K-NEAREST NEIGHBOR

Presented By : Siska Atmawan Oktavia, S.T., M.Cs.

2024

OVERVIEW

- Pengertian Knn
- Jarak Atribut Numerik
- Jarak Atribut Non Numerik
- Algoritma Knn
- Optimasi KNN
- Perbaikan KNN

Definisi Algoritma K-Nearest Neighbor

Algoritma K-Nearest Neighbor adalah suatu metode algoritma klasifikasi yang bekerja berdasarkan tingkat kemiripan yang dihitung berdasarkan jarak (distance) terdekat dari data pembelajarannya (data latih dan data uji) (Boiculese, Dimitriu and Moscalu, 2013).

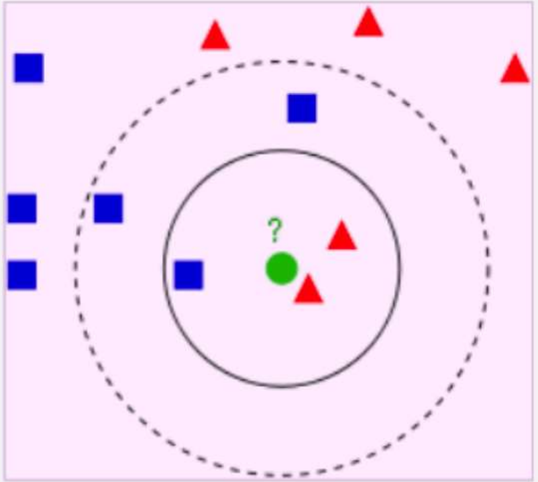
Apa itu K-Nearest Neighbor?

- Termasuk dalam pembelajaran supervised learning / ada kelas output
- Hasil query instance yang baru diklasifikasikan berdasarkan mayoritas kedekatan jarak dari kategori yang ada dalam KNN
- Bisa dipakai untuk masalah classification dan regression
- Menggunakan rumus perhitungan jarak seperti Euclidean distance, Hamming distance, Manhattan distance dan Minkowski distance.

Contoh Algoritma K-Nearest Neighbor

Jika K = 3, maka lingkaran hijau akan masuk dalam kategori segitiga merah

Jika K=5, maka lingkaran hijau akan masuk dalam kategori segiempat biru



Dataset

Dataset dibagi menjadi dua yaitu **Data Training** (data latih) dan **Data Testing** (data uji). Data training adalah data yang sudah memiliki kelas, sedangkan data testing (data uji) merupakan data yang akan dicari kelasnya.

Data Training (data latih) berperan sebagai pembentuk suatu pola/model/pengetahuan, dan **Data Testing** (data uji) sebagai alat ukur evaluasi algoritma.

Metrik Jarak

- **Euclidean Distance** → ukuran jarak yang paling umum, yang mengukur garis lurus antara titik kueri dan titik lain yang sedang diukur.
- **Hamming Distance** → metrik tumpang tindih, adalah teknik yang digunakan dengan vektor Boolean atau string untuk mengidentifikasi di mana vektor tidak cocok. Metode ini mengukur jarak antara dua string dengan panjang yang sama. Metode ini sangat berguna untuk kode deteksi kesalahan dan koreksi kesalahan.
- **Manhattan Distance** → ukuran jarak yang populer, yang mengukur nilai absolut antara dua titik. Jarak ini direpresentasikan pada kisi, dan sering disebut sebagai geometri taksi - bagaimana Anda bepergian dari titik A (titik kueri Anda) ke titik B (titik yang sedang diukur)?
- **Minkowski Distance** →

Euclidean Distance

Metode Euclidean distance secara umum digunakan untuk data numerik. Untuk rumus perhitungan Euclidean distance ada pada perhitungan 4.1.

