Praktikum 6
Algoritma Klasifikasi SVM

Tujuan

Setelah mengikuti praktikum ini, mahasiswa diharapkan mampu:

1. Mengetahui konsep dasar algoritma SVM
2. Mengimplementasikan algoritma SVM berbasis Stochastic Gradient Descent

Dasar Teori

Support Vector Machine (SVM) merupakan sebuah algoritma klasifikasi yang mampu mengenali dua kelas dalam data. Dengan demikian, SVM termasuk dalam algoritma klasifikasi biner, karena hanya mampu mengenali dua kelas, yaitu positif dan negatif. Cara kerja SVM adalah mencari *hyperplane* dengan fungsi $wx + b = 0$ yang berfungsi sebagai garis pemisah antar dua kelas. Kelas positif akan memiliki nilai $wx + b \geq 1$ sedangkan kelas negatif memiliki nilai $wx + b \leq 1$.

SVM mencari *hyperplane* optimal, yaitu *hyperplane* yang memiliki *cost function* minimal. *Cost function* digunakan untuk mengukur seberapa baik suatu *hyperplane*. Terdapat banyak *cost function* yang dapat digunakan pada SVM, salah satunya adalah:

$$J(w) = \frac{1}{2}||w||^2 + C\left[\frac{1}{N}\sum_{i}^{n} \quad max(0,1 - y_i * (w.x_i + b)\right]$$

Keterangan:
J(w) : Cost function dari hyperplane w
C : Parameter regularization
N : Banyak data
y : label data
x : data

Optimasi *hyperplane* memerlukan perhitungan gradien dari *cost function*. Persamaan gradien dari *cost function* yang digunakan adalah

$$\nabla_w J(w) = \frac{1}{N}\sum_{i}^{n} \begin{cases} w & , \text{jika } max(0, 1 - y_i * (w.x_i)) = 0 \\ w - Cy_i x_i & \text{lainnya} \end{cases}$$

Nilai gradien dari *cost function* digunakan oleh algoritma Stochastic Gradient Descent (SGD) untuk melakukan minimasi nilai *cost function*. Cara kerja algoritma SGD secara umum adalah:

1. Hitung nilai gradien dari *cost function.*
2. Lakukan perubahan bobot dengan arah yang berlawanan terhadap nilai gradien.
3. Ulangi langkah 1-2 sampai konvergen.

Tahapan pelatihan pada SVM dilakukan dengan cara optimasi *cost function*, salah satunya dengan SGD. Tahapan pengujian dilakukan dengan menghitung nilai dot product antara bobot dengan data uji. Selanjutnya, kelas data uji ditentukan berdasarkan tanda (positif atau negatif) dari nilai dot product.

Praktikum

1. Import Data

Unduh dataset yang akan digunakan pada praktikum kali ini. Anda dapat menggunakan aplikasi wget untuk mendowload dataset dan menyimpannya dalam Google Colab. Jalankan cell di bawah ini untuk mengunduh dataset

```
! wget
https://gist.githubusercontent.com/netj/8836201/raw/6f9306ad21398
ea43cba4f7d537619d0e07d5ae3/iris.csv
```

Setelah dataset berhasil diunduh, langkah berikutnya adalah membaca dataset dengan memanfaatkan fungsi readcsv dari library pandas. Lakukan pembacaan berkas csv ke dalam dataframe dengan nama data menggunakan fungsi readcsv. Jangan lupa untuk melakukan import library pandas terlebih dahulu

```
import pandas as pd
import numpy as np
data = pd.read_csv('iris.csv')
```

Cek isi dataset Anda dengan menggunakan perintah **head()**

```
data.head()
```

Metode Support Vector Machine (SVM) dasar hanya mampu melakukan klasifikasi data yang terdiri dari dua kelas, atau disebut klasifikasi biner. Dataset iris memiliki 3 kelas, sehingga salah satu kelasnya, yaitu Iris-virginica perlu dihapus. Cell berikut menghapus data yang memiliki kelas Iris-virginica

```
data.drop(data[data['variety']=='Virginica'].index,inplace=True)
```

SVM juga memiliki keterbatasan, yaitu hanya mampu menerima kelas dalam bentuk numerik. Cell berikut mengubah kelas menjadi bilangan, Iris-setosa menjadi -1 dan Iris-versicolor menjadi 1

```
data['variety']=data['variety'].map({'Setosa':-1,'Versicolor':1})
```

Tampilkan beberapa data untuk mengecek hasilnya

```
data.head()
```

2. Membagi Data Menjadi Data Latih dan Data Uji

Metode pembelajaran mesin memerlukan dua jenis data :

1. Data latih : Digunakan untuk proses training metode klasifikasi
2. Data uji : Digunakan untuk proses evaluasi metode klasifikasi

Data uji dan data latih perlu dibuat terpisah (mutualy exclusive) agar hasil evaluasi lebih akurat. Data uji dan data latih dapat dibuat dengan cara membagi dataset dengan rasio tertentu, misalnya 80% data latih dan 20% data uji.

Library Scikit-learn memiliki fungsi train_test_split pada modul model_selection untuk membagi dataset menjadi data latih dan data uji. Bagilah dataset anda menjadi dua, yaitu **data_latih** dan **data_uji**.

```
from sklearn.model_selection import train_test_split
data_latih,data_uji = train_test_split(data,test_size=0.2)
```

Parameter **test_size** pada fungsi **train_test_split** menunjukkan persentase data uji yang dihasilkan. Pada contoh ini, 20% data digunakan sebagai data uji.

