Maulana Hafiz/1103223105

1. Jika menggunakan model MLP dengan 3 hidden layer (256-128-64) menghasilkan underfitting pada dataset ini, modifikasi apa yang akan dilakukan pada arsitektur? Jelaskan alasan setiap perubahan dengan mempertimbangkan bias-variance tradeoff!
2. Selain MSE, loss function apa yang mungkin cocok untuk dataset ini? Bandingkan kelebihan dan kekurangannya, serta situasi spesifik di mana alternatif tersebut lebih unggul daripada MSE!
3. Jika salah satu fitur memiliki range nilai 0-1, sedangkan fitur lain 100-1000, bagaimana ini memengaruhi pelatihan MLP? Jelaskan mekanisme matematis (e.g., gradien, weight update) yang terdampak!
4. Tanpa mengetahui nama fitur, bagaimana Anda mengukur kontribusi relatif setiap fitur terhadap prediksi model? Jelaskan metode teknikal (e.g., permutation importance, weight analysis) dan keterbatasannya!
5. Bagaimana Anda mendesain eksperimen untuk memilih learning rate dan batch size secara optimal? Sertakan analisis tradeoff antara komputasi dan stabilitas pelatihan!

JAWABAN

1. Modifikasi Arsitektur MLP untuk Mengatasi Underfitting
   Perubahan yang Direkomendasikan:
   - Tambah Jumlah Neuron per Layer (misal: 512-256-128) untuk meningkatkan kapasitas model.
   - Tambah Layer (misal: 4-5 hidden layer) jika data sangat kompleks.
   - Gunakan Aktivasi Non-Linear (ReLU, Swish) di setiap layer untuk menangkap pola kompleks.
   - Kurangi Regularisasi (kurangi Dropout/L2 weight decay) jika ada.

   Alasan (Bias-Variance Tradeoff):

   - Underfitting terjadi karena model terlalu sederhana (high bias). Dengan meningkatkan kapasitas (neuron/layer), model dapat mempelajari representasi data yang lebih baik.
   - Risiko Overfitting: Jika perubahan drastis (misal: neuron/layer berlebihan), model mungkin high variance. Solusnya gunakan teknik validasi (early stopping) atau regularisasi *setelah* underfitting teratasi.

2. Selain MSE, beberapa alternatif loss function yang cocok untuk dataset ini antara lain MAE, Huber Loss, dan Log-Cosh Loss. MAE lebih robust terhadap outlier karena tidak membesar-besarkan error besar seperti MSE, tetapi gradiennya konstan sehingga konvergensi mungkin lebih lambat. Huber Loss menggabungkan kelebihan MSE dan MAE dengan bersifat smooth untuk error kecil dan robust untuk error besar, meskipun memerlukan tuning hyperparameter δ. Log-Cosh Loss mirip dengan MSE tetapi kurang sensitif terhadap outlier, meskipun komputasinya lebih mahal. Pemilihan loss function tergantung pada karakteristik data: MAE atau Huber Loss lebih cocok untuk data

dengan banyak outlier, sedangkan Log-Cosh Loss ideal untuk situasi di mana error besar jarang terjadi tetapi smoothness tetap penting.

3. Perbedaan skala antarfitur, seperti satu fitur dengan range 0-1 dan lainnya 100-1000, dapat mengganggu pelatihan MLP. Fitur dengan skala lebih besar akan mendominasi gradien, menyebabkan pembaruan berat tidak seimbang dan konvergensi yang tidak stabil. Secara matematis, gradien untuk fitur skala besar menjadi dominan dalam perhitungan $\partial \text{Loss}/\partial W$, sehingga weight update untuk fitur tersebut lebih signifikan. Hal ini mengakibatkan optimasi yang tidak efisien, seringkali dengan pola "zig-zag" pada nilai loss. Solusinya adalah normalisasi fitur menggunakan StandardScaler atau MinMaxScaler agar semua fitur berada dalam skala serupa, sehingga gradien dan pembaruan berat menjadi lebih seimbang.
4. Untuk mengukur kontribusi relatif setiap fitur tanpa mengetahui namanya, beberapa metode teknikal dapat digunakan. Permutation Importance adalah pendekatan sederhana dengan mengacak nilai satu fitur dan mengamati penurunan performa model, meskipun tidak menunjukkan arah pengaruh fitur. Weight Analysis pada layer pertama MLP dengan menghitung norma berat juga dapat mengindikasikan kontribusi fitur, tetapi hanya efektif untuk model linear atau MLP sederhana. SHAP Values memberikan interpretasi lebih mendalam dengan menghitung kontribusi per sampel, tetapi komputasinya lebih intensif. Pemilihan metode tergantung pada kebutuhan: Permutation Importance untuk analisis cepat, sedangkan SHAP untuk interpretasi detail.
5. Mendesain eksperimen untuk memilih learning rate dan batch size secara optimal memerlukan pendekatan bertahap. Untuk learning rate, gunakan grid search pada nilai eksponensial seperti 1e-5 hingga 1e-3 atau gunakan Cyclic Learning Rate untuk mengidentifikasi range optimal. Untuk batch size, coba kelipatan 2 seperti 32, 64, atau 128, di mana batch size kecil (32-64) cenderung memberikan konvergensi lebih baik tetapi lebih noisy, sementara batch size besar (128+) lebih stabil tetapi membutuhkan lebih banyak memori. Tradeoff antara komputasi dan stabilitas perlu dipertimbangkan: batch size kecil memerlukan lebih banyak iterasi tetapi lebih akurat, sedangkan batch size besar mempercepat pelatihan per epoch tetapi mungkin memerlukan penyesuaian learning rate. Tools seperti TensorBoard atau learning rate finder dapat membantu memvisualisasikan pengaruh hyperparameter terhadap pelatihan.