Discovery of multi-component portfolio strategies with continuous tuning to the changing market micro-regimes using input-dependent boosting

V.V. Gavrishchaka[1], O. Barinova[2], A. Vezhnevets[2], and M.A. Monina[3]
[1]*Alexandra Investment Management, New York, NY, USA*
[2]*Moscow State University, Dept. of Computational Mathematics and Cybernetics, Graphics and Media Lab, Moscow, Russian Federation*
[3]*Moscow State University of Instrument Engineering and Computer Science, Moscow, Russian Federation*

Abstract

Recently proposed boosting-based optimization offers a generic framework for the discovery of compact and interpretable portfolios of complimentary trading strategies with stable (non-resonant) performance over wide range of market regimes and robust generalization abilities. Inherent complexity control allows the framework to work with very large pools of heterogeneous base strategies with well-established properties. However, in its current version, the framework outputs a collection of dynamic strategies with fixed parameters and constant weights defining capital allocations. This excludes any adaptive regime adjustment or switching on the portfolio level for additional profitability from regime-specific patterns. In this work we extend boosting-based optimization framework by including capability to discover portfolio strategies with continuous and adiabatically smooth adjustment to the current market micro-regime. Such regime adaptivity is naturally provided by the input-dependent boosting. The proposed generalization preserves clarity and interpretability of the original framework since the dynamic base strategies of the multi-component portfolio and their optimal parameters remain fixed. However, the weights of the base strategies are adaptively varied in time according to the implicit rule discovered by boosting. Operational details of the new framework and encouraging results are illustrated using real market data. More rigorous theoretical foundation for the general concept of the boosting-based optimization is also outlined.

Keywords: adaptive boosting, ensemble learning, regime switching, trading strategies, portfolio optimization.

1 Introduction

One of the most pronounced challenges in financial markets modeling and forecasting is nonstationarity of the individual time series and co-dependency relations. Usually it is very difficult or practically impossible to find a single global model based on observable and well-defined variables with desired performance in all market regimes. However, with certain degree of simplification such complex nonstationary dynamics can be approximated as switching between different regimes of market dynamics where each such regime is relatively simple and easy to describe by its own parsimonious model. More generally, one can introduce additional non-observable (hidden) state variables to represent market regimes and dynamical transitions between them.

Intuitively, timely switching or smooth transition between different regimes in such structured models could significantly improve modeling and forecasting of the nonstationary time series. More importantly, this may help to create dynamic trading strategies that consistently exploit regime-specific market patterns. However, although introduction of the hidden variables in the simplified econometric models [Fabozzi] could lead to better prediction accuracy and overall explanatory power [application papers], the usefulness of such models for the discovery of the realistic trading strategies is often very limited. Moreover, potential of model improvement through the introduction of the finer structure of the market regimes (states) is also very limited since adequate model estimation is possible only when historical training data contains enough transitions between different regimes which is often not the case in most financial applications.

Certain simplifications, apriory assumptions and other drawbacks of the typical econometric models can be alleviated using dynamic machine learning approaches. For example, recurrent neural networks (NN) maintain an adaptive internal memory of past inputs that allows implicit regime adjustment or switching and do not require any specific assumptions about hidden state variables and mechanisms of regime transition [Hykin, Wan, Disser]. However, training procedures for such NNs are often significantly more complex and less stable compared to already standard feed-forward NNs such as multi-layer perceptron (MLP) that are not dynamic [Hykin, Disser]. Moreover, complex dynamic NNs lack interpretability and operational stability control that are very important requirements in financial applications.

Both statistical and machine learning forecasting models are trained using objectives that are not directly relevant to the trading strategy which limits usefulness of such models even when formal forecasting performance measures are reasonable [GB]. Technical trading strategies, directly optimized to achieve desirable profit/loss (PL) distributions, usually have more practical value than pure forecasting models mentioned above. However, single trading strategy with reasonable complexity (to ensure out-of-sample performance) still cannot warranty stability across different market regimes.

Many modern automatic systems for systematic trading offer an option for the dynamic capital reallocation among different strategies using explicit user-specified rule that may be a function of the most recent PL time series generated by each strategy and other factors. The goal of such approach is to alleviate limitations of the single trading strategy that may be profitable only in certain market regimes while maintaining simplicity and interpretability of such portfolio of trading strategies. However, these systems do not provide any generic and theoretically-sound frameworks capable to discover such regime-adjusted portfolio strategies with stable performance over wide range of market regimes. For example, usage of resonance strategies tuned to the specific market regimes and simplified empirical switching between different strategies could be very unstable since it requires a very accurate timing which is difficult to achieve in most modern markets.

