Artificial Neural Network (ANN)

1. Cara kerja Algoritma

Artificial Neural Networks (ANN) bekerja dengan meniru cara kerja otak manusia untuk memproses informasi. Mereka terdiri dari banyak unit kecil yang disebut neuron, yang dihubungkan satu sama lain dengan lapisan/*layer*. Secara umum, ANN terdiri dari tiga jenis *layer*: *input layer*, *hidden layers*, dan *output layer*.

Input layer menerima data masukan dengan setiap neuron mewakili satu fitur atau variable dari data masukan. *Hidden layer* menerima informasi data dari *input layer*. Dalam satu model ANN bisa terdapat lebih dari satu *hidden layer*. *Output layer* menerima menerima informasi data dari *hidden layer* yang kemudian dikembalikan sebagai hasil prediksi.

Setiap neuron dihubungkan dengan variable bobot (*w*) yang berbeda-beda. Selain itu, setiap neuron kecuali yang ada di *input layer* juga memiliki variable bias (*b*) dan fungsi aktivas nya masing-masing. Terdapat beberapa jenis fungsi aktivasi, seperti sigmoid, linear, ReLu.

Secara garis besar, ANN dibagi menjadi dua tahap: *forward propagation* dan *backward propagation*. *Forward propagation* melibatkan input yang diteruskan dari lapisan ke lapisan sampai mencapai lapisan output. Setiap neuron menghitung outputnya dengan mengalikan data masukan dengan variable berat masing-masing lalu ditambah dengan biasnya. Nilai yang dihasilkan akan dimasukkan ke fungsi aktivasi. Hasil dari fungsi aktivasi inilah yang akan menjadi nilai output masing-masing neuron.

Output neuron pada *output layer* akan dihitung error-nya dengan loss function pilihan, Mean Squared Error untuk regresi dan Cross-Entropy Loss untuk klasifikasi pada lazimnya. Nilai error ini akan digunakan pada tahap berikutnya: *backward propagation*. Proses yang kedua ini bekerja dengan penghitungan gradient dari fungsi loss terhadap bobot-bobot nilai pada neuron untuk meminimalkan error. Proses ini diulang selama beberapa iterasi atau epoch sampai model mencapai tingkat akurasi yang diinginkan.

2. Perbandingan Hasil Evaluasi Model

Model yang saya buat memiliki score yang sedikit lebih buruk dan kecepatan algoritma 3 kali lebih cepat dari model dari Tensorflow. Perbedaan ini dipengaruhi oleh perbedaan learning rate dari kedua model. Learning rate dari kedua model berbeda, model saya menggunakan 0.02 sedangkan model dari Tensorflow menggunakan learning rate 0.0001. Saya melakukan ini karena setelah mencoba beberapa kali, model dari Tensorflow sangat sensititf terhadap learning rate dibuktikan dengan score yang bernilai

nan di learning rate lebih tinggi dan score yang tidak stabil (memiliki variansi yang cukup besar) ketika learning ratenya 0.001. Terdapat beberapa alasan hal ini bisa terjadi:

a. Optimisasi Kalkulasi

Tensorflow menggunakan optimizer yang memiliki mekanisme tambahan seperti, gradient clipping, numerical stability tweaks, yang mungkin berkontrubi terhadap sensitivitas model terhadap learning rate.

b. Batch Handling

Batch size yang cukup besar (1000) mungkin diolah dengan cara berbeda oleh tensorflow.

c. Inisialisasi Weight

Walaupun menggunakan jenis inisialisasi yang sama (Xavier/glorot), bisa jadi Tensorflow memiliki sedikit perbedaan dalam implementasinya.

```
Custom R² Score: 0.3949727679645779
Sklearn R² Score: 0.407552408335215
Custom Mean Squared Error: 22510.848060621247
Tensorflow Mean Squared Error: 22042.805701456677
```

3. Improvement

Untuk mempercepat waktu kalkulasi, model perlu menggunakan sparse matriks. Selain itu, penambahan optimizer dan loss function lain juga bisa menambah versatilitas model.

Bonus

Jelaskan apa yang dimaksud dengan convolution layer dan max pooling layer!

Convolution layer adalah lapisan utama dalam Convolutional Neural Network (CNN) yang dirancang untuk mengenali pola-pola dalam data, khususnya dalam gambar. Lapisan ini menerapkan operasi matematika yang disebut "convolution" pada input, di mana filter (atau kernel) berukuran kecil, biasanya 3x3 atau 5x5, bergerak melintasi input gambar. Setiap kali filter bergerak, ia melakukan operasi dot product antara filter dan bagian input yang dilaluinya, menghasilkan sebuah output yang dikenal sebagai "feature map".

Max pooling layer adalah lapisan yang mengikuti lapisan convolution dalam CNN. Fungsinya adalah untuk mengurangi dimensi spasial dari feature map sambil

tetap mempertahankan informasi penting. Max pooling melibatkan pengambilan nilai maksimum dari sekumpulan nilai dalam suatu region (biasanya dengan size 2x2 atau 3x3, size merupakan hyperparameter) dari feature map.

Jelaskan mengapa CNN relatif memiliki kinerja yang lebih baik dibandingkan FC layer untuk kasus klasifikasi gambar!

CNN memiliki parameter yang jauh lebih sedikit daripada Fully Connected (FC) layer karena CNN menggunakan filter yang sama di seluruh gambar. Sebaliknya, FC layer menghubungkan setiap neuron dari satu lapisan ke setiap neuron dari lapisan berikutnya, yang memerlukan lebih banyak parameter, terutama ketika dimensi input tinggi (misalnya, gambar resolusi tinggi).

Gambar memiliki struktur spasial di mana piksel yang berdekatan cenderung memiliki korelasi kuat. CNN dirancang untuk mengeksploitasi korelasi lokal ini melalui convolution, sedangkan FC layer tidak memperhitungkan informasi spasial ini dan memperlakukan setiap piksel secara independen.

CNN dapat mengenali fitur-fitur pada gambar, dari yang sederhana (seperti tepi atau sudut) hingga yang lebih kompleks (seperti bentuk atau objek). Hal ini memungkinkan CNN untuk lebih efektif dalam menangkap informasi penting dari gambar.

CNN lebih tahan terhadap perubahan kecil dalam gambar, seperti translasi, rotasi, atau skala karena penggunaan pooling layer dan arsitektur convolutional. FC layer tidak memiliki properti ini dan lebih rentan terhadap perubahan kecil dalam input.