Artificial Intelligence
Kuliah 5: k-Nearest Neighbours

Ali Akbar Septiandri
Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Al Azhar Indonesia
aliakbars@live.com

1 Bertanya kepada Tetangga

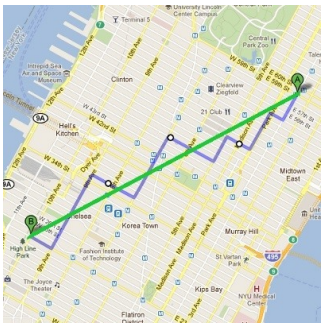
Secara harfiah, *nearest neighbours* berarti tetangga terdekat. Jadi, algoritma ini dapat didefinisikan sebagai pengambilan keputusan dengan menjadikan keputusan tetangga-tetangga terdekat sebagai referensi. Sebagai contoh, ketika Anda bingung untuk mengambil keputusan siapa yang harus Anda pilih saat pilkada, maka salah satu referensi yang umum digunakan adalah dengan bertanya ke teman atau anggota keluarga terdekat Anda. Pertanyaannya, apa yang mendefinisikan "terdekat"?

Dalam contoh kasus di atas, seringnya interaksi dengan orang tersebut atau bisa jadi karena tinggal di satu rumah membuat Anda lebih mudah untuk mempercayai keputusan mereka. Jadi, kalau Anda menggunakan keputusan ibu atau ayah Anda, maka definisi **jarak** dalam kasus ini adalah kedekatan tempat tinggal. Sebaliknya, jika Anda menggunakan keputusan sahabat Anda di kampus, maka jaraknya ditentukan oleh intensitas interaksi per hari, dengan asumsi bahwa Anda menghabiskan lebih banyak waktu untuk berinteraksi dengan teman-teman di kampus dibandingkan keluarga di rumah.

Secara matematis, umumnya kita mendefinisikan jarak dalam **Euclidean distance**. Jarak antara titik a dan b dengan Euclidean distance dapat dihitung dengan

$$d([a_1, a_2, ..., a_d], [b_1, b_2, ..., b_d]) = \sqrt{\sum_{i=1}^{d} (a_i - b_i)^2}$$

dengan d menunjukkan jumlah dimensi yang digunakan. Dalam Gambar 1, Euclidean distance (L_2-norm) digambarkan sebagai garis hijau antara titik A dan B. Sebagai pembanding, jumlah panjang garis ungu merupakan metode pengukuran jarak lain yang biasa disebut sebagai **Manhattan distance** (L_1-norm).



Gambar 1: Perhitungan jarak dengan Euclidean dan Manhattan distance

Pertanyaan berikutnya adalah, “Bagaimana jika kita ingin mengambil keputusan dengan referensi lebih dari satu tetangga terdekat?” Sederhananya, Anda dapat mengambil referensi dari k tetangga terdekat, lalu mengambil keputusan berdasarkan **label mayoritas**. Perhatikan bahwa untuk melakukan hal ini (termasuk untuk hanya satu tetangga terdekat), Anda perlu menghitung jarak untuk setiap pasangan data yang ada (*pairwise distance*). Oleh karena itu, algoritma ini bisa jadi sangat mahal dari sisi komputasi. Dalam notasi big-O, kompleksitas algoritma *nearest neighbours* tanpa modifikasi adalah $O(nd)$ dengan n adalah jumlah data dan d adalah jumlah dimensi atau atribut yang digunakan.

