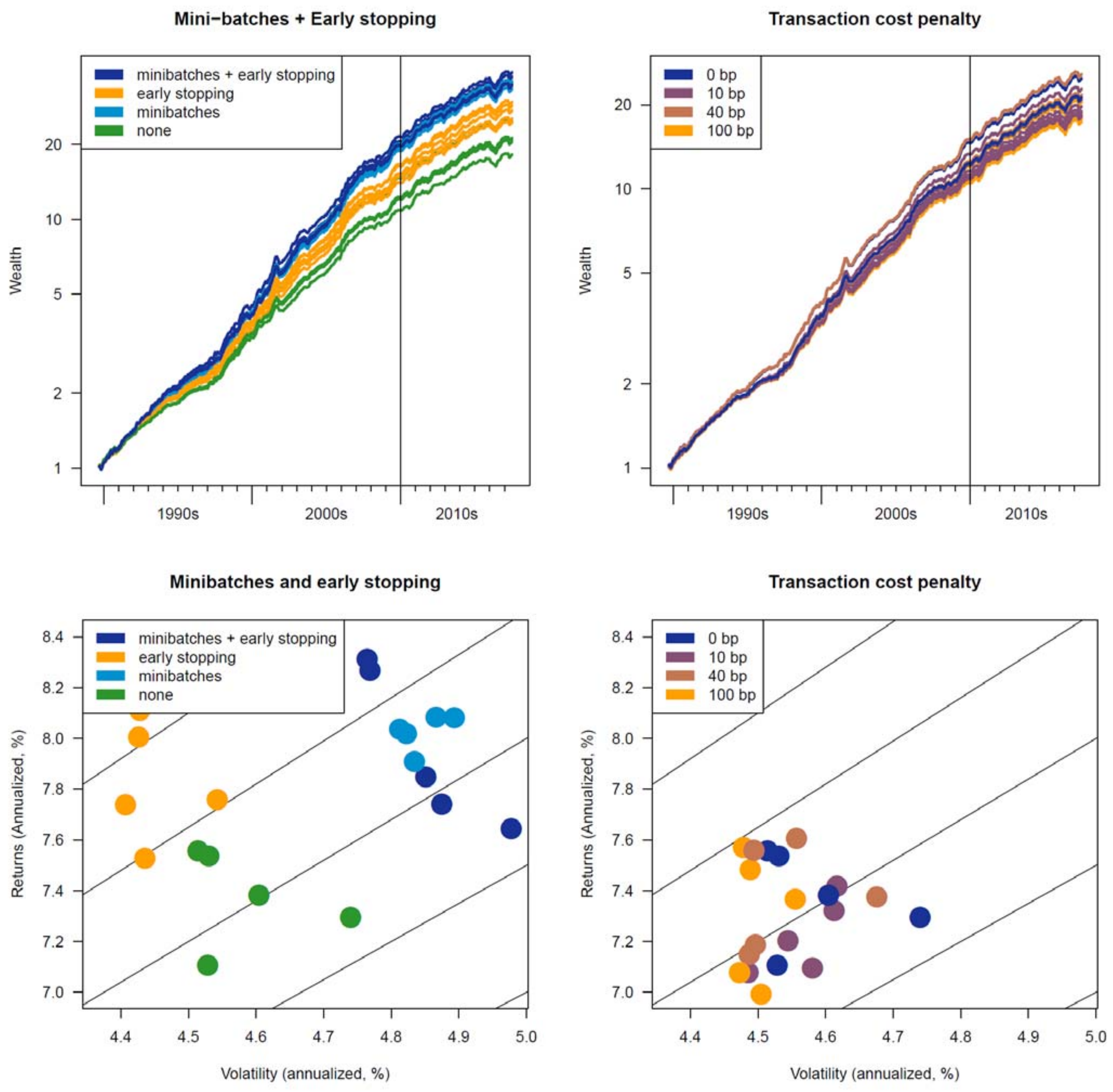
Source: Factset, S&P, Thomson Reuters, IHS Markit, Deutsche Bank Quantitative Strategy Research

Final model

It is possible to combine all those regularization procedures, as shown on Figure 41.

Our final strategy will be a linear model, estimated with stochastic gradient descent (i.e., with mini-batches), without early stopping (when combined with mini-batches, it seems to add noise), with assumed transaction costs of 100bp (not because the actual transaction costs are that high, but because of their regularizing effect); 10 models will be estimated and the resulting stock weights will be averaged (ensembling).

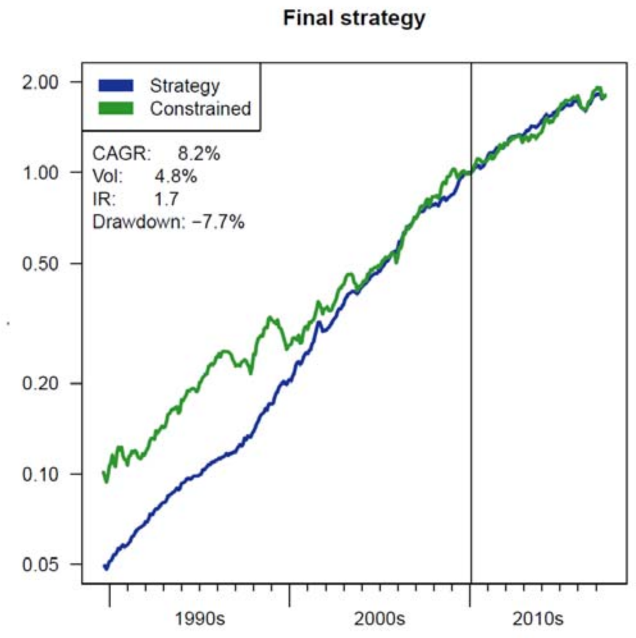
Figure 41: Combining different regularization procedures. Early stopping and mini-batches improve the in- and out-of-sample performance (top left) but combining both does not improve much on mini-batches alone and produces more dispersed results (bottom left). The transaction costs do not have a significant impact on the returns (top right) but they slightly reduce volatility (bottom right).

