American Journal of Artificial Intelligence
2022; 6(2): 36-47
http://www.sciencepublishinggroup.com/j/ajai
doi: 10.11648/j.ajai.20220602.12
ISSN: 2639-9717 (Print); ISSN: 2639-9733 (Online)

Dominance Pruning in Machine Learning for Solving Financial Trading and Real-Time Multimedia Applications

Benjamin Wan-Sang Wah

Department of Computer Science and Engineering, The Chinese University of Hong Kong, Shatin, Hong Kong

Email address:
bwah@cuhk.edu.hk

To cite this article:
Benjamin Wan-Sang Wah. Dominance Pruning in Machine Learning for Solving Financial Trading and Real-Time Multimedia Applications. *American Journal of Artificial Intelligence*. Vol. 6, No. 2, 2022, pp. 36-47. doi: 10.11648/j.ajai.20220602.12

Received: December 28, 2022; **Accepted:** January 16, 2023; **Published:** February 6, 2023

Abstract: This paper presents the design of dominance relations to reduce the space traversed in machine learning for solving two applications in financial trading and real-time multimedia. A machine-learning algorithm designed for an application with a huge search space will need to perform an efficient traversal of the space during learning. It will be more effective if it employs a powerful pruning mechanism to eliminate suboptimal candidates before using them in the learning algorithm. In our approach, we present dominance relations for pruning subspaces with suboptimal kernels that are otherwise evaluated in learning, where kernels represent the statistical quality, average density, or probability of solutions in a subspace. Specifically, when one subspace dominates another by a dominance relation, we can prune the latter and guarantee without searching both that the kernel of the latter cannot be better than that of the first. As a result, a significant portion of the search space will be pruned by those non-dominated subspaces during learning. In the financial trading application studied, we use mean reversion as our strategy for learning the set of promising stocks and Pareto-optimality as our dominance relation to reduce the space evaluated in learning. In the multimedia application, we propose a dominance relation using an axiom from our past work to approximate the subspace of perceptual qualities within an error threshold. The pruning mechanism allows the learning of the mapping from controls to perceptual qualities while eliminating the evaluation of all those mappings that are within the error thresholds. In both cases, we can harness the complexity of machine learning by reducing the candidate space evaluated.

Keywords: Kernels, Dominance Relations, Machine Learning, Financial Trading, Mean Reversion, Real-time Multimedia, Perceptual Quality

1. Introduction

With the pervasive availability of the Internet, applications involving large quantities of data and complex optimizations have become increasingly popular. In general, there is a lack of standard algorithms, infrastructures, and theories for solving these applications. We have identified three unique properties of these applications.

Property 1.1. The applications' data space is so vast and their optimizations so complex that it is infeasible to scan the data once and find either all or the optimal solutions. Hence, we are interested in looking for good but not necessarily optimal results that satisfy the application requirements.

Property 1.2. We characterize the search space by some incomplete and heuristic problem-dependent attributes. These are heuristics because the multidimensional data space may be unbounded, ill-defined, or non-smooth, making it difficult to characterize all its attributes. Hence, efficient algorithms will need to exploit domain-specific properties to effectively traverse the space.

Property 1.3. The solutions satisfying the application requirements have distributions that are likely non-uniform and non-IID (*independent and identically distributed*). Moreover, they may be non-stationary over time, making it difficult to use automated methods to acquire their distributions. In general, statistical techniques based on some uniform models may not generalize well.

These properties lead to the understanding that such applications have diverse domain-dependent characteristics, possibly non-stationary statistical properties, and have high complexities to store, traverse, and process. Hence, good

algorithms to look for solutions will need to identify promising and manageable subspaces before drilling down into each.

As an illustration of the properties, consider a financial trading application studied in Section 3. Here, the data available each day in the stock market is enormous because many stocks are traded, each having a lot of past historical information. According to the efficient market hypothesis [1], stock prices do not have a well-defined statistical model, as they evolve in a random walk. Even in non-random-walk cases, their behavior can be highly dynamic and non-stationary and may depend on ill-defined exogenous factors, such as market conditions and political or social events.

The second application studied in Section 4 on the perceptual quality of real-time multimedia is also complex and data-intensive. The space of perceptual quality as a function of controls is vast, ill-defined, and unbounded, making it difficult to find good operating points that optimize perceptual quality.

Figure 1 shows a data space B for an application with a solution set $\sum_B$. A solver A_B on B can determine whether $i \in \sum_B$. Because B is so large or ill-defined that its space cannot be enumerated, we resort to a dimension-reduced subspace $S \subseteq B$ with a solution set $\sum_S \subseteq \sum_B$.

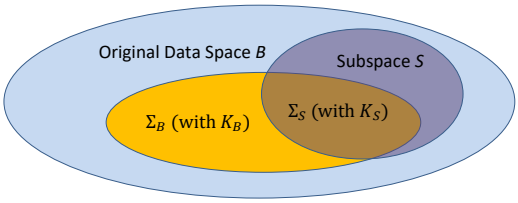


Figure 1. *The solver A_S looks for solutions in a dimension-reduced subspace S with kernel K_S.*

The original solver A_B may need to be adapted into a subspace application solver A_S to work on S. We define A_S using some application-specific attributes $\overline{x_S} = x_1, \ldots, x_q$ (Property 1.2) and their values $\overline{c_S} = c_1, \ldots, c_q$. Let F_S be a function that maps $\overline{x_S}$ with values $\overline{c_S}$ to the solver A_S:

$$A_S = F_S(\overline{x_S} \mid \overline{c_S}). \quad (1)$$

We further define the *kernel* K_S of subspace S to be some aggregate statistical quantity, average density, or probability of solutions in S, where each point in S has one or more quality metrics. Using the kernel, an application solver can prioritize points in the subspace for evaluation or prune suboptimal ones.

In general, machine-learning algorithms are built on an application's search space and may require an efficient abstraction of the space before learning is performed. Alternatively, learning may rely on some domain-specific strategies to traverse the search space during learning. In either case, the quality of learning will depend on how efficient the space can be traversed. In the literature, pruning in machine learning refers to removing redundant or the least important parts of a model or search space [2]. There were numerous studies on pruning learning models (such as neural networks [3] and decision trees [4]) to prevent overfitting and reduce computational complexity. On the other hand, pruning the search space is more general and is not necessarily tied to machine learning. We study in Section 2 some effective strategies for reducing the search space.

In this paper, we focus on pruning an application's search space during learning to improve its effectiveness. By identifying dominated subspaces, we can conduct machine learning much more efficiently in finding good solutions. Our goal is to develop new dominance relations for pruning suboptimal subspaces during learning on the two aforementioned applications.

Referring to the stock trading application studied (see Table 1), the data space B consists of the historical prices of all the stocks traded, and the solution set $\sum_B$ is the set of stocks with positive returns after some delay. To reduce B, we define subspace S in which stocks are traded daily, and $\sum_S$ are those stocks with a positive return on the following day. In Section 3, we define A_S to select stocks using mean-reversion (MR) in which we restrict $\overline{x_S}$ in Eq. (1) to be the thresholds on returns at the end-of-day (EOD) and $\overline{c_S}$ to be the set of threshold values learned for each trading day. We further reduce the large space of MR thresholds evaluated in learning using Pareto optimality as our dominance relation.