Recently proposed boosting-based optimization provides a generic framework for the discovery of portfolios of trading strategies with stable (non-resonant) performance over wide range of market regimes using intelligent combination of the complimentary, low-complexity base strategies with well-known properties [VG]. This framework offers practical solutions for many problems encountered in other approaches. However, in its current version, the framework outputs a collection of dynamic strategies with fixed parameters and constant weights defining capital allocations. This excludes any adaptive regime adjustment or switching on the portfolio level for additional profitability from regime-specific patterns. In this work we extend boosting-based optimization framework by including capability to discover portfolio strategies with continuous and adiabatically smooth adjustment to the current market micro-regime. In many practical settings, the proposed framework could resolve or alleviate limitations of other approaches used for discovery of the regime-adjusted portfolio strategies.

2 Limitations of the existing regime-switching and regime-adjusted models

A flexible and generic approach to incorporate multiple interchanging regimes or states is to use models with hidden variables, i.e., auxiliary variables that are not directly observable [GeneralRef]. A well-known example of such a framework is hidden Markov models (HMM) that are successfully used alone or in combination with other statistical or machine learning algorithms in speech recognition systems [..], bioinformatics [..], and other applications. Examples of hidden variable models used in financial econometrics include regime-switching models, the GARCH family of models, and credit risk models, where hidden variables are used to represent different economic/market regimes, volatility, and credit worthiness, respectively [Fabozzi and references therein].

One of the widely used types of models with hidden variables is a linear state-space model that can be written in the following way [Fabbozi]:

$$y_\tau = Az_\tau + Bx_\tau + \varepsilon_\tau$$
$$z_{\tau+1} = Cz_\tau + Dx_\tau + \eta_\tau \quad (1)$$

Here x_τ, y_τ, and z_τ are the vectors of deterministic inputs, observable outputs, and latent (nonobservable) state variables, respectively. Observation and transition equation white noise are given by ε_τ and η_τ. Observation matrix, input matrix of the observation equation, transition matrix, and input matrix to the transition equation are given by *A*, *B*, *C*, and *D*.

A typical maximum likelihood (ML) based estimation for (1) would require approximate calculation of the nonobservable states *z*, which enters corresponding likelihood expression together with observable variables. Kalman filter [KFReference] is usually used to provide optimal estimation of the state variables. Originally, Kalman filter was designed as an adaptive filter that provide optimal estimation and forecasting of the "true" states of the dynamical system from the multi-dimensional noisy observations in engineering applications. The filter is initialized with the initial conditions and computations are carried out recursively to the desired time, i.e. full historical information is used to make current estimation and forecasting.

The intuitive notion of the "states of the market" (e.g. bull, bear, and side markets) is well known to market practitioners. From a modeling perspective, this implies that different models and strategies should be used in different market states. If we add rules that prescribe the switching from one model (or set of models) to another one when market state changes, we arrive at the regime-switching model [Fabozi, Hamilton].

A broad class of regime-switching models are the Markov-switching vector autoregressive models (MS-VAR) [Fabozi, FinMetrix]. Markov-switching model is a VAR model whose coefficients are driven by a Markov chain:

$$y_\tau = \mu(s) + \left(\sum_{i=1}^{p} A_i(s) L^i \right) y_\tau + \varepsilon_\tau(s), \quad (2)$$

where the matrices $A_i(s)$ are the coefficients of the process at lag *i* in state *s* and the noise terms $\varepsilon_\tau(s)$ are independent normal variables. The process is driven by a *k*-states Markov chain. A Markov chain is a discrete variable which can assume at each instant one of *k* possible values with transition probabilities:

$$P(s_\tau = i \mid s_{\tau-1} = j) = p_{ij} \quad (3)$$

The realized state *s* determines the coefficients and the vector of the intercepts of the process at each moment, so that the innovation term of the process is distributed as a mixture of Gaussian distributions. The state variable that could be used to represent market regime is a hidden factor. Due to the state (regime) switching MS-VAR becomes a nonlinear model even though it is based on linear components.

