Enhancing Stock Movement Prediction with Adversarial Training

Fuli Feng[1], **Huimin Chen**[2], **Xiangnan He**[3*], **Ji Ding**[4], **Maosong Sun**[2] and **Tat-Seng Chua**[1]
[1]National University of Singapore
[2]Tsinghua Unversity
[3]University of Science and Technology of China
[4]University of Illinois at Urbana-Champaign
{fulifeng93,huimchen1994,xiangnanhe,chuats}@gmail.com, jiding2@illinois.edu, sms@tsinghua.edu.cn

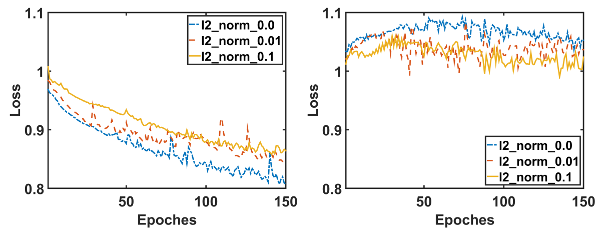
Abstract

This paper contributes a new machine learning solution for stock movement prediction, which aims to predict whether the price of a stock will be up or down in the near future. The key novelty is that we propose to employ adversarial training to improve the generalization of a neural network prediction model. The rationality of adversarial training here is that the input features to stock prediction are typically based on stock price, which is essentially a stochastic variable and continuously changed with time by nature. As such, normal training with static price-based features (*e.g.,* the close price) can easily overfit the data, being insufficient to obtain reliable models. To address this problem, we propose to add perturbations to simulate the stochasticity of price variable, and train the model to work well under small yet intentional perturbations. Extensive experiments on two real-world stock data show that our method outperforms the state-of-the-art solution [Xu and Cohen, 2018] with 3.11% relative improvements on average *w.r.t.* accuracy, validating the usefulness of adversarial training for stock prediction task.

1 Introduction

Stock market is one of the largest financial markets, having reached a total value of 80 trillion dollars[1]. Predicting the future status of a stock has always been of great interest to many players in a stock market. While the exact price of a stock is known to be unpredictable [Walczak, 2001; Nguyen *et al.*, 2015], research efforts have been focused on predicting the stock price movement — e.g., whether the price will go up/down, or the price change will exceed a threshold — which is more achievable than stock price prediction [Adebiyi *et al.*, 2014; Feng *et al.*, 2018; Xu and Cohen, 2018].

Stock movement prediction can be addressed as a classification task. After defining the label space and features to describe a stock at a time, we can apply standard supervised learning methods such as support vector machines [Huang *et al.*, 2005] and neural networks [Xu and Cohen, 2018] to build the predictive model. Although technically feasible, we argue that such methods could suffer from weak generalization due to the highly stochastic property of stock market. Figure 1 provides an empirical evidence on the weak generalization, where we split the data into training and validation by time, and train an Attentive LSTM model [Qin *et al.*, 2017] on the historical prices of stocks to predict their movements. From Figure 1(a), we can see the training loss gradually decreases with more training epochs, which is as expected. However, the validation loss shown in Figure 1(b) does not exhibit a decreasing trend; instead, it only fluctuates around the initialization state without a clear pattern. In other words, the benefits of the model learned on training examples do not translate to improvements on predicting unknown validation examples. We have thoroughly explored the L_2 regularization (results of different lines), a common technique to improve model generalization, however, the situation has not improved.



(a) Train (b) Validation

Figure 1: Training process of Attentive LSTM with L_2 regularization coefficient of $0, 0.01$, and 0.1.

We postulate the reason is that standard classification methods are assumed to learn from static inputs, such as pixel values in images and term frequencies in documents. When dealing with stochastic variable such as stock price, the static input assumption does not hold and such methods fail to generalize well. Specifically, existing methods for stock prediction typically feed into price-based features, such as the price at a particular time-step or average price on multiple time-steps [Edwards *et al.*, 2007; Nelson *et al.*, 2017]. Since a stock's price continuously changes with time (during market hours), price-based features are essentially stochastic variables, being fundamentally different from the traditional static inputs. To be more specific, the features of a training in-

*Corresponding author.

[1]https://data.worldbank.org/indicator/CM.MKT.TRAD.CD?view=chart

stance can be seen as a "sample" drawn from the distribution of input variables at a particular time-step. Without properly handling the stochasticity of input variables, the method can easily overfit the training data and suffer from weak generalization ability.

In this work, we propose to employ adversarial training to account for the stochastic property of stock market to learn stock movement prediction model. Our primary consideration is that given a training example at a particular time-step with fixed input features, the trained model is expected to generate the same prediction on other samples drawn from the inherent distribution of input variables. To implement this idea, we can generate additional samples (simulation of the stochasticity) by adding small perturbations on input features, and train the model to perform well on both clean examples and perturbed examples. It is the adversarial training method that has been commonly used in computer vision tasks [Kurakin *et al.*, 2017]. However, the problem is that the features to stock prediction models are usually sequential (see Figure 2), such that adding perturbations on the features of all time units can be very time-consuming; moreover, it may cause unintentional interactions among the perturbations of different units which are uncontrollable. To resolve the concern, we instead add perturbations on the high-level prediction features of the model, *e.g.,* the last layer which is directly projected to the final prediction. Since most deep learning methods learn abstract representation in the higher layers, their sizes are usually much smaller than the input size. As such, adding perturbations to high-level features is more efficient, and meanwhile it can also retain the stochasticity.

