BAB 2

TINJAUAN PUSTAKA

2.1 Tinjauan Non Statistik

Memuat tinjauan singkat dan jelas mengenai teori-teori, keterangan-keterangan atau segala sesuatu yang berkaitan dan mendukung atau menunjang masalah yang diajukan baik bersumber pada kepustakaan, jurnal, dan sumber akademis lainnya.

2.1.1 *Coronavirus Disease 2019 (COVID-19)*

Coronavirus merupakan keluarga besar virus yang menyebabkan penyakit pada manusia dan hewan. Pada manusia biasanya menyebabkan penyakit infeksi saluran pernapasan, mulai flu biasa hingga penyakit yang serius seperti Middle East Respiratory Syndrome (MERS) dan Sindrom Pernapasan Akut Berat/ Severe Acute Respiratory Syndrome (SARS). Penyakit ini menyebar melalui tetesan pernapasan dari batuk dan bersin (Kemendagri, 2020). Coronavirus merupakan virus RNA strain tunggal positif, berkapsul dan tidak bersegmen. Terdapat 4 struktur protein utama pada Coronavirus yaitu: protein N (nukleokapsid), glikoprotein M (membran), glikoprotein spike S (spike), protein E (selubung). Coronavirus tergolong ordo Nidovirales, keluarga Coronaviridae. Terdapat 4 genus yaitu alphacoronavirus, betacoronavirus, gammacoronavirus, dan deltacoronavirus. Sebelum adanya COVID-19, ada 6 jenis coronavirus yang dapat menginfeksi manusia, yaitu HCoV-229E (alphacoronavirus), HCoV-OC43 (betacoronavirus),

HCoVNL63 (alphacoronavirus) HCoV-HKU1 (betacoronavirus), SARS-CoV (betacoronavirus), dan MERS-CoV (betacoronavirus).

Coronavirus Disease 2019 (COVID-19) merupakan penyakit menular yang disebabkan oleh Coronavirus jenis baru. Corona virus jenis baru yang ditemukan pada manusia sejak kejadian luar biasa muncul di Wuhan Cina, pada Desember 2019. Coronavirus yang menjadi etiologi COVID-19 termasuk dalam genus betacoronavirus, umumnya berbentuk bundar dengan beberapa pleomorfik, dan berdiameter 60-140 nm. Hasil analisis filogenetik menunjukkan bahwa virus ini masuk dalam subgenus yang sama dengan coronavirus yang menyebabkan wabah SARS pada 2002-2004 silam, yaitu Sarbecovirus. Atas dasar ini, International Committee on Taxonomy of Viruses (ICTV) memberikan nama penyebab COVID-19 sebagai SARS-CoV-2.Virus ini berasal dari famili yang sama dengan virus penyebab SARS dan MERS. Meskipun berasal dari famili yang sama, namun SARS-CoV-2 lebih menular dibandingkan dengan SARS-CoV dan MERS-CoV (Kemenkes RI, 2020a).

2.2 Tinjauan Statistik

Memuat tinjauan singkat dan jelas mengenai metode-metode, keterangan-keterangan atau segala sesuatu yang berkaitan dengan metode statistika baik bersumber pada kepustakaan, jurnal, dan sumber akademis lainnya.

2.2.1 Analisis Deskriptif

Statistika deskriptif merupakan metode-metode yang berkaitan dengan pengumpulan dan penyajian suatu gugus data sehingga memberikan informasi yang berguna, tetapi tidak bermaksud menguji dan membuat kesimpulan yang lebih luas

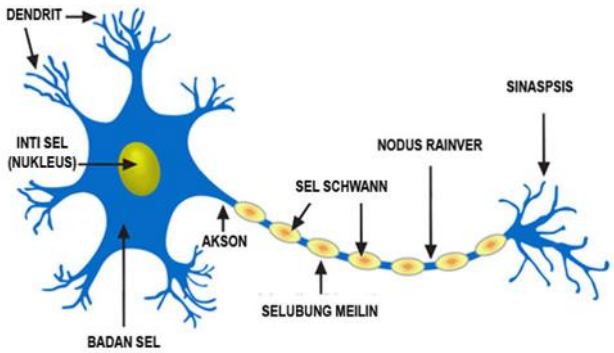
(Walpole, 1995). Analisis deskriptif yaitu bagian statistika mengenai pengumpulan data, penyajian, penentuan nilai-nilai statistika, pembuatan diagram atau gambar mengenai sesuatu hal (Nasution, 2017).

