BAB II

LANDASAN TEORI

2.1 Citra Tanda Tangan

Tanda tangan menurut KBBI adalah "Tanda sebagai lambang nama yang dituliskan dengan tanda tangan oleh orang itu sendiri sebagai penanda pribadi (telah menerima dan sebagainya)". Sampai saat ini tanda tangan merupakan salah satu hal terpenting yang digunakan sebagai bukti dan keterangan dalam media cetak ataupun *digital*. Tanda tangan merupakan hasil dari proses menulis seseorang yang bersifat khusus sebagai substansi simbolik (Karouni, et al., 2013). Sampai saat ini tanda tangan masih sering digunakan sebagai bukti dan informasi yang sah sebagai identifikasi keterkaitan seseorang dalam suatu perjanjian, dokumen, dan sebagainya.

2.2 Pengolahan Citra

Di dalam bidang *Artificial Intelligence*, citra banyak digunakan sebagai input yang akan di proses ke dalam *clustering* dan *classification*. *Clustering* dan *classification* adalah metode penganalisaan data, yang sering dimasukkan sebagai salah satu metode *Data Mining*, yang tujuannya adalah untuk mengelompokkan data dengan karakteristik yang sama ke suatu wilayah yang sama dan data dengan karakteristik yang berbeda ke wilayah yang lain. *Clustering* dipakai ketika tidak diketahuinya bagaimana data harus dikelompokkan. Jumlah kelompok diasumsikan sendiri tanpa ditentukan terlebih dahulu. Keluaran pendekatan ini adalah data yang sudah dikelompokkan. Sedangkan *classification*, terdapat informasi mengenai bagaimana data tersebut dikelompokkan. Kemudian dilakukan *training* pada sistem

dengan data yang sudah diberikan label (ke dalam kelompok manakah data tersebut dikelompokkan), selanjutnya sistem akan mengklasifikasikan data-data yang baru ke dalam kelompok yang ada (Ardiansyah, 2013).

Ketika citra yang diproses memiliki memori yang besar karena memiliki bagian data yang tidak mengandung informasi terkait dan merupakan pengulangan informasi yang sebelumnya telah dinyatakan, akan menghambat dalam proses transmisi data citra. Perkembanganan dalam bidang pengolahan citra *digital* mulai berkembang pada tahun 1960-an yaitu ketika teknologi komputer telah sanggup melakukan penghitungan yang dilakukan oleh berbagai algoritma dengan kecepatan proses serta kapasitas memori yang dibutuhkan (Ardiansyah, 2013).

2.3 Principal Component Analysis (PCA)

Principal Component Analysis (PCA) merupakan salah satu teknik ekstraksi fitur yang digunakan untuk mereduksi data yang memiliki dimensi yang besar/tinggi, menjadi data yang memiliki dimensi lebih kecil tanpa menghilangkan informasi yang menggambarkan keseluruhan data dari citra yang akan diekstraksi fiturnya. PCA bertujuan untuk mencari struktur hubungan antara sejumlah *variable* stokastik yang ditemukan dalam suatu pengamatan, dengan maksud untuk mencari karakteristik pokok data-datanya (Musfiroh, 2015).

Menurut Santosa (2007), *Principal Component Analysis* (PCA) adalah suatu teknik handal yang digunakan mengekstraksi struktur dari suatu set data dengan dimensi yang besar. Masalah di dalam PCA adalah untuk menemukan *eigenvalue* dan *eigenvector*. PCA adalah transformasi *orthogonal* (tegak lurus) dari sistem koordinat dimana data diuraikan. Nilai koordinat baru yang

mempresentasikan data dinamakan *principal component* atau PC. Koordinat ini dipilih dimana variansi dari data mencapai nilai maksimum. Hanya dengan beberapa *principal component* dapat menjelaskan struktur dari data aslinya (Santosa, 2007).