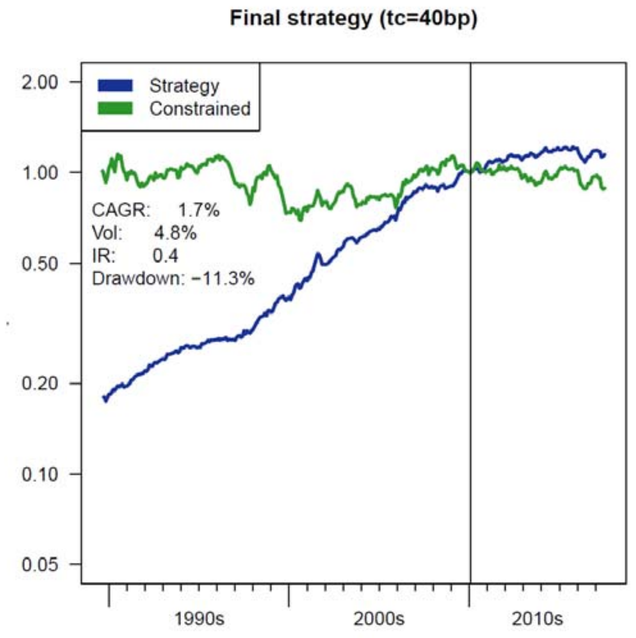


Source: Factset, S&P, Thomson Reuters, IHS Markit, Deutsche Bank Quantitative Strategy Research

Figure 42: Out-of-sample performance of the final strategy, compared with that from a constrained regression, as the transaction costs (TC) increase from 0bp to 40bp. The performance drops in both cases, but remains positive for the transaction-cost-aware strategy.

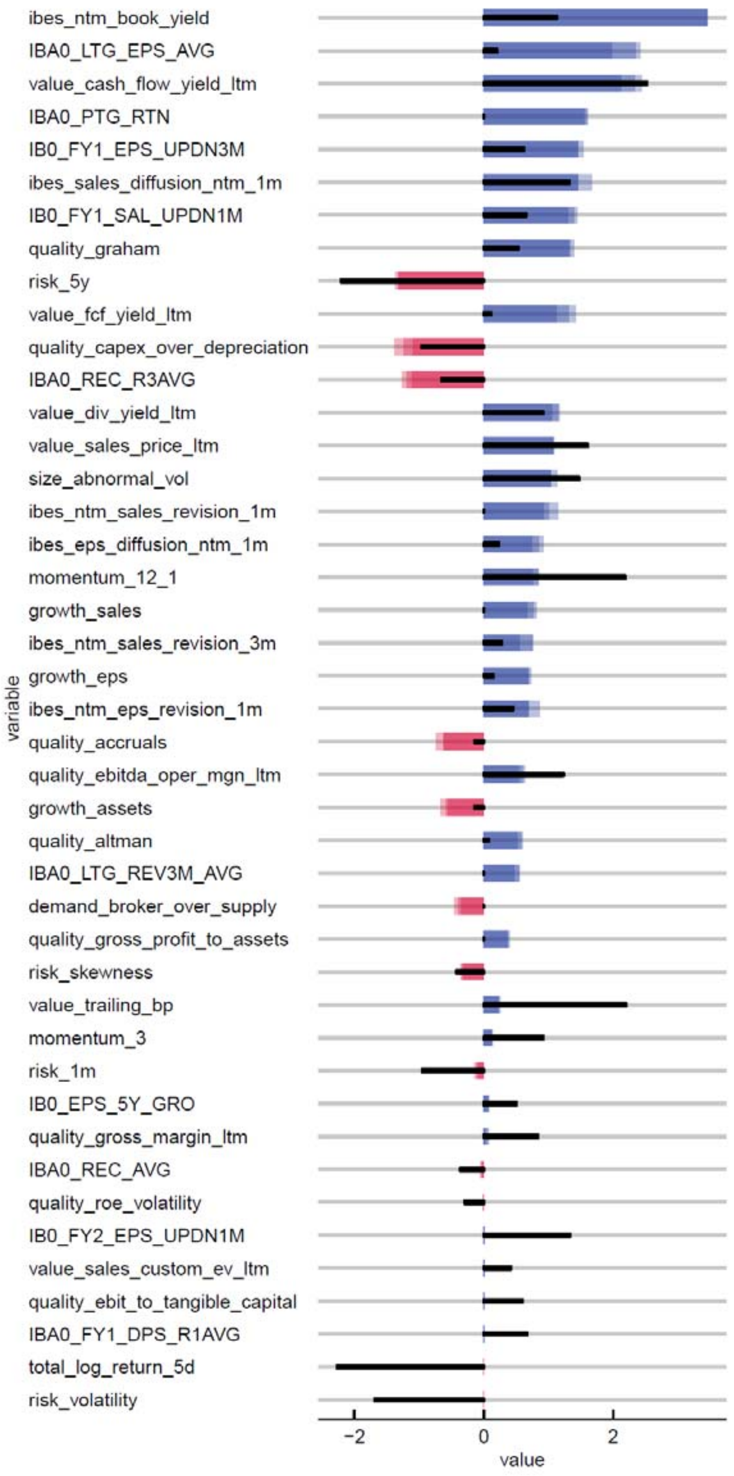
Strategy	TC	CumulativeReturn	CAGR (%)	AnnVol (%)	IR	tstat	Skew	Kurtosis	HitRatio (%)	MaxDD (%)	VaR95 (%)	ES95 (%)
optimized	0	80.4	8.2	4.8	1.70	4.6	−0.53	0.4	70	−7.7	−1.5	−2.5
optimized	10	60.9	6.5	4.8	1.36	3.7	−0.52	0.4	68	−8.5	−1.6	−2.6
optimized	20	43.4	4.9	4.8	1.02	2.8	−0.52	0.4	66	−9.4	−1.8	−2.8
optimized	30	27.8	3.3	4.8	0.69	1.9	−0.52	0.3	64	−10.4	−1.9	−2.9
optimized	40	13.9	1.7	4.8	0.37	1.1	−0.52	0.3	61	−11.3	−2.0	−3.0
constrained	0	80.0	8.1	7.3	1.11	3.0	−0.59	0.6	67	−10.0	−2.9	−4.2
constrained	10	50.6	5.6	7.3	0.76	2.1	−0.59	0.6	62	−10.7	−3.1	−4.4
constrained	20	26.0	3.1	7.3	0.43	1.3	−0.59	0.6	60	−11.9	−3.3	−4.6
constrained	30	5.4	0.7	7.3	0.10	0.4	−0.59	0.6	59	−13.6	−3.6	−4.8
constrained	40	−11.8	−1.7	7.3	−0.23	−0.5	−0.60	0.5	58	−18.2	−3.8	−5.0





