BAB II
DASAR TEORI

2.1 Penyakit Hati

2.1.1 Hepatitis C

Hepatitis C adalah salah satu penyakit yang dapat menyerang organ hati. Tahap penyakit hepatitis C di antaranya ialah inflamasi, fibrosis, sirosis, dan kanker hati. Hepatitis C yang dialami oleh usia tua biasanya lebih sulit untuk disembuhkan. Secara biologis wanita biasanya lebih mudah dalam proses penyembuhan bahkan cenderung sembuh dengan sendirinya karena memiliki sistem imun yang kuat. Timbulnya rasa lelah, bola mata atau kulit berwarna kekuningan, meriang atau demam, diare, dan mual merupakan beberapa gejala yang sering dialami oleh penderita penyakit hepatitis C. Pendeteksian penyakit hepatitis C biasanya dapat dilakukan dengan beberapa pemeriksaan seperti, pengambilan sampel darah untuk mengetahui jumlah sel darah merah, sel darah putih, hemoglobin, dan juga trombosit. Selain dari sampel tersebut, dilakukan juga pemeriksaan terhadap enzim-enzim yang terdapat dalam darah seperti, enzim *aspartate aminotransferase* (AST) dan *alanine aminotransferase* (ALT) dalam darah. Apabila terjadinya peningkatan enzim AST dan ALT dapat mengindikasikan adanya gangguan pada organ hati. Selain itu, untuk mengetahui seberapa banyaknya virus hepatitis C yang terdapat dalam tubuh seseorang dapat dilakukan pengukuran jumlah HCV RNA (Susanto and Nuri, 2022).

2.1.2 Sirosis Hati

Istilah sirosis hati berasal dari kata khirros yang berarti kuning oren *(orange yellow),* yang diberikan oleh Laence pada tahun 1819, dikarenakan nodul-nodul yang terbentuk terjadi perubahan warna. Suatu keadaan disorganisasi yang menyebar dari struktur hati yang semula normal kemudian nodul digantikan oleh jaringan baru sehingga mengalami *fibrosis* merupakan pengertian dari sirosis hati (Zebua *et al.*, 2012).

Penyebab utama kematian di negara maju salah satunya adalah sirosis hati yang biasa terjadi pada pasien yang berusia sekitar 45-46 tahun (setelah penyakit

kardiovaskuler dan kanker). Urutan ketujuh penyebab kematian di seluruh dunia ditempati oleh penyakit sirosis hati. Korban yang tidak dapat diselamatkan setiap tahunnya sekitar 25.000 jiwa akibat penyakit ini. Sirosis hati ditandai oleh gejala klinis yang sangat beragam, dimulai dari tanpa gejala hingga dengan gejala yang serius. Kasus sirosis hati pada pasien yang berobat ke rumah sakit kira-kira sekitar 30% dari seluruh populasi penyakit ini, dan sekitar 30% lainnya ditemukan secara tidak sengaja saat berobat untuk pemeriksaan penyakit lain, kemudian sisa lainnya ditemukan pada saat melakukan autopsi.

Kaum laki-laki lebih banyak ditemukan menderita penyakit sirosis hati daripada kaum wanita dengan perbandingan sekitar 1,6:1 dengan umur rata-rata paling banyak menderita penyakit ini yaitu kisaran umur 30-59 tahun dengan puncaknya sekitar 40 - 49 tahun. Penyakit ini biasanya sering disebabkan oleh hepatitis virus seperti, hepatitis B, C, *alcoholic liver/fatty liver, kolestasis*, obat, malnutrisi, dan faktor lain yang dapat menyebabkan kerusakan pada *liver*.

2.2 *Synthetic Minority Oversampling Technique* (SMOTE)

Ketidakseimbangan kelas dalam dataset kecil sangat merugikan dalam penelitian pada data *mining*, karena mesin pembelajaran mengalami kesulitan mengklasifikasikan kelas minoritas dengan benar. Sebuah kelas dianggap tidak seimbang jika rasio objek dalam kelas data lebih besar dari kelas lainnya. Kelas mayor adalah kelas yang memiliki data paling banyak, sedangkan kelas minor adalah kelas yang memiliki data lebih sedikit. Sebagian besar algoritma berasumsi bahwa distribusi kelas yang diuji sudah seimbang, sehingga menyebabkan klasifikasi nilai setiap kelas menjadi tidak tepat . Jika proporsi kategori minoritas dalam data kurang dari 35%, maka dikategorikan sebagai kategori tidak seimbang (Mustaqim *et al.*, 2019).

