



**LABORATORIUM PEMBELAJARAN ILMU KOMPUTER**  
**FAKULTAS ILMU KOMPUTER**  
**UNIVERSITAS BRAWIJAYA**

---

**BAB** : BACKPROPOGATION (1)

**NAMA** : RAIKHAN GEZA ALBURAMA  
**NIM** : 225150207111040  
**TANGGAL** : 28/10/2024  
**ASISTEN** : ALIFAH KHAIRUNNISA  
ANDHIKA IHSAN CENDEKIA

---

**A. Praktikum**

1. Buka Google Collaboratory melalui [tautan ini](#)
2. Tulis kode berikut ke dalam setiap *cell* pada *notebook* tersebut.
  - a. Fungsi Binary Encoding dan Decoding

```
def bin_enc(lbl):  
    mi = min(lbl)  
    length = len(bin(max(lbl) - mi + 1)[2:])  
    enc = []  
  
    for i in lbl:  
        b = bin(i - mi)[2:].zfill(length)  
        enc.append([int(n) for n in b])  
  
    return enc  
  
def bin_dec(enc, mi=0):  
    lbl = []  
  
    for e in enc:  
        rounded = [int(round(x)) for x in e]  
        string = ''.join(str(x) for x in rounded)  
        num = int(string, 2) + mi  
  
        lbl.append(num)  
  
    return lbl
```

**b. Percobaan Binary Encoding dan Decoding**

```
labels = 1, 2, 3, 4  
enc = bin_enc(labels)  
dec = bin_dec(enc, min(labels))
```

```
print(enc)
print(dec)
```

### c. Fungsi One-hot Encoding dan Decoding

```
import numpy as
np
def
onehot_enc(lbl,
min_val=0):
    mi = min(lbl)
    enc =
    np.full((len(lbl),
max(lbl) - mi +
1), min_val,
np.int8)
    for i, x in
enumerate(lbl):
        enc[i, x - mi] =
1
    return enc

def
onehot_dec(enc,
mi=0):
    return
[np.argmax(e) + mi
for e in enc]
```

### d. Percobaan Binary Encoding dan Decoding

```
labels = 1, 2, 3, 4
enc = onehot_enc(labels)
dec = onehot_dec(enc, min(labels))

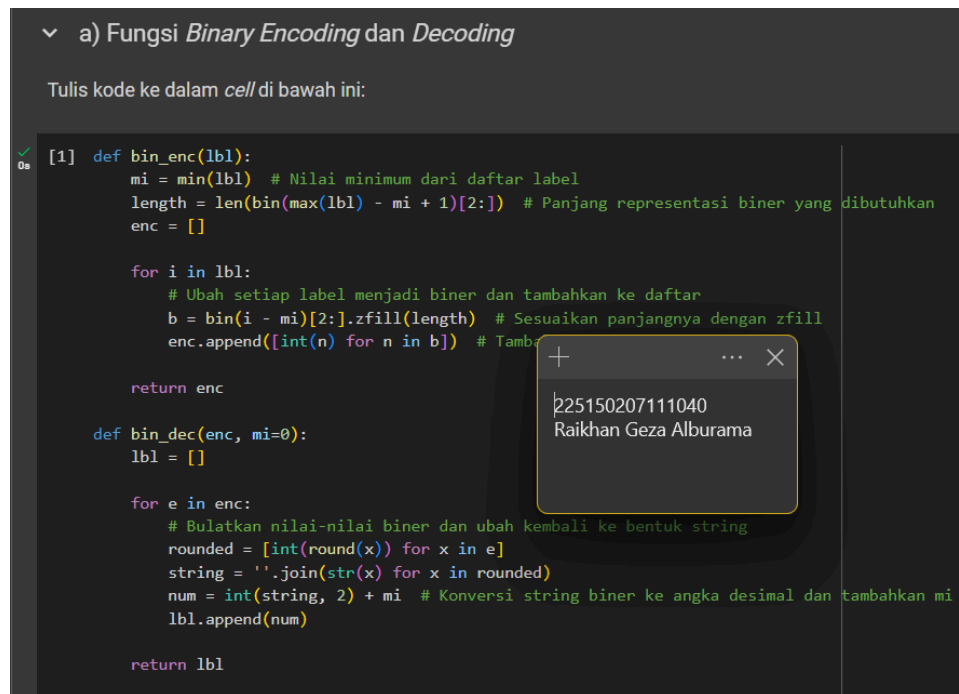
print(enc)
print(dec)
```