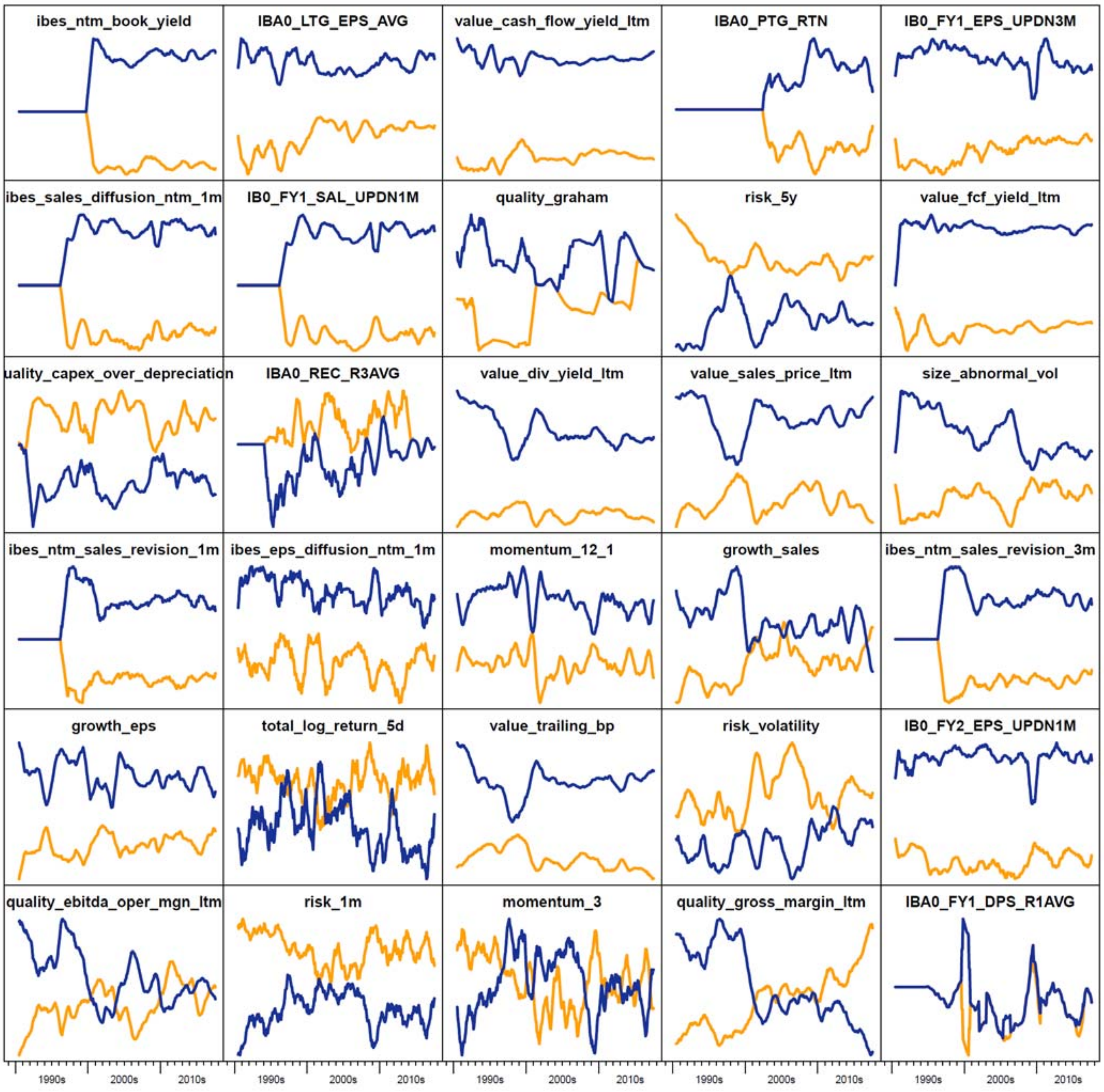
Source: Factset, S&P, Thomson Reuters, IHS Markit, Deutsche Bank Quantitative Strategy Research

Figure 43: Model weights for the models in the final ensemble; the transparency (red for negative values, blue for positive ones) indicates the median and quartiles; the black bars are the constrained regression model. Note that the weights are significantly different and that many short-term signals present in the constrained regression model have disappeared.