2.2.2 Prediksi

Prediksi adalah suatu proses memperkirakan secara sistematis tentang sesuatu yang paling mungkin terjadi di masa depan berdasarkan informasi masa lalu dan sekarang dimiliki, agar kesalahannya (selisih antara sesuatu yang terjadi dengan hasil perkiraan) dapat diperkecil. Prediksi tidak harus memberikan jawaban secara pasti kejadian yang akan terjadi, melainkan berusaha untuk mencari jawaban sedekat mungkin yang akan terjadi (Herdianto, 2013).

2.2.3 Jaringan Saraf Tiruan (JST)

Jaringan Saraf Tiruan (JST) atau *Artificial Neural Network* atau adalah sistem pemrosesan informasi yang memiliki karakteristik mirip dengan struktur jaringan saraf biologi, khususnya jaringan otak mausia. Jaringan saraf manusia terdiri atas sel-sel yang disebut *neuron*. Ada tiga komponen utama *neuron* yang fungsinya dapat dianalogikan yang terjadi pada JST, yaitu *dendrit, soma, dan akson*. *Dendrit* akan menerima sinyal-sinyal dari *neuron* lain. Sinyal tersebut merupakan impluls elektrik yang dikirim melalui *synaptic gap* melalui proses kimia. Sinyal tersebut dimodifikasi (diperkuat/diperlemah) di celah *synaptic*. Kemudian *soma* atau badan sel akan menjumlahkan sinyal-sinyal input yang masuk. Jika jumlahan tersebut cukup kuat dan melebihi batas ambang (*treshold*), maka sinyal tersebut akan diteruskan ke sel lain melalui *akson* dan *synaptic gap* (Fausett, 2004).



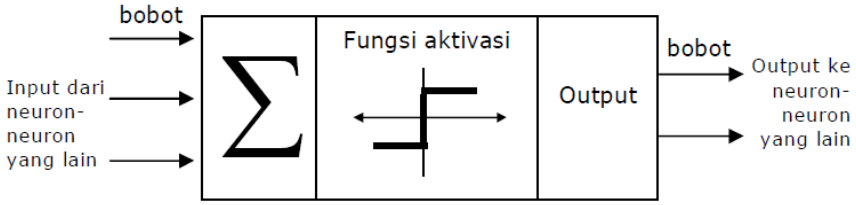
Gambar 0.1 Jaringan Saraf Manusia

Jaringan Saraf Tiruan juga didefinisikan sebagai sebuah prosesor yang terdistribusi paralel yang mempunyai kecenderungan untuk menyimpan pengetahuan yang didapatkan dari pengalaman dan membuatnya tetap tersedia untuk digunakan. Hal ini menyerupai kinerja otak asli dalam dua hal, yaitu: pengetahuan diperoleh melalui suatu proses belajar, kekuatan hubungan antar sel saraf dikenal dengan *synaptic* digunakan untuk menyimpan pengetahuan. Ada tiga komponen yang berperan penting dalam JST, yaitu (Fausett, 2004):

1. Pola hubungan antar neuron yang disebut artisektur.
2. Metode penentuan bobot penghubung yang disebut pelatihan (*Training*) atau pembelajaran (*learning*) atau algoritma.
3. Fungsi aktivasi yang dijalankan masing-masing neuron pada input jaringan untuk menentukan output.

Seperti halnya otak manusia, jaringan saraf tiruan juga terdiri dari beberapa neuron yang berhubungan untuk mentransformasikan informasi yang diterima melalui sambungan keluarnya, hubungan ini dikenal dengan nama bobot yang ditunjukkan dengan Gambar 2.2. Informasi tersebut disimpan pada suatu nilai tertentu pada bobot tersebut. Informasi *(input)* akan dikirim ke neuron dengan bobot

kedatangan. Input ini akan diproses oleh suatu fungsi perambatan yang akan menjumlahkan nilai-nilai semua bobot yang datang. Hasil penjumlahan ini kemudian akan dibandingkan dengan niai suatu ambang *(threshold)* tertentu melalui fungsi aktivasi setiap neuron.



