Symphony in the Latent Space: Provably Integrating High-dimensional Techniques with Non-linear Machine Learning Models

Qiong Wu*[1], Jian Li[2], Zhenming Liu[1], Yanhua Li[3], Mihai Cucuringu[4]
[1]William & Mary
[2]Tsinghua University
[2]Worcester Polytechnic Institute
[4]University of Oxford and The Alan Turing Institute

Abstract

This paper revisits building machine learning algorithms that involve interactions between entities, such as those between financial assets in an actively managed portfolio, or interactions between users in a social network. Our goal is to forecast the future evolution of ensembles of multivariate time series in such applications (e.g., the future return of a financial asset or the future popularity of a Twitter account). Designing ML algorithms for such systems requires addressing the challenges of high-dimensional interactions and non-linearity. Existing approaches usually adopt an **ad-hoc** approach to integrating high-dimensional techniques into non-linear models and recent studies have shown these approaches have questionable efficacy in time-evolving interacting systems.

To this end, we propose a novel framework, which we dub as the ***additive influence model***. Under our modeling assumption, we show that it is possible to decouple the learning of high-dimensional interactions from the learning of non-linear feature interactions. To learn the high-dimensional interactions, we leverage kernel-based techniques, with provable guarantees, to embed the entities in a low-dimensional latent space. To learn the non-linear feature-response interactions, we generalize prominent machine learning techniques, including designing a new statistically sound non-parametric method and an ensemble learning algorithm optimized for vector regressions. Extensive experiments on two common applications demonstrate that our new algorithms deliver significantly stronger forecasting power compared to standard and recently proposed methods.

Introduction

We revisit the problem of building machine learning algorithms that involve interactions between entities, such as those between users and items in a recommendation system, or between financial assets in an actively managed portfolio, or between populations in different counties in a disease-spreading process. Our proposed forecasting model uses information available up to time t to predict $\mathbf{y}_{t+1,i}$, the future behavior of entity i at time $t+1$ (e.g., the future price of stock i at time $t+1$), for a total number of d entities (Laptev et al. 2017; Farhangi et al. 2022). Designing such models has proven remarkably difficult, as one needs to circumvent two main challenges that require often incompatible solutions.

1. Cross-entity interaction: high-dimensionality. In many ensembles of multivariate time series systems, it is often the case that the current state of one entity could potentially impact the future state of another. When considering the equity market as an example, Amazon's disclosure of its revenue change in cloud services could indicate that the revenues of other cloud providers (e.g., competitors) could also change.

The interaction is high-dimensional because the total possible number of interactions is usually much larger than the number of available observations. For example, in a portfolio of 3,000 stocks, the total number of potential links between pairs of stocks is $3,000 \times 3,000 \approx 10^7$, but we often have only 2,500 data points (e.g., 10 years of daily data), and thus capturing the cross-entity interactions becomes a very challenging problem.

2. Feature-response interactions: non-linearity. Linear models are usually insufficient to characterize the relationship between the response/label and the available information (features), thus techniques beyond simple linear regressions are heavily needed. For example, in a financial context, economic productivity is non-linear in temperature for most countries; similarly, electricity consumption is a nonlinear function of temperature, and modeling this relationship is crucial for pricing electricity derivative contracts. As shown in Fig. 1(a), the existing relevant learning models can be categorized into the following two groups.

1. Provable cross-entity models (CEM) for high-dimensionality. Cross-entity models solve a vector regression problem $\mathbf{y}_{t+1} = f(\mathbf{x}_t) + \xi_t$, to forecast the future behavior of all entities, where $\mathbf{y}_{t+1} \triangleq (y_{t+1,1}, \ldots, y_{t+1,d})$, and $\mathbf{x}_t$ denotes the features of all entities, constructed from their historical data. Since the features of one entity can be used to predict the future behavior of another, CEMs have stronger expressive and predictive power. CEMs are both computationally and statistically challenging because we need to solve the "high-dimensional" (overparametrized) problem and mathematically understand the root cause of the overfitting. Extensive research has been undertaken to design regularization techniques (Chen, Dong, and Chan 2013; Friedman, Hastie, and Tibshirani 2001; Wu et al. 2021) to address the issue, and most algorithms in this category are

*Currently working at AT&T Labs.

linear and have theoretical guarantees.

2. Practical univariate models (UM) for non-linearity. Univariate models fit a function $\mathbf{y}_{t+1,i} = f(\mathbf{x}_{t,i}) + \xi_{t,i}$ to forecast one entity's feature behavior by using features constructed from that entity's historical data. Univariate models primarily learn the feature-response interaction by using off-the-shelf ML techniques such as Deep learning (DL) (Abadi et al. 2016; Hochreiter and Schmidhuber 1997; Wu et al. 2019) or gradient boosted algorithms (Chen and Guestrin 2016; Ke et al. 2017; Dorogush, Ershov, and Gulin 2018). These practical models are effective in extracting non-linear signals but they often do not come with theoretical guarantees.

Existing integration techniques: ad-hoc methods It remains unclear how to integrate two seemingly incompatible modeling processes (i.e., UM and CEM) with different design philosophies. In Fig. 1 (b), we show that existing integration solutions predominately follow an **ad-hoc** approach, in part due to the belief that deep learning is the "holly-grail" for practical problems (Sejnowski 2018). For example, one often adds an ℓ_1- or ℓ_2-regularizer to a neural net's cost function, hoping such regularizers will also magically work in neural nets (Abadi et al. 2016; Paszke et al. 2017). However, the mathematical properties of a provable technique often break when combined into a neural net. Furthermore, latent embedding models have also been recently introduced (Wang et al. 2019; Feng et al. 2019; Chen et al. 2019). The central idea is to project the entities into points in a low-dimensional space so that similar entities (i.e., stocks in the above works) are closer to each other in this embedding. Because point interactions are more restrictive in the latent space, they have the potential to address the overfitting issues (Wang et al. 2019). However, these lines of work do not offer any theoretical guarantees and are often not robust in practice. Recent studies have demonstrated that the efficacy of such ad-hoc approaches is questionable in many interacting systems (Dacrema, Cremonesi, and Jannach 2019; Rendle, Zhang, and Koren 2019; Qiong et al. 2021).

Our approach & contributions We propose a general latent position model dubbed as the *additive influence model* to enable us to seamlessly orchestrate mathematically rigorous high-dimensional techniques with practically effective machine learning algorithms. In Fig. 1(c), we show that it is possible to decouple the learning of high-dim interactions between entities from the learning of the non-linear signals.

We assume each entity is associated with an embedded position $\mathbf{z}_i$ and at timestamp t, entity i is also associated with an unobserved signal $\mathbf{s}_{i,t} \in \mathbf{R}$ that is a function of $\mathbf{x}_{i,t}$. We assume the generative model $\mathbf{y}_{i,t} = \sum_{i \le j} \kappa(\mathbf{z}_i, \mathbf{z}_j)\mathbf{s}_{j,t} + \epsilon_{i,t}$, where $\kappa(\mathbf{z}_i, \mathbf{z}_j)$ is a function that measures the interaction strength between $\mathbf{z}_i$ and $\mathbf{z}_j$, and can be any kernel function, such as a Gaussian kernel or simply an inner product, and $\epsilon_{i,t}$ denotes noise. Each entity could potentially influence $\mathbf{y}_{i,t}$. The influence of j on i depends on the "distance" or "similarity" between $\mathbf{z}_i$ and $\mathbf{z}_j$. On the other hand, we assume $\mathbf{s}_{j,t} = g(\mathbf{x}_{j,t})$ for some $g(\cdot)$, so that the model captures high-dimensional interactions via $\mathbf{z}_i$ and non-linearity via $g(\cdot)$.

Our proposed model allows for feature interactions through $g(\cdot)$, and addresses the overfitting problem arising from entity interactions because the distances (interaction strength) between entities are constrained by the latent Euclidean space: when both $(\mathbf{z}_i - \mathbf{z}_j)$ and $(\mathbf{z}_j - \mathbf{z}_k)$ are small, then $(\mathbf{z}_i - \mathbf{z}_k)$ is also small, and thus the degree of freedom for entity interactions becomes substantially smaller than $O(d^2)$.

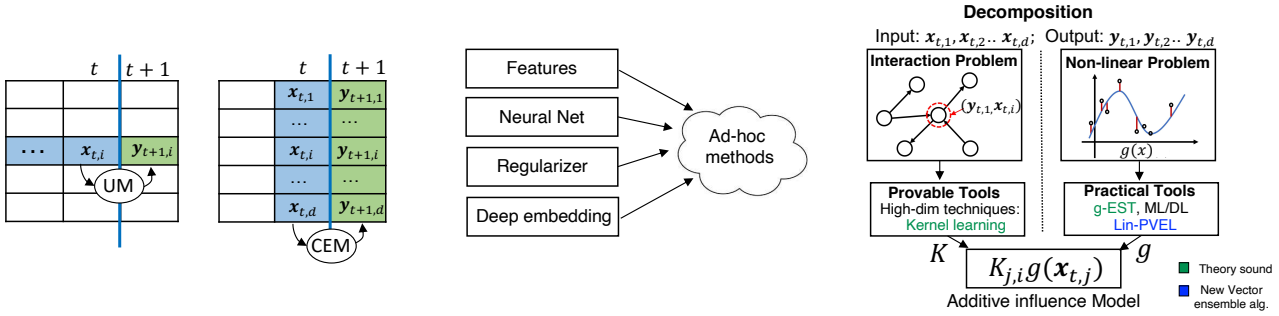
Our goal is to learn both the $\mathbf{z}_i$'s and $g(\cdot)$. We note that these two learning tasks can be *decoupled*: high-dimensional methods can be developed to *provably* estimate the $\mathbf{z}_i$'s *without the knowledge of* $g(\cdot)$, and when estimates of $\mathbf{z}_i$'s are given, an experiment-driven process can be used to learn $g(\cdot)$ by examining prominent machine learning methods such as neural nets and boosting. In other words, when we learn entity interactions, we do not need to be troubled by the overfitting problem escalated by fine-tuning $g(\cdot)$, and when we learn feature interactions, the generalization error will not be jeopardized by the curse of dimensionality from entity interactions.

- To learn the $\mathbf{z}_i$'s, we design a simple algorithm that uses low-rank approximation of $\mathbf{y}_t$'s covariance matrix to infer the closeness of the entities and develop a novel theoretical analysis based on recent techniques from high dimensionality and kernel learning (Belkin 2018; Tang et al. 2013; Wu et al. 2020a).
- To learn $g(\cdot)$, we generalize major machine learning techniques, including neural nets, non-parametric, and boosting methods, to the additive influence model when estimates of $\mathbf{z}_i$'s are known. We specifically develop a moment-based algorithm for non-parametric learning of $g(\cdot)$, and a computationally efficient boosting algorithm.
- Finally, we perform extensive experiments on a major equity market and social network datasets to confirm the efficacy of our modeling approaches and analysis.

Related work and comparison

Univariate machine learning models handle feature-response interactions and mostly rely on deep learning and GBRT (Goodfellow, Bengio, and Courville 2016; Wu et al. 2020b; Goodfellow, Bengio, and Courville 2016; Wüthrich, Permunetilleke et al. 1998; Chen and Guestrin 2016; Ke et al. 2017; Dorogush, Ershov, and Gulin 2018; Gong et al. 2017; Yang and Ding 2020; Ding et al. 2015; Zhang, Aggarwal, and Qi 2017; Feng, Polson, and Xu 2018; Han et al. 2018; Wu et al. 2020b; Chen, Pelger, and Zhu 2019; Kelly, Pruitt et al. 2019; Ke et al. 2019; Chen et al. 2019; Li et al. 2019; Wu et al. 2015). These models aim to optimize their empirical performance and limit theoretical investigations. Recent cross-entity models consider the high-dimensional interactions, where overfitting easily happens and theoretical justifications are essential to avoid spurious result in practice. Cross-entity models are mostly linear models (Bunea, She, and Wegkamp 2011; Koltchinskii, Lounici et al. 2011; Negahban and Wainwright 2011; Huang, Li, and Zhou 2019) that have theoretical guarantees, but they cannot effective for non-linear feature-response interactions. Efforts for building CEMs include (Tibshirani 1996; Candès and Wakin 2008; Tao and Series 2009; Hoerl and Kennard 1970; Tsigler and Bartlett 2020; Liu et al. 2019).

Ad-hoc approach for integration. Recent integrating solutions for high-dimensionality and nonlinearity challenges has



(a) UM and CEM. (b) Ad-hoc methods lead to overfitting. (c) Our proposed framework.

Figure 1: (a) UM for non-linearity and CEM for high-dimensionality. (b) Exsiting ad-hoc methods have questionable efficay. (c) Our framework decouples the high-dimensional learning of entity interactions and non-linear learning of feature interactions.

been a frustrating endeavor, which we can call the ad-hoc approaches and many were shown to have questionable efficacy in interacting systems. *1. Deep learning + Lasso/Ridge* For example, one (Abadi et al. 2016; Paszke et al. 2017) often adds an l_1- or l_2-regularizer to a neural net's cost function, hoping these regularizers can also magically work in neural nets. *2. Deep embedding.* Recent studies have addressed high-dimensional entity interactions by using deep embedding, based on the idea that when entities are embedded in low-dim Euclidean space, they can interact in a quite restricted way, therefore preventing overfitting (Zhao et al. 2020; Shen et al. 2022; Xie, Girshick, and Farhadi 2016; Zhang, Aggarwal, and Qi 2017; Hu, Liu et al. 2018; Li et al. 2019; Wang et al. 2019). While this idea is effective for linear models (Abraham et al. 2015; Li et al. 2017), deep embedding-based solutions may have very high false positive rates, for instance, when forecasting the returns of financial assets (Qiong et al. 2021; Wang et al. 2019).

Remark: *(i) Modeling framework.* Our framework proposes a key algorithmic insight that the latent position estimation should be decoupled from the learning link function $g(\cdot)$. We develop the first algorithm that can provably estimate the entity's latent positions and provide theoretical guarantees. Our novel analysis leverages a diverse set of tools from kernel learning, non-parametric methods, and random walks. *(ii) Comparison to deep embedding.* While embedding can be learned by deep learning (Hu, Liu et al. 2018; Wang et al. 2019), it usually does not provide any theoretical guarantee, whereas our framework makes stricter assumptions (e.g., how embedding and features should interact) and delivers a quality guarantee. Deep embedding also requires every component including the function $g(\cdot)$ in the architecture to be represented by a neural net to run SGD, whereas we allow $g(\cdot)$ to be learned by a wide range of algorithms such as boosting or non-parametric techniques.

Problem definition

Notations. For a matrix A, $\mathcal{P}_r(A)$ denotes its rank-r approximation obtained by keeping the top r singular values and the corresponding singular vectors. $\sigma_i(A)$ (resp. $\lambda_i(A)$) is the i-th singular value (resp. eigenvalue) of A. We use Python/MATLAB notation when we refer to a specific row or column. For example, $A_{1,:}$ is the first row of A, and $A_{:,1}$ is the first column. $\|A\|_F$ and $\|A\|_2$ denote the Frobenius and spectral norms, respectively, of A. In general, we use boldface upper case (e.g., $\mathbf{X}$) to denote data matrices and boldface lower case (e.g., $\mathbf{x}$) to denote one sample. $\mathbf{x}_{t,i}$, which refers to the features associated with stock i at time t, can be one or multi-dimensional. Let $(\mathbf{x}_{t,i})_j$ be the j-th coordinate (feature) of $\mathbf{x}_{t,i}$. An event occurring with high probability (whp) means that it happens with probability $\geq 1 - n^{-10}$, where 10 is an arbitrarily chosen large constant and is not optimized. A bivariate function is a Gaussian kernel if $\kappa(\mathbf{x}, \mathbf{x}') = \exp(-\|\mathbf{x} - \mathbf{x}'\|^2/\sigma^2)$, an inverse multi-quadratic (IMQ) kernel if $\kappa(\mathbf{x}, \mathbf{x}') = (c^2 + \|\mathbf{x} - \mathbf{x}'\|^2)^{-\alpha}$ ($\alpha > 0$), and an inner product kernel if $\kappa(\mathbf{x}, \mathbf{x}') = \langle \mathbf{x}, \mathbf{x}' \rangle$. A function $g(\cdot)$ is Lipschitz-continuous if $|g(\mathbf{x}_1) - g(\mathbf{x}_2)| \leq c\|\mathbf{x}_1 - \mathbf{x}_2\|$ for a constant c. A distribution $\mathcal{D}$ with bounded domain and probability density function $f_{\mathcal{D}}$ is near-uniform if $\frac{\sup f_{\mathcal{D}}(\mathbf{x})}{\inf f_{\mathcal{D}}(\mathbf{x})} = O(1)$.

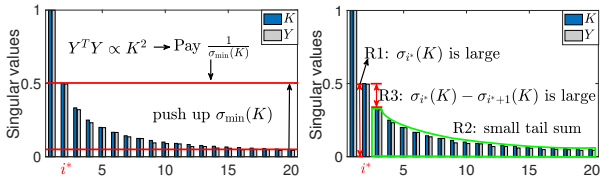
The forecasting problem. We operate in a time-dependent setting, where each timestamp t can be construed as the t^{th} round. An interacting system consisting of d entities (e.g., denoting stocks in the equity market or user accounts in a network), that are updated at each round, for a total number of T rounds. Let $\mathbf{y}_{t,i} \in \mathbf{R}$ denote the next-period forecast of entity i at the t-th round, and $\mathbf{y}_t = (\mathbf{y}_{t,1}, \ldots, \mathbf{y}_{t,d}) \in \mathbf{R}^d$. Our goal is to forecast $\mathbf{y}_t$ based on all information available up to (but excluding) round t.

Model Assumptions. Under the additive influence model, a generic model takes the form

$$\mathbf{y}_{t,i} = \sum_{j \leq d} \kappa(\mathbf{z}_i, \mathbf{z}_j) g(\mathbf{x}_{t,j}) + \xi_{t,i}, \tag{1}$$

and our goal is to learn $g(\cdot)$ and $\mathbf{z}_i$'s with a total number of n observations. Let $K \in \mathbf{R}^{d \times d}$ such that $K_{i,j} = \kappa(\mathbf{z}_i, \mathbf{z}_j)$. Here, we assume that • *(A.1)* the vector representations $\mathbf{z}_i$'s of the stocks and features $\mathbf{x}_{t,i}$ are i.i.d. samples from (two different) near-uniform distributions on bounded supports, • *(A.2)* $\mathbf{x}_{t,i} \in [-1, 1]$ and $\mathbb{E}[g(\mathbf{x}_{t,i})] = 0$, • *(A.3)* $g(\cdot)$ is Lipschitz-continuous, and • *(A.4)* $\xi_{t,i}$'s are zero-mean i.i.d. Gaussian random variables with standard deviation σ_ξ.

We remark that (A.1) is standard in the literature (Abraham et al. 2015; Sussman, Tang, and Priebe 2013; Tang et al. 2013; Li et al. 2017; Rastelli, Friel, and Raftery 2016). Assuming (A.2) simplifies the calculation and is without loss of generality, and (A.4) can also be relaxed to settings in which the $\xi_{t,i}$ variables are sub-Gaussian. See App. A for a more



(a) (b)

Figure 2: (a) We use the square root of $\mathcal{P}_{i^*}(\mathbf{Y}^{\mathrm{T}}\mathbf{Y})$ to approximate K so that we pay a factor of $1/\sigma_{i^*}(K)$, instead of $1/\sigma_{\min}(K)$. (b) Three key requirements for i^*: • (R1) $\sigma_{i^*}(K)$ is large, • (R2) $\mathcal{P}_{i^*}(K^2)$ is close to K^2, and • (R3) $\sigma_{i^*}(K) - \sigma_{i^*+1}(K)$ is large.

detailed discussion of the assumptions.

Our algorithms

This section introduces our algorithmic pipeline in full detail. Sec. 4 describes an algorithm for learning the embedding without knowing $g(\cdot)$. Sec. 4 explains the estimation of $g(\cdot)$ using machine learning techniques. Due to the space limit, detailed proofs of all the Props are deferred to App. B.