Source: Factset, S&P, Thomson Reuters, IHS Markit, Deutsche Bank Quantitative Strategy Research

Figure 44: Exposures of the final portfolio: weighted median of each (uniformized) signal, with the long (blue) and short (orange) portfolio weights, smoothed with a 1-year moving average, for the factors with the largest coefficients in the optimized strategy or the reference model (constrained regression). For some factors in the reference model but not in the optimized one, e.g., gross margin, the exposure is significant, but its sign switches over time. Since the model contains factors that may pull in opposite directions, this phenomenon may also happen, to a lesser extent, with factors in the model (e.g., growth sales).



Source: Factset, S&P, Thomson Reuters, IHS Markit, Deutsche Bank Quantitative Strategy Research

Conclusion

Methodology

From a methodological point of view, here is what we have seen in this report.

- **End-to-end portfolio construction**, going directly from a large set of investment signals to portfolio weights or even actual trades, in a 1-step procedure, is possible.
- We are no longer limited to the square loss function: we can minimize **any meaningful loss function**, in particular, arbitrary performance measures, which make more sense to investors.
- The **optimization algorithm** does not matter much, but those using gradient (or Hessian) information tend to be faster – in practice, we recommend to try several, check that they give similar results, and keep the fastest.
- **Deep learning frameworks** allow us to optimize arbitrarily complex portfolio construction procedures – for instance, part of the procedure could include the estimation of a risk model, or an actual portfolio optimization. They are valuable even when we are not looking into deep neural networks and just want a linear model.
- There are many ways of **regularizing** the model:
  - By using a different subset of the data at each step (mini-batches);
  - By using a different subset of the variables at each step (dropout);
  - With ensembling;
  - With early stopping;
  - By adding a meaningful penalty, e.g., by increasing the transaction costs.

Example

From a more practical point of view, we have presented a new strategy, combining 100 different signals, aware of transaction costs:

- Since the model is **linear**, and the sign of the coefficients **constrained**, the risk of overfitting the data is limited; this also ensures that the model remains interpretable.
- In spite of that simplicity, the strategy improves on the constrained regression introduced in our report on machine learning in finance. The improvement is mostly due to a reduction in risk, coming from the **objective function** – the information ratio penalizes for risk – and the **regularization** (via transaction costs and mini-batches).

- Running the optimization several times gives somewhat different portfolios, with similar risk and performance profiles, thereby partially alleviating fears of capacity limitations. If capacity were still an issue, we could rerun the optimization with different objective functions – most performance ratios should give portfolios with a comparable risk-return profile, the portfolio weights could be very different.

Possible improvements

What we have presented in this report is just a proof of concept, a general framework which can be adapted to each investor's needs. Here are a few ideas for improvements.

- Allow for arbitrary functions, instead of linear or monotonic ones, e.g., with Gaussian processes [34], splines [42] or deeper neural networks. In particular, we expect most of the investment factors to have a reliable monotonic predictive power on future returns for average values, but a less reliable and perhaps non-monotonic behaviour for extreme values.
- Add interactions between variables, e.g., between fast and slow signals [40].
- Include portfolio optimization (it is piecewise differentiable [3, 22]) and risk model estimation as preliminary steps, as suggested by Figure 25.
- Add more data. For instance, since most technical indicators can be modeled as simple, shallow neural networks, we could easily leverage daily data and ask the computer to re-invent those indicators. Adding intraday data would even allow us to include trade implementation at the end of our portfolio construction pipeline – this may require more complex techniques, in particular reinforcement learning [35];
- We do not have to limit ourselves to stock-level data: we can add monthly econometric time series (inflation, oil, etc.) – but we should make sure that those variables are not used to identify past periods: the more data we have, the more complex the model we can fit, but the higher the risk of overfitting.
- Instead of optimizing a single measure of performance, we could optimize several: multi-objective optimization does not look for a single solution, but a whole Pareto front of efficient solutions [10].
- The model could be made more Bayesian by outputting, not just a single set of portfolio weights, but a whole posterior distribution of weights, as a measure of the confidence the algorithm has in its results; Bayesian neural networks already do that [7].

References

[1] *Fundamental indexation*
R.D. Arnott, J. Hsu, and P. Moore (2005)
http://www.etf.com/docs/Fundamental%20Indexation_Arnott.pdf

[2] *Balanced baskets: a new approach to trading and hedging risks*
D.H. Bailey and M. Lopez de Prado (2012)
https://ssrn.com/abstract=2066170

[3] *An open-source implementation of the critical line algorithm for portfolio optimization*
D.H. Bailey and M. López de Prado (2013)
https://ssrn.com/abstract=2197616

[4] *Lectures on modern convex optimization*
A. Ben-Tal and A. Nemirovski (2012)
http://www2.isye.gatech.edu/~nemirovs/Lect_ModConvOpt.pdf

[5] *Algorithms for hyper-parameter optimization*
J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl (2011)
https://papers.nips.cc/paper/4443-algorithms-for-hyper-parameter-optimization.pdf

[6] *Hyperopt: a Python library for optimizing the hyperparameters of machine learning algorithms*
J. Bergstra, D. Yamins, and D.D Cox (SciPy 2013)
https://conference.scipy.org/proceedings/scipy2013/pdfs/bergstra_hyperopt.pdf
http://www.youtube.com/watch?v=Mp1xnPfE4PY

[7] *Weight uncertainty in neural networks*
C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra (2015)
https://arxiv.org/abs/1505.05424