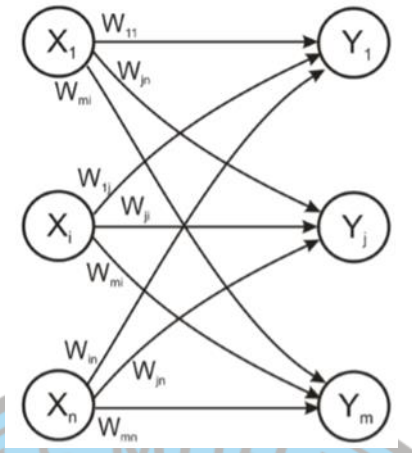
Gambar 0.2 Struktur Jaringan Saraf Tiruan.

2.2.3.1 Arsitektur Jaringan Saraf Tiruan

Di dalam jaringan saraf tiruan, neuron-neuron dikelompokkan dalam lapisan-lapisan (*layers*). Umumnya, neuron-neuron yang terletak pada lapisan yang sama akan memiliki keadaan yang sama. Faktor terpenting dalam menentukan kelakuan suatu neuron adalah fungsi aktivasi dan pola bobotnya. Pada setiap lapisan yang sama, neuron-neuron akan memiliki fungsi aktivasi yang sama. Ada beberapa arsitektur jaringan saraf, antara lain (Siang, 2005):

1. Jaringan Lapis Tunggal (*Single Layer*)

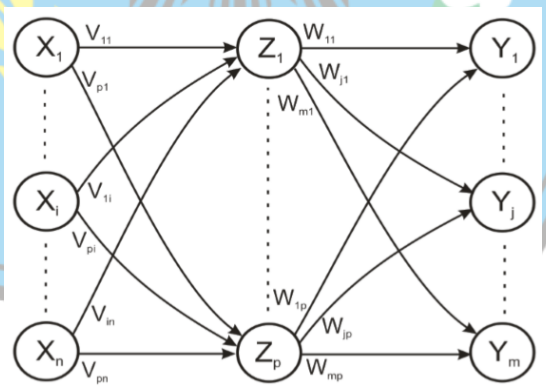
   Single layer net merupakan jaringan yang hanya mempunyai satu *layer* untuk menghubungkan nilai bobotnya. Neuron input langsung berhubungan dengan neuron output. Jaringan ini hanya menerima informasi dan langsung mengolahnya menjadi output tanpa melalui *hidden layer*. Ciri yang dimiliki *single layer net* yaitu hanya mempunyai satu *layer* input dan satu *layer* output.



Gambar 0.3 *Single Layer Net*

2. Jaringan Lapis Jamak (*Multilayer Net*)

Multilayer net adalah jaringan yang terdiri dari *input layer, output layer*, dan *hidden layer*. *Hidden layer* terletak diantara input dan *output layer*. Output dari *hidden layer* akan menjadi input bagi *layer* berikutnya. Jaringan ini paling tidak mempunyai satu *hidden layer*. *Multilayer net* bisa mengatasi masalah yang lebih rumit daripada *single layer net.*

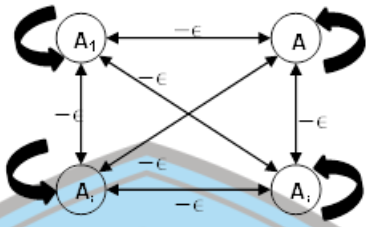


Gambar 0.4 *Multi Layer Net*

3. Jaringan *Reccurent*

Jaringan *reccurent* hampir sama dengan *single layer* maupun *multi layer*, hanya saja ada neuron output yang memberikan informasi pada unit input

(sering disebut *feedback loop*). Dalam jaringan ini, hubungan antar neuron tidak diperlihatkan pada diagram arsitektur (Haykin, 2005).