Learning vector representation provably

This section presents a provable algorithm to estimate the kernel matrix K and the embedding $\mathbf{z}_i$'s. Our algorithm does not require knowledge of $g(\cdot)$, thus providing a conceptually new approach to construct CEMs: high-dimensional learning of entity interactions can be decoupled from using ML techniques to fit the features. Because learning entity interactions could be a major source of causing overfitting, disentangling it from the downstream task of learning $g(\cdot)$ enables us to leverage the function-fitting power of ML techniques without the cost of amplifying generalization errors.

We next walk through our design intuition and start by introducing additional notation. Let $\mathbf{Y} \in \mathbf{R}^{n\times d}$ be such that $\mathbf{Y}_{t,i} = \mathbf{y}_{t,i}$ ($\mathbf{Y}$ is a matrix and $\mathbf{y}$ a random variable), $\mathbf{S} \in \mathbf{R}^{n\times d}$ with $\mathbf{S}_{t,i} = \mathbf{s}_{t,i} \triangleq g(\mathbf{x}_{t,i})$, and $E \in \mathbf{R}^{n\times d}$ with $E_{t,i} = \xi_{t,i}$. Recall that $K \in \mathbf{R}^{d\times d}$ s.t. $K_{i,j} = \kappa(\mathbf{z}_i, \mathbf{z}_j)$, and $\mathcal{P}_r(A)$ denotes A's rank-r approximation obtained by keeping the top r singular values and vectors. Finally, for any PSD matrix A with SVD $A = U\Sigma U^{\mathrm{T}}$, let $\sqrt{A} \triangleq U\Sigma^{\frac{1}{2}}U^{\mathrm{T}}$.

Eq. (1) can be re-written as $\mathbf{Y} = \mathbf{S}K + E$, in which we need to infer K using only $\mathbf{Y}$. We first observe that while none of the entries in $\mathbf{S}$ are known, the $\mathbf{S}_{t,i}$'s are i.i.d. random variables (because the $\mathbf{x}_{t,i}$'s are i.i.d.); therefore, our problem resembles a *dictionary learning* problem, in which K can be viewed as the dictionary to be learned, and $\mathbf{S}$ is the measurement matrix (see e.g., (Arora, Bhaskara et al. 2014)). However, in our case, K is neither low-rank nor sparse, and we cannot use standard dictionary learning techniques.

First, we observe that, if infinitely many samples were available, then $\mathbf{Y}^{\mathrm{T}}\mathbf{Y}/n$ approaches to K^2. Hence, intuitively we could use $\sqrt{\mathbf{Y}^{\mathrm{T}}\mathbf{Y}/n}$ to approximate $\sqrt{K^2} = K$. However, the existing standard matrix square root result has the notorious "$1/\sigma_{\min}$-blowup" problem, i.e., it gives us only $\|\sqrt{\mathbf{Y}^{\mathrm{T}}\mathbf{Y}/n} - KW\|_F \propto 1/\sigma_{\min}(K)$ (W a unitary matrix), where typically $\sigma_{\min}(K)$ is extremely small, thus rendering the bound too loose to be useful (Bhojanapalli, Kyrillidis, and Sanghavi 2016).

To tackle the problem, our algorithm uses $\sqrt{\mathcal{P}_{i^*}(\mathbf{Y}^{\mathrm{T}}\mathbf{Y})/n}$ to approximate K for a carefully chosen i^* so that we pay a factor of $\sigma_{i^*}(K)$, instead of $\sigma_{\min}(K)$, to substantially tighten the error. See Alg. 2 in App. B and Fig. 2(a). To implement this idea, we need to show that there always exists an i^* such that • *(R1):* $\sigma_{i^*}(K)$ is sufficiently large, • *(R2):* $\mathcal{P}_{i^*}(K^2)$ is close to K^2, and • *(R3):* the spectral gap $\sigma_{i^*}(K) - \sigma_{i^*+1}(K)$ is sufficiently large so that we can use the Davis-Kahan theorem to prove that $\mathcal{P}_{i^*}(K^2) \propto \mathcal{P}_{i^*}(Y^{\mathrm{T}}Y)$ (Stewart 1990). See also Fig. 2(b).

These three requirements may not always be met simultaneously. For example, when $\sigma_i(K^2) \propto \frac{1}{i}$, the gap is insufficient and the tail diverges (R2 and R3 are violated). Therefore, we integrate the following two results. • *(i)* The eigenvalues decay fast. This stems from two classical results from the *kernel learning* literature. First, when $\kappa(\cdot,\cdot)$ is sufficiently smooth (such as the Gaussian, IMQ, or inner product kernels), the eigenvalues of the kernel operator $\mathcal{K}$ associated with $\kappa(\cdot,\cdot)$ decay exponentially (e.g., $\lambda_i(\mathcal{K}) \leq \exp(-Ci^{\frac{1}{r}})$ for Gaussian kernels (Belkin 2018)). Second, it holds true that $\sum_{i\geq 1} |\lambda_i(\mathcal{K}) - \lambda_i(K/d)|_F^2 \propto \frac{1}{n}$, a convergence result under the PAC setting (Tang et al. 2013). Therefore, $\lambda_i(K)$ also approximately decays exponentially. • *(ii)* Combinatorial analysis between gaps and tails. We then leverage a recent analysis (Wu et al. 2020a) showing that when $\lambda_i(K)$ decays fast, it is always possible to find an i^* such that $\lambda_{i^*}(\mathcal{K}) - \lambda_{i^*+1}(\mathcal{K})$ is sufficiently large (R1 & R3 are satisfied) and $\sum_{j\geq i^*} \lambda_j^2(\mathcal{K}) = o(1)$ (R2 is satisfied). Putting all these together leads to the following statement.

Proposition 4.1. *Consider the additive influence model. Let $\kappa(\mathbf{z}_i, \mathbf{z}_j)$ be a Gaussian, inverse multi-quadratic (IMQ) or inner product kernel. Let $n \geq d$ be the number of observations and $\epsilon = \frac{c_0 \log^3 d}{\sqrt{d}}$. Assume that the noise level $\sigma_\xi = O(\sqrt{d})$. Let δ be a tunable parameter (also appeared in Alg. 2 in App. B) such that $\delta^3 = \omega(\epsilon^2)$. There exists an efficient algorithm that outputs $\hat{K}$ such that $\frac{1}{d^2}\|\hat{K} - K\|_F^2 = O(\frac{\epsilon^2}{\delta^3} + \delta^{\frac{4}{5}})(= \tilde{O}(d^{-\Theta(1)}))$.*

We remark that *(i)* the algorithm does not need to know the exact form of κ, so long as it is one of Gaussian, IMQ, or inner product kernels, *(ii)* once K is estimated, an Isomap-flavored algorithm may be used to estimate $\mathbf{z}_i$'s (Li et al. 2017), and *(iii)* knowing $\hat{K}$ (without reconstructing $\mathbf{z}_i$'s) is sufficient for the downstream $g(\cdot)$-learners.

Learning $g(\cdot)$

Here, we explain how prominent machine learning techniques, including neural nets (deep learning), non-parametric methods, and boosting, can be used to learn $g(\cdot)$. These techniques make different functional form assumptions of $g(\cdot)$, and possess different "iconic" properties: deep learning assumes that $g(\cdot)$ can be represented by a possibly sophisticated neural net and uses stochastic gradient descent to train the model; non-parametric methods learn a Lipschitz-continuous $g(\cdot)$ with statistical guarantees; boosting consolidates forecasts produced from computationally efficient weak learners.

Our setting has a different cost structure: in univariate models, $g(\mathbf{x}_{t,j})$ controls only one response $\hat{\mathbf{y}}_{t,j}$, but here, $g(\mathbf{x}_{t,j})$ impacts all responses $\hat{\mathbf{y}}_{t,i}$, $i \in [d]$, as $\hat{\mathbf{y}}_{t,i} = \sum_j K_{i,j} g(\mathbf{x}_{t,j})$. We generalize ML techniques under the new cost functions,

Algorithm 1 nparam-gEST:

Input $\mathbf{X}$, $\mathbf{Y}$, $\hat{K}$;
Output μ_1 (estimating other μ_i's is similar)

procedure NPARAM-GEST($\hat{K}, \mathbf{X}, \mathbf{Y}$)
  for all $t \leftarrow 1$ **to** n **do**
    $q_t = \mathrm{Rand}(d)$
    $L_{(t,q_t),j} = \textsc{Map-Regress}(q_t, \hat{K}, \mathbf{X}_{t,:})$
  return $\mu_1 \leftarrow \textsc{FlipSign}\ (q_t, \{\mathbf{y}_t, L_{(t,q_t),j}\}_{t\le n})$
procedure MAP-REGRESS($q_t, \hat{K}, \mathbf{x}_t$)
  Let $L_{(t,q_t),j} = 0$
  for all $k \leftarrow 1$ **to** d **do**
    $L_{(t,q_t),j} += \hat{K}_{q_t,k}$ with j s.t. $\mathbf{x}_{t,k} \in \Omega_j$.
  return $L_{(t,q_t),j}$
procedure FLIPSIGN($q_t, \{\mathbf{y}_t, L_{(t,q_t),j}\}_{t\le n}$)
  for all $t \leftarrow 1$ **to** n **do**
    $\hat{\Pi}_1^{(q_t)}(t) \triangleq L_{(t,q_t),1} - \frac{1}{\ell-1}\left(\sum_{j\neq 1} L_{(t,q_t),j}\right)$
    $\tilde{b}_{t,q_t} = \begin{cases} 1 & \text{if } \hat{\Pi}_1^{(q_t)}(t) \ge \frac{c}{\log d}\sqrt{\frac{d}{\ell}} \\ -1 & \text{if } \hat{\Pi}_1^{(q_t)}(t) < -\frac{c}{\log d}\sqrt{\frac{d}{\ell}} \\ 0 & \text{otherwise} \end{cases}$
  return $\mu_1 = \frac{\sum_{t\le n} \tilde{b}_{t,q_t}\mathbf{y}_{t,q_t}}{\sum_{t\le n} \tilde{b}_{t,q_t}\hat{\Pi}_1^{(q_t)}(t)}$

while retaining the iconic properties of each technique.

Technique 1. Learn $g(\cdot)$ using neural nets. When an estimate $\hat{K}$ is given, the training cost is $\sum_{t,i}(\mathbf{y}_{t,i} - \sum_{j\in[d]} \hat{K}_{i,j} g(\mathbf{x}_{t,j}))^2$, in which case one can employ stochastic gradient descent when $g(\cdot)$ is a neural net.

Technique 2. Learn $g(\cdot)$ using non-parametric methods. When the response is univariate, e.g., $\mathbf{y}_{t,i} = g(\mathbf{x}_{t,i}) + \xi_{t,i}$, we can use a neighbor-based approach to estimate $g(\mathbf{x})$ for a new $\mathbf{x}$: we identify one (or multiple) $\mathbf{x}_{t,i}$'s in the training set that are close to the new $\mathbf{x}$, and output $\mathbf{y}_{t,i}$ (or their averages, when multiple $\mathbf{x}_{t,i}$ are chosen), using $g(\mathbf{x}) \approx g(\mathbf{x}_{t,i})$, whenever $\mathbf{x}$ is close to $\mathbf{x}_{t,i}$.

Here, we do not directly observe the values of individual $g(\mathbf{x}_{t,i})$'s. Instead, each response is a linear combination of multiple $g(\cdot)$'s evaluated at different points, e.g., $\mathbf{y}_{t,1} = K_{i,1}\cdot g(\mathbf{x}_{t,1}) + \cdots + K_{i,d}\cdot g(\mathbf{x}_{t,d}) + \xi_{t,i}$. We show that finding neighbors reduces to solving a linear system. Furthermore, we design a moment-based algorithm, namely "nparam-gEST", which estimates $g(\cdot)$ with provable guarantees, as summarized in the following result.

Proposition 4.2. *Consider the problem of learning an additive influence model with the same setup/parameters as in Prop. 4.1. Assume that $\mathbf{x}_{t,i} \in \mathbf{R}^{O(1)}$. Let ℓ be a tunable parameter. There exists an efficient algorithm to compute $\hat{g}(\cdot)$, based on $\hat{K}$ such that $\sup_{\mathbf{x}} |\hat{g}(\mathbf{x}) - g(\mathbf{x})| \le (\log^6 n)\left(\sqrt{\gamma} + \sqrt{\frac{\ell}{n}} + \frac{1}{\ell}\right) = \tilde{O}(d^{-c})$ for suitable parameters, where $\gamma \triangleq \frac{\epsilon^2}{\delta^3} + \delta^{\frac{4}{5}}$.*

Our algorithm (Alg. 1) consists of the following 3 steps:
Step 1. Approximation of $g(\cdot)$. Partition $\Omega = [-1,1]^k$ into subsets $\{\Omega_j\}_{j\le\ell}$, and use piece-wise constant function to approximate $g(\cdot)$, i.e., $\tilde{g}(\mathbf{x}_{t,i})$ takes the same value for all $\mathbf{x}_{t,i}$ in the same Ω_j. We partition $\{\Omega_j\}_{j\le\ell}$ in a way such that $\Pr[\mathbf{x}_{t,i} \in \Omega_j]$ are the same for all j.

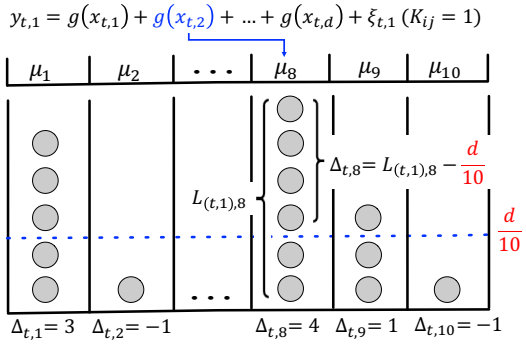


Figure 3: A toy example of nparam-gEST when $K_{i,j} = 1$ for all i and j and $\Omega = [-1,1]$ and is uniformly partitioned into 10 pieces. Sampling a $g(\mathbf{x}_{t,i})$ corresponds to randomly placing a ball into a total number of 10 bins. For example, $\mathbf{x}_{t,2}$ falls into the 8-th interval so μ_8 is used to approximate $g(\mathbf{x}_{t,2})$, which may be viewed as a new ball of type μ_8 (or in 8-th bin) is created. The mean load for each bin is $d/\ell = d/10$. We calculate $\sum_{i\le d} g(\mathbf{x}_{t,i})$ by counting the balls in each bin: $\mathbf{y}_{t,1} = 5\times\mu_1 + 1\times\mu_2 + \ldots + 6\times\mu_8 + 3\times\mu_9 + 1\times\mu_{10} + \xi_{t,1}$.

Step 2. Reduction to linear regression. Each observation can be construed as a linear combination of μ_j's ($j \in [\ell]$), where $\mu_j = \mathbb{E}[g(\mathbf{x}_{t,i}) \mid \mathbf{x}_{t,i} \in \Omega_j]$. For example, $\mathbf{y}_{t,1} = \sum_{i\le d} K_{1,i}\mu_{j_i} + \xi_{t,1} + o(1)$, where $\mathbf{x}_{t,i} \in \Omega_{j_i}$, and in general, we have

$$\mathbf{y}_{t,i} = \sum_{j\le\ell} L_{(t,i),j}\mu_j + \xi_{t,i} + o(1), \qquad (2)$$

where $L_{(t,i),j} = \sum_{m\in\mathcal{L}_{t,j}} K_{i,m}$ and $\mathcal{L}_{t,j} = \{m : \mathbf{x}_{t,m} \in \Omega_j\}$.

Therefore, our learning problem reduces to a linear regression problem, in which the $L_{(t,i),j}$'s are features and the $\{\mu_j\}_{j\le\ell}$ are coefficients to be learned.

Step 3. Moment-based estimation. An MSE-based estimator is consistent but finding its confidence interval (error bound) requires knowing the spectrum of the features' covariance matrix, which is remarkably difficult in our setting. Therefore, we propose a moment-based algorithm with provable performance (FLIPSIGN in Alg. 1).

We illustrate each steps above through a toy example, in which we assume $K_{i,j} = 1$ for all i and j so the model simplifies to $\mathbf{y}_{t,1} = \sum_{j\le d} g(\mathbf{x}_{t,j}) + \xi_{t,1}$. See Fig. 3 for additional details.

Steps 1 & 2. First, we view the generation of samples as a balls-and-bins process so that the $g(\cdot)$-estimation problem reduces to a regression problem (Steps 1 & 2). Specifically, we generate $(\mathbf{y}_{t,1}, \{\mathbf{x}_{t,i}\}_{i\le d})$ as first sequentially sampling $\{\mathbf{x}_{t,i}\}_{i\le d}$ and computing the corresponding $g(\mathbf{x}_{t,i})$, then summing each term up together with $\xi_{t,1}$ to produce $\mathbf{y}_{t,1}$. When an $\mathbf{x}_{t,i}$ is sampled, it falls into one of Ω_i's with uniform probability. Let j_i be the bin that $\mathbf{x}_{t,i}$ falls into. Then $g(\mathbf{x}_{t,i})$ is approximated by μ_{j_i} according to Step 1. Thus, we may view a ball of "type μ_{j_i}" (or in j_i-th bin) is created. For example, in Fig. 3, $\mathbf{x}_{t,2}$ falls into the 8-th interval so a ball is added in the 8-th bin. After all $\mathbf{x}_{t,i}$'s are sampled,

compute $\mathbf{y}_{t,1}$ by counting the numbers of balls in different bins. Recalling that the load of j-th bin is $L_{(t,1),j}$, we have $\mathbf{y}_{t,1} \approx \sum_{j\le d} L_{(t,1),j}\cdot\mu_j + \xi_{t,1}$. Let $\Delta_{t,j} = L_{(t,1),j} - d/\ell$ and using that $\mathbb{E}[L_{(t,1),j}] = d/\ell$ and $\sum_{j\le d}\mu_j = 0$, we have

$$\mathbf{y}_{t,1} = \Delta_{t,1}\mu_1 + \cdots + \Delta_{t,\ell}\mu_\ell + \xi_{t,1}. \quad (3)$$

Eq. (3) is a standard (univariate) regression: for each t, we know $\mathbf{y}_{t,1}$, and know all $\Delta_{t,j}$'s because all $\mathbf{x}_{t,j}$'s are observed so the number of balls in each bin can be calculated. We need to estimate the unknown μ_j's. Note that $\mathbb{E}[\Delta_{t,j}] = 0$.

Steps 3. We solve the regression (Step 3). Our algorithm "tweaks" the observations so that the features associated with μ_1 are always positive: let $b_{t,1} = 1$ if $\Delta_{t,1} > 0$ and -1 otherwise. Multiply $b_{t,1}$ to both sides of Eq. (3) for each t,

$$b_{t,1}\mathbf{y}_{t,1} = |\Delta_{t,1}|\mu_1 + \cdots + b_{t,1}\cdot\Delta_{t,\ell}\cdot\mu_\ell + b_{t,1}\xi_{t,1}. \quad (4)$$

We sum up the LHS and RHS of (4) and obtain

$$\sum_{t\le n} b_{t,1}\mathbf{y}_{t,1} = \big(\sum_{t\le n}|\Delta_{t,1}|\big)\mu_1 + \cdots + \big(\sum_{t\le n} b_{t,1}\cdot\Delta_{t,\ell}\big)\mu_\ell + \sum_{t\le n} b_{t,1}\xi_{t,1}. \quad (5)$$

Next, we have $\sum_{t\le n}|\Delta_{t,1}| = \Theta(n)$ whp. Also, we can see that $b_{t,1}$ and $\Delta_{t,j}$ are "roughly" independent for $j \neq 1$ (careful analysis will make it rigorous). Therefore, for any $j \neq 1$, $\mathbb{E}[b_{t,1}\cdot\Delta_{t,j}] = 0$, and thus $\sum_{t\le n} b_{t,1}\cdot\Delta_{t,j} = O(\sqrt{n})$ whp. Now (5) becomes $\sum_{t\le n} b_{t,1}\cdot\mathbf{y}_{t,1} = \big(\sum_t|\Delta_{t,1}|\big)\mu_1 + O(\ell\cdot\sqrt{n})$. Thus our estimator is $\hat{\mu}_1 \triangleq \frac{\sum_t b_{t,1}\cdot\mathbf{y}_{t,1}}{\big(\sum_t|\Delta_{t,1}|\big)} = \mu_1 + \frac{O(\ell\cdot\sqrt{n})}{\Theta(n)} = \mu_1 + O\big(\frac{\ell}{\sqrt{n}}\big)$. Here, the covariance analysis for the $\Delta_{t,j}$'s is circumvented because $\Delta_{t,j}$'s interactions are compressed into the term $O\big(\frac{\ell}{\sqrt{n}}\big)$. We remark that the above analysis contains some crude steps and can be tightened up, as we have done in App. C.

