BAB II

TINJAUAN PUSTAKA

2.1. Minyak Mentah

Minyak disebut bahan bakar fosil karena terbuat dari fosil purba. Minyak yang kita gunakan saat ini berasal dari tumbuhan dan hewan berbasis air yang hidup bahkan sebelum dinosaurus ada. Minyak adalah bahan bakar fosil yang terdiri dari hidrokarbon yang ditinggalkan oleh hewan dan tumbuhan jutaan tahun yang lalu (Walters, 2007). Selama jutaan tahun, sisa-sisa tumbuhan dan hewan ini dihancurkan dan dipanaskan di antara lapisan batu dan pasir hingga menjadi genangan minyak di bawah tanah. Kita menyebut minyak ini "minyak mentah" karena belum dimurnikan untuk digunakan. Minyak mentah adalah bahan bakar dan bahan dasar untuk produk. Minyak dapat menjadi bahan bakar jet, mobil, pemanas, dan generator (Mawad, 2020).

Minyak mentah dapat digunakan untuk membuat produk minyak bumi seperti tar, aspal, lilin parafin, dan minyak pelumas. Bensin, yang sangat diperlukan dalam kehidupan kita, terbuat dari minyak mentah yang sudah diolah dengan baik. Minyak mentah juga dapat digunakan untuk membuat produk yang biasanya tidak terkait dengan minyak bumi. Parfum, pupuk, dan komputer semuanya dibuat dengan minyak mentah. Ini adalah dasar untuk plastik, jadi apa pun dengan plastik dibuat dengan minyak mentah. Biasanya warna pada minyak mentah selain hitam atau coklat tua, ada warna yang lain juga seperti kemerahan, kekuningan bahkan kehijauan. Biasanya hal ini menunjukkan kompisisi warna kimia dari stok minyak mentah yang berbeda (Manescu, 2014).

2.2. Minyak Mentah *Brent* (*Brent Crude Oil*)

Brent adalah nama yang diberikan untuk minyak mentah yang relatif ringan yang terbuat dari campuran minyak mentah dari 19 ladang minyak di Laut Utara. Minyak Mentah *Brent* adalah salah satu dari tiga tolok ukur utama harga minyak mentah per barel, bersama dengan *West Texas Intermediate* (*WTI)* dari Amerika Utara dan Dubai Crude dari Teluk Persia. *Brent* juga merupakan nama ladang

minyak yang terletak di Laut Utara di lepas pantai Skotlandia, yang ditemukan pada tahun 1971 dan mulai berproduksi pada tahun 1976. *Brent* adalah akronim untuk *Broom*, *Rannoch*, *Etive*, *Ness* dan *Tarbert* – lima formasi geologi yang membentuk medan Jurassic Tengah (Manescu, 2014).

Minyak mentah *Brent* adalah salah satu dari tiga tolak ukur minyak utama yang digunakan oleh kontrak minyak perdagangan, berjangka, dan turunannya. Campuran *Brent* tidak diperdagangkan secara langsung tetapi berjangka, *Brent* diperdagangkan di *Intercontinental Exchange* (ICE) serta NYMEX, dengan tanggal pengiriman setiap harinya selama 12 bulan dalam setahun (Imsirovic, 2021).

Minyak mentah *Brent* dan *West Texas Intermediate* (WTI) mengandung kandungan sulfur dan gravitasi API yang berbeda, yang berkontribusi pada perbedaan harga minyak. *Brent Crude* adalah minyak mentah ringan, dan mengandung 0,37% belerang dibandingkan dengan 0,24% kandungan belerang di *West Texas Intermediate*. Oleh karena itu minyak mentah *Brent* lebih banyak digunakan untuk solar. *Brent Crude* diekstraksi dari Laut Utara dan terdiri dari *Brent Blend, Forties Blend, Oseberg*, dan minyak mentah *Ekofisk* (juga dikenal sebagai Kutipan BFOE). Karena *Brent Crude* diproduksi di dekat laut, sehingga biaya transportasi jauh lebih rendah (Manescu, 2014).

2.3. Analisis Deskriptif

Analisis deskriptif biasanya bertujuan untuk memberikan informasi dengan menggunakan banyak metode penyampaian yang sederhana dan mudah dimengerti seperti menampilkan tabel, grafik atau diagram dalam penyajian informasinya. Pengertian lain tentang analisis deskripsi menjelaskan bahwa tidak hanya menampilkan diagram, tabel maupun grafik akan tetapi juga menginterpretasikab data historis untuk lebih memahami perubahan yang terjadi dalam data. Analisis deskriptif juga menjelaskan penggunaan berbagai data pada masa lampau untuk melihat perbandingan. Selain menggunakan tabel, grafik maupun diagram, biasanya analisis data menyertakan beberapa uji statistik untuk menjelaskan gambaran data agar orang lain mudah memahami (Purnamasari, 2018).

