Chapter 4: Training Models

Bab ini berfokus pada proses inti pelatihan model machine learning, terutama model linear, serta membahas konsep optimisasi dan fungsi kerugian (loss function). Ini adalah dasar dari banyak algoritma machine learning modern, termasuk neural networks.

A. Tujuan Bab Ini

- Memahami bagaimana model machine learning belajar dari data
- Memahami cara kerja Linear Regression, Gradient Descent, dan Regularization
- Belajar tentang fungsi biaya, gradien, dan bagaimana meminimalkan error

B. Topik Utama Bab Ini

1. Linear Regression

Model paling sederhana: memprediksi target sebagai kombinasi linear dari fitur.

Bentuk umum:

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n$$

Implementasi:

Dengan Normal Equation:

```
theta_best = np.linalg.inv(X.T.dot(X)).dot(X.T).dot(y)
Dengan Scikit-Learn:
from sklearn.linear_model import LinearRegression
lin_reg = LinearRegression()
lin_reg.fit(X, y)
```

2. Gradient Descent (GD)

Metode numerik untuk menemukan parameter optimal dengan meminimalkan fungsi biaya.

Dimulai dari tebakan awal dan bergerak menuju minimum dengan langkah-langkah kecil.

- Fungsi biaya Linear Regression:

$$MSE(\theta) = (1/m) \sum (\hat{y}_i - y_i)^2$$

- Jenis-jenis Gradient Descent:

Batch GD: Menggunakan seluruh data tiap langkah

Stochastic GD (SGD): Menggunakan satu data per langkah (cepat tapi bising)

Mini-batch GD: Kombinasi keduanya

- Parameter penting:

Learning rate: menentukan seberapa besar langkah per iterasi

Terlalu kecil → lambat

Terlalu besar → bisa gagal konvergen

3. Polynomial Regression

Menambahkan fitur berpangkat (x^2, x^3, dll) untuk menangani data nonlinear.

Bisa menyebabkan overfitting → perlu regularisasi

4. Regularization

Tujuannya: menghindari overfitting dengan menghukum model yang terlalu kompleks.

- Ridge Regression (L2)

Menambahkan penalti L2 ke fungsi biaya:

$J(\theta) = MSE(\theta) + \alpha \sum \theta_j^2$

- Lasso Regression (L1)

Penalti L1 → bisa membuat beberapa koefisien menjadi nol → seleksi fitur otomatis

$J(\theta) = MSE(\theta) + \alpha \sum |\theta_j|$

- Elastic Net

Gabungan L1 dan L2, cocok ketika ada banyak fitur korelasi

C. Evaluasi dan Visualisasi

Menggunakan learning curves untuk melihat apakah model overfitting atau underfitting

Pelajari bagaimana model belajar seiring bertambahnya data

D. Proyek/Contoh di Bab Ini

Menggunakan dataset sederhana (misalnya housing) untuk menerapkan:

- Linear Regression
- Polynomial Regression
- Ridge/Lasso/ElasticNet Regression
- Perbandingan performa dengan train/test set dan visualisasi learning curve

E. Insight Penting

- Gradient Descent adalah inti dari hampir semua machine learning modern, termasuk deep learning.
- Regularisasi sangat penting untuk model kompleks agar tetap generalisasi.
- Learning curve sangat membantu untuk diagnosis underfitting/overfitting.
- Linear regression adalah alat dasar yang kuat dan menjadi landasan bagi model yang lebih kompleks.

Notebook: 04_training_linear_models.ipynb

A. Tujuan Notebook

Menunjukkan secara praktis bagaimana:

- Linear Regression dilatih
- Gradient Descent bekerja

- Regularisasi mencegah overfitting

BAGIAN PER BAGIAN

1. Setup & Imports

import numpy as np

import matplotlib.pyplot as plt

Library yang digunakan untuk numerik dan visualisasi.

2. Generate Data (Linear)

np.random.seed(42)

X = 2 * np.random.rand(100, 1)

y = 4 + 3 * X + np.random.randn(100, 1)

Penjelasan:

Membuat dataset sintetik linear: y = 4 + 3x + noise

Cocok untuk latihan Linear Regression

3. Analytical Solution (Normal Equation)

X_b = np.c_[np.ones((100, 1)), X] # add bias term (x_0 = 1)

theta_best = np.linalg.inv(X_b.T.dot(X_b)).dot(X_b.T).dot(y)

Penjelasan:

Ini solusi eksak dari Linear Regression.

Menggunakan formula Normal Equation.

4. Predicting

X_new = np.array([[0], [2]])

X_new_b = np.c_[np.ones((2, 1)), X_new]

y_predict = X_new_b.dot(theta_best)

Tujuan:

Memprediksi y untuk x=0 dan x=2

Menampilkan garis regresi

5. Visualisasi Hasil

plt.plot(X_new, y_predict, ...)

Menampilkan:

Titik data (scatter)

Garis regresi (line)

6. Linear Regression dengan Scikit-Learn

from sklearn.linear_model import LinearRegression

lin_reg = LinearRegression()

lin_reg.fit(X, y)

lin_reg.intercept_, lin_reg.coef_

Penjelasan:

Membangun model menggunakan library siap pakai

Sama dengan hasil dari Normal Equation

7. Gradient Descent Manual

eta = 0.1 # learning rate

n_iterations = 1000

m = 100

theta = np.random.randn(2,1)

```
for iteration in range(n_iterations):
   gradients = 2/m * X_b.T.dot(X_b.dot(theta) - y)
   theta = theta - eta * gradients
```

Penjelasan:

Manual SGD (full batch) untuk meminimalkan MSE

Menunjukkan bagaimana theta berubah tiap iterasi

8. Stochastic Gradient Descent (SGD)

```
from sklearn.linear_model import SGDRegressor
sgd_reg = SGDRegressor(max_iter=1000, penalty=None, eta0=0.1)
sgd_reg.fit(X, y.ravel())
```

Penjelasan:

Menggunakan Scikit-Learn untuk melakukan SGD otomatis

eta0 adalah learning rate awal

9. Polynomial Regression

```
from sklearn.preprocessing import PolynomialFeatures
poly_features = PolynomialFeatures(degree=2, include_bias=False)
X_poly = poly_features.fit_transform(X)
```

Penjelasan:

Menambahkan fitur x^2 ke data untuk menangkap nonlinearitas

10. Regularized Models

Ridge Regression (L2)

```
from sklearn.linear_model import Ridge
```

ridge_reg = Ridge(alpha=1, solver="cholesky")

- Lasso Regression (L1)

from sklearn.linear_model import Lasso

lasso_reg = Lasso(alpha=0.1)

- Elastic Net (L1 + L2)

from sklearn.linear_model import ElasticNet

elastic_net = ElasticNet(alpha=0.1, l1_ratio=0.5)

Penjelasan:

Semua model ini adalah variasi dari Linear Regression yang menambahkan penalti agar model tidak overfitting.

11. Early Stopping (Bonus Konsep)

Menggunakan SGDRegressor bersama warm_start=True dan validasi set:

Berhenti training jika validasi error naik → menghindari overfitting

B. Inti Pelajaran dari Notebook Ini:

Konsep	Penjelasan
Linear Regression	Belajar prediksi dengan kombinasi linear dari fitur
Gradient Descent	Metode numerik untuk menemukan parameter terbaik
Overfitting	Saat model terlalu cocok dengan training set, buruk untuk generalisasi
Regularization	Strategi mengurangi kompleksitas model
Polynomial Regression	Linear model yang bisa menangkap pola nonlinear
Scikit-Learn tools	Menyediakan implementasi praktis dan efisien