学外秘

2023 年 2 月 8 日

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

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　指導教員：江口浩二

1　研究背景

企業の実務現場においては，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．特に分類タスクを行う場合には，時系列属性から特徴量を得るため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．

図 1: 時系列属性の前処理イメージ

しかし，前処理は属人的な作業となりやすく，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．この課題を解決するため，Graph Deviation Network（GDN）[1] を応用し，「時系列属性の特徴量抽出（前処理部分）」と「分類器の学習」を一連の学習として行う GDN 応用モデル（End-to-End モデル）を構築することとした．本研究では，GDN 応用モデルに関連する 2 つの実験を行う．実験 1 では，従来モデルである勾配ブースティング（XGBoost[2]，LightGBM[3]）と精度を比較する．実験 2 では，損失関数に不均衡データの分類に用いられる Dice Loss[4] を適用し，Cross Entropy と精度を比較する．

2　関連研究

2.1　Dice Loss

Dice Loss[4] は，F 値（適合率（Precision）と再現率（Recall）の調和平均）に着目し，不均衡データに対応した損失関数である．サンプル数 N，クラス数 K のデータについて，確率 p_{ij}，教師データ y_{ij} とすると，Dice Loss はハイパーパラメータ γ を用いて，以下のように表される：

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

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．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：

$$\mathbf{x}^{(t)} = [\mathbf{s}^{(t-w)}, \mathbf{s}^{(t-w+1)}, \cdots, \mathbf{s}^{(t-1)}] \tag{2}$$

このとき，入力 $\mathbf{x}^{(t)}$ は，各ノードごとに d 次元ベクトル $v_i \in \mathbb{R}^d\ (i \in 1, 2, \cdots, N)$ に埋め込まれる．

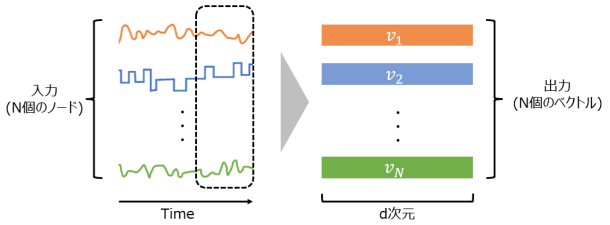

図 2: センサー埋め込み

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：

$$e_{ji} = \frac{\mathbf{v}_i^\top \mathbf{v}_j}{\| \mathbf{v}_i \| \cdot \| \mathbf{v}_i \|} \text{ for } j \in \mathcal{C}_i \tag{3}$$

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：

$$A_{ji} = \mathbf{1} \{ j \in \text{TopK}(\{e_{ki} : k \in \mathcal{C}_i\}) \} \tag{4}$$

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．

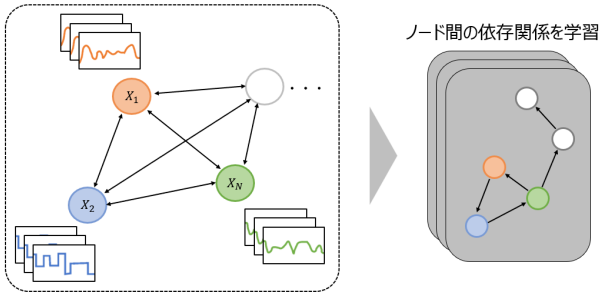

図 3: グラフ構造の学習

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：

$$\mathbf{z}^{(t)} = \text{ReLU}\left(\alpha_{i,i} \mathbf{W} \mathbf{x}_i^{(t)} + \sum_{j \in \mathcal{N}(i)} \alpha_{i,j} \mathbf{W} \mathbf{x}_j^{(t)} \right) \tag{5}$$

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．

注意係数 $\alpha_{i,j}$ は以下のように計算される：

$$\mathbf{g}_i^{(t)} = \mathbf{v}_i \oplus \mathbf{W} \mathbf{x}_i^{(t)} \tag{6}$$

$$\pi(i,j) = \text{LeakyReLU}(\mathbf{a}^\top (\mathbf{g}_i^{(t)} \oplus \mathbf{g}_j^{(t)})) \tag{7}$$

$$\alpha_{i,j} = \frac{\exp(\pi(i,j))}{\sum_{k \in \mathcal{N}(i) \cup \{i\}} \exp(\pi(i,k))} \tag{8}$$