[8] *Optimization methods in finance*
G. Cornuejols and R. Tütüncü (2006)

[9] *Equilibrated adaptive learning rates for nonconvex optimization*
Y.N. Dauphin, H. de Vries, and Y. Bengio (2015)
https://arxiv.org/abs/1502.04390

[10] *A fast and elitist multiobjective genetic algorithm: NSGA-II*
K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan (2000)

[11] *SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives*
A. Defazio, F. Bach, and S. Lacoste-Julien (NIPS 2014)
https://arxiv.org/abs/1407.0202

[12] *Efficient benchmarking of hyperparameter optimizers via surrogates*
K. Eggensperger, F. Hutter, H.H. Hoos, and K. Leyton-Brown (2015)
http://aad.informatik.uni-freiburg.de/papers/15-AAAI-Surrogates.pdf

[13] *Stock market diversity*
R. Fernholz (Intech, 2005)
http://www.q-group.org/wp-content/uploads/2014/01/Stock-Market_Diversity.pdf

[14] *Diversity-weighted indexing*
R. Fernholz, R. Garvy, and J. Hannon (1998)

[15] *Stochastic portfolio theory: an overview*
R. Fernholz and I. Karatzas (Intech, 2006)
http://www.intechinvestments.com/resources

[16] *Combining value and momentum*
G.S. Fisher, R. Shah, and S. Titman (2014)
https://ssrn.com/abstract=2472936

[17] *Numerical methods and optimization in finance*
M. Gilli, D. Maringer, and E. Schumann (2011)

[18] *Tensorflow*
https://www.tensorflow.org/

[19] *Neural Turing machines*
A. Graves, G. Wayne, and I. Danihelka (2014)
https://arxiv.org/abs/1410.5401

[20] *Learning to transduce with unbounded memory*
E. Grefenstette, K.M. Hermann, M. Suleyman, and P. Blunsom (NIPS 2015)
https://arxiv.org/abs/1506.02516

[21] *Algorithmic differentiation in finance explained*
M. Henrard (2017)

[22] *Automatic differentiation*
M.J. Johnson (Montreal deep learning summer school, 2017)
http://videolectures.net/deeplearning2017_johnson_automatic_differentiation/

[23] *Accelerating stochastic gradient descent using predictive variance reduction*
R. Johnson and T. Zhang (NIPS 2013)
https://papers.nips.cc/paper/4937-accelerating-stochastic-gradient-descent-using-predictive-variance-reduction.pdf

[24] *Adam: A method for stochastic optimization*
D.P. Kingma and J. Ba (2014)
https://arxiv.org/abs/1412.6980

[25] *Convex optimization in R*
R. Koenker and I. Mizera (Journal of Statistical Software, 2014)
https://www.jstatsoft.org/article/view/v060i05/v60i05.pdf

[26] *Selected applications of convex optimization*
L. Li (2015)

[27] *Risk and asset allocation*
A. Meucci (Springer, 2006)
https://www.arpm.co/book/

[28] *An empirical investigation of methods to reduce transaction costs*
T. Moorman (2014)
https://ssrn.com/abstract=2496416

[29] *Computational information geometry for machine learning*
F. Nielsen (MLSS, 2015)
https://www.youtube.com/watch?v=X3cBhBA1nNw

[30] *Madness: a package for multivariate automatic differentiation*
S. Pav (2016)
http://past.rinfinance.com/agenda/2016/talk/StevenPav.pdf

[31] *PerformanceAnalytics: Econometric tools for performance and risk analysis*
B.G. Peterson and P. Carl (2014)
https://CRAN.R-project.org/package=PerformanceAnalytics

[32] *PyTorch*
https://github.com/pytorch/pytorch

[33] An *overview of gradient descent optimization algorithms*
S. Ruder (2016)
https://arxiv.org/abs/1609.04747

[34] *Stochastic portfolio theory: a machine learning perspective*
Y.L.K. Samo and A. Vervuurt (2016)
https://arxiv.org/abs/1605.02654

[35] *Reinforcement learning*
D. Silver (University College London, 2015)
http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html

[36] *Practical Bayesian optimization of machine learning algorithms*
J. Snoek, H. Larochelle, and R.P. Adams (NIPS 2012)
https://arxiv.org/abs/1206.2944

[37] *Using dynamic programming to optimally rebalance portfolios*
W. Sun, A. Fan, L.W. Chen, T. Schouwenaars, and M.A. Albota (Journal of trading, 2006)
http://www.cfapubs.org/doi/full/10.2469/dig.v36.n4.4322

[38] *Convex optimization in Julia*
M. Udell, K. Mohan, D. Zeng, J. Hong, S. Diamond, and S. Boyd (2014)
https://stanford.edu/~boyd/papers/pdf/convexjl.pdf

[39] *Topics in stochastic portfolio theory*
A. Vervuurt (2015)
https://arxiv.org/abs/1504.02988

[40] *A portfolio manager's guidebook to trade execution*
K. Webster, Y. Luo, M.A. Alvarez, J. Jussa, S. Wang, S. Rohal, A. Wang, D. Elledge, and G. Zhao (Deutsche Bank Quantitative Strategy Research, 2015)

[41] *Machine learning in finance*
V. Zoonekynd, K. Le Binh, and H. Sambatur (Deutsche Bank Quantitative Strategy Research, 2016)

[42] *A first momentum playbook*
V. Zoonekynd and K. Le Binh (Deutsche Bank Quantitative Strategy Research, 2014)

Appendix 1

Important Disclosures

*Other information available upon request

Prices are current as of the end of the previous trading session unless otherwise indicated and are sourced from local exchanges via Reuters, Bloomberg and other vendors . Other information is sourced from Deutsche Bank, subject companies, and other sources. For disclosures pertaining to recommendations or estimates made on securities other than the primary subject of this research, please see the most recently published company report or visit our global disclosure look-up page on our website at http://gm.db.com/ger/disclosure/DisclosureDirectory.eqsr. Aside from within this report, important conflict disclosures can also be found at https://gm.db.com/equities under the "Disclosures Lookup" and "Legal" tabs. Investors are strongly encouraged to review this information before investing.