2.4. Peramalan

Peramalan adalah memprediksi masa depan seakurat mungkin mengingat semua informasi yang tersedia, termasuk data historis dan pengetahuan tentang peristiwa masa depan yang dapat mempengaruhi ramalan (pakaja, dkk, 2012). Peramalan bukan hanya sekedar perkiraan namun untuk menjadi perkiraan yang diinginkan maka diperlukan teknik-teknik tertentu.

Peramalan harus menjadi bagian yang terstruktur dari kegiatan pengambilan keputusan manajemen karena dapat memainkan peran penting dalam banyak hal. Subagyo (2002) menjelaskan bahwa peramalan bertujuan untuk menghasilkan nilai yang dapat meminimalkan nilai *error* yang biasanya diukur dengan nilai MSE (*Mean Square Error*), MAPE (*Mean Absolute Percentage Error*) maupun MAE (*Mean Absolute Error*). Sedangkan menurut Herjanto (2010) peramalan dibagi menjadi tiga jenis yaitu peramalan jangka pendek, jangka menengah dan jangka panjang.

2.5. Analisis *Time Series*

Tahun 1970 *Time Series* model dipublikasikan eleh George E.P. Box dan Gwilyn M. Jenkins dimana dalam buku karya mereka yang berjudul *Time Series Analysis : forecasting and control* menjelaskan bahwa analisis *Time Series* merupakan urutan titik data yang terjadi selama beberapa periode waktu atau biasa dikenal dengan sebutan deret berkala.

Analisis *Time Series* bertujuan untuk meramalkan suatu data atau kondisi untuk masa yang akan datang, melihat hubungan antar peubah dan mengetahui apakah prosesnya terkendali atau tidak. Jenis-jenis pola data juga menjadi hal yang perlu diperhatikan agar mendapatkan metode yang tepat. Data *Time Series* merupakan pola data yang ditentukan dengan peramalan kuantitatif dimana data tersebut didapatkan dari masa lalu yang berdasarkan waktu. (Sumihi, 2017)

2.5.1. *Autocorrelation Function* (ACF)

Fungsi Autokorelasi (ACF) didefinisikan bagaimana titik data dalam deret waktu terkait rata-rata, dengan titik data sebelumnya (Box, Jenkins, dkk, 1994). Dengan kata lain, ini mengukur kesamaan diri dari sinyal pada waktu tunda yang berbeda. Dengan demikian, ACF adalah fungsi dari penundaan/lag yang menentukan pergesaran waktu yang diambil dari masa lalu untuk memperkirakan kesamaan antara titik data (Bhavik, Nounou, 2000). ACF juga menjelaskan bagaimana korelasi antara 2 nilai sinyal berubah seiring perubahan jarak. Ini adalah ukuran domain waktu dari memori proses stokastik. Dan tidak memberikan informasi tentang konten frekuensi proses. Secara umum, untuk hal-hal seperti *error signal*, ACF didefinisikan sebagai berikut (Wei, 2006).

$$\gamma_k = Cov(X_t, X_{t+k}) = E(X_t - \mu)(X_{t+k} - \mu), \tag{2.1}$$

dan korelasi antara X_t dan X_{t+k} adalah :

$$\rho_k = \frac{Cov(X_t, X_{t+k})}{\sqrt{Var(X_t)}\sqrt{Var(X_{t+k})}} = \frac{\gamma_k}{\gamma_0} \tag{2.2}$$

Dengan $Var(X_t) = Var(X_{t+k}) = \gamma_0$.

2.5.2. *Partial Autocorrelation Function* (PACF)

Fungsi autokorealasi parsial (PACF) secara umum merupakan korelasi bersyarat. Hal ini merupakan korelasi antara 2 variabel dengan asumsi bahwa nilai dari beberapa set variabel lainnya diketahui dan diperhitungkan. Makridakis (1999) menjelaskan bahwa autokorelasi parsial dipergunakan untuk mengukur taraf kekuatan/keeratan antara Z_t dan Z_{t+k} jika dampak dari lag waktu $1, 2, 3, \ldots k$ diklaim terpisah.

