End-to-end Multi-tasks Recurrent Neural Network and Higher-order Markov Random Field Learning For Stock Price Prediction

Chang Li
UBTECH Sydney AI Centre,
SCS, University of Sydney
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

This paper provides a sample of a LaTeX document for final submission to Sigkdd Explorations, the official newsletter of ACM Sigkdd. This is a modified version of the ACM Proceedings sample file.

The developers have tried to include every imaginable sort of "bells and whistles", such as a subtitle, footnotes on title, subtitle and authors, as well as in the text, and every optional component (e.g. Acknowledgements, Additional Authors, Appendices), not to mention examples of equations, theorems, tables and figures.

To make best use of this sample document, run it through LaTeX and BibTeX, and compare this source code with the printed output produced by the dvi file.

1. INTRODUCTION

It has been a long known fact that stocks'price movement are highly correlated to other stocks rather than independent [28, 31] and change in a non-synchronous way [7, 28]. This correlated yet non-synchronous price movement phenomenon is sometimes described as lead-lag relationship [18] within a group of stocks. Different speed of information diffusion has been believed to be the main reason of lead-lag relationship [2, 28, 30]. When an information hits the market, some stocks'price tend to react faster than others do. Therefore, identification of those leading stocks and their lead-lag relationship to other lagging stocks will provide strong evidence in predicting the latter's price movement direction.

There are three steps involved in utilizing lead-lag relationship: 1. Analyzing which group of stocks (industry, supply chain etc.) will be affected by newly arrived information and corresponding leading stocks; 2. Identifying lagging stocks in this group and modeling their relationships; 3. Predicting price movement of each stock by jointly considering group knowledge and individual stock's price movement at current stage.

Step 1. is extremely difficult for an algorithm to automate using nowadays technologies. It requires an expert level understanding of the finance system and dynamics behind market news and stock price. Also there is lacking of training dataset. However, according to the well-known efficient market hypothesis [29], stock price reflects all available market information. Therefore, informational stock price changes can be used as an proximity to market news arrival. The complexity of step 1 becomes pattern recognition of informational price changes in individual stock.

Economists have spent decades trying to use patterns hidden inside historical trading price and volume to predict future price movement [11, 21]. Those models are called technical analysis [22]. However, most of those models have been proven stopped generating profitable signals since the early 1990s [33]. On the other hand, since trading strategies based on technical analysis rules are public available and easy to replicate, informed institutional traders are motivated to manipulate market price and trap retail (individual) traders following their manipulated price to gain excess profit [38]. Problem of pattern recognition in stock price series still remains open.

To overcome those difficulties, we propose an end-to-end hierarchical multi-task [8] approach to extract informational changes from raw market price without using any of handcrafted features such as technical analysis indicators. Good performance on price prediction relies on rich representations and a multi-task framework can leverage complementary aspects from diverse tasks [37]. Given raw market price data only contains 6 features (opening price, low price, high price, closing price, volume and amount) at each time interval, we leverage a hierarchical multi-task network to extract features on different tasks first and then concatenate those complementary feature vectors to predict final results.

To model lead-lag relationships (step 2. and 3.), we propose a binary Markov Random Fields (MRFs) with weighted lower linear envelope as higher order energy function [13, 14, 24, 32]. In our implementation, we treat each stock as a node in MRFs and each stocks group which has lead-lag relationships as a maximum clique in MRFs. We use pre-defined industry classification list [1] as prior domain knowledge of each stock's belonging maximum clique. 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 networks are used as unary feature in MRFs. By minimizing energy function contains both unary and higher order features, we are able to predict each stock's future price movement by jointly considering individual market price trending together with lead-lag relationships it has.

We verify our methodology on Chinese stock market because it is generally considered less mature than other developed countries'thus potentially to be more profitable [3].

Even though immature, Yan [42] found that the average duration of information arrival-conduction-integration-release process is 4.04 minutes. Therefore, a minute-level frequency model is mandatory to leverage from lead-lag relationship. To our best knowledge, our model is the first one demonstrating lead-lag relationship at minute-level frequency on Chinese stock market.

Our main contributions are the following: (1) We propose a hierarchical multi-task architecture for learning stock price patterns without any hand-crafted features. (2) We propose the first model that encode lead-lag relationship among stocks using higher order MRFs to the best of our knowledge. (3) We design an algorithm to learn weighted lower linear envelope with latent variables as higher order energy function under latent structural SVMs framework. Adding latent variable into higher order function enables our model to learn much richer representation than previous study [14].

2. RELATED WORKS AND BACKGROUND

Lead-lag Relationship Lead-lag relationship has been found to be a long existence phenomenon in stock market. Many reasons can lead to this such as diffusion of information, sector (industry) rotation, investment style rotation, event driven trading and nonsynchronous trading [9, 10, 15, 28]. Generally it is believed that lead-lag relationship is more prevalent for firms from the same industry [18]. This assumption give rise to our setting that we use pre-defined industry classification list [1] as prior domain knowledge of each stock's belonging maximum clique.

Multi-task Learning Caruana [8] shows that inductive knowledge learned from multiple tasks can transfer between tasks and help improving generalization on all tasks. Many works in Natural Language Processing (NLP) area take advantage of multi-task framework and achieves state-of-the-art performance while using simple models for each of these tasks [17, 37]. However, as pointed out by many researchers [8, 36], there are lacking of theories on choosing a diverse set of tasks and hierarchical architecture of chosen tasks. Recent works [17, 37] apply the intuition that the complexity of task should be increasing along with hierarchical level. We follow this intuition in our implementation. Because technical analysis indicators can be categorized into four categories (trend, momentum, volatility and volume) [22] and volume is included in market price data, we propose an architecture that using trend and volatility tasks as our lower level tasks and price movement classification (upward or downward) as higher level task. Other selection of tasks and hierarchical designations remain open for further research.

