2022年度
修士論文

時系列・非時系列属性の混在データに対する埋込モデルに関する研究

指導教員　　江口 浩二 教授

審査教員　主査　江口 浩二 教授

審査教員　副査　山田 宏　教授

審査教員　副査　ANDRADE SILVA DANIEL GEORG 准教授

広島大学大学院 先進理工系科学研究科

情報科学プログラム

M216916　稲葉 勇哉

時系列・非時系列属性の混在データに対する埋込モデルに関する研究

稲葉　勇哉

要旨

本研究では，時系列属性と非時系列属性が混在したデータセットに対し，効果的な特徴量抽出と，それに基づく埋込モデルを応用した分類モデルを構築することを目的とする．

従来より，実務現場では機械学習での分類予測（例：顧客属性を利用した最適な商品の分類など）において，非時系列属性（静的属性）と時系列属性（動的属性）の混在データを用いる際には，時系列属性を前処理し特徴量抽出を行ったうえで分類モデルに投入することが多く，人の手による前処理では特徴量抽出が十分に最適化できていないという課題がある．

本研究では以上の課題を解決するため，時系列属性の特徴量抽出から，非時系列属性と統合した後の分類予測までを学習する End-to-End モデルの構築を行う．具体的には，Graph Deviation Network（GDN）を時系列属性の埋込表現学習に使用し，サンプルに含まれている複数の時系列属性の相互依存を考慮しながら各時系列属性をベクトルで表現する．それに加えて非時系列属性を考慮した分類モデルとして構築する．Pre-training で時系列ごとの「次点の値」を回帰タスクとして学習し，そこで得られた重みパラメータを Fine-tuning によって再学習する．ただし，この Fine-tuning による再学習の際に，時系列属性の埋込ベクトル表現と非時系列属性を統合し，最終層を分類器とする．

使用するデータセットとして Yahoo!ファイナンスで公開されている上場企業のデータを用いる．時系列属性として過去の株価，非時系列属性として企業の属性（会社規模，業種など）が含まれている．このデータを用いて，該当銘柄の株価が「上昇するか／変わらないか（変動率〇%以内）／下落するか」を予測する多値分類を行う．

本研究の提案手法である GDN 応用モデルによる分類予測の実験では，従来手法（XGBoost，LightGBM）の精度（Accuracy およびマクロ平均 F 値）を上回る結果となった．さらに追加実験として行ったクラス不均衡を設定した場合の GDN 応用モデルの損失関数（Cross Entropy と Dice Loss）の比較では，Dice Loss を設定したモデルがより良い性能を示した．しかしながら，実用化できるような精度には至っておらず，特徴量抽出における課題も残されている．GDN 応用モデルにおける課題を解決できれば，さらに精度を向上させることができると考えている．

目 次

第 1 章　序論

企業の実務現場においては，回帰や分類タスクといった教師あり学習のモデル構築を行うにあたり，分析に用いるデータの説明変数に時系列（動的）属性と非時系列（静的）属性が混在することがある．一般的な機械学習モデルにおいては，混在データのままモデルへ投入することができないため，時系列属性か非時系列属性のいずれかに着目して設計される場合が多く,「データに何らかの前処理を行うこと」もしくは「時系列・非時系列属性の混在データに適したモデルを新たに設計すること」が求められる．前者（前処理）におけるアプローチとしては，時系列属性から適切に特徴量を抽出し，非時系列属性と併せて説明変数として用いることが考えられる．後者（新たなモデル設計）におけるアプローチとしては，時系列モデルを拡張して非時系列属性を考慮できるようにすることが考えられる．研究では,「時系列属性からの特徴量抽出」および「非時系列属性を考慮した時系列モデルの拡張」に焦点を当て，多クラスの分類タスクを想定したうえで，効果的に時系列属性と非時系列属性の混在データに対処できるような応用モデルの構築に取り組む．

そもそも実務現場では時系列属性と非時系列属性が混在するデータセットを用いて分類タスクを行う場合，いずれかの属性に着目することが多く，例えば非時系列属性に着目した場合，時系列属性から特徴量を得るため前処理を行う必要がある．このような前処理を行うことで，後続のモデル（決定木や勾配ブースティングなど）での分析が可能となる．しかし，前処理はしばしば属人的な作業となりやすく，上記のような前処理〜モデル構築での分析の一連において特徴量の抽出が不十分であることに起因する予測精度の低下が課題となっている．

Fig. 1.1 : 時系列属性の前処理イメージ

この課題を解決するためには,「時系列属性からの特徴量抽出」を適切に行うこと，および「非時系列属性を考慮した時系列モデルの拡張」により両属性（時系列属性および非時系列属性）を取り扱うことの 2 点が必要であると考える．そこで，系列モデル（ディープラーニング）を用いた応用モデルを構築し予測精度を向上させることを本研究のねらいとする．従来は「時系列属性の特徴量抽出」と「分類器の学習」を独立して行う必要があったが，この応用モデルを用いることで，一つのモデルで「時系列属性の特徴量抽出」と「分類器の学習」を一連の学習として行うことができ，学習パラメータの更新を「時系列属性の特徴量抽出」にまで逆伝播させることで，従来と比較し，より効果的な特徴量抽出を実現することが可能となる．さらに，本研究で構築するモデルは,「非時系列属性を考慮した時系列モデルの拡張」を実現した End-to-End の応用モデルであり，時系列属性の特徴量抽出から最終的な予測までを行うことができるものである．この応用モデルにより，実務上の「担当者間の分析精度や手順のバラつき」,「前処理の複雑化」といった併発する課題の解決にも資すると考える．

上記のような課題解決に向けて，非時系列属性と時系列属性が混在するデータセットとして，株価データを使用し，該当の株価が一定期間のうちに「上昇するか／変わらないか（変動率〇%以内）／下落するか」を予測する分類タスクを設定する．

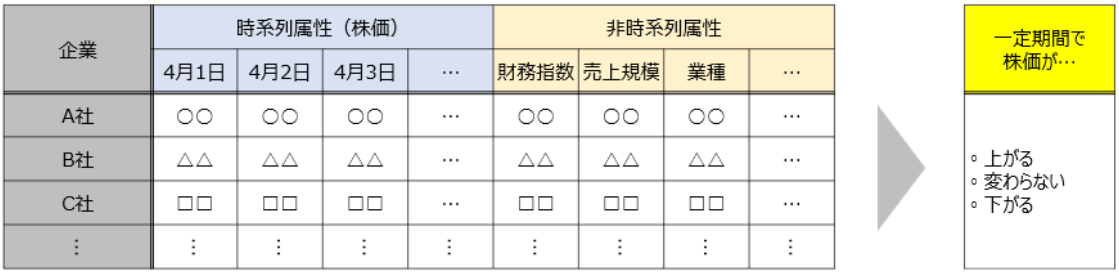

Fig. 1.2：株価の分類予測イメージ

使用するモデルについては，データ形式（本研究ではテーブル形式）への適合性，各種タスク（本研究は分類タスク）における汎用性，系列データの分析精度を総合的に勘案して，Graph Deviation Network（GDN）[1] を採用することとした．

実験では，GDN モデルを用いた応用モデルと従来モデルである勾配ブースティングで分類予測の精度を比較する．勾配ブースティングは複数の決定木を組み合わせて学習するアンサンブル学習の一つであり，2 次のテイラー展開による最適化を用いたものである．代表的な勾配ブースティングとしては，XGBoost[2]，LightGBM[3] やカテゴリカル変数に適した CatBoost[4] が挙げられる．本研究では，設定するタスクの内容から XGBoost，LightGBM でモデル構築をし，予測精度の比較を行う．さらに，損失関数には多クラス分類に用いられる Dice Loss[5] を適用し，従来の損失関数である多クラス版 Cross Entropy との比較を行う．Cross Entropy は予測確率と正解／不正解の組み合わせに着目する損失関数であるが，不均衡データではバランスよく学習できない懸念がある．対して，Dice Loss は F 値に着目しつつ，小さいクラスの正誤へはペナルティを大きく，大きいクラスの正誤へはペナルティを小さくすることで不均衡データにおいてバランスよく学習するよう設計された損失関数である．検証では，Accuracy とマクロ平均 F 値で比較を行う．

本稿の構成は以下のとおりである．第 2 章では，Dice Loss，GDN の研究について紹介する．第 3 章では，本研究で提案する GDN の応用モデルと損失関数 Dice Loss の適用方法について述べる．第 4 章では，実験設定と実験結果について述べる．第 5 章では，考察と結論について述べる．

