Multi-task Recurrent Neural Networks and Higher-order Markov Random Fields for Stock Price Movement Prediction

Chang Li
UBTECH Sydney AI Centre, SCS,
University of Sydney
Capital Markets CRC, Australia
chli4934@uni.sydney.edu.au

Dongjin Song*
NEC Laboratories America, Inc.
dsong@nec-labs.com

Dacheng Tao*
UBTECH Sydney AI Centre, SCS,
University of Sydney
dacheng.tao@sydney.edu.au

ABSTRACT

Stock price movement not only depends on the history of individual stock movements, but also complex hidden dynamics associated with other correlated stocks. Despite the substantial effort made to understand the principles of stock price movement, few attempts have been made to predict movement direction based upon a single stock's historical records together with its correlated stocks. Here, we present a multi-task recurrent neural network (RNN) with high-order Markov random fields (MRFs) to predict stock price movement direction. Specifically, we first design a multi-task RNN framework to extract informative features from the raw market data of individual stocks without considering any domain knowledge. Next, we employ binary MRFs with unary features and weighted lower linear envelopes as the higher-order energy function to capture higher-order consistency within the same stock clique (group). We also derive a latent structural SVM algorithm to learn higher-order MRFs in a polynomial number of iterations. Finally, a sub-gradient algorithm is employed to perform end-to-end training of the RNN and high-order MRFs. We conduct thorough empirical studies on three popular Chinese stock market indexes and the proposed method outperforms baseline approaches. To our best knowledge, the proposed technique is the first to investigate intra-clique relationships with higher-order MRFs for stock price movement prediction.

KEYWORDS

Deep learning, Graphical model, Multivariate time series prediction

ACM Reference Format:
Chang Li, Dongjin Song*, and Dacheng Tao*. 2019. Multi-task Recurrent Neural Networks and Higher-order Markov Random Fields for Stock Price Movement Prediction. In *The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '19), August 4–8, 2019, Anchorage, AK, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3292500.3330983

*Dongjin Song and Dacheng Tao are corresponding authors.

1 INTRODUCTION

It is well known that single the price movement of an individual stock not only depends on historical records but also highly correlated to other stocks [29, 32] and may change in a non-synchronous manner [7, 29]. This correlated yet asynchronous price movement is sometimes referred to as the lead-lag relationship [19] between a group of stocks and is thought to arise from the different speed of information diffusion[2, 29, 31]. When new information hits the market, some stocks react faster than others and identification of these leading stocks and their lead-lag relationships to other lagging stocks provides strong predictive evidence to the latter's price movement.

However, there are three key challenges in utilizing the lead-lag relationship: (1) discovering which stock will be affected by newly arriving information (such as news); (2) identifying the group (*e.g.*, industry, supply chain, *etc.*) it belongs to along with the leading and lagging stocks in this group and modeling their relationships; (3) predicting the price movement of each stock by jointly considering knowledge in the correlated group and an individual stock'price movement at that moment.

The first challenge is extremely difficult, not only because it requires an expert level of understanding of the finance system and market dynamics and the stock price, but also due to a lack of training data. However, according to the efficient market hypothesis [30], stock price reflects all available market information. Therefore, informative stock price changes can be employed as an approximate of market news arrival. In this way, the complexity of the first challenge is transformed to the detection of informative price changes in individual stocks.

Economists hitherto to used patterns hidden inside historical trading prices and volume to predict future price movements [11, 22]. As a result, hundreds of hand-crafted features, known as technical analysis indicators [23], have been designed. However, most of these models have stopped generating profitable signals since the early 1990s [35]. Since trading strategies based on technical analysis rules are publicly available and easy to replicate, informed institutional traders have been motivated to manipulate the market price and encourage retail (individual) traders to follow their manipulated price to generate excess profit [40].

To overcome these problems and address the first challenge, here we employ an end-to-end hierarchical multi-task [8] RNN to extract informative changes from raw market prices without using hand-crafted features such as technical analysis indicators. Good price prediction relies on rich representations and a multi-task framework that can leverage complementary aspects from diverse tasks [39]. Specifically, given raw market price data, which only contains six features (opening price, low price, high price, closing price, volume, and amount) at each time interval, we leverage a hierarchical multi-task network to first extract features on different tasks and then concatenate those complementary feature vectors to make the final prediction.

To model lead-lag relationships and address the other two challenges, we also present a binary Markov Random Fields (MRFs) with weighted lower linear envelopes as higher order (when the clique contains more than two nodes) energy functions [13, 14, 25, 34]. In our implementation, we treat each stock as a node in MRFs and each stock's group with lead-lag relationships as a maximum clique in MRFs. We use a pre-defined industry classification list [1] as the prior domain knowledge of each maximum clique for each stock. By using a weighted version of higher order functions, stocks have higher weights in the above list can be seen as leading stocks, and vice versa. Finally, the complexity of modeling dynamics between leading and lagging stocks becomes encouraging consistency over large cliques under weighted lower linear envelope potentials. Logits from hierarchical RNN networks are used as unary features in MRFs. By minimizing the energy function which contains both unary and higher order features, we can predict each stock's future price movement by jointly considering individual market price trends together with lead-lag relationships.

Unlike the first challenge trying to avoid prior knowledge, we consider being able to embed prior knowledge as an advantage. Definitions of sectors as well as leading and lagging stocks in each sector require solid financial industry research. Statistical evidence learned automatically from market price data are usually insufficient for determining such relationships.

We demonstrate the effectiveness of the proposed technique using three popular Chinese stock market indexes, and the proposed method outperforms baseline approaches. To our best knowledge, the proposed technique is the first one to investigate intra-clique relationships with higher-order MRFs on stock price movement prediction.

To summarize, the main contributions of this paper as follows:

- We propose a hierarchical multi-task RNN architecture to learn stock price patterns without hand-crafted features. To our best knowledge, this is the first work proposing a multi-task neural networks for stock price movement prediction.
- We propose the first model that encode lead-lag relationships between stocks using higher-order MRFs.
- We develop an algorithm to learn the weighted lower linear envelope with latent variables as a higher order energy function under the latent structural SVM framework. Adding latent variables to higher order functions enables our model to learn richer representations than previously study [14]. Furthermore, our algorithm is not limited to stock price movement prediction but can easily be applied to other time series tasks and computer vision tasks.

2 RELATED WORKS

This work is closely related to lead-lag relationships, multi-task learning, high-order MRFs, and latent structural SVMs.

Lead-lag relationships: Lead-lag relationships have long been recognized in the stock market. They can arise for many reasons such as information diffusion, sector (industry) rotation, investment style rotation, event-driven trading, and asynchronous trading [9, 10, 15, 29]. It is generally believed that lead-lag relationships are more prevalent in firms in the same industry [19], justifying our use of pre-defined industry classification list [1] as prior domain knowledge of each stock's maximum clique. Several studies [2, 7, 19, 31] have shown that stocks with larger capital size and higher liquidity tend to be leading stocks and vice versa. To replicate potential lead-lag relationships, we assign each stock a different weight from its corresponding indexes created by the China Securities Index Company, Ltd. More complicated dynamics hidden behind a clique of stocks are learned by higher-order MRFs.

Multi-task learning: Caruana [8] showed that inductive knowledge learned from multiple tasks can transfer between tasks and help improving generalization of all tasks. Many Natural Language Processing (NLP) tasks take advantage of multi-task frameworks and achieve state-of-the-art performance while using simple models for each of these tasks [17, 39]. However, as noted elsewhere [8, 38], there is a lack of theory on underpinning a diverse set of tasks and the hierarchical architecture of the chosen tasks. Recent works [17, 39] apply the principle that the task complexity should increase according to hierarchical level, and we do likewise. Because technical analysis indicators can be categorized into trend, momentum, volatility and volume [23], and volume is included in market price data, we propose an architecture that uses trend and volatility tasks as lower level tasks and price movement prediction (upward or downward) as a higher level task. Other task selection and hierarchical designations remain open for further research.