We implement our adversarial training proposal on an Attentive LSTM model, which is a highly expressive model for sequential data. We add perturbations to the prediction features of the last layer, and dynamically optimize the perturbations to make them change the model's output as much as possible. We then train the model to make it perform well on both clean features and perturbed features. As such, the adversarial training process can be understood as enforcing a dynamic regularizer, which stabilizes the model training and makes the model perform well under stochasticity.

The main contributions of this paper are summarized as:

- We investigate the generalization difficulty in stock movement prediction and highlight the necessity of dealing with the stochastic property of input features.
- We propose an adversarial training solution to address the stochastic challenge, and implement it on a deep learning model for stock movement prediction.
- We conduct extensive experiments on two public benchmarks, validating improvements over several state-of-the-art methods and showing that adversarial learning makes the classifier more robust and more generalizable.

2 Problem Formulation

We use bold capital letters (*e.g.,* $\boldsymbol{X}$) and bold lower letters (*e.g.,* $\boldsymbol{x}$) to denote matrices and vectors, respectively. In addition, normal lower case letters (*e.g.,* x) and Greek letters (*e.g.,* λ) are used to represent scalars and hyper-parameters, respectively. All vectors are in column form, if not otherwise

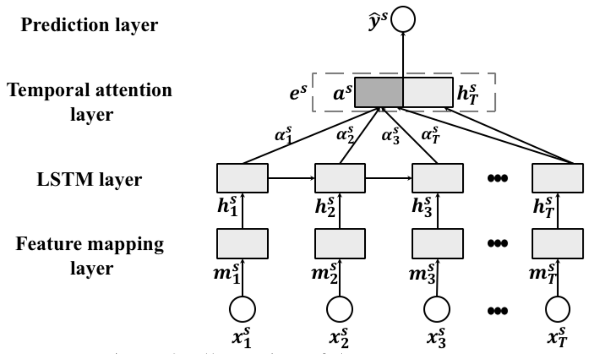


Figure 2: Illustration of the *Attentive LSTM*.

specified. The symbols $tanh$ and σ stand for the hyperbolic tangent function and sigmoid function, respectively.

The formulation of stock movement prediction task is to learn a prediction function $\hat{y}^s = f(\boldsymbol{X}^s; \boldsymbol{\Theta})$ which maps a stock (s) from its sequential features ($\boldsymbol{X}^s$) to the label space. In other words, the function f with parameters $\boldsymbol{\Theta}$ aims to predict the movement of stock s at the next time-step from the sequential features $\boldsymbol{X}^s$ in the latest T time-steps. $\boldsymbol{X}^s = [\boldsymbol{x}_1^s, \cdots, \boldsymbol{x}_T^s] \in \mathbb{R}^{D \times T}$ is a matrix which represents the sequential input features (*e.g.,* open and close prices, as detailed in Table 1) in the lag of past T time-steps, where D is the dimension of features.

Assuming that we have S stocks, we learn the prediction function by fitting their ground truth labels $\mathbf{y} = [y^1, \cdots, y^S] \in \mathbb{R}^S$, where y^s (1/-1) is the ground truth label of stock s in the next time-step. We then formally define the problem as:

Input: A set of training examples $\{(\boldsymbol{X}^s, y^s)\}$.
Output: A prediction function $f(\boldsymbol{X}^s; \boldsymbol{\Theta})$, predicting the movement of stock s in the following time-step.

In the practical scenario, we could typically access a long history of each stock, and construct many training examples for each stock by moving the lag along the history. Nevertheless, we use a simplified formulation without loss of generality by only considering one specific lag (*i.e.,* one training example for each stock) for briefness of presenting the proposed method.

3 Adversarial Attentive LSTM (Adv-ALSTM)

3.1 Attentive LSTM

The Attentive LSTM (ALSTM) mainly contains four components: feature mapping layer, LSTM layer, temporal attention, and prediction layer, as shown in Figure 2.

Feature mapping layer. Previous work shows that a deeper input gate would benefit the modeling of temporal structures of LSTM [Graves *et al.*, 2013; Wu *et al.*, 2018]. Inspired by their success, we employ a fully connected layer to project the input features into a *latent representation*. At each time-step, it performs as $\boldsymbol{m}_t^s = tanh(\boldsymbol{W}_m \boldsymbol{x}_t^s + \boldsymbol{b}_m)$, which projects the input features to a latent space with dimensionality of E. $\boldsymbol{W}_m \in \mathbb{R}^{E \times D}$ and $\boldsymbol{b}_m \in \mathbb{R}^E$ are parameters to be learned.