Technique 3. Learn $g(\cdot)$ using boosting. In the univariate setting, we have $\mathbf{y}_{t,i} = \sum_{m\le b} g_m(\mathbf{x}_{t,i}) + \xi_{t,i}$, in which each $g_m(\mathbf{x}_{t,i})$ is a weak learner. Standard boosting algorithms, such as (Quinlan 1986; Chen and Guestrin 2016), assume that each $g_m(\cdot)$ is represented by a regression tree and constructed sequentially. A greedy strategy is used to build a new tree, e.g., iteratively splitting a node in a tree by choosing a variable that optimizes prediction improvement. In our setting, $\mathbf{y}_{t,i}$ depends on evaluating $g_m(\cdot)$ at d different locations $\mathbf{x}_{t,1},\ldots,\mathbf{x}_{t,d}$, so the splitting procedure either is d (e.g. 3000 for equity market) times slower in a standard implementation, or requires excessive engineering tweak of existing systems.

Here, we propose a simple and effective weak learner based on the intuition of the tree structure. Let

$$(\mathbf{x}_t)_i = \big((\mathbf{x}_{t,1})_i, (\mathbf{x}_{t,2})_i, \ldots, (\mathbf{x}_{t,d})_i\big) \in \mathbf{R}^d,$$
$$(\mathbf{x}_t)_{i,j} = \big((\mathbf{x}_{t,1})_i\cdot(\mathbf{x}_{t,1})_j, \ldots, (\mathbf{x}_{t,d})_i\cdot(\mathbf{x}_{t,d})_j\big) \in \mathbf{R}^d,$$

and $(\mathbf{x}_t)_{i,j,k}$ can be defined in a similar manner. We observe that regression trees used in GBRT models for equity return are usually shallow and can be *linearized*: we may unfold a tree into disjunctive normal form (DNF) (Abasi, Bshouty, and Mazzawi 2014), and approximate the DNF by a sum of multiple interaction terms, e.g., $I((\mathbf{x}_{t,i})_1 > 0)\cdot I((\mathbf{x}_{t,i})_2 > 0)$ can be approximated by $(\mathbf{x}_{t,i})_1\cdot(\mathbf{x}_{t,i})_2$.

Our algorithm, namely LIN-PVEL (linear projected vector ensemble learner), consists of weak learners in linear forms. Each linear learner consists of a subset of features and their interactions. The number of features included and the depth of their interactions are hyper-parameters corresponding to the depth of the decision tree. For example, if the first three features are included in the learner, we need to fit $\mathbf{y}_{t,i}$ against

$$\sum_{j\in[d]} \underbrace{\hat{K}_{i,j}}_{\text{given}}\cdot\Big[\underbrace{\beta_1(\mathbf{x}_{t,j})_1 + \ldots}_{\text{linear terms}}\underbrace{+\beta_4(\mathbf{x}_{t,j})_{1,2} + \cdots + \beta_7(\mathbf{x}_{t,j})_{1,2,3}}_{\text{interaction terms}}\Big], \quad (6)$$

by MSE. Conceptually, although we use linearized models to approximate the trees, the "target" trees are unavailable (for the computational efficiency reasons above). We need a new procedure to select features for each learner. Our intuition is that, if an interaction term could have predictive power, each feature involved in the interaction should also have predictive power. Our procedure is simply to select a fixed number of i's with the largest $\mathrm{corr}((\mathbf{y}_{\mathrm{Res}})_t, \hat{K}(\mathbf{x}_t)_i)$, where $(\mathbf{y}_{\mathrm{Res}})_t$ is the residual error.

Using feature interactions to approximate DNF ($I((\mathbf{x}_{t,i})_1 > 0)\cdot I((\mathbf{x}_{t,i})_2 > 0) \approx (\mathbf{x}_{t,i})_1\cdot(\mathbf{x}_{t,i})_2$) may not always be accurate, however, in our setting, linear interaction models often outperform decision trees or DNFs. We believe this occurs because interaction terms are continuous (whereas DNFs are discrete functions), and thus they are more suitable to model smooth changes.

Evaluation

We evaluate our algorithms on two real-world data sets: an equity market to predict stock returns, and a social network data set to predict user popularity, respectively. Additional details and experiments for the equity market and Twitter data sets are in APP. G. We remark that this is a *theoretical* paper; examining the performance on more data sets and baselines is a promising direction for future work.

Models under our framework. We estimate K and $g(\cdot)$ separately. To estimate K, we use both the algorithm discussed in Sec. 4 and other refinements discussed in App. B. To estimate $g(\cdot)$, we use SGD-based algorithms (MLP and LSTM), nparam-gEST, and LIN-PVEL.

Baselines. Our baselines include the commonly used models and domain specific models. *(i) The UMs* include linear, MLP, LSTM, GBRT, and SFM (Zhang, Aggarwal, and Qi 2017). We also implement a "poor man's version" of both LIN-PVEL and nparam-gEST for UM, which assumes that influences from other entities are 0; *(ii) The CEMs* include a standard linear VAR (Negahban and Wainwright 2011), ARRR (Wu et al. 2020a). *(iii) Ad-hoc integration* AlphaStock (Wang et al. 2019), and HAN (Hu, Liu et al. 2018) for the equity data set; Node2Vec (Grover and Leskovec 2016) for the Twitter data set.

Predicting equity returns. We use 10 years of equity data from an emerging market to evaluate our algorithms and focus on predicting the next 5-day returns, for which the last three years are out-of-sample. The test period is substantially

	Universe 800				*Full universe*				Backtesting	
Models	corr	w_corr	t-stat	w_t-stat	corr	w_corr	t-stat	w_t-stat	PnL	Sharpe
Ours: LIN-PVEL	**0.0764**	**0.0936**	**6.7939**	**6.3362**	**0.0944**	**0.1009**	**8.2607**	**6.4435**	**0.5261**	**10.97**
Ours: nparam-gEST	**0.0446**	**0.0320**	**3.2961**	**1.5753**	**0.0618**	**0.0553**	**5.7327**	**3.5212**	**0.3386**	**7.59**
Ours: MLP	**0.0550**	**0.0567**	**6.4782**	**5.0172**	**0.0738**	**0.0692**	**9.2034**	**6.4151**	**0.4202**	**9.43**
Ours: LSTM	**0.0286**	**0.0347**	**3.4517**	**3.0261**	**0.0473**	**0.0491**	**6.3615**	**4.2385**	**0.2487**	**7.10**
UM: poor man Lin-PVEL	0.0674	0.0866	6.0947	5.7312	0.0827	0.0884	7.4297	5.6659	0.4565	9.76
UM: poor man nparam-gEST	0.0432	0.0309	3.1505	1.4912	0.0584	0.0509	5.0098	3.0844	0.3070	6.59
UM: MLP	0.0507	0.5050	6.0234	4.4966	0.0606	0.0467	8.2857	4.4555	0.2782	6.38
UM: LSTM	0.0178	0.0200	2.2136	1.8077	0.0352	0.0297	4.0602	2.3619	0.175	4.33
UM: Linear models	0.0106	0.0192	1.6471	2.3030	0.0290	0.0251	4.4711	2.6010	0.1888	4.79
UM: GBRT	0.0516	0.0591	7.5739	5.6310	0.0673	0.0747	9.3379	7.8931	0.3858	4.45
UM: SFM	0.0027	0.0032	0.4688	0.4050	0.0147	0.0051	1.2683	0.3892	0.0169	0.54
Existing CEM: VR	0.0156	0.0159	2.4997	1.7046	0.0041	-0.0025	0.8847	-0.3021	0.0430	1.20
Existing CEM: ARRR	0.0314	0.0382	2.5336	2.4213	0.0222	0.0273	1.8557	1.8968	0.1674	3.24
Ad-hoc integration: AlphaStock	0.0085	0.0063	2.1045	1.2516	0.0027	0.0032	0.4688	0.4050	0.0045	0.10
Ad-hoc integration: HAN	0.0105	0.0081	1.7992	1.0017	0.0080	0.0050	1.5716	0.7340	0.0570	2.02
Consolidated: All Ours	**0.0775**	**0.0950**	**6.8687**	**6.4108**	**0.0958**	**0.1025**	**8.5703**	**6.6487**	**0.5346**	**11.30**

Table 1: Summary of results for equity raw return forecasts. LIN-PVEL is the gradient boosting method with the linear learner. Boldface denotes the best performance in each group. Backtesting results pertain to the *Full universe*.

longer than those employed in recent works (Zhang, Aggarwal, and Qi 2017; Hu, Liu et al. 2018; Li et al. 2019), adding to the robustness of our results. We constructed 337 standard technical factors to serve as a feature database for all models. We consider two universes: (i) *Universe 800* can be construed as an equivalence to the S&P 500 in the US, and consists of 800 stocks, and (ii) *Full universe* consists of all stocks except for the very illiquid ones. Visualizations are shown in App. G.

We next describe our evaluation metrics and argue why they are more suitable and different from those employed in standard ML problems (see App. G) • *(i) Correlation vs MSE.* While the MSE is a standard metric for regression problems, correlations are better-suited metrics for equity data sets (Zhou and Jain 2014). • *(ii) Significance testing.* The use of t-statistics estimators (Newey and West 1986) can account for the serial and cross-sectional correlations (App. G) • *(iii) Stock capacity/liquidity considerations.* Predicting illiquid stocks is less valuable compared to predicting liquid ones because they cannot be used to build large portfolios. We use a standard approach to weight correlations (w_corr) and t-statistics by a function of historical *notional (dollar) traded volume* to reflect the capacity of the signals.

Results. See Table 1 for the results and the simulated Profit & Loss (PnL). The experiments confirm that • *(i)* Models under our framework consistently outperform prior works. In addition, our LIN-PVEL model has the best performance; • *(ii)* By using a simple consolidation algorithm, the aggregated signal outperforms all individual ones. Our new models pick up signals that are orthogonal to existing ones because we rely on a new mechanism to use stock and feature interactions.

Predicting user popularity in social networks. We use a Twitter data set to build models for predicting a user's *next 1-day* popularity, defined as the sum of retweets, quotes, and replies received by the user. We collected 15 months of Twitter data streams related to US politics. In total, there are 804 million tweets and 19 million distinct users. User u has one interaction if and only if he or she is retweeted/replied/quoted by another user v. Due to the massive scale, we extract the subset of 2000 users with the most interactions, for evaluation purposes. For each user, we compute his/her daily popularity for 5 days prior to day t as the features.

Results. We report the MSE and correlation for both *in-sample* and *out-of-sample* in Table 2. We observe the consistent results with equity return experiments: *(i)* Methods under our framework achieve better performance in out-of-sample MSE and correlation, with LIN-PVEL attaining the overall best performance. *(ii)* Our methods yield the best generalization error by having a much smaller gap between training and test metrics.

Models	MSE (in)	MSE (out)	Corr (in)	Corr (out)
Ours: Lin-PVEL	0.472	**0.520**	0.733	**0.712**
Ours: nparam-gEST	0.492	**0.559**	0.688	**0.658**
Ours: MLP	0.486	**0.547**	0.716	**0.692**
Ours: LSTM	0.484	**0.541**	0.724	**0.703**
UM: Poor man Lin-PVEL	0.488	0.552	0.710	0.684
UM: Poor man nparam-gEST	0.544	0.584	0.634	0.605
UM: Poor man MLP	0.506	0.562	0.703	0.673
UM: Poor man LSTM	0.496	0.559	0.710	0.679
UM: Linear models	0.616	0.663	0.618	0.592
UM: Random forest	0.611	0.659	0.623	0.587
UM: Xgboost	0.530	0.571	0.671	0.647
CEM: VR	0.540	0.729	0.649	0.408
CEM: ARRR	0.564	0.652	0.610	0.573
Ad-hoc: Node2Vec	0.537	0.690	0.693	0.468
Consolidated: All Ours	**0.459**	**0.502**	**0.767**	**0.742**

Table 2: Overall in-sample and out-of-sample performance on the Twitter data set. Boldface denotes the best performance in each group.

Conclusion

This paper revisits the problem of building machine learning algorithms that involve interactions between entities. We propose an *additive influence framework* that enables us to decouple the learning of the entity-interactions from the learning of feature-response interactions. Our upstream entity interaction learner has provable performance guarantees, whereas our downstream $g(\cdot)$-learners can leverage a wide set of effective ML techniques. All these methods under our framework are proven to be superior to the existing baselines.

Acknowledgement

We thank anonymous reviewers for helpful comments and suggestions. Jian Li was supported in part by the National Natural Science Foundation of China Grant 62161146004, Turing AI Institute of Nanjing and Xi'an Institute for Interdisciplinary Information Core Technology. Yanhua Li was supported in part by NSF grants IIS-1942680 (CAREER), CNS-1952085, CMMI- 1831140, and DGE-2021871. Zhenming Liu and Qiong Wu were supported by NSF grants NSF-2008557, NSF-1835821, and NSF-1755769.

Contents

Additional notes on problem definition

Independence of $\mathbf{x}_{t,i}$. Our analysis assumes that $\mathbf{x}_{t,i}$ are independent across t's and i's. Our discussion assumes that $\mathbf{x}_{t,i} \in \mathbf{R}$. The arguments can easily generalize to multi-dimensional $\mathbf{x}_{t,i}$. When $\mathbf{x}_{t,i}$ are correlated across stocks, we can apply a factor model to obtain

$$\mathbf{x}_t = L\mathbf{f}_t + \tilde{\mathbf{x}}_t, \tag{7}$$

where $\mathbf{x}_t = (\mathbf{x}_{t,1}, \mathbf{x}_{t,2}, \ldots, \mathbf{x}_{t,d}) \in \mathbf{R}^d$, $\mathbf{f}_t$ is a low-dimensional vector that explains the co-moving (correlated) components, L is the factor loading matrix, and $\tilde{\mathbf{x}}_t = (\tilde{\mathbf{x}}_{t,1}, \ldots, \tilde{\mathbf{x}}_{t,d}) \in \mathbf{R}^d$ is the idiosyncratic component. There exists a rich literature on algorithms that identify latent factors (Colby and Meyers 1988; Fama and French 1993; Hurst, Black, and Simaika 1965; Kakushadze 2016). The shared factors driving the co-movements of the features can be utilized in other ways to forecast equity returns (Ming et al. 2014). We can use the idiosyncratic component $\tilde{\mathbf{x}}_t$ as input features in our model, since the coordinates in $\tilde{\mathbf{x}}_t$ are independent. In the setting where serial correlation is presented in $\tilde{\mathbf{x}}_t$, one can use the standard differencing operator for decorrelating purposes (Hamilton and Tegmark 2000).

Estimation of K

We prove Proposition 4.1 and explain other variations of estimating K. For exposition purposes, our analysis focuses on the case where κ is Gaussian kernel or IMQ. The case for κ being an inner product function can be analyzed in a similar manner. See also Remark at the end of this section.

In Sec. B, we first describe the background (e.g., notation and building blocks) needed. In Sec. B, we present our proof for Prop 4.1. Our analysis assumes that $n \leq d^2$ to simplify calculations and ease the exposition. The case $n \geq d^2$ corresponds to the scenario when abundant samples are available, and is easier to analyze. In Sec. B, we explain additional algorithms for estimating K.

Background

Notation. Let $A = \frac{1}{d^2} K^{\mathrm{T}} K$ and $B = \frac{1}{d^2 n} Y^{\mathrm{T}} Y$. Let V_k^A be the first k eigenvectors associated with A and V_k^B be the first k eigenvectors associated with B. Note that A and B are symmetric. Let $\mathcal{P}_A = V_{i^*}^A (V_{i^*}^A)^{\mathrm{T}}$ and $\mathcal{P}_B = V_{i^*}^B (V_{i^*}^B)^{\mathrm{T}}$, where i^* is defined in Alg. 2.

Distance between matrices. For any positive-definite matrix A, there could be multiple square roots of A (the square root is defined as any matrix B such that $BB^{\mathrm{T}} = A$). Any pair of square roots of the same matrix differ only by a unitary matrix and should be considered as "the same" in most of our analysis. We adopt the following (standard) definition to measure the difference between two matrices.

Definition B.1. *(Distance between two matrices) Let $X, Y \in \mathbf{R}^{d_1 \times d_2}$. The distance between X and Y is defined as*

$$\mathrm{Dist}^2(X, Y) = \min_{W \text{ unitary}} \|XW - Y\|_F^2. \tag{8}$$

Building blocks related to distances.

Lemma B.2. *(From (Bhojanapalli, Kyrillidis, and Sanghavi 2016)) For any two rank-r matrices U and X, we have*

$$\mathrm{Dist}^2(U, X) \leq \frac{1}{2(\sqrt{2}-1)\sigma_r^2(X)} \|UU^{\mathrm{T}} - XX^{\mathrm{T}}\|_F^2.$$

Lemma B.3. *(From (Ge, Jin et al. 2017)) Let M_1 and M_2 be two matrices such that*

$$M_1 = U_1 D_1 V_1^{\mathrm{T}} \quad \text{and} \quad M_2 = U_2 D_2 V_2^{\mathrm{T}}. \tag{9}$$

It holds true that

$$\|U_1 D_1 U_1^{\mathrm{T}} - U_2 D_2 U_2^{\mathrm{T}}\|_F^2 + \|V_1 D_1 V_1^{\mathrm{T}} - V_2 D_2 V_2^{\mathrm{T}}\|_F^2 \leq 2\|M_1 - M_2\|_F^2. \tag{10}$$

Building block related to gap vs. tail.

Lemma B.4. *Let $\{\lambda_i\}_{i \geq 1}$ be a sequence such that $\sum_{i \geq 1} \lambda_i = 1$, $\lambda_i \leq c i^{-\omega}$ for some constant c and $\omega \geq 2$. Assume also that $\lambda_1 < 1$. Define $\delta_i = \lambda_i - \lambda_{i+1}$, for $i \geq 1$. Let δ_0 be a sufficiently small number, and c_1 and c_2 be two suitable constants. For any $\delta < \delta_0$, there exists an i^* such that $\delta_{i^*} \geq \delta$ and $\sum_{j \geq i^*} \lambda_j = O\left(\delta^{\frac{4}{5}}\right)$.*

Kernel learning. Let $\kappa(\mathbf{x}, \mathbf{x}')$ be a smooth radial basis function, i.e., $\kappa(\mathbf{x}, \mathbf{x}') = \kappa(\|\mathbf{x} - \mathbf{x}'\|)$, and use the notation $f(\cdot) = \kappa(\sqrt{\cdot})$. We assume that $|f^{(\ell)}(r)| \leq \ell! M^\ell$, for all ℓ sufficiently large and $r > 0$. Note that both Gaussian kernels and inverse multi-quadratic kernels satisfy this property.

Define an integral operator $\mathcal{K}$ as

$$\mathcal{K} f(\mathbf{x}) = \int \kappa(\mathbf{x}, \mathbf{x}') f(\mathbf{x}') dF(\mathbf{x}'), \tag{11}$$

where $F(\cdot)$ is the cumulative probability function over the support of $\mathbf{x}$. Let $\mathcal{H}$ be the rank of $\mathcal{K}$, which can be either finite or countably infinite. Let $\psi_1, \psi_2, \ldots, \psi_{\mathcal{H}}$ be the eigenfunctions of $\mathcal{K}$, and $\lambda_1, \lambda_2, \ldots, \lambda_{\mathcal{H}}$ be the corresponding eigenvalues such that $\lambda_1 \geq \lambda_2 \geq \ldots$. Let $K \in \mathbf{R}^{d\times d}$ be the Gram matrix such that $K_{i,j} = \kappa(\|\mathbf{z}_i - \mathbf{z}_j\|)$.

Our analysis relies on the following two key building blocks.