ML-based estimation of the MS-VAR model is presently the mostly widely used approach [Fabozi]. Similar to space-state model (1) and other models with hidden variables, one writes a likelihood function that depends on both observables and hidden variables. In the case of linear state-space models (1),

Kalman filter is naturally applicable and used to estimate values of hidden variables (states). However, in nonlinear MS-VAR models, Expectation-Maximization (EM) algorithm is used instead. The algorithm is an optimization procedure for finding the maximum of log-likelihood functions in the presence of missing or hidden variables [EMReference]. The idea of EM algorithm is to iterate between the estimate of the best parameters given the missing (hidden) data and the estimate of the missing (hidden) data given the best estimate of parameters.

The obvious limitation of econometric models with hidden state variables mentioned above is the usage of the simplified and pre-specified functional forms as in (1) and (2), (3). Compared to similar econometric models without hidden variables, the requirements for the data sample size will grow much faster with the model complexity increase. For example, even for simple regime-switching model given by (2)-(3), the increase of the regime number beyond 2 or 3 could make adequate estimation of the model unrealistic since the available training data would not have enough transitions between different regimes to ensure reasonable out-of-sample performance. Moreover, even when formal forecasting performance measures are reasonable for explanatory purposes or large-scale risk management applications, the direct usefulness of the econometric-type models to trading strategies may often be limited due to the difference in objectives [GB].

Machine learning approaches offer significantly more flexibility in many different applications including modeling of the non-stationary or multi-regime time series. However, the well-known feed-forward NNs such as widely used MLP with standard back-propagation training algorithm may be used to perform nonlinear prediction only on stationary time series. It is done by using delayed vector of inputs as follows:

$$y_\tau = F(x_{\tau-1}, x_{\tau-2}, ..., x_{\tau-p}) \quad (4)$$

Here F is a general functional dependence that NN will try to approximate when p delayed input variables will be presented at the input and one output variable will be generated at the NN output. The adequate choice of the lag space could be a difficult problem-dependent task. However, after the lag space is chosen, it remains constant. This limits NN's adaptivity and makes it a static model which can describe only stationary time series.

To make NN dynamic it must be given an "internal" memory [Elman, Hykin]. To accomplish this and to retain simplicity of the feed-forward architecture, one can introduce time delays into the synaptic structure of the NN and to adjust their values during the learning phase. Such time delays are also neurobiologically motivated. One of such practical models where MLP with synapses represented by a finite-duration impulse response (FIR) filter was introduced by Wan [WanDiss]. The efficient training procedure called temporal back-propagation was also proposed for such FIR MLP [WanDiss, Hykin]. One of the examples when effectiveness of the FIR MLP was demonstrated is the winning of the time series prediction competition [TSCompetRef].

A more general dynamic model denoted as a recurrent NN is obtained when the connectivity of the feed-forward network is extended to include feedback

connections from the outputs of the units back to their inputs [Hykin, ReccNNDiss]. The most general recurrent NN is obtained if the output from every unit in the NN is fed back to the inputs of all units. The advantage of the recurrent NNs compared to the feed-forward NNs is due to an internal memory of past inputs introduced by the feedback connections. This internal memory is adaptive, i.e., during training it may be adapted to encompass those previous inputs which are relevant to the current problem. Such adaptive memory may completely relieve the user from specifying a lag space, since a fully recurrent NN is able to work entirely from its own internal memory, created from only a single external input [ReccNNRef]:

$$y_\tau = F(x_{\tau-1}) \tag{5}$$

However, despite their advantages recurrent NNs have not gained popularity similar to that of feed-forward NNs. Generally, it is more difficult to handle recurrent NNs in practice. In particular it has been found that training using widely accepted algorithms (e.g., gradient descent method and its extensions) is not sufficiently "powerful" to train recurrent NNs [ReccNNRef]. The problem could be in slow convergence or in complete failure to provide practically acceptable solution.

Even when training challenges are resolved, the remaining common problems with NNs and similar black-box machine learning models are poor interpretability and occasional instable behavior which is hard to control and predict. Although these drawbacks are especially undesirable in financial applications, the high noise-to-signal ratio in financial time series enhances probability of unstable behavior of such models.

One of the machine learning approaches to compensate for deficiency of the individual models is to combine several models to form a committee [Bishop, Hustee]. Committee can compensate limitations of the individual models due to both incomplete data and specifics of the algorithms (e.g., multiple local minima in the NN error surface). A number of different ensemble learning techniques to build optimal committees have been proposed over the years in different research communities [e.g., VG and references therein]. The most relevant framework of this type in the context of the regime-switching or regime-adjusted models is mixture of expert (ME) model [Jacobs].

