Ibnu Fazril

1103201241

TK-44-G4

Chapter 00 Pytorch

```python
# Matriks_A
tensor_A = torch.tensor([[1, 2],
                         [3, 4],
                         [5, 6]], dtype=torch.float32)

# Matriks_B
tensor_B = torch.tensor([[7, 10],
                         [8, 11],
                         [9, 12]], dtype=torch.float32)
```

Menghilangkan fungsi torch.matmul karena tidak bisa melakukan perkalian terhadap matriks 3x2 dengan 3x2 dan hanya bisa melakukan perkalian jika matriks b di transpose menjadi matriks 2x3.

```python
import torch

# Create tensor (default on CPU)
tensor = torch.tensor([1, 2, 3])

# Move tensor to GPU (if available)
device = "cuda" if torch.cuda.is_available() else "cpu"
tensor_on_gpu = tensor.to(device)

# Move tensor back to CPU
tensor_on_cpu = tensor_on_gpu.cpu()

# Now you can convert it to NumPy array
numpy_array = tensor_on_cpu.numpy()

print(numpy_array)
```

Mengubah langsung kode yang error dengan membuat awalnya di cpu lalu dipindahkan ke gpu dan dibalikkan lagi ke cpu agar kode dapat berjalan tanpa error dikarenakan sebelumnya kodenya menggunakan fungsi numpy langsung pada gpu.

Chapter 00 PyTorch merupakan kumpulan perintah atau skrip yang menggunakan PyTorch untuk melakukan berbagai operasi dasar dengan tensor, mulai dari membuat tensor dengan berbagai dimensi hingga operasi matriks seperti perkalian matriks.

Berikut adalah beberapa poin utama dari kode tersebut:

- Pengenalan Tensor: Kode dimulai dengan mengenalkan tensor dan menyajikan contoh-contoh tensor dengan berbagai dimensi, termasuk skalar, vektor, matriks, dan tensor tiga dimensi.

- Operasi Dasar: Kode menyajikan operasi dasar seperti pengecekan dimensi, bentuk (shape), dan pembuatan tensor dengan nilai acak, nol, dan satu. Penggunaan fungsi torch.arange() juga ditunjukkan untuk membuat tensor dengan rentang nilai tertentu.

- Pengaturan Tipe Data dan Perangkat: Anda dapat melihat bagaimana mengatur tipe data dan perangkat (CPU atau GPU) untuk tensor.

- Operasi Matematika pada Tensor: Kode juga mencakup contoh operasi matematika pada tensor seperti penambahan, perkalian, dan operasi matematika lainnya. Ada juga contoh perkalian matriks.

- Penggunaan GPU (Jika Tersedia): Kode menunjukkan bagaimana memeriksa ketersediaan GPU, mengubah tensor ke GPU jika tersedia, dan mengembalikannya ke CPU.

- Operasi dengan NumPy: Kode menyajikan cara mengonversi antara tensor PyTorch dan array NumPy.

- Operasi Perbandingan dan Random: Terdapat bagian yang menunjukkan operasi perbandingan antara dua tensor dan penggunaan angka acak dengan seed untuk membuat tensor yang konsisten.

- Menampilkan Informasi Tertentu dari Tensor: Terdapat beberapa bagian yang menunjukkan cara mendapatkan informasi tertentu dari tensor, seperti nilai maksimum, minimum, dan lainnya.

- Manipulasi Dimensi Tensor: Ada contoh operasi seperti menambahkan dimensi, mengubah tampilan (view), menggabungkan tensor, menghapus dimensi tambahan, dan merubah urutan sumbu pada tensor.

- Operasi Matriks Linear dengan Model Linear PyTorch: Kode menunjukkan contoh penggunaan model linear PyTorch dengan matriks bobot.

- Operasi Tensor pada GPU: Ada beberapa bagian yang menunjukkan cara memeriksa ketersediaan GPU dan melakukan operasi pada GPU jika tersedia.

- Menampilkan Informasi GPU: Terdapat perintah untuk menampilkan informasi GPU menggunakan !nvidia-smi (asumsi CUDA dan NVIDIA GPU tersedia).

Kode tersebut mencakup berbagai konsep dasar PyTorch dan operasi tensor, serta memberikan pemahaman dasar tentang penggunaan PyTorch untuk komputasi tensor.