Lemma B.5. *((Belkin 2018)) Let λ_i^* be the i-th eigenvalue of $\mathcal{K}$. There exist constants C and C' such that*

$$\lambda_i^* \leq C' \exp(-Ci^{\frac{1}{r}}). \tag{12}$$

Lemma B.6. *Let λ_i^* be the i-th eigenvalue of $\mathcal{K}$. Let $\lambda_i(K)$ be the i-th eigenvalue of K. Let $\hat{\lambda}_j = \lambda_j(K)/d$. Let $\tau > 0$ be a tunable parameter. With probability at least $1 - \exp(-c_0\tau)$ for some constant c_0, it holds true that*

$$\left(\sum_{j\geq 1}(\lambda_j^* - \hat{\lambda}_j)^2\right)^{\frac{1}{2}} \leq 2\sqrt{\frac{\tau}{d}}. \tag{13}$$

In addition, with probability at least $1 - \exp(-c_0\tau)$,

$$\left(\sum_{j\geq 1}\left((\lambda_j^*)^2 - (\hat{\lambda}_j)^2\right)^2\right)^{\frac{1}{2}} \leq c\sqrt{\frac{\tau}{d}} \tag{14}$$

for some constant c.

Proof of Lemma B.6. Eq. 13 is from Theorem B.2 from (Tang et al. 2013). Now to prove Eq. 14, we have

$$\left(\sum_{j\geq 1}\left((\lambda_j^*)^2 - (\hat{\lambda}_j)^2\right)^2\right)^{\frac{1}{2}} = \left(\sum_{j\geq 1}(\lambda_j^* - \hat{\lambda}_j)^2(\lambda_j^* + \hat{\lambda}_j)^2\right)^{\frac{1}{2}} \leq c'\left(\sum_{j\geq 1}(\lambda_j^* - \lambda_j)^2\right)^{\frac{1}{2}} \leq c\sqrt{\frac{\tau}{d}}.$$

□

Proof for Prop 4.1

Algorithm 2 Data-driven (DD) estimation of K

Input $\mathbf{X}, \mathbf{Y}$; **Output** $\hat{K}$

1: $[V, \Sigma, V^{\mathrm{T}}] = \mathrm{svd}\left(\frac{1}{n}\mathbf{Y}^{\mathrm{T}}\mathbf{Y}\right)$
2: Let $\sigma_i = \Sigma_{i,i}$
3: $i^* = \max\{i : \sigma_{i^*} - \sigma_{i^*+1} \geq \delta d^2\}$ ▷ δ is tunable
4: **return** $\hat{K} = \mathcal{P}_{i^*}(V\Sigma^{\frac{1}{2}}V^{\mathrm{T}})$

Our analysis consists of four steps:

- **Step 1.** Show that $\frac{1}{n}\mathbf{Y}^{\mathrm{T}}\mathbf{Y} - K^{\mathrm{T}}K$ is sufficiently small.
- **Step 2.** Show that a low rank approximation of $\mathbf{Y}^{\mathrm{T}}\mathbf{Y}$ is sufficiently close to $\mathbf{Y}^{\mathrm{T}}\mathbf{Y}$.
- **Step 3.** Show that $\mathcal{P}_{i^*}(\frac{1}{n}\mathbf{Y}^{\mathrm{T}}\mathbf{Y})$ is close to $\mathcal{P}_{i^*}(\mathbf{K}^{\mathrm{T}}\mathbf{K})$.
- **Step 4.** Use results from the first three steps, together with Lemma B.4, to prove the first part of Theorem 4.1.

Step 1. $\frac{1}{n}\mathbf{Y}^{\mathrm{T}}\mathbf{Y}$ and $K^{\mathrm{T}}K$ are close. To formally prove this step, we rely on the following proposition.

Proposition B.7. *Consider the problem of learning the stock latent embedding model. Let n be the number of observations. Let $\mathbf{Y} \in \mathbf{R}^{n\times d}$ be such that $\mathbf{Y}_{i,:}$ contains the i-th observation. Assume that $n \leq d^2$ and $\sigma_\xi = O(\sqrt{d})$. With overwhelming probability, it holds true that*

$$\left\|\frac{1}{n}\mathbf{Y}^{\mathrm{T}}\mathbf{Y} - K^{\mathrm{T}}K\right\|_F = O\left(\frac{d^2\log^3 n}{\sqrt{n}}\right). \tag{15}$$

Proving Proposition B.7 requires a standard manipulation of concentration inequalities for matrices. See the proof in App. F.

Step 2. $\mathcal{P}_{i^*}(K^{\mathrm{T}}K)$ is close to $K^{\mathrm{T}}K$.

Lemma B.8. *There exists a sufficiently large d_0 so that when $d \geq d_0$, Algorithm 2 always terminates. In addition, it holds true that*

$$\sum_{i\geq i^*}\lambda_i^2\left(\frac{K}{d}\right) = O(\delta^{\frac{4}{5}}).$$

Proof. Let $\tilde{\delta} = 10\delta \geq \frac{c\log^3 d}{\sqrt{d}}$ for a suitably large c. By Lemma B.4, we have that there exists an $\tilde{i}$ such that

1. $(\lambda_{\tilde{i}}^*)^2 - (\lambda_{\tilde{i}+1}^*)^2 \geq \tilde{\delta}$.

2. $\sum_{i\geq\tilde{i}}(\lambda_i^*)^2 \leq \left(\tilde{\delta}\right)^{\frac{4}{5}}$.

We first show that the algorithm terminates. We have that

$$\left|(\lambda_i^*)^2 - \lambda_i^2(K/d)\right| = O\left(|\lambda_i^* - \lambda_i(K/d)|\right) = O\left(\sqrt{\frac{\log d}{d}}\right)$$

The last equality uses Proposition B.7. Next, by using Lemma B.4, we have

$$\left|\lambda_i\left(\frac{1}{nd^2}\mathbf{Y}^{\mathrm{T}}\mathbf{Y}\right) - \lambda_i\left(\frac{K^2}{d^2}\right)\right| = O\left(\frac{\log^3 n}{\sqrt{n}}\right).$$

Therefore, we can also see that

$$\lambda_{\tilde{i}}\left(\frac{1}{nd^2}\mathbf{Y}^{\mathrm{T}}\mathbf{Y}\right) - \lambda_{\tilde{i}+1}\left(\frac{1}{nd^2}\mathbf{Y}^{\mathrm{T}}\mathbf{Y}\right) \geq \delta.$$

Our algorithm always terminates. In addition, we have $i^* \geq \tilde{i}$. Finally, we have

$$\begin{aligned}\sum_{i\geq i^*}\lambda_i^2\left(\frac{K}{d}\right) &\leq \sum_{i\geq\tilde{i}}\lambda_i^2\left(\frac{K}{d}\right)\\ &\leq 2\left(\sum_{i\geq\tilde{i}}(\lambda_{\tilde{i}}^*)^2 + \left(\lambda_{\tilde{i}}^* - \lambda_i\left(K/d\right)\right)^2\right)\\ &= O\left(\tilde{\delta}^{\frac{4}{5}}\right) = O\left(\delta^{\frac{4}{5}}\right).\end{aligned}$$

□

Step 3. Analysis of the projection. To show that $\mathcal{P}_{i^*}\left(\frac{1}{n}\mathbf{Y}^{\mathrm{T}}\mathbf{Y}\right)$ and $\mathcal{P}_{i^*}(K^{\mathrm{T}}K)$ are close. We have the following lemma.

Lemma B.9. *Consider running Algorithm* ESTIMATE-K *in Alg. 2 for estimating K. Let $A = \frac{1}{d^2}K^{\mathrm{T}}K$ and $B = \frac{1}{d^2 n}\mathbf{Y}^{\mathrm{T}}\mathbf{Y}$. Let $\mathcal{P}_A$ and $\mathcal{P}_B$ be defined as above. Let $\epsilon \leq \frac{c_0\log^3 d}{\sqrt{d}}$ for some constant c_0, and δ be the gap parameter in Alg. 2 such that $\delta^3 = \omega(\epsilon^2)$. With high probability, we have*

$$\|\mathcal{P}_A - \mathcal{P}_B\|_2 = O\left(\frac{\|A-B\|_2}{\delta}\right). \tag{16}$$

Proof of Lemma B.9. Define $S_1 = [\lambda_{i^*}(A) - \delta/10, \infty)$ and $S_2 = [0, \lambda_{i^*}(A) + \delta/10]$. By Lemma B.7, we have $\left\|\frac{1}{nd^2}\mathbf{Y}^{\mathrm{T}}\mathbf{Y} - \frac{1}{d^2}K^{\mathrm{T}}K\right\|_F = O\left(\frac{\log^2 n}{\sqrt{n}}\right)$. Also using that $\delta \geq \frac{c\log^3 n}{\sqrt{n}}$, we have that S_1 contains the first i^* eigenvalues of A and B, whereas S_2 contains the rest of eigenvalues. We may then use a variant of the Davis-Kahan (Stewart 1990) theorem to show that

$$\|\mathcal{P}_A - \mathcal{P}_B\|_2 \leq \frac{\|A-B\|_2}{0.8\delta} \leq \frac{2\epsilon}{\delta}.$$

□

Step 4. Gluing everything. Recall that $A = \frac{1}{d^2}K^{\mathrm{T}}K$ and $B = \frac{1}{d^2 n}Y^{\mathrm{T}}Y$. Let $A_{i^*} = \mathcal{P}_A(A)(= \mathcal{P}_{i^*}(A))$ and $B_{i^*} = \mathcal{P}_{i^*}(B)$. By Lemma B.9, we have

$$\begin{aligned}\|A_{i^*} - B_{i^*}\|_F =& \|\mathcal{P}_A(A) - \mathcal{P}_B(B)\|_F,\\ =& \|\mathcal{P}_A(A) - \mathcal{P}_B(A) + \mathcal{P}_B(A) - \mathcal{P}_B(B)\|_F,\\ \leq& \|\mathcal{P}_A - \mathcal{P}_B\|_2\|A\|_F + \|A-B\|_F,\\ \leq& \Theta\left(\frac{\epsilon}{\delta} + \epsilon\right) = \Theta(\epsilon/\delta).\end{aligned}$$

Next, we define the following matrix notation

$$A_{i^*}^{\frac{1}{2}} = U_{i^*}^A(\Sigma_{i^*}^A)^{\frac{1}{2}} \quad \text{and} \quad B_{i^*}^{\frac{1}{2}} = U_{i^*}^B(\Sigma_{i^*}^B)^{\frac{1}{2}}.$$

By Lemma B.2, there exists a unitary matrix W such that

$$\left\|U_{i^*}^A(\Sigma_{i^*}^A)^{\frac{1}{2}}W - U_{i^*}^B(\Sigma_{i^*}^B)^{\frac{1}{2}}\right\|_F^2 = O\left(\frac{\epsilon^2}{\delta^3}\right). \tag{17}$$

By Lemma B.3, we obtain

$$\|U_{i^*}^A(\Sigma_{i^*}^A)^{\frac{1}{2}}U_{i^*}^A - U_{i^*}^B(\Sigma_{i^*}^B)^{\frac{1}{2}}U_{i^*}^B\|_F^2 = \left\|U_{i^*}^A(\Sigma_{i^*}^A)^{\frac{1}{2}}W - \mathcal{P}_{i^*}\left(\frac{K}{d}\right)\right\|_F^2 = O\left(\frac{\epsilon^2}{\delta^3}\right).$$

Together with $\|\mathcal{P}_{i^*}(K/d) - K/d\|_F^2 = O\left(\delta^{\frac{4}{5}}\right)$, we have

$$\begin{aligned}\left\|\frac{1}{d^2}\hat{K} - \frac{1}{d^2}K\right\|_F &= \left\|U_{i^*}^A(\Sigma_{i^*}^A)^{\frac{1}{2}}W - \frac{K}{d}\right\|_F = \left\|U_{i^*}^A(\Sigma_{i^*}^A)^{\frac{1}{2}}W - \mathcal{P}_{i^*}\left(\frac{K}{d}\right)\right\|_F + \left\|\mathcal{P}_{i^*}\left(\frac{K}{d}\right) + \frac{K}{d}\right\| \\ &= O\left(\sqrt{\frac{\epsilon^2}{\delta^3}} + \sqrt{\delta^{\frac{4}{5}}}\right).\end{aligned}$$

Remark. Our analysis relies only on the eigenvalues of $\mathcal{K}$ decaying sufficiently fast. Many other kernels, such as inner product kernels with points on the surface of a unit ball (Ha 1986; Azevedo and Menegatto 2015), also exhibit this property. In conclusion, our algorithms for estimating K can be generalized to these $\kappa(\cdot,\cdot)$ functions.

Additional estimators for K

This section explains additional possible ways to estimate K. Our intuition is that estimations of K effectively rely only on $\frac{1}{n}\mathbf{Y}^{\mathrm{T}}\mathbf{Y}$, which is the empirical covariance (aka risk) of the equities' returns/users' popularity. There are multiple ways to enhance the estimation of covariance matrix. Here, we focus on describing the estimation algorithm that is most effective in practice. We estimate K based on dynamically evolving hints. Specifically, we assume that the latent positions evolve. Let K_t be the Gram matrix at round t.

Construct K_t from Twitter dataset We faithful implementation of our algorithm discussed in Sec. 4. Moreover, we also estimate the evolving K_t for the Twitter dataset by maintain a sliding window T (e.g. we use the data samples between $t-T$ and day t) and T is a hyper-parameter to be tuned.

Construct K_t from Equity dataset. Because K_t is evolving, we may not have sufficient data to track $\hat{K}$ in the market dataset. Therefore, we derive a new algorithm to estimate $\hat{K}$ using the so-called "hint" matrices, based on two observations: *(i)* $\mathbf{Y}^{\mathrm{T}}\mathbf{Y}$ is effectively the covariance matrix of the returns. Third-party risk models such as Barra provide a more accurate estimation of the covariance matrix in practice. Thus, we may directly use the risk matrix produced by Barra as our estimation for K. *(ii)* The movements of two stocks are related because they are economically linked. It is possible to estimate these links by using fundamental and news data. Specifically, we assume that $K_t = \exp(\beta_1 K_t^{(1)} + \cdots + \beta_c K_t^{(c)})$, where $K_t^{(i)}$ $(i \le c)$ can be observed. We then need only tune β_i's to determine $\hat{K}$. Each of $K_t^{(i)}$ is considered as our "hint". We use a hint matrix $K_t^{(1)}$ constructed from Barra factor loading and a hint matrix $K^{(2)}$ constructed from news so that $\hat{K}_t = \exp(\beta_1 K_t^{(1)} + \beta_2 K^{(2)})$. The hint matrices are constructed as follows.

$K_t^{(1)}$ from Barra loading. Let $F_{t,i} \in \mathbf{R}^{10}$ be the factor exposure of the i-th stock on day t. Construct $\hat{K}_t^{(1)}$ using two standard methods.

- *Inner product.* $(\hat{K}_t^{(1)})_{i,j} = \langle F_{t,i}, F_{t,j}\rangle$.
- *Distance.* $(\hat{K}_t^{(1)})_{i,j} = \exp(-\lambda|F_{t,i} - F_{t,j}|^2)$, where λ is a hyperparameter

$K_t^{(2)}$ from News Data. We next build $K_t^{(2)}$ from the news using two steps. *Step 1.* Construct $\tilde{K}_t^{(2)} \in \mathbf{R}^{d\times d}$ such that $(\tilde{K}_t^{(2)})_{i,j}$ represents the number of news articles that mention both stock i and stock j between day $t-k$ and day t (i.e., we maintain a sliding window of k days and k is a hyper parameter). *Step 2.* Then construct $\hat{K}_t^{(2)}$ by taking a moving average of $\tilde{K}_t^{(2)}$.

Construction of $\hat{K}_t$. $\hat{K}_t$ be constructed from $\hat{K}_t^{(1)}$ (produced from Barra data) or $\hat{K}_t^{(2)}$ (constructed from news data set), or a consolidation of $\hat{K}_t^{(1)}$ and $\hat{K}_t^{(2)}$. We shall examine the following consolidation algorithm. Specifically, we let $\hat{K}_t = \exp(\beta\hat{K}_t^{(1)} + (1-\beta)\hat{K}_t^{(2)})$, where $\beta \in [0,1]$ and β is a hyperparameter.

Estimating $g(\cdot)$ with non-parametric methods

This section proves Proposition 4.2, i.e., we describe our non-parametric algorithm (nparam-gEST) for $\mathbf{x}_{t,i} \in \mathbf{R}^{O(i)}$ and analyze its performance. Assume that the probability cumulative density function $F_x(\cdot)$ of $\mathbf{x}_{t,i}$ is known. In practice, this can be substituted by standard non-parametric density estimation methods (Tsybakov 2008).

We first describe a high-level roadmap of our algorithm analysis and then proceed to present the full analysis.

Overview of our algorithms

As shown in Alg. 1, our algorithm consists of three steps.

- *Step 1.* Partition the feature space $[-1,1]^k$ into $\{\Omega_j\}_{j\le\ell}$ so that $\Pr[\mathbf{x}_{t,i} \in \Omega_j]$ are equal for all j.
- *Step 2.* Reduce the original problem to a linear regression problem.

• *Step 3.* Implement the *FlipSign* algorithm for the scenario when only an estimated $\hat{K}$ available.

We first comment on Steps 1 and 2. Then we explain the challenges in implementing the FlipSign idea, as well as our solution.

Step 1. Construction of $\{\Omega_j\}_{j\le\ell}$. We use a simple algorithm to find axis-parallel Ω_j's so that $\Pr[\mathbf{x}_{t,i}\in\Omega_j]$ is uniform for all j. Recall that we assume that the cumulative probability function of $\mathbf{x}_{t,i}$ is known (denoted as $F_{\mathbf{x}}(\cdot)$).

We describe the method for the case $k=1,2$ (recall that k is the dimension of the feature $\mathbf{x}_{t,i}$). Extensions to the case where $k\ge 3$ can be easily generalized. When $k=1$, each Ω_j is simply an interval, and thus we only need to find $\{\mathbf{x}_1=-1,\mathbf{x}_2,\ldots,\mathbf{x}_{\ell+1}=1\}$ such that $F_{\mathbf{x}}(\mathbf{x}_{t+1})-F_{\mathbf{x}}(\mathbf{x}_t)=1/\ell$ for all $1\le t\le\ell$. For example, note that in the $k=1$ case, for $l=4$, the recovered values $\{\mathbf{x}_1,\ldots,\mathbf{x}_5\}$ are simply identified with the usual quantiles of the distribution.

Step 2. We next explain how the original problem can be reduced to a set of regression problems. Using MAP-REGRESS (line 8 in Alg. 1). Recalling that for any $\mathbf{y}_{t,i}=\sum_{j\le d}K_{i,j}g(\mathbf{x}_{t,j})+\xi_{t,i}$ (with fixed i and t), we can approximate it as $\mathbf{y}_{t,i}=\sum_{j\le d}K_{i,j}\tilde{g}(\mathbf{x}_{t,j})+\xi_{t,i}$. We may then re-arrange the terms and obtain

$$\mathbf{y}_{t,i}\approx\sum_{j\le\ell}L_{(t,i),j}\mu_j+\xi_{t,i},\ \text{where } L_{(t,i),j}=\sum_{m\in\mathcal{L}_{t,j}}K_{i,m}\text{ and }\mathcal{L}_{t,j}=\{m:\mathbf{x}_{t,m}\in\Omega_j\}. \tag{18}$$

Here, $\{\mu_j\}_{j\le\ell}$ are unknown coefficients whereas $\mathbf{y}_{t,i}$ and $L_{(t,i),j}$ are observable.

Note that for any fixed t, there is a total number of d observations (i.e., $\{(\mathbf{y}_{t,i},\{L_{(t,i),j}\}_{i\le d})\}_{i\le d}$ and these observations are all correlated: $L_{(t,i),j}$ and $L_{(t,i'),j}$ depend on the same set $\mathcal{L}_{t,j}$. So our algorithm chooses only one i for each fixed t.

Sec 4 asserts that we can use the same i for different t when we have accurate information on K. In practice, we have only an estimate $\hat{K}$ of K. In addition, the estimation quality for any fixed i depends on $\|K_{i,:}-\hat{K}_{i,:}\|_F^2$. We do not know a priori which row of $\hat{K}$ is more accurate, although we know that on average, $\hat{K}$ is sufficiently close to K (i.e., $\frac{1}{d^2}\|K-\hat{K}\|_F^2=o(1)$ from Proposition 4.1). To avoid the same "bad" i being picked up repeatedly, we run a randomized procedure: let q_t be a random number from $[d]$. We use the observations $\{(\mathbf{y}_{t,q_t},\{\mathcal{L}_{(t,q_t),1}\}\}_{t\le n}$ to learn the variables μ_1.