Table 1. *The Attributes of the Two Applications Studied.*

Application Objective	Domain Controls	Kernel Metrics	Sub-space S	Dominance Relation	Domain Knowledge
		Financial Trading Application			
Max. avg. return over horizon	$(d_L, d_U]$, $d_L < d_U \leq 0$	Annual ret., downside deviation	Daily traded stocks	Domain-indep. Pareto optimality	Mean reversion
		Real-time Multimedia Application			
Optimal perceptual quality	End-to-end delay	Awareness as function of controls	Subspace on awareness	Monotonicity of awareness to controls	Axiom on perceptual quality

In the multimedia application studied (see Table 1), B is the huge data space for mapping past perceptual qualities to run-time controls. This space is ill-defined because perceptual quality is an abstract concept. To reduce B, we consider perceptual qualities measured by awareness [5] in subjective tests that are conducted offline over a past window of time. However, the subspace S is still unmanageable because the space of past scenarios is infinite in size. To this end,

we employ an axiom from our past work as a dominance relation to prune a majority of suboptimal past scenarios and to interpolate S through a few subjective tests. This approach allows S to be fully specified, making it feasible to learn from the reduced subspace good operating points at run time.

This paper is divided into five sections. We classify in Section 2 three approaches for finding effective subspaces according to the data distribution supported, pruning methods used, and data transformations employed. In Section 3, we use Pareto optimality as a domain-independent dominance relation when learning mean-reversion parameter sets according to their expected annual return and downside deviation. In Section 4, we present a dominance relation that helps prune the data subspace of perceptual quality in a multimedia application to within a prescribed error threshold. (See the summary in Table 1). Finally, conclusions are drawn in Section 5.

2. Finding Effective Search Subspaces

We survey in this section three general approaches for identifying effective subspaces and their kernels. By reducing the space traversed, we can greatly improve the performance when solving the application problem by machine learning. Interested readers should refer to a more complete article published on this topic [6].

2.1. Structural Analysis to Discover New Structures

The structural analysis aims to discover new structures in existing data that allows a more efficient traversal of the search. It achieves this goal by evaluating the underlying data structure to either identify some existing embedded relations or discover a new structure. The process is heuristic as it depends on the domain knowledge of the data relations. We classify these techniques developed for structured, semi-structured, and unstructured data.

a) Structured data consists of data with some embedded relationships, such as relational datasets, community networks, and knowledge maps. The structural analysis uses a combination of observations and hypotheses to discover those structures that help find solutions faster.

An *observation-driven approach* starts from the application's goal that defines the kernel metrics. It then relies on user-guided statistical analysis to transform the original data into alternative structures and to select the target structure with the best kernel metrics. An example is in the credit allocation in scientific publications [7] that uses the credit on each citing article and co-cited ones to discover a new structure relating Nobel laureates to the prize-winning papers. The new structure leads to an efficient credit-allocation algorithm.

A *hypothesis-driven approach* uses the application's goal and informal observations to develop a hypothesis, propose a model, and verify the model experimentally or analytically. An example is in hypothesizing the complex relations among humans in a society. The Small-World Problem [8] starts from a hypothesis and some initial observations on any two persons in the world knowing each other with a probability distribution and finds a new small-world model.

b) Semi-structured data consists of data entities with relations in one or more dimensions but none otherwise. Examples include social-media journals, blogs, Twitter, and financial news that evolve but have non-apparent structures otherwise. An example of the approach is the prediction of the probabilities of propagating news from one person to another in an evolving social network [9]. By scaling the temporal dimension using a power-law relationship between the propagation probability and the duration since the last interaction, the study transforms the data stream into a uniform sequence over time with low prediction errors.

c) Unstructured data consists of entities without any uniform relations. An example of the approach is in the attribute extraction of web-based images that transforms the original high-dimensional feature space into a lower-dimensional hypersphere space by preserving the images' various structures and relevance relationships [10]. The result is an improved retrieval of relevant images from the database.

2.2. Pruning Subspaces with Suboptimal Kernels

The idea is to use general or domain-specific properties to reduce the search complexity by eliminating candidate points in a subspace. We consider two general classes here: heuristic and dominance pruning.

a) Heuristic pruning eliminates candidate solution points using heuristic properties on data relations and distributions. It uses partial, incomplete, or experiential information to eliminate candidates without relying on a formal analytical foundation. As a result, it is often evaluated on benchmarks, and its performance is not guaranteed on unseen data. An example application of the approach is on the burst-topic discovery in microblogs (such as Twitter) that aims to find high-quality topics in streaming data [11]. Only heuristic methods have been developed for filtering burst topics from non-burst ones because blogs are typically short, diverse, and noisy posts, with some common but unrelated information.

b) Dominance pruning use a dominance relation [12] between two subspaces S_i and S_j to prune S_j. It is a formal property because with the available information on S_i and S_j and without evaluating all their data points, one can prove that the kernel K_j of S_j cannot be better than K_i. The dominating kernels found are, therefore, guaranteed to be better. We illustrate the use of dominance relations on two applications in Sections 3 and 4, respectively, to prune their search spaces before applying machine learning.

2.3. Projecting Data into New Dimensions

This approach transforms the original data space into a different dimension, using heuristic and domain-specific methods. There are generally three transformations: spatial, temporal, or a combination of space and time.

a) Spatial transformation entails projecting the original data space into a more structured spatial organization. Since the

number of projections is extremely large or infinitely many, the choice is often driven by domain-specific heuristics. The approach is illustrated in the detection of users' communities in an Internet social network [13] by clustering the spatial dimension based on the degree of overlaps in the communities.

b) Temporal transformation entails the projection of data into a new temporal scale that reorganizes the data over time with a better structure. An example is in the prediction of information propagation from one person to another in an evolving social network [9] (Section 2.1).

c) Space-time transformation involves projecting the data space along the temporal and spatial dimensions to find an embedded structure. An example is in the persistent data-sketching space-time transformation [14] that holds a sliding window over space and time to approximate the entire data set in streaming data and that allows a large fraction of all the past events to be queried.

Summary. In this section, we have examined three general approaches for transforming the original data space into a different form to allow more efficient traversals. The success of these approaches requires domain-specific information and an in-depth understanding of the application aided by observations, hypotheses, and experimentation. In the next two sections, we apply dominance relations to prune the search space of two applications before applying machine learning. Without pruning, it will be inefficient for machine learning to find good strategies and solutions.

3. Dominance Pruning in Learning (Financial Trading Application)

This Section studies a financial trading application for maximizing the average next-day return on daily traded stocks. As is discussed in Section 1, we reduce the original data space B to a more manageable subspace S in which stocks are traded daily. We first discuss the kernel metrics used and the Pareto optimality as a domain-independent dominance relation for pruning candidates with suboptimal kernels in learning. We present an application solver A_S based on the mean-reversion method that learns the best MR parameters from past historical data to define the stocks for trading each day. Lastly, we show some experimental results on using the approach.