Menurut Turk dan Pentland (1991), PCA melibatkan prosedur matematis yang mentransformasikan beberapa variabel yang memiliki korelasi menjadi kumpulan fitur yang tidak berkolerasi yang jumlahnya lebih sedikit yang disebut *principal component*. Proses ini akan menghasilkan beberapa *eigenvector* yang merupakan kombinasi seluruh variasi fitur yang terdapat dalam seluruh data. Jika objek yang digunakan berupa gambar wajah, *eigenvector* tersebut sering disebut juga *eigenfaces*. Untuk melakukan hal ini, data atau gambar yang akan direduksi dimensinya harus diubah menjadi kumpulan matriks kolom $D1, D2, \dots, \dots, Di$ dimana m merupakan jumlah dari sampel yang tersedia. Rata-rata dari setiap data dapat dihitung dengan Rumus 2.1.

$$M = \frac{1}{m}\sum_{i=1}^{m} D_i \quad \dots (2.1)$$

Keterangan :

M = rata-rata dari setiap data

m = jumlah dari sampel

D_i = dimensi data ke-i

Selisih antara setiap data dengan rata-ratanya dapat direpresentasikan dengan Rumus 2.2.

$$N_i = D_i - M \quad \dots (2.2)$$

Keterangan :

N_i = matriks data ke-i

M = rata-rata dari setiap data

D_i = dimensi data ke-i

Dari matriks N, matriks *covariance* dapat dihitung untuk kemudian digunakan dalam mencari *eigenvector*. Matriks *covariance* dapat dihitung dengan Rumus 2.3.

$$C = \frac{1}{m} N \; x \; N^T \quad \text{... (2.3)}$$

Keterangan :

C = matriks *covariance*

N^T = matriks transformasi

N = matriks

Langkah berikutnya adalah menghitung *eigenvector* dan *eigenvalue* dari matriks *covariance* tersebut. Dari *eigenvector* dan *eigenvalue* yang dihasilkan, dipilih k *eigenvector* yang memiliki *eigenvalue* terbesar. Kumpulan k *eigenvector* tersebut merupakan *eigenfaces* yang dapat digunakan untuk memproyeksikan data ke dalam *eigenface*.

Proses PCA secara sistematis dapat didefinisikan sebagai berikut (Musfiruoh, 2015).

- Misal diketahui suatu populasi $x_{m,n}$ sejumlah *m* data dengan indeks j dan jumlah parameter yang akan diukur adalah *n* indeks *i*, maka populasi tersebut dapat dituliskan dengan Persamaan 2.4.

$$\begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ \vdots \\ X_m \end{bmatrix} = \begin{bmatrix} x_{1,1} & x_{1,2} & x_{1,3} & \cdots & x_{1,n} \\ x_{2,1} & x_{2,2} & x_{2,3} & \cdots & x_{2,n} \\ x_{3,1} & x_{3,2} & x_{3,3} & \cdots & x_{3,n} \\ & \vdots & \vdots & \vdots & \vdots \\ x_{m,1} & x_{m,2} & x_{m,3} & \cdots & x_{m,n} \end{bmatrix} \quad \text{... (2.4)}$$

- Langkah berikutnya adalah mencari nilai rata-rata / *mean centered* parameter ke-*i* dari Persamaaan (2.4).

$$\mu_i = \frac{x_{1,i} + x_{2,i} + x_{3,i} + \cdots\cdots\cdots + x_{m,i}}{m} \quad \text{... (2.5)}$$

$$= \frac{\sum_{j=1}^{m} x_{j,i}}{m}$$

$$= [\mu_1, \mu_2, \mu_3, \cdots\cdots\cdots\cdots, \mu_n]$$

- Model matriks data pelatihan dapat dituliskan dengan menggunakan Persamaan (2.4). Jika n>>m dan n merupakan dimensi citra, maka m merupakan jumlah data yang dilatih. Berdasarkan Persamaaan (2.4), maka rata-rata dari seluruh data sampel dapat dihitung dengan Persamaan (2.5). Hasil Persamaan (2.5) merupakan suatu vektor yang berisi nilai rata-rata, karena jumlah parameter (dimensi) adalah sejumlah n, maka nilai array rata-rata seluruh data adalah $[\mu_1, \mu_2, \mu_3, \cdots\cdots\cdots\cdots, \mu_n]$. Rata-rata nol *atau zero mean* dari suatu sampel dapat dihitung dengan proses pengurangan nilai masing-masing data sampel dengan rata-rata data seluruh sampel. Tetapi dikarenakan dimensi yang tidak sama antara data sampel $(m\ x\ n)$, dan sedangkan dimensi dari rata-rata seluruh data sampel $(i\ x\ n)$, maka perlu disamakan dimensinya dengan menggandakan rata-rata seluruh citra sebanyak m, sehingga rata-rata seluruh data sampel mempunyai dimensi