Implementing the FlipSign algorithm

Building a robust estimator. We focus on estimating μ_1 (See Line 15 in Algorithm 1). The estimations for other μ_i's are the same. Let b_{t,q_t} be the sign of $L_{(t,q_t),1}-d/\ell$ but we only observe an estimate of $L_{(t,q_t),1}$ (referred to as $\hat{L}_{(t,q_t),1}$ in the forthcoming discussion). A major error source is that when $L_{(t,q_t),1}$ gets too close to d/ℓ, the sign of $\hat{L}_{(t,q_t),1}$ can be different from $L_{(t,q_t),1}$ (i.e., b_{t,q_t} is calculated incorrectly). We slove this problem by keeping only the observations when $|\hat{L}_{(t,q_t),1}-d/\ell|$ is large. Specifically, let

$$\Pi_1^{(q_t)}(t)\triangleq\sum_{k\in\mathcal{L}_{t,1}}K_{q_t,k}-\left(\sum_{k\notin\mathcal{L}_{t,1}}K_{q_t,k}\right)\frac{1}{\ell-1}, \tag{19}$$

and let $\hat{\Pi}_1^{(q_t)}(t)$ be computed using the estimate $\hat{K}$. We now define a robust variable $\tilde{b}_{t,q_t}$ to control the estimator

$$\tilde{b}_{t,q_t}=\begin{cases}1 & \text{if }\Pi_1^{(q_t)}(t)\ge\frac{c}{\log d}\sqrt{\frac{d}{\ell}}\\ -1 & \text{if }\Pi_1^{(q_t)}(t)<-\frac{c}{\log d}\sqrt{\frac{d}{\ell}}\\ 0 & \text{otherwise.}\end{cases} \tag{20}$$

In this case, the chance of obtaining an incorrect $\tilde{b}_{t,q_t}$ (i.e., $\hat{\Pi}_1^{q_t}(t)>\frac{c}{\log d}\sqrt{\frac{d}{\ell}}$ but $\Pi_1^{(q_t)}\le-\frac{c}{\log d}\sqrt{\frac{d}{\ell}}$ or vice verse) is significantly reduced (see line 19 in Alg. 1).

Analysis of the estimator. Recall that $\{\Omega_j\}_{j\le\ell}$ is a partition such that $\Pr[\mathbf{x}_{t,i}\in\Omega_j]$ is uniform for all j. We first formalize the "ideal" μ_j that we want to track. Specifically, let $\mu_j=\mathbb{E}[g(\mathbf{x}_{t,i})\mid\mathbf{x}_{t,i}\in\Omega_j]$. Our error analysis aims to track $\{\mu_j\}_{j\le\ell}$ (i.e., we aim to find $(\hat{\mu}_j-\mu_j)^2$). We then articulate $\tilde{g}(\mathbf{x})$ as

$$\tilde{g}(\mathbf{x})=\mu_j,\ \text{where }\mathbf{x}\in\Omega_j.$$

Our analysis consists of two parts.

Part 1. Analysis of a stylized model. We analyze a model in which the observations are assumed to be generated from

$$\mathbf{y}_{t,i}=\sum_{j\le d}K_{i,j}\tilde{g}(\mathbf{x}_{t,j})+\xi_{t,i}, \tag{21}$$

where $K_{i,j}$ is assumed to be known.

Next, we analyze Alg. 1 when it is executed over this stylized model with the assumption that K is given.

Part 2. Analysis of the original problem with $g(\cdot)$ and unknown K. When we run Alg. 1 over the original process, we need to analyze two perturbations (deviations):

1. $\mathbf{y}_{t,i}$ is generated through $g(\cdot)$, instead of $\tilde{g}(\cdot)$.

2. Our algorithm uses only an estimate of K.

Warm-up and notation. Before proceeding, let us introduce additional notation. Recall that $\mathcal{L}_{t,j} = \{k : \mathbf{x}_{t,k} \in \Omega_j\}$, i.e., the set of $\mathbf{x}_{t,k}$ that falls into the j-th bin Ω_j on time t. Also, recall that

$$L_{(t,i),j} = \sum_{k \in \mathcal{L}_{t,j}} K_{i,k}.$$

We have

$$\mathbf{y}_{t,i} = \sum_{j \leq d} L_{(t,i),j}\mu_j + \xi_{t,i} = \left(\sum_{k \in \mathcal{L}_{t,1}} K_{i,k}\right)\mu_1 + \sum_{k \notin \mathcal{L}_{t,1}} K_{i,k}\tilde{g}(\mathbf{x}_{t,k} \mid \mathbf{x}_{t,k} \notin \Omega_1) + \xi_{t,i}.$$

We interpret the meaning of the above equation. We treat $\mathbf{x}_{t,k}$ and $\mathcal{L}_{t,j}$ as random variables and the $\mathcal{L}_{t,j}$'s are measurable by $\mathbf{x}_{t,i}$. We imagine that an observation is generated by using the following procedure:

- Step 1. Generate $\mathcal{L}_{t,1}$. That is, we determine the subset of "balls" (those $\mathbf{x}_{t,i}$ for a fixed t) that fall into Ω_1.
- Step 2. Generate the rest of $\mathbf{x}_{t,k}$ for $k \notin \mathcal{L}_{t,1}$ sequentially. This corresponds to the terms $\sum_{k \notin \mathcal{L}_{t,1}} K_{i,k}\tilde{g}(\mathbf{x}_{t,k} \mid \mathbf{x}_{t,k} \notin \Omega_1)$. $\mathbf{x}_{t,k}$ is sampled from the conditional distribution $\mathbf{x}_{t,k} \mid \mathbf{x}_{t,k} \notin \Omega_1$. This explains why we write $\tilde{g}(\mathbf{x}_{t,k} \mid \mathbf{x}_{t,k} \notin \Omega_1)$.
- Step 3. After $\mathcal{L}_{t,1}$ and $\mathbf{x}_{t,k} \mid \mathbf{x}_{t,k} \notin \Omega_1$ are fixed, we generate $\mathbf{y}_{t,i}$ using the stylized model.

Let

$$\tilde{g}_j(\mathbf{x}_{t,i}) = \tilde{g}(\mathbf{x}_{t,i}) - \mathbb{E}[\tilde{g}(\mathbf{x}_{t,i}) \mid \mathbf{x}_{t,i} \notin \Omega_j] = \tilde{g}(\mathbf{x}_{t,i}) + \frac{\mu_j}{\ell - 1}.$$

We have

$$\begin{aligned}\mathbf{y}_{t,i} &= \underbrace{\left(\sum_{k \in \mathcal{L}_{t,1}} K_{i,k} - \left(\sum_{k \notin \mathcal{L}_{t,1}} K_{i,k}\right)\frac{1}{\ell - 1}\right)}_{\Pi_1^{(i)}(t)}\mu_1 + \underbrace{\sum_{k \in \mathcal{L}_{t,1}} K_{i,k}\tilde{g}_j(\mathbf{x}_{t,i} \mid \mathbf{x}_{t,i} \notin \Omega_1)}_{\Pi_2^{(i)}(t)} + \xi_{t,i} \\ &= \Pi_1^{(i)}(t)\mu_1 + \Pi_2^{(i)}(t) + \xi_{t,i}.\end{aligned} \tag{22}$$

We use the following abbreviation.

- $\hat{b}_t$ is an abbreviation for $\hat{b}_{t,q_t}$.
- $\Pi_1(t)$ is an abbreviation for $\Pi_1^{(q_t)}(t)$.
- $\Pi_2(t)$ is an abbreviation for $\Pi_2^{(q_t)}(t)$.

Let $\Delta = \hat{K} - K \in \mathbf{R}^{d \times d}$ and $\Delta^2(q_t) = \sum_{i \leq d} \Delta_{q_t,i}^2$. Let $\hat{\mathcal{B}} = \{t \in [n] : \hat{b}_{t,q_t} = 1\}$. Also, let

$$s_t = \begin{cases} 1 & \text{if } \Pi_1(t) > 0 \\ -1 & \text{otherwise.} \end{cases} \tag{23}$$

Part 1. Analysis of the stylized model Our main lemma in this section is an anti-concentration result on $\Pi_1^{(i)}(t)$ for any i and t.

Lemma C.1. *Let $\ell = O(d/\log^2 d)$. There exist constants c_0 and c_1 such that*

$$\Pr\left[|\Pi_1^{(i)}(t)| \geq \frac{c_0}{\log d}\sqrt{\frac{d}{\ell}}\right] \geq c_1$$

The probability is over the random tosses of $\{\mathbf{x}_{t,i}\}_{i \leq d}$.

Proof. We use a random-walk interpretation of $\Pi_1^{(i)}$. For each $k \in [d]$, with probability $1/\ell$, it (i.e., $\mathbf{x}_{t,k}$) falls into $\mathcal{L}_{t,j}$. When this happens, $\Pi_1^{(i)}(t)$ is incremented by $K_{i,k}$. With probability $1 - 1/\ell$, it does not fall into $\mathcal{L}_{t,j}$. In this case, $\Pi_1^{(i)}(t)$ is decremented by $K_{i,t}/(\ell - 1)$.

We define a sequence $\{Z_k\}_{k \leq d}$ to clarify the random-walk interpretation.

$$Z_k = \begin{cases} K_{i,k} & \text{with probability } \frac{1}{\ell}. \\ -\frac{K_{i,k}}{\ell - 1} & \text{with probability } 1 - \frac{1}{\ell}. \end{cases}$$

We couple $\Pi_1^{(i)}(t)$ with $\{Z_i\}_{i \leq d}$ such that $\Pi_1^{(i)}(t) = \sum_{k \leq d} Z_k$. Apply Lemma F.4 (a folklore that generalizes Littlewood-Offord-Erdős) to prove our Lemma. □

Part 2. Analysis of the original problem with $g(\cdot)$ and unknown K Our analysis consists of three components.

Part 2.1. Building blocks. We develop the essential building blocks needed in our analysis.

Part 2.2. Using $\hat{K}$. We show that when K is substituted by $\hat{K}$, the error of the estimator is well-managed.

Part 2.3. Using $g(\cdot)$. We show that when $\tilde{g}(\cdot)$ is substituted by $g(\cdot)$, not much additional error is introduced.

Part 2.1. Building blocks. We start with a variance-based Chernoff bound (Chung and Lu 2006).

Theorem C.2. *Suppose that X_i are independent random variables satisfying $X_i \le M$ for $1 \le i \le n$. Let $X = \sum_{i=1}^n X_i$ and $\|X\| = \sqrt{\sum_{i=1}^n \mathbb{E}[X_i^2]}$. Then we have*

$$\Pr[X \ge \mathbb{E}[X] + \lambda] \le \exp\left(-\frac{\lambda^2}{2(\|X\|^2 + M\lambda/3}\right). \tag{24}$$

Lemma C.3. *Consider running Alg. 1 to learn the stylized model. Let $\hat{K}$ be such that $|\hat{K} - K|_F^2 \le \gamma d^2$, for $\gamma = o(1)$. Let $\Pi_1^{(i)}(t)$ and $\hat{\Pi}_1^{(i)}(t)$ be those defined around Eq. 19. Let $\Delta = \hat{K} - K \in \mathbf{R}^{d\times d}$ and $\Delta^2(q_t) = \sum_{i\le d} \Delta_{q_t,t}^2$. Let λ_t be any random variable that is measurable by q_t. With high probability we have*

$$\Pr\left[\left|\hat{\Pi}_1^{(q_t)}(t) - \Pi_1^{(q_t)}(t)\right| \ge \lambda_t \mid q_t\right] \le \exp\left(-\frac{\lambda_t^2}{\Delta^2(q_t)/\ell + \lambda_t/3}\right),$$

and

$$\sum_{t\le n}\left|\hat{\Pi}_1^{(q_t)}(t) - \Pi_1^{(q_t)}(t)\right| = O\left(n\log n\sqrt{\frac{\gamma d}{\ell}}\right).$$

Proof. We shall again use random-walk techniques to analyze $\left|\hat{\Pi}_i^{(q_t)}(t) - \Pi_i^{(q_t)}(t)\right|$. Let

$$Z_i = \begin{cases} \Delta_{q_t,i} & \text{with probability } \frac{1}{\ell} \\ -\frac{\Delta_{q_t,i}}{\ell-1} & \text{with probability } 1 - \frac{1}{\ell}. \end{cases}$$

We have $\mathbb{E}[Z_i^2] = O\left(\frac{\Delta_{q_t,i}^2}{\ell}\right)$, which implies that $\sqrt{\sum_{i\le d}\mathbb{E}[Z_i^2]} = \sqrt{\frac{\sum_{i\le d}\Delta_{q_t,i}^2}{\ell}}$. Also, we can use a standard way to couple Z_i's with $\hat{\Pi}_1^{(q_t)}(t)$ and $\Pi_1^{(q_t)}(t)$ such that

$$\left|\sum_{i\le d} Z_i\right| = \left|\hat{\Pi}_i^{(q_t)}(t) - \Pi_i^{(q_t)}(t)\right|.$$

By using a Chernoff bound from Theorem C.2, we have

$$\Pr\left[\left|\sum_{i\le d} Z_i\right| \ge \lambda_t \mid q_t\right] \le \exp\left(-\frac{\lambda_t^2}{\frac{1}{\ell}\left(\sum_{i\le d}\Delta_{q_t,i}^2\right) + \frac{\lambda}{3}}\right) = \exp\left(-\frac{\lambda_t^2}{\Delta^2(q_t)/\ell + \lambda_t/3}\right).$$

This proves the first part of the Lemma. Next, we set $\lambda_t = c_0(\log d)\sqrt{\frac{\Delta^2(q_t)}{\ell}}$. Then we obtain $\Pr\left[\left|\sum_{i\le d} Z_i\right| \ge \lambda_t \mid q_t\right] = \exp(-\Theta(\log^2 d))$. Now, conditioned on knowing $\{q_t\}_{t\le n}$, with high probability, we have

$$\sum_{t\le n}\left|\hat{\Pi}_1^{(i)}(t) - \Pi_1^{(i)}(t)\right| \le \sum_{t\le n}\lambda_t = \frac{\log d}{\sqrt{\ell}}\sum_{t\le n}\sqrt{\Delta^2(q_t)}.$$

Next, we give a concentration bound for $\sum_{t\le n}\sqrt{\Delta^2(q_t)}$. Let $v_i^2 = \|K_{i,:} - \hat{K}_{i,:}\|^2$. We know that $\Delta^2(q_t)$ can only take values from $v_1^2, \ldots, v_d^2$ with $\sum_{i\le d} v_i^2 = \|\hat{K} - K\|_F^2 \le \gamma d^2$. We have $\sqrt{\Delta^2(q_t)} \le \sqrt{\gamma} d$.

Again using the condition that $\|\hat{K} - K\|_2^2 \le \gamma d^2$ and Jensen's inequality, we have $\mathbb{E}[\sqrt{\Delta^2(q_t)}] \le \sqrt{\gamma} d$.

Use the Chernoff bound, we have

$$\Pr\left[\sum_{t\le n}\sqrt{\Delta^2(q_t)} \ge \mathbb{E}\left[\sum_{t\le n}\sqrt{\Delta^2(q_t)}\right] + \lambda\right] \le \exp\left(-\frac{\lambda^2}{\gamma dn + \sqrt{\gamma d}\lambda/3}\right).$$

We set $\lambda = \frac{\sqrt{\gamma dn}}{\log^2 d}$ such that the right hand side is negligible. Now with high probability we have

$$\sum_{t\le n}\left|\hat{\Pi}_1^{(i)}(t) - \hat{\Pi}_1^{(i)}(t)\right| \le \frac{\log d}{\sqrt{\ell}}\sum_{t\le n}\sqrt{\Delta^2(q_t)} = O\left(n\log n\sqrt{\frac{\gamma d}{\ell}}\right).$$

□

Fact C.1. *For any j,*

$$\mathbb{E}[g(\mathbf{x}_{t,k}) \mid \mathbf{x}_{t,k} \notin \Omega_j] = \mathbb{E}[\tilde{g}(\mathbf{x}_{t,k}) \mid \mathbf{x}_{t,k} \notin \Omega_j] = -\frac{\mu_j}{\ell - 1}$$

Proof. By our model assumption,

$$\mathbb{E}[\mathbf{s}_{t,k}] = \mathbb{E}[g(\mathbf{x}_{t,k})] = \mu_1 + \cdots + \mu_\ell = 0. \tag{25}$$

On the other hand,

$$\mathbb{E}[g(\mathbf{x}_{t,k}) \mid \mathbf{x}_{t,k} \notin \Omega_j] = \frac{\mu_1 + \cdots + \mu_{j-1} + \mu_{j+1} + \cdots + \mu_\ell}{\ell - 1} = -\frac{\mu_j}{\ell - 1}.$$

Similarly, we prove that $\mathbb{E}[g(\mathbf{x}_{t,k}) \mid \mathbf{x}_{t,k} \notin \Omega_j] = \mathbb{E}[\tilde{g}(\mathbf{x}_{t,k}) \mid \mathbf{x}_{t,k} \notin \Omega_j]$. □

Lemma C.4. *Let $\hat{\mathcal{B}} = \{t \in [n] : \hat{b}_{t,q_t} = 0\}$, where $\hat{b}_{t,q_t}$ is defined in Sec C. With high probability we have $|\hat{\mathcal{B}}| = \Omega(n)$.*

This can be shown by Lemma C.1 and a Chernoff bound.