LSTM layer. Owing to its ability to capture long-term dependency, LSTM has been widely used to process sequential data [Qin *et al.*, 2017; Chen *et al.*, 2018a]. The general idea of LSTM is to recurrently project the input sequence into a sequence of *hidden representations*. At each time-step, the LSTM learns the hidden representation ($\boldsymbol{h}_t^s$) by jointly considering the input ($\boldsymbol{m}_t^s$) and previous hidden representation ($\boldsymbol{h}_{t-1}^s$) to capture sequential dependency. We formulate it as $\boldsymbol{h}_t^s = LSTM(\boldsymbol{m}_t^s, \boldsymbol{h}_{t-1}^s)$ of which the detailed formulation can be referred to [Hochreiter and Schmidhuber, 1997]. To capture the sequential dependencies and temporal patterns in the historical stock features, an LSTM layer is applied to map $[\boldsymbol{m}_1^s, \cdots, \boldsymbol{m}_T^s]$ into hidden representations $[\boldsymbol{h}_1^s, \cdots, \boldsymbol{h}_T^s] \in \mathbb{R}^{U \times T}$ with the dimension of U.

Temporal Attention Layer. The attention mechanism has been widely used in LSTM-based solutions for sequential learning problems[Cho *et al.*, 2014; Chen *et al.*, 2018a]. The idea of attention is to compress the hidden representations at different time-steps into an *overall representation* with adaptive weights. The attention mechanism aims to model the fact that data at different time-steps could contribute differently to the representation of the whole sequence. For stock representation, status at different time-steps might also contribute differently. For instance, days with maximum and minimum prices in the lag might have higher contributions to the overall representation. As such, we use an attention mechanism to aggregate the hidden representations as,

$$\boldsymbol{a}^s = \sum_{t=1}^{T} \alpha_t^s \boldsymbol{h}_t^s, \ \ \alpha_t^s = \frac{exp^{\widetilde{\alpha}_t^s}}{\sum_{t=1}^{T} exp^{\widetilde{\alpha}_t^s}}, \qquad (1)$$
$$\widetilde{\alpha}_t^s = \boldsymbol{u}_a^T tanh(\boldsymbol{W}_a \boldsymbol{h}_t^s + \boldsymbol{b}_a),$$

where $\boldsymbol{W}_a \in \mathbb{R}^{E' \times U}$, $\boldsymbol{b}_a$ and $\boldsymbol{u}_a \in \mathbb{R}^{E'}$ are parameters to be learned; and $\boldsymbol{a}^s$ is the aggregated representation that encodes the overall patterns in the sequence.

Prediction Layer. Instead of directly making prediction from $\boldsymbol{a}^s$, we first concatenate $\boldsymbol{a}^s$ with the last hidden state $\boldsymbol{h}_T^s$ into the *final latent representation* of stock s,

$$\boldsymbol{e}^s = [\boldsymbol{a}^{sT}, \boldsymbol{h}_T^{s\,T}]^T, \qquad (2)$$

where $\boldsymbol{e}^s \in \mathbb{R}^{2U}$. The intuition behind is to further emphasize the most recent time-step, which is believed to be informative for the following movement [Fama and French, 2012]. With $\boldsymbol{e}^s$, we use a fully connected layer as the predictive function to estimate the classification confidence $\hat{y}^s = \boldsymbol{w}_p^T \boldsymbol{e}^s + b_p$. Note that the final prediction is $sign(\hat{y}^s)$.

3.2 Adversarial Training

As with most classification solutions, the *normal* way of training the ALSTM is to minimize an objective function Γ:

$$\sum_{s=1}^{S} l(y^s, \hat{y}^s) + \frac{\alpha}{2}\|\boldsymbol{\Theta}\|_F^2, \ l(y^s, \hat{y}^s) = max(0, 1 - y^s \hat{y}^s). \qquad (3)$$

The first term is hinge loss [Rosasco *et al.*, 2004], which is widely used for optimizing classification models (more reasons of choosing it is further explained in the end of the section). The second term is a regularizer on the trainable parameters to prevent overfitting.

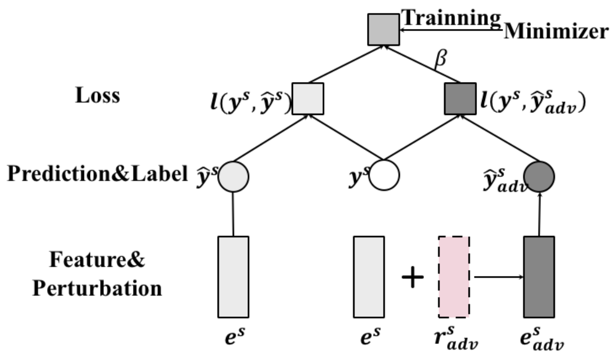


Figure 3: Illustration of the *Adversarial Attentive LSTM*.

Despite the wide usage of *normal training*, we argue that it is inappropriate for learning stock prediction models. This is because normal training assumes that the inputs are static, ignoring the stochastic property of these features (a training example is a sample drawn from the stochastic distribution of input variables). Note that the features are calculated from stock price, which continuously changes with time and is affected by stochastic trading behaviours at a particular time-step [Musgrave, 1997]. As such, normal training might lead to model that overfits the data and lacks generalization ability (as shown in Figure 1). Note that is a model performs well under stochasticity would make same predictions for samples drawn from the inherent distribution. Considering that stock price is continuous, our intuition is to intentionally simulate samples by adding small perturbations on static input features. By enforcing the predictions on the simulated samples to be same, the model could capture stochasticity.