($m\ x\ n$). Matriks rata-rata citra data sampel yang telah digandakan sebanyak m kali dapat dutulis dengan Persamaan 2.6.

$$\mu = \begin{bmatrix} \mu_{1,1} & \mu_{1,2} & \mu_{1,3} & \cdots & \mu_{1,n} \\ \mu_{2,1} & \mu_{2,2} & \mu_{2,3} & \cdots & \mu_{2,n} \\ \mu_{3,1} & \mu_{3,2} & \mu_{3,3} & \cdots & \mu_{3,n} \\ & \vdots & \vdots & \vdots & \vdots \\ \mu_{m,1} & \mu_{m,2} & \mu_{m,3} & \cdots & \mu_{m,n} \end{bmatrix} \quad \text{... (2.6)}$$

- Hasil dari Persamaan 2.6 digunakan untuk menghitung *zero mean*, yang dapat dimodelkan menggunakan Persamaan 2.7.

$$\emptyset_{j,i} = x_{j,i} - \mu_i \quad \text{... (2.7)}$$

- Hasil perhitungan *zeromean* dari Persamaan 2.7 akan digunakan untuk mecari nilai matrik kovarian dengan mengalikan hasil perhitungan *zeromean* dengan transposenya sendiri, dengan menggunakan Persamaan 2.8.

$$C = \emptyset_{j,i}\emptyset_{j,i}^{t} \quad \text{... (2.8)}$$

- Untuk mendapatkan ciri dari suatu data sampel yang dipresentasikan dalam bentuk matrik, maka dihitung *eigenvector* dan *eigenvalue* dari matrik kovarian dengan Persamaan 2.9.

$$CV = \lambda V \quad \text{... (2.9)}$$

Skalar λ merupakan *eigenvalue* dari *C* dan *V* sebagai *eigenvector* dari *C* yang berpadanan terhadap λ. Persamaan 2.9 akan digunakan untuk mendapatkan *eigenvector* dan *eigenvalue.*

$$(\lambda - C)V = 0 \quad \text{... (2.10)}$$

$$Det\ (\lambda I - C) = 0 \quad \text{... (2.11)}$$

- Setelah *eigenvector* dari C didapatkan dengan *eigenvalue* yang diambil bukan 0, *eigenvector* yang diambil diurutkan *eigenvalue*nya dari nilai yang terbesar ke nilai yang terkecil. *Eigenvevtor* dengan *eigenvalue* terbesar ini merupakan *principal component* yang memiliki varian terbesar pada seluruh citra.

- Untuk menghasilkan dimensi yang lebih kecil akan digunakan Rumus 2.12.

$$\widetilde{\emptyset_{j,i}} = V_{j,i}^{T} \emptyset_{j,i} \quad \text{... (2.12)}$$

Sehingga ruang vector yang dihasilkan akan berdimensi $(M\ x\ N)$, jauh lebih kecil daripada $(m\ x\ n)$.

2.4 K-Nearest Neighbor (K-NN)

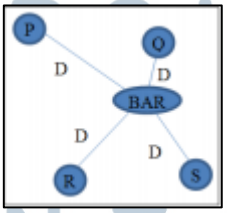
K-Nearest Neighbor (K-NN) merupakan sebuah metode yang digunakan mengklasifikasikan suatu objek berdasarkan data pembelajaran (*learning*) yang jaraknya paling dekat dengan objek tersebut. Data pembelajaran diproyeksikan ke dalam ruang berdimensi banyak, dimana masing-masing dari dimensi tersebut akan merepresentasikan fitur dari data (Liantoni & Febri, 2015).