Untuk formula dari fungsi autokorelasi parsial adalah sebagai berikut (Wei, 2006).

$$\rho_k = \frac{Cov\left[(Z_t - \hat{Z}_t), (Z_{t+k} - \hat{Z}_{t+k})\right]}{\sqrt{Var((Z_t - \hat{Z}_t)}\sqrt{Var(Z_{t+k} - \hat{Z}_{t+k})}} \tag{2.3}$$

Sampel dari PACF biasa dinotasikan dengan $\emptyset_{kk}$ dan dengan perhitungan. (Wei, 2006).

$$\phi_{kk} = \frac{\begin{vmatrix} 1 & \rho_1 & \rho_2 & \cdots & \rho_{k-2} & \rho_1 \\ \rho_1 & 1 & \rho_1 & \cdots & \rho_{k-3} & \rho_2 \\ \vdots & \vdots & \vdots & \cdots & \vdots & \vdots \\ \rho_{k-1} & \rho_{k-2} & \rho_{k-3} & \cdots & \rho_1 & \rho_k \end{vmatrix}}{\begin{vmatrix} 1 & \rho_1 & \rho_2 & \cdots & \rho_{k-2} & \rho_1 \\ \rho_1 & 1 & \rho_1 & \cdots & \rho_{k-3} & \rho_2 \\ \vdots & \vdots & \vdots & \cdots & \vdots & \vdots \\ \rho_{k-1} & \rho_{k-2} & \rho_{k-3} & \cdots & \rho_1 & 1 \end{vmatrix}} \quad (2.4)$$

2.6. Jaringan Syaraf Tiruan

Jaringan syaraf tiruan (JST) bekerja layaknya otak manusia. Karakteristik utama yang dimiliki oleh JST adalah kemampuan dalam belajar, hal ini dapat diartikan menjadi proses penyesuaian parameter berbobot sebab *output* yang diinginkan tergantung pada nilai pembobotan interkoneksi yang dimiliki set. Pada proses ini akan dihentikan jika nilai *error* telah dianggap cukup kecil untuk semua pasang data latih. Jaringan yang sedang melakukan proses belajar biasanya berada dalam tahap pelatihan, baru setelah selesai akan ada tahap pengujian suatu objek (Pangaribuan, 2017).

Jaringan syaraf tiruan (JST) merupakan salah satu bagian dari machine learning dan merupakan teknik untuk memodelkan pemrosesan informasi berdasarkan kemampuan sistem saraf otak yang ada di manusia. Cara kerja pada otak manusia inilah yang ditiru oleh Jaringan syaraf tiruan dalam pengolahan informasi dengan mengatasi sejumlah proses perhitungan pada komputer (Kusumadewi, 2003).

Pada jaringan syaraf tiruan memiliki tahap pengujian dan tahap pelstihsn dimana hasil bobot pada tahap pelatihan akan menghasilkan nilai *error* yang minimum dan kecil pada tahap pengujian. Sedangkan untuk tahap pelatihan sendiri data latih akan dimasukan kedalam jaringan dimana ada *node* yang terhubung dan

data uji yang belum pernah dilatih akan diujikan dengan bobot yang ada di tahap pelatihan (Warsito, 2009).

Jaringan syaraf tiruan terdiri dari 3 lapisan yaitu Input Layer, Hidden Layer dan Output Layer. Pada tiap lapisan bertanggung jawab untuk melakukan fungsi yang sama yaitu melengkapi sistem. Murphy (2008) mengemukakan terdapat 3 hal dalam jaringan syaraf tiruan. Yaitu:

1) Pola hubungan antara neuron yang biasa disebut dengan arsitektur jaringan.
2) Metode untuk menentukan bobot yang biasa disebut dengan metode pembelajaran.
3) Fungsi Aktivasi, yaitu fungsi yang akan digunakan.

2.6.1. Komponen Jaringan Syaraf Tiruan

Jaringan saraf tiruan (JST) secara khusus dirancang berdasarkan cara kerja bagian dalam otak manusia. Model-model ini meniru fungsi *neuron* yang saling berhubungan dengan melewatkan fitur *input* melalui beberapa lapisan yang disebut sebagai *perceptrons*, masing-masing mengubah *input* menggunakan serangkaian fungsi. Maka komponen-komponen yang terdapat didalam jaringan syaraf tiruan adalah sebagai berikut :

a. *Neuron* / Unit

   Istilah *neuron* buatan atau paling sering hanya *neuron* adalah istilah yang setara dengan unit, tetapi menyiratkan hubungan yang erat dengan neurobiologi dan otak manusia sementara pembelajaran mendalam sangat sedikit hubungannya dengan otak (misalnya, sekarang dianggap bahwa biologis *neuron* lebih mirip dengan seluruh perceptron *multilayer* daripada satu unit dalam jaringan saraf). Istilah *neuron* didorong setelah musim dingin AI terakhir untuk membedakan jaringan saraf yang lebih sukses dari perceptron yang gagal dan ditinggalkan. Namun, sejak keberhasilan pembelajaran mendalam setelah 2012, media sering mengambil istilah "*neuron*" dan berusaha menjelaskan pembelajaran mendalam sebagai mimikri otak manusia, yang sangat menyesatkan dan berpotensi berbahaya

bagi persepsi bidang pembelajaran yang mendalam. Sekarang istilah *neuron* tidak disarankan dan istilah unit yang lebih deskriptif harus digunakan sebagai gantinya (Shidhart, 2020).