Higher-order Markov random fields: Markov random fields are also known as undirected graphical models that can be regarded as a regularized joint log-probability distribution of arbitrary non-negative functions over a set of maximal cliques of the graph [4]. Utilizing MRFs usually involves three steps: defining energy functions, solving inference problem (MAP or energy minimization) and learning parameters. With respect to energy functions, our work focuses on a class of higher-order potentials defined as a concave piecewise linear function which is known as lower linear envelope potentials over a clique of binary variables. It has been raising much interest due to its capability of encoding consistent constraints over large subsets of pixels in an image [25, 34]. We follow Gould [14] to construct a graph-cut algorithm to solve an exact inference problem and propose our novel learning algorithms under latent structural SVM in Section 4.1.

In the second step, in order to solve the inference problem, Kohli et al. [27] proposed a method to represent a class of higher order potentials with lower (upper) linear envelope potentials. By introducing auxiliary variables [24], they reduced the linear representation to a pairwise form and proposed an approximate algorithm with standard linear programming methods. However, they only show an exact inference algorithm on at most three terms. Following their approach, Gould [14] extended their method to a weighted lower linear envelope with arbitrary many terms solved with an efficient algorithm. They showed that the energy function with auxiliary variables is submodular by transforming it into a quadratic pseudo-Boolean form [5] and that graph-cuts like algorithms [6, 12, 16] can be applied for exact inference.

In the third step, Gould [14] solved the learning problem of the lower linear envelope under the max margin framework [43]. In their work, they highlighted the potential relationship between their auxiliary representation and latent SVM [47]; our work is closely based on their research. We continue to use the higher order energy function and inference algorithm [13] and extend their max margin learning algorithm to include latent variables. The learning algorithm used here is an extension of the max margin framework known as "latent structural SVM" [47].

Latent structural SVMs: The max-margin framework [42, 43] is a principled approach to learn weights of pairwise MRFs. Szummer et al. [41] adapted this framework to optimize pairwise MRFs parameters inferred by graph-cuts method. To adapt higher-order energy functions to max margin framework, Gould [13] approximated the energy function using equally spaced break-points. Gould [14] extended this framework with additional linear constraints to enforce concavity on the weights, thus allowing them to be used to learn lower linear envelope potentials. However, these methods only approximately learn higher-order functions. In this paper we propose an algorithm to optimize the energy function exactly by introducing auxiliary variables back into the feature vector and solving the learning problem using the latent structural SVM framework [47]. To include unobserved information, Yu and Joachims [47] extended the joint feature function in structural SVM with latent variables and re-wrote the objective function of SSVM into a difference of two convex functions. This formulation can be solved using the Concave-Convex Procedure (CCCP)[48] which is two-stages algorithm that guarantee to convergence to a local minimum.

In contrast to SVM, the latent structural SVM cannot directly use data. In order to use it, the inference algorithm, as well as the MRF feature function, loss function, and latent variable completion problem[47] must first be implemented. Our implementation is described in section 4.

3 METHODS

In this section, we introduce the multi-task RNN-MRFs architecture which is constructed with two parts. The first part is a Multi-task Market Price Learner, which consists of three dual stage attention based recurrent neural network (DARNN) [36] modules. The goal of the first part is to tackle the first challenge, *i.e.*, automatically extracting informative representations of the raw market price without considering any hand-crafted feature and technical indicator. The second part is an "Intra-clique Predictor" which is a binary Markov random fields model with weighted higher order energy functions. Those higher order functions are applied to sector lists (used as maximum cliques) defined by financial experts. The domain knowledge about leading stocks and lagging stocks are assigned as higher and lower weights in energy function accordingly. The goal of this part is to tackle the second and third challenges. Unary features learned by DARNN modules are jointly employed to maintain higher order consistency among stocks belonging to the same sector. The detailed architecture is shown in Figure 1.

3.1 Multi-task Market Price Learner

Stock price movement can be interpreted from many aspects such as investors sentiment, temporal patterns and cycles, flow of funds and market strength, *etc.* Ideal features should represent those

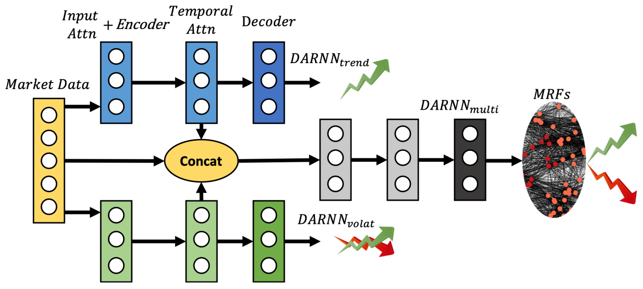


Figure 1: Multi-task RNN-MRFs architecture. Note that the output of DARNN$_{\text{multi}}$ only corresponds to one node's unary feature in MRFs.

aspects as much as possible. Multi-task learning has shown its effectiveness to learn inductive knowledge among tasks and improve performance as well as generalization capability [8]. Therefore, we propose a multi-task RNN framework entitled "Multi-task Market Price Learner (MMPL)" to tackle the first challenge: extracting informational representations from raw market price.

However, as noted elsewhere [8, 38], there is a lack of theory on underpinning a diverse set of tasks and the hierarchical architecture of the chosen tasks. We follow this intuition to construct our model. Most technical indicators fall into four categories: trend, momentum, volatility and volume [23]. Since volume is included in input for all low level tasks and we assume that momentum information can be learned by high level task, we propose an architecture that using trend and volatility tasks as our low level tasks and price movement prediction (upward or downward) as the high level task.

The "Multi-task Market Price Learner(MMPL)" contains two levels, three modules of DARNNs. DARNNs [36] are used as our basic module not only because of its capability of selecting relevant deriving series as well as temporal features, but also due to its superior performance for time series prediction compared to LSTM [18] and attention based LSTM [3]. Specifically, the bottom level contains two separate DARNN modules. They are supervised by low-level tasks which aim to predict future price as well as volatility based upon the raw market price data. The key difference among those modules is the loss function. At the top level, it is supervised by a high-level task that learns to use representations extracted by two low-level modules as well as raw market price data to predict positive / negative price movement of stocks. Logits of the last layer are passed to Intra-clique Predictor described in section 3.2 as unary features.

All three DARNN modules share the same raw market price data. Here we denote the time-series dataset as $\boldsymbol{X}$ where $\boldsymbol{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T) \in \mathbb{R}^{N \times T}$. Here $\boldsymbol{x}^n = (x_1^n, x_2^n, \ldots, x_T^n) \in \mathbb{R}^T$ denotes a driving series of T time-steps and $\boldsymbol{x}_t = (x_t^1, x_t^2, \ldots, x_t^N) \in \mathbb{R}^N$ denotes a snapshot at time-step t of all N features.

For both DARNN modules at the low level, the input matrix is a concatenation of exogenous matrix $\boldsymbol{X} \in \mathbb{R}^{5 \times T}$ which contains 5 exogenous driving series: opening price, low price, high price, volume, amount and 1 target series $\boldsymbol{y} = (y_1, y_2, \ldots, y_T) \in \mathbb{R}^T$. These two modules aim to predict target series y_{t+p} in the next p time steps:

$$\hat{y}_{t+p} = \text{DARNN}(y_1, \ldots, y_t, x_1, \ldots, x_t)$$

The target series $\boldsymbol{y}_{\text{trend}}$ of DARNN$_{\text{trend}}$ is closing price. The target series $\boldsymbol{y}_{\text{volat}}$ of DARNN$_{\text{volat}}$ is the standard deviation of closing price over M constant time-steps. In our implementation we set $M = 10$. We use Mean Squared Error (MSE) as the loss function to train those two modules separately.