The probabilistic form of ME model can be written as [Bishop]

$$p(y \mid x) = \sum_{t=1}^{T} \pi_t(x)\, p_t(y \mid x) \tag{6}$$

Here mixing coefficients $\pi_t(x)$ are known as gating functions and the individual component densities $p_t(y|x)$ are called experts. This terminology is due to the fact that different components can model the distribution in different regions of input space (they are "experts" at making predictions in their own regions), and the gating functions determine which components are dominant in which region. If the experts are linear (regression or classification) models, then the whole model can be fitted efficiently using the EM algorithm. An even more flexible model is obtained by using a multilevel gating function to give the hierarchical mixture of experts (HME) model [Bishop]. This model can be imagined as a mixture

distribution in which each component in the mixture is itself a mixture distribution.

More complex machine learning models can be used as components in both ME and HME frameworks. For example, NNs can be used as experts as well as gating functions [Hykin]. However, increasing complexity of the models could often lead to practical problems in training of such multi-component system and to poor out-of-sample performance and instability. Thus, it is highly desirable to have committee of the well-understood and low-complexity expert models that consistently demonstrate acceptable combined performance.

Adaptive boosting is a powerful ensemble learning algorithm that combines many desirable features [Shap, Ram, Hastee]. Many ensemble learning algorithms including "random sample" techniques like bagging can reduce only variance part of the model error, i.e. they make a combined model more stable. Boosting, on the other hand, can reduce both bias and variance parts of the model error. It means that one can start with simple model ("rule of thumb") with low accuracy and produce committee with much higher accuracy. Therefore, boosting can be applied to the pool of the well-understood low-complexity models to produce qualitatively different but interpretable combined model with significantly higher accuracy and stability [vg]. Moreover, boosting tries to maximize margin to ensure good out-of-sample performance, i.e. it is a large-margin classifier [Shap, Ram, Hastee].

In our previous works [vg] we propose a boosting-based optimization framework for the discovery of the stable portfolios of trading strategies from the low-complexity base strategies. It was argued that boosting for classification can be used as a basis for such stage-wise optimization framework. The final output of the boosting-based optimization is a collection S of complimentary base trading strategies BS_i with optimal parameter vectors p_i and combination weights w_i that specify capital allocation for each strategy [vg]:

$$S \to \left\{[w_1, BS_1(p_1)],...,[w_T, BS_T(p_T)]\right\} \tag{7}$$

We have shown empirically [vg] that boosting-based optimization can generate portfolios (7) with stable performance over wide range of market regimes while using just well-known technical base strategies. However, in its current version, the framework outputs a collection of dynamic strategies (7) with fixed parameters $\boldsymbol{p}$ and constant weights $\boldsymbol{w}$. This excludes any adaptive regime adjustment or switching on the portfolio level for additional profitability from regime-specific patterns. Many modern automatic systems for systematic trading offer an option for the dynamic capital reallocation among different strategies in (7) using explicit user-specified rule. However, these systems do not provide any generic and theoretically-sound frameworks capable to discover regime-adjusted portfolio strategies with stable performance over wide range of market regimes. Switching-rule specification based just on the semi-quantitative discretional arguments could often lead to the unreliable trading system.

In this work we extend boosting-based optimization framework by including capability to discover portfolio strategies (7) where weights $\boldsymbol{w}$ are continuously adjusted by the implicit rule discovered by the input-dependent boosting. In the next two sections we summarize our previous results on the boosting-based

optimization as well as provide more formal theoretical foundation for the framework. After that we introduce extended framework based on the input-dependent boosting [Jin] and present regime-adjusted trading strategy obtained from the new framework using real-market data.

3 Boosting-based optimization: Discovery of the regime-independent portfolio strategies

As described in our previous works [VG], boosting for optimization could be based on different boosting frameworks. However, the generalized AdaBoost algorithm for classification [Shapiro, Ram] could be a reasonable choice in many applications due to its simplicity, comprehensive theoretical foundation, and proven robust performance in a large number of realistic classification problems. For our purposes it is sufficient to describe boosting algorithm only for two-class classification problem, where classifier outputs either +1 or -1. Generalized AdaBoost for two-class classification consists of the following steps [Ram]:

$$w_n^1 = 1/N \tag{8.1}$$

$$\varepsilon_t = \sum_{n=1}^{N}\left(w_n^t I(-y_n h_t(x_n))\right) \tag{8.2}$$

$$\gamma_t = \sum_{n=1}^{N}\left(w_n^t y_n h_t(x_n)\right) \tag{8.3}$$

$$\alpha_t = \frac{1}{2}\ln\left(\frac{1+\gamma_t}{1-\gamma_t}\right) - \frac{1}{2}\ln\left(\frac{1+\rho}{1-\rho}\right) \tag{8.4}$$

$$w_n^{t+1} = w_n^t \exp\left(-\alpha_t y_n h_t(x_n)\right)/Z_t \tag{8.5}$$

$$H(x) = \sum_{t=1}^{T}\alpha_t h_t(x) / \sum_{t=1}^{T}\alpha_t \tag{8.6}$$

Here N is a number of training data points, x_n is a model/classifier input set of the n-th data point and y_n is the corresponding class label (i.e., -1 or +1), $I(z) = 0$ for $z<0$ and $I(z)=1$ otherwise, T is a number of boosting iterations, w_n^t is a weight of the n-th data point at t-th iteration, Z_t is weight normalization constant at t-th iteration, $h_t(x_n)$->[-1; +1] is the best base hypothesis/model at t-th iteration, ρ is a margin control parameter, and $H(x)$ is a final weighted linear combination of the base hypotheses.

Boosting starts with equal and normalized weights for all training data (step (8.1)). A base classifier $h_t(x)$ is trained using weighted error function ε_t (step

(8.2)). If a pool of several types of base classifiers is used, then each of them is trained and the best one (according to error function) is chosen at the current iteration. The training data weights for the next iteration are computed in steps (8.3)-(8.5).

According to (8.5), at each boosting iteration, data points misclassified by the current best model (i.e., $y_n\, h_t(x_n) < 0$) are penalized by the weight increase for the next iteration. In subsequent iterations, AdaBoost constructs progressively more difficult learning problems that are focused on hard-to-classify patterns. This process is controlled by the weighted error function (8.2).

Steps (8.2)-(8.5) are repeated at each iteration until stop criteria $\gamma_t < \rho$ (i.e., $\varepsilon_t >= 1/2(1-\rho)$) or $\gamma_t = 1$ (i.e., $\varepsilon_t = 0$) occurs. Step (8.6) represents the final combined (boosted) model that is ready to use. The model classifies unknown sample as class +1 when $H(x) > 0$ and as -1 otherwise.

Portfolio strategy discovery is a direct optimization rather than classification problem. However, it was argued [VG] that for a large class of objective functions, boosting for classification (8.1)-(8.6) can be efficiently used as a basis for the framework that could be labeled as “boosting for optimization” or “boosting-based optimization”.

One of the natural and robust objectives for the trading strategy optimization is to require returns (r) generated by the strategy on a chosen time horizon (τ) to be above certain conservative threshold (r_c). By calculating strategy returns on a series of intervals of length τ shifted with a step $\Delta\tau$ and encoding them as +1 (for $r >= r_c$) and -1 (for $r < r_c$), one obtains symbolically encoded distribution of strategy returns.

Contrary to the classification problems, here the purpose is not to correctly classify (between +1 and -1), but rather to increase the number of +1 samples. This can still be considered as classification problem with potentially uneven sample number between two classes. The described objective can be incorporated into the boosting operation (8.1)-(8.6) by considering output -1 as misclassification. In such setting, boosting (8.1)-(8.6) provides a framework for optimization, where maximization objection function is a “hit rate”, i.e., number of +1 samples divided by the total number of samples. The objective function can be generalized to include any complex condition with combination of different objectives for profit maximization and risk minimization.

In the case of trading strategy optimization, the final usage of boosting output is different from the classical case of boosting for classification. Instead of using weighted linear combination (8.6) of the base models as a final model for classification, one uses boosting weights to construct portfolio of strategies. The initial capital is distributed among different base strategies in amounts according to the weights ($\alpha_t/\Sigma\alpha_t$) obtained from boosting which are already normalized.

As discussed in [VG], boosting can be used to discover the optimal combination of different dynamic trading strategies for a single financial instrument as well as the simultaneous combination of trading strategies and

different instruments. In both cases, boosting steps (8.1)-(8.6) are applied to a pool of base strategies $\{BS_i(\boldsymbol{p}_i)\}$, where $\boldsymbol{p}_i$ is a vector of adjustable parameters for strategy BS_i. However, in the first case all base strategies are applied to a time series of a single financial instrument FI_0, while set $\{BS_i\}\text{x}\{FI_j\}$ of all possible pairs of strategies BS_i and instruments FI_j should be used in the latter case.