2.1 Attention Mechanism

An interesting problem in time series modeling is automatically selecting informational features. The Nonlinear autoregressive exogenous (NARX) model, for example, is a model that predict future value of a time series by using itself previous value together with other features temporal information. When the potential informational features pool is large, it is difficult for the model to select relevant features at each time step and encode long-range dependencies. Qin et al. [34] proposed a dual-stage attention mechanism to demonstrate these issues. Their main contribution is at the first stage. An attention mechanism is applied between the input temporal feature matrix and the first hidden layer. This input attention layer calculates attention weights for each feature at each time step t by combining the information of each feature vector and the hidden state from first layer at time step $t-1$. Then the attention weights are applied to feature matrix to get a weighted input feature matrix at time step t. The second stage is a conventional attention layer which attends to all hidden states in the first layer. Their result outperforms previous state-of-the-art models on NASDAQ 100 Stock dataset. By employing a dual-stage attention mechanism, their model can not only solve long-range dependencies problem but also select informational features adaptively at each time step.

2.2 Markov Random Fields

Markov Random Fields are also known as *undirected graphical model* that can be seen as a regularized joint log-probability distribution of arbitrary non-negative functions over a set of maximal cliques of the graph [4]. Let C_j denotes a maximal clique in one graph and $\boldsymbol{y}_C$ denotes the set of variables in that clique, where $\boldsymbol{y}_c = \{y_i : i \in C_j\}$ are discrete random variables and $C_j \subseteq \{1, ..., n\}$ is a subset of variable indices. Then the joint distribution can be written as:

$$p(\boldsymbol{y}) = \frac{1}{Z} exp(-\sum_{C} E_C(\boldsymbol{y}_C)) \tag{1}$$

where E_C is called *energy function* which can be arbitrary function. Therefore, to infer labels which best explains input data set, we can solve the *energy minimization* problem, which is also known as *inference*:

$$\boldsymbol{y}^* = \underset{\boldsymbol{y}}{\operatorname{argmax}}\, p(\boldsymbol{y}) = \underset{\boldsymbol{y}}{\operatorname{argmin}}(-\sum_{C} E_C(\boldsymbol{y}_C)) \tag{2}$$

To optimize the performance we can also consider a weighted version of energy functions. In order to do this we can decompose energy functions over nodes $\mathcal{N}$, edges $\mathcal{E}$ and higher order cliques $\mathcal{C}$ [39] then add weights on them accordingly. Let $\boldsymbol{w}$ be the vector of parameters and ϕ be arbitrary feature function, then the energy can be decomposed as a set of linear combinations of weights and feature vectors:

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

where U denotes *unary* terms, P denotes *pairwise* terms and H denotes *higher order* terms (when $|C| > 2$ namely each clique contains more than two variables).

A weight vector $\boldsymbol{w}$ is more preferable if it gives the ground-truth assignments $\boldsymbol{y}_t$ less than or equal to energy value than any other assignments y:

$$E(y_t, w) \leq E(y, w)\ ,\ \forall y \neq y_t,\ y \in \mathcal{Y} \tag{4}$$

Thus the goal of *learning* MRFs is to learn the parameter vector $\boldsymbol{w}^*$ which returns the lowest energy value for the ground-truth labels y_t relative to any other assignments y [39]:

$$\boldsymbol{w}^* = argmax_{\boldsymbol{w}}(E(y_t, w) - E(y, w))\ ,\ \forall y \neq y_t,\ y \in \mathcal{Y} \tag{5}$$

Up to now we have outlined the framework of using MRFs: defining *energy functions*(eq (1)), solving *inference* problem

(MAP or energy minimization) (eq (2)) and *learning* parameters (eq (5)). As for energy functions, our work focus 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 consistency constraints over large subsets of pixels in an image [24, 32]. We follow Gould [14] to construct a graph-cut algorithm to solve exact inference problem and propose our novel learning algorithms under latent structural SVM in section 4.1.

Kohli et al. [26] proposed a method to represent a class of higher order potentials with lower (upper) linear envelope potentials. By introducing auxiliary variables [23], 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 routine, Gould [14] extended their method to a weighted lower linear envelope with arbitrary many terms which can be solved with an efficient algorithm. They showed the energy function with auxiliary variables is submodular by transforming it into a quadratic pseudo-Boolean form [5] and how *graph-cuts* [6, 12, 16] like algorithm can be applied to do exact *inference*.

Gould [14] solved *learning* problem of lower linear envelope under the max margin framework [41]. In their work they pointed out the potential relationship between their auxiliary representation and latent SVM [43]. Our work is closely based on their research. We continue to use the higher order energy function and inference algorithm developed in their previous work [13] and extend their max margin learning algorithm to include latent variables. The learning algorithm we use is an extension of max margin framework which is known as "latent structural SVM" [43].

2.3 Latent Structural SVMs

The max-margin framework [40, 41] is a principled approach to learn the weights of pairwise MRFs. Szummer et al. [39] adapted this framework to optimize parameters of pairwise MRFs inferred by graph-cuts method. In our previous work Gould [14] extended this framework with additional linear constraints which enforces concavity on weights thus can be used for learning lower linear envelope potentials.

