4C-01

ストロークデータの圧縮手法の比較と改良

小林亮太郎　松崎拓也

東京理科大学 理学部第一部 応用数学科

1. はじめに

タブレット端末やスマートペンシルの普及により、筆記データをストロークデータとして保存する機会が増えている。しかし、ストロークデータはサンプリング周期を短くすると大きな記憶領域を必要とするという問題がある。そこで本研究ではストロークデータをより小さく圧縮するために、差分符号化、ハフマン符号化、スプライン近似における節点数の削減などの手法を組み合わせ、圧縮率を比較した。結果として、差分符号化とハフマン符号化を組み合わせた手法で元のデータの約10%まで圧縮できた。

本研究に使用したデータは、1ストロークがペン位置座標(x, y)の列になっており、各x, yは4バイトの整数で保存されている。各座標は、約120Hz でサンプリングされている。また、比較の際に用いる圧縮率は元データのサイズに対する圧縮後のサイズの割合と定義する。

2. 差分符号化とハフマン符号化

ストロークデータは線という連続したデータをサンプリングされた座標列として離散的に表している。そのため、隣接するサンプル点は距離が近いと仮定できる。この仮定より、ひとつ前のサンプル点からのx座標、y座標それぞれの差分は小さい値になることが予想される。また、ペンの動きが等速直線運動に近い場合、2階差分をとると0が多く出現する。以上の考察をもとに次の3つの圧縮手法を比較検討する。

2.1 差分符号化

実際に隣接するサンプル点座標の差分値を調べたところ、どのデータに対しても4ビット以内で表せると分かった。この結果を利用して以下の手順で圧縮した。

1. 各ストロークの先頭のサンプル点のx、y座標をそれぞれ4バイト整数として保存する。
2. 各ストロークの2つ目以降のサンプル点について、前サンプル点とのx座標、y座標の差分値をそれぞれ4ビット整数として保存する

2.2 2階差分値をハフマン符号化

2階差分値の分布を調べたところ全体の97%以上が$0, -1, 1$で構成されていた。この結果から2階差分値をハフマン符号化することでさらに圧縮することができると考え、次の手順で圧縮を行った。

1. 各ストロークの先頭のサンプル点のx, y座標をそれぞれ4バイト整数として保存する
2. 各ストロークの2番目のサンプル点と先頭のサンプル点のx, y座標の差分を計算する
3. 各ストロークの3番目以降のサンプル点についてx, y座標の2階差分を計算する
4. 2. の1階差分及び3. の2階差分値をハフマン符号化して保存する

2.3 2階差分のブロック符号化

上記の通り2階差分値はそのほとんどが$0, -1, 1$で占められる。この結果から2階差分の列には似た部分列が多く出現すると予想される。例えば、0が多く出現することから$0, 0$の並びが多く出現することなどが考えられる。そこで2階差分の整数列を2個ずつ区切って組を作り、２つまとめてハフマン符号化した。

3. スプライン近似と節点数削減

スプライン補間とは与えられた点列（節点）の間を多項式関数で滑らかに補間する手法である。$[t_i, t_{i+1}]$の区間における多項式関数を以下の$S_i(t)$として a_i, b_i, c_i, d_i を求めることによって関数を決定する。

$$S_i(t) = a_i + b_i(t - t_i) + c_i(t - t_i)^2 + d_i(t - t_i)^3$$

反復挿入法[1]はスプライン補間で元のストロークデータを近似する際にサンプル点の中からなるべく少数の節点を、誤差を小さくしつつ選ぶためのアルゴリズムである。選ばれなかった節点は保存しないことでデータを圧縮できる。

本研究では2次元の点列を補間するため助変数tを導入し$(t_i, x_i), (t_i, y_i)$の2つの系列に対してスプライン補間を適用する。t_iはストローク内のサンプル点の順番すなわち$t_i = i$と定義した。

具体的な手順としては、まず、反復挿入法を用いて節点数を削減し、残った節点のx座標、y座標とそれに対応するtを4バイト整数型で保存した。

Comparison and Improvement of Stroke Data Compression Methods
Ryotaro KOBAYASHI and Takuya MATSUZAKI
Department of Applied Mathematics, Faculty of science, Tokyo University of Science

Copyright ©2021 Information Processing Society of Japan.
All Rights Reserved.

