CA-004

ニューラルネットワークによる記号列の統計的性質の学習

佐藤 哲 †
パーソルキャリア株式会社 †

1. はじめに

企業で扱うデータは，本来，構造化されていたり一定の規則に基づき生成されているはずであるが，なんらかの理由により非構造に近いデータとなってしまっているものが多い．例えば履歴書や職務経歴書のデータは，フォーマットも決まっており自然言語で記述されているが，表記ゆれ等の自然言語処理特有の問題に加え，文字数の制限などにより入力者による要約処理が加わっている，強調のために記号が加わっていたり箇条書きになっている，データベース化される時にフォーマット変換がされているなど，データ分析の観点からは多くのノイズ混入が発生している．そこで，データの細部の局所的な特徴に基づくのではなく，データ全体の特徴をつかみデータを分析する手法が求められている．

そこで本研究では，生成規則が不明あるいは複雑なデータのデータ系列全体の特徴を把握する目標とする．このようなデータに対するアプローチとしては，データの圧縮によってデータの情報量を計測する手法が有効である．本研究では，データを圧縮するための特徴量を推定し，その特徴量によってデータの圧縮率あるいは圧縮後のサイズを推定するという 2 段階の手法を提案する．推定の非線形性を考え，推定手法にはニューラルネットワークを採用し，入力データと推定量の写像を学習する．本研究の特色は，データ圧縮に必要な情報を用意することで，圧縮処理を行わずに圧縮結果を推定しようとする点にある．データ圧縮は計算機にとって負荷が高い処理であり，事前学習によりある程度のデータ圧縮結果が推定できるようになれば，処理の高速化の観点から有益であると言える．圧縮結果の推定が可能であればデータ同士の距離を測定することが可能になり，様々なデータ分析手法につながる．

Learning of Statistical Properties of Symbol Strings by Neural Network
†Tetsu R. Satoh，PERSOL CAREER CO., LTD.

2. コルモゴロフ複雑性を用いた記号列間の距離推定

データの情報源が不明な場合，すなわちデータ系列を発生させる情報源のモデルやパラメータが分かっていない場合に，データ同士の関連性や類似度を計算する問題を考える．例えば，データが自然言語であれば言語構成のルールや語句の有限性から，モデルが仮定できたり既知のモデルや学習データが利用できる．しかし，現実世界で使われる自然言語由来のデータは，自然言語として表現されていてもなんらかの理由により文章の要約や継ぎ接ぎがされており，人間には内容が想像できても多くの法則性が失われているデータというものが存在する．つまり，この問題設定は現実世界のデータを分析しようとする場合に必要なることが多いものである．

入力データとしての記号列を以下のように表す：

$$s^{(n)} = s_1 s_2 \cdots s_n \tag{1}$$

ここで $n > 0$，s_i は任意の記号であり，時刻 $t = t_i$ の時点で入力されたものとする．この記号列に対し，別な記号列

$$\hat{s}^{(n)} = \hat{s}_1 \hat{s}_2 \cdots \hat{s}_n$$

が存在する場合，$s^{(n)}$ と $\hat{s}^{(n)}$ の差を評価する，すなわち比較することは，多量の記号列が存在するデータを分析する場合の基本課題である．$s^{(n)}$ と $\hat{s}^{(n)}$ の比較には，単純な手法としては Levenstein 距離や Jaccard 係数と言った文字列の類似度測定法を使用することが考えられる．しかし文字列の類似度測定法は記号や n グラム同士の比較など局所的な特徴に基づく確定的な手法であり性能が限定的であるため，より高機能な記号列の比較手法が求められる．

ここで，$s^{(n)}$ に対する統計量としては，自然言語処理や符号理論で重要な記号の出現頻度が考えられる．例えば出現頻度により次の記号を予測したり出現頻度に応じた長さの符号を割り当てるなどの利用方法がある．ただし本研究では自然言語のようにデータの既知の統計的性質を仮定しないため，記号のカウントによる単純な点推定では十分な推定精度が得られない．記号列 $s^{(n)}$ に対しては，既に観測されている記号より次の記号の出現確率を推定するという定式化をする：