Chapter 01 PyTorch

Kode tersebut mencakup beberapa konsep dasar tentang penggunaan PyTorch untuk pembelajaran mesin (machine learning). Di bawah ini adalah ringkasan dari beberapa poin utama yang dicakup dalam kode tersebut:

Definisi Tugas dan Pendahuluan:

- Definisi Tugas: Kode dimulai dengan mendefinisikan apa yang akan dicakup, seperti persiapan dan penggunaan data, pembangunan model, pelatihan model, prediksi, evaluasi model, penyimpanan, dan pembuatan model end-to-end.
- Import Modul: Mengimpor modul PyTorch dan beberapa modul lain yang diperlukan.

Persiapan dan Penggunaan Data:

- Generate Data: Membuat data sintetis untuk digunakan dalam contoh, termasuk pembagian data menjadi set pelatihan dan pengujian.
- Visualisasi Data: Menggunakan matplotlib untuk memvisualisasikan data pelatihan dan pengujian.

Pembangunan Model:

- Linear Regression Model: Membangun model regresi linear sederhana dengan PyTorch, baik dengan menggunakan parameter kustom maupun dengan menggunakan layer nn.Linear.

Pelatihan Model:

- Loss dan Optimizer: Menggunakan loss function (Mean Absolute Error - MAE) dan optimizer (Stochastic Gradient Descent - SGD) untuk melatih model.
- Iterasi Epochs: Melakukan iterasi sejumlah epochs tertentu untuk melatih model, dengan mencatat loss pada set pelatihan dan pengujian.

Evaluasi Model:

- Evaluasi Hasil: Melakukan evaluasi model pada set pengujian dan membandingkannya dengan set pelatihan.
- Visualisasi Prediksi: Memvisualisasikan hasil prediksi model pada data pengujian.

Penyimpanan dan Pemuatan Model:

- Penyimpanan Model: Menyimpan model ke file menggunakan torch.save.
- Pemuatan Model: Memuat kembali model dari file yang disimpan.

Penggunaan Perangkat GPU (Opsional):

- Cek Ketersediaan GPU: Memeriksa apakah GPU tersedia.
- Memindahkan Tensor ke GPU: Jika GPU tersedia, kode menunjukkan cara memindahkan tensor ke GPU.

Penutup:

- Pembandingan Model Asli dan Dicatat: Menunjukkan perbedaan antara parameter model sebelum dan sesudah pelatihan.

Kode ini memberikan gambaran umum tentang langkah-langkah dasar yang terlibat dalam menggunakan PyTorch untuk melatih dan menguji model. Setiap bagian mencakup penjelasan komentar yang membantu dalam memahami tujuan dan fungsionalitasnya.

Chapter 02 PyTorch

Kode tersebut merupakan implementasi dari beberapa konsep dasar dalam penggunaan PyTorch untuk machine learning, khususnya pada tugas-tugas seperti klasifikasi dan regresi. Berikut adalah penjelasan dalam bahasa Indonesia:

Membuat Data: Kode dimulai dengan membuat data sintetis menggunakan make_circles dari sklearn.datasets. Data ini terdiri dari dua fitur (X1 dan X2) dan label biner (0 atau 1) yang menciptakan lingkaran.

Visualisasi Data: Data kemudian divisualisasikan menggunakan scatter plot untuk melihat distribusinya.

Pemodelan dengan PyTorch:

- Data diubah menjadi tensor PyTorch.
- Data dibagi menjadi set pelatihan dan pengujian.

- Model neural network (NN) untuk klasifikasi dibuat menggunakan PyTorch dengan struktur model yang berbeda (V0, V1, V2).
- Model dievaluasi menggunakan metrik loss (BCEWithLogitsLoss) dan optimizer (SGD).

Pelatihan dan Evaluasi Model:

- Proses pelatihan dan evaluasi dilakukan dengan loop sebanyak epoch yang ditentukan.
- Untuk setiap epoch, dilakukan langkah-langkah training dan testing, serta dicetak metrik loss dan akurasi.
- Visualisasi decision boundary model diplot setiap 10 epoch.

Regresi Linear dengan PyTorch:

- Membuat data untuk regresi linear.
- Membagi data menjadi set pelatihan dan pengujian.
- Membuat model regresi linear menggunakan PyTorch dan melatihnya.