b. Lapisan

Lapisan atau sering kita dengan dengan sebutan *layer* merupakan blok bangunan tingkat tertinggi dalam *deep learning*. Lapisan adalah wadah yang biasanya menerima *input* berbobot, mengubahnya dengan serangkaian fungsi non-linier dan kemudian meneruskan nilai-nilai ini sebagai *output* ke lapisan berikutnya (Shidhart, 2020).

c. *Input*

Pada dasarnya kita tidak bisa menganalisis sebagaian besar kumpulan daya yang kita temukan di dunia ini dengan menggunakan data kualitatif berupa grafik, gambat mauapun suara. Pada jaringan syaraf tiruan, data yang diterima atau diolah adalah berupa data numerik. Yang artinya jika data masih berupa data kualitatif maka kita harus mentransformasikan terlebih dahulu ke dalam data numerik sebelum masuk ke proses jaringan syaraf tiruan (Shidhart, 2020).

d. *Output*

Output dari jaringan syaraf tiruan (JST) merupakan hasil dari penyelsaian terhadap masalah. Data dari *output* berupa data numerik (Shidhart, 2020)..

e. Bobot

Bobot adalah parameter dalam jaringan saraf yang mengubah data *input* di dalam lapisan tersembunyi jaringan. Jaringan saraf adalah serangkaian *node*, atau *neuron*. Dalam setiap *node* adalah satu set *input*, bobot, dan nilai bias. Saat *input* memasuki *node*, *input* dikalikan dengan nilai bobot dan *output* yang dihasilkan diamati, atau diteruskan ke lapisan berikutnya dalam jaringan saraf. Seringkali bobot jaringan saraf terkandung di dalam lapisan tersembunyi jaringan (Shidhart, 2020).

f. Bias

Bias adalah ciri yang menunjukkan hasil algoritma untuk mendukung atau menentang ide. Bias dianggap sebagai kesalahan sistematis yang terjadi

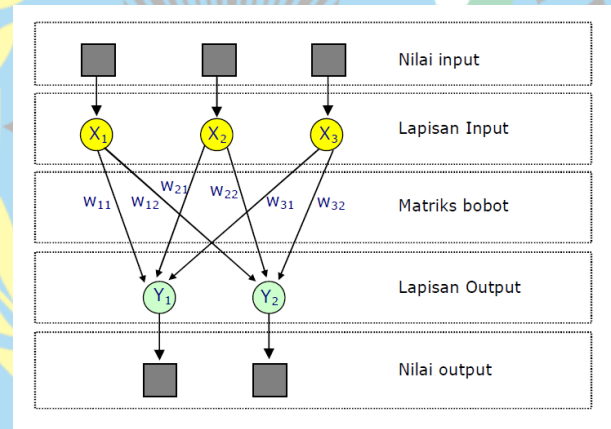
pada model pembelajaran mesin itu sendiri karena asumsi yang salah dalam proses *Machine Learning* (Shidhart, 2020).

2.6.2. Arsitektur Jaringan Syaraf Tiruan

Ada 3 jenis arsitektur jaringan syaraf tiruan (Kusumadewi, 2003).

a. *Single Layer Network*

Jaringan saraf satu lapis mewakili bentuk paling sederhana dari jaringan saraf, di mana hanya ada satu lapisan *node input* yang mengirim *input* berbobot ke lapisan berikutnya dari *node* penerima, atau dalam beberapa kasus, satu *node* penerima. Desain lapisan tunggal ini merupakan bagian dari fondasi untuk sistem yang kini menjadi jauh lebih kompleks. Setiap lapisan bobot yang terkoneksi memiliki sebuah jaringan single *layer* (Fausset, 1994).