Analyst Certification

The views expressed in this report accurately reflect the personal views of the undersigned lead analyst(s). In addition, the undersigned lead analyst(s) has not and will not receive any compensation for providing a specific recommendation or view in this report. Vincent Zoonekynd/Khoi LeBinh/Jiazi Tang/Hemant Sambatur/Elita Lai

Hypothetical Disclaimer

Backtested, hypothetical or simulated performance results have inherent limitations. Unlike an actual performance record based on trading actual client portfolios, simulated results are achieved by means of the retroactive application of a backtested model itself designed with the benefit of hindsight. Taking into account historical events the backtesting of performance also differs from actual account performance because an actual investment strategy may be adjusted any time, for any reason, including a response to material, economic or market factors. The backtested performance includes hypothetical results that do not reflect the reinvestment of dividends and other earnings or the deduction of advisory fees, brokerage or other commissions, and any other expenses that a client would have paid or actually paid. No representation is made that any trading strategy or account will or is likely to achieve profits or losses similar to those shown. Alternative modeling techniques or assumptions might produce significantly different results and prove to be more appropriate. Past hypothetical backtest results are neither an indicator nor guarantee of future returns. Actual results will vary, perhaps materially, from the analysis.

Additional Information

The information and opinions in this report were prepared by Deutsche Bank AG or one of its affiliates (collectively "Deutsche Bank"). Though the information herein is believed to be reliable and has been obtained from public sources believed to be reliable, Deutsche Bank makes no representation as to its accuracy or completeness. Hyperlinks to third-party websites in this report are provided for reader convenience only. Deutsche Bank neither endorses the content nor is responsible for the accuracy or security controls of these websites.

If you use the services of Deutsche Bank in connection with a purchase or sale of a security that is discussed in this report, or is included or discussed in another communication (oral or written) from a Deutsche Bank analyst, Deutsche Bank may act as principal for its own account or as agent for another person.

Deutsche Bank may consider this report in deciding to trade as principal. It may also engage in transactions, for its own account or with customers, in a manner inconsistent with the views taken in this research report. Others within Deutsche Bank, including strategists, sales staff and other analysts, may take views that are inconsistent with those taken in this research report. Deutsche Bank issues a variety of research products, including fundamental analysis, equity-linked analysis, quantitative analysis and trade ideas. Recommendations contained in one type of communication may differ from recommendations contained in others, whether as a result of differing time horizons, methodologies or otherwise. Deutsche Bank and/or its affiliates may also be holding debt or equity securities of the issuers it writes on. Analysts are paid in part based on the profitability of Deutsche Bank AG and its affiliates, which includes investment banking, trading and principal trading revenues.

Opinions, estimates and projections constitute the current judgment of the author as of the date of this report. They do not necessarily reflect the opinions of Deutsche Bank and are subject to change without notice. Deutsche Bank provides liquidity for buyers and sellers of securities issued by the companies it covers. Deutsche Bank research analysts sometimes have shorter-term trade ideas that are consistent or inconsistent with Deutsche Bank's existing longer term ratings. Trade ideas for equities can be found at the SOLAR link at http://gm.db.com. A SOLAR idea represents a high conviction belief by an analyst that a stock will outperform or underperform the market and/or sector delineated over a time frame of no less than two weeks. In addition to SOLAR ideas, the analysts named in this report may from time to time discuss with our clients, Deutsche Bank salespersons and Deutsche Bank traders, trading strategies or ideas that reference catalysts or events that may have a near-term or medium-term impact on the market price of the securities discussed in this report, which impact may be directionally counter to the analysts' current 12-month view of total return or investment return as described herein. Deutsche Bank has no obligation to update, modify or amend this report or to otherwise notify a recipient thereof if any opinion, forecast or estimate contained herein changes or subsequently becomes inaccurate. Coverage and the frequency of changes in market conditions and in both general and company specific economic prospects make it difficult to update research at defined intervals. Updates are at the sole discretion of the coverage analyst concerned or of the Research Department Management and as such the majority of reports are published at irregular intervals. This report is provided for informational purposes only and does not take into account the particular investment objectives, financial situations, or needs of individual clients. It is not an offer or a solicitation of an offer to buy or sell any financial instruments or to participate in any particular trading strategy. Target prices are inherently imprecise and a product of the analyst’s judgment. The financial instruments discussed in this report may not be suitable for all investors and investors must make their own informed investment decisions. Prices and availability of financial instruments are subject to change without notice and investment transactions can lead to losses as a result of price fluctuations and other factors. If a financial instrument is denominated in a currency other than an investor's currency, a change in exchange rates may adversely affect the investment. Past performance is not necessarily indicative of future results. Unless otherwise indicated, prices are current as of the end of the previous trading session, and are sourced from local exchanges via Reuters, Bloomberg and other vendors. Data is sourced from Deutsche Bank, subject companies, and in some cases, other parties.

The Deutsche Bank Research Department is independent of other business areas divisions of the Bank. Details regarding our organizational arrangements and information barriers we have to prevent and avoid conflicts of interest with respect to our research is available on our website under Disclaimer found on the Legal tab.