Adversarial training [Goodfellow *et al.*, 2015; Kurakin *et al.*, 2017] implements the aforementioned intuition. It trains a model with both clean examples (*i.e.,* examples in the training set) and adversarial examples (AEs) [Szegedy *et al.*, 2013]. The AEs are malicious inputs generated by adding intentional perturbations to features of clean examples. The perturbation, named as *adversarial perturbation* (AP) is the direction that leads to the largest change of model prediction. Despite its success in image classification [Kurakin *et al.*, 2017], it is infeasible to be directly applied to stock prediction. This is because calculating perturbations relies on calculation of the gradients regarding the input, which would be time-consuming (caused by the back-propagation through time-step of the LSTM layer). Besides, considering the fact that the gradients of the input are dependent across different time-steps, there might be unintentional interactions among the perturbations on different time-steps, which are uncontrollable. To address these problems, we propose to generate AEs from latent representation $\boldsymbol{e}^s$, as shown in Figure 3.

Before introducing the calculation of AEs, we first elaborate the objective function of Adv-ALSTM:

$$\Gamma_{adv} = \sum_{s=1}^{S} l(y^s, \hat{y}^s) + \beta \sum_{s=1}^{S} l(y^s, \hat{y}^s_{adv}) + \frac{\alpha}{2}\|\boldsymbol{\Theta}\|_F^2. \qquad (4)$$

The second term is an adversarial loss where $\hat{y}^s_{adv}$ is the classification confidence of the AE of stock s. β is a hyper-parameter to balance the losses of clean and adversarial examples. By minimizing the objective function, the model is encouraged to correctly classify both clean and adversarial examples. Note that a model correctly classifying an AE can make right predictions for examples with arbitrary perturbations at the same scale. This is because AP is the direction leading to the largest change of model prediction. Therefore, adversarial learning could enable ALSTM to capture the stochastic property of stock inputs.

At each iteration, the latent representation of an AE ($\boldsymbol{e}^s_{adv}$) is generated by the following formulation,

$$\boldsymbol{e}^s_{adv} = \boldsymbol{e}^s + \boldsymbol{r}^s_{adv}, \boldsymbol{r}^s_{adv} = \arg \max_{\boldsymbol{r}^s, \|\boldsymbol{r}^s\| \le \epsilon} l(y^s, \hat{y}^s_{adv}), \quad (5)$$

where $\boldsymbol{e}^s$ (introduced in Equation 2) is the final latent representation of stock s. $\boldsymbol{r}^s_{adv}$ is the associated AP. ϵ is a hyper-parameter to explicitly control the *scale* of perturbation. Since it is intractable to directly calculate $\boldsymbol{r}^s_{adv}$, we employ the fast gradient approximation method [Goodfellow *et al.*, 2015], $\boldsymbol{r}^s_{adv} = \epsilon \frac{\boldsymbol{g}^s}{\|\boldsymbol{g}^s\|}, \boldsymbol{g}^s = \frac{\partial l(y^s, \hat{y}^s)}{\partial \boldsymbol{e}^s}$. Specifically, the calculated perturbation is the gradient of loss function regarding the latent representation $\boldsymbol{e}^s$ under a L_2-norm constraint. Note that the gradient denotes the direction where the loss function increase the most at the given point $\boldsymbol{e}^s$, *i.e.,* , it would lead to the largest change on the model prediction.

Figure 4 illustrates the generation of adversarial examples. In a training iteration, given a clean example having loss larger than 0 (*i.e.,* $y^s \hat{y}^s < 1$), an AE is generated. The model is then updated to jointly minimize the losses for clean and adversarial examples, which would enforce the margin between clean examples and the decision boundary[2]. As such, it would benefit the model to predict examples with perturbations into the same class as the clean one. That is, the model could correctly predict samples drawn from the inherent stochastic distribution of inputs, capturing the stachasticity. While traditional models like support vector machines also push the decision boundary far from clean examples, the adversarial training adaptively adjusts the strength of enforcing margins during the training process since the AP ($\boldsymbol{r}^s_{adv}$) varies across iterations. Note that we select the hinge loss to encourage the training process to focus more on the examples close to the decision boundary.

4 Experiments

4.1 Experimental Settings

We evaluate the proposed method on two benchmarks on stock movement prediction, *ACL18* [Xu and Cohen, 2018] and *KDD17* [Zhang *et al.*, 2017].

ACL18 contains historical data from Jan-01-2014 to Jan-01-2016 of 88 high-trade-volume-stocks in NASDAQ and NYSE markets. Following [Xu and Cohen, 2018], we first align the trading days in the history, *i.e.,* removing weekends and public holidays that lack historical prices. We then move

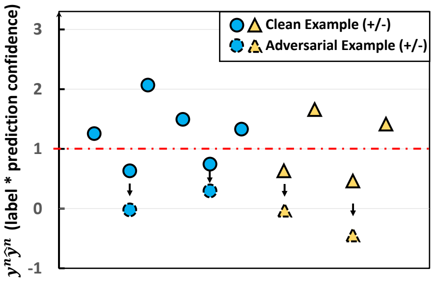


Figure 4: Intuitive illustration of adversarial examples.