According to error function (8.2), if the objective is to maximize the number of supercritical returns on the shifted intervals, the following optimization problems are solved for all base strategies $BS_i(\boldsymbol{p}_i)$ and financial instruments FI_j at each boosting iteration:

$$\min_{p_i}\left[\sum_{n=1}^{N} w_n^{(t)} I\left(r_c - r_n^{\tau}\left(BS_i(p_i), FI_j\right)\right)\right] \tag{9}$$

Here, r_n^{τ} is a return produced by the strategy $BS_i(p_i)$ applied to the instrument FI_j over n-th shifted interval of length τ and r_c is a chosen threshold value. Often linear function $\sim (r_c - r_n)$ is better choice for $z<0$ compared to standard step-function I. Based on the results of these minimization procedures for all (i,j) pairs, the best pair "strategy-instrument" of the current iteration is added to the portfolio. This procedure was also generalized to allow simultaneous discovery of new synthetic instruments expressed as multiple spreads between base instruments and dynamic trading strategies for such spreads [vg].

The AdaBoost algorithm given by (8.1)-(8.6) is one of the classical versions and it was described as an example suitable for boosting-based optimization in all our previous works. However, a slightly different version is both more natural and more convenient for the extension to the input-dependent boosting [Jin]. For simplicity, we will omit the margin parameter ρ in all the following formulations. The other standard AdaBoost formulation given in [Jin], results in a different weight update procedure. Instead of (8.5) one applies:

$$w_n^T = \frac{e^{-H_{T-1}(x_n)y_n}}{\sum_{j=1}^{N} e^{-H_{T-1}(x_j)y_j}} \tag{10}$$

Expression for α is given by the following form equivalent to (8.4) with omitted margin parameter:

$$\alpha_T = \frac{1}{2}\ln\left(\frac{\sum_{n=1}^{N} w_n^T I(h_T(x_n), y_n)}{\sum_{n=1}^{N} w_n^T I(-h_T(x_n), y_n)}\right) = \frac{1}{2}\ln\left(\frac{1-\varepsilon_T}{\varepsilon_T}\right) \tag{11}$$

As mentioned in our previous publications [vg], boosting also allows natural and easy incorporation of other ensemble learning techniques on the inside and outside levels for further improvement of the accuracy and stability. For example, at each boosting iteration, instead of choosing a single best model, one can choose mini-ensemble of models using other ensemble learning techniques. In the simplest case, this could be an equal weight mini-ensemble of several comparable best models. One can also use boosting in the HME-like framework by building portfolio of portfolios.

As summarized in this section, it is quite intuitive to use boosting for classification as a basis for the boosting-based optimization framework. However, to better understand basic features of the boosting-based optimization and its possible generalizations and limitations, it is still useful to outline formal theoretical foundation for the framework. A short version of such foundation is presented in the next section.

4 Theoretical foundation for boosting-based optimization

Portfolio strategy discovery is a direct optimization rather than classification problem. However, several formal arguments presented in this section should clearly demonstrate that boosting for classification can be naturally generalized for the framework of boosting-based optimization as was suggested in our previous papers [vg].

One of the operational interpretations of the boosting algorithm shows that boosting fits an additive logistic regression model by a stage-wise optimization of expected exponential loss [Friedman]:

$$\mathrm{E}e^{-yF(x)} \to \min \tag{12}$$

One of the motivations for such choice of the loss function is that exponential loss is an upper bound on the classification error [Schapire]:

$$P[yF(x) < 0] < \mathrm{E}e^{-yF(x)} \tag{13}$$

This kind of upper bound can be generalized for any event *A*:

$$P[A] \le \mathrm{E}e^{-(2I_A - 1)|F(x)|}, \tag{14}$$

where $I_A=1$ if *A* takes place and $I_A=0$ otherwise.

For example, consider an event "return of a combined strategy *S* is lower than a threshold r_c", i.e. $r(S(x))<r_c$. Then inequality (14) can be written as

$$P[r\left(S(x)\right) < r_c] \le \mathrm{E}e^{-\left(2I_{[r(F(x))<r_c]}-1\right)|S(x)|} \tag{15}$$

Looking at the right part of inequality (15) we can see that optimization task can be reduced to classification task in a way proposed in [VG]. Thus, we get a generalization of boosting framework for optimization tasks, where instead of minimizing classification error we are trying to minimize a number of intervals where $[r(S(x))<r_c]$. Minimization of $P[r(S(x))<r_c]$ is equivalent to the maximization of $P[r(S(x))>=r_c]$.

