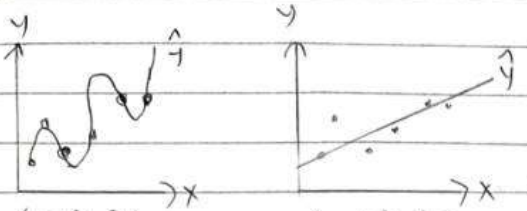
Edgrant H.S.

2206021016

Kelompok 1:
- Edgrant : 1a, 1b, 1c
- George : 1d, 1e, 2a
- Evandita
- M. Billie 2b, 2c, 2d

1. Regularisasi

a. Definisi dan Tujuan



Grafik a.1

Grafik a.2

$\hat{y} = \theta_0 + \theta_1 x + \theta_2 x^2 + \dots + \theta_{10} x^{10}$; $\hat{y} = \theta_0 + \theta_1 x$

Dari grafik a.1, dapat dilihat grafik/model $\hat{y}$ sangat menyesuaikan pada data. Namun, hal ini sebenarnya tidak baik karena data pada dunia nyata pasti rentan terhadap noise. Jika model digunakan untuk prediksi data baru, dapat menghasilkan output yang salah dibanding model sederhana seperti pada grafik a.2

Contoh pada grafik a.1

Jika $\theta_{10} = 1$

↳ perubahan kecil pada x

↳ perubahan besar terhadap x^{10}

↳ model terlalu sensitif

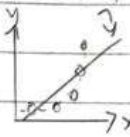
Contoh pada grafik a.2

Perubahan kecil pada x

↳ perubahan kecil pada $\theta_1 x$

↳ model tidak terlalu sensitif

Salah satu solusinya adalah menggunakan model sederhana, tapi jika data seperti ini:



model sederhana linear tidak memodelkan dengan baik.

Jadi, kita ingin model yang tidak sensitif tetapi juga dapat memodelkan jika data complex → hal ini dapat dicapai dengan regularisasi

Selain itu, model sensitif sering menimbulkan model yang overfit, atau terlalu "menghafal" data training. Regularisasi dapat mengurangi overfitting

b. L_1, L_2, dan Dropout

Secara overview

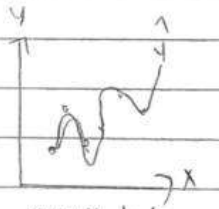
↳ L_1 : menghilangkan / membuat jadi 0 untuk weight yang tidak penting dan meminimalkan sisanya.

↳ Dropout : Menghilangkan weight pada Neural Network.

↳ L_2 : Meminimalkan / membuat kecil weight tetapi tidak 0.

L_1 dan L_2 mirip, tetapi L_1 dapat membuat weight menjadi 0, L_2 tidak. Alasannya akan dibahas pada nomor 3.

Contoh:

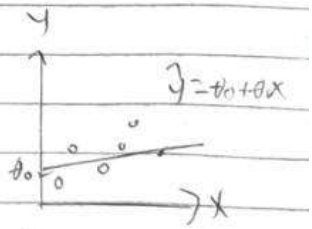


Grafik b.1

Grafik ini overfit

L_1

$\theta \begin{cases} \theta_{10} \to 0 \\ \vdots \\ \theta_2 \to 0 \\ \theta_1 \to \text{kecil} \end{cases} \to$

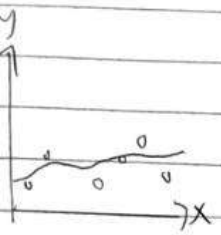


$\theta_0 \to$ tidak efek y-intercept

Dengan θ_1 yang kecil, model grafik akan lebih landai :) y lebih tidak sensitif terhadap y

L_2

$\theta \begin{cases} \theta_{10} \to \text{kecil} \\ \vdots \\ \theta_2 \to \text{kecil} \\ \theta_1 \to \text{kecil} \end{cases}$

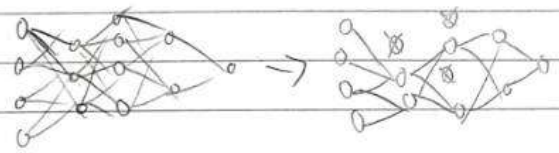


L_2 menyebabkan model kompleks tetapi weightnya rendah → tidak overfit

L_1 → bagus jika mayoritas fitur tidak mempengaruhi model

L_2 → bagus jika mayoritas fitur mempengaruhi model

Dropout merupakan jenis regularisasi yang berbeda dengan L_1/L_2 dengan menghilangkan node pada Nueral Network