3.1. Kernel Metrics

We define the kernel metrics at a given time to be the projected annual return (*AR*) of n stocks over an average of 252 trading days per year and its downside deviation (*DD*) with a minimum acceptable return b (chosen as zero) [15]:

$$AR = \prod_{i=1}^{n}(1+r_i)^{252/n} - 1, \tag{2}$$

$$DD = \sqrt{\frac{\sum_{i=1}^{n}\min(0, r_i - b)^2}{n-1}}, \tag{3}$$

where r_i is the return of stock i. To allow these metrics to be computed, we assume that the returns in a subspace are independent and identically distributed (IID), which is generally true. Note that *DD* measures the unbiased standard deviation of only the negative returns.

The kernel metrics above represent tradeoffs on the quality of the stocks selected. To compare two alternative sets, we would like the returns of the set selected to be high (captured in *AR*); at the same time, we would like the likelihood of the set to have negative returns to be small (captured in *DD*). Hence, this is a multi-objective problem with two counteracting objectives.

3.2. Pareto Optimality

Pareto optimality is a domain-independent dominance relation [16] for prioritizing subspaces evaluated by two or more metrics in a multi-objective optimization of allocating resources. Pareto optimality or Pareto efficiency is reached when it is impossible to reallocate the resources to make any objectives better without making another worse off [17]. A Pareto frontier is a set of efficient solutions that do not dominate each other.

Using the kernel metrics of *AR* and *DD*, we can define a dominance relation between two sets of stocks S_1 and S_2. Specifically, when S_1 *dominates or is equal to* S_2 ($S_1 \succeq S_2$), it implies that S_1 has the same or a better *AR* ($AR_{S_1} \geq AR_{S_2}$) and the same or a smaller *DD* ($DD_{S_1} \leq DD_{S_2}$) than those of S_2. That is:

$$S_1 \succeq S_2 \iff (AR_{S_1} \geq AR_{S_2}) \wedge (DD_{S_1} \leq DD_{S_2}). \tag{4}$$

We can also define the case when S_1 and S_2 are non-dominated to each other:

$$\begin{aligned} S_1 \neq S_2 &\iff [(S_1 \not\succeq S_2) \wedge (S_1 \not\preceq S_2)] \\ &\iff [(AR_{S_1} < AR_{S_2}) \wedge (DD_{S_1} < DD_{S_2})] \\ &\quad \vee [(AR_{S_1} > AR_{S_2}) \wedge (DD_{S_1} > DD_{S_2})]. \end{aligned} \tag{5}$$

It should be clear that Eq's (4)-(5) satisfy transitivity but not commutativity or reflexivity. The dominance relation defined is domain-independent because it only relies on the kernel metrics.

We are interested in looking for a set $\mathcal{P} = \{S_i \,|\, i = 1, ...\}$ with tradeoffs between AR_{S_i} and DD_{S_i} in such a way that element $i \in \mathcal{P}$ with AR_{S_i} and DD_{S_i} is not dominated by element $j \notin \mathcal{P}$. By pairwise comparison of alternatives with known kernels, we use Eq. (4) to eliminate those dominated sets. The remaining non-dominated sets $\mathcal{P}$ lie on a Pareto optimal frontier of the 2-D region spanned by *AR* and *DD*. Here, $\mathcal{P}$ and $\overline{\mathcal{P}}$, the set of dominating and dominated sets, respectively, satisfy two properties.

$$\begin{aligned} &\mathcal{P} \text{ and } \overline{\mathcal{P}} \text{ satisfy (a) } \forall\, i, j \in \mathcal{P}, S_i \neq S_j, \\ &\qquad\qquad \text{(b) } \forall\, k \in \overline{\mathcal{P}}, \exists\, i \in \mathcal{P} \text{ such that } S_i \succeq S_k. \end{aligned} \tag{6}$$

Intuitively, each point on the Pareto frontier has a larger or equal *AR* and a smaller or equal *DD* when compared to at

least one point that is not on the frontier. Further, those on the frontier do not dominate each other; that is, each has either smaller *AR* and *DD* or larger *AR* and *DD*, when compared to another point on the frontier. We are interested in those points on the frontier because each is not better than another as far as *AR* and *DD* are concerned. Moreover, those not on the frontier are inferior to at least some Pareto-optimal points.

Transitivity in Pareto optimality is crucial because it allows the pruning of all suboptimal points, including unsearched ones, when limited by the Pareto frontier. Note that the frontier found by searching a subset of the alternatives is not the true frontier of the application because we have not exhausted all the possibilities. However, the resulting boundary will approach the true frontier when sampling more points. The advantage of this approach over heuristic pruning is that it guarantees that the pruned alternatives have suboptimal kernels, whereas heuristic pruning does not have the transitivity to have such guarantees.

One of the popular approaches for finding Pareto frontiers is scalarizing methods [18] that assign weights on objectives to convert a multi-objective problem into a single-objective one. Other methods include lexicographic ordering, evolutionary multi-objective optimization, no-preference schemes, and interactive methods. In our application, we apply a guided generate-and-test approach to enumerate multiple combinations in finding a Pareto frontier.

3.3. Mean Reversion on Stock Returns

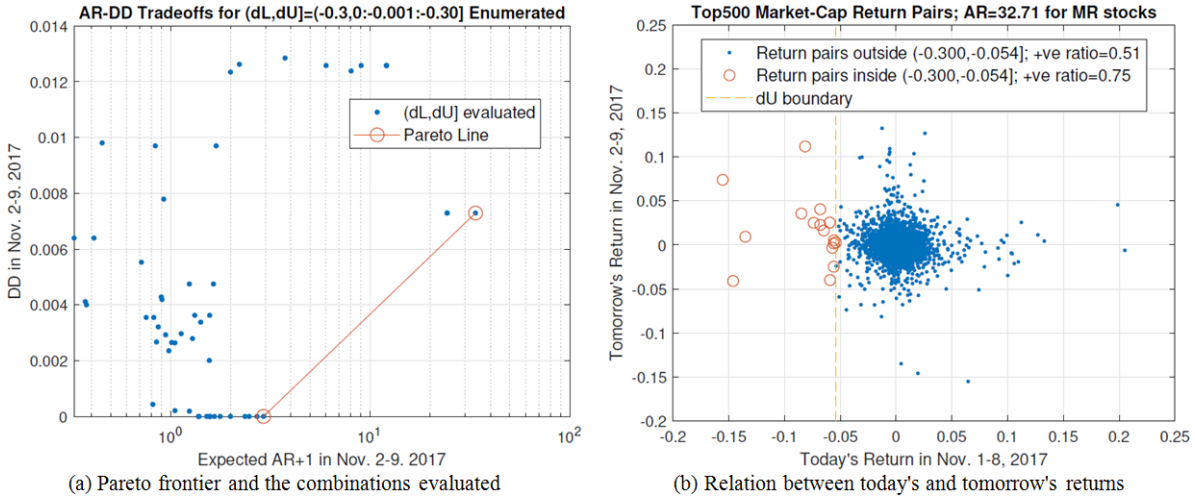
We first illustrate the idea of mean reversion before showing our algorithm and results. Consider a financial trading strategy that looks for a portfolio of stocks on a given day using two thresholds $(d_L, d_U]$ from 500 NYSE and NASDAQ stocks with top market caps traded at the end-of-day (EOD) of each day over a past period T. (We use the aggregate return over T to mitigate the day-to-day fluctuations.) The thresholds define a portfolio at the EOD of day $t \in T$ to include stock i when its return $r_t^i \in (d_L, d_U]$, where $r_t^i = \frac{p_t^i}{p_{t-1}^i} - 1$ and p_t^i is the price of stock i at the EOD of day t.