To construct the high level DARNN module, which aims to predict the price movement, we concatenate context vectors $\boldsymbol{c}_T$ from each of low level DARNN module's encoder and raw market price matrix as the input. The target series $\boldsymbol{y}^{\text{binary}}$ is constructed by the sign function $y_t^{\text{binary}} = sign(y_{t+p} - y_t)$ where y_t denotes closing price at time-step t. We use cross-entropy as loss function to train the final DARNN$_{\text{multi}}$. Logits (outputs before going through *softmax*) of DARNN$_{\text{multi}}$ are then passed to "Intra-clique Predictor" as unary features.

In order to train MMPL together with MRFs in an end-to-end manner, we follow the subgradient method proposed by Witoonchart and Chongstitvatana [44]. Since our inner loop proposed in section 4.2 is actually a Structural SVM. Only gradients of parameters and feature functions need to be re-calculated. In our framework, outputs of MMPL are only used as unary features in MRFs' energy functions, our back-propagation rules can be defined by taking derivative of equation (3):

$$\frac{\partial L}{\partial \boldsymbol{w}^U} = \psi^U(y) - \psi^U(y^*) \tag{1}$$

where y is the ground-truth label and y^* is inferenced label. ψ^U is unary feature function described in section 3.2.1, here it contains logits calculated from DARNN$_{\text{multi}}$. $\boldsymbol{w}$ is parameter vector of DARNN$_{\text{multi}}$. And gradients of parameters can be calculated by

$$\frac{\partial L}{\partial \psi^U} = \boldsymbol{w}^U \tag{2}$$

Equations (1) and (2) can be directly plugged into sub-gradient algorithm proposed in [44]. Other configurations stay the same with their algorithm.

3.2 Intra-clique Predictor

In this section, we show how to construct an "Intra-clique Predictor" to model lead-lag relationships and address the other two challenges as mentioned in the introduction. Specifically, we present a binary Markov Random Fields (MRFs) with weighted lower linear envelopes as higher order (when the clique contains more than two nodes) energy functions. Note that the algorithm proposed here is a general framework for classification tasks. Besides time series classification, it can also be applied to computer vision tasks.

3.2.1 Higer Order Energy: The Weighted Lower Linear Envelope Function. Energy functions can be decomposed over nodes $\mathcal{N}$, edges $\mathcal{E}$ and higher order cliques C [41]. Let $\boldsymbol{w}$ be vector of parameters and ψ be arbitrary feature function, then the energy can be decomposed as a set of linear combinations of weights and feature vectors:

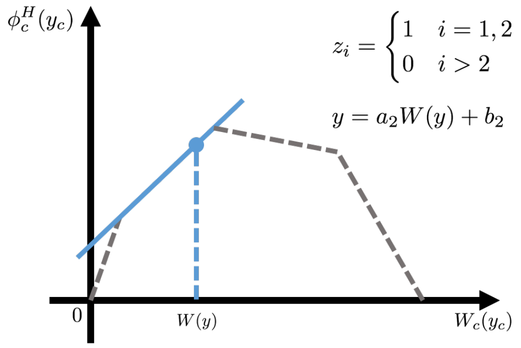


Figure 2: Example piecewise-linear concave function of $W_c(\boldsymbol{y}_c) = \sum_{i \in c} w_i^c y_i$. Assume the second linear function is active namely $z^c = (1, 1, 0, 0)$ (equation 8). The result of linear combination of parameter vector and feature vector is same as quadratic psuedo-Boolean function.

$$E(\boldsymbol{y}; \boldsymbol{w}) = \sum_{i \in \mathcal{N}} \boldsymbol{w}_i^U \psi^U(\boldsymbol{y}_i) + \sum_{(i,j) \in \mathcal{E}} \boldsymbol{w}_{ij}^P \psi^P(\boldsymbol{y}_i, \boldsymbol{y}_j) + \sum_{\boldsymbol{y}_C \in C} \boldsymbol{w}_C^H \psi^H(\boldsymbol{y}_C) \tag{3}$$

where U denotes *unary* terms, P denotes *pairwise* terms, H denotes *higher order* terms. In this section we mainly focus on one class of higher-order potentials ψ^H defined as a concave piecewise linear function which is known as *lower linear envelope potentials*. This has been studied extensively in Markov Random Fields area for encouraging consistency over large cliques [13, 25, 34].

Let C denotes the set of all maximal cliques in an image and $\boldsymbol{y}_c = \{y_i | \text{ for } i \in C_j\}$ denotes set of binary random variables where $y_i \in \{0, 1\}$ in clique C_j, a weighted lower linear envelope potential over $\boldsymbol{y}_c$ is defined as the minimum over a set of K linear functions as:

$$\psi_c^H(\boldsymbol{y}_c) = \min_{k=1,\dots,K} \{a_k W_c(\boldsymbol{y}_c) + b_k\}. \tag{4}$$

where $W_c(\boldsymbol{y}_c) = \sum_{i \in c} w_i y_i$ with $w_i^c \geq 0$ and $\sum_{i \in c} w_i^c = 1$ which are weights for each clique. $(a_k, b_k) \in \mathbb{R}^2$ are the linear function parameters. We illustrate an example with four linear functions in Figure 2.

Inference on energy function contains lower linear potentials is the same as the standard equation (3) and is given by:

$$\boldsymbol{y}^* = \operatorname{argmin} E(\boldsymbol{y}) \tag{5}$$

To ensure potentials do not contain redundant linear functions (functions that would never be active), Gould [14] proposed a constraint on parameters of the envelope. The k-th linear function is not redundant if the following condition is satisfied:

$$0 < \frac{b_k - b_{k-1}}{a_{k-1} - a_k} < \frac{b_{k+1} - b_k}{a_k - a_{k+1}} < 1. \tag{6}$$

Another important property of equation (5) is shift invariant (vertically). We write $\widetilde{\psi}_c^H(\boldsymbol{y}_c)$ by shift equation (4) vertically with an

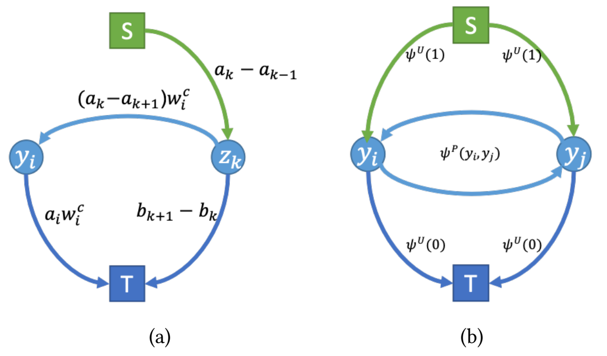


Figure 3: st-graph construction for equation (9), unary and pairwise terms. Every cut corresponds to an assignment to the random variables, where variables associated with nodes in the $\mathcal{S}$ set take the value one, and those associated with nodes in the $\mathcal{T}$ set take the value zero. With slight abuse of notation, we use the variables to denote nodes in our graph.

abitrary amount $b^{const} \in R$

$$\widetilde{\psi}_c^H(\boldsymbol{y}_c) = \min_{k=1,\dots,K} \left\{ a_k W_c(\boldsymbol{y}_c) + b_k + b^{\text{const}} \right\}$$

Then we have

$$\operatorname*{argmin}_{\boldsymbol{y}_c} \psi_c^H(\boldsymbol{y}_c) = \operatorname*{argmin}_{\boldsymbol{y}_c} \widetilde{\psi}_c^H(\boldsymbol{y}_c). \tag{7}$$

Therefore, in the following discussion without loss of generality we assume $b_1 = 0$ thus $b_k \geq 0$ for $k = 1, \dots, n$.

3.2.2 *Exact Inference.*

Exact inference on MRFs has been extensively studied in past years. Researchers found that, energy functions which can be transformed into quadratic pseudo-Boolean functions [20, 21, 37] are able to be minimized exactly using *graph-cuts* like algorithms [12, 16] when they satisfy submodularity condition [5]. Kohli et al. [26] and Gould [13] adapted those results to perform exact inference on lower linear envelope potentials. In this section we mainly focus on describing the *st min cut* graph constructed by Gould [13, 14] for exact inference (5) of energy function containing lower linear envelope potentials.

