Nama : Arfara Yema Samgusdian
NIM : 1103202004
Kelas : TK-44-04

Zero to Mastery Learn PyTorch for Deep Learning

02. PyTorch Neural Network Classification

Arsitektur Neural Network Klasifikasi
- Menunjukkan arsitektur umum dari neural network klasifikasi.
- Hyperparameter yang digunakan untuk klasifikasi biner dan klasifikasi banyak kelas.

Persiapan Data Klasifikasi
- Membuat dataset klasifikasi sederhana menggunakan make_circles dari Scikit-Learn.
- Mengubah data menjadi format tensor dan membaginya menjadi set pelatihan dan pengujian.

Membangun Model Klasifikasi PyTorch
- Membuat model menggunakan PyTorch, dengan dua layer linear.
- Menggunakan fungsi aktivasi ReLU di hidden layer dan sigmoid di output layer untuk klasifikasi biner.
- Atau, menggunakan nn.Sequential untuk model yang serupa.

Setup Fungsi Loss dan Optimizer
- Memilih fungsi loss yang sesuai untuk klasifikasi biner (BCEWithLogitsLoss).
- Menggunakan optimasi Stochastic Gradient Descent (SGD) dengan learning rate 0.1.

Pelatihan Model dan Evaluasi
- Melakukan pelatihan model pada data pelatihan.
- Mengukur performa model menggunakan loss dan metrik evaluasi (accuracy).
- Mengevaluasi model pada data pengujian.

Peningkatan Model dan Non-Linearitas
- Mengeksplorasi cara untuk meningkatkan model, seperti peningkatan jumlah hidden units atau layer.
- Menambahkan fungsi aktivasi non-linear (ReLU) untuk memodelkan hubungan non-linear.

Klasifikasi Multi-Kelas
- Menggabungkan konsep-konsep yang telah diajarkan untuk menangani masalah klasifikasi banyak kelas.

Forward Pass
Selama fase forward pass, model menghasilkan keluaran mentah yang disebut logits. Logits ini tidak mudah diinterpretasikan, sehingga kita menggunakan fungsi aktivasi sigmoid untuk mengubahnya menjadi probabilitas prediksi.

Pelatihan dan Pengujian
Kita telah membangun loop pelatihan untuk 100 epoch. Setiap epoch melibatkan langkah-langkah berikut:

Forward pass untuk menghasilkan prediksi
Perhitungan kerugian (loss) dan akurasi
Pembaharuan parameter model menggunakan optimasi gradien
Selain itu, kita juga melakukan pengujian pada data pengujian dan mencetak hasilnya setiap 10 epoch.

Evaluasi Model
Meskipun kita telah melatih model, hasilnya masih kurang memuaskan dengan akurasi yang tidak melampaui 50%. Hal ini menunjukkan bahwa model kita hanya memprediksi secara acak dan belum belajar pola prediktif dari data.

Visualisasi Hasil
Untuk lebih memahami kinerja model, kita melakukan visualisasi dengan membuat plot keputusan (decision boundary) yang dihasilkan oleh model pada data pelatihan dan pengujian. Hasilnya menunjukkan bahwa model kita sedang mencoba memisahkan data dengan garis lurus, padahal data kita berbentuk lingkaran. Oleh karena itu, model mengalami underfitting.

Menambahkan lebih banyak lapisan (layers)
Setiap lapisan potensial meningkatkan kemampuan pembelajaran model.
Menambahkan lapisan dapat membantu model memahami pola yang lebih kompleks.

Menambahkan lebih banyak unit tersembunyi (hidden units):
Lebih banyak unit tersembunyi dalam setiap lapisan dapat meningkatkan kemampuan pembelajaran model.
Ini dikenal sebagai membuat jaringan saraf menjadi lebih "lebar."

Pelatihan lebih lama (lebih banyak epoch):
Memberi model lebih banyak kesempatan untuk melihat data dan belajar.

Mengganti fungsi aktivasi:
Menggunakan fungsi aktivasi non-linear, seperti ReLU, untuk membantu model memahami pola yang lebih kompleks.

kita membuat kembali data untuk memulai dari awal dengan lingkaran merah dan biru. Kemudian, kita membaginya menjadi set pelatihan dan pengujian menggunakan 80% data untuk pelatihan dan 20% untuk pengujian.

Kemudian, kita membangun model dengan menambahkan fungsi aktivasi non-linear ReLU (Rectified Linear Unit) di antara lapisan tersembunyi. Model ini disebut CircleModelV2. ReLU memungkinkan model untuk memperkenalkan kompleksitas non-linear ke dalam pembelajarannya.

Setelah membangun model, kita menentukan fungsi kerugian (loss function) dan optimizer. Kita menggunakan fungsi loss BCEWithLogitsLoss untuk tugas klasifikasi biner.

Setelahnya, kita melatih model dengan langkah-langkah pelatihan yang telah ditentukan. Hasilnya menunjukkan peningkatan yang signifikan dalam performa model, yang dapat memodelkan pola non-linear pada data lingkaran.

Terakhir, kita mengevaluasi model dengan melihat prediksi yang dihasilkannya terhadap data uji dan membandingkannya dengan label sebenarnya. Visualisasi menunjukkan bahwa model dengan fungsi aktivasi non-linear dapat menghasilkan batas keputusan yang lebih baik daripada model tanpa fungsi aktivasi non-linear.

Pada akhirnya, kita membahas beberapa fungsi aktivasi non-linear lainnya seperti ReLU dan sigmoid, serta bagaimana kita dapat menggunakan kombinasi dari fungsi-fungsi tersebut untuk memberikan model kita kemampuan untuk menemukan pola-pola yang kompleks dalam data, baik linear maupun non-linear.

Pembuatan Data Multi-Class
Data multi-class dibuat menggunakan metode make_blobs dari Scikit-Learn.
Data tersebut diubah menjadi tensor dan dibagi menjadi set pelatihan dan uji.

Pembuatan Model Multi-Class Classification
Model dibangun menggunakan PyTorch dengan subclass nn.Module.
Model memiliki tiga lapisan linear dan dapat menangani data multi-class.

Pembuatan Fungsi Kerugian dan Optimizer
Fungsi kerugian nn.CrossEntropyLoss() digunakan untuk masalah klasifikasi multi-kelas.
Optimizer menggunakan SGD dengan laju pembelajaran 0.1.

Perolehan Probabilitas Prediksi untuk Model Multi-Class
Prediksi dilakukan dengan model, dan fungsi softmax digunakan untuk mengonversi logits menjadi probabilitas prediksi.
Argmax dari probabilitas prediksi memberikan label prediksi.

Pelatihan dan Evaluasi Model

Dilakukan pelatihan model untuk 100 epoch dengan evaluasi setiap 10 epoch.
Performa model diukur menggunakan akurasi.

Evaluasi Tambahan dengan Metric Klasifikasi
Diperkenalkan metric klasifikasi tambahan seperti presisi, recall, F1-score, confusion matrix, dan classification report.
TorchMetrics digunakan untuk menghitung akurasi.

Visualisasi Prediksi Model
Prediksi model divisualisasikan menggunakan plot_decision_boundary().