Figure 2a plots the relation between r_{t+1}^i and r_t^i, $i \in \{1, ..., 500\}$, at the EOD of $t \in T = \{$1-9 Nov. 2017$\}$. (Note that we have six trading days in T and an extra day to compute the next-day return of the last day traded.) Using $[d_L, d_U] = (-0.3, -0.054]$, the fraction of stocks with positive next-day returns is 0.75. That is, stocks whose returns are down by 5.4% or more in one EOD during the six trading days have a 75% chance of positive gain in the following EOD. The aggregate return of the 500 stocks is 8.81% over the period and a projected *AR* of 32.71 times. On the other hand, stocks in the set bounded by $[d_L, d_U] = (-0.054, 1]$ have only 51% of positive returns over T.

Figure 2b further plots the (AR, DD) pairs of the 301 combinations enumerated using $d_L = -0.3$ and $d_U = 0 : -0.001 : -0.03$. The power of dominance pruning is apparent as the infinitely many points to the left of the Pareto frontier can be eliminated without evaluation during learning.

The search of $(d_L, d_U]$ over a past period allows us to learn the best combination and to generalize it to the next day. In the next section, we apply this iterative approach to learn the best $(d_L, d_U]$ on a given day to decide on the stocks to be traded on the following day, using dominance pruning to significantly reduce the number of candidates evaluated in learning.

The idea of choosing $(d_L, d_U]$ in Figure 2 is due to *mean reversion* (MR) [19] in stock trading. Jegadeesh first observed this phenomenon, which shows that if a stock performs worse than others on one trading day, it tends to do better on the following trading day. The regression toward the mean behavior [20] is reasonable because returns do not go down forever and will eventually recover. Note that the use of two parameters d_L, d_U in our example is not unique, and there are many ways of realizing the general mean-reversion method.



(a) Pareto frontier and the combinations evaluated

(b) Relation between today's and tomorrow's returns

Figure 2. *An illustration of mean reversion and the Pareto frontier defined by* $(d_L, d_U] = (-0.3, -0.054]$. *(a) The graph shows the Pareto frontier obtained by enumerating the 301 combinations of* $d_L = -0.3$ *and* $d_U = 0 : -0.001 : -0.03$. *Note that the infinitely many points to the left of the frontier can be pruned without evaluation in learning. (b) Each point in the graph represents the next-day return to today's return of the 500 stocks traded on Nov. 1-9, 2017. The portfolio defined by* $(d_L, d_U] = (-0.3, -0.054]$ *has 75% of positive next-day returns.*

3.4. Learning Algorithm and Implementation

Our learning algorithm is a guided generate-and-test that learns the MR parameters in two levels: the macro-level and the micro-level. Figure 3 shows the learning process, and Figure 4, the pseudo-code.

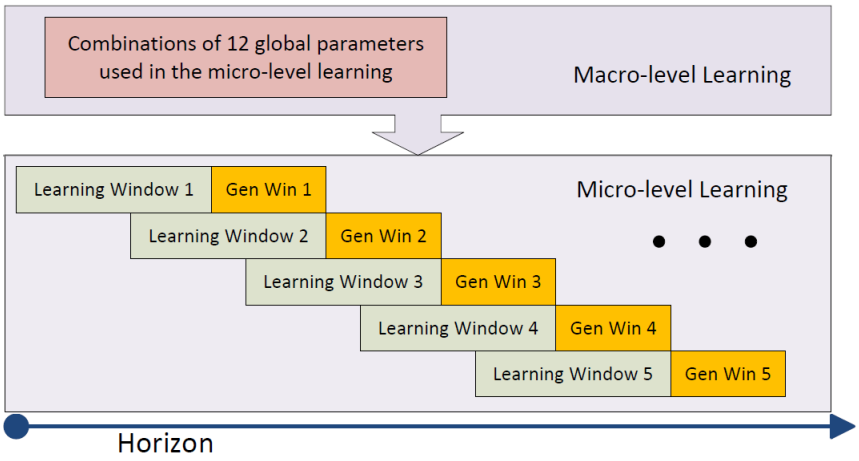


Figure 3. *The macro-level and micro-level learning process over the horizon.*

```
/* macro-level learning enumerates combinations of global parameters and find the pareto optimal sets, using */
 /* the mean-risk results reported by the micro-level over the horizon. */
1.  function paretoPoint = macroLearn (H_L, SL)                 /* input learning horizon H_L and a list of stocks SL*/
2.      initialize D_L, D_U, L, NP_min, NP_max;                 /* initialize D_L, D_U, L, NP_min, NP_max  in the enumerations */
3.      for LW ∈ LW_size, GW ∈ GW_size                          /* learning/generalization window combinations evaluated */
4.          for NS ∈ NS_th, NT ∈ NT_th                          /* LB threshold on #stocks and #effective trading days */
5.              for DD ∈ DD_th                                  /*DD_th downside deviation thresholds evaluated */
6.                {meanRiskPts} = microLearn(LW, GW, NS, NT, DD, NP_min, NP_max, D_L, D_U, L, H_L, SL);
7.              end
8.          end
9.      end
10.     paretoPoint = findParetoPoint ({meanRiskPts});          /* find Pareto optimal points to use in H_L */
11. end

    /* micro-level learning learns the MR thresholds in LW and generalizes them to GW over the horizon, using the 12 */
    /* global parameters supplied by the macro-level; it reports the mean-risk results of the horizon to the macro-level. */
12. function meanRiskPts = microLearn(LW, GW, NS, NT, DD, NP_min, NP_max, D_L, D_U, L, H_L, SL)
    /* given each LW over the horizon, find the best d_L, d_U, ℓ combination for each ω and generalize to the next GW */
13.   for ω ∈ {1, … , W}                                        /* enumerate ω over W windows of size LW in H_L */
14.     acceptabled_L d_U ℓ = learnMR(ω, NS, NT, DD, NP_min, NP_max, D_L, D_U, L, SL);
15.     apply acceptabled_L d_U ℓ to each day in GW with #stocks ∈ [NP_min, …, NP_max] in each day of GW;
16.   end
17.   meanRiskPts = (m_r^{H_L}, σ_r^{H_L});                    /* compute AR and DD of daily returns in GW over H_L */
18. end

19. function acceptabled_L d_U ℓ = learnMR(ω, NS, NT, DD, NP_min, NP_max, D_L, D_U, L, SL)
    /* over all days in an LW,  ω, find the best d_L, d_U, ℓ combination and return acceptable list of d_L, d_U, ℓ combinations */
20.     for d_L ∈ D_L, d_U ∈ D_U, ℓ ∈ L                          /* MR thresholds d_L, d_U, ℓ to use in  ω */
21.       for j ∈ ω and across all stocks k ∈ SL                 /* enumerate all stocks k over day j in LW ω */
22.         if r^ω_{j-1:j-L,k} ∈ (d_L, d_U], record r^ω_{j,k}; end   /* record next-day return if MR of stock k happens in day j-1 */
23.         evaluate r_j^ω and n_j^ω for j ∈ ω ;                /* mean next-day return over k stocks and #stocks for day j */
24.       end
25.       eval m_r^ω (AR) and σ_r^ω (DD) for ω with trading days ≥ NT (with n_j^ω ≥ NS in each);
26.       save (d_L, d_U, ℓ) in acceptabled_L d_U ℓ list if it has large m_r^ω and σ_r^ω < DD and #stocks are in range NP_min, NP_max
27.     end
28. end
```