Features	Calculation
c_open, c_high, c_low	*e.g.,* $c\_open = open_t/close_t - 1$
n_close, n_adj_close	*e.g.,* $n\_close = (close_t/close_{t-1} - 1$
5-day, 10-day, 15-day, 20-day, 25-day, 30-day	*e.g.,* $5\text{-}day = \frac{\sum_{i=0}^{4} adj\_close_{t-i}/5}{adj\_close_t} - 1$

Table 1: Features to describe the daily trend of a stock.

a lag with length of T along the aligned trading days to construct candidate examples (*i.e.,* one example for a stock on every trading day). We label the candidate examples according to the movement percent of stock close prices. Given a candidate example of stock s in the lag of $[T' - T + 1, T']$, the movement percent is calculated as $p^s_{T'+1}/p^s_{T'} - 1$, where $p^s_{T'}$ is the adjusted close price of stock s on day T'. Examples with movement percent $\geq 0.55\%$ and $\leq -0.5\%$ are identified as positive and negative examples, respectively. We temporally split the identified examples into training (Jan-01-2014 to Aug-01-2015), validation (Aug-01-2015 to Oct-01-2015), and testing (Oct-01-2015 to Jan-01-2016).

KDD17 includes longer history ranging from Jan-01-2007 to Jan-01-2016 of 50 stocks in U.S. markets. As the dataset is originally collected for predicting stock prices rather than movements, we follow the same approach as *ACL18* to identify positive and negative examples. We then temporally split the examples into training (Jan-01-2007 to Jan-01-2015), validation (Jan-01-2015 to Jan-01-2016) and testing (Jan-01-2016 to Jan-01-2017).

Features. Instead of using the raw EOD data, we define 11 temporal features ($\mathbf{x}^s_t$) to describe the trend of a stock s at trading day t. Table 1 elaborates the features associated with calculation. Our aim of defining these features are to: 1) normalize the prices of different stocks; 2) and explicitly capture the interaction of different prices (*e.g.,* open and close).

Baselines. We compare the following methods:

- Momentum (*MOM*) is a technical indicator that predicts negative or positive for each example with the trend in the last 10 days.
- Mean reversion (*MR*) predicts the movement of each example as the opposite direction of latest price towards the 30-day moving average.
- *StockNet* uses a Variational Autoencoder (VAE) to encode the stock input so as to capture the stochasticity, and a temporal attention to model the importance of different timesteps [Xu and Cohen, 2018]. Here we take our temporal features in Table 1 as inputs and tune its hidden size, dropout ratio, and auxiliary rate (α).

[2]Minimizing the hinge loss of the AE is adjusting $\boldsymbol{w}_p$ to enlarge $y^s \hat{y}^s_{adv} = y^s(\boldsymbol{w}_p^T \boldsymbol{e}^s + b) + y^s \boldsymbol{w}_p^T \boldsymbol{r}^s_{adv}$, which would increase the first term $y^s(\boldsymbol{w}_p^T \boldsymbol{e}^s + b) = y^s \hat{y}^s$. The results in Figure 5 (in Section 4) empirically demonstrate the effect of enforcing margins.

Method	ACL18		KDD17	
	Acc	MCC	Acc	MCC
MOM	47.01±—–	-0.0640±—–	49.75±—–	-0.0129±—–
MR	46.21±—–	-0.0782±—–	48.46±—–	-0.0366±—–
StockNet	54.96±—–	0.0165±—–	51.93±4e-1	0.0335±5e-3
LSTM	53.18±5e-1	0.0674±5e-3	51.62±4e-1	0.0183±6e-3
ALSTM	54.90±7e-1	0.1043±7e-3	51.94±7e-1	0.0261±1e-2
Adv-ALSTM	**57.20±—–**	**0.1483±—–**	**53.05±—–**	**0.0523±—–**
RI	4.02%	42.19%	2.14%	56.12%

Table 2: Performance comparison on the two datasets. RI denotes the relative improvement of Adv-ALSTM compared to the best baseline. We copy StockNet from the origin paper.

Datasets	Acc	MCC
ACL18	55.08±2e0	0.1103±4e-2
KDD17	52.43±5e-1	0.0405±8e-3

Table 3: Performance of Rand-ALSTM on the two datasets.

- *LSTM* is a neural network with an LSTM layer and a prediction layer [Nelson *et al.*, 2017]. We tune three hyper-parameters, number of hidden units (U), lag size (T), and weight of regularization term (λ).
- *ALSTM* is the *Attentive LSTM* [Qin *et al.*, 2017], which is optimized with normal training. Similar as *LSTM*, we also tune U, T, and λ.

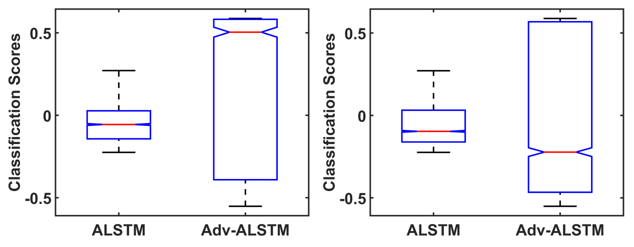
Evaluation Metrics. We evaluate the prediction performance with two metrics, Accuracy (Acc) and Matthews Correlation Coefficient (MCC) [Xu and Cohen, 2018] of which the ranges are in $[0, 100]$ and $[-1, 1]$. Note that better performance is evidenced by higher value of the metrics.