Algoritma K-NN termasuk ke dalam metode yang menggunakan algoritma *supervised* dimana hasil dari data uji yang baru diklasifikan berdasarkan mayoritas

dari kategori pada K-NN (Sany dkk., 2017). Tujuan dari algoritma K-NN adalah untuk mengklasifikasi objek baru berdasarkan atribut dan *training samples*. Dimana hasil dari sampel uji yang baru diklasifikasikan berdasarkan mayoritas dari kategori pada KNN. Pada proses pengklasifikasian, algoritma ini tidak menggunakan model apapun untuk dicocokkan dan hanya berdasarkan pada memori (Liantoni & Febri, 2015).

Algoritma K-NN menggunakan klasifikasi ketetanggaan sebagai nilai prediksi dari contoh data uji yang baru. Jarak yang digunakan adalah jarak *Euclidean Distance*. Jarak *Euclidean* adalah jarak yang paling umum digunakan pada data *numeric*. Algoritma K-NN merupakan algoritma yang menentukan nilai jarak pada pengujian data *testing* dengan data *training* berdasarkan nilai terkecil dari nilai ketetanggaan terdekat (Liantoni & Febri, 2015).

Leidiyana (2013) memberikan gambaran kasus, misal diinginkan untuk mencari solusi terhadap masalah seorang pasien baru dengan menggunakan solusi dari pasien lama. Untuk mencari solusi dari pasien baru tersebut digunakan kedekatan dengan kasus pasien lama, solusi dari kasus lama yang memiliki kedekatan dengan kasus baru digunakan sebagai solusinya. Terdapat pasien baru dan 4 pasien lama, yaitu P, Q, R, dan S (Gambar 2.1). Ketika ada pasien baru maka yang diambil solusi adalah solusi dari kasus pasien lama yang memiliki kedekatan terbesar.



Gambar 2.1 Ilustrasi Kasus Algoritma K-NN

Misal D1 adalah jarak antara pasien baru dengan pasien P, D2 adalah jarak antara pasien baru dengan pasien Q, D3 adalah jarak antara pasien baru dengan pasien R, D4 adalah jarak antara pasien baru dengan pasien S. Dari ilustrasi gambar terlihat bahwa D2 yang paling terdekat dengan kasus baru. Dengan demikian maka solusi dari kasus pasien Q yang akan digunakan sebagai solusi dari pasien baru tersebut.

Ada banyak cara untuk mengukur jarak kedekatan antara data baru dengan data lama (data *training*), di antaranya *euclidean distance* dan *manhattan distance (city block distance)*, yang paling sering digunakan adalah *euclidean distance* (Bramer, 2007), seperti pada Rumus 2.13.

$$d = \sqrt{\sum_{i=1}^{p} (x_{2i} - x_{1i})^2} \quad \text{... (2.13)}$$

Dimana d = jarak *Euclidean, p* = dimensi data, x_1= data latih, dan x_2 = data uji.

Untuk mengukur jarak dari atribut yang mempunyai nilai besar, seperti atribut pendapatan, maka dilakukan normalisasi. Normalisasi bisa dilakukan dengan *min-max normalization* atau *Z-score standardization* (Larose, 2006). Jika data *training* terdiri dari atribut campuran antara numerik dan kategori, lebih baik gunakan *min-max normalization* (Larose, 2006). Untuk menghitung kemiripan kasus, digunakan Rumus 2.14 (Kusrini & Luthfi, 2009).

$$Similarity\ (p,q) = \frac{\sum_{i=1}^{n} f(p_i, q_i)\ X\ w_i}{w_i} \quad \text{... (2.14)}$$

Keterangan :

p = Kasus baru

q = Kasus yang ada dalam penyimpanan

n = Jumlah atribut dalam tiap kasus

i = Atribut individu antara 1 sampai dengan n

f = Fungsi *similarity* atribut i antara kasus p dan kasus q

w = Bobot yang diberikan pada atribut ke-i