Global Parameters and Sample Ranges of Values Evaluated
D_L: $[0:-0.001:-0.05, -0.055:-0.005:-0.15, -0.16:-0.01:-0.2]$
D_U: $[0:-0.001:-0.05, -0.055:-0.005:-0.15, -0.16:-0.02:-0.4]$
L: $1:5$; NP: $[2, 10]$; LW_{size}: 20; GW_{size}: $[5, 10]$;
NS_{th}: 2; NT_{th}: 9; DD_{th}: 0.014; H_L: 4000

Figure 4. *The pseudo-code of our algorithm has three routines: (a)* macroLearn *for learning a set of global control parameters, based on the range of values in the table; (b)* learnMR *for learning the best combination of* $[d_L, d_U, \ell]$ *that satisfies the MR condition in a learning window (LW), and (c)* microLearn *for applying the best combination of MR parameters learned to the next generalization window (GW) and evaluating the mean-risk results of all the GWs over the horizon.*

The macro-level algorithm is used to generate and prioritize the combinations of 12 global parameters that are common across the horizon. Examples of these parameters include the range of the number of stocks traded each day and the durations of the learning window (*LW*) and the generalization window (*GW*). Using the global parameters supplied by the macro-level, the micro-level learning algorithm learns the MR parameters in each *LW* to decide on the stocks traded in the next *GW*. After repeating the process over all *GW*s, it reports the aggregate *AR* and *DD* found for the horizon to the macro-level. Based on the results of all the combinations, the macro-level applies Pareto optimality to prune a majority of the combinations, including those not evaluated. It then selects the best combination of the global parameters for the horizon.

The macro-level is implemented by the routine *macroLearn* (Lines 1-11) with two arguments (horizon H_L and stock list SL). It enumerates three sets of parameters: (a) LW_{size} (learning window size), (b) GW_{size} (generalization window size), and (c) thresholds for deciding whether to generalize the parameters found in a *learning window* (*LW*) to the immediate next *generalization window* (*GW*). The latter set includes NS_{th} (lower bound on the number of stocks traded in a day in *LW*); NT_{th} (lower bound on the number of days in *LW* with stocks traded); and DD_{th} (upper-bound *DD* allowed in *LW*). The table in Figure 4 shows a sample of the range of values used in our implementation. The ranges are either user-specified or learned through experimentation.

The micro-level learning algorithm starts by using a set of 12 global parameters supplied by the macro level. It learns the best MR parameters in each *LW* before generalizing them to the immediate next *GW*. In each *LW*, it evaluates the next-day returns of various candidate sets of stocks selected by the MR thresholds $(d_L, d_U]$, and computes *AR* and *DD* for each combination of d_L, d_U, ℓ, where ℓ is the window for smoothing daily return fluctuations over the past ℓ days. It then applies the Pareto optimality to prune suboptimal MR thresholds, including those not evaluated. Since these combinations generally range from high *AR* with high *DD* to low *AR* with low *DD*, we constrain the maximum *DD* tolerated by DD_{th} and pick the one with the highest *AR*. The result is a (d_L, d_U, ℓ) combination that will be used for selecting stocks for trading in the following *GW*. We repeat the learning and generalization process over the horizon by staggering the *LW*s and *GW*s and report the *AR* and *DD* obtained over all the *GW*s in the horizon to the macro level. Note that the kernel metrics in an *LW* have statistical stationarity in some temporal localities. This observation forms the basis of generalizing the MR parameters learned in one *LW* to the following *GW*.

The micro-level is implemented by two routines. The first routine *microLearn* (Lines 12-18) is called by *macroLearn*. It evaluates the MR parameters for all the *LW*s over the horizon one at a time and generalizes the parameters to the following *GW*. The process is iterative (Lines 13-16): in each *LW*, it calls *learnMR* to find the best d_L, d_U, ℓ for this *LW*, generalizes the combination to the next *GW*, and computes the corresponding return. The routine then iterates over all *LW*s and *GW*s in the horizon and returns the *AR* and *DD* of the aggregate sequence of *GW*s evaluated (Line 17).

The second micro-level routine *learnMR* (Lines 19-28) is called by *microLearn*. It enumerates all the parameters for each *LW* and records the next-day return of stock $k \in SL$ when MR happens (Line 22). The MR condition occurs when the average smoothed return over the past $\ell \in L$ days is in $(d_L, d_U]$ and the number of stocks each day is in the range $[NP_{\text{min}}, NP_{\text{max}}]$. We then record *AR* ($m_r^\omega$) and *DD* ($\sigma_r^\omega$) when the number of significant trading days (each with at least NS_{th} stocks) in ω is greater than or equal to NT_{th}. The routine returns to *microLearn* $[d_L, d_U, \ell]$ of all *LW*s with high *AR* and with *DD* smaller than some upper bound *DD*.

3.5. Experimental Results

In this section, we report the best set of simulation results using a list of 1000 NYSE and NASDAQ stocks with top market caps on 8/14/2020 and a horizon of 4,000 days (9/27/2004 to 8/14/2020). We assume that trading would be done on Interactive Brokers Group (IBKR Lite) that charges no commission on trading US exchange-listed stocks [21].

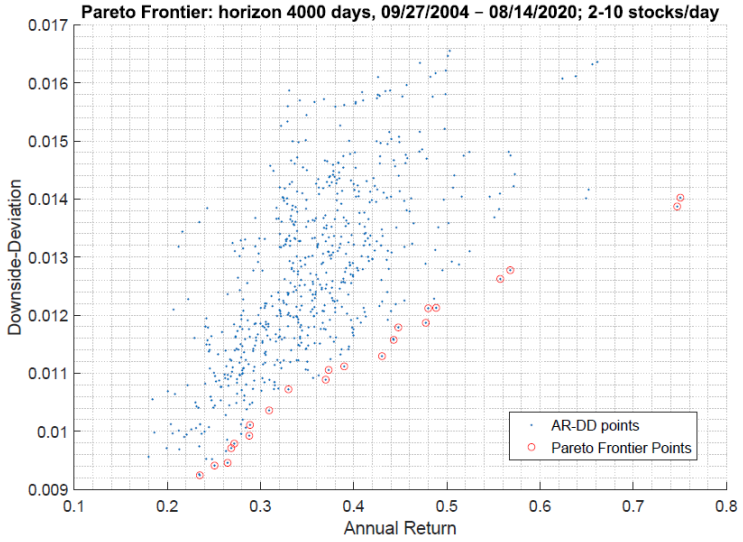
Figure 5a plots the (AR, DD) of evaluating 640 sets of the 12 control parameters generated by *macroLearn* over the 4000-day horizon H_L ending at 14 August 2020. (In other experiments, we evaluated 10880 combinations of the control parameters but the improvement is marginal.) Each point corresponds to a complete run of *microLearn* over the horizon using a given combination of parameters. We set the number of stocks traded each day in *GW* to $[NP_{\text{min}}, NP_{\text{max}}] = [2, 10]$, $[2, 15]$, and $[2, 20]$, respectively. Note that all those points to the left of the Pareto frontier are pruned.