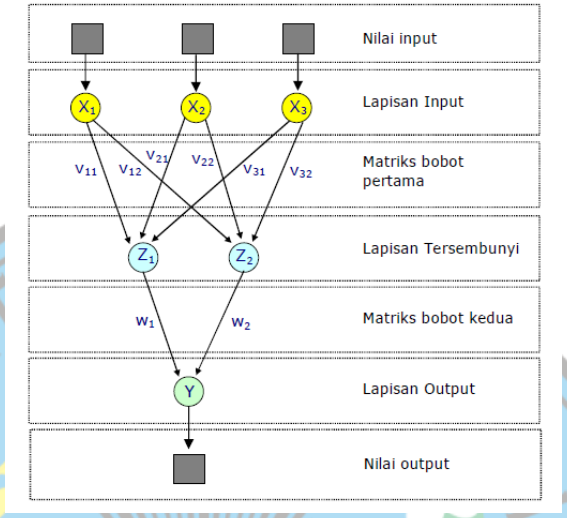


Gambar 2.1 Jaringan syaraf tiruan untuk *Single Layer Network* (Anggraeny,2019, p.20)

b. *Multi-Layer Network*

Jaringan saraf *multi-layer network* dapat diatur dengan berbagai cara. Biasanya, mereka memiliki setidaknya satu lapisan *input* yang mengirimkan *input* berbobot ke serangkaian lapisan tersembunyi dan lapisan *output* di bagian akhir. Pengaturan yang lebih canggih ini juga dikaitkan dengan bangunan nonlinier menggunakan sigmoid dan fungsi lain untuk mengarahkan pengaktifan atau aktivasi *neuron* buatan. Sementara beberapa

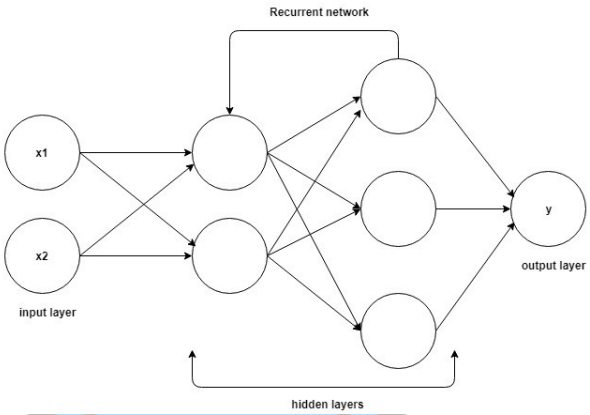
dari sistem ini dapat dibangun secara fisik, dengan bahan fisik, sebagian besar dibuat dengan fungsi perangkat lunak yang memodelkan aktivitas saraf (Fausset, 1994).



Gambar 2.2 Jaringan syaraf tiruan untuk *Multi-Layer Network* (Anggraeny,2019, p.21)

c. *Reccurent Network*

Recurrent Network adalah jenis jaringan saraf tiruan yang biasa digunakan dalam pengenalan suara dan pemrosesan bahasa alami. Jaringan saraf ini mengenali karakteristik sekuensial data dan menggunakan pola untuk memprediksi kemungkinan skenario berikutnya. *Recurrent Network* digunakan dalam *deep learning* dan dalam pengembangan model yang mensimulasikan aktivitas *neuron* di otak manusia. Jaringan ini sangat baik dalam permasalahan penggunaan di mana konteks sangat penting untuk memprediksi hasil, dan juga berbeda dari jenis jaringan saraf tiruan lainnya karena mereka menggunakan *loop* umpan balik untuk memproses urutan data yang menginformasikan hasil akhir. *Loop* umpan balik ini memungkinkan informasi untuk bertahan. Efek ini sering digambarkan sebagai memori (Britz, 2015).



Gambar 2.3 Jaringan syaraf tiruan untuk *Recurrent Network* (Vidushi, 2018)

2.6.3. Fungsi Aktivasi

Net input merupakan unit terpenting dalam struktur jaringan syaraf tiruan yang diproses dan diubah menjadi hasil keluaran yang dikenal sebagai aktivasi unit dengan menerapkan fungsi yang disebut fungsi aktivasi atau fungsi ambang atau fungsi transfer yang merupakan transformasi skalar ke skalar. (Siddhart, 2020)

Jaringan syaraf tiruan terdapat beberapa jenis dari fungsi aktivasi (Julpan, 2015).

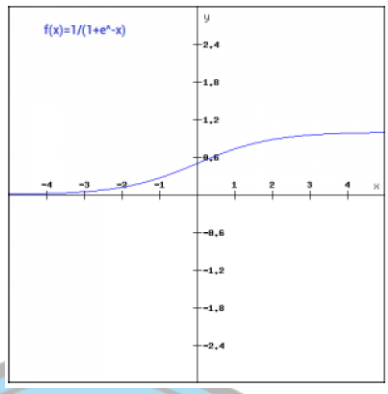
a. Fungsi Sigmoid

Ini adalah fungsi aktivasi yang paling banyak digunakan karena merupakan fungsi *non-linear*. Fungsi sigmoid mengubah nilai dalam rentang 0 hingga 1. Dapat didefinisikan sebagai:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.5}$$