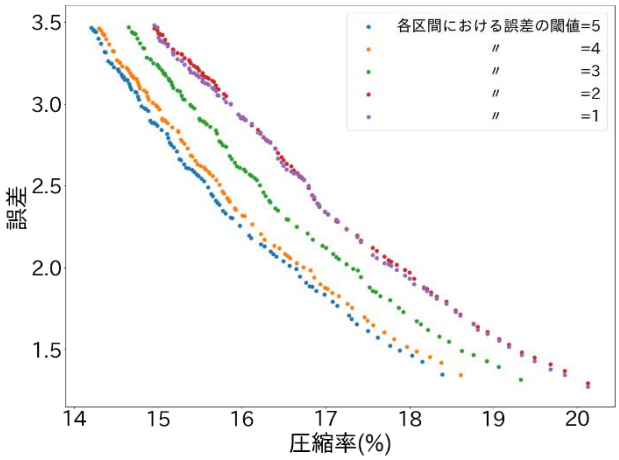

図1：反復挿入法の節点数と誤差

反復挿入法には各区間における誤差の閾値とストローク全体に対する誤差の閾値という2つのパラメータがある。パラメータを変化させ、誤差と削減後の圧縮率の関係を調べた結果、図1のようになった。他の手法との比較の際は各区間における誤差の閾値が5の系列の中で誤差が最も小さくなるパラメータを採用した。

さらに、反復挿入法により削減した節点に対して、その座標の2階差分をとってハフマン符号化することでより圧縮することを試みた。

4. ニューラルネットワークの利用

畳み込みネットワークを用いたオートエンコーダでは複数の畳み込み層を重ねることにより中間層のノード数を減らしている。圧縮対象のデータを入力として、同じデータを出力するように学習させ、中間層の重みを保存することにより、ストロークデータを圧縮することを試みた。例えば、1次元の畳み込み層を2層用いて各層のストライドを2にした場合、元データの約1/4に圧縮することができる。

本研究では、畳み込み層2層と逆畳み込み層2層で合計4層のニューラルネットワークを使って実験を行った。第1層の後に活性化関数としてRelu関数を用いた。

4.1 実験結果

元のデータとニューラルネットを用いて復元した結果の例を図2に示す。例からわかる通り、復元時の誤差を減らそうとすると復元後のデータサイズが元データよりも大きくなり、圧縮はできなかった。また、第3層の逆畳み込み層にも活性化関数を用いた場合も試したが、同様に圧縮できなかった。この原因として逆畳み込み層の多数の重みパラメータを8バイトの浮動小数点数で保存する必要があることなどが考えられる。

圧縮率　82.0%　　圧縮率　636%　　圧縮率　1100%

図2：元データ（青）と復元結果（黒）

5. 圧縮手法の比較

iPad と Apple Pencil を用いて、手書き文書データを4つ作成した。各データの平均ストローク数は137、各ストロークに含まれるサンプル数の平均は45となった。各手法を用いてこれらのデータを圧縮したときの圧縮率の平均を表1に示す。また、いくつかの汎用圧縮アルゴリズムによる結果を表2に示す。これら全ての中で、2階差分を取ってハフマン符号化する手法が最も効率よく圧縮できた。

表1：実験結果

圧縮手法	圧縮率
差分4ビット符号化	15.8%
2階差分ハフマン符号化	10.2%
2階差分ブロックハフマン符号化	10.9%
節点数削減	18.4%
節点数削減＋2階差分ハフマン符号化	18.7%

表2：汎用圧縮アルゴリズムを適用した結果

アルゴリズム	圧縮率
Deflate	27.5%
Bzip2	29.6%
LZMA2	14.2%
PPMd	29.1%

6. おわりに

差分符号化とハフマン符号化を組み合わせたときに元データの約10%まで圧縮でき、最も圧縮効率が高かった。これは各ストロークで等速直線運動に近い動きが多いためだと考えられる。本研究では整数型のストロークデータを用いたが、実際の筆記データは浮動小数点型で記録されることが多いため浮動小数点型のデータに対する有効な手法の開発が今後の課題である。

参考文献

[1]小林富士男、尾関孝史. スプライン関数による手書き文字の節点数削減. 画像電子学会誌, 第37巻, 第2号, pp127-132, 2008.

Copyright ©2021 Information Processing Society of Japan.
All Rights Reserved.