Synthetic Minority Oversampling Technique (SMOTE) merupakan suatu teknik yang digunakan untuk menyeimbangkan data yang tidak seimbang antar kelasnya. SMOTE bekerja dengan teknik *oversampling* yaitu, menduplikasi data dari kelas minor agar seimbang dengan data pada kelas mayor. Data pada kelas minoritas diperbanyak dengan menggunakan data sintetik yang berasal dari

replikasi data pada kelas minoritas itu sendiri dengan pendekatan yang bersifat ketetanggaan atau mencari tetangga terdekat. Teknik SMOTE digunakan karena memiliki kemampuan untuk mengurangi *overfitting* pada suatu data pelatihan. Algoritma yang berjalan pada SMOTE bekerja dengan mengambil perbedaan antara vektor fitur kelas minoritas dan nilai tetangga terdekat dari kelas minoritas dan mengalikan nilai tersebut dengan angka acak antara 0-1. Kemudian hasil perhitungan ditambahkan ke vektor fitur, sehingga diperoleh nilai vektor baru (Sutoyo and Fadlurrahman, 2020).

Persamaan (2.1) digunakan untuk membangkitkan data dengan metode SMOTE (Syukron *et al.*, 2020):

$$x_{syn} = x_i + (x_{knn} - x_i)\gamma \quad (2.1)$$

x_{syn} adalah hasil pengamatan pembangkitan data baru, x_i ialah pengamatan ke-i, x_{knn} ialah x paling dekat dari x_i, dan γ adalah bilangan acak antara 0 dan 1. Kemudian data nominal akan diisi dengan nilai mayoritas pada *k*-tetangga terdekat. Perhitungan jarak pada SMOTE dihitung dengan mengganti variabel kategori (jika ada) dengan kuadrat median standar deviasi variabel kontinyu kelas minoritas jika terjadi perbedaan antara nilai kategori pada pengamatan ke-i dan j (Syukron *et al.*, 2020).

2.3 Jaringan Saraf Tiruan (JST)

Istilah jaringan saraf tiruan atau yang dikenal dengan JST diartikan sebagai kecerdasan buatan otak manusia dikarenakan jaringan tersebut diimplementasikan dengan menggunakan suatu program pada komputer dengan kemampuan melakukan berbagai proses perhitungan (Br Sitepu, 2021).

JST menggambarkan suatu model matematis dan komputasi untuk melakukan fungsi aproksimasi non-linear, klasifikasi data *cluster* dan regresi non-parametrik (simulasi model saraf biologi) (Haryati *et al.*, 2016). Model matematis dari pemahaman manusia (*human cognition*) secara umum menghasilkan JST yang berlandaskan atas asumsi sebagai berikut:

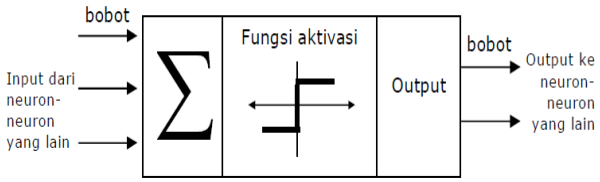
1. Elemen sederhana yang disebut neuron merupakan tempat terjadinya pemrosesan informasi.

2. Suatu isyarat yang bergerak di antara sel saraf (neuron) melalui suatu saluran penghubung.
3. Setiap saluran penghubung memiliki bobot yang selaras.
4. Setiap sel saraf merupakan fungsi aktivasi terhadap isyarat hasil penjumlahan yang memiliki bobot masuk untuk kemudian menentukan isyarat keluarannya (Jumarwanto *et al.*, 2009).

Setiap sel saraf (neuron) akan mempunyai satu inti sel, inti sel tersebut yang akan berperan dalam melakukan pemrosesan informasi. Informasi yang telah diproses sebelumnya diterima oleh dendrit. Keluaran dari suatu pemrosesan informasi tersebut ialah akson yang disertai oleh dendrit. Kemudian informasi hasil pemrosesan digunakan oleh neuron lain antar dendrit sebagai masukan yang dipertemukan dengan sinapsis. Rangsangan yang melintasi dendrit merupakan informasi yang dikirimkan oleh kedua neuron tersebut. Informasi yang telah diproses masuk dan diterima dijumlahkan oleh dendrit yang kemudian dikirim lewat akson ke ujung dendrit yang bergesekan dengan dendrit dari neuron yang lain. Jika informasi tersebut sudah memenuhi batasan tertentu maka akan diterima oleh neuron lain, yang biasa dikenal dengan nama ambang batas (*threshold*) yang sudah teraktivasi (Iqbal, 2018).

2.4 Komponen JST

JST sama halnya dengan jaringan saraf biologis manusia yang juga memiliki neuron. Gambar 2.1 menunjukkan struktur neuron pada jaringan saraf.



Gambar 2. 1 Struktur neuron jaringan saraf (Sumber : Iqbal, 2018)

Gambar 2.1 menunjukkan kesamaan dengan sel neuron biologis pada manusia. Neuron-neuron buatan tersebut mempunyai prinsip kerja yang sama dengan neuron biologis pada manusia. Masukan akan dikirim ke neuron dengan bobot tertentu yang datang. Kemudian masukan tersebut diproses oleh suatu fungsi perambatan

yang menjumlahkan semua nilai bobot *input*. Setelah itu, hasil akumulasi dibandingkan dengan fungsi aktivasi masing-masing neuron dengan ambang batas (*threshold*) tertentu. Neuron tersebut diaktifkan ketika masukan melebihi ambang batas tertentu. Ketika sebuah neuron aktif, neuron mengirimkan keluaran melalui bobot keluarannya ke semua neuron yang terhubung dengannya (Bisri *et al.*, 2013).