Persamaan untuk update weight adalah

$$\theta_i = \theta_i - \alpha \frac{\delta L(\theta)}{\delta \theta_i}$$

$$\theta_i = \theta_i - \alpha \left(\frac{\delta ((y(\theta) - \hat{y}(\theta))^2)}{\delta \theta_i} + \frac{\delta R(\theta)}{\delta A_i} \right)$$

Namun, dropout adalah algoritma, bukan persamaan.

1. Pada hidden layer, membuat weight beberapa node secara acak bernilai 0
2. Training forward pass & backward pass
3. Mengulang langkah 1 dan 2 dengan node dan layer yang berbeda
4. Melakukan cross validation dan mengambil model paling bagus

C. Persamaan Matematis

Regularisasi diterapkan pada loss function seluruh model.

$$L = L_{data} + R(\theta)$$

↳ regularisasi

Contohnya untuk regresi

$$L_i = (\hat{y}_i - y_i)^2 + R(\theta)$$

L_1 memiliki persamaan

$$R_{L_1}(\theta) = \lambda \sum_{i=1}^{n} |\theta_i|$$

↳ Mengambil total dari setiap θ yang mutlak

L_2 memiliki persamaan

$$R_{L_2}(\theta) = \lambda \sum_{i=1}^{n} (\theta_i)^2$$

↳ Mengambil total dari setiap kuadrat θ

λ adalah hyperparameter seberapa pengaruh regularisasi terhadap loss function

Cara optimasi: model tetap sama, mengurangi cost Function / loss function. Ini dapat dengan gradient descent

1.1. Soal Regulasi

1. Regularisasi (L1 vs. L2 vs. Dropout)

d. * Screenshot program L1 Regularization :

```
from sklearn.linear_model import Lasso
from sklearn.model_selection import train_test_split
from sklearn.datasets import make_regression
from sklearn.metrics import mean_squared_error

X, y = make_regression(n_samples=100, n_features=5, noise=0.1, random_state=42)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

lasso = Lasso(alpha=0.1)
lasso.fit(X_train, y_train)

y_pred = lasso.predict(X_test)

mse = mean_squared_error(y_test, y_pred)
print(f"Mean Squared Error: {mse}")

print("Coefficients:", lasso.coef_)

Mean Squared Error: 0.06362420021332456
Coefficients: [60.50305581 98.52475354 64.3920265  56.96061238 35.52028502]
```

Program di atas membentuk model dengan L1 Regularization (Lasso). Fungsi Lasso (alpha = 0.1) menentukan bahwa hyperparameter $\lambda = 0.1$ untuk rumus Cost Function :

$$Cost = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda \sum_{i=1}^{m} |w_i|$$

Selain itu fungsi Lasso akan menekan koefisien node / neuron menjadi 0, sebagai feature selection.

* Screenshot program L2 Regularization :

```
from sklearn.linear_model import Ridge
from sklearn.datasets import make_regression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

X, y = make_regression(n_samples=100, n_features=5, noise=0.1, random_state=42)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

ridge = Ridge(alpha=1.0)
ridge.fit(X_train, y_train)
y_pred = ridge.predict(X_test)

mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)
print("Coefficients:", ridge.coef_)

Mean Squared Error: 4.316850771972589
Coefficients: [59.82954832 97.15093898 61.24168738 56.35996833 35.18901136]
```

Hampir sama dengan L1, Fungsi Ridge digunakan untuk membentuk model dengan L2 Regularization (Ridge). alpha yang diinput pada Ridge (alpha = 0.1) adalah hyperparameter λ yang ada pada rumus :

$$Cost = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda \sum_{i=1}^{m} w_i^2$$

Pada L2, penalti w dikuadratkan untuk membuat bobot menjadi kecil tapi tidak 0 untuk tetap menghitung dampak dari semua fitur.

