麻雀AI開発における畳み込みニューラルネットワークを用いた手牌読みモデルの実験研究

ゲームAI班

1.序論

近年四人零和有限不確定不完全情報ゲームである麻雀をプレイするAIの開発が盛んであるが、現在広く普及している麻雀AIに用いられている手法とは異なるアルゴリズムを用いた麻雀AIの開発を目指し、その全体の構造を考案した。その上で、全体の判断の基盤となる、麻雀というゲームの不完全情報性に対応するための手牌読みのモデルの開発を試みた。開発においては、用いる河のデータの特徴量となりうる要素が多く、またそのパターンが莫大であるという特性から、囲碁や将棋のAIにも用いられる畳み込みニューラルネットワークと呼ばれる手法を採用した。実際の麻雀の試合のデータを用いる前に、ルールを簡易化したゲームにおけるデータを用いて学習させてモデルを構築し、その精度を調べた。

2.目的

河(捨て牌)の情報から手牌を構成する牌を予測するモデルを作成するために有効な機械学習の方法及びパラメータを見つけ出す。

3.実験

手順　情報の単純化のため、他の情報を排除して河と手牌の情報のみについて考えるモデルを作成した。

手牌のみの情報から得点期待値の最も高い打牌を選択するプログラムを作成し、模擬一人麻雀を行うプログラムを作成し、その各巡目において打牌後の手牌構成と河の情報を記録した。河の状態は、時系列に基づく捨てられた二牌の組み合わせを二次元の配列化したものをデータとした。それを4000局分作成し、今回の学習やテストに用いるデータセットとした。

その後、様々な条件を変化させながらデータを学習させ、各学習回ごとの精度を記録した。

評価　以下の2つの2通りの選択の組み合わせで4通りの方法を考え、それで精度を評価した。

i.学習に使用したデータ　　ii.学習に使用していないデータ

※ i : ii = 7:3

a.学習時に使用した損失関数　b.L1(絶対誤差)

i→LOSS　　ii→L1

a→train　　b→test

条件　条件を変えながら以下の○パターンに分けて実験をした。①を基準の実験とし、各実験で変更した条件のセルには着色を施している。

	損失関数	勾配計算式	学習率	conv1	conv2	FullyConnect1	FullyConnect2	正規化
A	MSE	SGD	0.01	(3・1・2)	(3・2・4)	(196・384)	(384・34)	(1/8・1/4)
B	MSE	Adam	0.01	(3・1・2)	(3・2・4)	(196・384)	(384・34)	(1/8・1/4)
C	MSE	SGD	0.001	(3・1・2)	(3・2・4)	(196・384)	(384・34)	(1/8・1/4)
D	MSE	SGD	0.01	(3・1・2)	(3・2・4)	(196・384)	(384・34)	×
E	MSE	SGD	0.01	(1・1・2)	(1・2・4)	(256・384)	(384・34)	(1/8・1/4)
F	MSE	SGD	0.01	(9・1・32)	(4・32・64)	(1600・1600)	(1600・34)	(1/8・1/4)
G	Custom	SGD	0.01	(9・1・32)	(4・32・64)	(1600・1600)	(1600・34)	(1/8・1/4)

※損失関数Custom:

$$L = MAX(0, |data - label| - \epsilon)^2$$

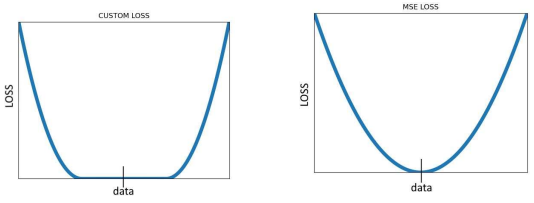
※conv(x,y,z)=

conv(カーネルサイズ、入力チャンネル数、出力チャンネル数)