In this section we introduce *latent structural SVM* [43] which extends the max-margin framework by encoding latent information in feature vector. In section 4.1 we will show how this framework can be adapted to learn parameters for higher order energy function with latent variables.

Given an a linear combination of features vector $\phi(\boldsymbol{x}, \boldsymbol{y}) \in \mathbb{R}^m$ and weights $\boldsymbol{\theta} \in \mathbb{R}^m$, and a set of n training examples $\{\boldsymbol{y}_i\}_{i=1}^n$ max-margin framework can be used to solve optimized solution $\boldsymbol{\theta}^*$. To include unobserved information in the model, Yu[43] extended the joint feature function[41] $\phi(\mathbf{x}, \mathbf{y})$ with a latent variable $\mathbf{h} \in \mathcal{H}$ to $\phi(\mathbf{x}, \mathbf{y}, \mathbf{h})$. So the inference problem becomes

$$f_\theta(x) = \underset{(\mathbf{y}\times\mathbf{h})\in\mathcal{Y}\times\mathcal{H}}{\operatorname{argmax}} \theta \cdot \phi(\mathbf{x}, \mathbf{y}, \mathbf{h}) \tag{6}$$

Accordingly, the loss function can be extended as

$$\Delta((\mathbf{y}_i, \mathbf{h}_i^*(\theta)), (\hat{\mathbf{y}}_i(\theta), \hat{\mathbf{h}}_i(\theta)))$$

where

$$(\hat{\mathbf{y}}_i(\theta), \hat{\mathbf{h}}_i(\theta)) = \underset{(\mathbf{y}\times\mathbf{h})\in\mathcal{Y}\times\mathcal{H}}{\operatorname{argmax}} \theta \cdot \phi(\mathbf{x}_i, \mathbf{y_i}, \mathbf{h}) \tag{7}$$

$$\mathbf{h}_i^*(\theta) = \underset{\mathbf{h}\in\mathcal{H}}{\operatorname{argmax}} \theta \cdot \phi(\mathbf{x}_i, \mathbf{y}_i, \mathbf{h}) \tag{8}$$

Yu and Joachims [43] argued that in real world applications hidden variables are usually intermediate results and are not required as an output[43]. Therefore, the loss function can only focus on the inferenced hidden variables $\hat{\mathbf{h}}_i(\theta)$. Thus the upper bound used in standard structural SVMs[41] can be extended to:

$$\begin{aligned}
&\Delta((\mathbf{y}_i, \mathbf{h}_i^*(\theta)), (\hat{\mathbf{y}}_i(\theta), \hat{\mathbf{h}}_i(\theta))) \\
\leq &\Big(\max_{(\hat{\mathbf{y}}\times\hat{\mathbf{h}})\in\mathcal{Y}\times\mathcal{H}} [\theta \cdot \Psi(\mathbf{x}_i, \hat{\mathbf{y}}, \hat{\mathbf{h}}) + \Delta(\mathbf{y}_i, \hat{\mathbf{y}}, \hat{\mathbf{h}})] \Big) \\
&- \max_{\mathbf{h}\in\mathcal{H}} \theta \cdot \Psi(\mathbf{x}_i, \mathbf{y}_i, \mathbf{h})
\end{aligned} \tag{9}$$

Hence the optimization problem for Structural SVMs with latent variables becomes

$$\begin{aligned}
\min_\theta \Bigg(&\frac{1}{2}\|\theta\|^2 \\
&+ C \sum_{i=1}^n \big(\max_{(\hat{\mathbf{y}}\times\hat{\mathbf{h}})\in\mathcal{Y}\times\mathcal{H}} [\theta \cdot \Psi(\mathbf{x}_i, \hat{\mathbf{y}}, \hat{\mathbf{h}}) + \Delta(\mathbf{y}_i, \hat{\mathbf{y}}, \hat{\mathbf{h}})] \big) \Bigg) \\
&- C \sum_{i=1}^n \big(\max_{\mathbf{h}\in\mathcal{H}} \theta \cdot \Psi(\mathbf{x}_i, \mathbf{y}_i, \mathbf{h}) \big)
\end{aligned} \tag{10}$$

which is a difference of two convex functions. Problem of this formulation can be solved using the Concave-Convex Procedure (CCCP)[44] which is guaranteed to converge to a local minimum. Yu and Joachims [43] proposed a two stages algorithm. In the first step the latent variable $\boldsymbol{h}_i^*$ which best explains training pair $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ is found by solving equation (8). This step is also called the "latent variable completion" problem. In the second step $\boldsymbol{h}_i^*$ is used as completely observed to substitute $\boldsymbol{h}$ in equation (10). Therefore, solving equation (10) is equivalent to solve the standard structural SVM problem.

The Latent Structural SVM can not take inputs data directly such as the SVM. To use Latent Structural SVM to optimize higher order MRFs with latent variables, the MRF inference algorithm (eq (7)), as well as the MRF feature function, loss function, and latent variable completion problem (eq (8)) need to be implemented first. In order to adapt the energy function to max margin framework, Gould [13] approximated the energy function using equally spaced break-points thus removed those auxiliary variables. In this thesis 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. We will present this in detail in section 4.1.

3. METHODS

In this section, we introduce our multi-tasks DNNs-MRFs architecture from lower part to the top part. The whole

model is constructed with two parts. The first part is a "Holistic Market Price Learner", which contains three DARNN modules. The goal of the first part is to extract informational representations of raw market price end-to-end without any input of hand-crafted features and technical indicators from stock market. The second part is called "Section Rotation Predictor" which is a binary Markov Random Fields model contains higher order energy functions. Those higher order functions are applied to financial experts defined sector lists (used as maximum cliques). The goal of this part is to utilizing unary features learned by DNNs while utilizing domain knowledge from financial experts by modeling higher order consistency of stocks belong to the same sector. We summarized the whole architecture in figure 1.