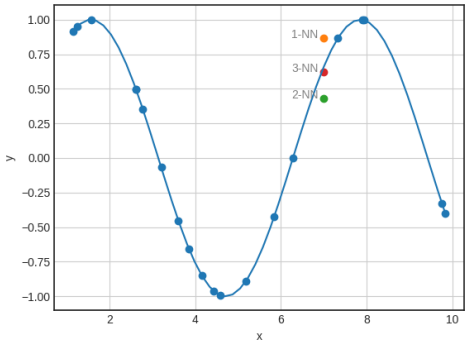
Sebagai contoh, klasifikasi dengan 7-NN pada data MNIST dengan menjadikan setiap pixel (ada 784 pixels dalam satu gambar) sebagai atribut akan menghasilkan prediksi seperti pada Gambar 2. Gambar angka di paling kanan yang berada dalam kotak adalah gambar yang ingin kita prediksi nilainya, “Angka berapa yang tertera pada gambar/tulisan tersebut?” Tujuh gambar di sebelah kirinya adalah gambar yang telah kita ketahui labelnya. Perhatikan bahwa tetangga terdekatnya tidak selalu menunjukkan angka yang sama, karena yang diperhatian hanyalah gambar mana yang punya konfigurasi pixel yang paling mirip? Jadi, gambar nomor empat dari atas akan diprediksi sebagai angka 1.

Gambar 2: Klasifikasi 7-NN pada MNIST

Ide yang sama kemudian bisa digunakan untuk melakukan regresi. Gambar 3 menunjukkan bahwa jika diketahui $x = 7$, maka nilai $\hat{y}$ adalah **nilai rata-rata** dari k tetangga terdekat. Secara matematis, hal ini dapat ditulis sebagai

$$\hat{y} = f(x) = \frac{1}{k} \sum_{j=1}^{k} y_{i_j}$$

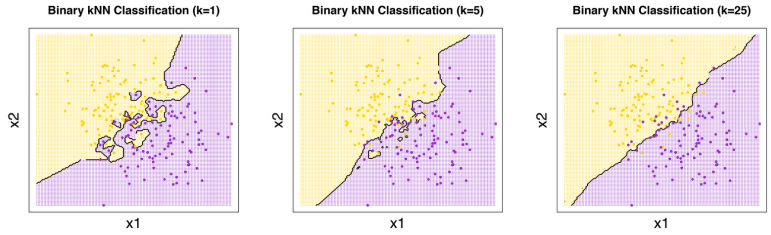
Dalam kasus tersebut, 3-NN menghasilkan error yang paling kecil. Jadi, apakah nilai $k = 3$ selalu menghasilkan prediksi terbaik?



Gambar 3: Regresi dengan k-NN

2 Beberapa Isu

Masalahnya, tidak ada nilai yang pasti untuk nilai k terbaik. Untuk nilai k yang terlalu besar, maka $\hat{y} = P(y)$ (sesuai dengan proporsi label dalam data) pada kasus klasifikasi atau $\hat{y} = \bar{y}$ (rata-rata label) pada kasus regresi. Sebaliknya, saat k terlalu kecil, batas keputusannya menjadi tidak stabil[1] dan proses inferensinya menjadi terlalu variatif seperti diilustrasikan dalam Gambar 4. Jadi, sebaiknya kita membagi datanya agar mempunyai data validasi yang dapat digunakan untuk menguji saat kita melakukan ***hyperparameter tuning***.



Gambar 4: Pengaruh nilai k pada batas keputusan

Isu lain yang mungkin terjadi pada algoritma k-NN adalah kemungkinan untuk mendapatkan hasil seri pada kasus klasifikasi. Jika kita memilih k genap dan kedua kelas sama kuat, maka alternatif solusi untuk masalah ini adalah:

1. Gunakan k ganjil sejak awal
2. Pilih kelas secara acak, e.g. dengan lemparan koin
3. Gunakan *prior probability* $P(y)$ (proporsi kelas dalam keseluruhan dataset)
4. 1-NN

Pertanyaan untuk Anda, "Apakah kasus yang serupa juga bisa terjadi pada regresi?"

k-NN juga punya masalah dari sisi penanganan *missing values*. Karena konsep jarak akan sangat bergantung pada nilai tiap atribut, maka nilai yang hilang tersebut harus diganti (***impute***). Anda dapat merujuk ke [1] untuk cara-cara imputasi[2].