### e. Fungsi Aktivasi Sigmoid dan Derivatifnya

```
def sig(X):  
    return [1 / (1 + np.exp(-x)) for x in X]  
  
def sigd(X):  
    output = []  
  
    for i, x in enumerate(X):  
        s = sig([x])[0]  
  
        output.append(s * (1 - s))  
  
    return output
```

## B. Screenshot

### a. Fungsi Binary Encoding dan Decoding



▼ a) Fungsi *Binary Encoding* dan *Decoding*

Tulis kode ke dalam *cell* di bawah ini:

```
[1] def bin_enc(lbl):  
    mi = min(lbl) # Nilai minimum dari daftar label  
    length = len(bin(max(lbl) - mi + 1)[2:]) # Panjang representasi biner yang dibutuhkan  
    enc = []  
  
    for i in lbl:  
        # Ubah setiap label menjadi biner dan tambahkan ke daftar  
        b = bin(i - mi)[2:].zfill(length) # Sesuaikan panjangnya dengan zfill  
        enc.append([int(n) for n in b]) # Tambahkan ke daftar encoding  
  
    return enc  
  
def bin_dec(enc, mi=0):  
    lbl = []  
  
    for e in enc:  
        # Bulatkan nilai-nilai biner dan ubah kembali ke bentuk string  
        rounded = [int(round(x)) for x in e]  
        string = ''.join(str(x) for x in rounded)  
        num = int(string, 2) + mi # Konversi string biner ke angka desimal dan tambahkan mi  
        lbl.append(num)  
  
    return lbl
```

### b. Percobaan Binary Encoding dan Decoding

### ▼ b) Percobaan *Binary Encoding* dan *Decoding*

Tulis kode ke dalam *cell* di bawah ini:

```
✓ 0s [2] labels = 1, 2, 3, 4
enc = bin_enc(labels)
dec = bin_dec(enc, min(labels))
print(enc)
print(dec)
```

```
→ [[0, 0, 0], [0, 0, 1], [0, 1, 0], [0, 1, 1]]
[1, 2, 3, 4]
```

225150207111040  
Raikhan Geza Alburama

### c. Fungsi One-hot Encoding dan Decoding

#### ▼ c) Fungsi *One-hot Encoding* dan *Decoding*

Tulis kode ke dalam *cell* di bawah ini:

```
✓ 0s [3] import numpy as np

def onehot_enc(lbl, min_val=0):
    mi = min(lbl) # Nilai minimum dari daftar label
    # Buat array dengan ukuran (jumlah label, jumlah kategori) dan diisi dengan min_val
    enc = np.full((len(lbl), max(lbl) - mi + 1), min_val, np.int8)

    for i, x in enumerate(lbl):
        enc[i, x - mi] = 1 # Set nilai yang sesuai ke 1 (one-hot)

    return enc

def onehot_dec(enc, mi=0):
    # Kembalikan indeks maksimum dari setiap baris (yang mewakili label asli)
    return [np.argmax(e) + mi for e in enc]
```

225150207111040  
Raikhan Geza Alburama

d. Percobaan Binary Encoding dan Decoding

▼ d) Percobaan *Binary Encoding* dan *Decoding*

Tulis kode ke dalam *cell* di bawah ini:

```
0s ▶ labels = 1, 2, 3, 4
enc = onehot_enc(labels)
dec = onehot_dec(enc, min(labels))
print(enc)
print(dec)
```

```
[[1 0 0 0]
 [0 1 0 0]
 [0 0 1 0]
 [0 0 0 1]]
[1, 2, 3, 4]
```

225150207111040  
Raikhan Geza Alburama

e. Fungsi Aktivasi Sigmoid dan Derivatifnya

▼ e) Fungsi Aktivasi Sigmoid dan Derivatifnya

Tulis kode ke dalam *cell* di bawah ini:

```
0s ▶ def sig(X):
    return [1 / (1 + np.exp(-x)) for x in X]
def sigd(X):
    output = []
    for i, x in enumerate(X):
        s = sig([x])[0]
        output.append(s * (1 - s))
    return output
```

