Generalized Regression Neural Network

(by Abdi Negara Guci S.Mat)

Halo Teman - Teman, Kali ini saya akan menjelaskan terkait sebuah metode *Neural Network* , metode ini bernama **Generalized Regression Neural Network (GRNN)**. Metode ini merupakan jenis *Neural Network* yang umum digunakan untuk tugas regresi, yang melibatkan suatu prediksi nilai kontinu berdasarkan suatu *Input* data. GRNN ini juga menurut saya sangat unik, karena menggunakan suatu kernel *Radial Basis Function* (RBF) yang biasanya kita temui kalau kita belajar lebih mendalam tentang kernel pada *Support Vector Machine*

Karakteristik GRNN

Berikut adalah beberapa karakteristik kunci dari GRNN:

1. **Fungsi Basis Radial (RBF)**: GRNN menggunakan fungsi basis radial untuk mentransformasi data masukan ke dalam ruang fitur berdimensi lebih tinggi. Fungsi basis radial adalah fungsi matematika yang bergantung hanya pada jarak dari titik pusat, yang dikenal sebagai pusat atau prototipe. Fungsi basis radial yang paling umum digunakan dalam GRNN adalah fungsi Gaussian.
2. **Pembelajaran Sekali Jalan:** GRNN melakukan proses pembelajaran sekali jalan, yang berarti mereka belajar set data latih segera setelah disajikan. Hal ini membuatnya efisien terutama untuk tugas regresi dengan kumpulan data berukuran kecil hingga sedang.
3. **Pelatihan Cepat:** Pelatihan GRNN umumnya lebih cepat dibandingkan dengan jenis jaringan saraf lainnya karena tidak melibatkan proses optimisasi iteratif seperti gradien turun. Sebaliknya, ia menghitung bobot selama pelatihan berdasarkan data pelatihan dan menyimpannya untuk digunakan nanti selama prediksi.
4. **Pendekatan Universal:** Seperti banyak arsitektur jaringan saraf lainnya, GRNN adalah pendekatan fungsi universal, yang berarti mereka memiliki kapasitas untuk memperkirakan fungsi kontinu apa pun dengan akurasi sewenang-wenang dengan jumlah neuron tersembunyi yang cukup.
5. **Tugas Regresi:** Sementara beberapa jaringan saraf dirancang untuk tugas klasifikasi (di mana keluarannya adalah kategori atau label), GRNN secara khusus ditujukan untuk tugas regresi, di mana tujuannya adalah untuk memprediksi variabel keluaran kontinu berdasarkan fitur masukan.

Pendekatan Matematika GRNN

Untuk membahas GRNN secara matematis, maka pada kesempatan kali ini saya berusaha memahami konsepnya dari jurnal aslinya langsung dari bapak *Donald F. Specht* pada papernya berjudul ***A General Regression Neural Network*** Namun sebelum itu, ada beberapa konsep matematis yang menurut saya perlu dipahami lebih lanjut untuk itu, simak beberapa penjelasan berikut ini.

1. **Fungsi Kepadatan peluang Bersama:** Jika X dan Y adalah dua variabel acak kontinu, maka fungsi kepadatan peluang bersama $f(x, y)$ untuk X dan Y adalah fungsi yang memenuhi sifat berikut:

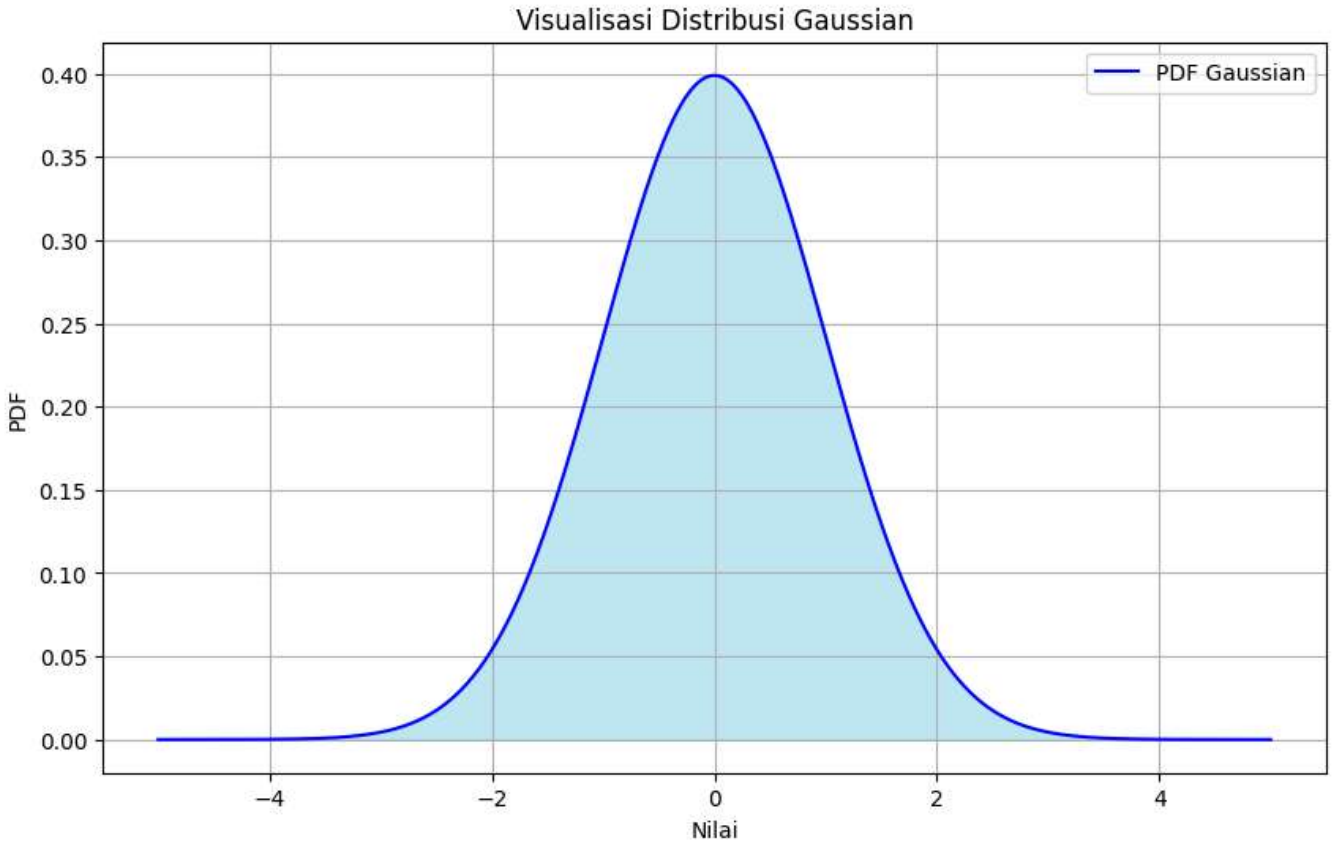
- $f(x, y) \geq 0$ untuk setiap (x) dan (y).
- $\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y)\, dx\, dy = 1$, yaitu total area di bawah kurva $f(x, y)$ sama dengan 1.
- Untuk setiap daerah R di bidang xy, probabilitas X dan Y jatuh dalam R adalah $P((X, Y) \in R) = \iint_R f(x, y)\, dx\, dy$.
- Misalkan (X) adalah variabel acak kontinu dengan fungsi kepadatan peluang:

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \ldots (1)$$

di mana μ adalah rata-rata (mean) dari distribusi X dan σ^2 adalah varians (variance) dari distribusi X. Persamaan (1) adalah bentuk standar dari fungsi kepadatan peluang untuk distribusi normal (Gaussian), di mana μ adalah rata-rata (mean) dan σ^2 adalah varians (variance) dari distribusi X.

Contoh $f(x)$ atau nilai PDF pada distribusi Gaussian

Show code



2. **Window Parzen:** jendela Parzen atau Window Parzen adalah salah satu metode non-parametrik untuk estimasi fungsi kepadatan peluang (PDF) dari data. Metode ini termasuk dalam kategori metode kernel density estimation (KDE), yang digunakan untuk mengevaluasi distribusi probabilitas dari sampel data

   Secara umum, untuk titik data x, PDF diestimasi sebagai jumlah dari kontribusi dari setiap titik data yang terletak di sekitarnya, yang diberi bobot sesuai dengan jarak dari titik data tersebut ke titik x.Rumus umum untuk estimasi PDF menggunakan metode jendela Parzen adalah:

   $$\hat{f}\,(x) = \frac{1}{n}\sum_{i=1}^{n} K_h(x - x_i) \ldots (2)$$

   Persamaan (2) Menyatakan Bahwa:

   - $\hat{f}\,(x)$ adalah estimasi PDF di titik x.
   - n adalah jumlah total titik data.
   - x_i adalah titik data.