$$p(s_i)|_{t=t_i} = p(s_i|s_{i-1}, s_{i-2}, \cdots s_{i-m})$$

ここで m は過去を記憶する量を表すハイパーパラメータである．記号の出現確率が得られれば，記号列を例えば算術符号で表現可能である．算術符号は記号列の圧縮表現であるため，符号化された記号列は，記号列の情報量を表し，記号列のコルモゴルフ複雑性を近似していると考えられる．記号列 $s^{(n)}$ に対するコルモゴロフ複雑性 $K(s^{(n)})$ に相当する量が計算できれば，Normalized Information Distance(NID)[1] により記号列間の距離を計算可能である：

$$NID(x,y) = \frac{\max(K(x|y), K(y|x))}{\max(K(x), K(y))} \qquad (2)$$

ここで x 及び y は記号列 $s^{(n)}$ 及び $\hat{s}^{(n)}$ を示す．$K(x|y)$ は相対コルモゴロフ複雑性であり，y についての情報を利用できる場合の x のプログラム記述長であるので，x と y が似ているほど小さい値を取る．ここで，NID の計算にはコルモゴロフ複雑性すなわちモデルを記述するデータ・プログラムのサイズが必要なことに着目する．コルモゴロフ複雑性の正しい値は計算できないが，近似量としてデータの圧縮サイズが利用できることが知られており，圧縮サイズを推定できれば NID の近似値が計算可能である．関連研究ではニューラルネットワークを用いて非可逆データ圧縮を実現しているが [2][3][4]，本研究では圧縮されたデータ自体は必要ないため，圧縮処理は実施しない．従来研究 [2] では，通常の圧縮手法のようにデータ全体の統計量を調べてテーブルを作ることはコストが高いと主張しているが，使用しているニューラルネットワーク構造のために推定精度が低かった．ただしニューラルネットワークを用いて記号の出現確率を推定するアプローチは本研究と同様である．従来研究 [3] は，データ圧縮の圧縮・伸長の対称性よりオートエンコーダを利用したモデル推定と，Asymmetric Numerical Systems(ANS) を利用してデータ圧縮を実現している．従来研究 [4] は，本研究と同様に記号の出現確率を推定したのち算術符号によりデータ圧縮を実現している．本研究では，圧縮処理を行わずに圧縮サイズを推定することを目指すもので，従来研究とはアプローチが異なる．圧縮処理自体を目的にしない理由は，式（2）の計算のためには圧縮されたデータ自体は必要なく，圧縮サイズのみが必要だからである．

3. ニューラルネットワークによるデータ圧縮サイズの推定

データ圧縮サイズの推定は，(1) 記号の出現確率の推定（出現確率推定ネットワーク），(2) 出現確率による圧縮サイズの推定（圧縮情報推定ネットワーク），の 2 段階によって行う．

まず出現確率推定には，図 1 のようなニューラルネットワークを用いる．Embedding Layer は記号を one-hot 符号化表現に変換するための層で，中間層は LSTM Layer，出力層の活性化関数は softmax 関数で出現確率を推定結果を出力する．入力は記号列 $s^{(n)}$ であり，出力の教師データとしては one-hot 符号化された入力データを用いる．入力データは複数の記号列であり 2 次元行列であるが，各記号を one-hot 符号化するために図 2 のように LSTM 層に入力するデータは 3 次元行列となる．図 2 は 2 進記号列 $\{0,1\}$ に対し，$0 \to \{1,0\}$，$1 \to \{0,1\}$ のように one-hot 符号化した例である．one-hot 符号化データに対し以下で定義される softmax 活性化関数による出力を学習させることにより，記号の出現確率を推定する：

$$softmax(x) = \frac{\exp(x)}{\sum \exp(x_i)}$$

ここで x 及び x_i は，LSTM 層の出力を指す．

圧縮サイズの推定には，図 3 のようなシンプルな全結合ネットワークを用いる．入力は記号列 $s^{(n)}$ の出現確率 $p(s^{(n)})$ であり，出力の教師データとしては圧縮ライブラリにより記号列 $s^{(n)}$ を圧縮した結果のサイズ $K(s^{(n)})$ を用いる．出現確率 $p(s^{(n)})$ の推定には，出現確率推定ネットワークを用いる．

