Laporan Tugas 4 – Implementasi *Multi Layer Perceptron*

Mata Kuliah *Machine Learning*

Nama : Rimba Erlangga

NIM : 16/398526/PA/17487

---

1) Dataset

Dataset yang digunakan adalah data iris.csv. Terdapat 150 observasi/baris, 4 variabel/fitur/kolom dan 1 variabel target. Target terdiri dari 3 kelas, yang masing-masing kelas terdiri 50 baris.

2) Menyiapkan data latih dan validasi

Data latih diambil dari 80% dataset keseluruhan, yaitu 120 baris, sedangkan 30 baris sisanya digunakan sebagai data validasi. Data latih terdiri dari 3 kelas yang masing-masing kelas terdiri dari 40 baris, sedangkan data validasi terdiri dari 3 kelas yang masing-masing kelas terdiri dari 10 baris.

3) Skenario Eksperimen

1. Akan dilakukan dua training berbeda. Satu training dengan learning rate 0.1, dan satu lagi dengan learning rate 0.8. Untuk setiap proses training, akan dilihat hasil loss (error) dan akurasinya, dengan epoch seoptimal mungkin. Suatu epoch dianggap optimal jika akurasinya pernah mencapai 100%. Nilai loss akan dikenai operasi log sehingga mempermudah analisis grafik.
2. Selain itu, dari kedua proses tersebut, akan dilihat akurasi untuk setiap prediksi kelasnya, baik pada proses training maupun validasi. Dan juga akan dilihat waktu yang diperlukan untuk trainingnya.

4) Arsitektur

1. **Banyak layer** : dua layer, yaitu 1 output layer dan 1 hidden layer
2. **Banyak neuron** : 4 neuron di hidden layer, dan 3 neuron di output layer
3. **Fungsi aktivasi** : fungsi sigmoid untuk hidden layer, dan fungsi softmax untuk output layer

   Rumus:

$$h_n \coloneqq \text{sigmoid}(h_n) = \frac{1}{1 + \exp(-h_n)}$$

$$p_n = \text{softmax}(y_n) = \frac{\exp(y_n)}{\sum_{k=1}^{3} \exp(y_k)}$$

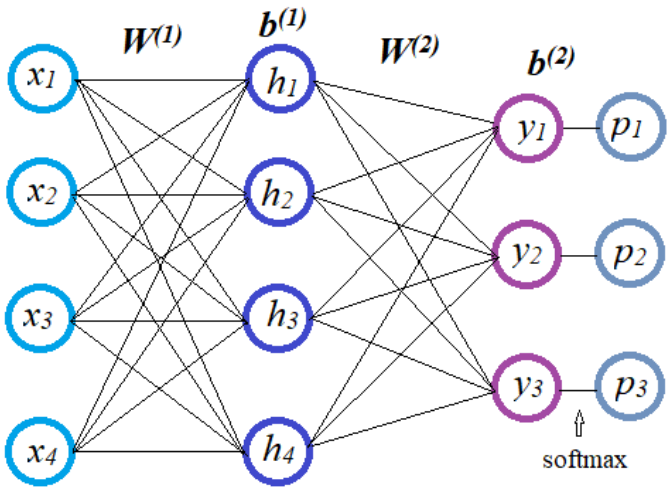
di mana:

h_n = nilai pada neuron ke-*n* di hidden layer

y_n = nilai yang masuk ke neuron ke-*n* di output layer

p_n = nilai y_n setelah diaktivasi

4. **Ilustrasi arsitektur**



5. **Representasi Parameter**

- **Bobot hidden layer** : Bobot pada hidden layer disimpan dalam matriks $\boldsymbol{W^{(1)}}$ berukuran 4x4. $W_{mn}^{(1)}$ menyatakan bobot dari garis yang menghubungkan x_m ke h_n
- **Bobot output layer** : Bobot pada output layer disimpan dalam matriks $\boldsymbol{W^{(2)}}$ berukuran 4x3. $W_{mn}^{(2)}$ menyatakan bobot dari garis yang menghubungkan h_m ke y_n
- **Bias** : Bias pada hidden dan output layer berturut-turut disimpan dalam vektor kolom $\boldsymbol{b^{(1)}}$ dan $\boldsymbol{b^{(2)}}$ yang berukuran 4 dan 3. $b_n^{(1)}$ menyatakan bias yang ditambahkan ke neuron h_n dan $b_n^{(2)}$ menyatakan bias yang ditambahkan ke neuron y_n
- **Neuron setiap layer** : Neuron pada input, hidden, dan output layer berturut-turut disimpan dalam vektor kolom $\boldsymbol{x}$, $\boldsymbol{h}$, dan $\boldsymbol{y}$. Vektor $\boldsymbol{h}$ dan $\boldsymbol{y}$ didapat dari perhitungan **feed forward** berikut:

$$\boldsymbol{h} = \text{sigmoid}\left(\boldsymbol{W^{(1)T}x} + \boldsymbol{b^{(1)}}\right)$$

$$\boldsymbol{y} = \boldsymbol{W}^{(2)T}\boldsymbol{h} + \boldsymbol{b}^{(2)}$$
$$\boldsymbol{p} = \text{softmax}(\boldsymbol{y})$$

- **Kelas hasil prediksi** : Kelas prediksi ditentukan oleh nilai argmax vektor **p** (indeks *k* di mana nilai p_k terbesar). Contoh: jika **p** = [0.8, 0.1, 0.1], maka kelas prediksinya adalah 0 (setosa). Jika **p** = [0.3, 0.2, 0.5], maka kelas prediksinya adalah 2.
- **Variabel kelas target** : Kelas target disimpan dalam sebuah vektor dummy (vektor biner yang entrinya hanya 1 atau 0), sebut saja vektor *t*. Jika kelas target adalah 0, 1, atau 2, maka vektor dummy *t* berturut-turut adalah [1, 0, 0], [0, 1, 0], atau [0, 0, 1].

6. **Fungsi loss** : *sum squared error*
   Rumus:

$$L = \sum_{k=1}^{3} \frac{1}{2}(p_k - t_k)^2$$

dengan *L* adalah nilai loss dan t_k adalah vektor dummy dari kelas target.

7. **Backpropagation**
   Untuk meng-update nilai dari seluruh parameter $\boldsymbol{W}^{(1)}, \boldsymbol{W}^{(2)}, \boldsymbol{b}^{(1)}$ dan $\boldsymbol{b}^{(2)}$, digunakan metode ***stochastic gradient descent*** yang rumusnya adalah

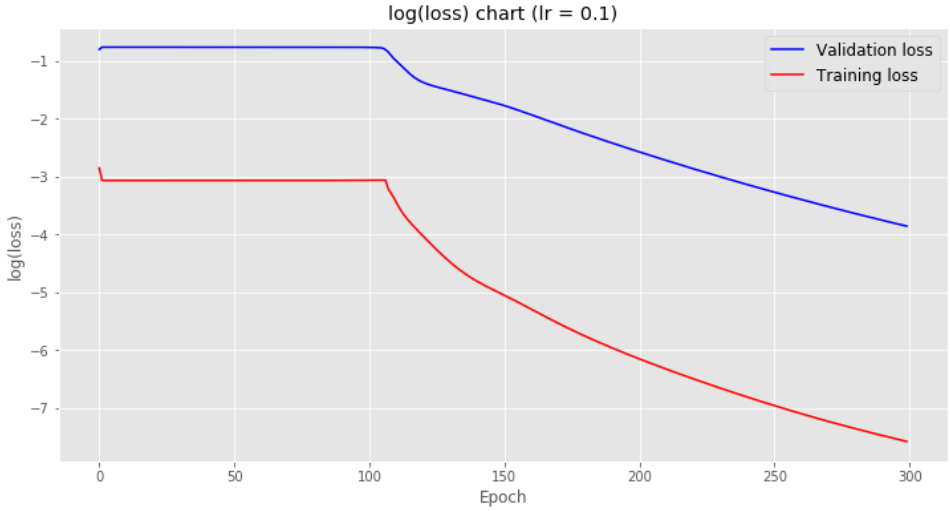
$$\boldsymbol{W}^{(1)} \coloneqq \boldsymbol{W}^{(1)} - \alpha\Delta\boldsymbol{W}^{(1)} \qquad \boldsymbol{W}^{(2)} \coloneqq \boldsymbol{W}^{(2)} - \alpha\Delta\boldsymbol{W}^{(2)}$$
$$\boldsymbol{b}^{(1)} \coloneqq \boldsymbol{b}^{(1)} - \alpha\Delta\boldsymbol{b}^{(1)} \qquad \boldsymbol{b}^{(2)} \coloneqq \boldsymbol{b}^{(2)} - \alpha\Delta\boldsymbol{b}^{(2)}$$