Fungsi sigmoid terdiferensialkan terus menerus dan fungsi berbentuk S halus. Turunan dari fungsi tersebut adalah:

$$f'(x) = f(x)(1 - f(x)) \tag{2.6}$$



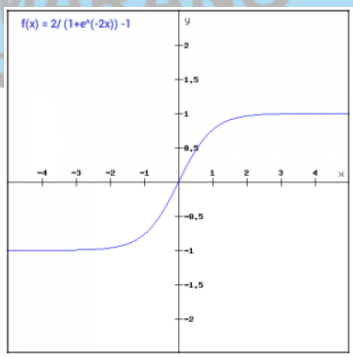
Gambar 2.4 Fungsi Sigmoid

b. Fungsi Tanh

Ini adalah fungsi Tangen Hiperbolik. Fungsi Tanh mirip dengan fungsi sigmoid tetapi simetris dengan sekitar titik asal. Ini menghasilkan tanda-tanda keluaran yang berbeda dari lapisan sebelumnya yang akan diumpankan sebagai *input* ke lapisan berikutnya. Ini dapat didefinisikan sebagai:

$$f(x) = \frac{1 - e^{-x}}{1 + e^{-x}} \tag{2.7}$$

Fungsi Tanh kontinu dan terdiferensiasi, nilainya terletak pada rentang -1 hingga 1. Dibandingkan dengan fungsi sigmoid, gradien fungsi tanh lebih curam. Tanh lebih disukai daripada fungsi sigmoid karena memiliki gradien yang tidak dibatasi untuk bervariasi dalam arah tertentu dan juga berpusat di nol.



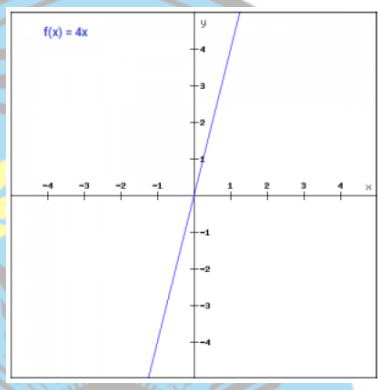
Gambar 2.5 Fungsi Tanh

c. Fungsi Aktivasi Linear

Fungsi aktivasi linier berbanding lurus dengan *input*. Kelemahan utama dari fungsi langkah biner adalah memiliki gradien nol karena tidak ada komponen x dalam fungsi langkah biner. Untuk menghilangkannya, dapat digunakan fungsi linier. Itu bisa didefinisikan sebagai berikut:

$$f(x) = ax \tag{2.8}$$

Nilai variabel a dapat berupa nilai konstan yang dipilih oleh pengguna.



Gambar2.6 Fungsi Aktivasi Linear

Di sini turunan dari fungsi f(x) tidak nol tetapi sama dengan nilai konstanta yang digunakan. Gradien tidak nol, tetapi nilai konstanta yang tidak tergantung pada nilai *input* x yang menyiratkan bahwa bobot dan bias akan diperbarui selama langkah *Backpropagation* meskipun faktor pemutakhiran akan sama. Tidak banyak manfaat menggunakan fungsi linier karena jaringan saraf tidak akan memperbaiki kesalahan karena nilai gradien yang sama untuk setiap iterasi. Selain itu, jaringan tidak akan dapat mengidentifikasi pola kompleks dari data. Oleh karena itu, fungsi linier ideal di mana interpretasi diperlukan dan untuk tugas-tugas sederhana (Ali, 2016).

2.7. *Cascade Forward Neural Network* (CFNN)

Pada dasarnya arsitektur jaringan dari *Cascade Forward Neural Network* terlihat sama dengan arsitektur jaringan yang dimiliki oleh *Feedforward Neural Network*. Pada perceptron koneksi yang terbentuk antara *input* dan *output* merupakan bentuk hubungan langsung sedangkan pada FFNN koneksi yang

terbentuk antara *input* dan *output* merupakan hubungan tidak langsung Koneksi berbentuk nonlinier melalui fungsi aktivasi di lapisan tersembunyi. Jika bentuk koneksi pada jaringan perceptron dan *multilayer* digabungkan, maka jaringan dengan koneksi langsung antara lapisan *input* dan lapisan *output* terbentuk, selain koneksi tidak langsung. Jaringan yang terbentuk dari pola koneksi ini disebut *Cascade Forward Neural Network* (Warsito, 2018).