225150207111040  
Raikhan Geza Alburama

### C. Analisis

1. Download dataset Iris dalam format CSV di <https://datahub.io/machine-learning/iris/r/iris.csv> .n berhenti?

```
import pandas as pd

from google.colab import files
uploaded = files.upload()

df = pd.read_csv("iris_csv.csv")
df.head()
```

1. Download dataset Iris dalam format CSV di <https://datahub.io/machine-learning/iris/r/iris.csv> .

```
[7] import pandas as pd
```

```
from google.colab import files
uploaded = files.upload()
```

Choose Files iris\_csv.csv

- iris\_csv.csv(text/csv) - 4753 bytes, last modified: 10/28/2024 - 100% done

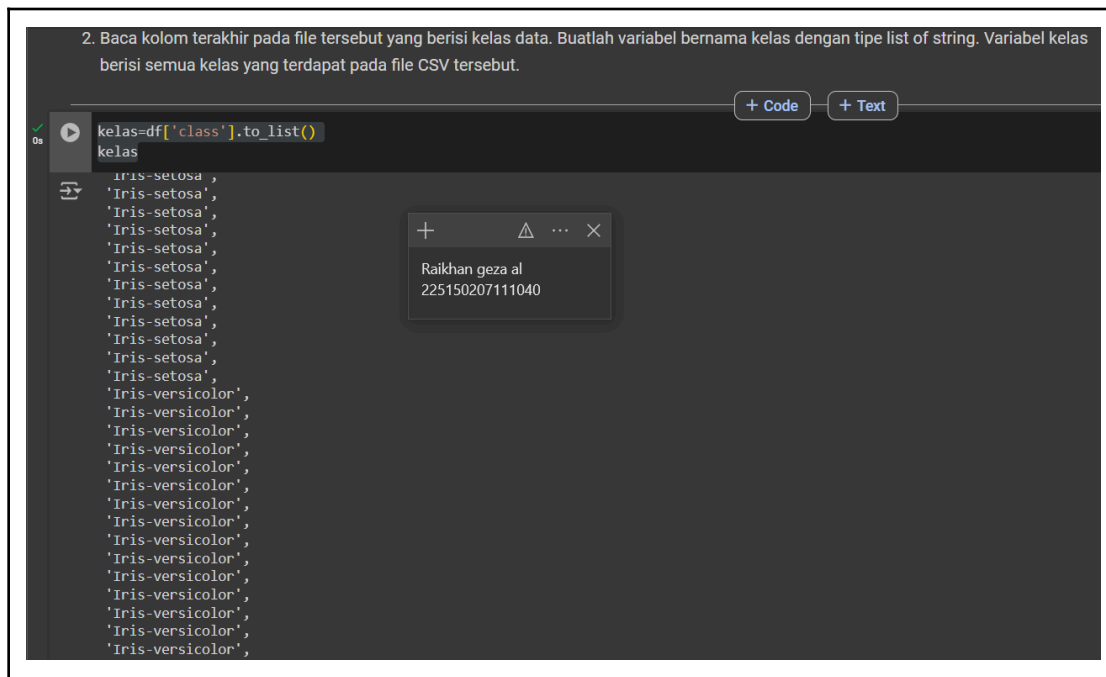
Saving iris\_csv.csv to iris\_csv.csv

```
[9] df=pd.read_csv("iris_csv.csv")
df.head()
```

	sepalwidth	sepalwidth	petalwidth	petalwidth	class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

2. Baca kolom terakhir pada file tersebut yang berisi kelas data. Buatlah variabel bernama kelas dengan tipe list of string. Variabel kelas berisi semua kelas yang terdapat pada file CSV tersebut.

```
kelas=df['class'].to_list()
kelas
```