第 2 章　関連研究

本章では，本研究で用いた要素技術である Dice Loss，GDN およびスケーリングについて説明する．

2.1　Dice Loss

一般的に分類タスクにおいては，Cross Entropy が用いられることが多い．データ X において，インスタンス $x_i \in X$，ラベル $y_i = [y_{i0}, y_{i1}]$，各クラス予測確率 $p_i = [p_{i0}, p_{i1}]$ であるとき，Cross Entropy は以下のように表される：

$$\text{Cross Entropy} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j\in\{0,1\}} y_{ij}\log p_{ij} \tag{2.1}$$

各クラスのサンプル数が大きく異なる不均衡データの問題に対しては，ポジティブサンプルとネガティブサンプルについて，モデル内の損失関数の重み付けを行う重み付け手法があり，その中でも Xiaoya Li ら（2020 年）が Dice Loss [5] を用いて，過去の損失関数を上回る精度を記録している．

Dice Loss は，F 値（適合率（Precision）と再現率（Recall）の調和平均）に着目した不均衡データに対応できる損失関数である．

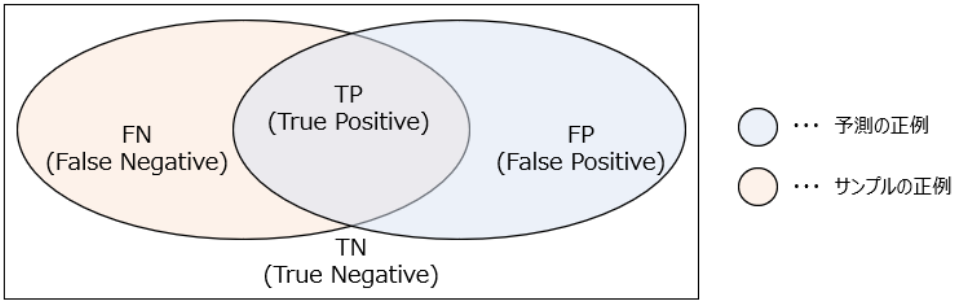

Fig. 2.1 : F 値と Dice Loss の関係

また，F 値は以下のように表される：

$$\mathrm{F1} = \frac{2 \times \mathrm{Precision} \times \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}} = \frac{2 \times \frac{\mathrm{TP}}{\mathrm{TP+FN}} \times \frac{\mathrm{TP}}{\mathrm{TP+FP}}}{\frac{\mathrm{TP}}{\mathrm{TP+FN}} + \frac{\mathrm{TP}}{\mathrm{TP+FP}}} \tag{2.2}$$

これを利用し，分母と分子にハイパーパラメータ γ を加えることで，小さいクラスの誤答にはよりペナルティを大きく，大きいクラスへの誤答にはよりペナルティを少なくし，全体の学習のバランスを良くする仕組みとなっている．

Dice Loss は，以下のように表される：

$$\mathrm{Dice\ \ Loss} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j\in\{0,1\}}\frac{2p_{ij}y_{ij}+\gamma}{p_{ij}+y_{ij}+\gamma} \tag{2.3}$$

ハイパーパラメータ $\gamma = 0$ のとき，Dice Loss は F 値そのものを表し，バランスよく学習できているかどうかの一つの指標となる．なお，γ は 0 以上の値をとり，値を大きくするほど Dice Loss の効果は大きくなる．

2.2 Graph Deviation Network

Graph Deviation Network（GDN）[1] は複数の時系列データからノード間の関係性の構造を学習する Graph Neural Network を利用したアルゴリズムで，センサーの多変量時系列データセットにおける異常検知を目的としたモデルである．本研究ではセンサー時系列を株価時系列に置き換えた多変量時系列の解析に GDN を適用する．GDN は以下の 4 つのアプローチからなる．

1. センサー埋め込み

N 個のノード，時刻 T_{train} までのデータ $\mathbf{s}_{train} = \left[\mathbf{s}_{train}^{(1)}, \ldots, \mathbf{s}_{train}^{(T_{train})}\right]$ を学習用データとする．このとき，時刻 t における N 個のノードの値は，N 次元ベクトル $\mathbf{s}_{train}^{(t)} \in \mathbb{R}^N$ で表される．また，$\mathbf{s}_{train}$ から切り取られる時刻 t における窓幅 w の部分時系列は以下のように表される：

$$\mathbf{x}^{(t)} = [\mathbf{s}^{(t-w)}, \mathbf{s}^{(t-w+1)}, \cdots, \mathbf{s}^{(t-1)}] \tag{2.4}$$

このとき，入力 $\mathbf{x}^{(t)}$ は，各ノードごとに d 次元ベクトル $v_i \in \mathbb{R}^d$ $(i \in 1, 2, \cdots, N)$ に埋め込まれ，モデルの他のパラメータと同時に最適化される．

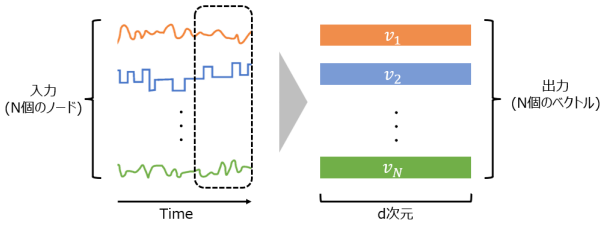

Fig. 2.2：センサー埋め込み

2. グラフ構造の学習

ノード間の依存関係を表すため，有向グラフが用いられる．ノード i の候補関係を $\mathcal{C}_i (\subseteq \{1, 2, \cdots, N\} \backslash \{i\})$ とするとき，ノード i とノード j の類似度 e_{ji} は，コサイン類似度で表される：

$$e_{ji} = \frac{\mathbf{v}_i^\top \mathbf{v}_j}{\| \mathbf{v}_i \| \cdot \| \mathbf{v}_i \|} \text{ for } j \in \mathcal{C}_i \tag{2.5}$$

さらに，計算されたコサイン類似度のうち，類似度の高いノードを抽出するため，隣接行列 A を以下のように定める：

$$A_{ji} = \mathbf{1}\ \{j \in \text{TopK}(\{e_{ki} : k \in \mathcal{C}_i\})\} \tag{2.6}$$

このとき TopK は計算されたコサイン類似度 e_{ji} のうち上位 k 個のインデックスを表す．例えば，TopK=5 とした場合，あるノード i と類似度の高いノード 5 つが選択され，それらをノード i の隣接ノードと仮定する．結果として，時系列データが互いに類似したセンサー間に辺を描くようなグラフが得られる．

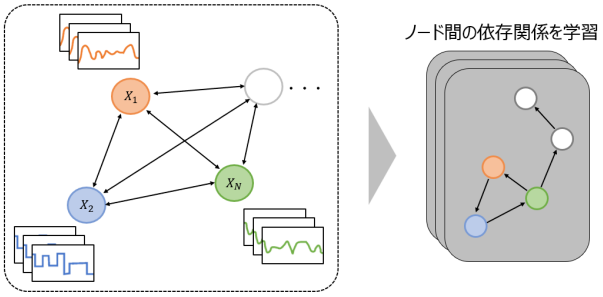

Fig. 2.3 : グラフ構造の学習

3. グラフ Attention に基づく予測

時刻 t におけるノード i の集約された表現 $\mathbf{z}^{(t)}$ は $\mathbf{x}^{(t)}$ を用いて，以下のとおり表される：

$$\mathbf{z}^{(t)} = \mathrm{ReLU}\left(\alpha_{i,i}\mathbf{W}\mathbf{x}_i^{(t)} + \sum_{j\in\mathcal{N}(i)} \alpha_{i,j}\mathbf{W}\mathbf{x}_j^{(t)}\right) \tag{2.7}$$

ただし，ReLU は活性化関数 Rectified Linear Unit，$\alpha_{i,j}$ は注意係数，$\mathcal{N}(i) = \{j|A_{ji} > 0\}$ は隣接行列 A のうち要素が正であるインデックスの集合，$W \in \mathbb{R}^{d\times w}$ は線形変換の表現行列である．なお，ReLU の定義は付録 A.1 を参照されたい．

注意係数 $\alpha_{i,j}$ は以下のように計算される：

$$\mathbf{g}_i^{(t)} = \mathbf{v}_i \oplus \mathbf{W}\mathbf{x}_i^{(t)} \tag{2.8}$$

$$\pi(i,j) = \mathrm{LeakyReLU}(\mathbf{a}^\top(\mathbf{g}_i^{(t)} \oplus \mathbf{g}_j^{(t)})) \tag{2.9}$$

$$\alpha_{i,j} = \frac{\exp(\pi(i,j))}{\sum_{k\in\mathcal{N}(i)\cup\{i\}}\exp(\pi(i,k))} \tag{2.10}$$

ただし，$\oplus$ は Concatenate（2 つのベクトルを連結する操作），LeakyReLU は活性化関数 Leaky Rectified Linear Unit である．なお，LeakyReLU の定義は付録 A.2 を参照されたい．