3.1 Holistic Market Price Learner

The "Holistic Market Price Learner(HMPL)" contains two levels, three modules of DARNNs[34]. Recall the goal of HMPL is to replace hand-crafted features and financial technical indicators with representations learned end-to-end by neural networks. To achieve this, HMPL needs to be trained on a set of diverse and complementary tasks in order to encode holistic market price information. Bottom level contains two separate DARNN modules. They are supervised by low-level tasks such as regression to future price and volatility using raw market price data. At top level, it is supervised by high-level task that learns to use representations extracted by two bottom modules as well as raw market price data to predict ascending / descending price movement of stocks. Logits of the last layer are passed to Sector Rotation Predictor described in section 3.2 as unary features.

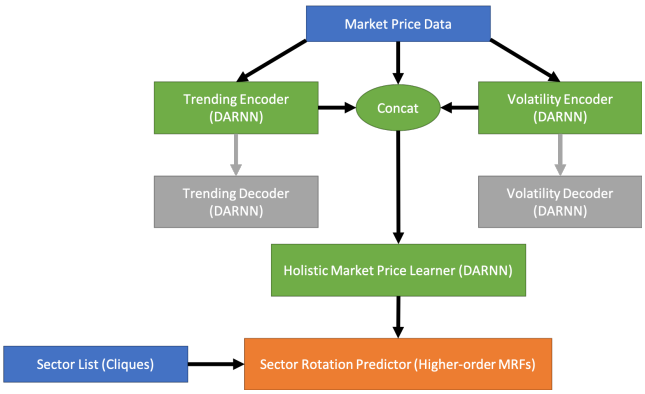

Figure 1: Multi-tasks DNN MRFs Architecture

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 driving series.

For both DARNN modules at the bottom 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 $\boldsymbol{y} = (y_1, y_2, \ldots, y_T) \in \mathbb{R}^T$. The task of those DARNNs are 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 $\boldsymbol{y}_{\text{trending}}$ of Trending DARNN is closing price. The target series $\boldsymbol{y}_{\text{volatility}}$ of Volatility DARNN is the standard deviation of closing price over T time-steps. We use Mean Squared Error (MSE) as loss function to train those two modules separately.

To train the top level module, which is a classification DARNN, we concatenate context vectors $\boldsymbol{c}_t$ from each of bottom level module's second stage encoder and raw market price matrix as input matrix. 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. Logits (the output of the DARNN before going through the softmax) of the final DARNN are passed to "Sector Rotation Predictor" as unary features.

Since outputs of HMPL are only used as unary features in MRFs' energy functions, our back-propagation rules can be defined by taking derivative of equation (3):

$$\frac{\partial L}{\partial \boldsymbol{w}^U} = \phi^U(y) - \phi^U(y^*) \tag{11}$$

where y is the ground-truth label and y^* is inferenced label. And

$$\frac{\partial L}{\partial \phi^U} = \boldsymbol{w}^U \tag{12}$$

3.2 Sector Rotation Predictor

We begin with a brief review of our choices of unary, pairwise and higher-order potential functions. We then show how to perform exact inference in models with these potentials. In section 4.1 we will discuss learning the parameters under the latent structural SVM framework and also how to back-prop gradients to neural networks.

3.2.1 *Higer Order Energy: The Lower Linear Envelope Function*

From section 2.2 we have already introduced that an *energy function* may contain *unary*, *pairwise* and *higher-order* potentials (see equation (3)). In this section we mainly focus on one class of higher-order potentials ϕ^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, 24, 32].

Let $\mathcal{C}$ 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 random variables in the clique C_j, a weighted lower linear envelope potential [14] over $\boldsymbol{y}_c$ is defined as the minimum over a set of K linear functions as:

$$\psi_c^H(\boldsymbol{y}_c) = \min_{k=1,\ldots,K} \{a_k W_c(\boldsymbol{y}_c) + b_k\}. \tag{13}$$

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 [14] with three linear functions in Figure 2.

Inference on energy function contains lower linear potentials

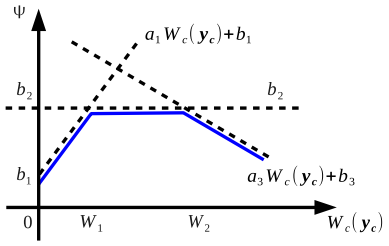

Figure 2: Example lower linear envelope $\psi_c^H(\boldsymbol{y}_c)$ (shown solid) with three terms (dashed). When $W_c(\boldsymbol{y}_c) \leq W_1$ the first linear function is active, when $W_1 < W_c(\boldsymbol{y}_c) \leq W_2$ the second linear function is active, otherwise the third linear function is active.

is the same as the standard equation (3) and is given by:

$$\boldsymbol{y}^* = \operatorname{argmin} E(\boldsymbol{y}) \tag{14}$$

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:

$$0 < \frac{b_k - b_{k-1}}{a_{k-1} - a_k} < \frac{b_{k+1} - b_k}{a_k - a_{k+1}} < 1. \tag{15}$$

Another important property of equation (14) is shift invariant [14] (vertically). We write $\widetilde{\psi}_c^H(\boldsymbol{y}_c)$ by shift equation (13) vertically with an abitrary amount $b^{const} \in R$

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

Then we have

$$\operatorname*{argmin}_{\boldsymbol{y}_c} \psi_c^H(\boldsymbol{y}_c) = \operatorname*{argmin}_{\boldsymbol{y}_c} \widetilde{\psi}_c^H(\boldsymbol{y}_c). \tag{16}$$

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

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 [19, 20, 35] are able to be minimized exactly using *graph-cuts* like algorithms [12, 16] when they satisfy submodularity condition [5]. Kohli et al. [25] 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 (14) of energy function containing lower linear envelope potentials.