Boosting is known to demonstrate stable generalization abilities measured by its performance on out-of-sample data. Standard Vapnik-Chernovenkis (VC) analysis of boosting bounds generalization loss by empirical risk (here $\hat{\mathrm{P}}\mathrm{r}[\bullet]$ denotes empirical probability measured on the training set) and a term dependent on VC-dimension of weak learner and number of boosting rounds [Freund]:

$$P[H(x) \ne y] \le \hat{\mathrm{P}}\mathrm{r}[H(x) \ne y] + \tilde{O}\left(\sqrt{\frac{Td}{m}}\right) \tag{16}$$

where *d* is the VC-dimension (capacity) of weak learner, *m* is the size of the training set, and *T* is the number of weak classifiers in the committee. However, this bound could often contradict with practical experience, which shows that boosting can decrease its test error even when empirical risk is already zero.

The best explanation of boosting generalization capabilities so far is margin theory [Schapire98]. Margin theory provides an upper generalization bound independent of number of iterations made by boosting. This bound suggests that boosting may not overfit even if ran for many rounds. For any θ:

$$P[H(x) \neq y] \leq \hat{\Pr}[\text{margin}(x, y) \leq \theta] + \tilde{O}\left(\sqrt{\frac{d}{m\theta^2}}\right), \tag{17}$$

where margin is

$$\text{margin}(x, y) = \frac{y\sum_t \alpha_t h_t(x)}{\sum_t \alpha_t}. \tag{18}$$

Margin is the measure of confidence of a boosted predictor in its decision. It is positive if the decision is correct and negative otherwise. Theory states that for better generalization one should increase the margins of training samples.

In the context of the trading strategy boosting margin can be reformulated as

$$\text{margin}(r_n^{\tau}) = \frac{\sum_t \alpha_t I\left(r_c - r_n^{\tau}\left(BS_t(p_t), S_t\right)\right)}{\sum_t \alpha_t} \tag{19}$$

In such interpretation we want to maximize the confidence of our boosted strategy for over each of *n* shifted intervals of length τ. In order to strictly apply theory for strategy boosting, some additional statements should be proven (e.g., finite capacity of base strategies – analogous to VC-dimension). Although the rigorous proof is out of scope of this paper, the usage of the low-complexity base strategies should not violate these assumptions in most practical settings.

Boosting is proven, under some conditions on a weak learner, to be a regularized search for a maximum margin solution [Rudin]. That means that in the limit, the boosted committee may have a maximum possible lower margin. Recent studies of boosting suggest that one may expect overfitting in case of increasing VC-dimension of weak learner through iterations [Reyzin], or in the case of the overlapping class distributions [Vezhnevets]. Again, usage of the low-complexity base strategies should be able to alleviate these problems in most practical applications.

5 Optimization framework based on input-dependent boosting: Discovery of the regime-adjusted portfolio strategies

Majority of the known boosting algorithms including classical AdaBoost assume that the combination weights are fixed constants and therefore do not take

particular input patterns into consideration. Application of such algorithms as a basis for the boosting-based optimization allows discovering portfolio strategies with fixed weights, i.e., fixed capital allocations among the individual strategies. This excludes any dynamic regime-adjustment to exploit regime-specific patterns for further increase of the portfolio strategy profitability.

However, several input-dependent boosting algorithms have been recently proposed [Jin, id2, id3]. One of these algorithms (called WeightBoost) [Jin] seems to be the most appealing practical choice due to its close relation to the original AdaBoost and flexibility to vary the form and degree of the regime adjustment. WeightBoost not only introduces input dependency but also provides practical regularization mechanism that helps to avoid overfitting problems that boosting may encounter in applications with high level of noise.

The original motivation of this input-dependent boosting algorithm [Jin] is to alleviate two limitations of the classical AdaBoost and its extensions. One is constant combination weights that prevent from using full potential of the base models in the regimes of their expertise. The other problem is potential overfitting in high-noise cases that requires introduction of the practical and efficient regularization. Previously proposed regularization methods can be summarized into two groups: changing the cost function and introducing soft margin [reg]. However, most of the regularization algorithms are unsuccessful either due to the high computational cost or lack of strong empirical results of improvement.