以上の計算により，時刻 t における集約された表現 $\{\mathbf{z}_1^{(t)}, \cdots, \mathbf{z}_N^{(t)}\}$ と埋め込みベクトル $\{\mathbf{v}_1, \cdots, \mathbf{v}_N\}$ から予測値 $\hat{\mathbf{s}}^{(t)}$ を得る．さらに損失関数は以下のように

計算される：

$$\hat{\mathbf{s}}^{(t)} = f_\theta \left(\left[\mathbf{v}_1 \circ \mathbf{z}_1^{(t)}, \cdots, \mathbf{v}_N \circ \mathbf{z}_N^{(t)} \right] \right) \tag{2.11}$$

$$L_{\mathrm{MSE}} = \frac{1}{T_{\mathrm{train}} - w} \sum_{t=w+1}^{T_{\mathrm{train}}} \left\| \hat{\mathbf{s}}^{(t)} - \mathbf{s}^{(t)} \right\|_2^2 \tag{2.12}$$

ただし，f_θ は活性化関数（本研究では，ReLU を採用），$\circ$ はアダマール積を表す．

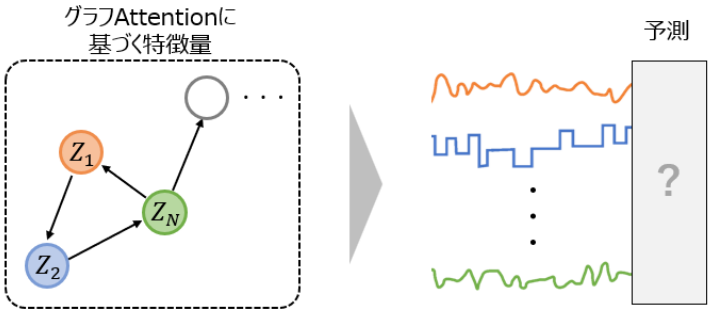

Fig. 2.4：グラフ Attention に基づく予測

4. グラフ逸脱度のスコアリング

時刻 t でのノード i の異常を示すスコア Err および正規化表現 $a_i(t)$ を Err の時間全体の中央値 $\widetilde{\mu}_i$ および四分位範囲 $\widetilde{\sigma}_i$ を用いて表し，その最大値 $A(t)$ を計算する：

$$\mathrm{Err}_i(t) = \left|\hat{\mathbf{s}}^{(t)} - \mathbf{s}^{(t)}\right| \tag{2.13}$$

$$a_i(t) = \frac{\mathrm{Err}_i(t) - \widetilde{\mu}_i}{\widetilde{\sigma}_i} \tag{2.14}$$

$$A(t) = \max_i a_i(t) \tag{2.15}$$

$A(t)$ が固定された閾値を超えた場合に，時刻 t が異常としてラベル付けされる．

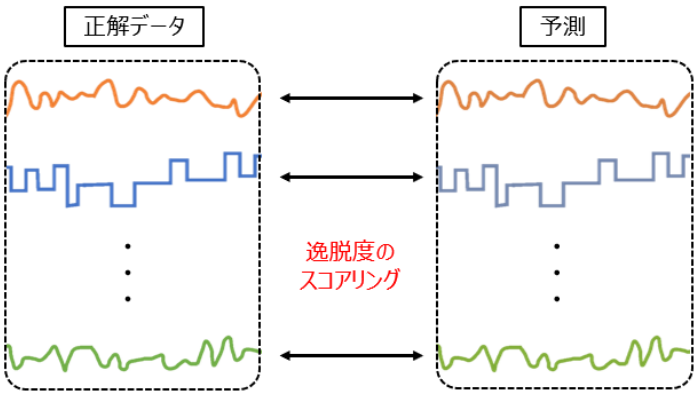

Fig. 2.5 : グラフ逸脱度のスコアリング

2.3 スケーリング

GDN 機構では，ノードから部分ノードを切り取り，入力データとして投入することになるが，この際にスケーリングを行う．Deep AR[6] 内で述べられている Scale Handling を適用する．切り取られた部分ノード $\mathbf{x}^{(t)}$ をその平均値 $\bar{\mathbf{x}}^{(t)}$ で除算したのちに，GDN 機構へ投入する．さらに GDN 機構の出力で，平均値 $\bar{\mathbf{x}}^{(t)}$ を乗算し，逆のスケーリングを行う．これにより，ネットワークの入力層で適切な範囲にスケーリングされ，出力層で予測値を元の大きさに戻す操作を行うことになる．

第 3 章　提案モデル

本章では，GDN の回帰予測の機能を応用し，時系列属性および非時系列属性の混在データに対応した応用モデルの提案をする．

3.1　提案モデルのねらい

本研究での目的は，時系列属性および非時系列属性の混在データにおいて，分類予測を適切に行うことである．このためには，以下の条件を満たすモデル構築が必要となる：

1. 時系列属性を適切にベクトル化できること
2. 複数時系列に対応し，時系列間の相互依存を学習できること
3. 非時系列属性と統合し，最終的に分類タスクを構築できること
4. 逆伝播により可能な限り多くの学習パラメータを更新できること

以上の条件を満たすアルゴリズムとして，GDN が応用できると考えた．GDN を構成する要素には，時系列属性の畳み込み処理および時系列間のコサイン類似度を計算する構造がある．このモデルの最終層に分類器を適用し，Pre-training と Fine-tuning を応用することにより，時系列属性を適切に処理しながら非時系列属性と統合し，最終的に最適に分類予測を行うモデルが構築できると考えた．

3.2　提案モデル

本研究で構築する GDN 応用モデルを図式化したものを Fig.3.1 に示す.

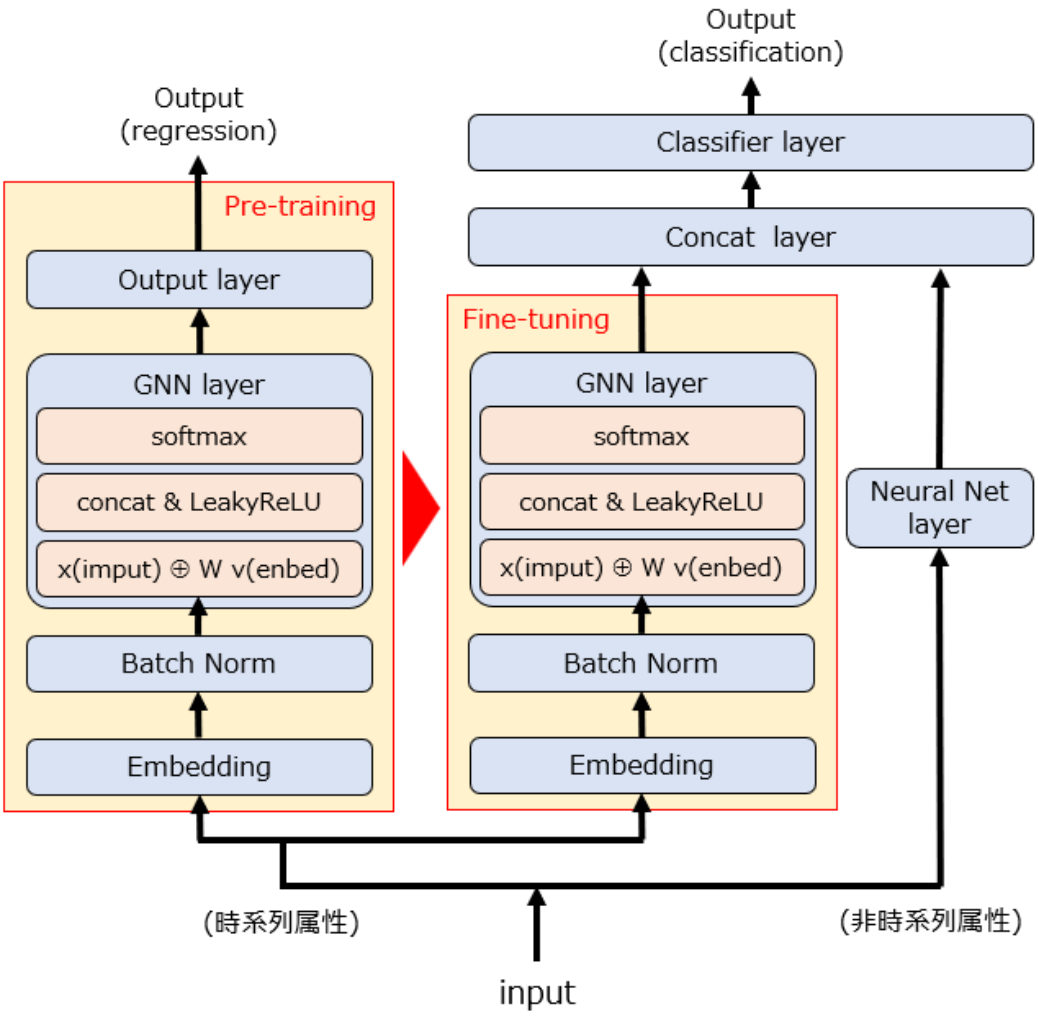

Fig. 3.1 : GDN 提案モデル

GDN 応用モデルが分類予測をするまでの手順は以下のとおりである：

1. データ入力

　時系列属性および非時系列属性の混在データを入力データとする．データ入力後，時系列属性は非時系列属性と切り離され，GDN 機構の入力となる．

2. Embedding 層，GNN 層，Output 層

これらの層は，GNN モデルの中核となる機構である．時系列属性から一定の窓幅の部分時系列に分割し，投入される．Embedding 層では，入力された時系列属性を指定された次元数に埋め込み，GNN 層では，複数の時系列間の依存関係を学習する．Output 層では，Embedding 層で得られた埋込ベクトルと GNN 層から得られたノード表現から回帰予測を行う．