Following the approach of Kohli and Kumar [24], Gould [13, 14] transformed the weighted lower linear envelope potential (4) into a quadratic pseudo-Boolean function by introducing $K - 1$ auxiliary variables $\boldsymbol{z} = (z_1, \dots, z_{K-1})$ with $z_k \in \{0, 1\}$:

$$\begin{aligned} E^c(\boldsymbol{y}_c, \boldsymbol{z}) = {} & a_1 W_c(\boldsymbol{y}_c) + b_1 \\ & + \sum_{k=1}^{K-1} z_k \left((a_{k+1} - a_k) W_c(\boldsymbol{y}_c) + b_{k+1} - b_k \right) \end{aligned} \tag{8}$$

for a single clique $c \in C$. Under this formulation, minimizing the pseudo-Boolean function over $\boldsymbol{z}$ is equivalent to selecting (one of) the active functions(s) from equation (4). Another important property of optimized $\boldsymbol{z}$ under this formulation is that it automatically satisfies the constraint: $z_{k+1} \leq z_k$. This property give rise to further development of parameter vector and feature vector (equation (12) and (13)) which are used in latent structural SVM.

In order to construct the *st-min-cut* graph, we rewrote equation (8) into *posiform* [5]:

$$\begin{aligned} E^c(\boldsymbol{y}_c, \boldsymbol{z}) = {} & b_1 - (a_1 - a_K) + \sum_{i \in c} a_1 w_i^c y_i \\ & + \sum_{k=1}^{K-1} (b_{k+1} - b_k) z_k + \sum_{k=1}^{K-1} (a_k - a_{k+1}) \bar{z}_k \\ & + \sum_{k=1}^{K-1} \sum_{i \in c} (a_k - a_{k+1}) w_i^c \bar{y}_i z_k \end{aligned} \tag{9}$$

where $\bar{z}_k = 1 - z_k$ and $\bar{y}_i = 1 - y_i$. a_1 is assumed to be greater than 0 so that all coefficients are positive (recall we assume $b_1 = 0$ in section 3.2.1 and we have $a_k > a_{k+1}$ and $b_k < b_{k+1}$). Since the energy function (9) is submodular, the *st-min-cut* graph can be constructed based on equation (9). The construction (including unary and pairwise) is explained in Figure 3.

4 OPTIMIZATION

4.1 Transforming Between Representations

With the inference algorithm in hand, we now can develop the learning algorithm for weighted lower linear envelope potentials using the latent structural SVM framework. We begin by transforming the equation (8) into a linear combination of parameter vector and feature vector. Then a two-step algorithm was developed to solve the latent structural SVM.

The latent structural SVM formulation requires that the energy function be formulated into a linear combination of features and weights while our higher-order potential is represented as the minimum over a set of linear functions. However, in 3.2.2 we reformulated the piesewise linear functions into a quadratic pseudo-Boolean function (8) by introducing auxiliary variables. Now we show function (8) itself is an inner product of parameter vector and feature vector with latent information. Note that the function can be expanded as a summation of $2K - 1$ terms:

$$\begin{aligned} E^c(y_c, z) = {} & a_1 W_c(y_c) + b_1 \\ & + \sum_{k=1}^{K-1} z_k((a_{k+1} - a_k) W_c(y_c) + b_{k+1} - b_k) \\ = {} & a_1 W_c(y_c) + \sum_{k=1}^{K-1} (a_{k+1} - a_k) z_k W_c(y_c) \\ & + \sum_{k=1}^{K-1} (b_{k+1} - b_k) z_k \end{aligned} \tag{10}$$

Here we use the fact of equation (7) and let $b_1 = 0$. Now we can reparameterize the energy function as

$$E^c(\boldsymbol{y}_c, \boldsymbol{z}; \boldsymbol{\theta}) = \boldsymbol{\theta}^T \psi(\boldsymbol{y}_c, z) \tag{11}$$

where:

$$\theta_k = \begin{cases} a_1 & \text{for } k = 1 \\ a_k - a_{k-1} & \text{for } 1 < k \leq K \\ b_{k+1-K} - b_{k-K} & \text{for } K < k \leq 2K - 1 \end{cases} \tag{12}$$

$$\psi_k = \begin{cases} W_c(\boldsymbol{y}_c) & \text{for } k = 1 \\ W_c(\boldsymbol{y}_c)z_k & \text{for } 1 < k \le K \\ z_k & \text{for } K < k \le 2K - 1 \end{cases} \tag{13}$$

Under this formulation, inference problems in [47] can be written as:

$$(\hat{\mathbf{y}}_k(\boldsymbol{\theta}), \hat{\mathbf{z}}_k(\theta)) = \underset{(\mathrm{y}\times\mathrm{z})\in\mathcal{Y}\times\mathcal{Z}}{\operatorname{argmin}} \ \boldsymbol{\theta}^T \cdot \psi(\mathbf{y}_k, \mathbf{z}_k) \tag{14}$$

and

$$\mathbf{z}^*_k(\boldsymbol{\theta}) = \underset{\mathrm{z}\in\mathcal{Z}}{\operatorname{argmin}} \, \boldsymbol{\theta}^T \cdot \psi(\mathbf{y}_k, \mathbf{z}_k) \tag{15}$$

There are 2 facts worth to mention. The first fact is that in our previous construction of minimum-st-cut graph the latent variable z is already included. Therefore, we can apply our inference algorithm directly on our 2 new formulations.

More interesting for equation (15) there exists more efficient algorithm. At training stage the ground-truth labels y_i is a function input thus completely observed. Therefore, the term $((a_{k+1} - a_k)W_c(\boldsymbol{y}_c) + b_{k+1} - b_k)$ in equation (10) becomes constant. So we can infer latent variable z explicitly by:

$$z^c_k = \begin{cases} 0 & \text{if } ((a_{k+1} - a_k)W_c(y_c) + b_{k+1} - b_k) \ge 0 \\ 1 & \text{otherwise.} \end{cases} \tag{16}$$

Therefore, assignments inferred by graph-cut algorithm can be directly encoded into a linear combination by using our latent structural SVM formulation for learning purpose. The remaining task is to ensure the concavity of $\boldsymbol{\theta}$. We do this by adding following constraint:

$$A\boldsymbol{\theta} \ge \epsilon, \ \ A = \begin{bmatrix} \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{P} \end{bmatrix} \in \mathbb{R}^{(2K-1)\times(2K-1)} \tag{17}$$

where $\mathbf{-1}$ is a matrix of size $(K - 1) \times (K - 1)$ and $\mathbf{P}$ is an identity matrix of size $(K-1)\times(K-1)$. One subtle problem we found during experiments is that the algorithm can be stuck with small numerical value. To avoid this we add small slack variables $\epsilon = \mathbf{1}^{-15}$ on those constraints.

4.2 Latent Structural SVM Learning

With the inner product formulation (equation (11)) of higher order energy function in hand, we now able to develop our latent structural SVM learning algorithm. The energy function (higher order function together with unary and pairwise functions) can be written as:

$$E_{all}(y, z) = \begin{bmatrix} \boldsymbol{\theta}^H \\ \theta^{unary} \\ \theta^{pairwise} \end{bmatrix}^T \cdot \begin{bmatrix} \psi^H \\ \psi^{unary} \\ \psi^{pairwise} \end{bmatrix} = \theta^T_{all} \cdot \psi_{all} \tag{18}$$

where $\boldsymbol{\theta}^H \in \mathbb{R}$ is the parameter vector in higher order equation (11) of size $2K - 1$. θ^{unary} and $\theta^{pairwise}$ are both scalars. $\psi^{\text{unary}} = \sum_i \psi^U_i(y_i)$ and $\psi^{\text{pairwise}} = \sum_{ij} \psi^P_{ij}(y_i, y_j)$. Therefore, the size of θ_{all} is $2K + 1$.