Gambar 0.5 Jaringan *Reccurent*

2.2.3.2 Pemrosesan Informasi pada Jaringan Saraf Tiruan

Secara garis besar jaringan saraf tiruan mempunyai dua tahap pemrosesan informasi, yaitu tahap pelatihan dan pengujian:

1. Pelatihan Jaringan (*Training*)

   Tahap pelatihan dimulai dengan pemasukan pola-pola pelatihan (data *Training*) ke dalam jaringan. Dengan menggunakan pola-pola ini jaringan akan mengubah bobot yang menjadi penghubung antar neuron. Selanjutnya, bobot ini menjadi dasar pengetahuan pada tahap pangujian (Warsito, 2009).

   Berdasarkan permasalahan yang ditangani, proses pelatihan pada *neural network* dapat dibedakan menjadi dua macam, yaitu:

   a) Pelatihan Terawasi (*Supervisied Learning*)

      Pada proses pelatihan terawasi, target yang diharapkan dari input yang diterima jaringan telah diketahui sebelumnya. Satu pola input akan diberikan ke satu neuron pada lapisan output. Pola ini akan dirambatkan di sepanjang jaringan saraf sehingga sampai ke neuron pada lapisan output.

Lapisan output ini akan membangkitkan pola output yang nantinya akan dicocokkan pada pola target. Apabila terjadi perbedaan antara pola output hasil pembelajaran dengan pola target, maka akan muncul *error*. Apabila nilai *error* masih cukup besar, mengindikasikan bahwa masih perlu dilakukan lebih banyak pembelajaran lagi. Tujuan dari pelatihan terawasi adalah untuk menghasilkan pemetaan dari vektor input ke nilai output, sehingga akan diperoleh output yang sesuai dengan target. Pengubahan nilai bobot dilakukan untuk mencapai tujuan ini.

b) Pelatihan Tak Terawasi (*Unsupervisied Learning*)

Metode pembelajaran ini diterapkan pada permasalahan-permasalahan dimana output jaringan tidak diketahui sebelumnya. Pada metode ini, pengubahan nilai bobot dilakukan dengan sendirinya tanpa menggunakan output yang diinginkan. Tujuan pembelajaran ini adalah mengelompokkan input yang serupa (Warsito, 2009).

2. Pengujian Jaringan (*Testing*)

Pada tahap ini dilakukan pengujian terhadap suatu pola masukan yang belum pernah dilatihkan sebelumnya (data *Testing*) mengunakan bobot-bobot yang telah dihasilkan pada tahap pelatihan. Diharapkan bobot-bobot hasil pelatihan yang sudah menghasilkan *error* minimal juga akan memberikan *error* yang kecil pada tahap pengujian.

2.2.3.3 Fungsi Aktivasi

Fungsi aktivasi merupakan suatu fungsi yang akan mentransformasikan suatu input menjadi suatu output tertentu. Pada JST, suatu informasi akan diterima

oleh input, kemudian input tersebut akan diproses melalui suatu fungsi perambatan (Amalina, 2016a). Argumen fungsi aktivasi adalah net masukan (kombinasi linear masukan dan bobotnya). Jika $net = \sum x_i w_i$, maka fungsi aktivasinya adalah:

$$f(net) = f(\sum x_i w_i) \quad (0.1)$$

Beberapa fungsi aktivasi yang sering digunakan adalah sebagai berikut (Siang, 2005):

a. Fungsi *Treshold* (batas ambang)

$$f(x) = \begin{cases} 1 & jika\ x \geq \alpha \\ 0 & jika\ x < \alpha \end{cases} \quad (0.2)$$

Untuk beberapa kasus, fungsi *treshold* yang dibuat tidak berharga 0 atau 1, tetapi berharga -1 atau 1 (sering disebut *treshold* bipolar).

Jadi:

$$f(x) = \begin{cases} 1 & jika\ x \geq \alpha \\ -1 & jika\ x < \alpha \end{cases} \quad (0.3)$$

b. Fungsi Sigmoid

Fungsi sigmoid, yang grafiknya berbentuk S, sejauh ini merupakan bentuk fungsi aktivasi yang paling umum dalam jaringan saraf. Fungsi ini didefinisikan sebagai peningkatan fungsi yang menunjukkan keseimbangan antara perilaku linear dan non-linear (Haykin, 2005).

$$f(x) = \frac{1}{1 + e^{-x}} \quad (0.4)$$

Fungsi sigmoid sering dipakai karena nilai fungsinya yang terletak antara 0 dan 1 serta dapat diturunkan dengan mudah.

$$f'(x) = f(x)(1 - f(x)) \quad (0.5)$$

c. Fungsi Identitas

Fungsi identitas disebut juga sebagai fungsi linear. Fungsi linear memiliki nilai output yang sama dengan nilai input (Amalina, 2016b) .

$$f(x) = x \qquad (0.6)$$

Fungsi identitas sering dipakai apabila kita menginginkan output jaringan berupa sembarang bilangan riil (bukan hanya pada range [0,1] atau [-1,1]).