Parameter Settings. We implement the *Adv-ALSTM* with Tensorflow and optimize it using the mini-batch Adam[Diederik and Jimmy, 2015] with a batch size of 1,024 and an initial learning rate of 0.01. We search the optimal hyper-parameters of *Adv-ALSTM* on the validation set. For U, T, and λ, *Adv-ALSTM* inherits the optimal settings from *ALSTM*, which are selected via grid-search within the ranges of [4, 8, 16, 32], [2, 3, 4, 5, 10, 15], and [0.001, 0.01, 0.1, 1], respectively. We further tune β and ϵ within [0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1] and [0.001, 0.005, 0.01, 0.05, 0.1], respectively. We report the mean testing performance when *Adv-ALSTM* performs best on the validation set over five different runs. Code could be accessed through https://github.com/hennande/Adv-ALSTM.

4.2 Experimental Results

Performance Comparison. Tables 2 shows the prediction performance of compared methods on the two datasets regarding Acc and MCC, respectively. From the table, we have the following observations:

- *Adv-ALSTM* achieves the best results in all cases. Compared to the baselines, *Adv-ALSTM* exhibits an improvement of 4.02% and 42.19% (2.14% and 56.12%) on the *ACL18* (*KDD17*) dataset regarding Acc and MCC, respectively. It justifies the effectiveness of adversarial training, which might be due to enhancing the model generalization via adaptively simulating perturbations during training.
- Specifically, compared to *StockNet*, which captures stochasticity of stock inputs with VAE, *Adv-ALSTM* achieves significant improvements. We postulate the reason is that *StockNet* cannot explicitly model the scale and direction of stochastic perturbation since it relies on Monte Carlo sampling during the training process.
- Among the baselines, *ALSTM* outperforms *LSTM* by 1.93% and 48.69% on average *w.r.t.* Acc and MCC, which validates the impact of attention [Qin *et al.*, 2017]. Besides, *MOM* and *MR* performs worse than all the machine learning-based methods as expected, which justifies that historical patterns help in stock prediction task.



(a) Validation of ACL18 (b) Testing of ACL18

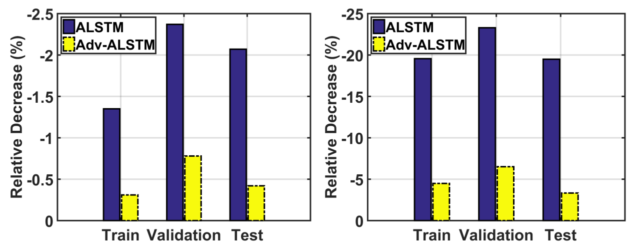
Figure 5: Distributions of classification confidences assigned by ALSTM and Adv-ALSTM for clean examples.

Stochastic Perturbation *VS.* Adversarial Perturbation

We further investigate the effectiveness of adversarial training via comparing adversarial perturbations and random ones. *Rand-ALSTM* is a variance of *Adv-ALSTM*, which generates additional examples by adding random perturbations to the input of clean examples. Table 3 shows the performance of *Rand-ALSTM* on the two datasets. By cross comparing it with Table 2, we observe that: 1) Compared to *Rand-ALSTM*, *Adv-ALSTM* achieves significant improvements. For instance, its performance *w.r.t.* Acc on *ACL18* is 3.95% better than that of *Rand-ALSTM*. It demonstrates that adversarial perturbations are helpful for stock prediction, similar to that reported in the original image classification tasks [Goodfellow *et al.*, 2015]. 2) *Rand-ALSTM* outperforms *ALSTM*, which is purely trained with clean examples, with an average improvement of 0.64% *w.r.t.* Acc on the two datasets. This highlights the necessity of dealing with stochastic property of stock features.

We now investigate the impacts of adversarial training to answer: 1) Whether the adversarial training *enforces the margin* between clean examples and the decision boundary. 2) Whether the adversarial training *enhances the robustness* of the model against adversarial examples. Note that we only show the results on the *ACL18* dataset as the results on *KDD17* admit the same observations.

Recall that the difference between *Adv-ALSTM* and *ALSTM* is learning parameters with adversarial training or standard training. We answer the first question by comparing the classification confidence of clean examples (larger value denotes larger margin to the decision boundary) assigned by *Adv-ALSTM* and *ALSTM*. Figure 5 shows the distributions of the classification confidences assigned by *ALSTM* and *Adv-ALSTM*. As can be seen, the confidences of *Adv-ALSTM* distribute in a range ([-0.6, 0.6] roughly), which is about 1.5 times larger than that of *ALSTM* ([-0.2, 0.3]). It indicates that adversarial training pushes the decision boundary far from clean examples, which is believed to help enhance the robustness and generalization ability of the model.



(a) Acc (b) MCC

Figure 6: Robustness against adversarial example of ALSTM and Adv-ALSTM. Each plotted number is the RPD of a model on adversarial examples compared to clean ones.