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Keterangan:

d(x,y) = jarak kedekatan antara data x ke data y

x_i = data testing (data uji) ke i

y_i = data training (data latih) ke i

n = jumlah atribut 1 sampai n

Hamming Distance

Jarak Hamming mengukur perbedaan antara dua string atau dua kategori dengan cara menghitung jumlah posisi yang berbeda antara dua objek. Ini sering digunakan untuk atribut yang berupa nilai kategorikal atau biner (misalnya, "ya" atau "tidak", "A" atau "B").

Rumus:

$$d(p, q) = \sum_{i=1}^{n} \delta(p_i, q_i)$$

Di mana $\delta(p_i, q_i) = 0$ jika $p_i = q_i$, dan 1 jika $p_i \neq q_i$.

Contoh: Jika kita membandingkan dua string kategorikal:

- String 1: ["Merah", "Biru", "Hijau"]
- String 2: ["Merah", "Kuning", "Hijau"]

Jarak Hamming adalah 1, karena ada 1 perbedaan pada kategori kedua ("Biru" vs "Kuning").

Manhattan Distance

Jarak ini adalah jumlah dari perbedaan absolut antara dua titik pada setiap dimensi. Manhattan distance juga dikenal sebagai **L1 norm** dan digunakan saat pengukuran lebih sesuai dengan konsep "jarak blok kota" atau jarak "jalur zig-zag".

Rumusnya adalah:

$$d(p, q) = \sum_{i=1}^{n} |p_i - q_i|$$

Contoh Penggunaan: Jarak Manhattan sering digunakan dalam analisis ketika perubahan kecil di satu dimensi lebih signifikan daripada perubahan besar di dimensi lain, seperti pada grid city (tata kota berbasis blok).

Minkowski Distance

Jarak Minkowski merupakan generalisasi metrik jarak Euclidean dan Manhattan, yang memungkinkan terciptanya metrik jarak lainnya. Jarak ini dihitung dalam ruang vektor bernorma. Dalam jarak Minkowski, p adalah parameter yang menentukan jenis jarak yang digunakan dalam perhitungan. Jika p=1, maka jarak Manhattan digunakan. Jika p=2, maka jarak Euclidean digunakan.

Contoh: Menentukan Jarak Atribut Numerik Menggunakan Rumus Euclidean Distance

Jurnal: Perbandingan penghitungan jarak pada k-nearest neighbour dalam klasifikasi data tekstual . Penelitian ini membandingkan empat ukuran jarak, yaitu Euclidean, Manhattan, Minkowski, dan Chebyshev

Tabel 1 Akurasi tiap model percobaan terhadap nilai K

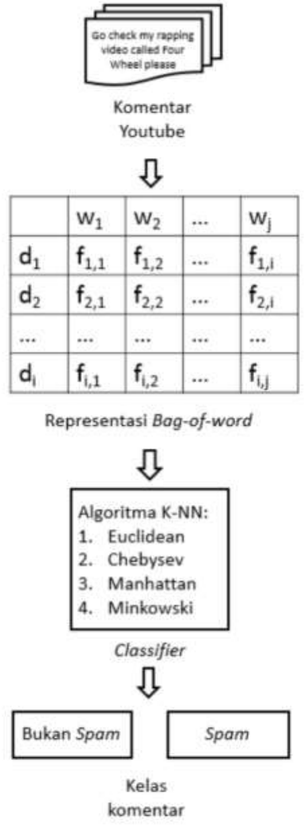
K	Euclidean	Chebyshev	Manhattan	Minkowski
1	0,8504	0,6585	0,8548	0,8504
3	0,8550	0,6187	0,8505	0,8550
5	0,8372	0,6187	0,8325	0,8372
7	0,8372	0,6096	0,8259	0,8372
9	0,8216	0,6183	0,8079	0,8216
11	0,8128	0,6119	0,7968	0,8128
13	0,8041	0,6359	0,7814	0,8041
15	0,8041	0,6359	0,7814	0,8041
17	0,7971	0,6047	0,7724	0,7971
19	0,7972	0,5960	0,7679	0,7972

$$D(d_x, d_y) = \sqrt{(f_{x,1} - f_{y,1})^2 + \ldots + (f_{x,j} - f_{y,j})^2} \quad (1)$$

$$D(d_x, d_y) = max\ (|f_{x,1} - f_{y,1}|, \ldots, |f_{x,j} - f_{y,j}|) \quad (2)$$

$$D(d_x, d_y) = |f_{x,1} - f_{y,1}| + \ldots + |f_{x,j} - f_{y,j}| \quad (3)$$

$$D(d_x, d_y) = \sqrt[p]{(f_{x,1} - f_{y,1})^p + \ldots + (f_{x,j} - f_{y,j})^p} \quad (4)$$