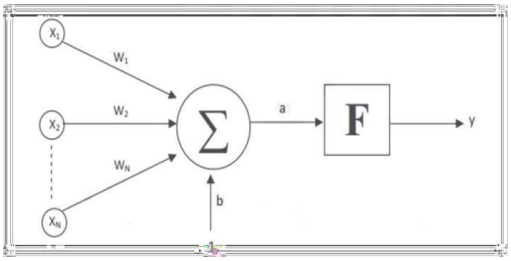
Neuron dikumpulkan dalam lapisan yang berlapis-lapis (*layer*) yang disebut dengan lapisan neuron (*neuron layer*) dalam JST. Biasanya, neuron dari sebuah lapisan terhubung ke lapisan depan dan belakang (kecuali lapisan masukan dan keluaran). Informasi dari lapisan masukan terhubung ke lapisan keluaran melalui lapisan tersembunyi (*hidden layer*). Algoritma pembelajaran mempengaruhi metode propagasi (Iqbal, 2018).

2.5 Arsitektur JST

Hubungan antar neuron dalam jaringan saraf mengikuti pola tertentu yang bergantung pada arsitektur jaringan saraf tersebut (Iqbal, 2018). Arsitektur JST memiliki tiga jenis lapisan yaitu:

a) Jaringan saraf dengan lapisan tunggal (*single layer net*)

Jaringan lapisan tunggal hanya memiliki satu lapisan yang terhubung ke bobot. Jaringan *single-layer* hanya menerima masukan kemudian memprosesnya langsung menjadi keluaran tanpa melalui *hidden layer* seperti yang ditunjukkan oleh Gambar 2.2.



Gambar 2. 2 Jaringan saraf tunggal (Sumber : Iqbal, 2018)

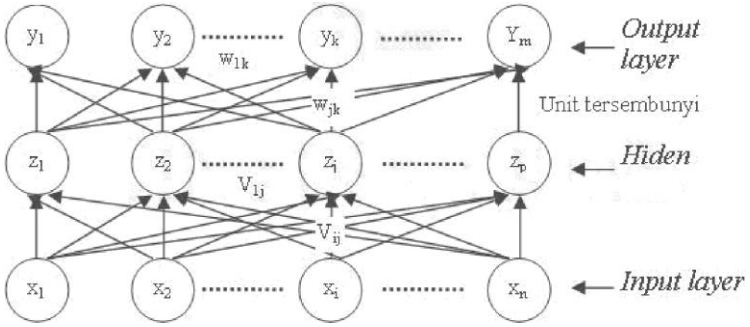
Gambar 2.2 menunjukkan sebuah neuron pada saat memproses *N input* (x_1, x_2,…, x_n) yang masing-masing memiliki bobot w_1, w_2,…, w_n. Fungsi aktivasi F dalam hal ini, mengaktifkan *y_in* sebagai keluaran jaringan *y*.

b) Jaringan saraf dengan banyak lapisan (*multilayer net*)

Jaringan banyak lapisan memiliki satu atau lebih lapisan yang terletak di antara lapisan masukan dan lapisan keluaran, atau disebut lapisan tersembunyi. Jaringan banyak lapisan ini memiliki keunggulan yang cukup baik dibanding lapisan tunggal yaitu dapat menyelesaikan masalah yang lebih kompleks. Algoritma propagasi balik menggunakan banyak dari lapisan ini. Dasar penentuan jumlah neuron pada lapisan tersembunyi metode propagasi balik adalah dengan persamaan (2.2) (Afrianty *et al.*, 2014).

$$l < m < 2l \tag{2.2}$$

l merupakan jumlah masukan, m merupakan jumlah lapisan tersembunyi, dan $2l$ merupakan 2 kali jumlah masukan. Arsitektur dari jaringan saraf lapisan banyak dapat dilihat pada Gambar 2.3.