Following the approach of Kohli and Kumar [23], Gould [13, 14] transformed the weighted lower linear envelope potential (13) into a quadratic pseudo-Boolean function by introducing $K-1$ auxiliary variables $\boldsymbol{z} = (z_1, \ldots, z_{K-1})$ with $z_k \in \{0, 1\}$:

$$\begin{aligned} E^c(\boldsymbol{y}_c, \boldsymbol{z}) &= a_1 W_c(\boldsymbol{y}_c) + b_1 \\ &\quad + \sum_{k=1}^{K-1} z_k \left((a_{k+1} - a_k) W_c(\boldsymbol{y}_c) + b_{k+1} - b_k \right) \end{aligned} \tag{17}$$

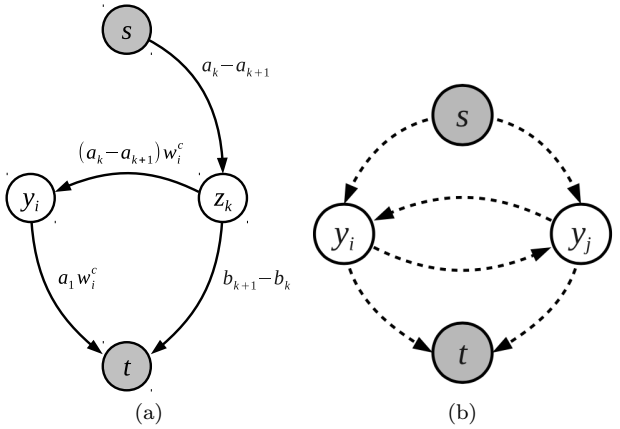

Figure 3: *st*-graph construction [14] for equation (19), unary and pairwise terms. Every cut corresponds to an assignment to the random variables, where variables associated with nodes in the $\mathcal{S}$ set take the value one, and those associated with nodes in the $\mathcal{T}$ set take the value zero. With slight abuse of notation, we use the variables to denote nodes in our graph.

for a single clique $c \in \mathcal{C}$. Under this formulation, minimizing the pseudo-Boolean function over $\boldsymbol{z}$ is equivalent to selecting (one of) the active functions(s) from equation (13). Another important property of optimized $\boldsymbol{z}$ under this formulation is that it automatically satisfies the constraint [14]:

$$z_{k+1} \leq z_k \tag{18}$$

this property give rise to further development of parameter vector (22) and feature vector (23) which are used in latent structural SVM.

In order to construct the *st-min-cut* graph, we rewrote equation (17) into *posiform* [5]:

$$\begin{aligned} E^c(\boldsymbol{y}_c, \boldsymbol{z}) &= b_1 - (a_1 - a_K) + \sum_{i \in c} a_1 w_i^c y_i \\ &\quad + \sum_{k=1}^{K-1} (b_{k+1} - b_k) z_k + \sum_{k=1}^{K-1} (a_k - a_{k+1}) \bar{z}_k \\ &\quad + \sum_{k=1}^{K-1} \sum_{i \in c} (a_k - a_{k+1}) w_i^c \bar{y}_i z_k \end{aligned} \tag{19}$$

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 $b_k < b_{k+1}$). After proving *submodularity* of the energy function (19), Gould [14] constructed the *st-min-cut* graph based on equation (19).

The construction is explained in Figure 3. Figure (a) denotes construction for equation (19). For each lower linear envelope potential edges are added as follows: for each $i \in c$, add an edge from y_i to t with weight $a_1 w_i^c$; for each $i \in c$ and $k = 1, \ldots, K-1$, add an edge from z_k to y_i with weight $(a_k - a_{k+1}) w_i^c$; and for $k = 1, \ldots, K-1$, add an edge from s to z_k with weight $a_k - a_{k+1}$ and edge from z_k to t with weight $b_{k+1} - b_k$. Figure (b) denotes construction for unary and pairwise terms (see [27]). For unary edges (4 edges on

both sides), weights on each edge are corresponding to values in input unary terms accordingly. For pairwise edges (2 edges in the middle), both edges share the same weight which equals to the input pairwise term.

