文字認識精度の向上のために

tomorrow

目 次

1　はじめに

1.1　概要

この論文は「文字認識の精度を向上させるためにはどうすればいいか」という問題に対して,画像の処理や変換,機械学習やディープラーニングの手法を変えて実験をすることで,文字認識の最適な手法を考えるものである.文字認識は文字の中で最も簡単で基本的である数字で実験を行う.

2　基礎知識

2.1　理論

2.1.1

この論文で用いる文字認識の方法を説明する.今回認識する数字の画像は,縦28px ,横28px,文字の部分が黒,背景が白となっている. 画像中のそれぞれのピクセルにおける黒色の濃さを表した値をデータとして捉えていく. 前述のように,今回扱う画像は縦28px,横28px,つまり784次元のベクトルとして表現されている.

$$\text{batch\_num} : \text{データの個数}$$

$$\mathrm{x} = [\text{batch\_num}, 784]$$

これを用いて,0〜9の中から正しい物を導き出すために重みとして,行列[784, 10]をかけることで,10通りの結果が生まれ,それにバイアスとして[10]を付け足すことで,10種類の数字に対する出力となる. この重みはそれぞれのピクセルの密度が濃い部分がそれぞれの数字であることを肯定する証拠ならば正,反する証拠ならば,負とするものである.つまり,それぞれのピクセルにおける濃さとそのピクセルが示す数字を表す重みの積の和10種類を求めている.

$$\begin{aligned} \text{weight} &= [784, 10] \\ \text{bias} &= [10] \\ \mathrm{y} &= \mathrm{x} \times \text{weight} + \text{bias} \end{aligned}$$

これらの値が大きい物がその数字を表している可能性高いととわかるのでsoftmax関数を用いて確率に変換することで,その数字が表している数字が何かを判別す

ることができる.

$$p = \mathrm{softmax}(y)$$

これまで述べてきた重みとバイアスを機械学習によって求めさせるために,誤差関数としてクロスエントロピーを用いる. クロスエントロピーは,実際の答えと現在の重みとバイアスにより導かれた答えとの差を現すので,これが最小になるようにgradient descentアルゴリズムを用いることで,重みとバイアスを計算していく.これをくりかえすことで最適な重みとバイアスを決定しようとする.

$$\mathrm{cross\_entropy} = \sum_i y'_i \log(p_i)$$

これによって得られた重みとバイアスを用いて,テスト用のデータと照らし合わせることで,文字認識の精度を測定することができる.

2.1.2

2.2 プログラム

以上に示した方法によって文字の認識精度を調べるためにMNISTを学習用データとして,TensorFlowを用いて機械学習を行い,テスト用データを使って,認識精度の測定を行う.以下にそのソースコードを示す.

Listing 1: Pythonのコード

```
# -*- coding: utf-8 -*-
import tensorflow as tf
import input_data

#get mnist_data
mnist = input_data.read_data_sets("../MNIST_data/", one_hot = True)

x = tf.placeholder(tf.float32, [None, 784])

weight = tf.Variable(tf.zeros([784, 10]))
bias = tf.Variable(tf.zeros([10]))

y = tf.nn.softmax(tf.matmul(x, weight)+ bias)

y_ = tf.placeholder(tf.float32, [None, 10])
```

```
cross_entropy = -tf.reduce_sum(y_ * tf.log(y))

train_step = tf.train.GradientDescentOptimizer(0.01).minimize(cross_entropy

init = tf.initialize_all_variables()

sess = tf.Session()
sess.run(init)

for i in range(1000):
    batch_xs, batch_ys = mnist.train.next_batch(100)
    sess.run(train_step, feed_dict = {x: batch_xs, y_: batch_ys})

correct_prediction = tf.equal(tf.argmax(y, 1), tf.argmax(y_, 1))

accuracy = tf.reduce_mean(tf.cast(correct_prediction, "float"))

print("", sess.run(accuracy, feed_dict = {x: mnist.test.images, y_: mnist.t
```

3 問題提起

3.1 問題点

2章に示したような方法を用いて文字認識の精度を測ると認識精度は平均して91.33％というけっかになった. 一見すると,精度が高く見えるが,機械学習の中では高いというよりもむしろ低い値で,改善の余地がある. まず第一に,

3.2　問題点の改善点

3.2.1　学習回数

3.2.2　学習アルゴリズム

4　実験方法

5　実験結果

6　結論

参考文献

[1] hoge