Lemma C.5. *Let s_t be defined in Eq. 23. Recall that $\hat{\mathcal{B}} = \{t \in [n] : \hat{b}_{t,q_t} = 1\}$. We have*

$$\sum_{t \in \hat{\mathcal{B}}} \Pi_2(t) s_t = \sum_{t \in \hat{\mathcal{B}}} s_t \left(\sum_{k \notin \mathcal{L}_{t,1}} K_{q_t,k} \tilde{g}_1(\mathbf{x}_{t,k} \mid \mathbf{x}_{t,k} \notin \Omega_1) \right) = O(\sqrt{nd}).$$

Proof. Our key observation is that conditioned on $t \in \hat{\mathcal{B}}$ and $\mathbf{x}_{t,k} \notin \Omega_1$, $\tilde{g}_1(\mathbf{x}_{t,k})$'s are bounded independent zero-mean random variables. Also $|\hat{\mathcal{B}}| = \Omega(n)$ (Lemma C.1). Therefore, a standard Chernoff bound gives $\sum_{t \in \hat{\mathcal{B}}} s_t \Pi_2(t) = O(\sqrt{nd})$. □

Lemma C.6. *Recall that $\hat{\mathcal{B}} = \{t \in [n] : \hat{b}_{t,q_t} = 1\}$, we have*

$$\left| \sum_{t \in \hat{\mathcal{B}}} \hat{b}_t \Pi_1(t) - \sum_{t \in \hat{\mathcal{B}}} |\Pi_1(t)| \right| \le 2n \log^5 d \sqrt{\frac{d}{\ell}} \gamma$$

Proof. Recall that

$$s_t = \begin{cases} 1 & \text{if } \Pi_1(t) > 0 \\ -1 & \text{otherwise.} \end{cases}$$

We have

$$\left\| \sum_{t \in \hat{\mathcal{B}}} \hat{b}_t \Pi_1(t) - \sum_{t \in \hat{\mathcal{B}}} |\Pi_1(t)| \right\| \le 2 \sum_{t \in \hat{\mathcal{B}}} |\Pi_1(t)| I(\hat{b}_t \neq s_t),$$

where $I(\cdot)$ is an indicator function that sets to 1 if and only if its argument evaluates to true. Note that $I(\hat{b}_t \neq s_t)$'s are i.i.d. random variables for different t's. We compute

$$\begin{aligned}
&\Pr[I(\hat{b}_t \neq s_t)] \\
\le &\Pr[s_t = -1 \wedge \hat{b}_t = 1] + \Pr[s_t = 1 \wedge \hat{b}_t = -1] \\
= &\Pr\left[\Pi_1(t) < 0 \wedge \hat{\Pi}_1(t) > \frac{c_0}{\log d} \sqrt{\frac{d}{\ell}} \right] + \Pr\left[\Pi_1(t) > 0 \wedge \hat{\Pi}_1(t) < -\frac{c_0}{\log d} \sqrt{\frac{d}{\ell}} \right] \\
\le &\Pr\left[\left| \Pi_1(t) - \hat{\Pi}_1(t) \right| > \frac{c_0}{\log d} \sqrt{\frac{d}{\ell}} \right] \\
\le &\Pr\left[\left(|\Pi_1(t) - \hat{\Pi}_1(t)| > \log d \sqrt{\frac{\Delta^2(q_t)}{\ell}} \right) \vee \left(\log d \cdot \sqrt{\frac{\Delta^2(q_t)}{\ell}} \ge \sqrt{\frac{d}{\ell}} \frac{c_0}{\log d} \right) \right] \\
\le &\frac{1}{n^{10}} + \Pr\left[\log d \cdot \sqrt{\frac{\Delta^2(q_t)}{\ell}} \ge \sqrt{\frac{d}{\ell}} \frac{c_0}{\log d} \right] \qquad \text{(by Lemma C.3)} \\
= &\Pr\left[(\log^4 d) \Delta^2(q_t) \ge d \right] + n^{-10}.
\end{aligned}$$

Note that $\mathbb{E}[\Delta^2(q_t)] = \gamma d$. Using a Markov inequality, we have

$$\Pr\left[\log^4 d \Delta^2(q_t) > d \right] \le \gamma \log^4 d. \tag{26}$$

Using the fact that $I(\hat{b}_t \neq s_t)$ are independent across t, with high probability we have

$$\left| \sum_{t \in \hat{\mathcal{B}}} \hat{b}_t \Pi_1(t) - \sum_{t \in \hat{\mathcal{B}}} |\Pi_1(t)| \right| \le 2n \log^5 d \sqrt{\frac{d}{\ell}} \gamma$$

□

Part 2.2. When K is substituted by $\hat{K}$. When K is substituted by $\hat{K}$, our estimator becomes

$$\begin{aligned}\hat{\mu}_1 &= \frac{\left(\sum_{t\in\hat{\mathcal{B}}}\hat{b}_t\Pi_1(t)\right)+\sum_{t\in\hat{\mathcal{B}}}\left(\hat{b}_t\Pi_2(t)+\hat{b}_t\xi_{t,q_t}\right)}{\sum_{t\in\hat{\mathcal{B}}}\hat{b}_t\hat{\Pi}_1(t).}\\ &= \frac{\sum_{t\in\hat{\mathcal{B}}}\hat{b}_t\Pi_1(t)}{\sum_{t\in\hat{\mathcal{B}}}\hat{\Pi}_1(t)}\mu_1+\frac{\sum_{t\in\hat{\mathcal{B}}}\left(\hat{b}_t\Pi_2(t)+\hat{b}_t\xi_{t,q_t}\right)}{\sum_{t\in\hat{\mathcal{B}}}\hat{b}_t\hat{\Pi}_1(t)}.\end{aligned}$$

We note that

$$\left|\sum_{t\in\hat{\mathcal{B}}}\hat{b}_t\hat{\Pi}_1(t)-\sum_{t\in\hat{\mathcal{B}}}\hat{b}_t\Pi_1(t)\right|\le \sum_{t\in\hat{\mathcal{B}}}\left|\hat{\Pi}_1(t)-\Pi_1(t)\right|\le\sum_{t\le n}\left|\hat{\Pi}_1(t)-\Pi_1(t)\right|\le n\sqrt{\frac{\gamma d}{\ell}}\log d \qquad \text{(Lemma C.3).}$$

Also, we can see that (Lemma C.6)

$$\left|\sum_{t\in\hat{\mathcal{B}}}\hat{b}_t\Pi_1(t)-\sum_{t\in\hat{\mathcal{B}}}|\Pi_1(t)|\right|\le c_0 n\log^5 d\sqrt{\frac{d}{\ell}}\gamma.$$

Both of the inequalities above imply that with high probability, the following holds true

$$\left|\sum_{t\in\hat{\mathcal{B}}}\hat{b}_t\hat{\Pi}_1(t)-\sum_{t\in\hat{\mathcal{B}}}|\Pi_1(t)|\right|=O\left(n\log^5 d\sqrt{\frac{d}{\ell}}\sqrt{\gamma}\right).$$

Next, by Lemma C.4 and Lemma C.1, we have

$$\sum_{t\in\hat{\mathcal{B}}}|\Pi_1(t)|=\Omega\left(\frac{n}{\log n}\sqrt{\frac{\ell}{d}}\right).$$

Therefore,

$$\begin{aligned}\sum_{t\in\hat{\mathcal{B}}}\hat{b}_t\Pi_1(t)&=(1+\tau_1)\sum_{t\in\hat{\mathcal{B}}}|\Pi_1(t)|\\ \sum_{t\in\hat{\mathcal{B}}}\hat{b}_t\hat{\Pi}_1(t)&=(1+\tau_2)\sum_{t\in\hat{\mathcal{B}}}|\Pi_1(t)|,\end{aligned}$$

where $|\tau_1|,|\tau_2|=O\left(\log^6 n\sqrt{\gamma}\right)$. This implies that

$$\left|\frac{\sum_{t\in\hat{\mathcal{B}}}\hat{b}_t\Pi_1(t)}{\sum_{t\in\hat{\mathcal{B}}}\hat{b}_t\hat{\Pi}_1(t)}-1\right|=O(\tau_1).$$

Now we analyze the second term. By Lemma C.4 and Lemma C.1, we have $\sum_{t\in\hat{\mathcal{B}}}\hat{b}_t\hat{\Pi}_1(t)=\Omega\left(\frac{n}{\log d}\sqrt{\frac{d}{\ell}}\right)$. By Lemma C.5, we have $\sum_{t\in\hat{\mathcal{B}}}\left(\hat{b}_t\Pi_2(t)+\hat{b}_t\xi_{t,q_t}\right)=O(\sqrt{nd})$, which implies

$$\frac{\sum_{t\in\hat{\mathcal{B}}}\left(\hat{b}_t\Pi_2(t)+\hat{b}_t\xi_{t,q_t}\right)}{\sum_{t\in\hat{\mathcal{B}}}\hat{b}_t\hat{\Pi}_1(t)}=O\left(\log d\sqrt{\frac{\ell}{n}}\right).$$

Therefore,

$$\hat{\mu}_1=(1+O(\tau))\mu_1+O\left(\log d\sqrt{\frac{\ell}{n}}\right),$$

where $\tau=\log^6 n\sqrt{\gamma}$.

Part 2.3. Analysis when observations are from $g(\cdot)$. We assume that the process is generated by $g(\cdot)$ instead of $\tilde{g}(\cdot)$. We aim to understand how the estimator changes. To distinguish the observations produced from two "worlds", we let

$$\begin{aligned}\mathbf{y}^{(1)}_{t,i}&=\sum_{j\le d}K_{i,j}\tilde{g}(\mathbf{x}_{t,j})+\xi_{t,i},\\ \mathbf{y}^{(2)}_{t,i}&=\sum_{j\le d}K_{i,j}g(\mathbf{x}_{t,j})+\xi_{t,i}.\end{aligned}$$

Let their corresponding estimators be $\mu_1^{(1)}$ and $\mu_1^{(2)}$. We next bound the difference between these two estimators. Our crucial

observation is that each $\tilde{g}(\mathbf{x}_{t,i}) - g(\mathbf{x}_{t,i})$ are bounded zero mean independent random variables. Seeing that $|\hat{\mu}_1^{(1)} - \hat{\mu}_1^{(2)}| = O(\sqrt{nd})$. Therefore, we still have

$$\hat{\mu}_1^{(2)} = \left(1 + O(\sqrt{\gamma}\log^6 n)\right)\mu_1 + O\left(\log d\sqrt{\frac{\ell}{n}}\right).$$

This proves the second part of the theorem.

Remark. We use only 1 observation for each day because our analysis relies on different $\Pi_1^{(i)}(t)$ and $\Pi_2^{(i)}(t)$ are being independent. The FlipSign algorithm does not need more samples because K is near low-rank (Theorem B.5).

Estimating $g(\cdot)$ with boosting

As shown in Alg. 3, LIN-PVEL's weak learner first performs a variable selection (i.e., selects the 3 features that correlate the most with the residual returns), and then fits a linear model with both linear and quadratic interaction terms over the selected variables. The final model is a **linear** one with features and their interactions terms.

Algorithm 3 LIN-PVEL

Input $\mathbf{X}$, $\mathbf{Y}$, $\hat{K}$, η, b
Output $\{g_m(\cdot)\}_{m\le b}$

procedure BOOSTING-ALGORITHM($\mathbf{Y}$, $\mathbf{X}$, $\hat{K}$, η, b)
  $\mathbf{Y}_{\text{Res}} \leftarrow \mathbf{Y}$ ▷ η is the learning rate
  for all $m \leftarrow 1$ **to** b **do**
    $g_m \leftarrow$ LINEAR-FIT($\mathbf{Y}_{\text{Res}}, \mathbf{X}, \hat{K}$)
    $(\hat{\mathbf{Y}})_m \leftarrow \hat{K} g_m(\mathbf{X})$.
    $\mathbf{Y}_{\text{Res}} \leftarrow \mathbf{Y}_{\text{Res}} - \eta(\hat{\mathbf{Y}})_m$
  return $\{g_m(\cdot)\}_{m\le b}$
procedure LINEAR-FIT($\mathbf{Y}$, $\mathbf{X}$, $\hat{K}$) ▷ $\mathbf{x}_{t,i} \in \mathbf{R}^k$ and $\mathbf{F}^{(t)} \in \mathbf{R}^{k\times d}$
  for all $i \leftarrow 1$ **to** d **do**
    $\mathbf{F}^{(t)}_{:,i} = \sum_{j\in[d]} \hat{K}_{i,j}\mathbf{x}_{t,j}$
  for all $j \leftarrow 1$ **to** k **do**
    $r_j = \sum_{t\le n} \text{corr}(\mathbf{F}^{(t)}_{j,:}, \mathbf{y}_t)$.
  Let j_1, j_2, j_3 be the indices with the largest r_j.
  $g(\cdot) = \arg\min_{\beta_1,\ldots,\beta_6} \sum_{\substack{t\le n\\ i\in[d]}} (\mathbf{y}_{t,i} - \sum_{j\in[d]}$
    $\hat{K}_{i,j}\left(\beta_1(\mathbf{x}_{t,j})_{j_1} + \cdots + \beta_6(\mathbf{x}_{t,j})_{j_2}\cdot(\mathbf{x}_{t,j})_{j_3}\right))^2$. ▷ Fits a linear model with linear and quadratic interaction terms.
  return $g(\cdot)$

Consolidation/Ensemble model

We next describe how we consolidate forecasts generated by multiple models. We do not intend to design a new consolidation algorithm. Instead, we use a "folklore" algorithm that weighs each model forecast by its recent historical performance. Specifically, we let $C = \{\hat{\mathbf{y}}_1, \ldots, \hat{\mathbf{y}}_m\}$ be the set of models to be consolidated. Our consolidated forecast is a linear combination of all forecasts $\hat{\mathbf{y}} = \sum_{j=1}^m w_j\hat{\mathbf{y}}_j$, where w_j is simply the t-statistics of the j-th model computed through the Newey-West estimation algorithm from the in-sample data. For example, when $m = 2$ and the t-statistics for $\hat{\mathbf{y}}_1$ and $\hat{\mathbf{y}}_2$ are 3 and 5 respectively, we set the consolidated forecast be proportional to $3\times\hat{\mathbf{y}}_1 + 5\times\hat{\mathbf{y}}_2$. The consolidated forecast needs to be properly re-scaled (e.g., set the daily standard deviation to be constant). In the forecasting models, correlation with the ground-truth is more important than MSE. Therefore, the scale of a forecast is less important than its direction. This t-statistics based consolidation algorithm is used in the following two situations.

Allowing nparam-gEST to use all factors. Our theoretically sound algorithm in Sec. 4 allows us to use only a small number of features. Now we may use a two-step procedure to let this algorithm simultaneously use hundreds of technical factors constructed in-house. *Step 1.* For each improvable factor, we build a model that uses only this factor as the feature. *Step 2.* After we obtain multiple models (the number of models is the same as the number of improvable factors), we use the above consolidation algorithm to produce the final forecast.

Consolidating multiple models. We also use the consolidation trick to aggregate the forecasts of all our models (LIN-PVEL, nparam-gEST, MLP). The result in Table 1 (last line) shows that the consolidated signal is stronger than any individual signal. Even if a model may not have the best out-of-sample performance, it may still be useful for constructing consolidated signals.

Additional proofs and calculations

In this section we give some additional proofs and calculations for App. B and App. C.

Proof of Proposition B.7

We can see that

$$\frac{1}{n}\mathbf{Y}^{\mathrm{T}}\mathbf{Y} = \frac{1}{n}\left(K^{\mathrm{T}}\mathbf{S}^{\mathrm{T}}\mathbf{S}K + K^{\mathrm{T}}\mathbf{S}^{\mathrm{T}}E + E^{\mathrm{T}}\mathbf{S}K + E^{\mathrm{T}}E\right) \tag{27}$$

$$= K^{\mathrm{T}}K + \mathcal{E}_1 + \mathcal{E}_2 + \mathcal{E}_3 + \mathcal{E}_4, \tag{28}$$

where $\mathcal{E}_1 = K^{\mathrm{T}}\left(\frac{\mathbf{S}^{\mathrm{T}}\mathbf{S}}{n} - I\right)K$, $\mathcal{E}_2 = \frac{K^{\mathrm{T}}\mathbf{S}^{\mathrm{T}}E}{n}$, $\mathcal{E}_3 = \frac{E^{\mathrm{T}}\mathbf{S}K}{n}$, and $\mathcal{E}_4 = \frac{E^{\mathrm{T}}E}{n}$

We next show that each $\mathcal{E}_i$ ($i \leq 4$) is small.

Bounding $\mathcal{E}_1$. We need the following lemma.

Lemma F.1. *Let $\mathbf{S} \in \mathbf{R}^{n\times d}$ be such that each row $\mathbf{S}_{i,:}$ is an i.i.d. random vector $\|S_{i,:}\|_\infty \leq 1$ and $\mathbb{E}[\mathbf{S}_{i,:}^{\mathrm{T}}\mathbf{S}_{i,:}] = I$. We have*

$$\Pr\left[\left\|\frac{\mathbf{S}^{\mathrm{T}}\mathbf{S}}{n} - I_{d\times d}\right\|_2 \geq \epsilon\right] \leq 2n^2\exp\left(-\frac{n\epsilon^2}{\log^4 n}\right), \tag{29}$$

where ϵ is a tunable parameter.

Here, we shall set $\epsilon = \frac{\log^3 n}{\sqrt{n}}$. This implies that with high probability $\left\|\frac{\mathbf{S}^{\mathrm{T}}\mathbf{S}}{n} - I_{d\times d}\right\|_2 = O\left(\frac{\log^3 n}{\sqrt{n}}\right)$. On the other hand, we can see that $\|K\|_F^2 = \Theta(d^2)$ and $\|K\|_F = \Theta(d)$. This implies

$$\|\mathcal{E}_1\|_F = \left\|K^{\mathrm{T}}\left(\frac{\mathbf{S}^{\mathrm{T}}\mathbf{S}}{n} - I_{d\times d}\right)K\right\|_F \leq \left\|\frac{\mathbf{S}^{\mathrm{T}}\mathbf{S}}{n} - I_{d\times d}\right\|_2 \|K\|_F^2 = \Theta\left(\frac{d^2\log n}{\sqrt{n}}\right) \tag{30}$$

Bounding $\mathcal{E}_2$ and $\mathcal{E}_3$. Recall that $\mathcal{E}_2 = \frac{K^{\mathrm{T}}\mathbf{S}^{\mathrm{T}}E}{n}$ and $\mathcal{E}_3 = \frac{E^{\mathrm{T}}\mathbf{S}K}{n}$. We have the following lemma.

Lemma F.2. *Let $\mathbf{S} \in \mathbf{R}^{n\times d}$ be such that each row $\mathbf{S}_{i,:}$ is an i.i.d. random vector with $\|\mathbf{S}_{i,:}\|_\infty \leq 1$ and $\mathbb{E}[\mathbf{S}_{i,:}^{\mathrm{T}}\mathbf{S}_{i,:}] = I$. Let $E \in \mathbf{R}^{n\times d}$ be such that $E_{i,j}$ are i.i.d. Gaussian with standard deviation σ_ξ. We have with overwhelming probability*

$$\|\mathbf{S}^{\mathrm{T}}E\|_F^2 \leq c_0\sigma_\xi d^2 n \tag{31}$$

for some constant c_0.

Proof of Lemma F.2. First, note that

$$\mathbb{E}[\|\mathbf{S}^{\mathrm{T}}E_{:,i}\|_F^2 \mid \mathbf{S}] = \sigma_\xi^2\|S\|_F^2.$$

Therefore, we have $\mathbb{E}[\|\mathbf{S}^{\mathrm{T}}E_{:,i}\|_F^2] = \sigma_\xi^2 dn$. This also implies that

$$\mathbb{E}[\|\mathbf{S}^{\mathrm{T}}E\|_F^2] = \sum_{i\leq d}\mathbb{E}[\|\mathbf{S}^{\mathrm{T}}E_{:,i}\|^2] = \sigma_\xi^2 d^2 n.$$

By a standard Chernoff bound, we have whp

$$\|\mathbf{S}^{\mathrm{T}}E\|_F^2 \leq c_0\sigma_\xi^2 d^2 n \tag{32}$$

for some constant c_0, i.e., whp $\|\mathbf{S}^{\mathrm{T}}E\|_F = O(\sigma_\xi d\sqrt{n})$. □

We next use Lemma F.2 to bound $\mathcal{E}_2$ and $\mathcal{E}_3$:

$$\|\mathcal{E}_2\|_F = \|\mathcal{E}_3\|_F = \frac{1}{n}\|K^{\mathrm{T}}\mathbf{S}^{\mathrm{T}}E\|_F = \frac{1}{n}\|K\|_2\|\mathbf{S}E\|_F. \tag{33}$$

Now we have $\|K\|_2 = O(d)$ and $\|\mathbf{S}E\|_F = O(d\sqrt{n})$ whp. Therefore, with high probability

$$\|\mathcal{E}_2\|_F = \|\mathcal{E}_3\|_F = O\left(\frac{d^2}{\sqrt{n}}\right).$$

Bounding $\mathcal{E}_4$. With the assumption that $d = O(n)$, we have

$$\left\|\frac{E^{\mathrm{T}}E}{n}\right\|_F^2 \leq \frac{\mathrm{Rank}(E^{\mathrm{T}}E)\|E\|_2^2}{n} = O\left(\frac{\sigma_\xi^2 dn}{n}\right) = O(\sigma_\xi^2 d). \tag{34}$$

Above, we used a finite sample version of semi-circle law (i.e., $\|E\|_2^2 = O(n)$ whp (Rudelson and Vershynin 2010)).

Summing up above and using that $n < d^2$ and $\sigma_\xi = O(\sqrt{d})$, we have $\left\|\frac{1}{n}\mathbf{Y}^{\mathrm{T}}\mathbf{Y} - K^{\mathrm{T}}K\right\|_F = O\left(\frac{d^2\log^3 n}{\sqrt{n}}\right)$.

Anti-concentrations

Theorem F.3. *(Littlewood-Offord-Erdos; e.g., (Krishnapur 2016)) Let $L_1, \ldots, L_d \geq 1$. Let $\xi_1, \ldots \xi_n$ be independent Bernoulli ± 1 unbiased random variables such that $\Pr[\xi_i = 1] = \frac{1}{2}$. Let $S = \sum_{i\leq n}\xi_i L_i$. For any open interval I of length 2, we have*

$$\Pr[S \in I] = O(n^{-\frac{1}{2}}). \tag{35}$$

Lemma F.4. *Let $\ell \le d/\log^2 d$. Let $L_1, L_2, \ldots, L_d$ be positive numbers such that $L_i = \Omega(1)$. Define a random variable*

$$Z_i = \begin{cases} L_i & \text{with probability } \frac{1}{\ell} \\ -\frac{L_i}{\ell-1} & \text{with probability } 1 - \frac{1}{\ell}. \end{cases}$$

There exist constants c_0 and c_1 such that

$$\Pr\left[\sum_{i \le d} Z_i \ge \frac{c_0}{\log d}\sqrt{\frac{d}{\ell}}\right] \ge c_1$$

Proof. We shall use Theorem F.3 to prove Lemma F.4. Theorem F.3 requires that random variables ξ_i (or Z_i in our setting) to be symmetric, which is violated in our setting. Our goal is to reduce our problem to the original setting.