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 (17) 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 (see equation (6)) requires that the energy function be expressed as 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 (17) by introducing auxiliary variables. Now we show function (17) itself is an inner product of parameter vector and feature vector with latent information. We first noticed that the function can be expanded as a summation of $2K-1$ terms:

$$\begin{aligned} E^c(y_c, z) =& a_1 W_c(y_c) + b_1 \\ &+ \sum_{k=1}^{K-1} z_k((a_{k+1} - a_k)W_c(y_c) + b_{k+1} - b_k) \\ =& a_1 W_c(y_c) + \sum_{k=1}^{K-1} (a_{k+1} - a_k) z_k W_c(y_c) \\ &+ \sum_{k=1}^{K-1} (b_{k+1} - b_k) z_k \end{aligned} \tag{20}$$

Here we use the fact of equation (16) and let $b_1 = 0$. Now we can reparameterize the energy function as

$$E^c(\boldsymbol{y}_c, \boldsymbol{z}; \boldsymbol{\theta}) = \boldsymbol{\theta}^T \phi(\boldsymbol{y}_c, \boldsymbol{z}) \tag{21}$$

where:

$$\theta_k = \begin{cases} a_1 & \text{for } k = 1 \\ a_k - a_{k-1} & \text{for } 1 < k \leq K \\ b_{k+1-K} - b_{k-K} & \text{for } K < k \leq 2K-1 \end{cases} \tag{22}$$

$$\phi_k = \begin{cases} W_c(\boldsymbol{y}_c) & \text{for } k = 1 \\ W_c(\boldsymbol{y}_c)\boldsymbol{z}_k & \text{for } 1 < k \leq K \\ \boldsymbol{z}_k & \text{for } K < k \leq 2K-1 \end{cases} \tag{23}$$

Under this formulation, inference problems (7) and (8) introduced in section 2.3 can be written as:

$$(\hat{\mathbf{y}}_k(\boldsymbol{\theta}), \hat{\mathbf{z}}_k(\theta)) = \underset{(\mathbf{y}\times\mathbf{z})\in\mathcal{Y}\times\mathcal{Z}}{\operatorname{argmin}} \boldsymbol{\theta}^T \cdot \phi(\mathbf{y}_k, \mathbf{z}_k) \tag{24}$$

and

$$\mathbf{z}_k^*(\boldsymbol{\theta}) = \underset{\mathbf{z}\in\mathcal{Z}}{\operatorname{argmin}} \boldsymbol{\theta}^T \cdot \phi(\mathbf{y}_k, \mathbf{z}_k) \tag{25}$$

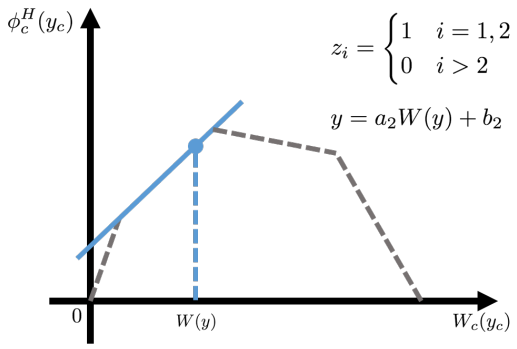

Figure 4: Example piecewise-linear concave function of $W_c(\boldsymbol{y}_c) = \sum_{i\in c} w_i^c y_i$. Assume the second linear function is active namely $\boldsymbol{z}^c = (1, 1, 0, 0)$. The result of linear combination of parameter vector and feature vector is same as quadratic psuedo-Boolean function.

There are 2 facts worth to mention. The first fact is that in our previous construction of minimum-*st*-cut graph the latent variable $\boldsymbol{z}$ is already included. Therefore, we can apply our inference algorithm directly on our 2 new formulations. However, for equation (25) 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 (20) becomes constant. So we can infer latent variable $\boldsymbol{z}$ explicitly by:

$$z_k^c = \begin{cases} 0 & \text{if } ((a_{k+1} - a_k)W_c(y_c) + b_{k+1} - b_k) \geq 0 \\ 1 & \text{otherwise.} \end{cases} \tag{26}$$

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 the following constraint:

$$A\boldsymbol{\theta} \geq 0, \quad A = \begin{bmatrix} 1 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -\mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{P} \end{bmatrix} \in \mathbb{R}^{(2K-1)\times(2K-1)} \tag{27}$$

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 ϵ on those constraints:

$$A\boldsymbol{\theta} \geq \epsilon, \quad A = \begin{bmatrix} 1 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -\mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{P} \end{bmatrix} \in \mathbb{R}^{(2K-1)\times(2K-1)} \tag{28}$$

where ϵ equals to $\mathbf{1}^{-15}$ in our experiments.

4.2 Latent Structural SVM Learning

With the inner product formulation (equation (21)) 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:

$$E_{all}(y,z) = \begin{bmatrix} \boldsymbol{\theta}^H \\ \theta^{unary} \\ \theta^{pairwise} \end{bmatrix}^T \cdot \begin{bmatrix} \phi^H \\ \phi^{unary} \\ \phi^{pairwise} \end{bmatrix} = \theta_{all}^T \cdot \phi_{all} \quad (29)$$

where $\boldsymbol{\theta}^H \in \mathbb{R}$ is the parameter vector in higher order equation (21) of size $2K-1$. θ^{unary} and $\theta^{pairwise}$ are both scalars. $\phi^{\text{unary}} = \sum_i \psi_i^U(y_i)$ and $\phi^{\text{pairwise}} = \sum_{ij} \psi_{ij}^P(y_i, y_j)$. Therefore, the size of θ_{all} is $2K+1$.