3. Pre-training

部分時系列を投入し，時点の値を予測する回帰タスクの学習を繰り返し，部分時系列をスライドさせながら最適化する．

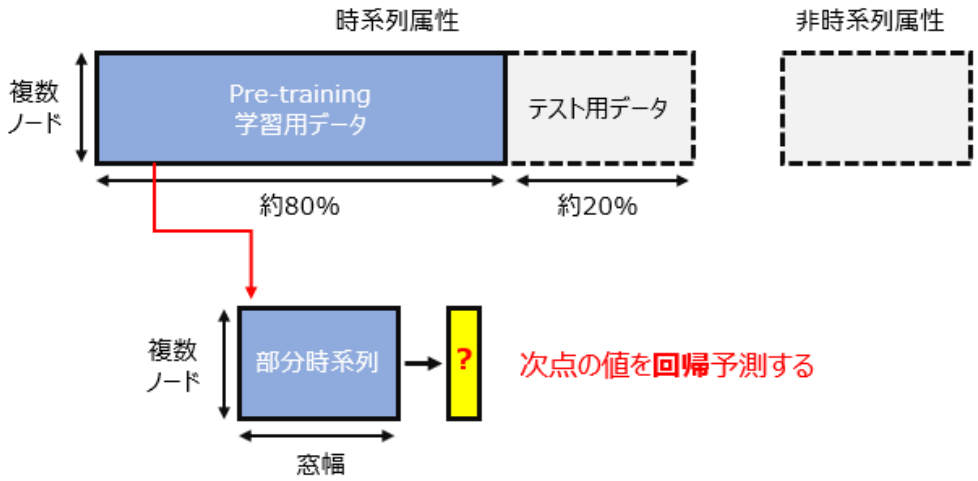

Fig. 3.2 : Pre-training

4. Concat 層，Classifier 層

Pre-training から得られた時系列属性の埋込ベクトルと非時系列属性を Concat 層で統合（Concatenate）し，Classifier 層で分類予測を行う．

5. Fine-tuning

Concat 層，Classifier 層による学習を繰り返し，分類予測を最適化する．このとき，Pre-training で学習済の Embedding 層に関係するパラメータを再度更新し，それ以外のパラメータはあらたに学習する．なお，Fine-tuning ではテスト用データでの精度を向上させるため Pre-training で用いた学習用データのうち直近部分のみを再利用する．

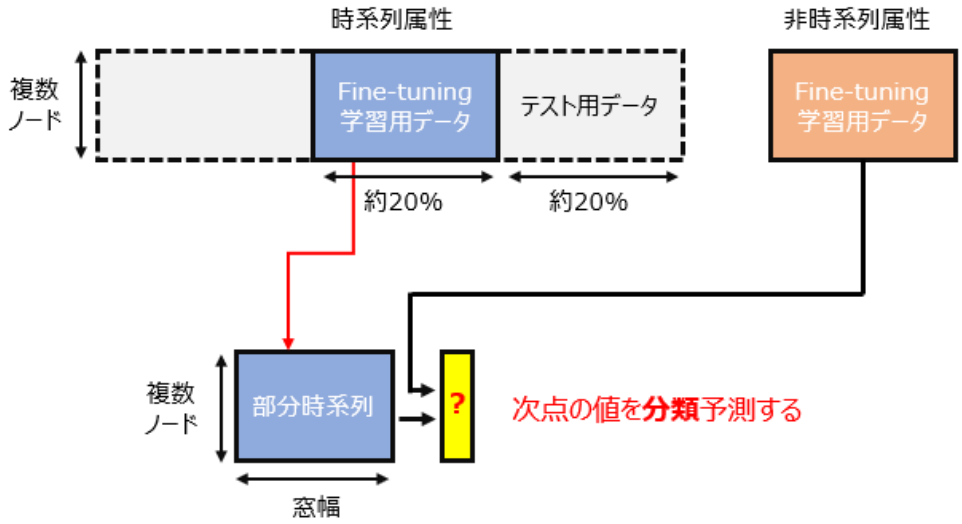

Fig. 3.3 : Fine-tuning

以上の 1〜5 の手順を経て，入力データの次点の株価の騰落の分類予測を行う．

3.3 Dice Loss の適用

分類タスクにおいては，最も基本的な損失関数として Cross Entropy が設定されることが多い．サンプル数 N，クラス数 K のデータについて出力 z_{ij}，確率 p_{ij}，教師データ y_{ij} とすると，Softmax 関数および多クラス版に拡張した Cross Entropy は以下のとおり表される：

$$p_{ij} = \mathrm{Softmax}(z_{ij}) = \frac{\exp(x_{ij})}{\sum_{k=1}^{K} \exp(x_{ik})} \tag{3.1}$$

$$L = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{K} y_{ij} \log(p_{ij}) \tag{3.2}$$

また，本来 Dice Loss は 2 クラス分類における不均衡データへの対処を想定したものであるが，本研究への応用として多クラス版の Dice Loss に拡張する．ハイパーパラメータを γ とすると，多クラス版 Dice Loss は以下のとおり表される：

$$L = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{K} \left(\frac{2\, p_{ij}\, y_{ij} + \gamma}{p_{ij} + y_{ij} + \gamma} \right) \tag{3.3}$$

第 4 章　評価実験

本章では，実験用データの取得及び加工手順を示し使用するデータセットの概要を説明する．次に実験設定と実験用の使用モデル（提案モデル 4 パターンおよび従来モデル 2 パターン）について説明する．最後に，実験（分類タスク）の結果について説明する．

4.1　データセット

本研究では，Yahoo!ファイナンスから取得したデータを実験用のデータとして用いた．yfinance というライブラリを用いることで API により東京証券取引所に上場する 3700 社以上の企業の株価や財務指標などのデータが取得できる．

十分な株価時系列データを確保するため，東証上場企業（3700 社以上）のうち，以下の条件を満たすものを今回対象とする企業として選択した：

- yfinance によりデータ取得が可能であるもの
- 2000 年から 2021 年までの間で 3000 日以上の株価推移データを有するもの

以上の条件を満たす全 2730 社のうち，ランダムに抽出した 100 社を対象とし，時系列属性および非時系列属性となり得るデータを以下のとおり抽出した：

時系列属性に利用するデータ

- 2000 年から 2021 年までの日次修正株価[1]

非時系列属性に利用するデータ

- 財務指標（ROA，ROE，売上高，売上高経常利益率，自己資本比率）
- 属性（業種，上場区分，会社規模）

[1]株式分割や配当に伴う権利落ち等を考慮した株価

なお，非時系列属性[2]に利用するデータのうち，財務指標は直近 2 年分および直近 2 四半期分の合計 4 断面を用いる．

4.2　データセットに対する前処理

時系列属性および非時系列属性に用いるデータは，それぞれ以下のように前処理を行った．

時系列属性に利用するデータ

- 説明変数としての影響力を揃えるため，企業ごとに標準化
- 傾向を見やすくするため 5 日ごとの平均をとる（データサイズは 1/5）

非時系列属性に利用するデータ

- 属性（業種・上場区分・会社規模）をダミー変数化
- 財務指標と属性（業種・上場区分・会社規模）を統合
- 財務指標における欠損値をホットデック法により補完
- すべての非時系列属性について属性ごとに標準化

以上の前処理を経て，時系列属性と非時系列属性の混在データを用意した．最終的にサンプル数は 100 社，時系列属性は 522 時点，非時系列属性は 36 属性とした．なお，以下は標準化加工のイメージである：