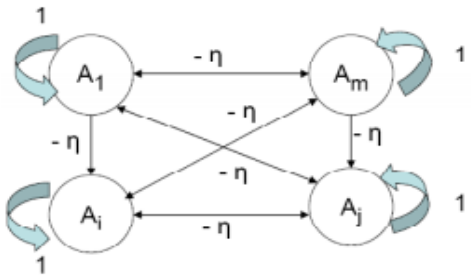


Gambar 2. 3 Arsitektur jaringan saraf dengan banyak lapisan (Sumber : Iqbal, 2018)

$x_1, x_2,\ldots, x_n$ merupakan lapisan *input*, $z_1, z_2,\ldots, z_p$ merupakan lapisan *hidden*, $y_1, y_2,\ldots, y_m$ merupakan lapisan *output*. v_{ij} merupakan bobot dari lapisan *input* ke lapisan *hidden*, sedangkan w_{jk} merupakan bobot dari lapisan *hidden* ke lapisan *output*.

c) Jaringan saraf dengan lapisan kompetitif (*competitive layer net*)

Arsitektur jaringan saraf dari lapisan kompetitif mempunyai bentuk yang berbeda, dengan neuron yang dapat terhubung antara satu sama lain. Algoritma *Learning Vector Quantization* (LVQ) menggunakan lapisan ini. Gambar 2.4 adalah contoh arsitektur jaringan lapisan yang kompetitif.



Gambar 2. 4 Jaringan saraf dengan lapisan kompetitif (Sumber : Iqbal, 2018)

2.6 *Hyperparameter*

Hyperparameter adalah parameter yang ditentukan sebelum melakukan proses pembelajaran oleh suatu algoritma dengan tetap konstan selama proses pelatihan maupun pengujian. *Hyperparameter* dilakukan dalam tahapan membangun suatu model mesin pembelajaran. Hal tersebut dilakukan untuk mendapatkan model dengan performa optimal atau hasil yang maksimal (Putra *et al.*, 2022).

2.6.1 Epoch

Epoch ialah *hyperparameter* yang menentukan pengulangan terhadap seluruh dataset dalam proses pelatihan jaringan. Ketika seluruh dataset sudah melalui proses pelatihan pada jaringan sampai kembali lagi ke awal maka hal tersebut merupakan pelatihan dalam satu *epoch*. Untuk mendapatkan hasil yang optimal diperlukan lebih dari satu *epoch*. Jumlah *epoch* ditentukan menyesuaikan dengan jumlah dan jenis dataset.

2.6.2 *Batch size*

Batch size ialah jumlah dari contoh yang dimasukkan dalam suatu jaringan sebelum penyesuaian bobot. Pada *batch* terakhir, hasil prediksi dibandingkan dengan keluaran untuk menghitung kesalahannya. Dari kesalahan ini, bobot diperbarui untuk memperbaiki model dengan menggunakan algoritma propagasi balik yang bergerak mundur dari lapisan terakhir menuju lapisan pertama. *Gradient descent* mempunyai 3 variasi yang berbeda berdasarkan *batch size* yang digunakan dalam proses memperbarui bobot yaitu:

a) *Batch Gradient Descent*

Batch gradient descent menggunakan *batch size* dengan ukuran yang sama dengan proses pelatihan. Sehingga proses memperbarui bobot dilakukan hanya sekali setelah keseluruhan proses propagasi maju selesai.

b) *Stochastic Gradient Descent*

Stochastic gradient descent ialah proses pembelajaran dengan setiap 1 data melakukan pembaruan.

c) *Mini-batch gradient descent*

Mini-batch gradient descent menggunakan *batch size* sebesar 2 pangkat m. Ukuran tersebut dipilih sesuai memori yang tersedia untuk mengoptimalkan memori yang digunakan.

2.6.3 *Activation Function*

Activation function atau fungsi aktivasi ialah fungsi yang terdapat dalam jaringan saraf yang mendefinisikan bagaimana jumlah bobot dari masukan diubah menjadi keluaran dari setiap lapisan jaringan saraf. Beberapa fungsi aktivasi yang sering digunakan, yaitu:

a) *Rectified Linear Activation* (ReLU)

Rectified Linear Activation (ReLU) ialah fungsi aktivasi yang sering digunakan pada lapisan tersembunyi karena mudah diterapkan dan efektif dalam mengatasi keterbatasan dari fungsi aktivasi populer lainnya, seperti sigmoid dan tanh. Fungsi aktivasi ReLU ialah lapisan aktivasi pada model yang menerapkan fungsi $(x) = max\ (0, x)$ dengan ReLU pada intinya hanya terbatas pada bilangan nol, yang artinya jika $x \leq 0$ maka $x = 0$ dan jika $x > 0$ maka $x = x$. Turunan dari fungsi aktivasi ReLU ditunjukkan oleh persamaan (2.3) (Putra *et al.*, 2022):

$$\frac{\partial y}{\partial x} = \begin{cases} 1, & x > 0 \\ 0, & x \leq 0 \end{cases} \tag{2.3}$$

b) *Logistic* (Sigmoid)

Logistic activation function atau disebut dengan fungsi sigmoid yang digunakan dalam algoritma klasifikasi *logistic regression*. Fungsi ini menggunakan nilai *real* atau nilai sebenarnya secara acak sebagai nilai masukan dan keluaran dalam rentang 0 hingga 1. Semakin besar nilai masukan atau semakin positif, maka

semakin dekat nilai keluaran dengan 1. Sebaliknya semakin kecil nilai masukan atau semakin negatif, maka semakin dekat nilai keluaran dengan 0. Fungsi sigmoid secara matematis dirumuskan oleh persamaan (2.4) (Putra *et al.*, 2022).