Following Yu and Joachims [47], we use the two stages Concave-Convex Procedure (CCCP) [48] to solve the optimization problem. We first imputes the latent variables z explicitly by equation (15). Namely solving the "latent variable completion" problem [47]:

$$z^*_i = \underset{\mathrm{z}\in\mathcal{Z}}{\operatorname{argmax}} \, \theta \cdot \psi(\mathbf{y}_i, \mathbf{z}) \tag{19}$$

The inference result z^*_i for $i = 1, \ldots, n$ is used as completely observed for later stage. With the latent variable z^*_i which best explains the ground-truth data y_i in hand, updating the parameter vector $\boldsymbol{\theta}$ reduces to solve the standard structural SVM problem:

$$\min_{\theta} \left(\frac{1}{2}\|\theta\|^2 + C\sum_{i=1}^{n} \Big(\max_{(\hat{\mathrm{y}}\times\hat{\mathrm{z}})\in\mathcal{Y}\times\mathcal{Z}} [\theta \cdot \psi(\hat{\mathbf{y}}, \hat{\mathbf{z}}) + \Delta(\mathbf{y}_i, \hat{\mathbf{y}}, \hat{\mathbf{z}})] \Big) \right) \tag{20}$$
$$-C\sum_{i=1}^{n} \left(\theta \cdot \psi(\mathbf{y}_i, \mathbf{z}^*_i)\right)$$

Our optimization algorithm is summarized in Algorithm 2. As we mentioned in Appendix A, although we proposed an end-to-end subgradient algorithm is section 3.1, MRFs updated by such algorithm take too many iterations to converge. Therefore, we propose a two-stage training procedure. At first stage, MMPL and MRFs are trained separately. Therefore, MRFs can take advantage of the efficient latent structural SVM and converge in a polynomial number of iterations. After all those models are converged, we then combine them together to conduct end-to-end training. Note that the CCCP Inner Loop in Algorithm 2 is actually solving standard structural SVM problem. Therefore, at the second stage, we use subgradient algorithm proposed in section 3.1 to replace the CCCP Inner Loop. Other settings remain the same.

The last problem remaining is the initialization method. Because our objective function (20) is not convex and the CCCP algorithm is only guaranteed to converge to a local minimum or saddle point[48], initialization of $\boldsymbol{\theta}$ might affect the performance of our algorithm. Since there are no theoretical solution for this problem, we propose an empirical initialization algorithm in Appendix A.1.

5 EXPERIMENT

In this section, we first introduce 3 stock datasets (index) and how we use them to construct our final input datasets. Then, we introduce the parameter settings of our model and other training details. Finally, we select four evaluation metrics and use them to demonstrate our framework's effectiveness by comparing with several baseline methods.

5.1 Dataset and Model Settings

To demonstrate the effectiveness of higher order consistency, we choose three exclusive and the most famous stock indexes on Chinese stock market to build our input datasets. Their index codes are: CSI (China Securities Index) 200, CSI 300 and CSI 500 which contain 200, 500 and 300 constituent stocks respectively. The CSI 300 index selects most liquid A-share stocks. It aims to reflect the overall performance of China A-share market. The CSI 200 and 500 indexes aim to reflect the overall performance of mid-to-large and small-to-mid capital A-shares respectively.

All these indexes are exclusive and are refined on a yearly basis. In this paper, we use fixed versions on 30-JAN-2015. We then collect their constituent stocks' minute-level data from 05-JAN-2015 to 29-DEC-2017. On Chinese stock market each trading day has 4 trading hours. So there are 240 samples (minutes) for each normally traded stock on each day. Each sample contains 6 features: opening price, high price, low price, closing price, volume, and amount. [1] For each stock, the first 80% days are used to construct the training set and the last 20% days are used as test set. Approximately training set and test set contain 33.6 million and 4.2 million samples, respectively. 49.5% of them are positive movements, 0.3% of them stay unchanged and 50.2% of them are negative movements. For binary classification task, we follow Mitchell and Pulvino [33]'s approach and label all positive movement samples 1 and 0 for the other samples. More labeling details are described in appendix A.

Table 1: Technical Indicators Selection

Category	Indicator Name
Momentum	Awesome Oscillator, Money Flow Index
Volume	Chaikin Money Flow On-balance volume mean
Volatility	Bollinger Bands (Upper and Lower Bands)
Trend	Average Directional Movement Index Moving Average Convergence Divergence

To demonstrate benefits of multi-task RNN over manually designed technical indicators, we also need to construct technical indicators datasets for each of those market price dataset collected above. We select 8 most popular indicators, 2 from each category [23] shown in Table 1. In implementation, we use open source package *Technical Analysis Library in Python*[2] to calculate those indicators and all hyperparameters are using package's default settings without any prior expert knowledge involved with. After technical indicators calculation, these 8 new features are concatenated to above market price dataset (5 features at each minute). So the final input dataset for each single task model contains 13 features in total. Before feeding into models, we normalize each stock with *z-score* function using standard deviation and mean calculated in the training set.

For brevity, we denote market price dataset which only contains 5 features as **Market** and the concatenated 13 features dataset as **Indicator**. As discussed in section 3.1, closing price at time t can be directly used as regression target for $\text{DARNN}_{\text{trend}}$. Standard deviation of closing price with a window size of 10 is used as regression target for $\text{DARNN}_{\text{volat}}$. The dimensions of hidden state and cell state are 32 for $\text{DARNN}_{\text{trend}}$ as well as $\text{DARNN}_{\text{volat}}$ and 128 for $\text{DARNN}_{\text{multi}}$. More training details are described in appendix A.

5.2 Results

In order to demonstrate the effectiveness of our framework, we compare 3 baseline methods, *i.e.*, LSTM [18], attention based LSTM Encoder_Decoder [3], and DARNN [36] on 3 different Chinese Securities Indexes with and without technical analysis indicators as inputs. Results are summarized in Table 2. All results are reported over the test sets. We select four metrics (Accuracy, Precision, Recall and F1 Score) as evaluation metrics to justify the effectiveness of the proposed approach. They are calculated by collecting all predicted labels of constituent stocks in each CSI index.

5.2.1 Effectiveness of multi-task framework. As mentioned earlier, to demonstrate effectiveness of multi-task framework, we use **Indicator** dataset, which contains both market price data and technical analysis indicators as inputs for baseline approaches and **Market** dataset which only contains market price data as inputs for MMPL (multi-task RNN) as well as baseline methods. For DARNN, we use a hidden size of 128. MMPL's configuration is described in section A.2. As we can see in Table 2, single task models (LSTM, LSTM Encoder_Decoder, DARNN) tested on **Market** dataset (without technical analysis indicators as inputs) generally have worse performance on all 4 metrics. In particular, performance of DARNN models tested on **Indicator** dataset is consistently better than the ones on **Market** dataset. This proves that even with hand-crafted features, deep learning models can still benefit from diversified and complementary features.

To test the effectiveness of multi-task framework, we conduct ablation study with only one low level task ($\text{DARNN}_{\text{trend}}$ or $\text{DARNN}_{\text{volat}}$) together with the high level task module $\text{DARNN}_{\text{multi}}$. Results indicate that these two variants have comparable or slightly worse result than DARNN on **Market**. This may because single task model does not provide diversified features while have more parameters than DARNN. Finally, MMPL outperforms all single task models and baseline methods on **Market**. This suggests that diversified and complementary tasks can help MMPL extract effective features. Specifically, by comparing MMPL and DARNN on **Market** as well as **Indicator**, we can see that MMPL generally outperforms DARNN on CSI200 and CSI300 indexes and is slightly worse on CSI500 index. We can conclude that by using multi-task RNN, we can extract better or at least comparable features compared with hand-crafted features.