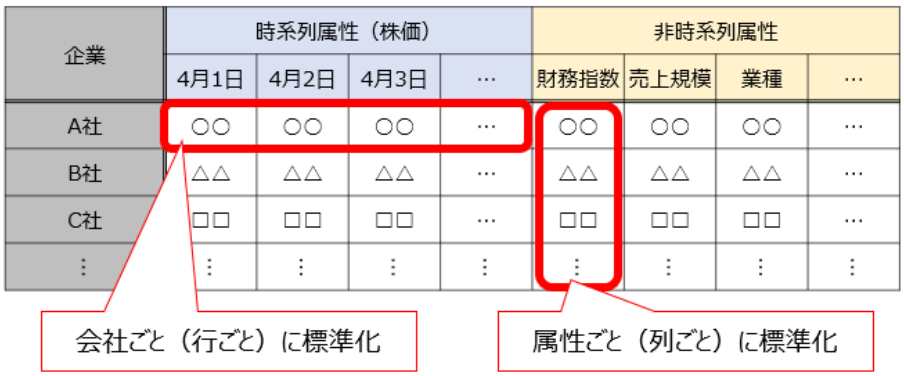

Fig. 4.1 : 標準化加工イメージ

[2]厳密には，低頻度の時系列属性を含むが，本論文では便宜上，非時系列属性と表記する．

4.3　実験設定と使用モデル

Fig.4.1 で用意したデータセットを用いて実験を行う．実験では，一定量の学習用データで学習を行ったあと，株価の騰落（「上昇するか／変わらないか（変動率〇%以内）／下落するか」）を予測する分類タスクを設定する．本研究における GDN 応用モデルと従来手法である勾配ブースティングの比較を行うが，より深く考察するため以下の 6 つのモデルを用意した：

GDN 応用モデル（End-to-End モデル）

Pre-training では MSE（平均二乗誤差）を用いて，時系列属性（株価）の回帰予測を行う．その Pre-training で得られた学習パラメータを Fine-tuning 時に初期値として与え，分類用の各種パラメータと同時に再学習を行う．モデルの詳細は，Fig.3.1 を参照されたい．

GDN プラグインモデル

GDN 応用モデルは，時系列属性を Pre-training および Fine-tuning で合計 2 回にわたり埋込表現が学習されるが，GDN プラグインモデルは Pre-training で調整した学習パラメータを Fine-tuning 時に固定するものである．これにより Fine-tuning 時の分類のための学習パラメータを優先的に学習する．

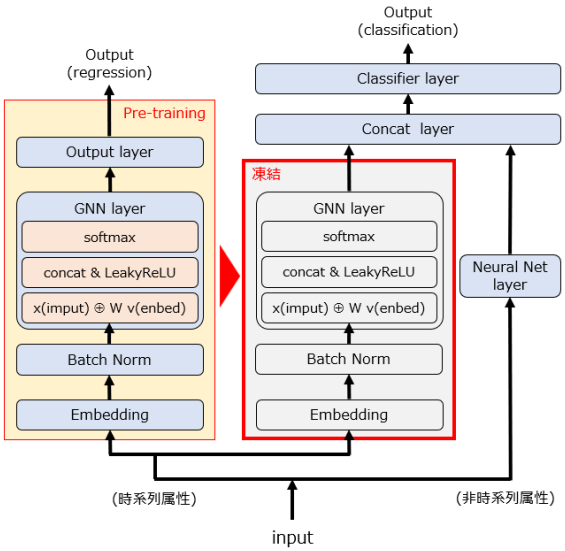

Fig. 4.2 : GDN プラグインモデル

GDN モデル（時系列属性のみ）

GDN 応用モデルに時系列属性のみを投入したモデルであり，時系列属性のみでどれだけ予測性能に寄与できるかを確認するためのものである．

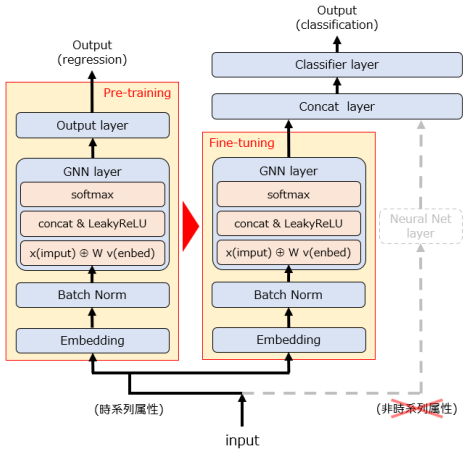

Fig. 4.3 : GDN モデル（時系列属性のみ）

GDN モデル（非時系列属性のみ）

GDN 応用モデルに非時系列属性のみを投入したモデルであり，非時系列属性のみでどれだけ予測性能に寄与できるかを確認するためのものである．

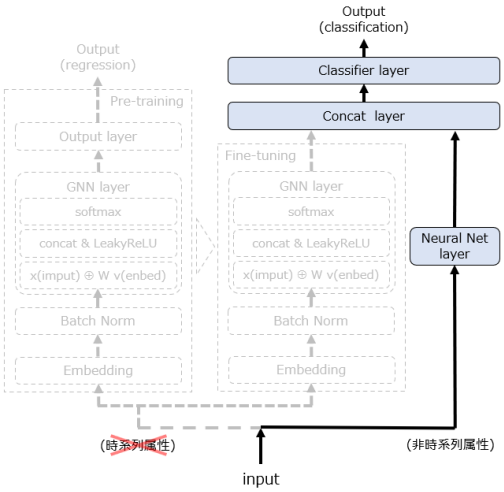

Fig. 4.4 : GDN モデル（非時系列属性のみ）

従来手法：XGBoost

従来手法の勾配ブースティングのうち，XGBoost による分類モデルである．上記 4 つのモデルと可能な限り条件を揃えるため，時系列属性の前処理に GDN を利用する．多変量の時系列属性を GDN に投入し，その回帰予測で得られた埋込ベクトルを抽出する．そのベクトルを時系列属性の前処理とみなし，非時系列属性と統合し，XGBoost の分類モデルに投入する．なお，XGBoost の詳細については付録.B.1 を参照されたい．

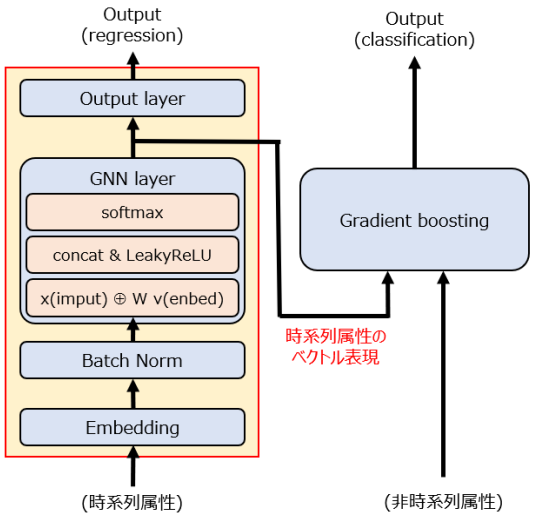

Fig. 4.5 : GDN による時系列属性の前処理イメージ

従来手法：LightGBM

モデル構築方法は XGBoost と同様．なお，LightGBM の詳細については付録.B.2 を参照されたい．

4.4 評価設定

本実験では，3 つのクラスに分類する多値分類のタスクを設定する．N：データ数，K：クラス数とし，クラス $k(k = 1, \cdots, K)$ を予測したときを Positive_k，クラス k 以外を予測したときを Negative_k，それぞれの真偽を TP_k（Positive_k が真である件数），TN_k（Negative_k が真である件数），FP_k（Positive_k が真である件数），FN_k（Negative_k が真である件数）とする．このとき以下の 2 つの指標によって予測精度を測る：

Accuracy

テスト用データに対する全体の正解率であり，もっとも単純な指標の一つである．

$$\text{Accuracy} = \frac{1}{N}\sum_{k=1}^{K}\text{TP}_k \tag{4.1}$$

マクロ平均 F 値

各クラスの Precision（適合率），Reacll（再現率），F1（F 値）を算出したのちにクラスごとの各値を平均化したものであり，クラス不均衡の場合などにバランスよく分類予測をできているかの指標となる．

$$\text{Precision}_k = \frac{\text{TP}_k}{\text{TP}_k + \text{FP}_k} \tag{4.2}$$

$$\text{Recall}_k = \frac{\text{TP}_k}{\text{TP}_k + \text{FN}_k} \tag{4.3}$$

$$\text{macro-F1} = \frac{1}{K}\sum_{k=1}^{K}\frac{2 \cdot \text{Precision}_k \cdot \text{Recall}_k}{\text{Precision}_k + \text{Recall}_k} \tag{4.4}$$