We now show that this can be done through "debiasing" the walk. We first define $\{B_i\}_{i\in[d]}$ such that B_i is a random binary indicator variable with $\Pr[B_i = 1] = \frac{\ell-2}{\ell}$ and $\Pr[B_i = 0] = \frac{2}{\ell}$.

We may generate Z_i by using B_i, i.e., when $B_i = 1$, we set $Z_i = -\frac{L_i}{\ell-1}$, and when $B_i = 0$, we set $Z_i = L_i$ with half of the probability and $Z_i = -\frac{L_i}{\ell-1}$ with the other half of the probability. Note that when $B_i = 0$, the probability that Z_i takes one of the possible values in $\frac{1}{2}$ (thus is uniform).

Next, let $\mathcal{B} = \{B_i : B_i = 1\}$ and $\bar{\mathcal{B}} = \{B_i : B_i = 0\}$. Let also that $T = |\bar{\mathcal{B}}|$. One can see that $\mathbb{E}[T] = \frac{2d}{\ell}$. In addition, because $d = \omega(\ell \log \ell)$, with overwhelming probability that $T \ge \frac{d}{\ell}$.

We now can see that

$$\mathbb{E}\left[\sum_{i\in\mathcal{B}} Z_i\right] = -\left(\sum_{i\le d} L_i\right)\frac{\ell-2}{\ell(\ell-1)}$$

In addition, $\mathbb{E}[Z_i \mid i \in \bar{\mathcal{B}}] = L_i\left(1 - \frac{1}{\ell-1}\right)\frac{1}{2}$ for any $i \in \bar{\mathcal{B}}$. Next, we define a random variable to "debias" Z_i, conditioned on $i \notin \mathcal{B}$, i.e., for any $i \in \bar{\mathcal{B}}$

$$\tilde{Z}_i = \begin{cases} L_i - L_i\left(1 - \frac{1}{\ell-1}\right)\frac{1}{2} & \text{with probability } \frac{1}{2} \\ -\frac{L_i}{\ell-1} - L_i\left(1 - \frac{1}{\ell-1}\right)\frac{1}{2} & \text{with probability } \frac{1}{2}. \end{cases} \tag{36}$$

Note that $\mathbb{E}[\tilde{Z}_i] = 0$ and $L_i - L_i\left(1 - \frac{1}{\ell-1}\right)\frac{1}{2} = \frac{L_i}{\ell-1} + L_i\left(1 - \frac{1}{\ell-1}\right)\frac{1}{2}$. Next, we have

$$\begin{aligned}\sum_{i\le d} Z_i &= \sum_{i\in\mathcal{B}}\left[-\frac{L_i}{\ell-1}\right] + \sum_{i\in\bar{\mathcal{B}}}\left(\tilde{Z}_i + L_i\left(1 - \frac{1}{\ell-1}\right)\frac{1}{2}\right) \\ &= \underbrace{\left(\sum_{i\in\mathcal{B}}\left(-\frac{L_i}{(\ell-1)}\right) + \sum_{i\notin\mathcal{B}} L_i\left(1 - \frac{1}{\ell-1}\right)\frac{1}{2}\right)}_{\Psi_1} + \underbrace{\left(\sum_{i\notin\mathcal{B}}\tilde{Z}_i\right)}_{\Psi_2}.\end{aligned}$$

One can see that *(i)* the sign of Ψ_2 is independent of the sign of Ψ_1, and *(ii)* one of $\Pr[\Psi_1 \ge 0] \ge \frac{1}{2}$ and $\Pr[\Psi_1 \le 0] \ge \frac{1}{2}$ must hold. Wlog, assume that $\Pr[\Psi_1 \ge 0] \ge \frac{1}{2}$. By Theorem F.3, we have $\Pr\left[\Psi_2 \ge \frac{c_1}{\log d}\sqrt{T}\right] = \Omega(1)$.

Finally, we have

$$\begin{aligned}\Pr\left[\Psi_1 + \Psi_2 \ge \frac{c_1}{\log d}\sqrt{\frac{d}{\ell}}\right] &\ge \Pr[\Psi_1 \ge 0]\Pr\left[\Psi_2 \ge \frac{c_2}{\log d}\sqrt{T} \mid \Psi_1 \ge 0\right] \\ &\ge \Pr[\Psi_{\ge} 0]\Pr\left[\Psi_2 \ge \frac{c_1}{\log d}\sqrt{\frac{d}{\ell}} \mid \Psi_1 \ge 0\right] \\ &= \Omega(1).\end{aligned}$$

The second inequality uses $T \ge \frac{d}{\ell}$ whp. □

Experiments

We evaluate our algorithms on an emerging market dataset and a social network dataset. We describe the dataset collection and setup of experiments, the evaluation metrics, additional explanation of baselines, and analysis for our performance for both datasets.

Equity returns

We use daily prices and volumes to generate the features and focus on predicting the next 5-day returns.

Datasets collection. The specific description of the used dataset is as follows:

(1) Chinese stock data: Our data set consists of daily prices and trading volumes of approximately 3,600 stocks between 2009 and 2018. We use open prices to compute the returns and we aim to predict the next 5-day returns, in which the last three years are out-of-sample. We examine two universes. (i) *Universe 800* is equivalent to the S&P 500 and consists of 800 stocks, and (ii) *Full universe* consists of all stocks except for illiquid ones. The average "size" (in either capital or trading volume) in *Universe 800* is larger than the average "size" of the *Full universe*.

(2) Technical factors: We manually build 337 technical factors based on previous studies (Gu, Kelly, and Xiu 2020; Colby and Meyers 1988; Kakushadze 2016; Amihud 2002; Posner 2014). All these factors are derived from price and dollar volume.

(3) Barra factor dataset: We use a third-party risk model known as the Barra factor model (Orr and Mashtaler 2012). The model uses 10 real-valued factors and 1 categorical variable to characterize a stock. The real-valued factors known as "style factors" include beta, momentum, size, earnings yield, residual volatility, growth, book-to-price, leverage, liquidity, and non-linear size. The categorical variable represents the industrial sector the stock is in. We do not use the categorical variable in our experiments. Table 3 defines the style factors.

Barra factors name	**Beta**	**Momentum**	**Size**	**Earnings Yield**	**Residual Volatility**
Description	Measure of volatility.	Rate of acceleration of a security's price or volume.	Total equity value in market.	The percentage of how much a company earned per share.	The volatility of daily excess returns.
Barra factors name	**Growth**	**Book-to-Price**	**Leverage**	**Liquidity**	**Non-linear Size**
Description	Measure of the growth rate.	firm's book value to its market capitalization.	Measure of a firm's leverage rate.	Measure of a firm's liquidity.	Non-linear transformation of size factor.

Table 3: Barra style factors from (Orr and Mashtaler 2012).

(4) News dataset: We crawled financial news between 2012 and 2018 from a major Chinese news website Sina. We collected a total number of 2.6 million news articles. Each article can refer to one or multiple stocks. On average, a piece of news refers to 2.94 stocks. We remark that our way to use news data sets deviates from standard news-based models for predicting equity returns (Ding et al. 2015; Hu, Liu et al. 2018). Most news-based models aim to extract sentiments and events that could directly impact one or more related stocks' prices. Rather than building links between events and the stock fluctuation, we use news dataset to identify similarities between stocks. i.e., when two stocks are mentioned often, they are more likely to be similar. This is orthogonal to how the news itself impacts the movement of stock prices.

Model and training. We use three years of data for training, 10 months of data for validation and one year of data for testing. We re-train the model every testing year. For example, the training set starts from Jan. 1, 2012, to Dec. 31, 2014. The corresponding validation period is from Jan. 15, 2015, to Dec. 16, 2015. We use the validation set to select the hyperparameters and build the model. Then we use the trained model to forecast returns of equity in the same universe from Jan. 1, 2016, to Dec. 31, 2016, where we set 10 trading days as the "gap". Then we re-train the model by using data in the second training period (Jan. 1, 2013, to Dec. 17, 2015). We set a "gap" between the training and validation periods, and the validation periods testing dataset to avoid looking-ahead issues.

Additional explanation about evaluation matrices and baselines

Computing t-statistics. Recall that $\mathbf{y}_t \in \mathbf{R}^d$ is a vector of responses and $\hat{\mathbf{y}}_t \in \mathbf{R}^d$ is the forecast of a model to be evaluated. We examine whether the signals are correlated with the responses, i.e., for each t we run the regression model $\mathbf{y}_t = \beta_t \hat{\mathbf{y}}_t + \epsilon$ and test whether we can reject the null hypothesis that the series $\beta_t = 0$ for all t. Note that the noises in the regression model are serially correlated so we use Newey-West (Newey and West 1986) estimator to adjust serial correlation issues. Consider, for example, a coin-tossing game, in which we make one dollar if our prediction of a coin toss is correct or lose one dollar otherwise. When our forecast has 51% accuracy, we are guaranteed to generate positive returns in the long run by standard concentration results. Testing whether our forecast has better than 51% accuracy needs many trials because, e.g., when there are only 100 tosses, there is a $\approx 40\%$ probability that a random forecast has a $\geq 51\%$ accuracy rate.

An example of compare correlation vs MSE. Consider a case where the true returns of Google and Facebook are +2% and +4%, respectively. Let forecast A be -1% (Google) and -1% (Facebook), and let forecast B be +20% (Google) and +40% (Facebook). While forecast A has a smaller MSE, forecast B is more accurate and more profitable (e.g., the directions of the returns are predicted correctly).

Sharpe Ratio. The popular Sharpe Ratio measures the performance of an investment by adjusting for its risk.

$$\text{Sharpe Ratio} = \frac{R_p - R_f}{\sigma_p}, \tag{37}$$

where R_p is the return of the portfolio, R_f is the risk-free rate, and σ_p is the standard deviation of the portfolio's excess return.

PnL. Profit & Loss (PnL) is a standard performance measure used in trading and captures the total profit or loss of a portfolio over a specified period. The PnL of all forecasts made on day t is given by

$$\text{PnL} = \frac{1}{d}\sum_{i}^{d} sign(\hat{\mathbf{y}}_{t,i}) * \mathbf{y}_{t,i}, \quad t = 1, \ldots, n, \tag{38}$$

Additional explanation for recent CAMs

• **SFM (Zhang, Aggarwal, and Qi 2017).** SFM decomposes the hidden states of an LSTM (Rather, Agarwal, and Sastry 2015) network into multiple frequencies by using Discrete Fourier Transform (DFT) so the model can capture signals at different horizons.

• **HAN (Hu, Liu et al. 2018).** This work introduces a so-called hybrid attention technique that translates news into signals.

• **AlphaStock (Wang et al. 2019).** This work is proposed by (Wang et al. 2019). AlphaStock integrates deep attention networks reinforcement learning with the optimization of the Sharpe Ratio. For each stock, AlphaStock uses LSTM (Sak, Senior, and Beaufays 2014) with attention on hidden states to extract the stock representation. Then AlphaStock uses CAAN, which is a self-attention layer, to capture the interrelations among stocks. Specifically, CAAN takes the stock representations as inputs to generate the stock's winning score. We implement LSTM with basic CAAN and change the forecast into return instead of winning scores.

• **ARRR** ARRR (Wu et al. 2020a) is a new regularization technique designed to address the overfitting issue in vector regression under the high-dimensional setting. Specifically, ARRR involves two SVD, the first SVD is for estimating the precision matrix of the features, and the second SVD is for solving the matrix denoising problem.

Experiment evaluation

Detailed results for each testing year Tables 4 and 5 list the results for each testing year in *Universe 800* and *Full universe*. The bold fonts denote the best performance in each group. The results are consistent with the Table 1. Note that we also report weighted correlation and weighted t-statistic. The weights are determined by the historical dollar volume of the asset. These statistics are useful because the positions taken by the optimizer are sensitive to historical dollar volumes.

		2016				2017				2018			
	Our CAMs	core	w_corr	t-stat	w_t-stat	corr	w_corr	t-stat	w_t-stat	corr	w_corr	t-stat	w_t-stat
LIN-PVEL	Opt.	0.1084	**0.1149**	9.9081	**7.9814**	**0.0388**	**0.0624**	**3.0441**	**3.8233**	**0.0820**	**0.1037**	**7.4293**	**7.2038**
	DD	0.1064	0.1109	9.7619	7.4212	0.0284	0.0541	2.1949	3.1475	0.0729	0.0972	6.9855	6.8733
nparam-gEST	Opt.	0.0800	**0.0595**	**6.1201**	**2.9453**	-0.0067	**-0.0095**	-0.4554	**-0.4800**	0.0604	0.0461	**4.2237**	2.2608
	DD	**0.0805**	0.0577	6.0357	2.7922	**-0.0051**	-0.0102	**-0.3472**	-0.5496	0.0582	**0.0446**	4.1457	**2.3772**
MLP	Opt.	**0.0958**	0.0917	**7.4846**	5.0039	**0.0050**	0.0182	0.3610	**0.9769**	**0.0641**	0.0602	5.7370	3.4793
	DD	0.0940	**0.0919**	7.3239	**5.0924**	0.0047	0.0165	0.3308	0.8749	0.0634	**0.0604**	6.0943	3.5154
LSTM	Opt.	**0.0662**	**0.0762**	5.7290	**4.5106**	**-0.0216**	-0.0144	-1.5466	-0.7624	**0.0413**	**0.0423**	**3.7099**	**2.4316**
	DD	0.0606	0.0682	4.8167	4.0641	-0.0025	**0.0017**	**-0.1520**	**0.0803**	0.0110	0.0356	0.9228	1.8596
Linear	Opt.	**0.0726**	**0.0708**	**5.1011**	**3.5324**	**0.0054**	0.0166	0.3429	0.8720	**0.0567**	**0.0679**	**4.1086**	**4.1605**
UM: poor man LIN-PVEL		**0.1093**	0.1128	**10.1352**	7.5786	0.0242	0.0499	1.7650	2.9580	0.0688	0.0970	6.3840	6.6570
UM: poor man nparam-gEST		0.0788	0.0579	6.0820	2.8561	-0.0061	-0.0096	-0.4083	-0.4862	0.0569	0.0442	3.7779	2.1036
UM: MLP		0.0861	0.0812	5.8771	4.2409	0.0052	0.0132	**0.3800**	0.6764	0.0609	0.0571	**6.1635**	**3.7053**
UM: LSTM		0.0619	0.0632	**6.5873**	4.1299	-0.0253	-0.0215	-1.7873	-1.1504	0.0169	0.0183	1.4487	1.0374
UM: Lasso		-0.0046	0.0088	-0.3889	0.5531	0.0282	**0.0333**	**2.1633**	**2.0936**	0.0083	0.0153	1.2997	1.6726
UM: Ridge		0.0290	0.0406	3.4617	2.8301	-0.0064	-0.0161	-1.3527	-1.3455	0.0091	0.0066	1.5421	0.5618
UM: GBRT		0.0655	0.0601	9.9051	5.4083	0.0419	0.0565	5.6987	5.0517	0.0476	0.0606	7.1179	6.4332
UM: SFM		0.0114	0.0102	0.9237	0.6828	0.0097	0.0081	0.6644	0.4479	0.0078	-0.0133	0.6194	-0.8263
Existing CAM: Alpha		0.0132	0.0165	2.3632	1.8841	0.0135	0.0133	2.5594	1.6109	-0.0062	-0.0110	-1.3995	-1.3236
Existing CAM: HAN		0.0096	0.0056	1.0205	0.4777	0.0060	0.0088	0.4980	0.5455	0.0160	0.0101	1.9352	0.7273
Existing CAM: VR		0.0207	0.0038	1.8590	0.2582	0.0087	0.0192	0.9239	1.6069	0.0174	0.0248	1.6513	1.3219
Existing CAM: ARRR		0.0593	0.0657	3.8366	3.2866	-0.0083	-0.0043	0.3975	0.578	0.0432	0.0533	3.3669	3.9343

Table 4: The by year results for *Universe 800*

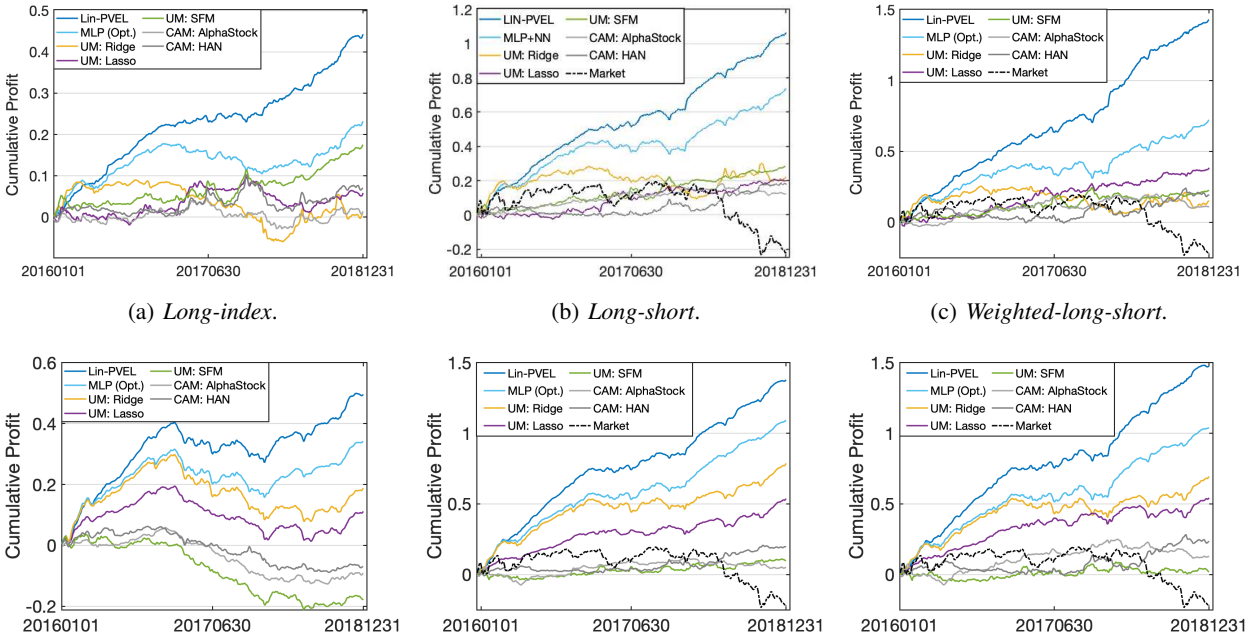
Simulation and PnL. Fig. 4 shows three ways to simulate investments on our signals for testing years from 2016 to 2018 for *Universe 800* and *Full universe*. (i) Long-index portfolio: Long-only minus the market index. (ii) Long-short portfolio[1]: By allowing short-selling, we can execute on negative forecasts to understand the overall forecasting quality. (iii) Weighted-Long-short portfolio: We weight an investment by the historical turnover of the asset. We conduct the trading in the daily granularity and select the stocks from the top 20% strongest forecast signals. We can see that our signals are consistently better than other baselines in both long/long-short. The results confirm that our method generates stronger and more robust signals for trading.

Visualization/Qualitative examination

[1]Short is implementable in the Chinese market only under special circumstances, e.g., through brokers in Hong Kong under special arrangements.