We then investigate the second question via comparing the performance of *ALSTM* and *Adv-ALSTM* on the clean and associated adversarial examples. Figures 6(a) and 6(b) illustrate the relative performance decrease (RPD) of *ALSTM* and *Adv-ALSTM* on adversarial examples regarding the one on clean examples, respectively. Note that larger absolute value of R-PD indicates that the model is more vulnerable to adversarial perturbations. As can be seen, the average RPD of *ALSTM* is 4.31 (6.34) times larger as compared to *Adv-ALSTM* regarding Acc (MCC). This justifies the potential of enhancing model robustness with adversarial training.

5 Related Work

Stock Movement Prediction methods mainly fall under two categories, *technical analysis* and *fundamental analysis* (FA). TA takes historical prices of a stock as features to forecast its movement. Most of recent TA methods mine stock movements with deep models [Lin *et al.*, 2017; Nelson *et al.*, 2017; Chong *et al.*, 2017]. Among them, recurrent neural networks like LSTM have become key components to capture the temporal patterns of stock prices [Nelson *et al.*, 2017; Lin *et al.*, 2017]. Besides, other advanced neural models, such as convolution neural network (CNN) [Lin *et al.*, 2017] and deep Boltzmann machine [Chong *et al.*, 2017], are also evidenced to be beneficial for capturing the non-linearity of stock prices.

In addition to price features, FA also examines related economic, financial, and other qualitative and quantitative factors [Hu *et al.*, 2018; Zhang *et al.*, 2018; Li *et al.*, 2018; Xu and Cohen, 2018]. For instance, Xu and Cohen [2018] incorporate signals from social media, which reflects opinions from general users, to enhance stock movement prediction. Specifically, they employ a VAE to learn a stock representation by jointly encoding the historical prices and tweets mentioning it. Moreover, Zhang *et al.* [2018] further consider news events related to a stock or the associated company via a coupled matrix and tensor factorization framework.

Both TA and FA studies show that price features play crucial roles in stock movement prediction. However, most of the existing works assume stock price as stationary, which thus lack the ability to deal with its stochastic property. S-tockNet [Xu and Cohen, 2018] is the only exception which tackles this problem via VAE. VAE encodes the inputs into a latent distribution and enforces samples from the latent distribution to be decoded with the same prediction. Generally, the philosophy behind is similar as the simulation of stochastic perturbations since one sample from the latent distribution can be seen as adding stochastic perturbation to the latent representation. As compared to our method, our perturbation is intentionally generated which indicates leads to hardest examples for the model to obtain the target prediction. In addition, the proposed method can be easily adapted to other solutions of stock movement predictions.

Adversarial Learning has been intensively studied by training a classification model to defense adversarial examples, which are intentionally generated to perturb the model. The existing works mainly concentrate on computer vision tasks like image classification [Goodfellow *et al.*, 2015; iyato *et al.*, 2017; Kurakin *et al.*, 2017; Yang *et al.*, 2018; Chen *et al.*, 2018b]. Owing to the property that image features are typically continued real values, adversarial examples are directly generated in the feature space. Recently, several works extend the adversarial learning to tasks with discrete inputs such as text classification (a sequence of words) [iyato *et al.*, 2017], recommendation (user and item IDs) [He *et al.*, 2018], and graph node classification (graph topology) [Dai *et al.*, 2018; Feng *et al.*, 2019]. Rather than in the feature space, these works generate adversarial examples from embedding of inputs such as word, user (item), and node embeddings. Although this work is inspired by these adversarial learning research efforts, it targets a distinct task—stock movement prediction, of which the data are time series with stochastic property. To the best of our knowledge, this work is the first one to explore the potential of adversarial training in time-series analytics.

6 Conclusion

We showed that neural network solutions for stock movement prediction could suffer from weak generalization ability since they lack the ability to deal with the stochasticity of stock features. To solve this problem, we proposed an *Adversarial Attentive LSTM* solution, which leverages adversarial training to simulate the stochasticity during model training. We conducted extensive experiments on two benchmark datasets and validated the effectiveness of the proposed solution, signifying the importance of accounting for the stochasticity of stock prices in stock movement prediction. Morever, the results showed that adversarial training enhances the robustness and generalization of the prediction model.

In future, we plan to explore the following directions: 1) we are interested in testing **Adv-ALSTM** in movement prediction of more assets such as commodities. 2) We plan to apply adversarial training to stock movement solutions with different structures such as the CNNs [Lin *et al.*, 2017]. 3) We will explore the effect of adversarial training over fundamental analysis methods of stock movement prediction.

Acknowledgments

This research is supported by the National Research Foundation Singapore under its AI Singapore Programme (AISG-100E-2018-012) and NExT Research under its IRC@SG Funding Initiative.

References

[Adebiyi *et al.*, 2014] A. Adebiyi, A. Adewumi, and C. Ayo. Comparison of arima and artificial neural networks models for stock price prediction. *JAM*, 2014.

[Chen *et al.*, 2018a] J. Chen, C. Ngo, F. Feng, and T. Chua. Deep understanding of cooking procedure for cross-modal recipe retrieval. In *MM*, pages 1020–1028. ACM, 2018.

[Chen *et al.*, 2018b] L. Chen, H. Zhang, J. Xiao, W. Liu, and S. Chang. Zero-shot visual recognition using semantics-preserving adversarial embedding networks. In *CVPR*, pages 1043–1052, 2018.

[Cho *et al.*, 2014] K. Cho, M. Van, C. Gulcehre, et al. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *EMNLP*, pages 1724–1734, 2014.