5.2.2 Effectiveness of higher-order MRFs. In Table 2, we can observe that MMPL-MRFs framework consistently outperforms other baselines on all 3 CSI index constituent stocks. It shows evidence that higher-order energy function can help with encoding clique level consistency thus improving overall prediction performance. One interesting point to note is that the recall rate of MMPL-MRFs is constantly lower than other baselines. This can be seen as a trade off between accuracy and recall rate. However, it is worth to mention that for stock price movement prediction, high accuracy and precision are much preferred than recall rate. Another interesting phenomenon is that MMPL-MRFs gives higher improvements on CSI200 and CSI300 while little improvements over DARNN trained with technical analysis indicators on CSI500. One possible reason is that CSI200 and CSI300 select most liquid and representative stocks in Chinese stock market. Those stocks exhibit much stronger and higher order consistency than illiquid stocks. CSI500 selects small-mid capital stocks which are less liquid and contains much more noisy movements.

During training time, our algorithm converges in from 4 to 19 CCCP outer loops. The average inference time of graph-cut algorithm is 34 seconds.

[1]During this period, there are some stocks de-listed (SZ000024, SH600485, SH600832 in CSI 200; SZ000693, SZ000748, SZ000982 in CSI 500; SH600485, SH600832, SZ000024, SH601299 in CSI 300). Therefore, in total we collect 197, 497 and 296 stocks during this period respectively.

[2]https://github.com/bukosabino/ta

Table 2: Results: Baselines and ablation study. All models have a window size (lag steps) of 20 and predict price movement label at the next time step.

Data Set	Models	Chinese Securities Index (CSI)											
		CSI200				CSI500				CSI300			
		Accuracy	Precision	Recall	F1 Score	Accuracy	Precision	Recall	F1 Score	Accuracy	Precision	Recall	F1 Score
Indicator	LSTM	62.30	73.82	70.70	72.23	60.35	68.56	70.03	69.29	60.16	71.39	68.50	69.91
	LSTM Encoder_Decoder	64.26	72.41	**74.34**	73.36	61.13	75.63	66.67	70.86	64.26	75.54	70.57	72.97
	DARNN	63.09	72.08	73.55	72.81	66.60	78.98	**74.13**	**76.48**	65.82	76.68	73.46	75.04
Market	LSTM	57.62	67.57	67.37	67.47	55.86	68.10	64.53	66.27	56.25	67.17	65.98	66.57
	LSTM Encoder_Decoder	59.57	71.60	66.86	69.15	58.40	69.53	68.12	68.81	61.33	71.87	68.91	70.36
	DARNN	61.13	71.26	71.47	71.37	63.09	77.04	67.87	72.16	63.87	72.09	73.59	72.83
	DARNN$_{\text{trend}}$ + DARNN$_{\text{multi}}$	63.67	74.77	71.38	73.04	62.89	75.30	69.38	72.22	62.69	73.84	66.56	70.00
	DARNN$_{\text{volat}}$ + DARNN$_{\text{multi}}$	62.50	73.97	71.06	72.49	62.30	74.46	69.20	71.74	61.91	72.98	68.51	70.67
	MMPL	65.04	74.04	73.39	73.72	65.43	76.04	72.80	74.38	66.60	71.67	**78.90**	75.11
	MMPL+MRFs	**67.97**	**77.51**	73.91	**75.67**	**66.80**	**79.65**	72.78	76.06	**68.95**	**78.55**	74.71	**76.58**

5.2.3 Visualization of higher-order consistency. In order to further investigate higher-order MRFs' effectiveness, we design a heat-map to visualize CSI300 index intra-clique higher-order relationship in figure 4.

We first select two sectors: nonferrous metal sector, which contains 10 constituent stocks, and infrastructure sector, which contains 35 constituent stocks from CSI300 index [3]. We then measure consistency level between each two of these constituent stocks. In order to capture their temporal relationship, we propose a novel consistency measure which is calculated on temporal intervals.

Let $\boldsymbol{y}_i^T = \{y_i^1, y_i^2, ..., y_i^T\}$ denotes time-series for stock i. $y_i^t \in \{0, 1\}$ is the binary price movement label at time t. We segment time-series $\boldsymbol{y}_i^T$ into $N = \lceil \frac{T}{P} \rceil$ non-overlapping intervals $\{y_i^n, y_i^{n+1}, ..., y_i^{n+P}\}$ with fixed length P. For any two stocks i and j, we calculate the difference $d_{ij}^n = \sum_n^{n+P} y_i^n - \sum_n^{n+P} y_j^n$ of how many times positive price movement happen in the nth time interval in each stock. Then the consistency level c_{ij} between stocks i and j can be calculated as a $\ell_1$norm:

$$c_{ij} = -\|\mathbf{d}_{ij}\|_1$$

where $\mathbf{d}_{ij} = \{d_{ij}^1, d_{ij}^2, ..., d_{ij}^N\}$. We normalize c_{ij} into interval $[-1, 1]$. Each entry in figure 4 denotes a consistency level measure c_{ij}. The larger the c_{ij} is, the higher of consistency level between stock i and stock j, the color of corresponding entry is closer to red, and vice versa. As we mentioned, the average duration of information arrival-conduction-integration-release process is 4.04 minutes [45]. Since which stock is leading at each time interval is elusive, we set $P = 9$ when calculating consistency measures.

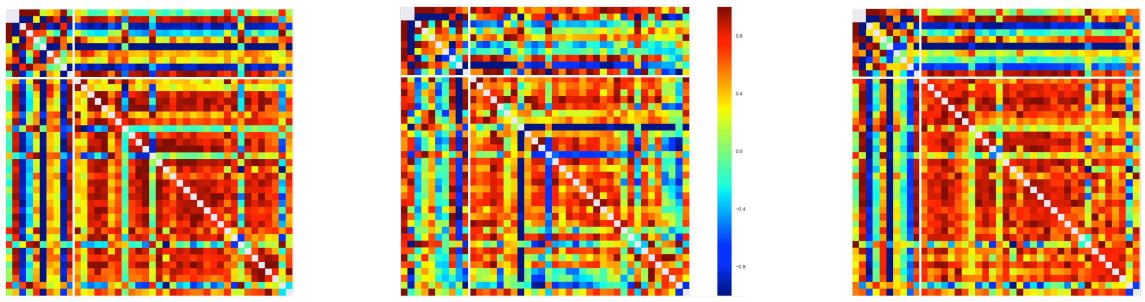
As we can see in figure (a), there is a significant red square area, which means ground-truth heat-map shows strong intra-clique consistency. This is an evidence that higher-order relationships do exist within clique of stocks. However, in figure (b), the red square area is fragmented into many little pieces. The whole area's color is closer to blue when compared to ground-truth heat-map, which means that MMPL captures little higher-order consistency. The reason we still can observe a shape of red square is that the accuracy of MMPL model on CSI300 is 66.6%. However, we can still conclude that the accuracy of single MMPL model mainly comes from unary features and it fails to capture higher order consistency relationships lies in clique of stocks. On the contrary, even though MMPL-MRFs model's accuracy on CSI300 index is only 2.35% better than MMPL model, we can observe that heat-map (c) is more close to ground-truth heat-map than heat-map (b). There is a much clear red square and the number of small fragments in red area is also less than figure (b). We can conclude that MMPL-MRFs models learn to utilize both unary features from MMPL as well as higher-order relationships encoded in MRFs.

[3]These two sectors are selected only because painting many sectors in one figure would be too messy to interpret and those two sectors have appropriate clique size (number of stocks) for visualization. Conclusions from these two sectors also apply to other sectors

6 CONCLUSIONS

Here we show how to model individual stock price predictions without hand-crafted features and encode lead-lag relationships between stocks using weighted higher-order MRFs. A multi-task neural network framework - Multi-task Market Price Learner (MMPL) - is proposed to automatically extract diversified and complementary features from individual stock price sequences. Features learned by MMPL are passed to a binary MRF with a weighted lower linear envelope energy function to utilize intra-clique higher-order consistency between stocks. An efficient latent structural SVM algorithm is designed to learn MRFs in polynomial time. Finally, the MRFs and MMPL are trained end-to-end using the sub-gradient algorithm. Extensive experiments are conducted on three major Chinese stock market indexes, and the proposed MMPL-MRFs achieve the best accuracy on all three indexes.

