IF3270 Pembelajaran Mesin

Implementasi Mini-batch Gradient-Descent

Oleh:

13517026 - Ahmad Mutawalli (K2)
13517033 - Harry Rahmadi Munly (K3)
13517062 - Ardysatrio Fakhri Haroen (K2)
13517077- Dandi Agus Maulana (K2)

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG

2020

1. **Penjelasan Implementasi**

   Modul MLP ini kami implementasikan menggunakan python, dengan bantuan library numpy dan pandas. Arsitektur yang digunakan adalah 3 layer seperti pada slides kuliah. Satu input layer, satu hidden layer dan satu output layer.

   - Pada input layer terdapat 4 unit karena ada 4 fitur pada data iris
   - Pada hidden layer kami buat agar jumlah unit dapat bervariasi, tapi yang digunakan untuk eksperimentasi adalah 4 unit
   - Pada output layer terdapat 3 unit, dengan detil sebagai berikut:
     - Misalkan nama ketiga unit output layer: *o1, o2, o3*
     - Ketiga unit tersebut merepresentasikan 3 kelas yang mungkin untuk diklasifikasikan. *o1* merepresentasikan kelas 0, *o2* untuk kelas 1, dan *o3* untuk kelas 2
     - Apabila target seharusnya merupakan kelas 2, maka o1 seharusnya bernilai 0, *o2* seharusnya bernilai 0, dan *o3* seharusnya bernilai 1. Dan hal yang sama berlaku untuk target output berupa kelas-kelas lain

   Berikut pula penjelasan variable-variable yang kami gunakan pada program implementasi kami:

   - Weight pada link input layer ke hidden layer kami simpan dalam sebuah matriks *weights_ItoH*
   - Weight pada link hidden layer ke output layer kami simpan dalam sebuah matriks *weights_HtoO*

   Berikut adalah penjelasan alur secara menyeluruh pada implementasi yang kami lakukan. Kami mengeset epoch untuk mempunyai batas tertentu yang dapat diubah pada konstruksi objek MLP. Kemudian setiap poin dibawah ini kami lakukan untuk setiap instance.

   a. **Feed Forward**
      - Dilakukan inisiasi seluruh bobot secara random. Bobot memiliki nilai antara -1 dan 1
      - Dilakukan perkalian dot product antara weight yang ada antara input layer dan hidden layer dengan setiap unit pada input untuk menghasilkan net output. Kemudian net output dimasukkan ke net function untuk mendapatkan hasil dari suatu hidden unit. Hal ini dilakukan untuk tiap unit pada hidden unit. Hasil dari fungsi aktivasi disimpan dalam suatu array yang besarnya sama dengan jumlah unit pada hidden unit
      - Dilakukan perkalian dot product antara weight yang ada antara input layer dan hidden layer dengan output dari tiap hidden unit untuk menghasilkan net output. Kemudian net output dimasukkan ke net function untuk

mendapatkan hasil dari suatu output unit. Hal ini dilakukan untuk tiap unit pada output unit. Hasil dari fungsi aktivasi disimpan dalam suatu array yang besarnya sama dengan jumlah unit pada output unit

b. Backpropagation

- Delta weight disimpan dalam sebuah array yang dimensinya sama dengan array untuk menyimpan weight (berlaku untuk semua layer). Pada iterasi pertama, delta weight dinisialisasi berisi 0
- Jika hitungan mini batch belum mencapai batasnya (belum saatnya untuk update weight), maka dilakukan backpropagation dengan mengubah delta weight
    - Delta weight untuk link antara hidden dan output layer dihitung terlebih dahulu. Kemudian hasilnya disimpan dalam matriks. Bila matriks sudah ada isinya, maka matriks yang disimpan adalah hasil penjumlahan per-elemen dari matriks yang menyimpan penjumlahan antara delta weight yang lama dan yang baru dihitung
    - Delta weight untuk link antara input dan hidden layer dihitung kemudian. Setelah itu hasilnya disimpan dalam matriks. Bila matriks sudah ada isinya, maka matriks yang disimpan adalah hasil penjumlahan per-elemen dari matriks yang menyimpan penjumlahan antara delta weight yang lama dan yang baru dihitung
- Jika hitungan mini batch sudah mencapai batas (sudah saatnya mengupdate weight), maka dilakukan backpropagation dengan meng-update tiap weight yang ada pada tiap layer
    - Update weight merupakan penjumlahan per-elemen matriks antara weight yang lama dengan delta weight
    - Setelah update weight untuk tiap layer selesai, maka matriks delta weight diinisialisasi menjadi 0