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.4}$$

e merupakan konstanta matematika, yang merupakan basis dari logaritma natural.

c) *Hyperbolic Tangent* (tanh)

Fungsi aktivasi *hyperbolic tangent* atau sering disebut dengan fungsi tanh, merupakan fungsi yang sangat mirip dengan fungsi sigmoid. Perbedaan fungsi ini yaitu menggunakan nilai masukan dan keluaran dalam rentang -1 hingga 1. Semakin besar nilai masukan atau semakin positif, maka semakin dekat nilai keluaran dengan 1 dan sebaliknya. Fungsi tanh secara matematis dirumuskan oleh persamaan (2.5) (Putra *et al.*, 2022).

$$\sigma(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.5}$$

e merupakan konstanta matematika, yang merupakan basis dari logaritma natural.

d) Linear

Fungsi aktivasi linear atau juga disebut dengan fungsi "identitas" (dikalikan dengan 1) atau "*no activation*" karena fungsi ini tidak mengubah jumlah bobot dari masukan dan mengembalikan nilai secara langsung. Fungsi linear secara matematis dirumuskan oleh persamaan (2.6) (Putra *et al.*, 2022).

$$L(X) = X \tag{2.6}$$

e) *Softmax*

Fungsi aktivasi *softmax* mengeluarkan nilai vektor berjumlah 1 yang dapat didefinisikan sebagai kemungkinan dari suatu keanggotaan kelas. Fungsi ini mirip dengan fungsi *arg max* yang mendefinisikan semua kelas dengan 0 dan kelas yang dipilih dengan 1. *Softmax* ialah versi dari fungsi *arg max* yang memungkinkan keluaran dari setiap kelas memiliki nilai kemungkinan yang apabila dijumlahkan akan berjumlah 1. Fungsi *softmax* secara matematis dirumuskan oleh persamaan (2.7) (Putra *et al.*, 2022).

$$s(x)_i = \frac{exp\ x_i}{\sum_{j=1}^{n} exp\ (x_j)} \quad (2.7)$$

x ialah vektor masukan, exp ialah fungsi eksponensial standar, xi ialah vektor masukan pada elemen ke-i, x_j ialah vektor masukan pada elemen ke-j yang akan dijumlahkan hasil perhitungan eksponensialnya hingga elemen ke-n, dan n ialah jumlah kelas. Fungsi *softmax* dapat bekerja dengan baik pada lapisan keluaran untuk tugas klasifikasi banyak kelas, karena dapat menghasilkan vektor dengan panjang yang sesuai dengan jumlah kelas dan dinormalisasi agar memiliki jumlah probabilitas atau kemungkinan sama dengan 1.

2.6.4 *Loss Function*

Loss function pada jaringan saraf berperan untuk menghitung kesalahan antara nilai prediksi yang dihasilkan oleh model mesin pembelajaran pada lapisan keluaran dengan nilai target. Dari kesalahan tersebut diperoleh gradien yang digunakan untuk memperbarui bobot dari setiap lapisan pada proses propagasi balik. *Cross-entropy* adalah *loss function* yang biasa digunakan untuk tugas klasifikasi. *Cross-entropy loss* juga disebut sebagai *logarithmic loss*, *log loss*, atau *logistic loss*. Nilai probabilitas atau kemungkinan yang diprediksi dari setiap kelas dibandingkan dengan kelas target yang diinginkan, yaitu 0 atau 1 untuk dihitung skor/*loss* yang menghitung probabilitas berdasarkan seberapa jauh dari nilai yang sebenarnya. Perhitungannya bersifat logaritmik yang menghasilkan perbedaan skor yaitu, skor besar untuk perbedaan besar yang mendekati 1 dan skor kecil untuk perbedaan kecil yang mendekati 0. *Cross-entropy* secara matematis dirumuskan dengan persamaan (2.8) (Putra *et al.*, 2022).

$$L = -\sum_{i=1}^{M} t_i\ log(p_i) \quad (2.8)$$

M adalah jumlah kelas, t_i merupakan nilai aktual dari kelas ke-i, dan p_i merupakan nilai probabilitas hasil prediksi dari kelas ke-i. Perhitungan *cross-entropy* memiliki 2 metode yang berbeda untuk masing-masing permasalahan *binary classification* dan *multiclass classification*, yaitu:

a) *Binary Cross-Entropy*

Binary cross-entropy ialah fungsi kesalahan yang digunakan dalam tugas *binary classification*. Fungsi ini hanya menjawab pertanyaan dengan dua pilihan yaitu, ya atau tidak, A atau B, 0 atau 1, kiri atau kanan, dan sebagainya. Jika jumlah *M* = 2, maka *binary cross-entropy* secara matematis dirumuskan oleh persamaan (2.9) (Putra *et al.*, 2022).