The following inductive form of the boosted classifier is used in AdaBoost derivation and its input-dependent extension given in [Jin]

$$H(x) = H_{T-1}(x) + \alpha_T h_T(x) \tag{20}$$

The potential overfitting problem of AdaBoost can be implied from the weight updating function (10). If there are some noisy data patterns that are difficult to classify correctly by the base classifier(s), the value of $-H_{T-1}(x_i)y_i$ for those data points will accumulate linearly (since $H(x)$ is a linear combination of base classifiers) and the corresponding weights will grow exponentially. Therefore, the particular sampling procedure within the AdaBoost algorithm will overemphasize the noisy training data points and may lead to poor generalization.

Since the potential overfitting is caused by the accumulation of errors within the function $H_{T-1}(x)$, one way to avoid this is to modify the expression form for $H_{T-1}(x)$. Instead of constant combination coefficients α_t, we can make them input dependent, i.e.

$$H_T(x) = \sum_{t=1}^{T} \alpha_t e^{-|\beta H_{t-1}(x)|} h_t(x) \tag{21}$$

Compared to (20), the above expression replaces the weighting constant α_t with $\alpha_t exp(-|\beta H_{t-1}(x)|)$. More interestingly, it can be shown that when α_t is bounded by some fixed constant, the value of $H_T(x)$ in (21) will increase at most logarithmically with respect to the number of iterations T and the weight of each data pattern will grow at most polynomial with the number of iterations.

Therefore, the problem of overemphasizing noisy data patterns will be alleviated substantially compared to the classical AdaBoost.

As pointed out before the other important problem with AdaBoost is fixed combination coefficients. According to AdaBoost, each base classifier (model) $h_t(x)$ is trained intentionally on the data patterns that either misclassified or weakly classified by previous classifiers (models) $H_{t-1}(x)$. Therefore, every base classifier $h_t(x)$ should be appropriate only for a subset of input patterns. However, in the prediction phase, the opinion of the base classifier $h_t(x)$ will always be weighted by the same number α_t no matter what test examples are.

On the contrary, in the new form of $H_T(x)$ given by (21), introduction of the instance-dependent factor $exp(-|\beta H_{t-1}(x)|)$ offers a tradeoff between the opinion of the base classifier $h_t(x)$ and that of the previously built meta-classifier $H_{t-1}(x)$. Since the value of $H_{t-1}(x)$ indicates its confidence on classifying the instance x, the factor $exp(-|\beta H_{t-1}(x)|)$ forces to consider the opinion of $h_t(x)$ seriously only when combination of the previous classifiers $H_{t-1}(x)$ is not confident about its decision. This implies that introduction of the input-dependent factor makes the base classifier $h_t(x)$ to be consistent between the training phase and the prediction phase, i.e., $h_t(x)$ is used for prediction of the particular type of input patterns that it has been trained on.

It is clear from (21) that the extent of the input-dependency can be conveniently controlled by factor β. When β goes to zero, we recover original AdaBoost algorithm. Varying β one can go from small weight modulations (mild regime adjustment) in almost stationary meta-model (small β) to the more regime-switching type of meta-model (large β). The optimal choice of β is determined by properties that user expects from the meta-model and stability of its out-of-sample performance.

Finally, one needs to obtain a learning procedure that is able to minimize the exponential cost function with the new combination form given by (21). As shown by Jin et al [..], following standard procedures for AdaBoost derivation, one obtains the new expression for the data weight modification at each boosting iteration:

$$w_n^T = \frac{e^{-H_{T-1}(x_n)y_n - |\beta H_{T-1}(x_n)|}}{\sum_{j=1}^{N} e^{-H_{T-1}(x_j)y_j - |\beta H_{T-1}(x_j)|}} \qquad (22)$$

All other relevant expressions given by (8.1)-(8.5) remain the same. Using similar arguments one can obtain learning procedure for an arbitrary bounded regularizer $f(x)$ instead of $exp(-|\beta H_{t-1}(x)|)$ used above [Jin].

How to adapt to boosting-based optimization. Unlike classifier, we do not know the base strategies performance to compute regime-adjusted weights according to (..). However, we need the weights to run the strategy portfolio. One of the natural ways to arrange this …. .

Operational challenges for daily or other low-frequency trading strategies. Nevertheless still achievable. The most simple operations with intraday strategies without overnight positions, i.e., … . In the following section we will give a real market example of such strategy.

6 Application example

7 Conclusions

References

[1] Fabbozzi
[2]