Zero to Mastery Learn PyTorch for Deep Learning Course Summary

01. PyTorch Workflow Fundamentals

Nama : Al Ghifary Akmal Nasheeri

NIM : 1103201242

Kelas : TK-44-06

Modul ini membuka pintu dalam pemahaman dasar tentang machine learning dan deep learning. Tujuan utamanya adalah menggunakan data masa lalu untuk membangun algoritma, seperti jaringan saraf (neural network), yang dapat menemukan pola dalam data tersebut. Setelah pola ditemukan, algoritma tersebut dapat digunakan untuk memprediksi peristiwa di masa depan.

Dalam modul ini, kita akan memulai dengan konsep yang sederhana, yakni garis lurus. Kemudian, kita akan melihat bagaimana membuat model PyTorch yang dapat mempelajari pola dari garis lurus tersebut dan mencocokkannya.

Berikut adalah langkah-langkah utama yang akan dibahas dalam modul ini:

1. Persiapan Data: Data dapat berasal dari berbagai sumber, namun pada awalnya, kita akan membuat data sederhana berupa garis lurus.
2. Pembuatan Model: Di sini, kita akan membuat model untuk mempelajari pola dalam data. Kita juga akan memilih fungsi kerugian (loss function), pengoptimal (optimizer), dan membangun loop pelatihan.
3. Penyesuaian Model dengan Data (pelatihan): Setelah memiliki data dan model, kita akan membiarkan model mencari pola dalam data pelatihan.
4. Pembuatan Prediksi dan Evaluasi Model (inferensi): Model kita telah menemukan pola dalam data, sekarang kita akan membandingkan temuannya dengan data pengujian aktual.
5. Penyimpanan dan Pemuatan Model: Mungkin Anda ingin menggunakan model Anda di tempat lain atau kembali menggunakannya nanti. Di sini, kita akan membahas cara menyimpan dan memuat model.
6. Penggabungan Semua Komponen: Mari menggabungkan semua langkah di atas dan melihatnya sebagai suatu kesatuan.

Persiapan dan Pemuatan Data

Dalam machine learning, data dapat berupa apa saja, mulai dari tabel angka, gambar, video, audio, hingga struktur protein dan teks. Proses machine learning melibatkan dua bagian utama, yakni mengubah data menjadi kumpulan angka representatif dan membangun atau memilih model untuk mempelajari representasi tersebut sebaik mungkin. Pada modul ini, kita mulai dengan menciptakan data garis lurus sebagai contoh. Langkah-langkah yang akan dijalankan meliputi:

1. Pembuatan parameter yang diketahui untuk data garis lurus.
2. Pembuatan data menggunakan linear regression dengan menggunakan PyTorch.
3. Pembagian data menjadi set pelatihan dan pengujian dengan proporsi 80:20.
4. Visualisasi data untuk memahami pola dan hubungan antara fitur (X) dan label (y).

Data yang sudah dipersiapkan dan divisualisasikan akan digunakan sebagai dasar untuk membangun model pada langkah selanjutnya.

Pembangunan Model

Dalam langkah ini, kita membangun model regresi linear menggunakan PyTorch. Model ini dibuat sebagai kelas Python yang mengimplementasikan fungsi regresi linear sederhana. Pada intinya, kita menggunakan modul **torch.nn** yang menyediakan blok bangunan untuk grafik komputasi, seperti **nn.Module**, **nn.Parameter**, dan **torch.optim**. Model regresi linear yang dibuat memiliki dua parameter, yaitu weight dan bias, yang akan disesuaikan selama proses pembelajaran.

Kita juga melakukan pengujian sederhana pada model yang telah dibuat dengan membuat beberapa prediksi pada data uji. Hasil prediksi awalnya buruk karena parameter model diinisialisasi secara acak. Pada langkah selanjutnya, kita akan melihat bagaimana model ini dapat disesuaikan untuk memprediksi data dengan lebih baik.

Rangkuman Bagian 3: Pelatihan Model

Pada langkah ini, kita melatih model regresi linear menggunakan PyTorch. Proses pelatihan ini melibatkan beberapa langkah kunci, yaitu:

1. **Inisialisasi Model**: Model diinisialisasi dengan parameter acak (weights dan bias).
2. **Pembuatan Fungsi Kerugian dan Optimizer**: Fungsi kerugian (loss function) digunakan untuk mengukur seberapa salah prediksi model dibandingkan dengan nilai sebenarnya. Optimizer, dalam hal ini Stochastic Gradient Descent (SGD), digunakan untuk mengupdate parameter model agar sesuai dengan pola data yang lebih baik.

3. **Loop Pelatihan (Training Loop)**: Model melakukan iterasi melalui data latih, melakukan prediksi, menghitung loss, melakukan backpropagation, dan memperbarui parameter untuk meminimalkan loss.
4. **Loop Pengujian (Testing Loop)**: Model diuji pada data uji untuk mengevaluasi seberapa baik model dapat memgeneralisasi ke data yang belum pernah dilihat selama pelatihan.
5. **Visualisasi Loss Curves**: Melihat kurva loss dari waktu ke waktu untuk menilai seberapa baik model telah belajar dari data.

Hasil dari pelatihan ini adalah model yang dapat memprediksi dengan lebih baik, dan kita melihat penurunan nilai loss pada setiap iterasi. Pada akhir pelatihan, kita melihat bahwa parameter model (weights dan bias) mendekati nilai yang sebenarnya, menunjukkan bahwa model telah mempelajari pola dalam data.