Figure 5b plots the cumulative returns found in the sequence of *GW*s over the horizon based on the Pareto point with the highest *AR*. For example, the *AR* and *DD* of 4,000 days for $d_L d_U \ell(a)$ are 0.75 and 0.1402, respectively. For comparison, we show B. Li *et al.*'s PAMR result [22] ran with default parameters with a similar *AR* but a higher *DD* of 0.197.

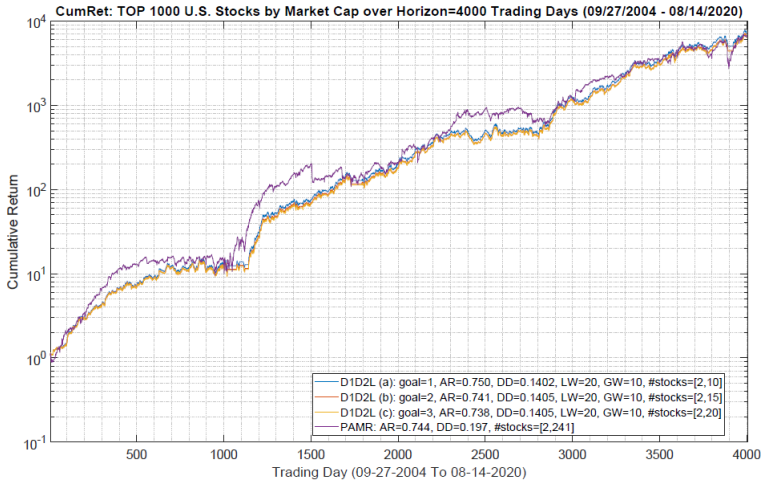
Note that the best parameters learned can generalize well to the future. The reason is that they were learned using a 4000-day horizon, and a small extension of the horizon would not change its optimality. The parameters may need to be relearned when the horizon is further extended.

Figure 5c compares the number of stocks traded each day (*NSPD*). Our $d_1 d_2 \ell$(a) strategy limits *NSPD* between 2 and 10, whereas Li's method does not have such a limit and has a much higher maximum *NSPD* of 241. Note that a large *NSPD* may be impractical in daily investments with limited capital.

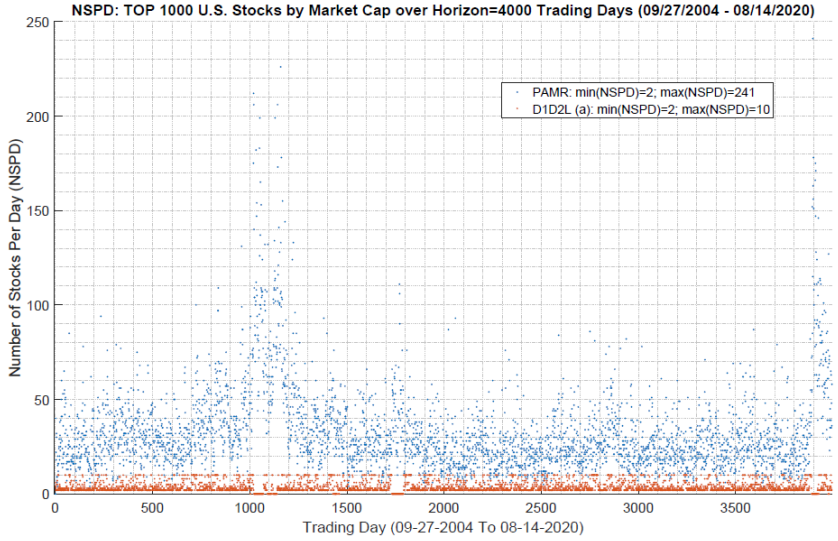
We have shown in this section the merit of using the domain-independent Pareto optimality for pruning suboptimal parameter sets during learning. When combined with the domain-dependent mean-reversion strategy for defining portfolios, our simulation results over 21 years lead to an average annual return of 175% with low downside deviation and a small number of stocks traded each day.



(a) *AR-DD* tradeoffs of the global control parameters



(b) Cumulative returns of the Pareto point in (a) with the highest *AR*



(c) Number of stocks traded each day

Figure 5. *Experimental results. (a) Scatter plot showing the AR-DD tradeoffs on the 640 sets of 12 global control parameters enumerated by* macroLearn *in Figure 4. The Pareto frontier points are those whose AR and DD are not dominated. (b) Cumulative returns of our proposed strategy and the parameters used in obtaining the results. The best AR is 0.75 with DD of 0.1402. The result of PAMR [22] shows a similar AR but a higher DD of 0.197. (c) A comparison of the number of stocks traded each day (NSPD) over the 4000-day horizon. Our* $d_1 d_2 \ell(a)$ *strategy has a maximum NSPD=10, whereas PAMR has a maximum NSPD=241.*

4. Dominance Pruning in Learning (Real-Time Multimedia Application)

Our goal in this section is to present a dominance relation for reducing the complexity of learning the relationship between the controls used in a multimedia application and the quality perceived by users. As is discussed in Section 1, we propose to reduce the original data space B to a subspace S that uses awareness as a perceptual-quality measure. However, S may involve infinitely many past scenarios, each requiring expensive offline subjective tests (on the order of minutes). To this end, we propose a dominance relation in Section 4.2, using an axiom from our past work, to approximate S to within an error threshold after a small number of subjective tests. The result is a significant reduction in the complexity in learning S that is within an error bound of the original S.

4.1. Background

Consider a voice-over-IP application that allows users to converse using voice over the Internet. In this application, the control is the mouth-to-ear delay (MED) between the speaker and the listener, and an "optimal" MED entails the tradeoff between voice quality and the conversation's responsiveness. Another example is a real-time online game in which increasing the delay of actions may make users feel that the game's responses are sluggish yet better synchronized.

There are two general concepts on perceptual quality in the literature: *just-noticeable distortion* (*JND*) and *awareness*. In psychophysics, *JND* is the minimal change of the original input (called *reference* I) to its modification (called *distortion* $I + \Delta I$), whose effect can be perceived by humans [5, 23]. In contrast, *awareness* p is the fraction of human subjects who can detect a change when I and $I + \Delta I$ are presented in a random order one after another. With this definition, a 75% awareness level is generally used in psychophysical studies. Given the three parameters (I, ΔI, p), we can relate them in a three-dimensional JND surface $p(I, \Delta I)$. In this application, the *kernel* is the awareness measured.

Using awareness as a measure of perceptual quality, the control of end-to-end delays to improve the user-perceived subjective quality is challenging because the function relating awareness to controls is ill-defined. Heuristic methods that combine quantitative metrics as a crude approximation to perceptual quality do not work well. A more accurate approach is to learn their relation using offline subjective tests conducted under a broad set of operating conditions and to generalize the relation learned to run-time operations. However, subjective tests are expensive to conduct even for one scenario, and there are infinitely many past scenarios to be evaluated.