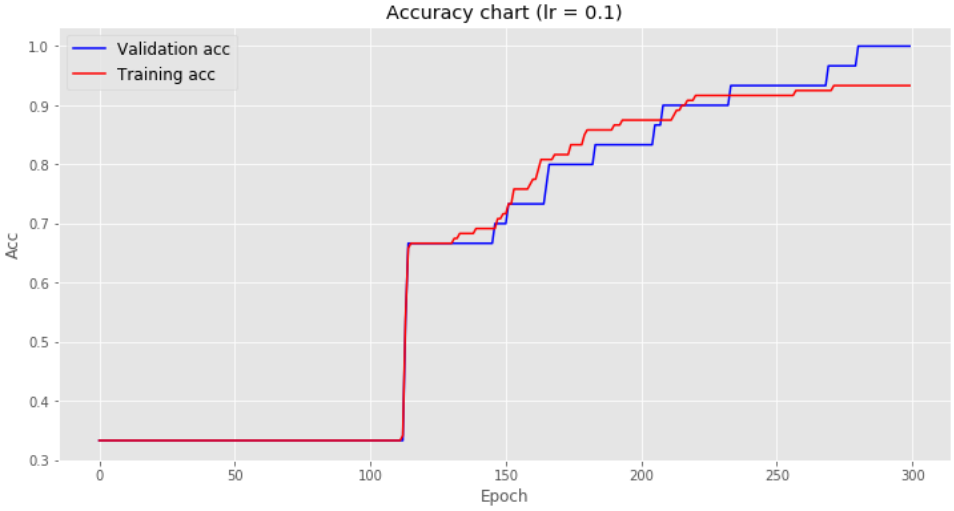
Dengan α adalah learning rate. Nilai dari perubahan setiap parameter didapat dengan rumus:

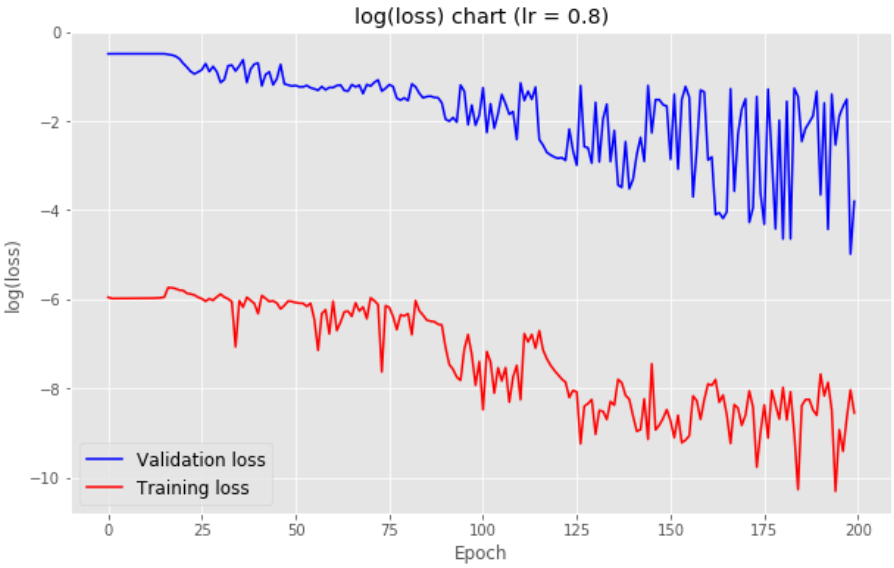
$$\Delta b_n^{(2)} = (p_n - t_n)\, p_n\, (1 - p_n)$$
$$\Delta W_{mn}^{(2)} = h_m\, \Delta b_n^{(2)}$$
$$\Delta b_n^{(1)} = h_n(1 - h_n) \sum_{k=1}^{3} \Delta b_n^{(2)} \Delta W_{nk}^{(2)}$$
$$\Delta W_{mn}^{(1)} = x_m \Delta b_n^{(1)}$$