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．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：

$$\hat{\mathbf{s}}^{(t)} = f_\theta \left(\left[\mathbf{v}_1 \circ \mathbf{z}_1^{(t)}, \cdots, \mathbf{v}_N \circ \mathbf{z}_N^{(t)} \right] \right) \tag{9}$$

$$L_{\text{MSE}} = \frac{1}{T_{\text{train}} - w} \sum_{t=w+1}^{T_{\text{train}}} \left\| \hat{\mathbf{s}}^{(t)} - \mathbf{s}^{(t)} \right\|_2^2 \tag{10}$$

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．

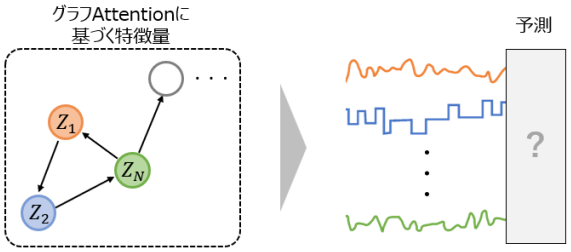

図 4: グラフ Attention に基づく予測

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：

$$\mathrm{Err}_i(t) = \left|\hat{\mathbf{s}}^{(t)} - \mathbf{s}^{(t)}\right| \tag{11}$$

$$a_i(t) = \frac{\mathrm{Err}_i(t) - \widetilde{\mu}_i}{\widetilde{\sigma}_i} \tag{12}$$

$$A(t) = \max_i a_i(t) \tag{13}$$

$A(t)$ 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．

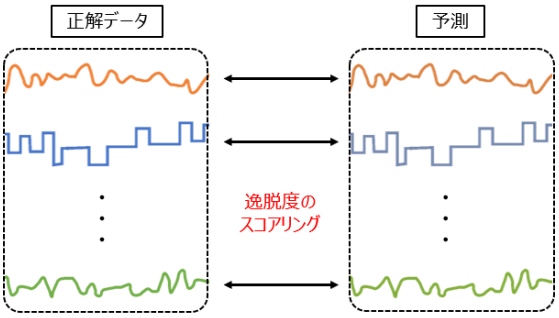

図 5: グラフ逸脱度のスコアリング

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[5] 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．

3 提案手法

本研究で構築する GDN 応用モデルは以下のとおりである：

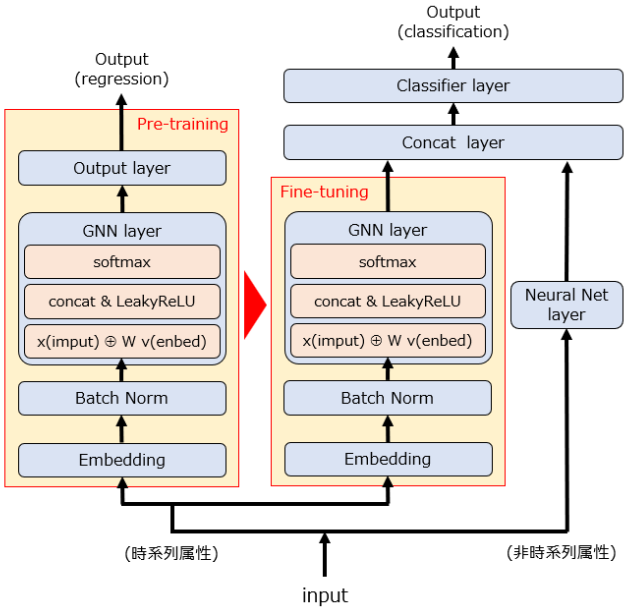

図 6: GDN 提案モデル

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．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．