4.2. Dominance Relation on Awareness

In this section, we formulate a dominance relation for pruning dominated operating points in a JND surface using an axiom we developed earlier [24]. We have observed from our past experiments on real-time multimedia the monotonicity of awareness to I and ΔI, respectively. The following axiom states this monotonicity property.

Axiom 4.1. *Monotonicity of Awareness [24].* The awareness in a JND surface has the monotonicity properties to the reference (I) and its modification (ΔI), respectively. (a) Awareness is monotonically non-increasing to I for a given ΔI. (b) Awareness is monotonically non-decreasing to ΔI for a given I. In other words, when given I (*resp.*, ΔI), awareness is a monotonic function to ΔI (*resp.*, I).

The axiom was originally developed for one reference input I but can be extended to multiple reference inputs in a straightforward manner [25]. It requires the continuity and smoothness of awareness to I and ΔI, respectively. These properties are valid because changes to awareness in a small region of I and ΔI (up to some granularity) are perceptually indistinguishable [26].

Figure 6a illustrates the axiom. For a given reference I on the x-axis, awareness is monotonically non-decreasing to its modification ΔI on the y-axis. Likewise, for a given ΔI on the y-axis, awareness is monotonically non-increasing with respect to I on the x axis.

We can draw the following observation from the monotonicity properties in Axiom 4.1.

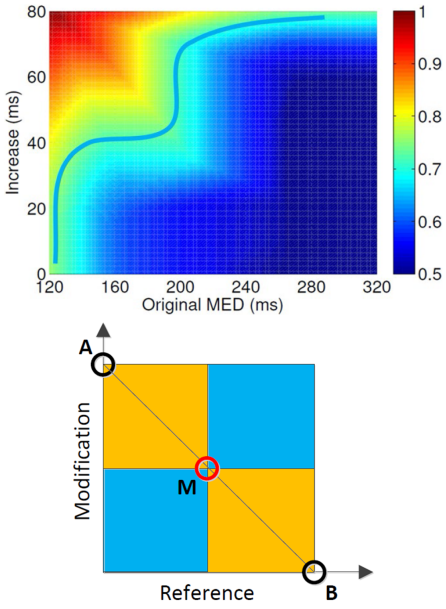


Figure 6. *(a) The JND surface of a voice-over-IP application shows a nonlinear relation among awareness p (color on the z axis), MED (I on the x axis) and increase in MED (ΔI on the y axis). The blue line represents a constant awareness of 0.75. The top-left-hand corner of the surface shows that subjects are very sensitive to changes in MED when the original MED is small. (b) The dominance relation is based on Axiom 4.1. The diagonal indicates the direction of monotonicity and points A, B, and M have subjective tests conducted with awareness p_A, p_B, and p_M, respectively. We can prune the top-left* (resp., *bottom-right) butter-colored block when $p_A - p_m \leq \delta$* (resp., *$p_M - p_B \leq \delta$*).

Observation 4.1. *Error bounds on awareness for points in a rectangular region of JND.* Consider two points $p_1(I_1, \Delta I_1)$ and $p_2(I_2, \Delta I_2)$. If $I_1 < I_2$ and $\Delta I_1 > \Delta I_2$, then for any point $p_3(I_3, \Delta I_3)$ where $I_1 \leq I_3 \leq I_2$ and $\Delta I_1 \geq \Delta I_3 \geq \Delta I_2$, we have $p_1 \geq p_3 \geq p_2$. As a result, in a rectangular region on a JND surface, the awareness of any point in the region is bounded by the awareness of the two diagonal corner

points A and B (see Figure 6b).

This observation leads to the following dominance relation for pruning points on the JND surface whose awareness is within a threshold δ from its actual value.

Property 4.1. *Dominance relation on awareness to within an error threshold* δ. Consider two diagonal corner points A and B of a rectangular block on a JND surface with awareness p_A and p_B (where $p_A \geq p_B$), respectively. If $p_A - p_B \leq \delta$, then any point C in the block have awareness p_C where $p_A - p_C \leq \delta$ and $p_C - p_B \leq \delta$. Hence, if we replace p_C by $p'_C = p_A$ or $p'_C = p_B$, then $|p'_C - p_C| \leq \delta$.

This property follows from Axiom 4.1 and Observation 4.1.

We have developed an efficient binary-divide algorithm that uses the dominance relation to find an approximate JND surface with an awareness that is within an error threshold δ. We start by testing the upper-left (A) and lower-right (B) diagonal points of the surface (Figure 6b). We stop the process if $p_A - p_B \leq \delta$, where δ is an acceptable threshold on the uncertainty in awareness. Otherwise, we conduct subjective tests at the center point (M) of the diagonal and divide the surface into four regions. As the two butter-colored blocks have awareness bounded by their two corner points, we can prune the top-left (*resp.*, bottom-right) block when $p_A - p_M \leq \delta$ (*resp.*, $p_M - p_B \leq \delta$). Since the two azure-colored regions cannot be pruned by p_M, we can apply the process to each. By repeating the process and subdividing a region into smaller blocks, we can either measure the awareness at the two diagonal and the midpoints or prune the region if its uncertainty is within δ. The awareness in the resulting JND surface is an interpolation of those points verified by subjective tests. We use interpolations to ensure that the values are smooth in the final JND surface, although we may not be able to guarantee that the interpolated points would satisfy δ.

By applying the algorithm, we can interpolate all the awareness in Figure 6a by performing subjective tests on five points using $\delta = 0.01$ [24]. For comparison, without the dominance relation, it would be necessary to perform expensive subjective tests at hundreds of points, rendering the problem intractable.

The JND surface described so far was generated under a given operating condition. For example, in the VoIP application, subjective tests are conducted under a given network loss and delay scenario. Under some general assumptions, a JND surface collected under one scenario can be extended to other scenarios without new subjective tests [24]. In cases when the conversational style changes, new subjective tests will need to be conducted. In short, very few JND surfaces would need to be collected offline to cover a wide range of operating conditions.

The ability to learn JND surfaces efficiently has opened many new opportunities that allow a real-time control system to generalize the JND surface learned offline to determine the best operating points in the next time instant. We have made extensions to combine multiple offline JND surfaces, each associated with one quality metric. The combined JND surface allows us to identify the best operating points for interactive multimedia applications at run-time [24].

For example, consider the VoIP application running at a suboptimal MED (on the x axis of the JND surface). The best run-time control is to increase or decrease the MED along the y axis (that defines ΔMED) until it reaches a new MED whose awareness is acceptable (say at 75%). The system then evaluates whether the signal quality and the interactivity are satisfactory and re-adjusts the MED if needed. The MED found can then be generalized to the near future by using the properties of the continuity of awareness and temporal locality.

Figure 7 illustrates the process of offline learning of the JND surface, run-time adaptation of the JND surface to the current network and operating conditions, and generalization of the adapted surface to the best operating points.