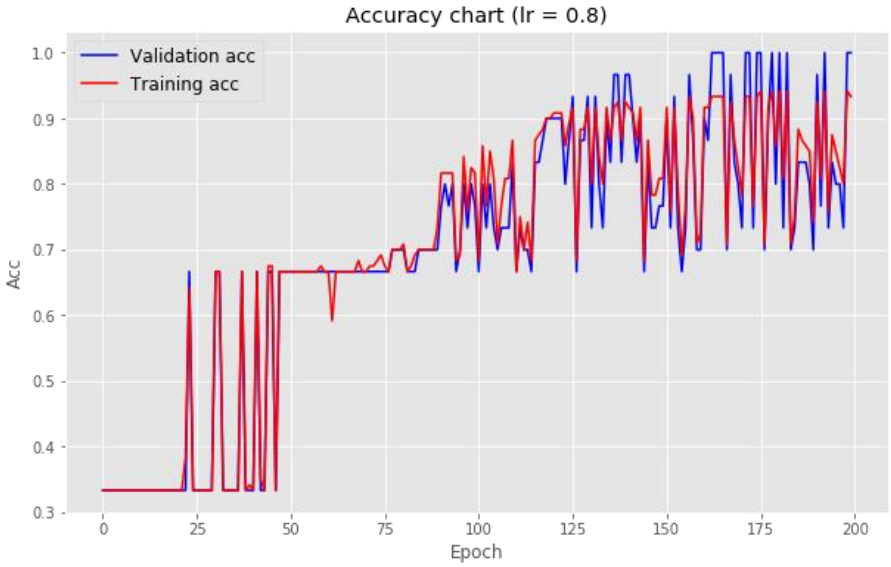
5) Hasil

Didapat grafik dari hasil training sebagai berikut:









Dari hasil eksperimen di atas, dapat diambil beberapa kesimpulan sebagai berikut:

1. Kedua model, baik model dengan learning rate 0.1 ataupun 0.8, tidak mengalami overfitting, karena loss pada training selalu lebih rendah dari pada loss pada validasi.
2. Model kedua (dengan learning rate 0.8), nilai loss dan akurasinya lebih fluktuatif dibandingkan model pertama. Model pertama memiliki grafik loss yang hampir tidak pernah naik dan grafik akurasi yang tidak pernah turun, baik pada proses training ataupun validasinya.
3. Model kedua mampu mencapai akurasi validasi 100% hanya dengan sekitar 150 epoch saja, sedangkan model kedua baru mencapai akurasi validasi 100% saat epoch-nya sekitar 270.

Kemudian, untuk prediksi masing-masing kelas, dapat dilihat dari confusion matrix berikut. Baris menyatakan kelas seharusnya, kolom menyatakan kelas yang terprediksi.

```
[[40  0  0]
 [ 0 32  8]
 [ 0  0 40]]

[[10  0  0]
 [ 0 10  0]
 [ 0  0 10]]

[[40  0  0]
 [ 0 32  8]
 [ 0  0 40]]

[[10  0  0]
 [ 0 10  0]
 [ 0  0 10]]
```

Confusion matrix pertama sampai keempat berturut-turut menyatakan hasil prediksi pada:
proses training dari model pertama, proses validasi dari model pertama, proses training dari model kedua, dan proses validasi dari model kedua. Didapat nilai akurasi:

- **Training model 1 dan 2**: kelas 0 100%, kelas 1 80%, kelas 2 100%
- **Validasi model 1 dan 2**: seluruh kelas mencapai akurasi 100%

Untuk waktu yang diperlukan pada proses training, model pertama memerlukan waktu 33.37 detik untuk 300 epoch, dan model kedua memerlukan waktu 27.51 detik untuk 200 epoch.

6) Tautan

https://github.com/hyperforest/Pembelajaran-Mesin/tree/master/Tugas%204