Our work provides a number of directions for future research. In this work we proposed a multi-task neural network for stock price prediction. While we directly use DARNN as a proof of concept, other, more dedicated architectures are worthy of exploration. Another obvious extension is to apply our method to multi-label MRFs to help with three-class price movement prediction. As well as time series tasks, we can also investigate how the latent SSVM framework performs on computer vision tasks. Another interesting direction is to investigate the implicit relationship between the



(a) Ground-truth (b) MMPL (c) MMPL-MRFs

Figure 4: Higher order consistency visualization. Figure (a) is calculated directly from ground truth labels on test set. Figure (b) is calculated using predicted labels of MMPL without MRFs on the test set. In figure (c), we use predicted labels of MMPL-MRFs on test set as inputs.

expert-defined index list and graph RNN [46], which could further help to reduce the domain knowledge required by our framework.

ACKNOWLEDGMENTS

We would like to thank Stephen Gould very much for explaining higer-order MRFs. We would also like to thank Peerajak Witoonchart for his explanation of sub-gradient SSVMs algorithm.

We acknowledge data provision and MQD research infrastructure support from CMCRC. Li is grateful for funding from the CMCRC. This work is also funded by China Hebei Provincial Department of Human Resources. Project ID: CL201723.

REFERENCES

[1] [n. d.]. TongHuaShun Industry Classification. http://q.10jqka.com.cn/thshy/. Accessed: 2019-01-15.
[2] Swaminathan G Badrinath, Jayant R Kale, and Thomas H Noe. 1995. Of shepherds, sheep, and the cross-autocorrelations in equity returns. *The Review of Financial Studies* 8, 2 (1995), 401–430.
[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).
[4] Christopher M. Bishop. 2006. *Pattern Recognition and Machine Learning*. Springer.
[5] Endre Boros and Peter L. Hammer. 2002. Pseudo-boolean optimization. *Discrete Applied Mathematics* 123 (2002), 155–225. Issue 1-3.
[6] Yuri Y. Boykov and Marie-Pierre Jolly. 2001. Interactive Graph Cuts for Optimal Boundary and Region Segmentation of Objects in N-D Images. In *ICCV*.
[7] Michael J Brennan, Narasimhan Jegadeesh, and Bhaskaran Swaminathan. 1993. Investment analysis and the adjustment of stock prices to common information. *The Review of Financial Studies* 6, 4 (1993), 799–824.
[8] Richard A Caruana. 1993. Multitask connectionist learning. In *In Proceedings of the 1993 Connectionist Models Summer School*. Citeseer.
[9] Tarun Chordia and Bhaskaran Swaminathan. 2000. Trading volume and cross-autocorrelations in stock returns. *The Journal of Finance* 55, 2 (2000), 913–935.
[10] Jennifer Conrad and Gautam Kaul. 1988. Time-variation in expected returns. *Journal of business* (1988), 409–425.
[11] Eugene F Fama and Marshall E Blume. 1966. Filter rules and stock-market trading. *The Journal of Business* 39, 1 (1966), 226–241.
[12] Daniel Freedman and Petros Drineas. 2005. Energy Minimization via Graph Cuts: Settling What is Possible. In *CVPR*.
[13] Stephen Gould. 2011. Max-margin Learning for Lower Linear Envelope Potentials in Binary Markov Random Fields. In *ICML*.
[14] Stephen Gould. 2015. Learning Weighted Lower Linear Envelope Potentials in Binary Markov Random Fields. *PAMI* 37, 7 (2015), 1336–1346.
[15] Allaudeen Hameed. 1997. Time-varying factors and cross-autocorrelations in short-horizon stock returns. *Journal of Financial Research* 20, 4 (1997), 435–458.
[16] Peter L. Hammer. 1965. Some network flow problems solved with psuedo-Boolean programming. *Operations Research* 13 (1965), 388–399.
[17] Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2016. A joint many-task model: Growing a neural network for multiple NLP tasks. *arXiv preprint arXiv:1611.01587* (2016).
[18] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.
[19] Kewei Hou. 2007. Industry information diffusion and the lead-lag effect in stock returns. *The Review of Financial Studies* 20, 4 (2007), 1113–1138.
[20] Hiroshi Ishikawa. 2003. Exact Optimization for Markov Random Fields with Convex Priors. *PAMI* 25 (2003), 1333–1336.
[21] Hiroshi Ishikawa. 2009. Higher-Order Clique Reduction in Binary Graph Cut. In *CVPR*.
[22] Michael C Jensen. 1967. Random walks: reality or mythâĂŤcomment. *Financial Analysts Journal* 23, 6 (1967), 77–85.
[23] Charles D Kirkpatrick II and Julie A Dahlquist. 2010. *Technical analysis: the complete resource for financial market technicians*. FT press.
[24] Pushmeet Kohli and M. Pawan Kumar. 2010. Energy Minimization for Linear Envelope MRFs. In *CVPR*.
[25] Pushmeet Kohli, M. Pawan Kumar, and Philip H. S. Torr. 2007. P3 & Beyond: Solving Energies with Higher Order Cliques. In *CVPR*.
[26] Pushmeet Kohli, Lubor Ladicky, and Philip H. S. Torr. 2008. *Graph Cuts for Minimizing Higher Order Potentials*. Technical Report. Microsoft Research.
[27] Pushmeet Kohli, Philip HS Torr, et al. 2009. Robust higher order potentials for enforcing label consistency. *IJCV* 82, 3 (2009), 302–324.
[28] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360* (2016).
[29] Andrew W Lo and A Craig MacKinlay. 1990. When are contrarian profits due to stock market overreaction? *The review of financial studies* 3, 2 (1990), 175–205.
[30] Burton G Malkiel and Eugene F Fama. 1970. Efficient capital markets: A review of theory and empirical work. *The journal of Finance* 25, 2 (1970), 383–417.
[31] Grant McQueen, Michael Pinegar, and Steven Thorley. 1996. Delayed reaction to good news and the cross-autocorrelation of portfolio returns. *The Journal of Finance* 51, 3 (1996), 889–919.
[32] Timothy S Mech. 1993. Portfolio return autocorrelation. *Journal of Financial Economics* 34, 3 (1993), 307–344.
[33] Mark Mitchell and Todd Pulvino. 2001. Characteristics of risk and return in risk arbitrage. *the Journal of Finance* 56, 6 (2001), 2135–2175.
[34] Sebastian Nowozin and Christoph H. Lampert. 2011. Structured Learning and Prediction in Computer Vision. *Foundations and Trends in Computer Graphics and Vision* 6, 3–4 (2011), 185–365.
[35] Cheol-Ho Park and Scott H Irwin. 2007. What do we know about the profitability of technical analysis? *Journal of Economic Surveys* 21, 4 (2007), 786–826.
[36] Yao Qin, Dongjin Song, Haifeng Cheng, Wei Cheng, Guofei Jiang, and Garrison Cottrell. 2017. A Dual-Stage Attention-Based Recurrent Neural Network for Time Series Prediction. *arXiv preprint arXiv:1704.02971* (2017).
[37] Carsten Rother, Pushmeet Kohli, Wei Feng, and Jiaya Jia. 2009. Minimizing Sparse Higher Order Energy Functions of Discrete Variables. In *CVPR*.
[38] Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098* (2017).
[39] Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Vol. 2. 231–235.
[40] SL Sun et al. 2016. A Decision Tree Model for Meta-Investment Strategy of Stock Based on Sector Rotating. *Fuzzy Systems and Data MiningII: Proceedings of FSDM 2016* 293 (2016), 194.
[41] Matrin Szummer, Pushmeet Kohli, and Derek Hoiem. 2008. Learning CRFs using Graph-Cuts. In *Proc. of the European Conference on Computer Vision (ECCV)*.

[42] Ben Taskar, Vasco Chatalbashev, Daphne Koller, and Carlos Guestrin. 2005. Learning Structured Prediction Models: A Large Margin Approach. In *ICML*.

[43] Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. 2005. Large margin methods for structured and interdependent output variables. In *Journal of Machine Learning Research*. 1453–1484.