- K_h adalah fungsi jendela dengan lebar jendela h.

Fungsi jendela K_h memungkinkan bobot titik data yang lebih dekat dengan x memiliki kontribusi yang lebih besar dalam estimasi PDF. Salah satu fungsi jendela yang umum digunakan adalah fungsi Gaussian.

Metode jendela Parzen dapat digunakan untuk memodelkan distribusi probabilitas dari data yang tidak mengikuti distribusi tertentu dengan baik, atau ketika tidak ada asumsi tertentu tentang distribusi data yang dapat dibuat. Namun, pemilihan bandwidth h yang tepat adalah penting dalam penggunaan metode ini. Jika bandwidth terlalu kecil, estimasi PDF akan terlalu bergejolak, sedangkan jika bandwidth terlalu besar, estimasi akan terlalu halus dan bisa kehilangan detail penting dari data.

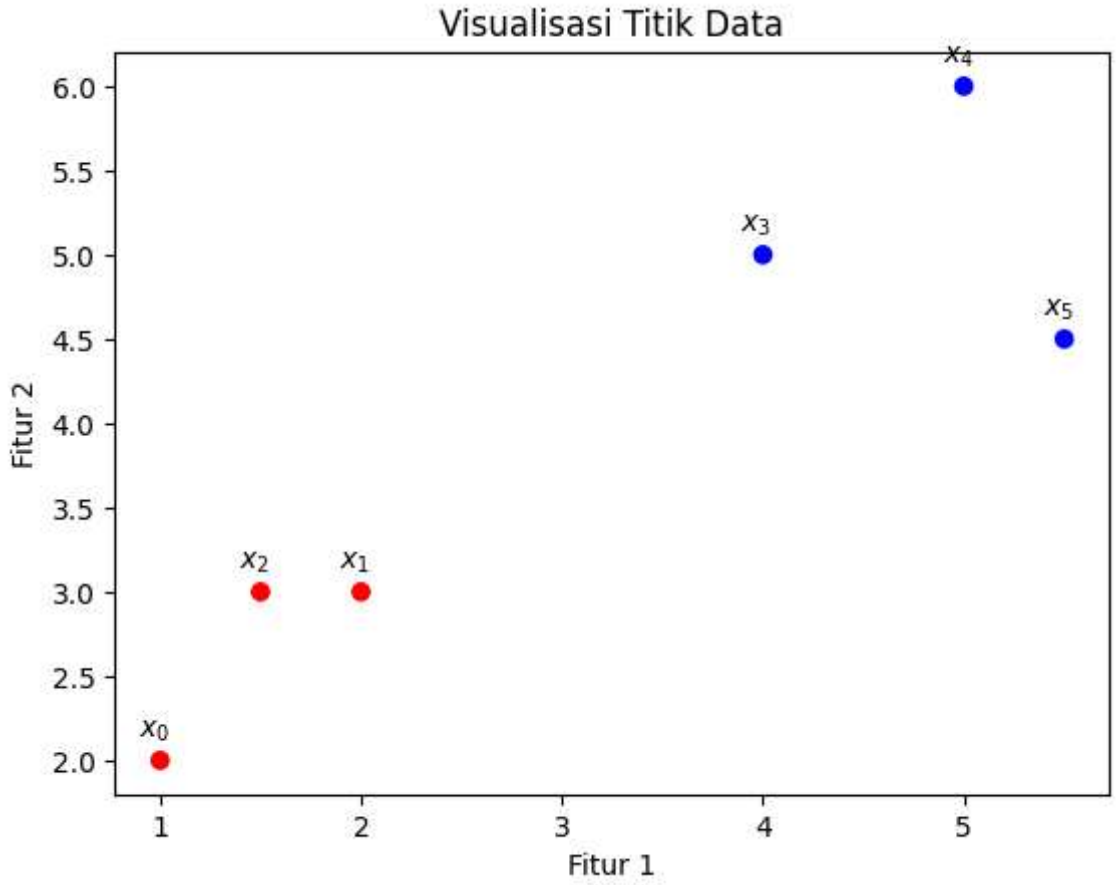
3. **Ekspektasi** $f(\mathbf{X}, Y)$: Berikut adalah rumus untuk menghitung ekspektasi bersyarat dalam konteks vektor:

$$E(Y|X) = \frac{\int_{-\infty}^{\infty} y \cdot f_{Y|X}(y|x)\, dy}{\int_{-\infty}^{\infty} f_{Y|X}(y|x)\, dy} \ldots (3)$$

di mana $f_{Y|X}(y|x)$ adalah fungsi kepadatan peluang bersyarat (PDF) dari Y diberikan $X = x$.Dalam konteks vektor, kita juga harus mempertimbangkan distribusi bersyarat dari X. Jika X adalah vektor, kita juga perlu memperhitungkan kondisi untuk setiap elemen dalam vektor X. Rumus di atas memberikan ekspektasi dari Y yang diberikan nilai X, yang dibagi oleh ekspektasi bersama dari X. Ini memungkinkan kita untuk mengetahui nilai rata-rata dari Y ketika nilai X diketahui.

Pengaruh Kernel RBF terhadap Kemiripan Data

Show code




```
Hasil Kernel Gaussian (Kesamaan Antar Data):
Kernel(Gaussian)(x_0, x_0) = 1.000
Kernel(Gaussian)(x_0, x_1) = 0.368
Kernel(Gaussian)(x_0, x_2) = 0.535
Kernel(Gaussian)(x_0, x_3) = 0.000
Kernel(Gaussian)(x_0, x_4) = 0.000
Kernel(Gaussian)(x_0, x_5) = 0.000
Kernel(Gaussian)(x_1, x_0) = 0.368
Kernel(Gaussian)(x_1, x_1) = 1.000
Kernel(Gaussian)(x_1, x_2) = 0.882
Kernel(Gaussian)(x_1, x_3) = 0.018
Kernel(Gaussian)(x_1, x_4) = 0.000
Kernel(Gaussian)(x_1, x_5) = 0.001
Kernel(Gaussian)(x_2, x_0) = 0.535
Kernel(Gaussian)(x_2, x_1) = 0.882
Kernel(Gaussian)(x_2, x_2) = 1.000
Kernel(Gaussian)(x_2, x_3) = 0.006
Kernel(Gaussian)(x_2, x_4) = 0.000
Kernel(Gaussian)(x_2, x_5) = 0.000
Kernel(Gaussian)(x_3, x_0) = 0.000
Kernel(Gaussian)(x_3, x_1) = 0.018
Kernel(Gaussian)(x_3, x_2) = 0.006
Kernel(Gaussian)(x_3, x_3) = 1.000
Kernel(Gaussian)(x_3, x_4) = 0.368
Kernel(Gaussian)(x_3, x_5) = 0.287
Kernel(Gaussian)(x_4, x_0) = 0.000
Kernel(Gaussian)(x_4, x_1) = 0.000
Kernel(Gaussian)(x_4, x_2) = 0.000
Kernel(Gaussian)(x_4, x_3) = 0.368
Kernel(Gaussian)(x_4, x_4) = 1.000
```

```
Kernel(Gaussian)(x_4, x_5) = 0.287
Kernel(Gaussian)(x_5, x_0) = 0.000
Kernel(Gaussian)(x_5, x_1) = 0.001
Kernel(Gaussian)(x_5, x_2) = 0.000
Kernel(Gaussian)(x_5, x_3) = 0.287
Kernel(Gaussian)(x_5, x_4) = 0.287
Kernel(Gaussian)(x_5, x_5) = 1.000
```

ChatGPT

```
import numpy as np
import math
import pandas as pd
class GRNN :

    def __init__(self,x_train,y_train,x_test,y_test):

        self.x_train= x_train
        self.y_train= y_train
        self.x_test= x_test
        self.y_test= y_test

        self.std     = np.ones((1,self.y_train.size))#np.random.rand(1,self.train_y.size) #St

    def activation_func(self,distances): # gaussian kernel

        return np.exp(- (distances**2) / 2*(self.std**2) )

    def output(self,i):#sometimes called weight

        distances=np.sqrt(np.sum((self.x_test[i]-self.x_train)**2,axis=1)) # euclidean distan

        return self.activation_func(distances)

    def denominator(self,i):

        return np.sum(self.output(i))

    def numerator(self,i):

        return  np.sum(self.output(i) * self.y_train)

    def predict(self):

        predict_array = np.array([])

        for i in range(self.y_test.size):
            predict=np.array([self.numerator(i)/self.denominator(i)])
            predict_array=np.append(predict_array,predict)

        return predict_array

    def mean_squared_error(self):
```