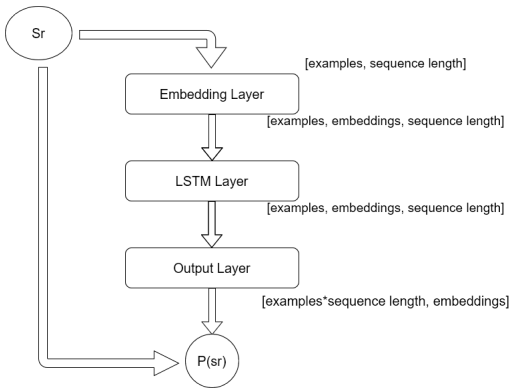

図 1: 出現確率推定ネットワーク

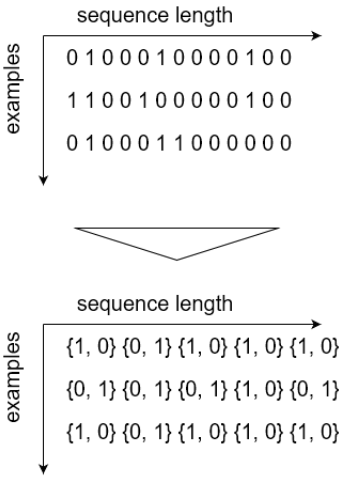

図 2: 記号列からの 3 次元行列変換

これらのネットワーク構成は試験的なものであり今後改良予定であるが，速報を目的に現在の構成を報告するものである．

4. 実験例

小規模な人工データを用いて，データの圧縮サイズを推定する実験を行う．ここでは記号列として 2 進文字列 $s_i \in \{0,1\}$ からなる記号列に対し，2 段階の学習及びデータ圧縮サイズを推定する．圧縮手法として xz[†]，bzip2[††]，gzip[†††] を用いる．中間層のノード数は 128 である．パラメータ η は記号 0 と 1 の偏り方を示しており，0.5 であれば等確率で出現する．表 1 及び図 4 に，$\eta = 0.1$ の場合のデータの平均圧縮率及び圧縮推定結果の平方根平均二乗誤差（RMSE）を示す．平均圧縮率 C は，データ $s_k^{(n)}$ に対し圧縮サイズ $K_k = K(s_k^{(n)})$，データ長を $l_k = |s_k^{(n)}|$，テストデータの数を T として

$$C = \frac{1}{T}\sum_{k=0}^{T-1}\frac{K_k}{l_k}$$

であり，RMSE は圧縮サイズの推定値を $\hat{l_k}$ として

$$RMSE = \sqrt{\frac{1}{T}\sum_{k=0}^{T-1}\left(K_k - \hat{l_k}\right)^2}$$

である．この場合は，0 の出現確率は 1 の 9 倍と大きく偏っており，データの圧縮がしやすい設定である．入力データサイズは $n = 256, 512, 1024$ と変化させている．つまり，式（1）において次のように設定した：

$$\begin{cases} n & \in & \{256, 512, 1024\} \\ s_i & \in & \{0, 1\} \end{cases} \tag{3}$$

結果より，概ね圧縮手法による圧縮率が高い方が誤差が小さいことが見て取れる．圧縮率は 17%から 45%までの開きがある．

表 2 及び図 5 に，$\eta = 0.5$ の場合の結果を示す．この場合は記号 0 と 1 の出現確率が等しいために圧縮が難しく，xz による圧縮率は約 52%，bzip2 と gzip では約 33%であった．$\eta = 0.1$ の場合に比べ圧縮手法による圧縮率の違いは小さいが推定精度にばらつきがあり，その原因については検討中である．

実装は scala-2.12 及び deeplearning4j[†] を用いており，deeplearning4j のバージョンは 1.0.0-beta7 である．圧縮手法の実装には Apache commons compress[††] を用いた．

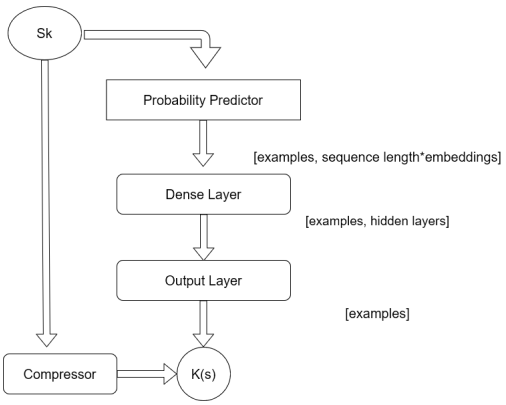

図 3: 圧縮情報推定ネットワーク

[†]https://tukaani.org/xz/
[††]https://www.sourceware.org/bzip2/
[†††]https://www.gnu.org/software/gzip/