Satu masalah lainnya dengan k-NN adalah pengukuran jarak yang digunakan membuat algoritma ini rentan terhadap perbedaan rentang variabel. Pada Gambar 5, titik x dan y terlihat berdekatan, padahal secara rentang, posisinya bisa jadi lebih jauh jika dibandingkan dengan jarak x dan z[3]. Hal seperti ini yang tidak bisa ditangkap secara langsung oleh mesin karena perhitungannya hanya didasarkan dengan angka tanpa normalisasi.

3 Evaluasi dan Generalisasi

Seperti yang sempat dibahas di bagian sebelumnya, penggunaan data validasi kadang diperlukan untuk proses *hyperparameter tuning*. Dalam kasus k-NN, nilai k merupakan *hyperparameter* yang dapat diubah. Masalahnya, jika kita mengubah nilainya dan mencocokkan ke data latih, maka sangat mungkin terjadi *overfitting*.

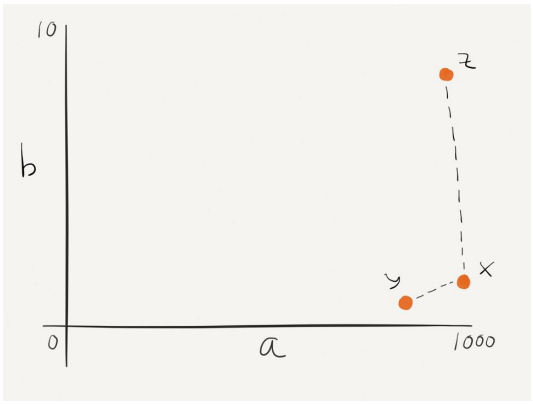
Dalam *machine learning*, model yang Anda hasilkan harus bisa bekerja untuk segala jenis data. Karena kita tidak mungkin mendapatkan semua konfigurasi data yang ada, maka solusinya adalah dengan membagi dataset menjadi data latih, validasi, dan uji. Data latih digunakan untuk membentuk model dengan nilai *hyperparameters* yang berbeda. Model yang dibentuk kemudian harus divalidasi pada data validasi. Data uji disimpan sebagai pengujian terakhir dari model yang paling baik. Anda tidak diperbolehkan untuk melihat data uji ini hingga saat pengujian.

Secara definisi, sebuah model F dikatakan mengalami ***overfitting*** jika:

[1] Saat kita punya data untuk semesta, nilai $k = 1$ justru membuat prediksinya *asymptotically correct*

[2] `http://www.stat.columbia.edu/~gelman/arm/missing.pdf`

[3] `https://tentangdata.wordpress.com/2015/08/31/klasifikasi-k-nearest-neighbours/`



Gambar 5: Perbedaan rentang variabel bisa mengacaukan klasifikasi k-NN

1. kita dapat menemukan model lain F'
2. dengan error lebih besar pada data latih: $E_{train}(F') > E_{train}(F)$
3. tetapi error lebih kecil pada data uji: $E_{gen}(F') < E_{gen}(F)$.

Kasus ini dapat terjadi karena model yang dihasilkan terlalu kompleks, terlalu fleksibel. Dengan kata lain, model yang dihasilkan mengenali dan memasukkan *noise* dari dalam data latih ke dalam model. Hal ini berarti bahwa model mengenali *pola yang tidak akan muncul lagi*. Referensi lebih lanjut mengenai *overfitting* bisa dilihat di [2]. Di sisi lain, model tersebut dikatakan mengalami ***underfitting*** jika:

1. model terlalu kaku, terlalu simpel;
2. tidak berhasil menemukan pola yang penting; dan
3. masih ada model yang bisa menghasilkan E_{train} dan E_{gen} yang lebih rendah.

References

[1] Gelman, A. and Hill, J., 2007. *Data analysis using regression and multilevelhierarchical models* (Vol. 1). New York, NY, USA: Cambridge University Press.

[2] Abu-Mostafa, Y.S., 2012. Machines that think for themselves. *Scientific American, 307*(1), pp.78-81.