$$L = -\sum_{i=1}^{2} t_i \, log(p_i) \tag{2.9}$$

t_i merupakan nilai target dari kelas ke-i dan p_i merupakan nilai probabilitas sigmoid hasil prediksi dari kelas ke-i. Satu-satunya fungsi aktivasi yang cocok dengan *binary cross-entropy loss function* adalah fungsi sigmoid, karena fungsi kesalahan ini perlu menghitung logaritma dari p dan (1 – p) yang hanya ada jika p bernilai antara 0 dan 1.

b) *Categorical Cross-Entropy*

Categorical cross entropy ialah fungsi kesalahan yang digunakan dalam tugas klasifikasi banyak kelas. Fungsi ini dirancang untuk mengukur perbedaan antara 2 distribusi probabilitas. Apabila *M*>2 (*multi class classification*) maka *loss* untuk setiap kelas dihitung secara terpisah dan hasilnya dijumlahkan. *Categorical cross-entropy* secara matematis dirumuskan oleh persamaan (2.10) (Putra *et al.*, 2022).

$$L = -\sum_{i=1}^{M} t_i \, log(p_i) \tag{2.10}$$

M ialah jumlah kelas, t_i ialah nilai aktual dari kelas ke-i, dan p_i ialah nilai probabilitas *softmax* hasil prediksi dari kelas ke-i. Satu-satunya fungsi aktivasi yang disarankan untuk digunakan dengan fungsi kesalahan *categorical cross-entropy* adalah fungsi *softmax*.

c) *Sparse Categorical Cross-Entropy* (SCC)

Sparse Categorical Cross-Entropy (SCC) merupakan salah satu fungsi kesalahan yang sering digunakan dalam proses klasifikasi yang memiliki kategori banyak kelas. Fungsi ini dapat digunakan untuk klasifikasi menggunakan fungsi aktivasi *softmax* dengan satu hasil yang benar. Distribusi dari probabilitas prediksi dengan distribusi dari kelas target yang benar dibandingkan dengan menggunakan

fungsi kesalahan SCC. Secara matematis SCC menggunakan persamaan (2.11) (Khamidatullailiyah, 2022):

$$Loss = -\sum_{i=1}^{output\ size} y_i \,.\, log\ \hat{y}_i \qquad (2.11)$$

$\hat{y}_i$ merupakan nilai skalar ke-i pada prediksi keluaran dan y_i merupakan nilai target yang sesuai.

2.6.5 *Root Mean Square Propagation* (RMSProp)

Optimizer digunakan untuk meningkatkan nilai akurasi. Bobot model selama proses pelatihan disesuaikan dengan menggunakan o*ptimizer*. Bentuk model yang akurat diperoleh dari *optimizer* dengan penyesuaian bobot yang didasarkan pada nilai kesalahan. *Root Mean Square Propagation* (RMSProp) merupakan salah satu algoritma optimasi. Cara kerja RMSProp yaitu dengan mempertahankan rata-rata bergerak gradien kuadrat untuk setiap bobot (Setiawan *et al.*, 2022).

2.7 Propagasi Balik (*Backpropagation*)

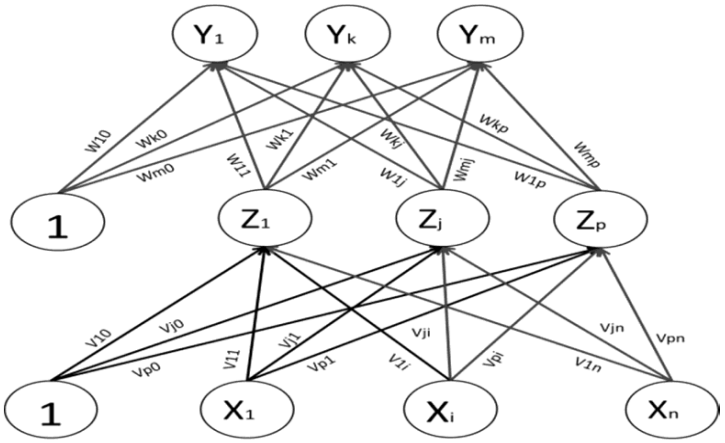
Propagasi balik (*backpropagation*) adalah salah satu metode yang sangat baik digunakan dalam menyelesaikan masalah pengenalan pola yang kompleks. Propagasi balik memiliki lapisan *input*, lapisan tersembunyi, dan lapisan keluaran dengan setiap objek yang saling terhubung. Jaringan ini terdiri dari beberapa lapisan (*multi-layer network*). Ketika jaringan diinisialisasi dengan pola pelatihan yang diberikan pada pola *input*, pola tersebut diumpankan ke unit lapisan tersembunyi, yang kemudian menuju ke unit lapisan terakhir, yaitu lapisan keluaran. Selain itu, unit dari lapisan keluaran sesuai dengan keluaran dalam format JST. Jika hasil keluaran tidak sesuai dengan yang diharapkan, maka keluaran kembali ke lapisan tersembunyi kemudian dari lapisan tersembunyi ke lapisan masukan (Maharani *et al.*, 2013).