2.2.3.4 Metode *Backpropagation*

Backpropagation merupakan metode pelatihan Jaringan Saraf Tiruan *multilayer* yang menggunakan prosedur pembelajaran terawasi (Saduf & Wani, 2013), dimana dilakukan penyesuaian bobot secara berulang untuk mendapatkan nilai *error* yang rendah. Dalam pemecahan masalah pada sistem, *Backpropagation* memiliki kelebihan yaitu bersifat adaptif (dapat menyesuaikan terhadap dataset) dan *fault tolerance* (kesalahan *error* kecil) (Azhar & Riksakomara, 2017). *Backpropagation* merupakan metode yang banyak digunakan untuk pengenalan pola-pola kompleks (Riedmiller & Braun, 1993). Hal ini terlihat dari kemampuannya dalam mengadaptasikan kondisi jaringan dengan data yang diberikan pada proses pembelajaran (Avan, 2017).

Backpropagation merupakan salah satu algoritma penentuan bobot. Algoritma *Backpropagation* biasa digunakan oleh *percepteron* dengan banyak lapisan untuk mengubah bobot-bobot yang terhubung dengan neuron yang ada pada *hidden layer*. Dasar perbaikan bobot pada algoritma *Backpropagation* dilakukan dengan cara mempertimbangkan kuadrat galat dari model. Perbaikan nilai bobot dalam *Backpropagation* dilakukan dalam arah mundur dengan memanfaatkan

turunan dari fungsi aktivasi yang dipakai pada jaringan saraf tiruan. Pembelajaran dilakukan dengan menggunakan *gradient descent*, dimana galat pada keluaran didorong kembali melalui propagasi balik untuk memperkirakan kesalahan pada *hidden layer* (Cowan, 2013 dalam (Avan, 2017)).

2.2.3.5 Arsitektur *Backpropagation*

Jaringan *backpropagation* memiliki beberapa neuron yang berada dalam satu atau lebih lapisan tersembunyi (*hidden layer)*. Setiap neuron yang berada dilapisan *input* terhubung dengan setiap neuron yang berada di *hidden layer*. Begitu juga pada *hidden layer*, setiap neuronnya terhubung dengan setiap neuron yang ada di *output layer*. Jaringan saraf tiruan *backpropagation* terdiri dari banyak lapisan (*multi layer*), yaitu:

a. Lapisan masukan (*input layer*)

   Input layer sebanyak 1 lapis yang terdiri dari neuron-neuron *input*, mulai dari neuron *input* pertama sampai neuron *input* ke-*n*. *Input layer* merupakan penghubung yang mana lingkungan luar memberikan sebuah pola ke dalam jaringan saraf. Sekali sebuah pola diberikan kedalam *input layer*, maka *output layer* akan memberikan pola yang lainnya. Pada intinya *input layer* akan merepresentasikan kondisi yang dilatihkan ke dalam jaringan. Setiap *input* akan merepresentasikan beberapa variabel bebas yang memiliki pengaruh terhadap *output layer*.

b. Lapisan tersembunyi (*hidden layer*)

   Hidden layer berjumlah minimal 1 lapis yang terdiri dari neuron-neuron tersembunyi mulai dari neuron tersembunyi pertama sampai neuron

tersembunyi ke-*p*. Menentukan jumlah neuron pada *hidden layer* merupakan bagian yang sangat penting dalam arsitektur jaringan saraf. Ada beberapa aturan metode berdasarkan pengalaman yang dapat digunakan untuk menentukan jumlah neuron yang akan digunakan pada *hidden layer.* Menurut Haykin (1999) jumlah *hidden* neuron 2 sampai dengan 9 sudah dapat menghasilkan hasil yang baik dalam jaringan, namun pada dasarnya jumlah hidden neuron yang digunakan dapat berjumlah sampai dengan tak berhingga (~). Sedangkan menurut Heaton (2008), ada beberapa aturan yang dapat digunakan untuk menentukan banyaknya jumlah neuron pada *hidden* layer yaitu:

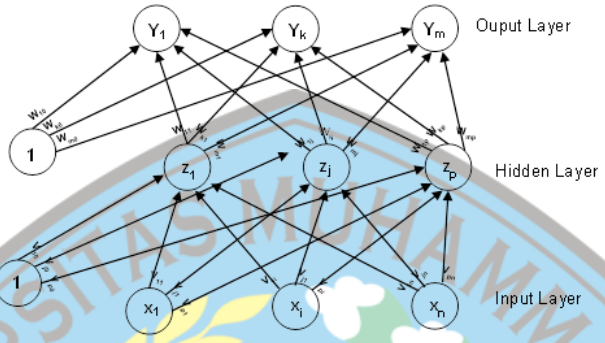
- Jumlah *hidden* neuron harus berada diantara ukuran *input layer* dan *output layer*.
- Jumlah *hidden* neuron harus 2/3 dari ukuran *input layer*, ditambah ukuran *output layer*.
- Jumlah *hidden neuron* harus kurang dari dua kali jumlah *input layer.*

Aturan-aturan tersebut hanya berupa pertimbangan dalam menentukan arsitektur jaringan saraf tiruan. Bagaimanapun, penentuan arsitektur jaringan akan kembali pada *trial and error* sesuai dengan masalah yang ditangani oleh jaringan.

c. Lapisan keluaran (*output layer*)

Output layer berjumlah satu lapis yang terdiri dari neuronneuron *output* mulai dari neuron *output* pertama sampai neuron *output* ke-*m*. *Output layer* dari jaringan saraf adalah pola yang sebenarnya diberikan oleh lingkungan

luarnya (*external environment*). Pola yang diberikan *output layer*dapat secara langsung ditelusuri kembali ke *input layer*nya. Jumlah dari neuron *output* tergantung dari tipe dan performa dari jaringan saraf itu sendiri.



Gambar 0.6 Arsitektur *Backpropagation*

2.2.3.6 Algoritma Pelatihan *Gradient Descent*

Pelatihan *Backpropagation* meliputi 3 fase. Fase pertama adalah fase maju. Pola masukan dihitung maju mulai dari layar masukan hingga layar keluaran menggunakan fungsi aktivasi yang ditentukan. Fase kedua adalah fase mundur. Selisih antara keluaran jaringan dengan target yang diinginkan merupakan kesalahan yang terjadi. Kesalahan tersebut dipropagasikan mundur, dimulai dari garis yang berhubungan langsung dengan unit-unit di layar keluaran. Fase ketiga adalah modifikasi bobot untuk menurunkan kesalahan yang terjadi. Adapun langkah-langkah secara lengkap algoritma *backpropagation* adalah sebagai berikut:

Langkah 0. Inisialisasi Bobot.

Langkah 1. Jika kondisi berhenti bernilai False lakukan langkah 2-9.

Langkah 2. Untuk setiap pasang data pelatihan (*Training*), lakukan langkah 3-8.

Feed Forward

Langkah 3. Tiap unit input $(x_i, i = 1,2,3, \dots, n)$ menerima sinyal input x dan sinyal tersebut disebarkan ke semua unit pada hidden layer.

Langkah 4. Tiap hidden layer $(z_j, j = 1,2,3, \dots, p)$ menjumlahkan bobot sinyal input,

$$z\_in_j = v_{0j} + \sum_{i=1}^{n} x_i v_{ij} \quad (0.7)$$

menggunakan fungsi aktivasi untuk menghitung sinyal output

$$z_j = f(z\_in_j) \quad (0.8)$$

dan mengirimkan sinyal ini ke setiap unit output.