		2016				2017				2018			
Method	Our CAMs	corr	w_corr	t-stat	w_t-stat	corr	w_corr	t-stat	w_t-stat	corr	w_corr	t-stat	w_t-stat
LIN-PVEL	Opt.	0.1328	**0.1316**	12.1131	8.9092	0.0564	0.0590	4.4505	3.3487	**0.0939**	**0.1122**	8.2186	7.0727
	DD	**0.1358**	0.1308	**12.7510**	**9.0186**	**0.0584**	**0.0632**	**4.8204**	**3.5678**	0.0859	0.1062	**9.5940**	**8.1365**
nparam-gEST	Opt.	**0.1045**	**0.0969**	**10.3212**	**6.6829**	0.0159	**0.0129**	1.2465	**0.7205**	**0.0650**	**0.0559**	**5.6303**	**3.1603**
	DD	0.1039	0.0941	8.9599	6.1395	**0.0174**	0.0118	**1.3356**	0.6298	0.0596	0.0463	4.8991	2.3960
MLP	Opt.	**0.1072**	**0.0983**	8.3802	6.1198	0.0290	**0.0219**	1.9765	**1.0954**	**0.0851**	**0.0876**	**11.0013**	**5.9272**
	DD	0.0935	0.0921	**8.4685**	**6.6603**	**0.0303**	0.0212	2.1287	1.0544	0.0776	0.0788	8.4386	5.0757
LSTM	Opt.	**0.0744**	**0.0750**	**7.0918**	**4.5892**	0.0210	0.0200	1.3738	0.8513	**0.0465**	**0.0523**	**5.6732**	**3.3210**
	DD	0.0476	0.0532	4.1179	3.9801	**0.0327**	**0.0292**	**2.7048**	**1.4664**	0.0441	0.0462	4.5315	2.5036
Linear	Opt.	**0.0995**	**0.0956**	7.6410	5.7275	0.0123	0.0041	0.7983	0.1940	**0.0527**	**0.0684**	**5.1804**	**4.4595**
UM: poor man LIN-PVEL		0.1279	0.1214	11.4010	8.2082	0.0488	0.0517	3.6993	2.8182	0.0713	0.0920	7.1887	5.9714
UM: poor man nparam-gEST		0.1002	0.0957	8.8982	6.2661	0.0169	0.0112	1.2822	0.5872	0.0580	0.0457	4.8490	2.4000
UM: MLP		0.0837	0.0830	6.3470	5.4216	0.0286	0.0123	**2.3251**	0.6869	0.0697	0.0449	8.1207	2.5859
UM: LSTM		0.0684	0.0577	5.7502	3.6079	-0.0007	-0.0057	-0.0401	-0.2410	0.0379	0.0370	3.4094	1.8486
UM: Lasso		0.0589	0.0612	**9.4412**	**8.2446**	-0.0032	-0.0028	-0.3080	-0.1819	0.0313	0.0169	2.6735	0.8430
UM: Ridge		0.0631	0.0636	5.9308	4.4291	**0.0152**	**0.0168**	**0.9798**	**0.8296**	0.0290	0.0413	2.7536	2.2439
UM: GBRT		0.0898	0.0842	13.6203	9.5775	0.0531	0.0687	5.8328	7.0454	0.0588	0.0711	8.5605	7.0563
UM: SFM		-0.0055	-0.0058	-0.4868	-0.4281	0.003	0.0096	0.3578	0.708	0.0107	0.0057	1.2447	0.4851
Existing CAM: Alpha		0.0076	0.0109	1.2236	1.3496	0.0093	0.0123	2.3817	1.9694	0.003	0.008	0.778	1.4379
Existing CAM: HAN		0.0135	0.0081	1.7515	0.7924	-0.0008	-0.0020	-0.0791	-0.1253	0.0114	0.0098	1.5316	0.7547
Existing CAM: VR		0.0031	0.0033	0.3887	0.2949	-0.0056	-0.0245	-0.7108	-1.868	0.0148	0.0138	2.0156	0.8331
Existing CAM: ARRR		0.0527	0.0714	2.9072	3.2284	-0.0067	-0.0169	0.6266	-0.2307	0.0205	0.0275	2.0334	1.2328

Table 5: Yearly results for *Full universe*.



(a) *Long-index.* (b) *Long-short.* (c) *Weighted-long-short.*

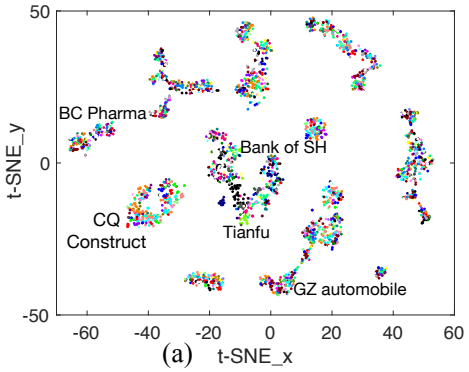
(d) *Long-index.* (e) *Long-short.* (f) *Weighted-long-short.*

Figure 4: Cumulative PnL (Profit & Loss) curves of the top quintile portfolio (i.e., on any given day, we consider a portfolios with only the top 20% strongest in magnitude predictions, against future market excess returns). (a)-(c) are for the *Universe 800* and (d)-(f) are for the *Full universe*.

(1) Visualization for learned stock latent space. We examine the latent positions we learned, and draw two observations. *(i) Latent positions are not driven by sectors.* One possible explanation of our models' forecasting power is that they capture sector-related signals, e.g., growth of one airline implies the growth of others. Our visualization in Fig. 5 shows this is not the case. *(ii). Interactions are fine-grained.* We also present the neighbors uncovered by our pipeline, and also those found by ALPHASTOCK for five stocks (all well known to the public). Our algorithm picks up different embeddings for these five stocks compared to ALPHASTOCK, which indicates we discover an orthogonal signal.

Predicting user popularity in Twitter dataset

We also evaluated our mode on the Twitter dataset and focus on predicting a user's next 5-day popularity. The popularity is defined as the sum of received quotes, retweets, and replies.



(a)

Center	Algorithm	Neighbor 1	Neighbor 2	Neighbor 3
CQ Construct 重庆建工 (construction)	Ours	Hunan Friendship	North China Pharma	Citic Helicopter
	AlphaStock	Yuntianhua	Ningbo Marine	Tianjin Teda
Tianfu Energy 天富能源 (utility)	Ours	Top Energy	Fuxing Tech	HuaYuan Property
	AlphaStock	SH Industrial	GZ Develop	Jilin Electric
GZ Automobile 广汽集团 (car)	Ours	SAIC Motor	SH Port	Huayu Automotive
	AlphaStock	Huayu Automobile	Hengyi Petrochemical	Kangde Material
BC Pharma 步长制药 (medicine)	Ours	Mantis Construct	China Meheco	ZJ Juhua
	AlphaStock	CR Pharma	National Medicines	Lingrui Pharma
Bank of SH 上海银行(bank)	Ours	Bank of Jiangsu	Bank of Beijing	Sinopec
	AlphaStock	Commercial Bank	Tianjin Realty	China Coal

(b)

Figure 5: (a): t-SNE for our latent embedding (colors are coded by sectors); (b): Examples of stocks and their neighbors.

Data collection We used Twitter streaming API and tracked the tweets with topics related to the political keywords "trump", "clinton", "kaine", "pence", and "election2016". In total, we collected 15 months the Twitter data from October 01, 2016, to December 31, 2017, from 19 million distinct users and 804 million tweets. The user u's interaction is defined as is and only if he or she is quoted/replied/retweeted by another use v. Due to the huge size, we extract the subset of 2000 users with the most interactions for evaluation.

Training and hyper-parameters We used October 01, 2016, to June 30, 2017, as the training period, July 01, 2017, to September 30, 2017 as the validation dataset to tune the hyper-parameters, and October 01, 2017, to December 31, 2017, as the testing dataset to evaluate the models' performance.

Models	MSE (in-sample)	MSE (out-of-sample)	Corr (in-sample)	Corr (out-of-sample)
Ours: Lin-PVEL	0.472	**0.520**	0.733	**0.712**
Ours: nparam-gEST	0.492	**0.559**	0.688	**0.658**
Ours: MLP	0.486	**0.547**	0.716	**0.692**
Ours: LSTM	0.484	**0.541**	0.724	**0.703**
UM: Poor man Lin-PVEL	0.488	0.552	0.710	0.684
UM: Poor man nparam-gEST	0.544	0.584	0.634	0.605
UM: Poor man MLP	0.506	0.562	0.703	0.673
UM: Poor man LSTM	0.496	0.559	0.710	0.679
UM: Linear models	0.616	0.663	0.618	0.592
UM: Random forest	0.611	0.659	0.623	0.587
UM: Xgboost	0.530	0.571	0.671	0.647
CEM: VR	0.540	0.729	0.649	0.408
CEM: ARRR	0.564	0.652	0.610	0.573
Ad-hoc: Node2Vec (Grover and Leskovec 2016)	0.537	0.690	0.693	0.468
Consolidated: All Ours	**0.459**	**0.502**	**0.767**	**0.742**

Table 6: Overall in-sample and out-of-sample performance on the Twitter data set. Boldface denotes the best performance in each group.

References

Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G. S.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Goodfellow, I.; Harp, A.; Irving, G.; Isard, M.; Jia, Y.; Jozefowicz, R.; Kaiser, L.; Kudlur, M.; Levenberg, J.; Mane, D.; Monga, R.; Moore, S.; Murray, D.; Olah, C.; Schuster, M.; Shlens, J.; Steiner, B.; Sutskever, I.; Talwar, K.; Tucker, P.; Vanhoucke, V.; Vasudevan, V.; Viegas, F.; Vinyals, O.; Warden, P.; Wattenberg, M.; Wicke, M.; Yu, Y.; and Zheng, X. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*.

Abasi, H.; Bshouty, N. H.; and Mazzawi, H. 2014. On exact learning monotone DNF from membership queries. In *ALT*.

Abraham, I.; Chechik, S.; Kempe, D.; and Slivkins, A. 2015. Low-distortion inference of latent similarities from a multiplex social network. *SICOMP*.

Amihud, Y. 2002. Illiquidity and stock returns: cross-section and time-series effects. *Journal of financial markets*.

Arora, S.; Bhaskara, A.; et al. 2014. More algorithms for provable dictionary learning. *arXiv preprint*.

Azevedo, D.; and Menegatto, V. A. 2015. Eigenvalues of dot-product kernels on the sphere. *Proceeding Series of the Brazilian Society of Computational and Applied Mathematics*.

Belkin, M. 2018. Approximation beats concentration? An approximation view on inference with smooth radial kernels. In *COLT*.

Bhojanapalli, S.; Kyrillidis, A.; and Sanghavi, S. 2016. Dropping convexity for faster semi-definite optimization. In *COLT*.

Bunea, F.; She, Y.; and Wegkamp, M. H. 2011. Optimal selection of reduced rank estimators of high-dimensional matrices. *ANN STAT*.

Candès, E. J.; and Wakin, M. B. 2008. An introduction to compressive sampling. *IEEE signal processing magazine*, 25(2): 21–30.

Chen, C.; Zhao, L.; Bian, J.; et al. 2019. Investment behaviors can tell what inside: Exploring stock intrinsic properties for stock trend prediction. In *KDD*.

Chen, K.; Dong, H.; and Chan, K.-S. 2013. Reduced rank regression via adaptive nuclear norm penalization. *Biometrika*.

Chen, L.; Pelger, M.; and Zhu, J. 2019. Deep learning in asset pricing. *Available at SSRN 3350138*.

Chen, T.; and Guestrin, C. 2016. Xgboost: A scalable tree boosting system. In *KDD*.

Chung, F.; and Lu, L. 2006. Concentration inequalities and martingale inequalities: a survey. *Internet Mathematics*.

Colby, R. W.; and Meyers, T. A. 1988. *The encyclopedia of technical market indicators*. Dow Jones-Irwin Homewood, IL.

Dacrema, M. F.; Cremonesi, P.; and Jannach, D. 2019. Are We Really Making Much Progress? A Worrying Analysis of Recent Neural Recommendation Approaches. In *Proceedings of the 13th ACM Conference on Recommender Systems*, RecSys '19.

Ding, X.; Zhang, Y.; Liu, T.; and Duan, J. 2015. Deep learning for event-driven stock prediction. In *IJCAI*.

Dorogush, A. V.; Ershov, V.; and Gulin, A. 2018. CatBoost: gradient boosting with categorical features suppfort. *arXiv preprint arXiv:1810.11363*.

Fama, E. F.; and French, K. R. 1993. Common risk factors in the returns on stocks and bonds. *JFE*.

Farhangi, A.; Bian, J.; Huang, A.; Xiong, H.; Wang, J.; and Guo, Z. 2022. AA-Forecast: Anomaly-Aware Forecast for Extreme Events. *arXiv preprint arXiv:2208.09933*.

Feng, F.; He, X.; Wang, X.; Luo, C.; Liu, Y.; and Chua, T.-S. 2019. Temporal relational ranking for stock prediction. *TOIS*.

Feng, G.; Polson, N. G.; and Xu, J. 2018. Deep learning in asset pricing. *arXiv preprint*.

Friedman, J.; Hastie, T.; and Tibshirani, R. 2001. *The elements of statistical learning*.

Ge, R.; Jin, C.; et al. 2017. No spurious local minima in nonconvex low rank problems: A unified geometric analysis. In *ICML*.

Gong, R.; Fonseca, E.; Bogdanov, D.; Slizovskaia, O.; Gomez, E.; and Serra, X. 2017. Acoustic scene classification by fusing LightGBM and VGG-net multichannel predictions. In *Proc. IEEE AASP Challenge Detection Classification Acoust. Scenes Events*.

Goodfellow, I.; Bengio, Y.; and Courville, A. 2016. *Deep learning*. MIT press.

Grover, A.; and Leskovec, J. 2016. node2vec: Scalable feature learning for networks. In *KDD*.

Gu, S.; Kelly, B.; and Xiu, D. 2020. Empirical asset pricing via machine learning. *The Review of Financial Studies*.

Ha, C.-W. 1986. Eigenvalues of differentiable positive definite kernels. *SIAM Journal on Mathematical Analysis*.

Hamilton, A.; and Tegmark, M. 2000. Decorrelating the power spectrum of galaxies. *Monthly Notices of the Royal Astronomical Society*.

Han, Y.; He, A.; Rapach, D.; and Zhou, G. 2018. What Firm Characteristics Drive US Stock Returns? *Available at SSRN 3185335*.

Hochreiter, S.; and Schmidhuber, J. 1997. Long short-term memory. *Neural computation*.

Hoerl, A. E.; and Kennard, R. W. 1970. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1): 55–67.

Hu, Z.; Liu, W.; et al. 2018. Listening to chaotic whispers: A deep learning framework for news-oriented stock trend prediction. In *WSDM*.

Huang, D.; Li, J.; and Zhou, G. 2019. Shrinking factor dimension: A reduced-rank approach. *Available at SSRN 3205697*.

Hurst, H.; Black, R.; and Simaika, Y. 1965. Long-term storage: an experimental study Constable. *London UK*.

Kakushadze, Z. 2016. 101 formulaic alphas. *Wilmott*.

Ke, G.; Meng, Q.; Finley, T.; Wang, T.; Chen, W.; Ma, W.; Ye, Q.; and Liu, T.-Y. 2017. Lightgbm: A highly efficient gradient boosting decision tree. In *NeurIPS*.

Ke, G.; Xu, Z.; Zhang, J.; Bian, J.; and Liu, T.-Y. 2019. DeepGBM: A deep learning framework distilled by GBDT for online prediction tasks. In *KDD*.

Kelly, B. T.; Pruitt, S.; et al. 2019. Characteristics are covariances: A unified model of risk and return. *JFE*.

Koltchinskii, V.; Lounici, K.; et al. 2011. Nuclear-norm penalization and optimal rates for noisy low-rank matrix completion. *ANN STAT*.

Krishnapur, M. 2016. Anti-concentration inequalities. *Lecture notes*.

Laptev, N.; Yosinski, J.; Li, L. E.; and Smyl, S. 2017. Time-series extreme event forecasting with neural networks at uber. In *ICML*.

Li, C.; Wong, F.; Liu, Z.; and Kanade, V. 2017. From which world is your graph. In *NeurIPS*.

Li, Z.; Yang, D.; Zhao, L.; Bian, J.; Qin, T.; and Liu, T.-Y. 2019. Individualized indicator for all: Stock-wise technical indicator optimization with stock embedding. In *KDD*.

Liu, A.; Wu, Q.; Liu, Z.; and Xia, L. 2019. Near-neighbor methods in random preference completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 4336–4343.

Ming, F.; Wong, F.; Liu, Z.; and Chiang, M. 2014. Stock market prediction from WSJ: text mining via sparse matrix factorization. In *ICDM*.

Negahban, S.; and Wainwright, M. J. 2011. Estimation of (near) low-rank matrices with noise and high-dimensional scaling. *ANN STAT*.

Newey, W. K.; and West, K. D. 1986. A simple, positive semi-definite, heteroskedasticity and autocorrelation consistent covariance matrix. Technical report.

Orr, D.; and Mashtaler, I. 2012. CNE5.

Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; and Lerer, A. 2017. Automatic differentiation in pytorch.

Posner, R. A. 2014. *Economic analysis of law*.

Qiong, W.; Brinton, C. G.; Zhang, Z.; Pizzoferrato, A.; Liu, Z.; and Cucuringu, M. 2021. Equity2Vec: End-to-end Deep Learning Framework for Cross-sectional Asset Pricing. *International Conference on AI in Finance*.

Quinlan, J. R. 1986. Induction of decision trees. *Machine learning*.

Rastelli, R.; Friel, N.; and Raftery, A. E. 2016. Properties of latent variable network models. *Network Science*.

Rather, A. M.; Agarwal, A.; and Sastry, V. 2015. Recurrent neural network and a hybrid model for prediction of stock returns. *EXPERT SYST APPL*.

Rendle, S.; Zhang, L.; and Koren, Y. 2019. On the difficulty of evaluating baselines: A study on recommender systems. *arXiv preprint arXiv:1905.01395*.

Rudelson, M.; and Vershynin, R. 2010. Non-asymptotic theory of random matrices: extreme singular values. In *ICM*.

Sak, H.; Senior, A. W.; and Beaufays, F. 2014. Long short-term memory recurrent neural network architectures for large scale acoustic modeling.

Sejnowski, T. J. 2018. *The deep learning revolution*. MIT press.

Shen, H.; Oh, J.; Zhao, S.; Wang, G.; Taghavi, T.; and Lee, S. 2022. Learning Personalized Representations using Graph Convolutional Network. In *17th International Workshop on Mining and Learning with Graphs, co-located with KDD 2022*.

Stewart, G. W. 1990. Matrix perturbation theory.

Sussman, D. L.; Tang, M.; and Priebe, C. E. 2013. Consistent latent position estimation and vertex classification for random dot product graphs. *TPAMI*.

Tang, M.; Sussman, D. L.; Priebe, C. E.; et al. 2013. Universally consistent vertex classification for latent positions graphs. *ANN STAT*.

Tao, T.; and Series, M. L. 2009. Compressed Sensing. *University of California*.

Tibshirani, R. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society*.

Tsigler, A.; and Bartlett, P. L. 2020. Benign overfitting in ridge regression. *arXiv preprint arXiv:2009.14286*.

Tsybakov, A. B. 2008. *Introduction to nonparametric estimation*.

Wang, J.; Zhang, Y.; Tang, K.; Wu, J.; and Xiong, Z. 2019. AlphaStock: A Buying-Winners-and-Selling-Losers Investment Strategy using Interpretable Deep Reinforcement Attention Networks. In *KDD*.

Wu, Q.; Hare, A.; Wang, S.; Tu, Y.; Liu, Z.; Brinton, C. G.; and Li, Y. 2021. Bats: a spectral biclustering approach to single document topic modeling and segmentation. *TIST*.

Wu, Q.; Hsu, W.-L.; Xu, T.; Liu, Z.; Ma, G.; Jacobson, G.; and Zhao, S. 2019. Speaking with actions-learning customer journey behavior. In *ICSC*.

Wu, Q.; Hui, L. C.; Yeung, C. Y.; and Chim, T. W. 2015. Early car collision prediction in VANET. In *ICCVE*. IEEE.

Wu, Q.; Wong, F.; Liu, Z.; Li, Y.; and Kanade, V. 2020a. Adaptive Reduced Rank Regression. In *NeurIPS*.

Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; and Philip, S. Y. 2020b. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*.

Wüthrich, B.; Permunetilleke, D.; et al. 1998. Daily prediction of major stock indices from textual www data. *Hkie transactions*.

Xie, J.; Girshick, R.; and Farhadi, A. 2016. Unsupervised deep embedding for clustering analysis. In *International conference on machine learning*, 478–487. PMLR.

Yang, X.; and Ding, J. 2020. A computational framework for iceberg and ship discrimination: Case study on Kaggle competition.

Zhang, L.; Aggarwal, C.; and Qi, G.-J. 2017. Stock price prediction via discovering multi-frequency trading patterns. In *KDD*.

Zhao, S.; Hsu, W.-L.; Ma, G.; Xu, T.; Jacobson, G.; and Rustamov, R. 2020. Characterizing and Learning Representation on Customer Contact Journeys in Cellular Services. In *KDD*.

Zhou, X.; and Jain, S. 2014. *Active Equity Management*.