4.5　実験手順

今回行う実験の手順は以下のとおりである：

1. データセット分割

   データセットには時系列属性と非時系列属性があるが，このうち時系列属性は学習用／検証用／テスト用の 3 つに分割する．分割イメージは以下のとおりである：

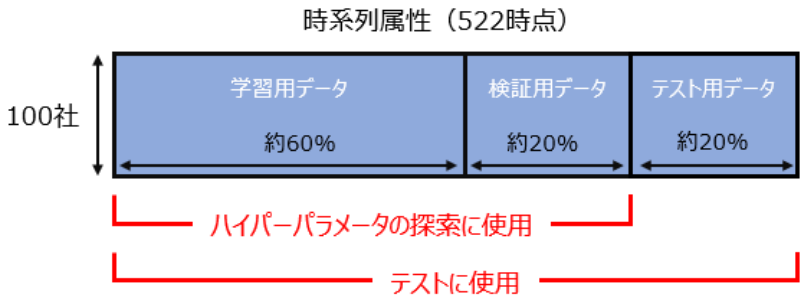

Fig. 4.6 : 時系列属性のデータセット分割方法

2. ハイパーパラメータ探索

   ハイパーパラメータの探索では，optuna（ベイズ最適化を行うライブラリ）を利用し，学習用データおよび検証用データを用いる．なお，時系列属性の学習用データと非時系列属性から分類タスクの学習を行い，検証用データによって精度の検証を行うことによって最適なハイパーパラメータを選択する．

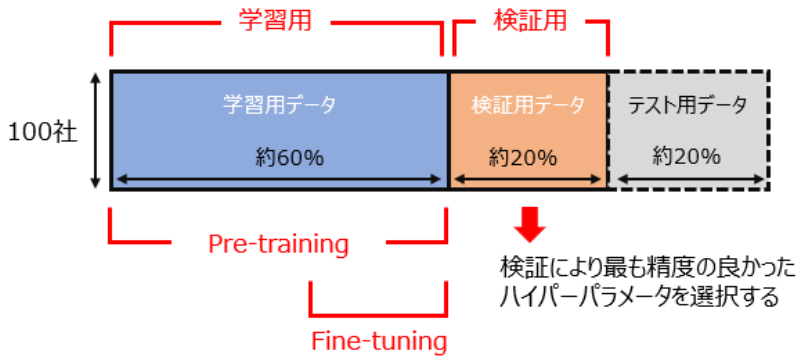

Fig. 4.7 : ハイパーパラメータ探索時のデータセット分割方法

3. テスト

最終的なテストでは，選択したハイパーパラメータを固定したうえで，学習用データと検証用データを学習に用い，テスト用データで精度の計測を行う．なお，テストでは Pre-training（回帰タスク）・Fine-tuning（分類タスク）・損失関数 Dice Loss の適用における挙動の確認を行う．

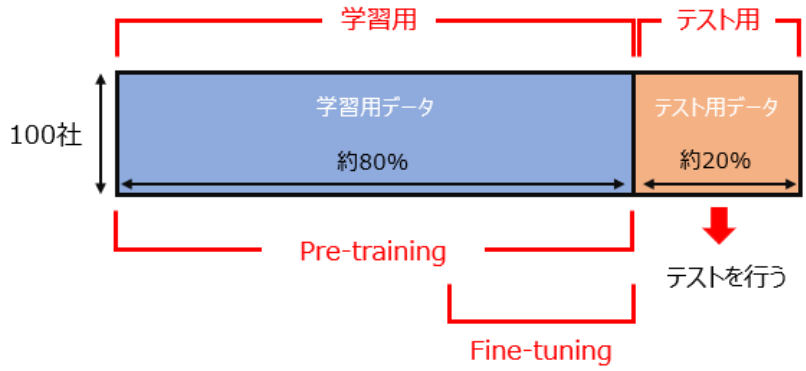

Fig. 4.8 : テスト時のデータセット分割方法

4.6 実験結果

ハイパーパラメータ探索結果

時系列属性のテスト用データ以外を使用しハイパーパラメータの探索を行った．GDN 応用モデルのうち，予測精度への影響度が高い可能性のある以下のハイパーパラメータのみ探索した：

- slide-win
  時系列属性から部分時系列を切り取る時の窓幅
- dim
  時系列属性から切り取られた部分時系列を埋め込むときの次元数
- net-hidden-size
  時系列属性と非時系列属性を統合したのちに投入されるニューラルネットワーク中間層の次元数
- dice-gamma
  損失関数 Dice Loss 内のハイパーパラメータ γ

ハイパーパラメータの探索については，以下のとおり 2 段階で行った：

1. slide-win，dim，net-hidden-size の探索
   GDN 応用モデルの損失関数を Cross Entropy に設定し，評価指標をマクロ平均 F 値の最大化とし，slide-win，dim，net-hidden-size の 3 つのハイパーパラメータのみ探索を行う．
2. dice-gamma の探索
   上記で決定した最適なハイパーパラメータ（slide-win，dim，net-hidden-size）の値を固定し，損失関数を Dice Loss にかえ，最適な dice-gamma の探索を行う．

はじめに，最適な slide-win，dim，net-hidden-size の値を選択するため，200 回の試行を行った．探索の結果（性能が良かった上位 5 試行）は以下のとおりである：

Table 4.1 : slide-win，dim，net-hidden-size 探索結果

slide-win	dim	net-hidden-size	macro-F1
(3 - 20)	(10 - 100)	(5 - 100)	—
19	70	73	0.3736
13	72	51	0.3612
16	46	84	0.3569
17	57	45	0.3441
18	47	56	0.3439

以上の結果から，slide-win=19，dim=70，net-hidden-size=73 に設定した．

次に，損失関数を Dice Loss に設定したときの最適な dice-gamma の値を選択するため，dice-gamma の値の探索範囲を 0 から 2.0 とし試行を行った．探索の結果（性能が良かった上位 5 試行）は以下のとおりである：

Table 4.2 : dice-gamma 探索結果

dice-gamma	macro-F1
(0 - 2.0)	—
0.9	0.4103
1.0	0.3942
1.7	0.3887
1.9	0.3856
1.5	0.3854

この結果から，損失関数 Dice Loss 内のハイパーパラメータは $\gamma = 0.9$ に設定する．

テスト（Pre-training）

GDN 応用モデルの Pre-training では，回帰予測のためにコサイン類似度の高いノードを複数選択し学習に利用しているが，その選択するノード数を TopK というハイパーパラメータで調整している．周辺ノードの特徴の取り込みが結果に与える影響を確認するため，TopK=0 の場合と TopK=5 の場合で比較した．なお，TopK=0 は周辺ノードの特徴を取り込まない（すなわち，単変量の自己回帰），TopK=5 は類似性の高い 5 つのノードの特徴を取り込むことを意味している．検証した結果は以下のとおりである：

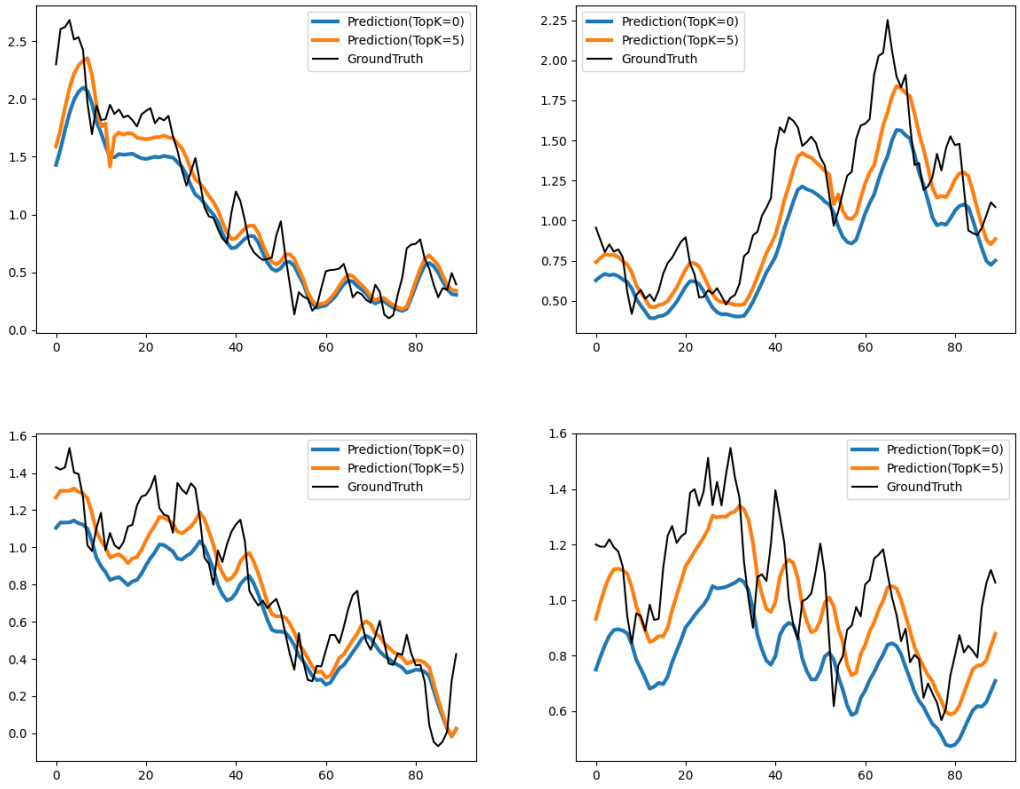

Fig. 4.9 : Pre-training での回帰予測（先頭 4 社分）

TopK=5 の場合のほうが，より正解データに近い予測をしていることが分かる．これは，類似度の高い時系列の特徴を回帰学習に取り入れることで，効率的に回帰予測できているためである．この結果より，GDN 応用モデルにおける Pre-training では，グラフ構造を用いることにより株価の時系列属性の特徴量を適切に学習していると判断できる．