2.7.1 Arsitektur Propagasi Balik (*Backpropagation*)

Setiap unit lapisan masukan dalam jaringan propagasi balik selalu terhubung dengan setiap unit lapisan tersembunyi, seperti halnya setiap unit lapisan tersembunyi selalu terhubung dengan unit lapisan keluaran (Maharani *et al.*, 2013). Jaringan propagasi balik terdiri dari beberapa lapisan (*multi-layer network*), yaitu:

1. Pada lapisan masukan (1 buah), yang terdiri dari 1 hingga *n* unit masukan.
2. Pada lapisan tersembunyi (minimal 1 buah), yang terdiri dari 1 hingga *p* unit tersembunyi.
3. Pada lapisan keluaran (1 buah), yang terdiri dari 1 hingga m unit keluaran.

Arsitektur propagasi balik ditunjukkan oleh Gambar 2.5.



Gambar 2. 5 Arsitektur propagasi balik (Sumber : Aditya *et al.*, 2014)

2.7.2 Pelatihan Jaringan Propagasi Balik

Pelatihan jaringan propagasi balik terdiri dari 2 tahapan, *feedforward* (umpan maju) dan *backward propagation* (umpan mundur). Jaringan memiliki sekumpulan contoh pelatihan yang disebut set pelatihan. Vektor fitur yang diasosiasikan dengan keluaran yang akan dilatih, disebut vektor masukan, menggambarkan set pelatihan. Dengan kata lain, set pelatihan terdiri dari vektor masukan dan vektor target keluaran. Keluaran dari jaringan adalah vektor keluaran aktual. Selain itu, perbandingan dilakukan dengan mengurangkan keluaran aktual yang dihasilkan dari keluaran target. Hasil pengurangan adalah kesalahan. Kesalahan tersebut menjadi dasar untuk melakukan perubahan pada bobot yang sudah ada kemudian dilakukan propagasi kembali (Maharani *et al.*, 2013).

Error akan berkurang setiap terjadi perubahan bobot. Pada setiap set pelatihan terjadi perubahan bobot sampai kondisi berhenti tercapai, suatu pelatihan akan tercapai apabila jumlah *epoch* yang diinginkan sudah diperoleh atau hingga suatu nilai ambang yang sudah ditetapkan dapat terlampaui. Berikut beberapa tahapan algoritma pelatihan jaringan propagasi balik:

1. Tahap umpan maju (*feedforward*)
2. Tahap umpan mundur (*backward propagation*)
3. Tahap memperbaharui bobot dan bias.

Secara rinci algoritma pelatihan jaringan propagasi balik dapat diuraikan sebagai berikut (Bisri *et al.*, 2013):

1. Inisialisasi bobot, konstanta laju pelatihan (a), toleransi kesalahan atau nilai bobot (jika menggunakan nilai bobot sebagai kondisi berhenti) atau atur periode maksimum *epoch* (jika menggunakan jumlah *epoch* sebagai kondisi berhenti).
2. Sampai kondisi terpenuhi, lakukan langkah selanjutnya. Pasangan dari setiap elemen yang akan dipelajari:

   Umpan maju:

a. Setiap unit *input* (x) (X_i, i = 1, 2, 3,…,n) menerima sinyal dan meneruskan sinyal ke semua unit lapisan tersembunyi.

b. Setiap unit lapisan tersembunyi (z) (Z_j, j = 1, 2, 3,…,p) menjumlahkan sinyal masukan yang mempunyai bobot:

$$z_{inj_j} = V_{0j} + \sum_{i=1}^{n} X_i V_{ij} \tag{2.12}$$

Untuk menghitung sinyal keluaran, perlu menggunakan fungsi aktivasi sigmoid biner:

$$z_j = f(z\_in_j)$$
$$z_j = \frac{1}{1 + e^{-z\_in_j}} \tag{2.13}$$

Hasil yang diperoleh dari sinyal keluaran (z_j) kemudian dikirimkan ke semua unit-unit keluaran.

c. Sinyal masukan terbobot (y_in_k) dijumlahkan oleh setiap unit keluaran (y) (y_k, k = 1, 2, 3,….,m):

$$y\_in_k = w_{0k} + \sum_{i=1}^{n} z_i\, w_{ij} \tag{2.14}$$

Menggunakan fungsi aktivasi sigmoid biner untuk menghitung sinyal *output* yang teraktivasi (yk):

$$y_k = f(y_{in_k})$$
$$y_k = \frac{1}{1 + e^{-y\_in_k}} \quad (2.15)$$

Kirimkan sinyal tersebut ke semua unit di lapisan unit-unit keluaran.