Bentuk model dari *Cascade Forward Neural Network* bisa dituliskan seperti dibawah ini :

$$y = \sum_{i=1}^{I} f^{i} u_{ik} x_i + f^{0}\left(w_{ok} + \sum_{j=1}^{J} w_{jk} f^{j}\left(v_{oj} + \sum_{j=1}^{J}\sum_{i=1}^{I} v_{ij} x_i\right)\right) \quad (2.9)$$

dimana :

v_{ij} : bobot pada lapisan *input* ke-i dan lapisan tersembunyi ke-j

w_{jk} : bobot pada lapisan *input* ke-j dan lapisan tersembunyi ke-k

u_{ik} : bobot pada lapisan *input* ke-i dan lapisan tersembunyi ke-k

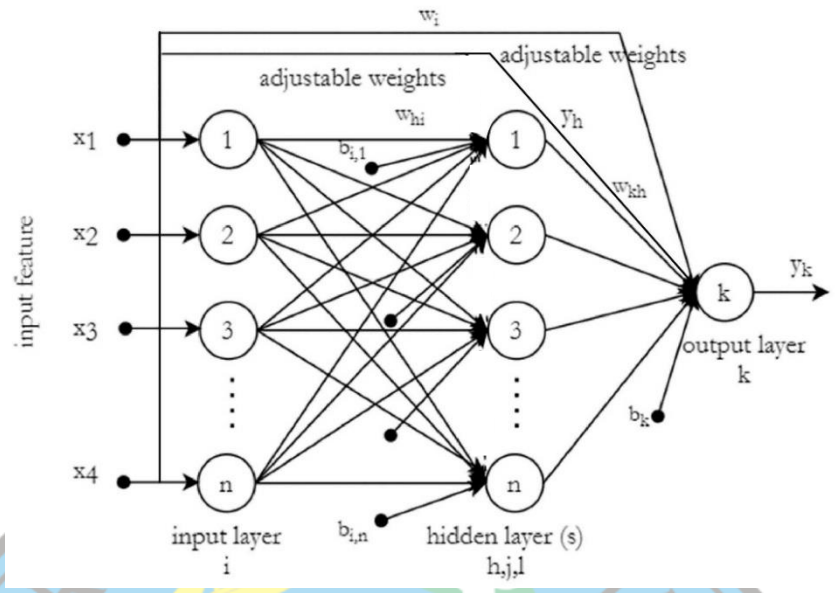
v_{oi} : bobot bias lapisan tersmbunyi ke-i

w_{ok} : bobot bias lapisan tersembunyi ke-k

f^{j} : fungsi aktivasi untuk lapisan tersembunyi

f^{0} : fungsi aktivasi untuk *output*

f^{i} : fungsi aktivasi dari *input* ke ouput



Gambar 2.7 Arsitektur jaringan *Cascade Forward Neural Network* (Gasim Hayder, 2020)

Gambar 2.7 menunjukkan bahwa terdapat hubungan secara langsung antara *input* dan *output*. Konsekuensi dari bentuk hubungan ini adalah bahwa bobot jaringan yang akan diestimasi meningkat sebanyak *neuron* dalam lapisan masukan. Seperti halnya FFNN, algoritma *Backpropagation* pada CFNN juga terdiri dari tiga tahap: *feedforward* dari pola *input*, penaksiran kesalahan dan bobot yang disesuaikan. Seperti yang dijelaskan sebelumnya, setelah tahap *feedforward* proses dilanjutkan dengan perhitungan *error* (selisih dari *output* ke sasaran, tujuan). Langkah selanjutnya adalah memperbarui bobot dan melakukan perhitungan ulang. Langkah ini dilakukan sampai tidak terjadi *error* atau sampai iterasi berhenti sesuai dengan kriteria *stop* yang ditentukan (Warsito, 2018).

2.8. *Backpropagation*

Algoritma *Backpropagation* adalah algoritma pembelajaran terawasi yang paling umum. Konsep dari algoritma ini adalah untuk mengatur bobot yang meminimalkan kesalahan antara *output* aktual dan *output* yang diprediksi dari JST menggunakan fungsi berdasarkan aturan delta. Ini melibatkan bekerja mundur dari lapisan *output* untuk menyesuaikan bobot yang sesuai dan mengurangi kesalahan

rata-rata di semua lapisan. Proses ini diulang sampai kesalahan *output* diminimalkan. (Benyounis, 2019)

Dengan menggunakan algoritma *Backpropagation* , pelatihan jaringan terdiri dari tiga tahap (Adiwijaya, et al, 2013).