Gambar 2. Alur penelitian keseluruhan

Contoh: Menentukan Jarak Atribut **Non-Numerik** Menggunakan Rumus Hamming Distance

Jurnal: Penerapan Algoritma K-Nearest Neighbors dalam Sistem Rekomendasi Makanan Berdasarkan Kebutuhan Nutrisi dengan Content-Based Filtering. (Luqyana Zakiya Almas, 2024)

Data

Tabel 1. Variabel Penelitian

No	Variabel	Keterangan	Jenis Data
1	Makanan	Nama Makanan	*String*
2	Kalori	Jumlah Kalori dalam Makanan	*Numeric*
3	Protein	Kandungan Protein	*Numeric*
4	Lemak	Kandungan Lemak	*Numeric*
5	Serat	Kandungan Serat	*Numeric*
6	Bahan	Resep dan Bahan Makanan	*String*

Rumus yang digunakan pada jurnal ini:

$$d(x,y) = \sqrt{\sum_{k=1}^{n}(x_k - y_k)^2}$$

Algortima K-Nearest Neighbor

- **Tentukan nilai k**: Pilih jumlah tetangga terdekat, yaitu k.
- **Hitung jarak**: Untuk setiap data uji, hitung jarak antara data uji dan setiap data pelatihan menggunakan metrik jarak, misalnya, Euclidean, Manhattan, atau Minkowski.
- **Urutkan tetangga**: Urutkan seluruh data pelatihan berdasarkan jarak yang telah dihitung.
- **Pilih k tetangga terdekat**: Ambil k data pelatihan terdekat.
- **Klasifikasi**: Tentukan kelas mayoritas dari tetangga terdekat.
- **Regresi**: Ambil rata-rata nilai dari tetangga terdekat.

Prediksi hasil: Tentukan kelas atau nilai yang diprediksi untuk data uji.

Optimasi Algoritma K-Neares Neighbor

Optimasi algoritma *K-Nearest Neighbor* (K-NN) bertujuan untuk meningkatkan kinerja dalam hal akurasi, kecepatan, atau efisiensi memori. Ada beberapa cara untuk mengoptimalkan K-NN, baik dari sisi parameter maupun dari metode pemrosesan datanya.

Beberapa pendekatan optimasi yang bisa dilakukan:

1. Normalisasi Data
2. Pemilihan Nilai K yang Optimal
3. Algoritma Pencarian Jarak yang Cepat
4. Dimensionality Reduction (Reduksi Dimensi)
5. Weighted K-NN

1. Normalisasi Data

Normalisasi Data adalah K-NN sangat sensitif terhadap skala data karena mengukur jarak antara titik data. Jika data memiliki skala yang berbeda-beda, fitur dengan nilai besar akan mendominasi perhitungan jarak. Oleh karena itu, penting untuk menormalkan atau menstandarkan data menggunakan metode seperti *min-max scaling* atau *z-score normalization*.

2. Pemilihan Nilai K yang Optimal

- Memilih nilai K yang tepat sangat penting. K terlalu kecil dapat menyebabkan *overfitting*, sedangkan K terlalu besar dapat menyebabkan *underfitting*.
- Untuk menentukan nilai K yang optimal, gunakan teknik seperti *cross-validation*. Biasanya, uji dengan beberapa nilai K dan pilih yang memberikan akurasi terbaik.