※FullyConnect(x,y)=

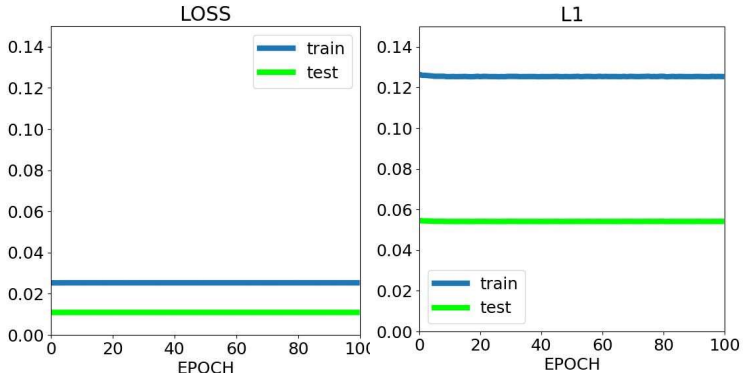
FullyConnect(入力サイズ、出力ベクトルサイズ)

※正規化(x,y)=(data,label)



4.結果

A



	A	B	C	D	E	F	G
LOSS-train	0.025276483	0.025353421	0.025260884	0.376041667	0.025277378	0.025274493	0.005697214
LOSS-test	0.010902367	0.010932079	0.011232877	0.163541667	0.010899336	0.010902057	0.002462061
L1-train	0.125278401	0.124000067	0.125479452	0.471875	0.12528207	0.1252707	0.151985854
L1-test	0.054070797	0.053482588	0.054246575	0.20625	0.054070945	0.054069248	0.065662526

Aの評価値のグラフは上図のようになった。他のグラフも同様に横ばいであり、学習回数が増えても損失が減少しておらず、学習が進んでいなかった。

具体的な出力値を見ると、入力値に関わらず出力値が一様に固定されていた。

各条件における学習完了後の評価値は上の表のようになった。

これを見ると、ほぼ共通した指標となるL1-testは、実質的にどの実験でもほぼ同じ値をとった。

5.考察

学習が進まなかった原因として、次のようなものが考えられる。

・不適切な損失関数

今回のラベルデータ及び出力値は小数の配列であったため、それらの誤差を総合的に評価する適切な方法が見つからなかった。

・学習データの量と質

河の情報を2牌の時系列の組み合わせで二次元の配列化してデータとしたが、畳み込みを行うときの縦横の関係が薄い部分があったために、結果に影響を及ぼした可能性がある。

また、データにはシュミレーション中の巡目の浅いデータと深いデータ、すなわち情報量の多いデータと少ないデータが入り混じっているため、全体の情報が薄くなっていると考えられる。同様にデータが薄くなっている原因と考えられることとして、各値の最大値を1にする正規化の処理を行っていたことも挙げられる。

さらに、出力値が一定となっているということは、内部のパラメータが単純に最適化されて過学習のような状態に陥っている可能性もある。

・畳み込み層の不足

人間にも推測の難しいことをCNNを用いて予想させようとしているが、その判断過程はかなり複雑なものになると考えられる。そのため、畳み込み層をより深くとって深層学習を行うことが必要な可能性がある。

6.結論

今回は麻雀というゲームの不完全情報性を克服するために単純化したゲームにおけるCNNを用いた予測モデルの作成を試みたが、CNNが出力するデータは数回学習を進めただけで一定となりそれ以上の精度を出すことはできなかった。また、それぞれの実験のlossの最小値を比較することによって最適なパラメーターを探索しようと試みたが、lossの値に違いはあれども、それは自作損失関数の許容誤差や正規化の有無による単位のちがいが原因であり、逆算するとほぼすべて同じ値をとった。またtrainとtestの評価値を比べるとすべての条件においてtestのほうが精度がよくこれも疑問としてあげられる。これが「4. 考察」で述べた学習データの量と質によるものなのか、もしくはそれに加えたその他の原因が絡まっているためなのかを慎重に検査しもう一度今回の実験を行う必要があるとわかった。

7.今後の展望

・各条件の調整に際して、パラメータを一つの値でなく様々に変えながらの検証
・データの形式を畳み込みにより適した形に改良しての検証
・学習に用いる自作関数を多数考案しての検証
・実行環境の見直しによる実験の高速・効率化

8.参考文献

- https://qiita.com/mathlive/items/8e1f9a8467fff8dfd03c
- https://dreamer-uma.com/pytorch-dataset/
- https://www.hellocybernetics.tech/entry/2017/06/19/084210