[†]https://deeplearning4j.org/
[††]https://https://commons.apache.org/proper/commons-

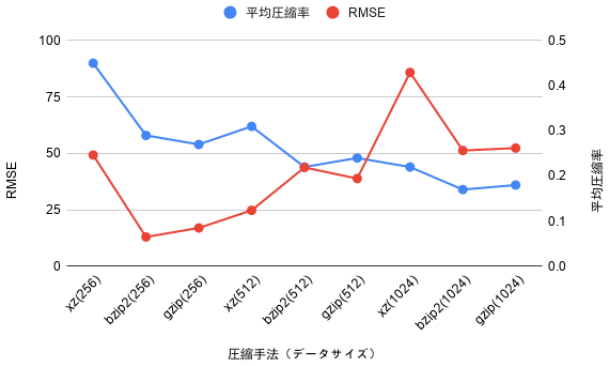

図 4: 圧縮サイズ推定誤差と平均圧縮率

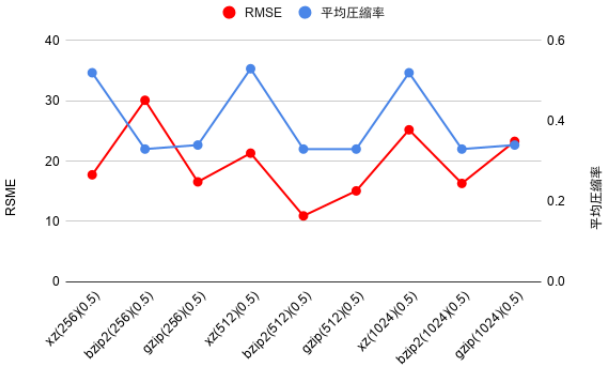

図 5: 圧縮サイズ推定誤差と平均圧縮率

5. おわりに

本研究では，記号列のデータより記号の出現確率を推定し，その結果より記号列全体の統計的性質を含むと考えられる非可逆データ圧縮サイズを推定する手法を考案し，予備実験結果について述べた．実験結果より，小規模なデータセットに対しては圧縮手法に対するデータセットの圧縮サイズが推定できていることが示唆されているが，データセットの種類及び圧縮手法により精度が異なり，今後の研究が必要なことが分かった．その他にも，出現確率推定ネットワーク及び圧縮情報推定ネットワークの改良，実際のデータを用いた実験，推定された圧縮サイズによる NID の近似計算など多くの課題が残されている．

表 1: 人工データによる実験例（$\eta = 0.1$）

圧縮手法	平均圧縮率	RMSE
xz(256)(0.1)	0.45	49.3
bzip2(256)(0.1)	0.29	13.0
gzip(256)(0.1)	0.27	17.0
xz(512)(0.1)	0.31	24.8
bzip2(512)(0.1)	0.22	43.8
gzip(512)(0.1)	0.24	38.9
xz(1024)(0.1)	0.22	85.9
bzip2(1024)(0.1)	0.17	51.3
gzip(1024)(0.1)	0.18	52.4

compress/

参考文献

[1] C.H. Bennett, P. Gács, M. Li, P.M.B. Vitányi and W. Zurek, Information Distance, IEEE Trans. Information Theory, Vol. 44, No. 4, pp. 1407–1423, 1998.

[2] J. Schmidhuber and S. Heil, Sequential neural text compression, IEEE Trans. Neural Networks, Vol. 7, No. 1, pp. 142–146, 1996.

[3] F. H. Kingma, P. Abbeel and J. Ho, BitSwap: Recursive Bits-Back Coding for Lossless Compression with Hierarchical Latent Variables, arXive: 1905.06845, 2019.

[4] M. Goyal, K. Tatwawadi, S. Chandak and I. Ochoa, DeepZip: Lossless Data Compression using Recurrent Neural Networks, arXiv: 1811.08162, 2020.

表 2: 人工データによる実験例（$\eta = 0.5$）

圧縮手法	平均圧縮率	RMSE
xz(256)(0.5)	0.52	17.73
bzip2(256)(0.5)	0.33	30.1
gzip(256)(0.5)	0.34	16.56
xz(512)(0.5)	0.53	21.32
bzip2(512)(0.5)	0.33	10.91
gzip(512)(0.5)	0.33	15.06
xz(1024)(0.5)	0.52	25.2
bzip2(1024)(0.5)	0.33	16.3
gzip(1024)(0.5)	0.34	23.28