Umpan mundur:

d. Setiap unit keluaran (Y) ((Y) k, $k = 1, 2, 3,\ldots,m$) menerima pola target yang terkait dengan pola masukan yang dipelajari, untuk menghitung informasi kesalahan digunakan persamaan (2.19):

$$\delta_k = (t_k - y_k) f'(y\_in_k) \quad (2.16)$$

Kemudian hitung kembali koreksi bobot (untuk memperbaiki nilai w_{jk}):

$$\Delta w_{jk} = \alpha \delta_k z_j \quad (2.17)$$

Menghitung nilai bias (untuk memperbaiki nilai w_{0k}):

$$\Delta w_{0k} = \alpha \delta_k \quad (2.18)$$

Kemudian δk dikirimkan ke unit-unit yang berada di lapisan bawahnya.

e. Setiap unit tersembunyi (z_j, j = 1, 2, 3,….,p) menjumlahkan delta masukan dari unit-unit lapisan diatasnya:

$$\delta\_in_j = \sum_{k=1}^{m} \delta_k w_{jk} \quad (2.19)$$

Nilai tersebut kemudian dikalikan dengan turunan dari fungsi aktivasi untuk menghitung *error*:

$$\delta_j = \delta\_in_j f'(z\_in_j) \quad (2.20)$$

Kemudian hitung koreksi bobot yang digunakan untuk memperbaiki nilai v_{ij}:

$$\Delta v_{jk} = \alpha \delta_j xi \quad (2.21)$$

Hitung juga koreksi bobot bias yang akan digunakan untuk memperbaiki nilai v_{0j}:

$$\Delta v_{0j} = \alpha \delta_j \quad (2.22)$$

Setiap unit keluaran (y_k, $k = 1, 2, 3,\ldots,m$) memperbaiki bias dan bobotnya (j = 0, 1, 2,…,p):

$$w_{jk}(baru) = w_{jk}(lama) + \Delta w_{jk} \quad (2.23)$$

Tiap-tiap unit tersembunyi (y_k, $k = 1, 2, 3,\ldots,p$) memperbaiki bias dan bobotnya (i=0,1,2,3,…,n):

$$v_{ij}(baru) = v_{ij}(lama) + \Delta v_{ij} \quad (2.24)$$

3. Tes kondisi berhenti.

2.8 *Confusion Matrix*

Confusion matrix adalah sebuah pengukuran performa yang biasa digunakan pada masalah klasifikasi atau prediksi dengan keluaran yang terdiri dari dua kelas atau lebih. Terdapat empat atribut yang merupakan kombinasi dari nilai yang diprediksi (*predicted*) dan nilai yang sebenarnya (aktual), yaitu (Putra *et al.*, 2022):

1. *True Positive*: Jumlah data yang bernilai positif untuk kategori prediksi dan sebenarnya.
2. *False Positive*: Jumlah data yang bernilai positif hanya pada kategori prediksi dan bernilai negatif pada kategori sebenarnya.
3. *True Negative*: Jumlah data yang bernilai negatif untuk kategori prediksi dan sebenarnya.
4. *False Negative*: Jumlah data yang bernilai negatif hanya pada kategori prediksi dan bernilai positif pada kategori sebenarnya.

Keempat atribut tersebut akan menjadi dasar perhitungan beberapa metrik evaluasi, yaitu:

2.8.1 Akurasi

Akurasi adalah rasio prediksi benar (positif dan negatif) dengan keseluruhan data. Metrik ini paling sering digunakan karena mudah dihitung dan digunakan. Kelemahan metrik ini adalah kurang akurat untuk perhitungan pada data yang tidak seimbang. Nilai akurasi dapat diperoleh dengan persamaan (2.25) (Putra *et al.*, 2022).

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (2.25)$$

2.8.2 *Precision*

Precision adalah rasio antara *True Positive* (TP) dengan keseluruhan data yang diprediksi positif. *Precision* digunakan untuk memperkecil terjadinya *False Positive* (FP). Nilai *precision* dapat dihitung dengan persamaan (2.26) (Putra *et al.*, 2022).

$$precision = \frac{TP}{TP + FP} \tag{2.26}$$

2.8.3 *Recall*

Recall adalah rasio antara *True Positive* (TP) dengan keseluruhan data yang sebenarnya bernilai positif. *Recall* digunakan untuk memperkecil terjadinya *False Negative* (FN). Nilai *recall* dapat dihitung dengan persamaan (2.27) (Putra *et al.*, 2022).

$$recall = \frac{TP}{TP + FN} \tag{2.27}$$

2.8.4 *F1-Score*

F1-score adalah rata-rata harmonik dari *precision* dan *recall*. Nilai *F1-score* dapat dihitung dengan persamaan (2.28) (Putra *et al.*, 2022).

$$F1\ score = 2 \times \frac{precision \times recall}{precision + recall} = \frac{2TP}{2TP + FP + FN} \tag{2.28}$$