Plug equation (24) and equation (25) into object function (10), the latent structural SVM object function for our problem can be derived as a difference of two convex functions:

$$\min_{\theta} \left(\frac{1}{2}\|\theta\|^2 + C\sum_{i=1}^{n} \big(\max_{(\hat{\mathbf{y}}\times\hat{\mathbf{z}})\in\mathcal{Y}\times\mathcal{Z}} [\theta\cdot\phi(\hat{\mathbf{y}},\hat{\mathbf{z}}) + \Delta(\mathbf{y}_i,\hat{\mathbf{y}},\hat{\mathbf{z}})] \big) \right) \quad (30)$$

$$-C\sum_{i=1}^{n} \big(\max_{\mathbf{z}\in\mathcal{Z}} \theta\cdot\phi(\mathbf{y}_i,\mathbf{z}) \big)$$

As mentioned by Yu and Joachims [43] the Concave-Convex Procedure (CCCP) [44] can be used to solve the optimization problem. Our algorithm contains two stages. We first imputes the latent variables $\boldsymbol{z}$ explicitly by equation (25). Namely solving the "latent variable completion" problem [43]:

$$\boldsymbol{z}_i^* = \operatorname*{argmax}_{\mathbf{z}\in\mathcal{Z}} \theta\cdot\phi(\mathbf{y}_i,\mathbf{z}) \quad (31)$$

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 $\boldsymbol{\theta}$ is similar to solve the standard max-margin optimization problem described in [14]:

$$\min_{\theta} \left(\frac{1}{2}\|\theta\|^2 + C\sum_{i=1}^{n} \big(\max_{(\hat{\mathbf{y}}\times\hat{\mathbf{z}})\in\mathcal{Y}\times\mathcal{Z}} [\theta\cdot\phi(\hat{\mathbf{y}},\hat{\mathbf{z}}) + \Delta(\mathbf{y}_i,\hat{\mathbf{y}},\hat{\mathbf{z}})] \big) \right) \quad (32)$$

$$-C\sum_{i=1}^{n} \big(\theta\cdot\phi(\mathbf{y}_i,\mathbf{z}_i^*) \big)$$

The last problem remaining is the initialization method. Because our objective function (32) is not convex and the CCCP algorithm is only guaranteed to converge to a local minimum or saddle point[44], initialization of $\boldsymbol{\theta}$ might affect the performance of our algorithm. Since there are no theoretical solution for this problem, we only propose an empirical Algorithm 1:

We assume that the more evenly distributed of $W_c(Y_c)$ where $c \in \mathcal{C}$ 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 $\mathcal{U}(upbound, upbound-0.5)$ 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 (22).

Our optimization algorithm is summarized in Algorithm 2.

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

```
gap = 1/K, a_1 = U(0, 1e6), b_1 = 0, sp_1 = (0,0), w_0 = 0, counter = 2
for each clique c ∈ C do
    Compute weighted clique value w_c = W_c(y_C)
    if w_c − w_{c−1} > gap then
        upbound = a_counter w_c + b_counter
        sp_counter = (w_c, U(upbound − 0.5, upbound))
        Calculate a_counter and b_counter using sp_{counter−1} and sp_counter
        counter = counter + 1
    end if
end for
If counter < K, remaining as and bs are all set to be a_counter and b_counter
Calculate θ using {a_k, b_k}_{k=1}^K
```

References

Tonghuashun industry classification. `http://q.10jqka.com.cn/thshy/`. Accessed: 2019-01-15.

Swaminathan G Badrinath, Jayant R Kale, and Thomas H Noe. Of shepherds, sheep, and the cross-autocorrelations in equity returns. *The Review of Financial Studies*, 8(2): 401–430, 1995.

Hendrik Bessembinder and Kalok Chan. The profitability of technical trading rules in the asian stock markets. *Pacific-Basin Finance Journal*, 3(2-3):257–284, 1995.

Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

Endre Boros and Peter L. Hammer. Pseudo-boolean optimization. *Discrete Applied Mathematics*, 123:155–225, 2002.

Yuri Y. Boykov and Marie-Pierre Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in n-d images. In *Proc. of the International Conference on Computer Vision (ICCV)*, 2001.

Michael J Brennan, Narasimhan Jegadeesh, and Bhaskaran Swaminathan. Investment analysis and the adjustment of stock prices to common information. *The Review of Financial Studies*, 6(4):799–824, 1993.

Richard A Caruana. Multitask connectionist learning. In *In Proceedings of the 1993 Connectionist Models Summer School*. Citeseer, 1993.

Tarun Chordia and Bhaskaran Swaminathan. Trading volume and cross-autocorrelations in stock returns. *The Journal of Finance*, 55(2):913–935, 2000.

Jennifer Conrad and Gautam Kaul. Time-variation in expected returns. *Journal of business*, pages 409–425, 1988.

Eugene F Fama and Marshall E Blume. Filter rules and stock-market trading. *The Journal of Business*, 39(1):226–241, 1966.

Daniel Freedman and Petros Drineas. Energy minimization via graph cuts: Settling what is possible. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

Stephen Gould. Max-margin learning for lower linear envelope potentials in binary Markov random fields. In *Proc. of the International Conference on Machine Learning (ICML)*, 2011.

Stephen Gould. Learning weighted lower linear envelope potentials in binary markov random fields. *IEEE transac-*

tions on pattern analysis and machine intelligence, 37(7): 1336–1346, 2015.

Allaudeen Hameed. Time-varying factors and cross-autocorrelations in short-horizon stock returns. *Journal of Financial Research*, 20(4):435–458, 1997.

Peter L. Hammer. Some network flow problems solved with psuedo-boolean programming. *Operations Research*, 13:388–399, 1965.

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. A joint many-task model: Growing a neural network for multiple nlp tasks. *arXiv preprint arXiv:1611.01587*, 2016.

Kewei Hou. Industry information diffusion and the lead-lag effect in stock returns. *The Review of Financial Studies*, 20(4):1113–1138, 2007.

Hiroshi Ishikawa. Exact optimization for Markov random fields with convex priors. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 25:1333–1336, 2003.