Penting untuk memahami konsep pelatihan dan evaluasi model, serta pengaruh hyperparameter seperti jumlah epoch dan learning rate dalam mencapai hasil yang baik. Langkah-langkah ini membantu kita memahami inti dari workflow pada PyTorch untuk mengembangkan model machine learning sederhana.

Melakukan Prediksi dengan Model PyTorch yang Telah Dilatih (Inference)

Setelah melatih model, langkah selanjutnya adalah membuat prediksi (inference) dengan model tersebut. Berikut adalah langkah-langkah utama untuk melakukan prediksi dengan model PyTorch:

1. **Mengatur Model ke Mode Evaluasi**: Menggunakan fungsi **model.eval()** untuk memastikan bahwa model berada dalam mode evaluasi, di mana beberapa fungsi yang tidak diperlukan selama pelatihan dinonaktifkan.
2. **Menggunakan Konteks Inference Mode**: Menggunakan **with torch.inference_mode():** sebagai konteks untuk membuat prediksi. Ini memastikan bahwa kalkulasi dilakukan dengan cepat dan efisien.
3. **Memastikan Kalkulasi pada Perangkat yang Sama**: Pastikan bahwa model dan data berada pada perangkat yang sama (CPU atau GPU) untuk mencegah kesalahan lintas perangkat.

Setelah langkah-langkah tersebut dilakukan, kita dapat membuat prediksi dengan menggunakan model yang telah dilatih pada data uji. Dalam contoh di atas, hasil prediksi (**y_preds**) telah dihasilkan dan visualisasi prediksi dilakukan menggunakan fungsi **plot_predictions()**.

Selanjutnya, kita akan menjelajahi cara menyimpan dan memuat model PyTorch.

Menyimpan dan Memuat Model PyTorch

Setelah melatih model PyTorch, langkah selanjutnya adalah menyimpannya untuk digunakan di lain waktu atau di tempat lain. PyTorch menyediakan beberapa metode utama untuk menyimpan dan memuat model, dan di bawah ini adalah tiga metode tersebut:

1. **torch.save**: Menyimpan objek ter-serialisasi ke disk menggunakan utilitas pickle Python. Model, tensor, dan berbagai objek Python lainnya seperti kamus dapat disimpan menggunakan torch.save.
2. **torch.load**: Menggunakan fitur deserialisasi pickle untuk memuat file objek Python ter-serialisasi (seperti model, tensor, atau kamus) ke dalam memori. Anda juga dapat menentukan perangkat mana objek harus dimuat ke (CPU, GPU, dll).
3. **torch.nn.Module.load_state_dict**: Memuat kamus parameter model (model.state_dict()) menggunakan objek state_dict() yang telah disimpan.

Disarankan untuk menyimpan dan memuat model hanya menggunakan state_dict(), karena metode ini lebih fleksibel dan tidak tergantung pada struktur kelas atau direktori yang spesifik.

1. **Menyimpan Model:**
   - Membuat direktori untuk menyimpan model (dalam contoh ini, "models").
   - Membuat jalur file untuk menyimpan model.
   - Menggunakan **torch.save(obj, f)** di mana obj adalah state_dict() model target dan f adalah nama file untuk menyimpan model.
2. **Memuat Model:**
   - Membuat instance baru dari model yang akan dimuat.
   - Memuat state_dict() model yang telah disimpan menggunakan **torch.load(f)** dan memasukkannya ke instance baru model menggunakan **loaded_model.load_state_dict()**.
3. **Melakukan Prediksi dengan Model yang Telah Dimuat:**
   - Mengatur model yang dimuat ke mode evaluasi.
   - Menggunakan konteks inference_mode untuk membuat prediksi.

Setelah memuat model, prediksi yang dihasilkan oleh model tersebut dapat dibandingkan dengan prediksi sebelumnya untuk memastikan bahwa model telah menyimpan dan memuat dengan benar.

Menggabungkan Semua Langkah

1. **Data**
    1. Membuat data dengan menentukan weight dan bias.
    2. Membuat range nilai antara 0 dan 1 untuk X.
    3. Menggunakan X, weight, dan bias untuk membuat label (y) menggunakan rumus regresi linear (y = weight * X + bias).
    4. Membagi data menjadi set pelatihan dan pengujian (80/20 split).
2. **Membangun Model Linear PyTorch**
    1. Membuat model linear menggunakan subclass nn.Module dan nn.Linear().
    2. Model memiliki satu lapisan linear dengan satu input dan satu output.
3. **Pelatihan Model**
    1. Menggunakan loss function nn.L1Loss() dan optimizer torch.optim.SGD().
    2. Pelatihan model selama serangkaian epoch dengan meletakkan data di perangkat yang tersedia (GPU jika tersedia, jika tidak, default ke CPU).
4. **Melakukan Prediksi**
    1. Mengubah model ke mode evaluasi.
    2. Melakukan prediksi pada data pengujian.
5. **Menyimpan dan Memuat Model**
    1. Menyimpan model ke file menggunakan torch.save() dengan menyertakan state_dict() model.
    2. Memuat model kembali menggunakan torch.load() dan load_state_dict().
    3. Evaluasi model yang dimuat untuk memastikan prediksinya sesuai dengan model sebelumnya.
6. **Keseluruhan**

Menggabungkan langkah-langkah di atas dalam satu rangkaian kode dengan membuatnya agnostik perangkat, sehingga dapat menggunakan GPU jika tersedia atau beralih ke CPU jika tidak.