[Chong *et al.*, 2017] E. Chong, C. Han, and F. Park. Deep learning networks for stock market analysis and prediction: Methodology, data representations, and case studies. *ESA*, 83:187–205, 2017.

[Dai *et al.*, 2018] H. Dai, H. Li, T. Tian, X. Huang, L. Wang, J. Zhu, and L. Song. Adversarial attack on graph structured data. In *ICML*, pages 1115–1124, 2018.

[Diederik and Jimmy, 2015] K. Diederik and B. Jimmy. Adam: A method for stochastic optimization. *ICLR*, 2015.

[Edwards *et al.*, 2007] R. Edwards, J. Magee, and C. Bassetti. *Technical analysis of stock trends*. CRC press, 2007.

[Fama and French, 2012] E. F. Fama and K. R. French. Size, value, and momentum in international stock returns. *JFE*, 105(3):457–472, 2012.

[Feng *et al.*, 2018] F. Feng, X. He, X. Wang, C. Luo, Y. Liu, and T. Chua. Temporal relational ranking for stock prediction. *TOIS*, 2018.

[Feng *et al.*, 2019] F. Feng, X. He, J. Tang, and T. Chua. Graph adversarial training: Dynamically regularizing based on graph structure. *arXiv*, 2019.

[Goodfellow *et al.*, 2015] I. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *ICLR*, 2015.

[Graves *et al.*, 2013] A. Graves, A. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *ICASSP*, pages 6645–6649. IEEE, 2013.

[He *et al.*, 2018] X. He, Z. He, X. Du, and T. Chua. Adversarial personalized ranking for recommendation. In *SIGIR*, pages 355–364, 2018.

[Hochreiter and Schmidhuber, 1997] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[Hu *et al.*, 2018] Z. Hu, W. Liu, J. Bian, X. Liu, and T. Liu. Listening to chaotic whispers: A deep learning framework for news-oriented stock trend prediction. In *WSDM*, pages 261–269, 2018.

[Huang *et al.*, 2005] W. Huang, Y. Nakamori, and S. Wang. Forecasting stock market movement direction with support vector machine. *COR*, 32(10):2513–2522, 2005.

[iyato *et al.*, 2017] MT. iyato, A. M. Dai, and I. Goodfellow. Adversarial training methods for semi-supervised text classification. *ICLR*, 2017.

[Kurakin *et al.*, 2017] A. Kurakin, I. Goodfellow, and S. Bengio. Adversarial machine learning at scale. *ICLR*, 2017.

[Li *et al.*, 2018] Q. Li, Y. Chen, J. Wang, Y. Chen, and H. Chen. Web media and stock markets: A survey and future directions from a big data perspective. *TKDE*, 30(2):381–399, 2018.

[Lin *et al.*, 2017] T. Lin, T. Guo, and K. Aberer. Hybrid neural networks for learning the trend in time series. In *IJCAI*, pages 2273–2279, 2017.

[Musgrave, 1997] G. Musgrave. A random walk down wall street. *Business Economics*, 32(2):74–76, 1997.

[Nelson *et al.*, 2017] D. Nelson, A. Pereira, and R. Oliveira. Stock market's price movement prediction with lstm neural networks. In *IJCNN*, pages 1419–1426. IEEE, 2017.

[Nguyen *et al.*, 2015] T. H. Nguyen, K. Shirai, and J. Velcin. Sentiment analysis on social media for stock movement prediction. *ESA*, 42(24):9603–9611, 2015.

[Qin *et al.*, 2017] Y. Qin, D. Song, H. Cheng, W. Cheng, G. Jiang, and G. Cottrell. A dual-stage attention-based recurrent neural network for time series prediction. In *IJCAI*, pages 2627–2633, 2017.

[Rosasco *et al.*, 2004] L. Rosasco, E. D. Vito, A. Caponnetto, M. Piana, and A. Verri. Are loss functions all the same? *Neural Computation*, 16(5):1063–1076, 2004.

[Szegedy *et al.*, 2013] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *arXiv*, 2013.

[Walczak, 2001] S. Walczak. An empirical analysis of data requirements for financial forecasting with neural networks. *JMIS*, 17(4):203–222, 2001.

[Wu *et al.*, 2018] L. Wu, C. Quan, C. Li, and D. Ji. Parl: Let strangers speak out what you like. In *CIKM*, pages 677–686. ACM, 2018.

[Xu and Cohen, 2018] Y. Xu and S. Cohen. Stock movement prediction from tweets and historical prices. In *ACL*, volume 1, pages 1970–1979, 2018.

[Yang *et al.*, 2018] X. Yang, H. Zhang, and J. Cai. Shuffle-then-assemble: Learning object-agnostic visual relationship features. In *ECCV*, pages 36–52, 2018.

[Zhang *et al.*, 2017] L. Zhang, C. Aggarwal, and G. Qi. Stock price prediction via discovering multi-frequency trading patterns. In *SIGKDD*, pages 2141–2149. ACM, 2017.

[Zhang *et al.*, 2018] X. Zhang, Y. Zhang, S. Wang, Y. Yao, B. Fang, and P. Yu. Improving stock market prediction via heterogeneous information fusion. *KBS*, 143:236–247, 2018.