Deep Learning dengan PyTorch

Harits Maulana Muzakki
1103204166
TK-44-04

Dasar Tensor

PyTorch adalah framework deep learning populer yang menyediakan alat yang kuat untuk membangun dan melatih jaringan saraf. Pada intinya, PyTorch menggunakan konsep tensor, yang merupakan larik multidimensi mirip dengan larik NumPy. Tensor dalam PyTorch digunakan untuk mewakili data, parameter, dan komputasi.

Tensor PyTorch memiliki beberapa properti penting, termasuk bentuknya, tipe data, dan perangkat yang digunakan (CPU atau GPU). Tensor dapat dibuat dari daftar Python, larik NumPy, atau langsung menggunakan fungsi PyTorch. Operasi tensor umum termasuk operasi aritmetika elemen, perkalian matriks, perubahan bentuk, dan pengindeksan.

Autograd

Autograd adalah fitur kunci dalam PyTorch yang memungkinkan diferensiasi otomatis. Autograd secara otomatis melacak operasi yang dilakukan pada tensor dan menghitung gradien terhadap tensor input. Ini memungkinkan perhitungan gradien yang efisien selama proses backpropagation, yang sangat penting untuk melatih jaringan saraf.

Secara default, setiap tensor dalam PyTorch memiliki fungsi gradien terkait yang melacak operasi yang melibatkan tensor tersebut. Gradien dapat dihitung dengan memanggil metode `backward()` pada tensor. Gradien kemudian diakumulasikan dalam atribut `.grad` dari tensor yang bersangkutan.

Backpropagation

Backpropagation adalah algoritma dasar yang digunakan untuk melatih jaringan saraf. Ini melibatkan perhitungan gradien dari fungsi kerugian terhadap parameter jaringan menggunakan aturan rantai kalkulus. Fungsi autograd dalam PyTorch membuat backpropagation menjadi otomatis dan efisien.

Selama backpropagation, gradien mengalir mundur melalui grafik komputasi, dan gradien parameter dihitung menggunakan aturan rantai. Optimizer kemudian menggunakan gradien ini untuk memperbarui parameter dalam arah yang meminimalkan fungsi kerugian.

Gradient Descent

Gradient descent adalah algoritma optimisasi yang digunakan untuk secara iteratif memperbarui parameter jaringan saraf guna meminimalkan fungsi kerugian. PyTorch menyediakan berbagai algoritma optimisasi dalam modul `torch.optim`, termasuk stochastic gradient descent (SGD), Adam, dan RMSprop.

Dalam gradient descent, gradien yang dihitung selama backpropagation digunakan untuk memperbarui parameter. Optimizer menyesuaikan parameter dengan mengambil langkah proporsional terhadap gradien negatif, dengan skala yang ditentukan oleh laju pembelajaran. Proses ini diulang beberapa kali hingga konvergensi atau mencapai kriteria berhenti yang ditentukan sebelumnya.

Pipa Pelatihan

Pipa pelatihan dalam PyTorch biasanya melibatkan langkah-langkah berikut:

1. Persiapan Data
Memuat dan memproses dataset menggunakan utilitas pemuatan data PyTorch. Membagi dataset menjadi set pelatihan, validasi, dan pengujian.

2. Definisi Model
Mendefinisikan arsitektur model jaringan saraf menggunakan kelas `nn.Module` PyTorch. Ini termasuk mendefinisikan lapisan-lapisan, fungsi aktivasi, dan parameter yang diperlukan.

3. Fungsi Kerugian
Memilih fungsi kerugian yang sesuai berdasarkan tugas yang dihadapi. Fungsi kerugian umum termasuk mean squared error (MSE) untuk tugas regresi dan cross-entropy loss untuk tugas klasifikasi.

4. Pemilihan Optimizer
Memilih optimizer dari modul `torch.optim` PyTorch, seperti SGD atau Adam. Tentukan laju pembelajaran dan hiperparameter lainnya.

5. Loop Pelatihan
Mengulangi set pelatihan dalam mini-batch. Meneruskan maju input melalui jaringan, menghitung kerugian, melakukan backpropagation untuk menghitung gradien, dan memperbarui parameter menggunakan optimizer.

6. Validasi dan Pengujian
Secara berkala mengevaluasi model pada set validasi untuk memantau kinerjanya. Setelah pelatihan selesai, mengevaluasi model final pada set pengujian untuk menilai kemampuannya dalam generalisasi.

Regresi Linear

Regresi linear adalah teknik sederhana namun kuat yang digunakan untuk memprediksi variabel target berkelanjutan berdasarkan satu atau lebih fitur input. Dalam PyTorch, regresi linear dapat diimplementasikan menggunakan jaringan saraf satu lapisan dengan fungsi aktivasi linear.

Model mempelajari bobot dan bias yang paling cocok untuk memetakan data input ke variabel target. Fungsi kerugian mean squared error (MSE) umumnya digunakan untuk tugas regresi linear.

Regresi Logistik

Regresi logistik adalah algoritma klasifikasi biner yang digunakan untuk memprediksi probabilitas suatu instansi termasuk ke dalam kelas tertentu. Dalam PyTorch, regresi logistik dapat diimplementasikan menggunakan jaringan saraf satu lapisan dengan fungsi aktivasi sigmoid.

Model mempelajari bobot dan bias yang memetakan fitur input ke probabilitas kelas. Fungsi kerugian cross-entropy umumnya digunakan untuk tugas regresi logistik.