Macroeconomic fluctuations often account for most of the risks associated with exposures to instruments that promise to pay fixed or variable interest rates. For an investor who is long fixed rate instruments (thus receiving these cash flows), increases in interest rates naturally lift the discount factors applied to the expected cash flows and thus cause a loss. The longer the maturity of a certain cash flow and the higher the move in the discount factor, the higher will be the loss. Upside surprises in inflation, fiscal funding needs, and FX depreciation rates are among the most common adverse macroeconomic shocks to receivers. But counterparty exposure, issuer creditworthiness, client segmentation, regulation (including changes in assets holding limits for different types of investors), changes in tax policies, currency convertibility (which may constrain currency conversion, repatriation of profits and/or the liquidation of positions), and settlement issues related to local clearing houses are also important risk factors to be considered. The sensitivity of fixed income instruments to macroeconomic shocks may be mitigated by indexing the contracted cash flows to inflation, to FX depreciation, or to specified interest rates – these are common in emerging markets. It is important to note that the index fixings may -- by construction -- lag or mis-measure the actual move in the underlying variables they are intended to track. The choice of the proper fixing (or metric) is particularly important in swaps markets, where floating coupon rates (i.e., coupons indexed to a typically short-dated interest rate reference index) are exchanged for fixed coupons. It is also important to acknowledge that funding in a currency that differs from the currency in which coupons are denominated carries FX risk. Naturally, options on swaps (swaptions) also bear the risks typical to options in addition to the risks related to rates movements.

Derivative transactions involve numerous risks including, among others, market, counterparty default and illiquidity risk. The appropriateness or otherwise of these products for use by investors is dependent on the investors' own circumstances including their tax position, their regulatory environment and the nature of their other assets and liabilities, and as such, investors should take expert legal and financial advice before entering into any transaction similar to or inspired by the contents of this publication. The risk of loss in futures trading and options, foreign or domestic, can be substantial. As a result of the high degree of leverage obtainable in futures and options trading, losses may be incurred that are greater than the amount of funds initially deposited. Trading in options involves risk and is not suitable for all investors. Prior to buying or selling an option investors must review the "Characteristics and Risks of Standardized Options”, at http://www.optionsclearing.com/about/publications/character-risks.jsp. If you are unable to access the website please contact your Deutsche Bank representative for a copy of this important document.

Participants in foreign exchange transactions may incur risks arising from several factors, including the following: (i) exchange rates can be volatile and are subject to large fluctuations; (ii) the value of currencies may be affected by numerous market factors, including world and national economic, political and regulatory events, events in equity and debt markets and changes in interest rates; and (iii) currencies may be subject to devaluation or government imposed exchange controls which could affect the value of the currency. Investors in securities such as ADRs, whose values are affected by the currency of an underlying security, effectively assume currency risk.
Unless governing law provides otherwise, all transactions should be executed through the Deutsche Bank entity in the investor's home jurisdiction. Aside from within this report, important conflict disclosures can also be found at https://gm.db.com/equities under the "Disclosures Lookup" and "Legal" tabs. Investors are strongly encouraged to review this information before investing.

Deutsche Bank (which includes Deutsche Bank AG, its branches and all affiliated companies) is not acting as a financial adviser, consultant or fiduciary to you, any of your agents (collectively, "You" or "Your") with respect to any information provided in the materials attached hereto. Deutsche Bank does not provide investment, legal, tax or accounting advice, Deutsche Bank is not acting as Your impartial adviser, and does not express any opinion or recommendation whatsoever as to any strategies, products or any other information presented in the materials. Information contained herein is being provided solely on the basis that the recipient will make an independent assessment of the merits of any investment decision, and it does not constitute a recommendation of, or express an opinion on, any product or service or any trading strategy.

The information presented is general in nature and is not directed to retirement accounts or any specific person or account type, and is therefore provided to You on the express basis that it is not advice, and You may not rely upon it in making Your decision. The information we provide is being directed only to persons we believe to be financially sophisticated, who are capable of evaluating investment risks independently, both in general and with regard to particular transactions and investment strategies, and who understand that Deutsche Bank has financial interests in the offering of its products and services. If this is not the case, or if You are an IRA or other retail investor receiving this directly from us, we ask that you inform us immediately.

United States: Approved and/or distributed by Deutsche Bank Securities Incorporated, a member of FINRA, NFA and SIPC. Analysts located outside of the United States are employed by non-US affiliates that are not subject to FINRA regulations.

Germany: Approved and/or distributed by Deutsche Bank AG, a joint stock corporation with limited liability incorporated in the Federal Republic of Germany with its principal office in Frankfurt am Main. Deutsche Bank AG is authorized under German Banking Law and is subject to supervision by the European Central Bank and by BaFin, Germany's Federal Financial Supervisory Authority.

United Kingdom: Approved and/or distributed by Deutsche Bank AG acting through its London Branch at Winchester House, 1 Great Winchester Street, London EC2N 2DB. Deutsche Bank AG in the United Kingdom is authorised by the Prudential Regulation Authority and is subject to limited regulation by the Prudential Regulation Authority and Financial Conduct Authority. Details about the extent of our authorisation and regulation are available on request.

Hong Kong: Distributed by Deutsche Bank AG, Hong Kong Branch or Deutsche Securities Asia Limited.

India: Prepared by Deutsche Equities India Pvt Ltd, which is registered by the Securities and Exchange Board of India (SEBI) as a stock broker. Research Analyst SEBI Registration Number is INH000001741. DEIPL may have received administrative warnings from the SEBI for breaches of Indian regulations.