テスト（Fine-tuning）

正解データの各クラスのサンプル数をほぼ均等（「上昇する」: 33.81 %／「変わらない（変動率 3 %以内）」: 33.22 %／「下落する」: 32.96 %）に設定し，損失関数を初期設定の Cross Entropy とした．以上の条件で，株価データセットを 6 パターンのモデルに投入し，12 回の試行（テスト）を行った．Accuracy およびマクロ平均 F 値の 12 回の試行の平均値と標準偏差の結果は以下のとおりである：

Table 4.3 : GDN 応用モデル精度

モデル	Accuracy		macro-F1	
	平均	標準偏差	平均	標準偏差
GDN 応用モデル	0.3674	0.0133	0.3466	0.0243
GDN プラグインモデル	0.3581	0.0090	0.3316	0.0162
GDN モデル（時系列属性のみ）	0.3619	0.0183	0.3345	0.0272
GDN モデル（非時系列属性のみ）	0.3504	0.0022	0.3157	0.0052
XGBoost	0.3523	0.0076	0.3078	0.0131
LightGBM	0.3526	0.0047	0.3117	0.0133

今回の提案手法である GDN 応用モデルが Accuracy およびマクロ平均 F 値で僅かに良い結果となった．続いて GDN モデル（時系列属性のみ）が良い結果となった．GDN モデル（非時系列属性のみ）を除いて，従来手法（XGBoost, LightGBM）より高い精度となった．

また，GDN プラグインモデルでは，Pre-training の学習パラメータを凍結することにより後続の Fine-tuning で分類予測のための学習パラメータを優先的に学習するモデルであるが，GDN 応用モデルより精度が低い結果となった．

テスト（Dice Loss の適用）

今回の提案モデルである GDN 応用モデルについて，クラス不均衡（「上昇する」：20.92 %／「変わらない（変動率 7 %以内）」：59.37 %／「下落する」：19.71 %）を設定し，挙動を確認した．上記のように，正解データの各クラス間のサンプル数の比率に差を設け，損失関数を Cross Entropy と Dice Loss で設定し，実験を行った．それぞれの精度は以下のとおりである：

Table 4.4: Dice Loss 適用時の精度比較

GDN 応用モデルの損失関数	Accuracy		macro-F1	
	平均	標準偏差	平均	標準偏差
Cross Entropy	0.4844	0.0853	0.3540	0.0388
Dice Loss（$\gamma = 0.9$）	0.5350	0.0292	0.3743	0.0293

結果としては，Dice Loss を適用した方が，Accuracy およびマクロ平均 F 値の両観点において，大きく精度改善された．

第 5 章　結論

5.1　考察

テスト（Fine-tuning）においては，GDN 応用モデルによる予測精度の向上を目的とし，株価の騰落を予測する分類タスクを設定した．Pre-training（回帰タスク）の段階では，時系列属性（株価）の部分時系列から次点の株価を予測できていることが，回帰誤差のグラフから判断できる．特に部分時系列に分割し GDN 機構に投入する際に平均化によるスケーリングを行ったことが回帰タスクにおける MSE（平均二乗誤差）の低減に大きく寄与したと考えられる．また，Fine-tuning（分類タスク）では，Accuracy およびマクロ平均 F 値において従来手法（XGBoost，LightGBM）の精度を上回り，特定のクラスに偏らない分類予測ができることを確認した．GDN 応用モデルは，Pre-training と Fine-tuning からなる構造をしており，時系列属性の特徴量抽出を 2 回に分けて行っている．特に今回のタスク設定（株価予測）においては時系列属性の特徴量抽出が重要な説明変数と捉え，Pre-training で学習したパラメータを Fine-tuning で再学習し時系列属性の前処理まで逆伝播させている．勾配ブースティングでは，この逆伝播が時系列属性の特徴量抽出まで及ばないため今回のような精度の違いが表れたものと推察する．

また，テスト（Dice Loss の適用）については，クラスのサンプル数をおよそ「1:3:1」としクラス不均衡を設定した．このように分類タスクの難易度を上げ，損失関数を Cross Entropy に設定した状態では，Accuracy およびマクロ平均 F 値は期待どおりの値とはならなかった．Cross Entropy に替え Dice Loss を設定したところ，Accuracy およびマクロ平均 F 値は大幅に改善された．ただし，今回の精度改善はクラス不均衡の程度が小さい場合に限るものであり，クラス不均衡の程度を大きくした場合は，より精度が悪化すると考えられる．そのような極端なクラス不均衡の状況下では，Dice Loss では対処できない可能性もある．

5.2 今後の課題

今回の実験では，GDN 応用モデルで従来手法の勾配ブースティングの精度を上回り，Dice Loss による不均衡データへの対処も一定水準の成果を確認できた．しかしながら，期待した精度には届いていない部分もあり，以下のような課題が残っている．

非時系列属性の説明力

今回のテスト（Fine-tuning）における GDN モデル（非時系列属性のみ）の結果から分かるように非時系列属性のみでの分類予測は非常に困難であった．そもそも yfinance から十分なデータを取得することができず，4 断面の財務データと企業属性だけでは，タイムリーな株価の変化に対応できなかった．例えば「従業員数」「本社所在地」「SGDs 評価」などの比較的重要視されるようなデータが確保できれば，非時系列属性の説明力は向上させることができると考える．

時系列属性と非時系列属性の統合

時系列属性と非時系列属性の統合によりモデルの精度が相乗効果的に向上すると期待していたが，今回の実験ではこれが十分に実現できなかった．GDN 応用モデル内では，比較的シンプルなニューラルネットワークでその実装をしていたため，さらに工夫すれば精度改善の余地はあると考える．

不均衡データへの対処

テスト（Dice Loss の適用）では，程度の小さいクラス不均衡を設定し，分類タスクの実験を行った．これより程度の大きいクラス不均衡を設定した場合には，学習自体がままならない状況となった．原著論文の中では，程度の大きいクラス不均衡でも成果を出しているため，Dice Loss による精度改善の余地はあると考えている．今回のような株価予測のような難しいタスク設定ではなく，比較的予測しやすいタスクを設定したうえでの実験も行う必要がある．

他手法との比較

本研究では，時系列属性の特徴量抽出を GDN を用いて行ったが，AR モデルなどの時系列分析モデルを用いる手法と比較することも考えられる．具体的には時系列属性を用いて回帰タスクで次点の値を学習し，そこで得られた学習パラメータと非時系列属性と統合する実験である．

5.3 結論

本研究では，従来モデルより僅かに精度（Accuracy およびマクロ平均 F 値）が向上し，程度の小さいクラス不均衡における損失関数 Dice Loss の一定水準の精度改善の成果が得られた．今回は，株価の騰落予測の分類タスクを行ったが，このモデルを用いることで，例えば最適な商品を顧客にレコメンドするという分類予測にも応用できると考えている．銀行顧客等の場合，顧客の属性の他に預金残高履歴や入出金履歴を保有している．従来であれば，主に非時系列属性（顧客属性）を中心に説明変数を設定し分類予測を行うことが多かったが，時系列属性も統合し予測をすることで，より精度の高い予測ができると考えられ，今回提案した GDN 応用モデルを異なるタスク設定下で活用できる可能性もある．

以上より，今回提案した GDN 応用モデルは従来から実務現場の課題として存在している予測精度のロスへの対処として，一定程度の効果があるものと考える．さらに，時系列属性の前処理の際の「担当者間の分析精度や手順のバラつき」,「前処理の複雑化」といった課題に対しても副次的に解決に資するものと思料する．