3. Buatlah fungsi bernama `bin_enc_str` yang berfungsi untuk melakukan binary encoding pada string. Fungsi ini menerima input berupa list of string dan menghasilkan output berupa representasi binary encoding dari list tersebut. Jangan lupa membuat fungsi decodernya juga dengan nama `bin_dec_str`

```
import pandas as pd

def bin_enc_str(kelas):
    lbl_num = pd.factorize(kelas)[0]
    lbl_rev = pd.factorize(kelas)[1]
    mi = min(lbl_num)
    length = len(bin(max(lbl_num) - mi + 1)[2:])
    enc = []
    for i in lbl_num:
        b = bin(i - mi)[2:].zfill(length)
        enc.append([int(n) for n in b])
    return enc, lbl_rev

def bin_dec_str(enc, lbl_rev, mi=0):
    lbl = []
    lbl_str = []
    for en in enc:
        rounded = [int(round(x)) for x in en]
        string = ''.join(str(x) for x in rounded)
        num = int(string, 2) + mi
```

```

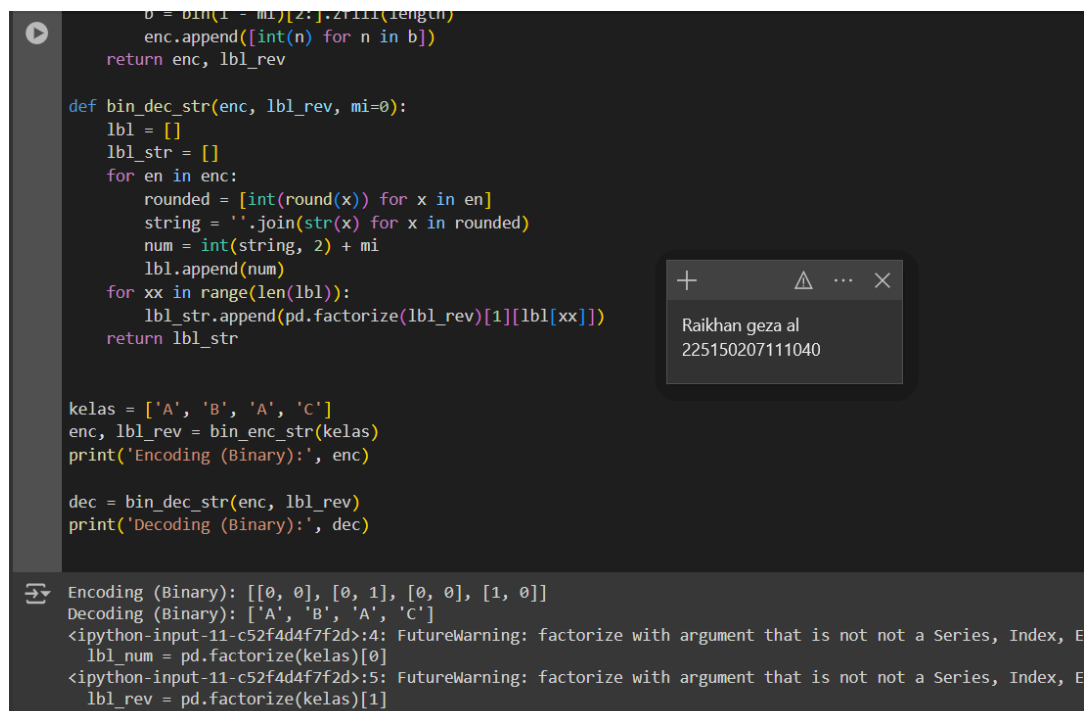
        lbl.append(num)
    for xx in range(len(lbl)):

lbl_str.append(pd.factorize(lbl_rev)[1][lbl[xx]])
    return lbl_str

kelas = ['A', 'B', 'A', 'C']
enc, lbl_rev = bin_enc_str(kelas)
print('Encoding (Binary):', enc)

dec = bin_dec_str(enc, lbl_rev)
print('Decoding (Binary):', dec)

```



```

0 = bin(1 - mi)[2:].zfill(length)
enc.append([int(n) for n in b])
return enc, lbl_rev

def bin_dec_str(enc, lbl_rev, mi=0):
    lbl = []
    lbl_str = []
    for en in enc:
        rounded = [int(round(x)) for x in en]
        string = ''.join(str(x) for x in rounded)
        num = int(string, 2) + mi
        lbl.append(num)
    for xx in range(len(lbl)):
        lbl_str.append(pd.factorize(lbl_rev)[1][lbl[xx]])
    return lbl_str

kelas = ['A', 'B', 'A', 'C']
enc, lbl_rev = bin_enc_str(kelas)
print('Encoding (Binary):', enc)

dec = bin_dec_str(enc, lbl_rev)
print('Decoding (Binary):', dec)