Langkah 5. Tiap unit output $(y_k, k = 1,2,3, \dots, m)$ menjumlahkan bobot sinyal masuk,

$$y\_in_k = w_{0k} + \sum_{j=1}^{j} z_j w_{jk} \quad (0.9)$$

menggunakan fungsi aktivasi untuk menghitung sinyal output

$$y_k = f(y\_in_k) \quad (0.10)$$

Error dari backpropagation

Langkah 6. Tiap unit output $(y_k, k = 1,2,3, \dots, m)$ menerima pola yang sesuai dengan pola pelatihan input dan dihitung errornya,

$$\delta_k = (t_k - y_k) f'(y\_in_k) \quad (0.11)$$

menghitung bobot terkoreksi (digunakan untuk memperbarui w_{jk}),

$$\Delta w_{jk} = \alpha \delta_k z_j \quad (0.12)$$

menghitung bias terkoreksi (untuk memperbarui w_{jk})

$$\Delta w_{0k} = \alpha \delta_k \quad (0.13)$$

dan mengirim δ_k ke unit pada layer di bawahnya.

Langkah 7. Tiap unit hidden $(Z_j, j = 1,2,3, \dots, p)$ menjumlahkan delta input (dari unit layer di atas),

$$\delta\_in_j = \sum_{k=1}^{m} \delta_k w_{jk} \quad (0.14)$$

dikalikan dengan turunan fungsi aktivasi untuk menghitung error,

$$\delta_j = \delta\_in_j f'(z\_in_j) \quad (0.15)$$

menghitung koreksi bobot (digunakan untuk memperbarui w_{ij})

$$\Delta w_{ij} = \alpha \delta_j x_i \quad (0.16)$$

mengitung bias koreksi (untuk memperbarui w_{ij})

$$\Delta w_{ij} = \alpha \delta_j \quad (0.17)$$

Perbaharui bobot dan bias

Langkah 8. Tiap unit output $(y_k, k = 1,2,3, \dots, m)$ memperbaharui bobot dan bias $(j = 0, \dots, p)$:

$$w_{jk}(new) = w_{jk}(old) + \Delta w_{jk} \quad (0.18)$$

tiap unit hidden $(z, j = 1,2,3, \dots, p)$ memperbaharui bobot dan bias $(i = 0, \dots, n)$:

$$w_{ij}(new) = w_{ij}(old) + \Delta w_{ij} \quad (0.19)$$

Langkah 9. Tes kondisi berhenti.

2.2.3.7 *Resilient Backpropagation (Rprop) Neural Network*

Resilient Backpropagation (Rprop) merupakan salah satu modifikasi dari *Backpropagation* untuk mempercepat proses pembelajaran. Algoritma ini melakukan adaptasi langsung dari langkah bobot yang didasarkan pada informasi gradient lokal (Riedmiller & Braun, 1993) dari setiap iterasi pembelajaran, sehingga jumlah iterasi yang diperlukan untuk mencapai target yang diinginkan lebih sedikit (Sulistijanti, 2013). Algoritma ini tidak jauh berbeda dengan algoritma *Backpropagation* dan merupakan perbaikan dari algoritma *Backpropagation*. Keunggulan algoritma *Resilient Backpropagation* (Rprop) dibandingkan algoritma pendahulunya (*Backpropagation*) adalah proses pelatihannya lebih cepat dan tidak mengharuskan untuk menentukan nilai parameter apapun dalam perhitungannya. Sementara dalam algoritma *Backpropagation* memerlukan parameter rasio pembelajaran (*learning rate*), dan biasanya juga memerlukan momentum (McCaffery, 2015 dalam (Ervina & Silvi, 2018) .

Tujuan algoritma *Resilient Backpropagation* adalah untuk menghilangkan besarnya efek dari turunan parsial dengan hanya menggunakan tanda turunan dan mengabaikan besarnya nilai turunan. Tanda turunan ini akan menentukan arah perbaikan bobot-bobot.

2.2.3.8 Algoritma Pelatihan *Resilient Backpropagation*

Sama seperti algoritma *Gradient Descent*, algoritma *Resilient* melaksanakan dua tahap yaitu tahap pembelaaran *feedforward* untuk mendapatkan error dan tahap *backward* untuk mengubah nilai bobot.

Adapun langkah-langkah secara lengkap algoritma pembelajaran Resilient *Backpropagation* adalah sebagai berikut:

Langkah 0. Inisialisasi Bobot.