Pemodelan Data Non-Linear:

- Data baru dibuat menggunakan make_circles.
- Model NN dengan fungsi aktivasi ReLU ditambahkan untuk menangani masalah klasifikasi data yang non-linear.

Menggunakan Sigmoid dan ReLU:

- Model dengan fungsi aktivasi ReLU ditambahkan.
- Proses pelatihan dan evaluasi dilakukan pada data non-linear.

Penggunaan Activation Functions:

- Model NN baru (CircleModelV2) dengan fungsi aktivasi ReLU ditambahkan untuk menangani data non-linear.

Klasifikasi Multi-Kelas:

- Data sintetis untuk klasifikasi multi-kelas dibuat.
- Model NN untuk klasifikasi multi-kelas (BlobModel) dengan softmax dan cross-entropy loss dibuat.

Pelatihan dan Evaluasi Model Multi-Kelas:

- Model multi-kelas dilatih dan dievaluasi.
- Decision boundary untuk set pelatihan dan pengujian diplot.

Menggunakan TorchMetrics:

- Menggunakan TorchMetrics untuk menghitung akurasi pada tugas klasifikasi multi-kelas.

Visualisasi Decision Boundary:

- Decision boundary model diplot untuk melihat kemampuan model dalam memahami batas keputusan pada data.

Keseluruhan, kode tersebut menggambarkan penerapan PyTorch pada beberapa jenis tugas machine learning, baik itu klasifikasi biner, klasifikasi multi-kelas, atau regresi. Juga, diperkenalkan beberapa konsep seperti fungsi aktivasi, penanganan data non-linear, dan penggunaan TorchMetrics untuk metrik evaluasi.

Chapter 03 PyTorch

Kode tersebut adalah implementasi dari sebuah model jaringan saraf menggunakan PyTorch untuk tugas klasifikasi menggunakan dataset FashionMNIST. Berikut adalah penjelasan singkat tentang beberapa bagian kodenya:

Pengenalan:

- Kode dimulai dengan mengimpor pustaka yang diperlukan seperti PyTorch, torchvision, dan matplotlib.
- Menyiapkan versi PyTorch yang sesuai.

Dataset dan Visualisasi:

- Mengunduh dataset FashionMNIST dan membaginya menjadi data pelatihan dan pengujian.
- Menampilkan beberapa sampel gambar dari dataset.

Pembuatan Dataloader:

- Menggunakan DataLoader untuk membuat iterator untuk data pelatihan dan pengujian dalam bentuk batch.
- Menampilkan informasi tentang panjang dataloader dan ukuran batch.

Pembuatan Model:

- Membuat tiga model jaringan saraf berbeda (FashionMNISTModelV0, FashionMNISTModelV1, FashionMNISTModelV2).
- Model menggunakan lapisan linear dan konvolusional untuk memproses data.

Pelatihan dan Evaluasi Model:

- Melatih model menggunakan data pelatihan dan mengukur kinerja pada data pengujian.
- Menggunakan fungsi kerugian CrossEntropyLoss dan optimizer Stochastic Gradient Descent (SGD).
- Menyimpan model terbaik berdasarkan kinerja.

Evaluasi Model dan Confusion Matrix:

- Mengevaluasi model pada data pengujian.
- Membuat confusion matrix untuk mengevaluasi performa model lebih lanjut.

Visualisasi Hasil:

- Membandingkan hasil tiga model berbeda dalam bentuk DataFrame.
- Menampilkan grafik batang horizontal untuk akurasi model.

Prediksi dan Confusion Matrix:

- Melakukan prediksi pada beberapa sampel data pengujian.
- Membuat confusion matrix untuk mengevaluasi performa model.

Simpan dan Muat Model:

- Menyimpan model terbaik ke dalam file.
- Memuat kembali model dari file yang disimpan.
- Mengevaluasi model yang dimuat untuk memastikan konsistensi hasil dengan model sebelumnya.

Plot Confusion Matrix:

- Menampilkan confusion matrix untuk evaluasi visual performa model.

Perbandingan Hasil:

- Membandingkan hasil antara model awal dan model yang dimuat untuk memastikan konsistensi.

Kode ini mencakup seluruh siklus pembuatan model, pelatihan, evaluasi, dan penyimpanan yang umumnya terlibat dalam pengembangan model menggunakan PyTorch.