Tampilkan banyaknya data pada **data_latih** dan **data_uji**. Seharusnya **data_latih** terdiri dari 80 data, dan **data_uji** terdiri dari 20 data

```
print(data_uji.shape[0])
print(data_latih.shape[0])
```

Pisahkan label/kategori dari data latih dan data uji menjadi variabel tersendiri. Beri nama variabelnya **label_latih** dan **label_uji.** Fungsi **pop()** pada dataframe berfungsi untuk mengambil suatu kolom dari dataframe, hampir sama dengan fungsi pop() pada struktur data stack.

```
label_latih = data_latih.pop('variety')
label_uji = data_uji.pop('variety')
```

3. Proses Training

Tujuan dari algoritma SVM adalah meminimalkan nilai *cost function*. Penghitungan nilai minimal dapat dapat dilakukan dengan menghitung nilai gradien dari *cost function* terlebih dahulu. Fungsi di bawah ini berguna untuk menghitung nilai gradien cost function seperti yang tertera pada bagian teori.

```
def hitung_cost_gradient(W,X,Y,regularization):
    jarak = 1 - (Y * np.dot(X,W))
    dw = np.zeros(len(W))
    if max(0,jarak)==0:
        di=W
    else:
        di = W - (regularization * Y * X)
    dw += di
    return dw
```

Terdapat beberapa cara untuk meminimalkan nilai cost function, salah satunya menggunakan Stochastic Gradient Descent (SGD) untuk melakukan minimasi. Minimasi cost function merupakan inti dari algoritma SVM. Fungsi di bawah ini merupakan implementasi algoritma SGD

```
from sklearn.utils import shuffle
def sgd(data_latih,label_latih,learning_rate =
0.000001,max_epoch=1000,regularization=10000):
  data_latih = data_latih.to_numpy()
  label_latih = label_latih.to_numpy()
  bobot = np.zeros(data_latih.shape[1])
  for epoch in range(1,max_epoch):
    X,Y =shuffle(data_latih,label_latih,random_state=101)
    for index,x in enumerate(X):
      delta=hitung_cost_gradient(bobot,x,Y[index],regularization)
      bobot = bobot - (learning_rate * delta)
  return bobot
```

Pada algoritma SGD, dilakukan perulangan sebanyak **max_epoch** dan dilakukan perhitungan bobot, yang merupakan parameter *hyperplane*. Pada setiap perulangan dilakukan pengacakan data, agar tidak terjadi pola perhitungan bobot yang sama. Nilai bobot diperoleh dari bobot dikurangi gradien dikali dengan *learning rate*. Nilai *learning rate* merupakan sebuah parameter yang mengatur seberapa besar perubahan bobot dilakukan.

Proses training dilakukan dengan memanggil fungsi **sgd** dengan parameter input berupa data latih dan label latih. Parameter learning rate dan max epoch menggunakan nilai *default* nya.

```
W = sgd(data_latih,label_latih)
print(W)
```

4. Proses Testing

Proses testing dilakukan dengan menghitung nilai *dot product* antara bobot hasil training dengan data uji. Kelas data ditentukan berdasarkan tanda (positif atau negatif) dari hasil dot product tersebut. Fungsi berikut mengimplementasikan proses testing.

```
def testing(W,data_uji):
    prediksi = np.array([])
    for i in range(data_uji.shape[0]):
        y_prediksi = np.sign(np.dot(W,data_uji.to_numpy()[i]))
        prediksi = np.append(prediksi,y_prediksi)
    return prediksi
```

Lakukan pengujian menggunakan seluruh data uji. Bandingkan hasilnya dengan nilai label sebenarnya untuk menghitung berapa banyak data uji yang berhasil diprediksi dengan benar.

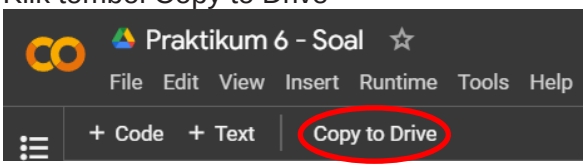
```
y_prediksi = testing(W,data_uji)
print(sum(y_prediksi==label_uji))
```

Tugas

Pada tugas kali ini Anda diminta melakukan klasifikasi SVM menggunakan data iris, tetapi menggunakan spesies dan parameter yang berbeda. Lengkapilah kerangka source code pada notebook yang tersedia dan jawablah pertanyaan yang ada.

Petunjuk pengerjaan soal:

1. Klik link berikut, pastikan Anda login menggunakan **akun student UB**. https://colab.research.google.com/drive/1rQgWJ0okP1mJETcePKVubBrNFN4e0-R7?usp=sharing
2. Klik tombol Copy to Drive



3. Beri nama file **BAB 6-Nama-NIM.ipynb**
4. Lengkapi nama, NIM, dan kelas Anda di bawah judul bab
5. Isilah cell yang kosong
6. Download file *.ipynb dengan cara klik **File -> Download .ipynb**
7. Kumpulkan file *.ipynb ke asisten (Google Classroom)