ただし，本研究についてはまだ精度改善の余地はあり，改良を重ねればより良いモデルを構築することは可能であると考えている．分類器に勾配ブースティングを応用したパターンやグラフニューラルネットワークに GDN 以外のものを適用したパターンなども検討できる．

謝辞

本研究の機会を与えてくださった株式会社ひろぎんホールディングスおよび株式会社広島銀行の関連部署の方々，ご指導を賜りました広島大学 先進理工系科学研究科 情報科学プログラムの関係者の方々に深く感謝の意を表します．また輪読会でご指導を頂きました広島大学 先進理工系科学研究科 情報科学プログラム Daniel Andrade 准教授，副査として面談などでご指導いただきました人間社会科学研究科 経済学プログラム 山田宏教授，日常の議論を通じて多くの知識や示唆を頂いた広島大学データ解析・モデリング研究室の皆様に感謝致します．

最後に，とりわけ研究内容に共感し 2 年間研究に携わって頂いた江口浩二教授，ビジネス観点からの助言と総合的なサポートを頂いた株式会社ひろぎんホールディングス デジタルイノベーション部 大江拓真 DX 戦略企画担当調査役，生活面をサポートしてくださった家族に感謝の意を表し，これをもって謝辞とさせていただきます．

参 考 文 献

[1] Ailin Deng, Bryan Hooi, "Graph Neural Network-Based Anomaly Detection in Multivariate Time Series", *arXiv preprint arXiv:2106.06947* (2021)

[2] Tianqi Chen, Carlos Guestrin, "XGBoost: A Scalable Tree Boosting System", *arXiv preprint arXiv:1603.02754* (2016)

[3] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, Tie-Yan Liu, "LightGBM: A Highly Efficient Gradient Boosting", *Advances in Neural Information Processing Systems* (2017)

[4] Anna Veronika Dorogush, Vasily Ershov, Andrey Gulin, "CatBoost: gradient boosting with categorical features support", *arXiv preprint arXiv:1810.11363* (2018)

[5] Xiaoya Li, Xiaofei Sun, Yuxian Meng, Junjun Liang, Fei Wu, Jiwei Li, "Dice Loss for Data-imbalanced NLP Tasks", *arXiv preprint arXiv:1911.02855* (2020)

[6] David Salinas, Valentin Flunkert, Jan Gasthaus, "DeepAR: Probabilistic Forecasting with Autoregressive Recurrent Networks", *arXiv preprint arXiv:1704.04110* (2017)

付 録A　活性化関数

活性化関数とは，ニューラルネットワークにおいて，ある入力を活性化させて有効な出力に変換することを目的とした関数である．本論文における重要な活性化関数であるReLUとLeakyReLUの定義を以下のとおり示す.

A.1　ReLU：Rectified Linear Unit

入力xに対し，活性化関数ReLUは以下のとおり定義される：

$$\mathrm{ReLU}(x) = \begin{cases} 0, & \text{if } x < 0 \\ x, & \text{if } x \geqq 0 \end{cases} \tag{A.1}$$

ReLUは微分しても勾配消失が起こりにくい特徴がある.

A.2　LeakyReLU：Leaky Rectified Linear Unit

入力xに対し，活性化関数LeakyReLUは以下のとおり定義される：

$$\mathrm{LeakyReLU}(x) = \begin{cases} -0.01x, & \text{if } x < 0 \\ x, & \text{if } x \geqq 0 \end{cases} \tag{A.2}$$

入力xが0より小さい範囲でもわずかに値を持つことから，ReLU関数と違ってxが負の値でも更新されるという特徴がある.

付 録B　勾配ブースティング

勾配ブースティング手法は，複数の決定木を組み合わせて学習することで高い精度を実現するアンサンブル学習の一つであり，小規模の学習器を追加していくことで，直列的につなぎ精度を向上させるアルゴリズムである．なお，学習器の追加によりモデルの予測値が学習用データに合致していき過学習を引き起こすことがあるが，これを防ぐためにしばしば正則化項が用いられる．

いずれもテーブル形式データを扱うタスクにおいて高く評価されており，計算負荷も軽く一定の精度で分析を行うことができることから，実データを利用した分析の実務現場でも多く用いられている．

B.1　XGBoost

勾配ブースティングでは，サンプル数 n，データ次元数 m からなるデータ $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}$ $(|\mathcal{D}| = n, \mathbf{x}_i \in \mathbb{R}^m, y_i \in \mathbb{R})$ において，加法的関数を用いて以下のように予測 $\hat{y}_i$ を出力する：

$$\hat{y}_i = \phi(\mathbf{x}_i) = \sum_{k=1}^{K} f_k(\mathbf{x}_i),\ f_k \in \mathcal{F} \tag{B.1}$$

ただし，$\mathcal{F} = \{f(\mathbf{x}) = w_{q(\mathbf{x})}\}$ $(q : \mathbb{R}^m \to T, w \in \mathbb{R}^T)$ とし，最終的な出力 $\hat{y}_i$ はそれぞれの木の重みを総和したものになる．また，損失関数 l には残差二乗平均に類似した微分可能な凸関数を用いて，各サンプルに対して予測値と教師データとの間の残差を l で表現し，すべてのサンプルに対して総和をとる．さらに，それぞれの木に対する正則化項 Ω の寄与を総和する．正則化項 Ω はさらに葉のノード数 T と重み w の L2 ノルム $\|w\|_2^2$ に分解できる．T が大きくなるほど損失関数が大きくなるため，木の構造を複雑化を抑制する効果がある．一方，$\|w\|_2^2$ は過学習を抑制する効果がある．このとき，損失関数 $\mathcal{L}(\phi)$ は正則化項のバランスを調整するハイパーパラメータ γ, λ を用い

て以下のように表される：

$$\mathcal{L}(\phi) = \sum_i l\left(\hat{y}_i, y_i\right) + \sum_k \Omega(f_k) \tag{B.2}$$

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \|w\|_2^2 \tag{B.3}$$

ここから，XGBoost のモデルを決定するために，以下の手順が必要となる．

1. 木の構造を決定
2. 各葉のノードの出力（$f_k(\mathbf{x}_i), w_q$）を決定

また，$f_k(\mathbf{x}_i)$ を更新するため，初期状態 $(t = 1)$ で生成した単一の決定木に $t = 2$ の決定木を追加していき，これを繰り返していくことにより，以下の式が成り立つ：

$$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + f_t(\mathbf{x}_i) \tag{B.4}$$

このとき，漸化式を用いると損失関数は以下のように表される：

$$\mathcal{L}^{(t)} = \sum_i^n l(y_i^{(t)}, \hat{y}_i^{(t)}) + \Omega(f_k) \tag{B.5}$$

$$= \sum_i^n l(y_i^{(t)}, \hat{y}_i^{(t-1)} + f_t(\mathbf{x}_i)) + \Omega(f_k) \tag{B.6}$$

ここで得られた損失関数について，$\mathbf{x}_i$ の周辺のテイラー展開による 2 次の項までの近似での最適化により，最終的に以下の式を得る：

$$\bar{\mathcal{L}}^{(t)} = \sum_i^n g_i f_t(\mathbf{x}_i) + \frac{1}{2} h_i f_t^2(\mathbf{x}_i) + \Omega(f_k) \tag{B.7}$$

$$g_i = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)}) \tag{B.8}$$

$$h_i = \partial^2_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)}) \tag{B.9}$$

この損失関数を最小化する重み w を求めることにより最適なモデルを構築することが可能となる．

B.2 LightGBM

LightGBM は，XGBoost と同様に決定木の勾配ブースティングをベースとした高速なアルゴリズムをもつモデルである．LightGBM は，訓練データの特徴量を階級に分けてヒストグラム化し，Gradient-based One-Side Sampling（GOSS）と Exclusive Feature Bundling（EFB）の技術を用いることで，意図的に厳密な枝分かれを探さず大規模なデータセットに対しても計算コストを抑え，比較的計算負荷の少ない予測を実現している．

初めに，GOSS について説明する．勾配ブースティングにおける勾配は，インスタンスの重要度の指標に用いられ，勾配が小さいことは訓練誤差が小さく学習が進んでいることを示す．GOSS は大きい勾配を持つインスタンスは残して，小さい勾配を持つインスタンスに対してはランダムサンプリングを行う．これにより効率の良いサンプリングを可能にしている．次に，EFB について説明する．特徴量の次元数が大きいデータセットは，スパースであることが多い．スパースなデータは情報を損失することなく次元削減できる．EFB では，値の衝突（conflict：同時に非ゼロの値をとること）の回数が一定の閾値以下であれば同じ bundle（特徴量の束）に追加し，同じ bundle に含まれる特徴量を，それぞれの値が取る範囲がかぶらないように範囲をシフトさせつつ一つの特徴量にまとめることで，データ次元数を削減している．