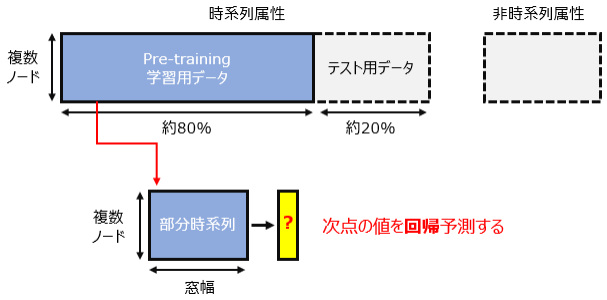

図 7: Pre-training

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．

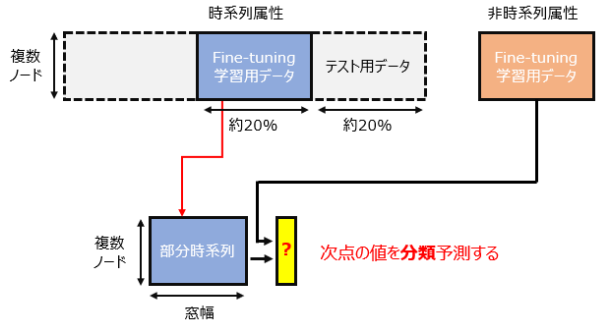

図 8: Fine-tuning

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．

4 評価実験

4.1 データセット

本研究では，Yahoo!ファイナンス[1] から取得した．東京証券取引所に上場する企業の株価や財務指標をデータセットとして用いる．

[1]https://finance.yahoo.co.jp/

東証上場企業（3700 社以上）のうち，以下の条件を満たす企業（2730 社）を抽出した：

- yfinance[2]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[3]

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）

4.2 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．

$$\text{Accuracy} = \frac{1}{N}\sum_{k=1}^{K} \text{TP}_k \tag{14}$$

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．

[2]Yahoo!ファイナンスから情報を取得するためのライブラリ
[3]株式分割や配当に伴う権利落ち等を考慮した株価

$$\text{Precision}_k = \frac{\text{TP}_k}{\text{TP}_k + \text{FP}_k} \tag{15}$$

$$\text{Recall}_k = \frac{\text{TP}_k}{\text{TP}_k + \text{FN}_k} \tag{16}$$

$$\text{macro-F1} = \frac{1}{K}\sum_{k=1}^{K} \frac{2 \cdot \text{Precision}_k \cdot \text{Recall}_k}{\text{Precision}_k + \text{Recall}_k} \tag{17}$$

4.3 実験 1（GDN 応用モデルの性能）

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，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：

表 1: GDN 応用モデル精度

	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
プラグインモデル	0.3581	0.0090	0.3316	0.0162
※時系列属性のみ[a]	0.3619	0.0183	0.3345	0.0272
※非時系列属性のみ[b]	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

[a]GDN 応用モデルに時系列属性のみを投入した実験
[b]GDN 応用モデルに非時系列属性のみを投入した実験

結果としては，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 値において従来手法を上回る精度となり，GDN 応用モデルに次いで「※時系列属性のみ」の実験が 2 番目に良い精度となった．

プラグインモデルは GDN 応用モデルのうち，Pre-training（回帰部分）の学習パラメータを凍結し，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．プラグインモデルは GDN 応用モデルより精度が低い結果となった．

4.4 実験 2（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：

表 2: Dice Loss 適用時の精度比較

	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	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．

5 考察と今後の課題

5.1 考察

Pre-training の考察

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．

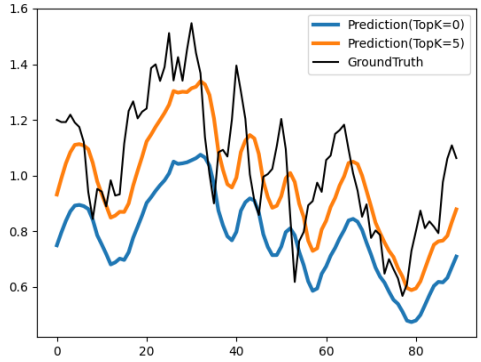

図 9: Pre-training での回帰予測

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．

実験 1（GDN 応用モデルの性能）の考察

実験 1 内の GDN 応用モデルとプラグインモデルの比較について，GDN 応用モデルは Fine-tuning において時系列属性まで逆伝播され学習されるが，プラグインモデルは時系列属性まで逆伝播されず，精度の違いがうまれていることが分かる．つまり，GDN 応用モデルでは時系列属性の特徴量抽出が最適化できていたと判断できる．

また，「※時系列属性のみ」と「※非時系列属性のみ」の比較から時系列属性の特徴量抽出が非時系列属性より GDN 応用モデルの精度向上に寄与していることが分かる．

実験 2（Dice Loss の適用）の考察

程度の小さい不均衡データに限り，一定水準の精度改善ができ，Dice Loss の有効性が示された．

5.2 今後の課題

本研究では，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．一方で，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．

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] 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)

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