Langkah 1. Jika kondisi berhenti bernilai False lakukan langkah 2-9.

Langkah 2. Untuk setiap pasang data pelatihan (*Training*), lakukan langkah 3-8.

Feed Forward

Langkah 3. Tiap unit input $(x_i, i = 1,2,3, \dots, n)$ menerima sinyal input x dan sinyal tersebut disebarkan ke semua unit pada hidden layer.

Langkah 4. Tiap hidden layer $(z_j, j = 1,2,3, \dots, p)$ menjumlahkan bobot sinyal input,

$$z\_in_j = v_{0j} + \sum_{i=1}^{n} x_i v_{ij} \quad (0.20)$$

menggunakan fungsi aktivasi untuk menghitung sinyal output

$$z_j = f(z\_in_j) \quad (0.21)$$

dan mengirimkan sinyal ini ke setiap unit output.

Langkah 5. Tiap unit output $(y_k, k = 1,2,3, \dots, m)$ menjumlahkan bobot sinyal masuk,

$$y\_in_k = w_{0k} + \sum_{j=1}^{j} z_j w_{jk} \quad (0.22)$$

menggunakan fungsi aktivasi untuk menghitung sinyal output

$$y_k = f(y\_in_k) \quad (0.23)$$

Error dari backpropagation

Langkah 6. Tiap unit output $(y_k, k = 1,2,3, \dots, m)$ menerima pola yang sesuai dengan pola pelatihan input dan dihitung errornya,

$$\delta_k = (t_k - y_k) f'(y\_in_k) \quad (0.24)$$

menghitung bobot terkoreksi (digunakan untuk memperbarui w_{jk}),

$$\Delta w_{jk} = \alpha \delta_k z_j \quad (0.25)$$

menghitung bias terkoreksi (untuk memperbarui w_{jk})

$$\Delta w_{0k} = \alpha \delta_k \quad (0.26)$$

dan mengirim δ_k ke unit pada layer di bawahnya.

Langkah 7. Tiap unit hidden $(Z_j, j = 1,2,3, \dots, p)$ menjumlahkan delta input (dari unit layer di atas),

$$\delta\_in_j = \sum_{k=1}^{m} \delta_k w_{jk} \quad (0.27)$$

dikalikan dengan turunan fungsi aktivasi untuk menghitung error,

$$\delta_j = \delta\_in_j f'(z\_in_j) \quad (0.28)$$

menghitung koreksi bobot (digunakan untuk memperbarui w_{ij})

$$\Delta w_{ij} = \alpha \delta_j x_i \quad (0.29)$$

mengitung bias koreksi (untuk memperbarui w_{ij})

$$\Delta w_{ij} = \alpha \delta_j \quad (0.30)$$

Perbaharui bobot dan bias

Langkah 8. Tiap unit output $(y_k, k = 1,2,3, \dots, m)$ memperbaharui bobot dan bias $(j = 0, \dots, p)$:

$$w_{jk}(new) = w_{jk}(old) + \Delta w_{jk} \quad (0.31)$$

tiap unit hidden $(z, j = 1,2,3, \dots, p)$ memperbaharui bobot dan bias $(i = 0, \dots, n)$:

$$w_{ij}(new) = w_{ij}(old) + \Delta w_{ij} \quad (0.32)$$

Langkah 9. Tes kondisi berhenti.

2.2.4 Pengukuran Performa Prediksi

Hasil prediksi yang akurat adalah yang bisa meminimalkan kesalahan memprediksi. Karena itu dalam mengukur kinerja error JST pada pengerjaan tugas akhir ini digunakan metode perhitungan MSE (*Mean Square Error*). *Mean Square Error* adalah indikator ukuran secara keseluruhan untuk melihat hasil *running* dari pelatihan telah berhasil atau tidak (Prathama, 2018). Penghitungan dengan MSE adalah menjumlahkan kuadrat kesalahan kemudian dibagi dengan jumlah observasi. Berikut ini formulanya:

$$MSE = \frac{\sum_{t=1}^{n}(y_t - \hat{y}_t)^2}{n} \quad (0.33)$$

Dimana:

n: jumlah periode peramalan

$\hat{y}_t$: nilai target pada periode ke t

y_t: nilai output pada periode ke t