```

Encoding (Binary): [[0, 0], [0, 1], [0, 0], [1, 0]]  
Decoding (Binary): ['A', 'B', 'A', 'C']  
<ipython-input-11-c52f4d4f7f2d>:4: FutureWarning: factorize with argument that is not not a Series, Index, E  
 lbl\_num = pd.factorize(kelas)[0]  
<ipython-input-11-c52f4d4f7f2d>:5: FutureWarning: factorize with argument that is not not a Series, Index, E  
 lbl\_rev = pd.factorize(kelas)[1]



4. Buatlah fungsi bernama `onehot_enc_str` yang berfungsi untuk melakukan one-hot encoding pada string. Fungsi ini menerima input berupa list of string dan menghasilkan output berupa representasi one-hot encoding dari list tersebut. Jangan lupa membuat fungsi decodernya juga dengan nama `onehot_dec_str`

```
def onehot_enc_str(lbl, min_val=0):
    lbl_num = pd.factorize(lbl)[0]
    lbl_rev = pd.factorize(lbl)[1]
    mi = min(lbl_num)
    enc = np.full((len(lbl_num), max(lbl_num) - mi + 1), min_val, np.int8)
    for i, x in enumerate(lbl_num):
        enc[i, x - mi] = 1
    return enc, lbl_num, lbl_rev

def onehot_dec_str(enc, lbl_rev, mi=0):
    lbl = []
    dec_num = []
    for e in enc:
        dec_num.append(np.argmax(e) + mi)
    for xx in range(len(dec_num)):
        lbl.append(pd.factorize(lbl_rev)[1][dec_num[xx]])
    return lbl

kelas = ['A', 'B', 'A', 'C']
enc_oh, lbl_num, lbl_rev = onehot_enc_str(kelas)
print('Encoding (Onehot):\n', enc_oh)

dec_one_h = onehot_dec_str(enc_oh, lbl_rev,
min(lbl_num))
print('Decoding (Onehot):\n', dec_one_h)
```

```
def onehot_enc_str(lbl_num):
    for i, x in enumerate(lbl_num):
        enc[i, x - mi] = 1
    return enc, lbl_num, lbl_rev

def onehot_dec_str(enc, lbl_rev, mi=0):
    lbl = []
    dec_num = []
    for e in enc:
        dec_num.append(np.argmax(e) + mi)
    for xx in range(len(dec_num)):
        lbl.append(pd.factorize(lbl_rev)[1][dec_num[xx]])
    return lbl

kelas = ['A', 'B', 'A', 'C']
enc Oh, lbl_num, lbl_rev = onehot_enc_str(kelas)
print('Encoding (Onehot):\n', enc Oh)

dec One_h = onehot_dec_str(enc Oh, lbl_rev, min(lbl_num))
print('Decoding (Onehot):\n', dec One_h)
```

Encoding (Onehot):  
[[1 0 0]  
[0 1 0]  
[1 0 0]  
[0 0 1]]

Decoding (Onehot):  
['A', 'B', 'A', 'C']

<ipython-input-12-dfcf7ac23e61>:2: FutureWarning: factorize with argument that is not not a Series, Index, ExtensionArray  
lbl\_num = pd.factorize(lbl)[0]  
<ipython-input-12-dfcf7ac23e61>:3: FutureWarning: factorize with argument that is not not a Series, Index, ExtensionArray  
lbl\_rev = pd.factorize(lbl)[1]

#### D. Kesimpulan

Backpropagation adalah algoritme pelatihan untuk jaringan saraf yang bekerja dengan memperbarui bobot melalui propagasi balik dari kesalahan. Dalam proses ini, kesalahan dihitung di lapisan output dan didistribusikan kembali ke lapisan-lapisan sebelumnya untuk memperbaiki bobot berdasarkan gradien. Agar backpropagation dapat berfungsi dengan benar, encoding diperlukan karena jaringan saraf hanya bekerja dengan data numerik. Label atau data kategorikal harus diubah ke dalam format angka agar jaringan dapat memprosesnya dan menghitung kesalahan dengan tepat.

Beberapa jenis encoding yang umum digunakan meliputi one-hot encoding, di mana setiap kelas diwakili oleh vektor biner dengan satu nilai 1 dan sisanya 0, binary encoding yang mengubah kategori menjadi representasi biner, serta label encoding yang menggantikan setiap kategori dengan angka urut. One-hot encoding sering digunakan untuk klasifikasi multikategori, sementara binary encoding lebih efisien untuk data dengan banyak kategori dan lebih sedikit dimensi.