[44] Peerajak Witoonchart and Prabhas Chongstitvatana. 2017. Application of structured support vector machine backpropagation to a convolutional neural network for human pose estimation. *Neural Networks* 92 (2017), 39–46.

[45] Fang Yan. 2012. *The Research of Information Integration Under Chinese Stock Market Impact*. Ph.D. Dissertation. Tianjin University.

[46] Jiaxuan You, Rex Ying, Xiang Ren, William Hamilton, and Jure Leskovec. 2018. Graphrnn: Generating realistic graphs with deep auto-regressive models. In *International Conference on Machine Learning*. 5694–5703.

[47] Chun-Nam John Yu and Thorsten Joachims. 2009. Learning structural SVMs with latent variables. In *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 1169–1176.

[48] Alan L Yuille, Anand Rangarajan, and AL Yuille. 2002. The concave-convex procedure (CCCP). *Advances in neural information processing systems* 2 (2002), 1033–1040.

A TRAINING DETAILS

A.1 Initialization of lower linear envelope

We assume that the more evenly distributed of $W_c(Y_c)$ where $c \in C$ on x axis, the more rich representation (number of linear functions) the energy function should have. In order to initialize $\boldsymbol{\theta}$, we first determine the x-coordinate of sampled points sp. Then we sample its y-coordinate from a uniform distribution $\mathcal{U}(\text{upbound}, \text{upbound}-0.5)$ to add some randomness in our initialization as well as maintain concavity. Linear parameters a_k and b_k are later calculated using those sampled points sp_k and sp_{k-1}. At last we encode $\{a_k, b_k\}_{k=1}^K$ into $\boldsymbol{\theta}$ using equation (12). This algorithm is summarized in Algorithm 1.

Algorithm 1 Empirical initialization algorithm for $\boldsymbol{\theta}$

```
1: $gap = \frac{1}{K}$, $a_1 = \mathcal{U}(0, 1e6)$, $b_1 = 0$, $sp_1 = (0,0)$, $w_0 = 0$,
   $counter = 2$
2: for each clique $c \in C$ do
3:     Compute weighted clique value $w_c = W_c(y_C)$
4:     if $w_c - w_{c-1} > gap$ then
5:         $upbound = a_{counter} w_c + b_{counter}$
           $sp_{counter} = (w_c, \mathcal{U}(upbound - 0.5, upbound))$
           Calculate $a_{counter}$ and $b_{counter}$ using $sp_{counter-1}$ and
           $sp_{counter}$
           $counter = counter + 1$
6:     end if
7: end for
8: If $counter < K$, remaining $a$s and $b$s are all set to be $a_{counter}$
   and $b_{counter}$
9: Calculate $\boldsymbol{\theta}$ using $\{a_k, b_k\}_{k=1}^K$
```

A.2 Multi-task training

To improve accuracy and reduce over-fitting, we add a drop out layer between input layer and LSTM layer with a ratio of 0.2. We also clip and normalize gradients during back-propagation stage with a maximum norm of 5.0 to prevent gradient exploding issue. As pointed out by Lample et al. [28], the question of "when should the training schedule switch from one task to another task?" or "should each task be weighted equally?" remains open. In our implementation, we follow the proportional sampling approach described by Søgaard and Goldberg [39]. After a backward pass completed, we randomly sample a new task as well as its batch data as the next task to be trained. In practice, we use a proportion of $[0.25, 0.25, 0.5]$ for three tasks respectively. This mechanism helps multi-task model to avoid *Catastrophic Forgetting* phenomenon which means lower level model forgets learned knowledge during higher level model back-propagation pass.

Even though we propose an end-to-end training algorithm for MMPL and MRFs in section 3.1, MRFs inference stage is still too slow to be trained jointly with MMPL. To overcome this difficulty, we implement a two stages training procedure. We first add a *softmax* layer on top of $\text{DARNN}_{\text{class}}$ and train MMPL separately from MRFs. We use *Negative Log-likelihood* as the loss function. At the second stage, after MMPL converge, we remove the *softmax* layer and retrain it together with MRFs. One issue we must mention is that,

Algorithm 2 Learning lower linear envelope MRFs with latent variables.

```
1: Set $MaxIter = 100$
2: input training set $\{\boldsymbol{y}_i\}_{i=1}^n$, regularization constant $C > 0$, and tolerance $\epsilon \geq 0$
3: Initialize $\boldsymbol{\theta}$ using Algorithm 1
4: repeat
5:     CCCP Outer Loop
6:     Set $iter = 0$
7:     for each training example, $i = 1, \ldots, n$ do
8:         compute $z_i^* = \text{argmax}_{\mathrm{z} \in \mathcal{Z}}\, \theta \cdot \psi(\mathrm{y}_i, \mathrm{z})$
9:     end for
10:    initialize active constraints set $C_i = \{\}$ for all $i$
11:    repeat
12:        CCCP Inner Loop
13:        solve the quadratic programming problem in equation 20 with respect to active constraints set $C_i$ for all $i$ and concavity constraints $A\boldsymbol{\theta} \geq \epsilon$ to get $\hat{\boldsymbol{\theta}}$ and $\hat{\boldsymbol{\xi}}$
14:        for each training example, $i = 1, \ldots, n$ do
15:            compute $\hat{\boldsymbol{y}}_i, \hat{z}_i = \text{argmin}_{\boldsymbol{y}}\, E(\boldsymbol{y}, z; \hat{\boldsymbol{\theta}}) - \Delta(\boldsymbol{y}, z, \boldsymbol{y}_i)$
16:            if $\hat{\xi}_i + \epsilon < \Delta(\hat{\boldsymbol{y}}_i, \hat{z}_i, \boldsymbol{y}_i) - E(\hat{\boldsymbol{y}}_i, \hat{z}_i; \hat{\boldsymbol{\theta}}) + E(\boldsymbol{y}_i, z_i^*; \hat{\boldsymbol{\theta}})$ then
17:                $C_i \leftarrow C_i \cup \{\boldsymbol{y}_i^\star\}$
18:            end if
19:        end for
20:    until no more violated constraints
21:    return parameters $\hat{\boldsymbol{\theta}}$
22:    Set $iter = iter + 1$
23: until $iter \geq MaxIter$
24: return parameters $\hat{\boldsymbol{\theta}}$
```

even though we use binary MRFs which can only predict positive / negative price movement, we find there is a significant amount of time when stock price remains no change. We find it benefits the performance a lot if we treat the classification as a three classes problem rather than a binary classification problem during the first stage. Therefore, at the first stage, the *softmax* layer will output probability for three labels: *negative movement*, *no changes* and *positive movement*. Since binary MRFs still needs a two dimension input as part of unary energy function, after the *softmax* layer is removed, we add an additional linear mapping layer between logits of MMPL and MRFs at the second stage.

A.3 End-to-end NN-MRFs training

With converged MMPL and MRFs at hand, now we can go forward to train them in an end-to-end manner. We only include pairwise energy function through section 3.2 and section 4 to show a general application of our proposed algorithm. In the case of Chinese stock market, to our best knowledge there is no public available definition of pairwise relationship between stocks. Therefore, in our implementation we only use unary and higher order energy function. Each stock is then treated as a node in MRFs and each stocks group which has lead-lag relationships is treated as a maximum clique in MRFs. One benefit of MRFs clique is that we can embed domain expert knowledge about industry classification as maximum cliques into our model. We choose to use Tonghuashun industry classification [1] in our model. One subtle but crucial detail about modeling lead-lag effect lies in equation (4). Recall that $W_c(\boldsymbol{y}_c) = \sum_{i \in c} w_i y_i$ with $w_i^c \geq 0$ and $\sum_{i \in c} w_i^c = 1$ which are weights for stocks in each clique. Therefore, leading stocks should have a higher weights while lagging stocks should have lower weights. In our implementation, we use constituents' weight defined in CSI200, CSI500 and CSI300 as their weights in equation (4) and normalize them to ensure the summation equals 1.