Nama: Alfikri
NIM: 1103223015
Kelas: TK-46-06

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!

   Jawaban:
   Modifikasi Arsitektur MLP untuk Mengatasi Underfitting:
   Untuk mengatasi underfitting pada model MLP dengan tiga hidden layer (256-128-64), ada beberapa langkah yang bisa diambil berdasarkan prinsip bias-variance tradeoff. Underfitting terjadi ketika model terlalu sederhana untuk menangkap kompleksitas data, yang mengarah pada bias tinggi dan variance rendah.

   - Meningkatkan Jumlah Neuron: Salah satu cara untuk mengatasi underfitting adalah dengan meningkatkan jumlah neuron di setiap layer atau menambahkan lebih banyak hidden layers. Misalnya, mengganti 256-128-64 dengan konfigurasi yang lebih besar seperti 512-256-128. Hal ini akan memungkinkan model untuk menangkap lebih banyak fitur dan kompleksitas data, mengurangi bias.

   - Menambahkan Layer: Jika model masih underfitting, bisa dipertimbangkan untuk menambahkan lebih banyak hidden layers. Hal ini memberi lebih banyak kapasitas model untuk belajar representasi yang lebih kompleks dari data. Misalnya, menambah satu layer lagi setelah layer 64 dengan neuron yang cukup besar.

   - Mengurangi Regularisasi (Dropout): Jika dropout terlalu tinggi, itu bisa menyebabkan model kehilangan terlalu banyak informasi selama pelatihan, yang menyebabkan underfitting. Menurunkan tingkat dropout, misalnya dari 0.2 menjadi 0.1, dapat membantu model untuk belajar lebih banyak dari data.

   - Penggunaan Aktivasi Non-Linear Lain: Mengganti fungsi aktivasi dari ReLU ke Leaky ReLU atau ELU mungkin membantu dalam hal gradient yang lebih stabil, yang bisa mengatasi masalah underfitting pada jaringan yang lebih dalam.

   Semua perubahan ini bertujuan untuk menyeimbangkan bias dan variance. Meningkatkan kapasitas model (neuron dan layer) dapat menurunkan bias, tetapi berisiko meningkatkan variance. Oleh karena itu, perubahan ini harus dipantau dengan menggunakan teknik seperti cross-validation untuk memastikan model tidak menjadi overfit.

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!

   Jawaban:

   Selain Mean Squared Error (MSE), loss function yang cocok untuk regresi adalah Huber Loss. Huber Loss adalah gabungan antara MSE dan Mean Absolute Error (MAE), yang lebih robust terhadap outliers dibandingkan MSE.

   Kelebihan Huber Loss:

   - Huber Loss lebih tahan terhadap outliers dibandingkan MSE. MSE memberikan penalti besar terhadap outliers karena perhitungan berbasis kuadrat, sementara Huber Loss memberikan penalti lebih kecil untuk kesalahan yang besar, sehingga lebih stabil.
   - Huber Loss memberikan keuntungan lebih ketika data mengandung noise atau outliers yang signifikan.

   Kekurangan Huber Loss:

   - Meskipun lebih robust terhadap outliers, Huber Loss sedikit lebih kompleks untuk diimplementasikan dan lebih sulit untuk dipahami secara intuitif daripada MSE.

   Huber Loss lebih unggul daripada MSE jika dataset mengandung banyak outliers atau noise besar. Misalnya, dalam data keuangan atau pengukuran teknis yang cenderung memiliki lonjakan nilai yang ekstrem, Huber Loss akan lebih stabil.

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!

   Jawaban:

   Fitur dengan rentang nilai yang sangat berbeda (misalnya, antara 0-1 dan 100-1000) dapat menyebabkan masalah dalam pelatihan MLP karena perbedaan skala ini mempengaruhi update gradien dan bobot. Pada MLP, update bobot dilakukan menggunakan algoritma backpropagation yang memanfaatkan gradien dari fungsi loss.

   Jika fitur-fitur memiliki rentang nilai yang sangat berbeda, gradien yang dihitung untuk fitur dengan nilai lebih besar akan lebih dominan dalam pembaruan bobot dibandingkan dengan fitur yang memiliki rentang lebih kecil. Hal ini dapat

menyebabkan model kesulitan dalam menemukan solusi yang optimal, karena pembaruan bobot akan terdistorsi oleh fitur dengan skala yang lebih besar.

Untuk mengatasi masalah ini, kita perlu melakukan normalisasi atau standardisasi pada fitur agar semua fitur berada dalam skala yang serupa (misalnya, dengan menggunakan StandardScaler atau MinMaxScaler). Dengan demikian, pembaruan bobot akan lebih seimbang dan pelatihan akan lebih stabil.

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!

   Untuk mengukur kontribusi relatif setiap fitur terhadap prediksi model tanpa mengetahui nama fitur, ada beberapa metode teknikal yang dapat digunakan:

   - Permutation Importance: Metode ini mengukur kontribusi fitur dengan cara mengacak nilai fitur satu per satu, kemudian mengukur perubahan pada performa model (misalnya, dengan menghitung perubahan pada R2 atau MSE). Fitur yang berpengaruh besar pada model akan menyebabkan penurunan performa yang signifikan ketika nilainya diacak. Kelemahan dari metode ini adalah memerlukan banyak iterasi untuk mendapatkan hasil yang akurat, yang bisa mahal secara komputasi.
   - Weight Analysis (untuk Linear Models atau Neural Networks): Dalam beberapa model seperti regresi linier atau jaringan saraf dengan model yang lebih transparan, kita bisa melihat bobot yang dihasilkan oleh setiap fitur. Fitur dengan bobot yang lebih besar akan memiliki pengaruh lebih besar terhadap prediksi model. Namun, ini kurang efektif pada model yang lebih kompleks, seperti jaringan saraf yang dalam, karena bobotnya tidak selalu mudah diinterpretasikan.
   - Keterbatasan: Kedua metode ini memiliki keterbatasan, seperti kesulitan dalam menangani fitur yang memiliki hubungan non-linear dengan target, serta kebutuhan akan evaluasi yang memakan waktu dan komputasi.

5. Bagaimana Anda mendesain eksperimen untuk memilih learning rate dan batch size secara optimal? Sertakan analisis tradeoff antara komputasi dan stabilitas pelatihan!

   Jawaban:

   Untuk memilih learning rate dan batch size secara optimal, eksperimen dilakukan dengan mencoba berbagai kombinasi nilai untuk kedua hyperparameter tersebut. Rentang nilai learning rate dan batch size ditentukan, misalnya learning rate dari 1e-5 hingga 1e-1 dan batch size 16, 32, 64, atau 128. Setiap kombinasi diuji, dan performa model dievaluasi berdasarkan hasil pada data validasi.

   Tradeoff yang perlu dipertimbangkan adalah antara kecepatan pelatihan dan stabilitas. Learning rate yang terlalu tinggi bisa membuat pelatihan tidak stabil, sementara yang terlalu rendah akan memperlambat konvergensi. Begitu juga dengan batch size: yang terlalu besar mempercepat pelatihan tapi bisa mengurangi kemampuan generalisasi, sementara yang kecil memberikan update yang lebih sering

namun bisa memperlambat pelatihan dan mengurangi stabilitas. Eksperimen ini membantu memilih kombinasi terbaik berdasarkan kecepatan dan kualitas model.