3. Algoritma Pencarian Jarak yang Cepat

Karena K-NN mengukur jarak antara data setiap kali memprediksi, optimasi proses pencarian jarak akan sangat meningkatkan kinerja, terutama pada dataset besar.

Beberapa metode pencarian cepat:

- **KD-Tree**: Mengorganisir data dalam bentuk struktur pohon untuk mengurangi jumlah perhitungan jarak.
- **Ball Tree**: Mirip dengan KD-Tree, tetapi lebih cocok untuk data berdimensi tinggi.
- **Approximate Nearest Neighbor (ANN)**: Alih-alih menghitung jarak untuk semua titik data, metode ini memperkirakan tetangga terdekat dengan lebih cepat, walaupun dengan sedikit kompromi pada akurasi.

4. Dimensionality Reduction (Reduksi Dimensi)

Pada dataset dengan banyak fitur, mengurangi dimensi dapat mempercepat perhitungan K-NN. Beberapa metode untuk reduksi dimensi:

- **Principal Component Analysis (PCA)**: Mentransformasikan data ke ruang baru dengan variansi maksimum.
- **Linear Discriminant Analysis (LDA)**: Mengurangi dimensi dengan memaksimalkan pemisahan antar kelas.
- **t-SNE atau UMAP**: Teknik yang lebih kompleks untuk visualisasi atau reduksi dimensi.

5. Weighted K-NN

Dalam K-NN standar, semua tetangga memiliki bobot yang sama. Namun, kita bisa memberi bobot lebih besar pada tetangga yang lebih dekat daripada yang lebih jauh. Pendekatan ini sering menghasilkan akurasi yang lebih tinggi, terutama jika data tidak homogen.

Langkah-Langkah algoritma KNN

1. Tentukan nilai parameter K (penentuan nilai k dipilih secara manual)
2. Hitung jarak antara data training (data latih) dan data testing (data uji)
3. Urutkan data training berdasarkan jarak yang terbentuk (dari terkecil ke terbesar)
4. Tetapkan kelas yang bersesuaian, pilihlah kelas dengan jumlah nilai k terbanyak pada data testing (data uji)

Kelebihan Algoritma KNN

1. KNN tidak memerlukan training sebelum prediksi, sehingga penambahan data baru dapat dilakukan secara mudah tanpa mengurangi keakuratannya.
2. KNN termasuk ke Lazy Learner sehingga KNN lebih cepat dibandingkan algoritma lainnya.
3. KNN sangat mudah diimplementasikan, karena hanya menggunakan dua parameter, yaitu nilai K dan fungsi jarak.

Kekurangan algoritma KNN :

1. Tidak mampu menangani data yang besar, karena dengan

2.data yang besar performa kinerja algoritma menurun yang disebabkan penghitungan jarak antara titik baru dengan yang ada lama.

2. Tidak dapat bekerja dengan baik ketika menangani dimensi yang tinggi. Karena dengan dimensi yang tinggi dan besar, algoritma KNN akan sulit untuk menghitung jarak setiap dimensinya.

3. Diperlukannya standarisasi dan normalisasi sebelum penerapan algoritma KNN ke dataset. Jika tidak melakukannya algoritma KNN akan menghasilkan prediksi yang salah.

4. Algoritma KNN sangat sensitif terhadap noise dataset, sehingga diperlukan penghitungan secara manual terhadap nilai yang hilang secara manual.

Optimasi KNN

KD-Tree dan Ball Tree: Untuk dataset yang besar, digunakan struktur data seperti KD-Tree atau Ball Tree untuk mempercepat pencarian tetangga terdekat.

Dimensionality Reduction: Pengurangan dimensi menggunakan algoritma seperti PCA dapat membantu mengurangi kompleksitas komputasi dengan mengurangi jumlah fitur.

Implementasi K-NN: K-NN dapat diimplementasikan menggunakan berbagai library dalam bahasa pemrograman seperti Python dengan **scikit-learn**:

Thank you for your attention friends, I hope you can implementation in your next project.