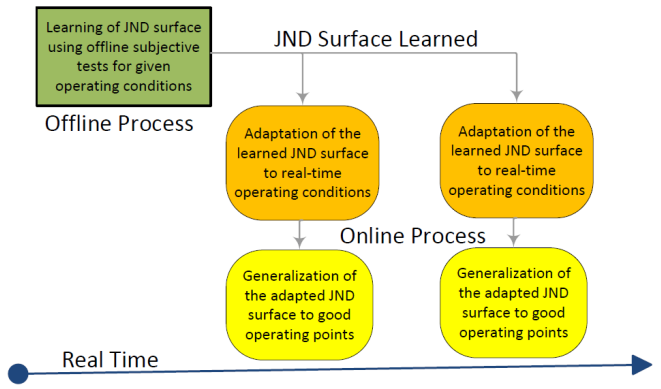


Figure 7. *Using the offline-learned JND surface in real-time to find the best operating points in a VoIP application.*

As another example, optimizing the vector of actions in a multi-player online game makes it possible to conceal the virtual delay of multi-player actions. The result allows users to perceive the game to run smoother (with more negligible virtual delay) without compromising the synchronization of action orders [25]. Interested readers can refer to the references for details [24, 25].

5. Conclusions

This paper presents the design of dominance relations to significantly reduce the space evaluated in machine learning for solving two data-intense applications. Using kernels to represent the statistical quality, average density, or probability of solutions in a subspace, we present the use of dominance relations to prune a subspace with suboptimal kernels.

In the financial trading application studied, we define a subspace in which stocks are traded daily. We present an application solver based on mean-reversion in which we learn thresholds for identifying stocks with a high probability of positive return the following day. Using the projected annual return and the downside deviation as our kernel metrics, we utilize the domain-independent Pareto optimality to prune suboptimal mean-reversion thresholds in the learning space.

In the real-time multimedia application studied, we reduce the space of abstract perceptual qualities to a subspace defined by the awareness kernel metric. As this subspace may require

expensive subjective tests on many past scenarios, we present a dominance relation that allows the pruning of a majority of suboptimal past scenarios and that guarantees the pruned scenarios to be within a prescribed error tolerance. With only a few subjective tests and interpolations, we can learn a complete mapping between awareness and control that can be used by the run-time application system to find the best operating points.

Acknowledgements

This project was supported by the National Key Basic Research Program of China (973 Program) No. 2014CB340401. The author would like to thank Mr. Kyle Xing for his support in developing the programs in Section 3.4.

References

[1] E. Fama, "Random walks in stock market prices," *Financial Analysis Journal*, vol. 21, no. 5, pp. 55-59, Sep.-Oct. 1965.

[2] Y. LeCun, J. Denker, and S. Solla, "Optimal brain damage," in *Advances in Neural Information Processing Systems* (D. Touretzky, ed.), vol. 2, Morgan-Kaufmann, 1989.

[3] D. Blalock, J. J. G. Oritz, J. Frankle, and J. Guggag, "What is the state of neural network pruning," in *Proc. of the 3rd MLSys Conference*, 2020.

[4] L. A. Breslow and D. W. Aha, "Simplifying decision trees: A survey," *The Knowledge Engineering Review*, vol. 12, no. 1, pp. 1-47, 1997.

[5] G. T. Fechner, E. G. Boring, and D. H. Howes, *Elements of Psychophysics [Elemente der Psychophysik (Translated by H. E. Adler)]*. Holt, Rinehart and Winston, 1966 [First published, 1860].

[6] B. W. Wah, "Using kernels to harness the complexity of big data applications," *Int'l Journal on Artificial Intelligence Tools*, vol. 31, no. 3, pp. 2241006-1-11, 2022.

[7] H.-W. Shen and A.-L. Barabasi, "Collective credit allocation in science," *Proc. National Academy of Sciences*, vol. 111, no. 34, pp. 12325-12330, 2014.

[8] S. Milgram, "The small-world problem," *Psychology Today*, vol. 1, no. 1, pp. 61-67, 1967.

[9] J. Huang, W.-Q. Wang, H.-W. Shen, G. Li, and X.-Q. Cheng, "Temporal scaling in information propagation," *Scientific Reports*, vol. 4, no. 5334, pp. 1-6, 2014.

[10] Z. Ji, Y. Pang, and X. Li, "Relevance preserving projection and ranking for web image search reranking," *IEEE Trans. on Image Processing*, vol. 24, no. 11, pp. 4137-4147, 2015.

[11] X. Yan, J. Guo, Y. Lan, J. Xu, and X. Cheng, "A probabilistic model for bursty topic discovery in microblogs," in *Proc. 29th AAAI Conference*, (Austin, TX), 2015.

[12] A. Jouglet and J. Carlier, "Dominance rules in combinatorial optimization problems," *European J. of Operational Research*, vol. 212, no. 3, pp. 433-444, Aug. 2011.

[13] J. Yang and J. Leskovec, "Overlapping community detection at scale: A nonnegative matrix factorization approach," in *Proc. ACM Int'l Conf. on Web Search and Data Mining (WSDM)*, (Rome, Italy), 2013.

[14] Z. Wei, X. Liu, F. Li, S. Shang, X. Du, and J.-R. Wen, "Matrix sketching over sliding windows," in *Proc. ACM SIGMOD*, (San Francisco, CA), 2016.

[15] B. Li and S. C. H. Hoi, "Online portfolio selection: A survey," *ACM Computing Surveys*, vol. 46, no. 3, p. 35, 2014.

[16] A. Jouglet and J. Carlier, "Dominance rules in combinatorial optimization problems," *European J. of Operational Research*, vol. 212, no. 3, pp. 433-444, 2011.

[17] C.-L. Hwang and A. S. M. Masud, *Multiple Objective Decision Making-Methods and Applications: A State-of-the-Art Survey*. Lecture Notes in Economics and Mathematical Systems, No. 164, Springer-Verlag, 1979.

[18] K. Deb, "Multi-objective optimization," in *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*, pp. 403-449, Springer US, 2014.

[19] N. Jegadeesh, "Evidence of predictable behavior of security returns," *Journal of Finance*, vol. 45, pp. 881-898, 1990.

[20] S. Stigler, "Regression toward the mean, historically considered," *Statistical Methods in Medical Research*, vol. 6, no. 2, pp. 103-114, 1997.

[21] InteractiveBrokers, "Pricing structure," Nov., 2022. https://www.interactivebrokers.com/en/pricing/commissions-home.php.

[22] B. Li, P. Zhao, S. C. H. Hoi, and V. Gopalkrishnan, "PAMR: Passive aggressive mean reversion strategy for portfolio selection," *Machine Learning*, vol. 87, no. 2, pp. 221-258, 2012.

[23] J. Ferwerda, "Psychophysics 101: how to run perception experiments in computer graphics," in *ACM SIGGRAPH 2008 classes*, (Los Angeles, CA), p. 87, 2008.

[24] J. X. Xu and B. W. Wah, "Optimizing the perceptual quality of real-time multimedia applications," *IEEE Multimedia*, vol. 22, no. 4, pp. 14-28, Oct-Dec 2015.

[25] J. X. Xu and B. W. Wah, "Consistent synchronization of action order with noticeable dealy in online games," *ACM Trans. on Multimedia Computing, Communications, and Applications*, vol. 8, no. 1, Jan. 2017.

[26] X. Xu and B. W. Wah, "Optimality of greedy algorithm for generating just-noticeable difference surfaces," *IEEE Trans. on Multimedia*, vol. 18, no. 7, pp. 1330-1337, July 2016.