1. **Feedforward**
   a. Tiap masukan sinyal x_i diteruskan ke lapisan tersembunyi.
   b. Tiap unit tersembunyi Z_j menjumlahkan sinyal-sinyal masukan berbobot :

$$Z_{inj} = v_{oj} + \sum_{i-1}^{p} x_i v_j \tag{2.10}$$

   Gunakan fungsi aktivasi untuk hitung sinyal keluaran :

$$z_j = f(Z_{inj}) \tag{2.11}$$

   c. Tiap unit keluaran (Y) menjumlahakan sinyal-sinyal masukan berbobot.

$$Y_{ink} = w_{0k} + \sum_{j=1}^{q} z_j w_{jk} \tag{2.12}$$

   Gunakan fungsi aktivasi untuk menghitung sinyal keluaran :

$$y = f(Y_{ink}) \tag{2.13}$$

2. **Backpropagation**
   a. Tiap unit keluaran (Y) menerima target pola yang berkaitan dengan pola masukan pelatihan, hitung nilai kesalahan :

$$\delta_k = (t_k - y) f'(y_{ink}) = (t_k - y)(1 - y) \tag{2.14}$$

   b. Tiap unit tersembunyi $(z_j, j = 1,2,3, \dots q)$ menjumlahkan delta masukan dari unit lapisan keluaran.

$$\delta_{in_j} = \sum_{j=1}^{q} \delta_k w_{jk} \tag{2.15}$$

   Nilai ini dikalikan turunan dari fungsi aktivasi untuk mendapatkan nilai kesalahan :

$$\delta_j = \delta_{in_j} f'(z_{inj}) = \delta_{in_j} z_j (1 - z_j) \tag{2.16}$$

c. Hitung gradient di unit output dari fungsi objektif yang sudah ditentukan.

$$g_{k+1} = \frac{1}{N}\sum_{n=1}^{p} \delta_{nk} y_{nk} \tag{2.18}$$

d. Hitung gradient di unit tersembunyi

$$g_{k+1} = \frac{1}{N}\sum_{n=1}^{p} \delta_{nj} y_{nj} \tag{2.19}$$

e. Hitung parameter β untuk semua neuron di unit tersembunyi dan unit output, dengan menggunakan formula Fletcher-Reeves:

$$\beta_k = \frac{g_k^T g_k}{g_{k-1}^T g_{k-1}} \tag{2.20}$$

f. Hitung direction untuk semua neuron di unit tersembunyi dan output

$$d_{k+1} = -g_{k+1} + \beta_k d_k \tag{2.21}$$

Untuk direction awal:

$$d_1 = -g_1 \tag{2.22}$$

3. Penyesuaian Bobot Baru

$$w_{k+1} = w_k + \alpha_{k+1} d_{k+1} \tag{2.23}$$

Keterangan :

x = unit pada masukan (*Input*)

z_j = unit ke-j pada lapisan tersembunyi (*Hidden*)

y = unit pada lapisan keluaran (*Output*)

f = fungsi aktivasi

v_j = nilai bobot dari *input* ke *hidden*

w_j = nilai bobot dari *hidden* ke *input*

$z_{net\ j}$ = keluaran untuk lapisan tersembunyi

y_{net} = masukan untuk keluaran

δ_j = faktor pengaturan nilai penimbang sambungan pada *hidden*

δ_k = faktor pengaturan nilai penimbang sambungan pada *output*

α = konstanta laju pembelajaran $0 < a < 1$

β_k = nilai parameter pada iterasi saat ini

g_{k+1} = gradien pada iterasi saat ini

g_k = gradien pada iterasi sebelumnya

d_{k+1} = *direction* pada iterasi saat ini

d_k = *direction* pada iterasi sebelumnya

2.9. Kriteria Model Terbaik

Untuk menghasilkan peramalan yang akurat dilakukan evaluasi dengan menggunakan metode MAPE (*Mean Absolute Percentage Error*). MAPE menunjukkan seberapa besar kesalahan dalam memprediksi dibandingkan dengan nilai sebenarnya. (Khair, 2017). Untuk menghitung nilai dari MAPE bisa menggunakan rumus seperti dibawah ini:

$$MAPE = \frac{1}{n}\sum_{t=1}^{n} \frac{\left|X_t - \hat{X}_t\right|}{X_t} \times 100\% \qquad 2.22$$

Keterangan :

n : ukuran sampel

X_t : nilai data aktual

$\hat{X}_t$: nilai data peramalan

Semakin kecil nilai MAPE menunjukkan bahwa persentase kesalahan yang dihasilkan oleh model juga semakin kecil. Nilai evaluasi yang dihasilkan mempunyai kriteria MAPE sebagai berikut (Halimi et al, 2013):

a. MAPE < 10% : Kemampuan peramalan sangat baik.
b. 10% < MAPE < 20% : Kemampuan peramalan baik
c. 20% < MAPE < 50% : Kemampuan peramalan cukup
d. MAPE ≥ 50% : Kemampuan peramalan buruk