Dataset dan Dataloader

Dalam PyTorch, kelas `torch.utils.data.Dataset` menyediakan antarmuka untuk mewakili dataset. Dataset khusus dapat dibuat dengan mewarisi kelas ini dan mengimplementasikan metode `__len__` dan `__getitem__` untuk memberikan akses ke sampel-sampel.

Kelas `torch.utils.data.DataLoader` digunakan untuk memuat data dari dataset secara paralel, membuat mini-batch, dan mengacak data saat pelatihan. Ini menyediakan objek yang dapat diulang melalui dataset yang dapat digunakan dalam loop pelatihan.

Transformasi Dataset

Transformasi dataset adalah operasi yang diterapkan pada data input untuk memproses dan meningkatkannya sebelum disajikan ke j

aringan saraf. Modul `torchvision.transforms` PyTorch menyediakan berbagai transformasi, seperti normalisasi, pengubah ukuran, pemangkasan, dan teknik augmentasi data seperti flipping atau rotasi acak.

Transformasi dapat diterapkan pada dataset selama proses pemuatan data untuk memastikan preprocessing data yang konsisten dan efisien.

Softmax dan Crossentropy

Softmax dan cross-entropy umum digunakan dalam tugas klasifikasi multi-kelas. Softmax adalah fungsi aktivasi yang mengubah output jaringan saraf menjadi distribusi probabilitas atas kelas-kelas. Cross-entropy adalah fungsi kerugian yang mengukur perbedaan antara probabilitas yang diprediksi dan label sebenarnya.

Dalam PyTorch, fungsi softmax tersedia dalam modul `torch.nn.functional`, dan loss cross-entropy dapat dihitung menggunakan kelas `torch.nn.CrossEntropyLoss`.

Fungsi Aktivasi

Fungsi aktivasi memperkenalkan non-linearitas ke dalam jaringan saraf, memungkinkannya untuk mempelajari hubungan kompleks antara input dan output. PyTorch menyediakan berbagai fungsi aktivasi dalam modul `torch.nn`, termasuk ReLU, sigmoid, tanh, dan lainnya.

Fungsi aktivasi biasanya diterapkan setelah transformasi linear pada lapisan-lapisan jaringan untuk memperkenalkan non-linearitas dan meningkatkan kemampuan model.

Jaringan Feed Forward

Jaringan saraf feed-forward, juga dikenal sebagai multi-layer perceptron (MLP), adalah jenis arsitektur jaringan saraf di mana informasi mengalir ke satu arah, dari lapisan input melalui satu atau lebih lapisan tersembunyi ke lapisan output. Setiap lapisan terdiri dari beberapa neuron yang melakukan transformasi linear diikuti oleh fungsi aktivasi non-linear.

Kelas `nn.Module` PyTorch dapat digunakan untuk mendefinisikan dan melatih jaringan saraf feed-forward. Arsitektur jaringan ditentukan dengan menumpuk beberapa lapisan dengan fungsi aktivasi yang sesuai.

Jaringan Saraf Konvolusi

Jaringan saraf konvolusi (CNN) banyak digunakan untuk klasifikasi gambar, deteksi objek, dan tugas-tugas visi komputer lainnya. CNN dirancang untuk secara otomatis mempelajari dan mengekstrak fitur-fitur dari gambar menggunakan lapisan konvolusi, lapisan penggabungan, dan lapisan terhubung penuh.

PyTorch menyediakan berbagai modul dalam modul `torch.nn` untuk membangun CNN, termasuk lapisan konvolusi (`torch.nn.Conv2d`), lapisan penggabungan (`torch.nn.MaxPool2d`), dan lainnya. Arsitektur CNN umumnya terdiri dari lapisan-lapisan konvolusi diikuti oleh lapisan terhubung penuh.

Transfer Learning

Transfer learning adalah teknik yang menggunakan model jaringan saraf yang telah dilatih sebelumnya untuk menyelesaikan tugas atau dataset baru. Ini melibatkan penggunaan model yang telah dilatih sebagai extractor fitur dan mengganti atau men

yesuaikan beberapa lapisan terakhir untuk tugas yang spesifik.

Modul `torchvision.models` PyTorch menyediakan model-model yang telah dilatih sebelumnya seperti ResNet, VGG, dan AlexNet. Model-model ini dapat dimuat, dimodifikasi, dan digunakan untuk transfer learning dengan membekukan lapisan awal dan hanya melatih lapisan yang ditambahkan atau dimodifikasi.

Tensorboard

TensorBoard adalah alat visualisasi yang disediakan oleh TensorFlow untuk melacak dan memvisualisasikan proses pelatihan model deep learning. Meskipun PyTorch tidak terintegrasi secara native dengan TensorBoard, ada pustaka pihak ketiga, seperti `tensorboardX`, yang menyediakan kompatibilitas dengan PyTorch.

Dengan menggunakan `tensorboardX` dapat mencatat berbagai informasi selama pelatihan, seperti nilai skalar, gambar, dan histogram, serta memvisualisasikannya di TensorBoard.

Menyimpan dan Memuat Model

PyTorch memungkinkan untuk menyimpan dan memuat model yang telah dilatih, yang penting untuk menggunakan ulang model, berbagi dengan orang lain, atau menerapkannya di produksi. Ini dapat menyimpan seluruh model atau hanya `state_dict` (parameter) model.

Untuk menyimpan model dapat menggunakan fungsi `torch.save()`, dengan menentukan model dan jalur file yang diinginkan. Untuk memuat model yang telah disimpan dapat menggunakan fungsi `torch.load()`, dengan menentukan jalur file, dan kemudian memuat `state_dict` model untuk inferensi atau pelatihan lebih lanjut.