2. Hasil Eksekusi

a. Weight input layer to hidden layer

Weight untuk hasil training seluruh data

	Hidden0	Hidden1	Hidden2	Hidden3
WInput0	-1.466713	0.455637	-0.487798	-1.173691
WInput1	0.308435	0.561372	-0.665985	-1.856365
WInput2	-0.859972	0.647075	0.043949	3.067730
WInput3	-0.157834	-0.075064	-0.126900	1.145672
WInput4	0.425976	0.994432	-0.009026	-0.089883

b. Weight hidden layer to output layer

Weight untuk hasil training seluruh data

	Output0	Output1	Output2
WHidden0	-0.721286	0.565732	-0.163582
WHidden1	0.294909	-0.141371	-2.520354
WHidden2	0.779879	0.512019	0.284924
WHidden3	-6.768946	0.301010	5.936865
WHidden4	1.464148	-1.137096	-2.265323

c. Hasil eksekusi dan data validasi asli

Validasi dilakukan dengan membagi dataset menjadi 80% untuk training dan 20% untuk testing

```
Hasil Prediksi
[0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 2, 1, 2, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2]
Data Validasi
[0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2]
Akurasi Prediksi
93.333%
```

3. Perbandingan dengan Hasil MLP Sklearn

Perbandingan dengan MLP Sklearn menggunakan data yang sama yang digunakan untuk validasi modul MLP kami (80% training dan 20% validasi).

a. Hasil myMLP

```
Hasil Prediksi
[0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 2, 1, 2, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2]
Data Validasi
[0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2]
Akurasi Prediksi
93.333%
```

b. Hasil MLP Sklearn dengan solver = adam

```
Prediction with MLP Classifier
Weight
[array([[-0.42125097, -0.31916908,  0.32616754, -0.39271807],
       [-0.63333635, -1.40701224,  1.30297073, -0.34262047],
       [-0.3985596 , -0.6314246 , -1.13983354, -0.39742393],
       [-0.52750759, -0.04919511, -1.30427959, -0.06105211]]), array([[-0.30454024],
       [ 0.4075164 ],
       [ 2.48072195],
       [ 0.42577987]]), array([[ 0.83991116, -1.10372187, -0.89195329]])]
[0 0 0 0 0 0 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2]
Akurasi Prediksi
66.667%
```

c. Hasil MLP Sklearn dengan solver = sgd

```
Prediction with MLP Classifier
Weight
[array([[-0.1436389 , -9.74782802, -0.86532603, -0.34217181],
       [-0.61148252, -5.21108848, -0.54309865, -0.2673418 ],
       [-0.17870046, -8.23460304, -0.13987818,  0.32062385],
       [-0.51160735, -2.01415277, -0.81811478,  0.29508745]]), array([[ 0.65851687],
       [-2.04994436],
       [-0.40853049],
       [ 0.4211138 ]]), array([[-0.70533644, -1.11686699,  0.64396041]])]
[2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2]
Akurasi Prediksi
43.333%
```

d. Hasil MLP Sklearn dengan solver = lbfgs

```
Prediction with MLP Classifier
Weight
[array([[-0.14371751, -0.37184364, -0.86579964, -0.34235909],
       [-0.6118172 , -1.0464925 , -0.54339589, -0.26748812],
       [-0.17879827, -0.55358554, -0.13995473,  0.32079933],
       [-0.51188736,  0.44488429, -0.81856255,  0.29524895]]), array([[ 0.65887728],
       [ 0.79330918],
       [-0.40875408],
       [ 0.42134428]]), array([[ 0.71088911, -1.42885644, -0.46092056]])]
[2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2]
Akurasi Prediksi
43.333%
```

Analisis:

Diuji dengan pembagian dataset tertentu, dengan arsitektur, nilai learning rate, dan maximum epoch yang sama, modul kami dapat memperoleh hasil yang lebih akurat dibandingkan modul pada Sklearn (MLPClassifier Sklearn dengan parameter default). Dimana modul yang kami buat dapat memperoleh akurasi sekitar 93% sedangkan modul Sklearn hanya dapat mencapai akurasi maksimal 67% dengan solver adam, dan menggunakan solver lain hanya mencapai akurasi 43%. Untuk mencapai akurasi yang tinggi, modul MLP Sklearn membutuhkan jumlah hidden unit yang sangat besar, sekitar 100 hidden unit. Akan tetapi, peningkatan hidden unit sebesar itu akan meningkatkan pula kemungkinan untuk terjadi *overfitting* dengan cukup drastis. Oleh karena itu, pada kondisi ini model yang kami buat dapat lebih baik memprediksi data iris karena dapat mencapai akurasi lebih baik dengan jumlah unit yang lebih minimum.

4. Pembagian Tugas

NIM	Tugas
13517026	Struktur dasar kelas network,fungsi-fungsi dasar (sedang sakit pada waktu pengerjaan)

13517033	Feed forward, print weight, analisis, dan laporan
13517062	Struktur dasar kelas network, Metode fitting: feed forward, backpropagation, weight updating. Debugging model Network, alur testing, analisis dan laporan
13517077	Struktur dasar kelas network, predict, MLP Classififer, dan laporan