Hiroshi Ishikawa. Higher-order clique reduction in binary graph cut. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

Michael C Jensen. Random walks: reality or mythcomment. *Financial Analysts Journal*, 23(6):77–85, 1967.

Charles D Kirkpatrick II and Julie A Dahlquist. *Technical analysis: the complete resource for financial market technicians*. FT press, 2010.

Pushmeet Kohli and M. Pawan Kumar. Energy minimization for linear envelope MRFs. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

Pushmeet Kohli, M. Pawan Kumar, and Philip H. S. Torr. P3 & beyond: Solving energies with higher order cliques. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

Pushmeet Kohli, Lubor Ladicky, and Philip H. S. Torr. Graph cuts for minimizing higher order potentials. Technical report, Microsoft Research, 2008.

Pushmeet Kohli, Philip HS Torr, et al. Robust higher order potentials for enforcing label consistency. *International Journal of Computer Vision*, 82(3):302–324, 2009.

Vladimir Kolmogorov and Ramin Zabih. What energy functions can be minimized via graph cuts? *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 26:65–81, 2004.

Andrew W Lo and A Craig MacKinlay. When are contrarian profits due to stock market overreaction? *The review of financial studies*, 3(2):175–205, 1990.

Burton G Malkiel and Eugene F Fama. Efficient capital markets: A review of theory and empirical work. *The journal of Finance*, 25(2):383–417, 1970.

Grant McQueen, Michael Pinegar, and Steven Thorley. Delayed reaction to good news and the cross-autocorrelation of portfolio returns. *The Journal of Finance*, 51(3):889–919, 1996.

Timothy S Mech. Portfolio return autocorrelation. *Journal of Financial Economics*, 34(3):307–344, 1993.

Sebastian Nowozin and Christoph H. Lampert. Structured learning and prediction in computer vision. *Foundations and Trends in Computer Graphics and Vision*, 6(3–4):185–365, 2011.

Cheol-Ho Park and Scott H Irwin. What do we know about the profitability of technical analysis? *Journal of Economic Surveys*, 21(4):786–826, 2007.

Yao Qin, Dongjin Song, Haifeng Cheng, Wei Cheng, Guofei Jiang, and Garrison Cottrell. A dual-stage attention-based recurrent neural network for time series prediction. *arXiv preprint arXiv:1704.02971*, 2017.

Carsten Rother, Pushmeet Kohli, Wei Feng, and Jiaya Jia. Minimizing sparse higher order energy functions of discrete variables. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

Anders Søgaard and Yoav Goldberg. 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)*, volume 2, pages 231–235, 2016.

SL Sun et al. A decision tree model for meta-investment strategy of stock based on sector rotating. *Fuzzy Systems and Data MiningII: Proceedings of FSDM 2016*, 293:194, 2016.

Matrin Szummer, Pushmeet Kohli, and Derek Hoiem. Learning CRFs using graph-cuts. In *Proc. of the European Conference on Computer Vision (ECCV)*, 2008.

Ben Taskar, Vasco Chatalbashev, Daphne Koller, and Carlos Guestrin. Learning structured prediction models: A large margin approach. In *Proc. of the International Conference on Machine Learning (ICML)*, 2005.

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

Fang Yan. *The Research of Information Integration Under Chinese Stock Market Impact*. PhD thesis, Tianjin University, 2012.

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

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

Algorithm 2 Learning lower linear envelope MRFs with latent variables.

Set $MaxIter = 100$
input training set $\{\boldsymbol{y}_i\}_{i=1}^n$, regularization constant $C > 0$, and tolerance $\epsilon \geq 0$
Initialize $\boldsymbol{\theta}$ using Algorithm 1
repeat
  Set $iter = 0$
  for each training example, $i = 1, \ldots, n$ **do**
    compute $\boldsymbol{z}_i^* = \text{argmax}_{\mathbf{z} \in \mathcal{Z}} \, \theta \cdot \phi(\mathbf{y}_i, \mathbf{z})$
  end for
  initialize active constraints set $\mathcal{C}_i = \{\}$ for all i
  repeat
    solve the quadratic programming problem in equation 32 with respect to active constraints set $\mathcal{C}_i$ for all i and concavity constraints $A\boldsymbol{\theta} \geq \epsilon$ to get $\hat{\boldsymbol{\theta}}$ and $\hat{\boldsymbol{\xi}}$
    for each training example, $i = 1, \ldots, n$ **do**
      compute $\hat{\boldsymbol{y}}_i, \hat{\boldsymbol{z}}_i = \text{argmin}_{\boldsymbol{y}} \, E(\boldsymbol{y}, \boldsymbol{z}; \hat{\boldsymbol{\theta}}) - \Delta(\boldsymbol{y}, \boldsymbol{z}, \boldsymbol{y}_i)$
      if $\hat{\xi}_i + \epsilon < \Delta(\hat{\boldsymbol{y}}_i, \hat{\boldsymbol{z}}_i, \boldsymbol{y}_i) - E(\hat{\boldsymbol{y}}_i, \hat{\boldsymbol{z}}_i; \hat{\boldsymbol{\theta}}) + E(\boldsymbol{y}_i, \boldsymbol{z}_i^*; \hat{\boldsymbol{\theta}})$ **then**
        $\mathcal{C}_i \leftarrow \mathcal{C}_i \cup \{\boldsymbol{y}_i^\star\}$
      end if
    end for
  until no more violated constraints
  return parameters $\hat{\boldsymbol{\theta}}$
  Set $iter = iter + 1$
until $iter \geq MaxIter$
return parameters $\hat{\boldsymbol{\theta}}$