Japan: Approved and/or distributed by Deutsche Securities Inc.(DSI). Registration number - Registered as a financial instruments dealer by the Head of the Kanto Local Finance Bureau (Kinsho) No. 117. Member of associations: JSDA, Type II Financial Instruments Firms Association and The Financial Futures Association of Japan. Commissions and risks involved in stock transactions - for stock transactions, we charge stock commissions and consumption tax by multiplying the transaction amount by the commission rate agreed with each customer. Stock transactions can lead to losses as a result of share price fluctuations and other factors. Transactions in foreign stocks can lead to additional losses stemming from foreign exchange fluctuations. We may also charge commissions and fees for certain categories of investment advice, products and services. Recommended investment strategies, products and services carry the risk of losses to principal and other losses as a result of changes in market and/or economic trends, and/or fluctuations in market value. Before deciding on the purchase of financial products and/or services, customers should carefully read the relevant disclosures, prospectuses and other documentation. "Moody's", "Standard & Poor's", and "Fitch" mentioned in this report are not registered credit rating agencies in Japan unless Japan or "Nippon" is specifically designated in the name of the entity. Reports on Japanese listed companies not written by analysts of DSI are written by Deutsche Bank Group's analysts with the coverage companies specified by DSI. Some of the foreign securities stated on this report are not disclosed according to the Financial Instruments and Exchange Law of Japan. Target prices set by Deutsche Bank's equity analysts are based on a 12-month forecast period.

Korea: Distributed by Deutsche Securities Korea Co.

South Africa: Deutsche Bank AG Johannesburg is incorporated in the Federal Republic of Germany (Branch Register Number in South Africa: 1998/003298/10).

Singapore: by Deutsche Bank AG, Singapore Branch or Deutsche Securities Asia Limited, Singapore Branch (One Raffles Quay #18-00 South Tower Singapore 048583, +65 6423 8001), which may be contacted in respect of any matters arising from, or in connection with, this report. Where this report is issued or promulgated in Singapore to a person who is not an accredited investor, expert investor or institutional investor (as defined in the applicable Singapore laws and regulations), they accept legal responsibility to such person for its contents.

Taiwan: Information on securities/investments that trade in Taiwan is for your reference only. Readers should independently evaluate investment risks and are solely responsible for their investment decisions. Deutsche Bank research may not be distributed to the Taiwan public media or quoted or used by the Taiwan public media without written consent. Information on securities/instruments that do not trade in Taiwan is for informational purposes only and is not to be construed as a recommendation to trade in such securities/instruments. Deutsche Securities Asia Limited, Taipei Branch may not execute transactions for clients in these securities/instruments.

Qatar: Deutsche Bank AG in the Qatar Financial Centre (registered no. 00032) is regulated by the Qatar Financial Centre Regulatory Authority. Deutsche Bank AG - QFC Branch may only undertake the financial services activities that fall within the scope of its existing QFCRA license. Principal place of business in the QFC: Qatar Financial Centre, Tower, West Bay, Level 5, PO Box 14928, Doha, Qatar. This information has been distributed by Deutsche Bank AG. Related financial products or services are only available to Business Customers, as defined by the Qatar Financial Centre Regulatory Authority.

Russia: This information, interpretation and opinions submitted herein are not in the context of, and do not constitute, any appraisal or evaluation activity requiring a license in the Russian Federation.

Kingdom of Saudi Arabia: Deutsche Securities Saudi Arabia LLC Company, (registered no. 07073-37) is regulated by the Capital Market Authority. Deutsche Securities Saudi Arabia may only undertake the financial services activities that fall within the scope of its existing CMA license. Principal place of business in Saudi Arabia: King Fahad Road, Al Olaya District, P.O. Box 301809, Faisaliah Tower - 17th Floor, 11372 Riyadh, Saudi Arabia.

United Arab Emirates: Deutsche Bank AG in the Dubai International Financial Centre (registered no. 00045) is regulated by the Dubai Financial Services Authority. Deutsche Bank AG - DIFC Branch may only undertake the financial services activities that fall within the scope of its existing DFSA license. Principal place of business in the DIFC: Dubai International Financial Centre, The Gate Village, Building 5, PO Box 504902, Dubai, U.A.E. This information has been distributed by Deutsche Bank AG. Related financial products or services are only available to Professional Clients, as defined by the Dubai Financial Services Authority.

Australia: Retail clients should obtain a copy of a Product Disclosure Statement (PDS) relating to any financial product referred to in this report and consider the PDS before making any decision about whether to acquire the product. Please refer to Australian specific research disclosures and related information at https://australia.db.com/australia/content/research-information.html

Australia and New Zealand: This research is intended only for "wholesale clients" within the meaning of the Australian Corporations Act and New Zealand Financial Advisors Act respectively.

Additional information relative to securities, other financial products or issuers discussed in this report is available upon request. This report may not be reproduced, distributed or published without Deutsche Bank's prior written consent.

International Locations

Deutsche Bank AG Deutsche Bank Place Level 16 Corner of Hunter & Phillip Streets Sydney, NSW 2000 Australia Tel: (61) 2 8258 1234	Deutsche Bank AG Mainzer Landstrasse 11-17 60329 Frankfurt am Main Germany Tel: (49) 69 910 00	Deutsche Bank AG Filiale Hongkong International Commerce Centre, 1 Austin Road West,Kowloon, Hong Kong Tel: (852) 2203 8888	Deutsche Securities Inc. 2-11-1 Nagatacho Sanno Park Tower Chiyoda-ku, Tokyo 100-6171 Japan Tel: (81) 3 5156 6770
Deutsche Bank AG London 1 Great Winchester Street London EC2N 2EQ United Kingdom Tel: (44) 20 7545 8000	Deutsche Bank Securities Inc. 60 Wall Street New York, NY 10005 United States of America Tel: (1) 212 250 2500		