* Screenshot program Dropout Regularization :

```
from sklearn.datasets import make_regression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.optimizers import Adam
import numpy as np

# Data
X, y = make_regression(n_samples=100, n_features=5, noise=0.1, random_state=42)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Neural network with Dropout
model = Sequential()
model.add(Dense(64, activation='relu', input_shape=(X_train.shape[1],)))
model.add(Dropout(0.5))  # Dropout applied here
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))  # Dropout again
model.add(Dense(1))  # Output layer

# Compile model
model.compile(optimizer=Adam(learning_rate=0.01), loss='mse')

# Train
model.fit(X_train, y_train, epochs=100, batch_size=10, verbose=0)

# Predict
y_pred = model.predict(X_test).flatten()
mse = mean_squared_error(y_test, y_pred)
```

Program Dropout Regularization pertama - tama membangun neural network seperti program L1 dan L2.

Kemudian menambahkan fungsi Dropout (0.5) pada model yang mematikan 0.5 atau 50% neuron pada setiap layer sebelum melakukan training. Keacakan yang dilakukan diharapkan dapat menghasilkan model yang lebih akurat.

e. Kriteria pertimbangan untuk menggunakan L1, L2, maupun Dropout adalah tipe data, ukuran dataset, struktur dataset, struktur model, tingkat overfitting, serta tujuan akhir dari model tersebut.

- L1 (Lasso) sebaiknya digunakan ketika terdapat banyak fitur, namun hanya sebagian kecil dari fitur tersebut yang dianggap penting, karena L1 akan menyederhanakan model dengan menjadikan fitur tidak penting dari model tersebut. L1 akan secara otomatis melakukan feature selection dan menghilangkan noise.
- L2 (Ridge) sebaiknya digunakan ketika semua fitur dianggap relevan dan kita menginginkan hasil akurat tanpa menghapus fitur, karena berbeda dengan L1, L2 hanya mengurangi bobot dari fitur yang tidak terlalu penting tanpa menjadikan fitur tersebut. L2 juga tahan terhadap noise, namun fitur yang tetap dipertahankan dapat menyulitkan interpretasi dari model tersebut.
- Dropout sebaiknya digunakan ketika dataset yang digunakan terbatas, namun kapasitas model terbatas. Dropout akan mematikan neuron secara acak sehingga dapat membantu model mencapai representasi yang lebih stabil dan tersebar.

2. Metode Regularisasi Tambahan & Ensembles

a. ✱Data Augmentation adalah proses membuat data training baru dengan memodifikasi data yang sudah ada tanpa mengubah labelnya untuk menambahkan variasi dan jumlah data secara efektif tanpa mengumpulkan data baru.
Data augmentation termasuk dalam regularisasi karena menambahkan pola pelatihan pada model, sehingga model mempelajari fitur yang lebih general dan mengurangi kemungkinan overfitting.

✱ Early Stopping adalah perhentian proses training yang lebih awal sebelum model mencapai loss minimum, ketika deviation/validation loss mulai memburuk.
Early Stopping juga dianggap sebagai regularisasi karena mencegah model mempelajari terlalu banyak data noise saat training.

✱ Model Ensemble adalah teknik penggabungan beberapa model berbeda dan diambil gabungan hasil prediksinya.
Model Ensemble dianggap sebagai teknik regularisasi karena mengurangi risiko overfitting dari satu model saja dengan menggabungkan banyak model, sehingga hasil lebih stabil.

Date

2. Kenapo Una mencoba Data augmentation dahulu?
dilakukan lebih awal karena:
- Tidak menambah kompleksitas ~~model~~ model atau waktu inference
- langsung memperkaya data tanpa mengubah proses pelatihan. secara signifikan
- Efektif dalam meningkatkan generalisasi, terutama pada data visual

3. Keuntungan utama ensemble & kapan efektif
2ID Ensemble meningkatkan akurasi dan stabilitas, karena menggabungkan banyak model yang belajar dengan cara berbeda.

Sangat berguna ketika:
- Model individual cukup baik tapi overfit.
- Ada cukup resource komputasi
- Kompetisi ML atau sistem production yang menuntut prediksi sangat akurat dan andal

4. Prioritas penerapan teknik regulasi
a) Data Augmentation - murah & efektif
b) Early stopping - Mudah diterapkan, hemat waktu pelatihan
g) Ensemble - mahal secara komputasi, dilakukan setelah model cukup baik

Kriteria pemilihan bergantung pada: Jenis data, kapasitas model, ketersediaan data, dan batasan komputasi

LOCOMOTIF