BAB II

LANDASAN TEORI

2.1 Neural Network

Pada sub bab ini mengulas tentang dasar-dasar neural network yang digunakan dalam pembuatan proses aplikasi ini.

2.1.1 Sejarah singkat Jaringan Saraf Tiruan (JST)

Perkembangan dari ilmu jaringan saraf biologi telah memungkinkan para peneliti untuk mengembangkan model matematika dari neuron untuk mensimulasikan tingkah laku jaringan saraf. Usaha keras untuk dapat mengenali cara kerja otak dikerjakan dan dimulai pertama kali oleh Ramon Cajal (1911), yang mengenalkan ide *neuron* yaitu struktur pokok dari otak manusia. Ide ini kemudian dikembangkan lagi awal tahun 1940-an ketika model abstrak pertama neuron diperkenalkan oleh MCCulloch dan Pitts (1943). Hebb (1949) menjelaskan bagaimana jaringan saraf belajar. Peneliti yang lain mengembangkannya dalam dua dekade berikutnya, seperti Minsky (1954) dan Rosenblatt (1958). Rosenblatt mengembangkan algoritma belajar perceptron. Pada waktu yang sama Windrow dan Hoff mengembangkan suatu variasi penting dari algoritma belajar perceptron, yang kemudian dikenal dengan Windrow-Hoff rule.

Kemudian Minsky dan Pert (1969) mengemukan teori batas model neural network *single-layer* yang ditulis pada buku mereka *Perceptron*. Dikarenakan rasa pesimistis terhadap proyek mengenai neural network ini, penelitian dari neural network mengalami perubahan dan kemunduran dalam dua dekade. Walapun dalam suasana yang tidak baik, beberapa peneliti masih meneruskan penelitian

mereka dan menghasilkan hasil yang berarti. Contohnya, Anderson (1977) dan Grossberg (1980), yang melakukan pekerjaan penting mengenai model *Psychological*. Kohonen (1977) menegambangkan asosiatif model memori.

Kebangkitan neural network pada awal 1980-an. Hopfield (1982) mengenalkan ide meminimalkan energi secara fisik dalam neural network. Akibat dari tulisannya tersebut mengilhami teknologi ini dengan momentum yang dapat diperbaharui. Feldman dan Ballard pada tahun 1982 mengemukan syarat "connectionist" yang terkenal.

Pada pertengahan 1980-an, dalam buku *Parallel Distibuted Processing* yang ditulis oleh Rumelhart dan McClelland (1986), menimbulkan dampak besar pada pengetahuan komputer, dan biologi. Khususnya, algoritma pembelajaran *backpropagation* yang dikembangkan oleh Rumelhart, Hinton, dan Williams (1986) menghasilkan suatu solusi yang sangat berati dalam melakukan suatu training terhadap *multilayer* neural network. Keberhasilan spektakuler dari pendekatan ini di demonstrasikan oleh system *NETtalk* yang dikembangkan Sejnowski dan Rosenberg (1987), yaitu suatu sistem yang dapat menterjemahkan teks dalam bahasa Inggris kedalam bentuk suara dengan gaya bahasa yang dapat dimengerti dengan sangat jelas.

Pendekatan simbolik yang telah lama didominasi oleh bagian dari AI akhirnya telah dimenangkan oleh pendekatan neural network. Telah ada spekulasi tentang apakah pendekatan pertama seharusnya mensubstistusikan ke yang lain atau apakah pendekatan kedua seharusnya jalan bersama dan dikombinasikan. Fakta-fakta lain yang menunjukkan lebih baik memilih integrasi yang pola *low-level* kemampuannya diakui oleh pendekatan neural network dan *high-level* yang

menjadi alasan kemampuan, disediakan oleh pendekatan secara simbolik yang dikomplemen dengan lainnya. (Kandel dan Langholz, 1992). Arsitektur yang optimal pada sistem berpikir masa depan mungkin lebih baik dibangun dari integrasi antara cara yang satu dengan lainnya. (Setiawan, 2003)

2.1.2 Pengertian Neural Network

Jaringan Saraf Tiruan (JST) merupakan simulasi dari otak biologis. Tujuan dari JST adalah untuk belajar mengenali pola-pola pada data dan mensimulasikan proses belajar adaptif biologis, walau dalam skala yang sangat sederhana.. Sekali JST telah dilatih terhadap data, akan dapat membuat prediksi dengan melakukan deteksi kemiripan/kesamaan pola-pola data masukan.

JST bukanlah duplikasi persis dari sistem biologis otak manusia, tetapi jaringan saraf tiruan ini dapat melakukan kemampuan seperti generalisasi, belajar, abstaksi, dan bahkan intuisi. Mudahnya, merupakan suatu model dari sistem saraf biologis yang disederhanakan sebagai suatu alternatif sistem komputer. Kenyataan menunujukan bahwa banyak masalah dalam kehidupan manusia yang sulit dipecahkan dengan "komputer konvensional" yang paling canggih sekalipun, namun manusia dapat menyelesaikannya dengan baik. Dengan kemampuannya untuk belajar, jaringan saraf tiruan ini diharapkan dapat menyelesaikan masalah yang tidak dapat diselesaikan oleh komputer konvensional.

Jaringan saraf tidak diprogram dalam arti tradisional. Sebaliknya dilatih dengan contoh. Latihan itu terdiri dari banyak pengulangan input yang mengungkapkan berbagai hubungan. Dengan memperhalus bobot node sistem (neuron yang disimulasikan) secara progresif, jaringa saraf tiruan ini "menemukan" hubungan antar input. Proses penemuan ini menandakan "belajar".

JST merupakan salah satu bentuk dari Kecerdasan Buatan. JST dipandang sebagai suatu Black Box yang dapat melakukan prediksi keluaran dari suatu pola masukan yang dia kenali. Untuk itu JST harus dilatih terlebih dahulu terhadap sejumlah pola masukan dan target yang diharapkan dari tiap pola masukan tersebut (Supervised Learning). Sekali dilatihkan, JST akan dapat mengenali kesamaan ketika dihadapkan terhadap pola masukan baru, dan menghasilkan prediksi pola keluarannya.

JST dapat mendeteksi kesamaan masukan, bahkan sebagian masukan yang mungkin belum pernah dilatihkan atau diberikan sebelumnya. Karena JST mempunyai kemampuan interpolasi yang hebat, terutama bila data masukan tidak eksak, banyak gangguan didalamnya. Sehingga memungkinkan JST untuk digunakan sebagai subtitusi langsung bagi auto korelasi, regresi multivariabel, regresi linier, trigonometri, dan teknik regresi lainnya. Ketika data dianalisa menggunakan JST, akan memungkinkan untuk melakukan prediksi pola yang penting sebagaimana bila seorang ahli menganalisa data tersebut, karena JST dapat beraksi seperti selayaknya seorang yang ahli di bidangnya. (Setiawan, 2003)

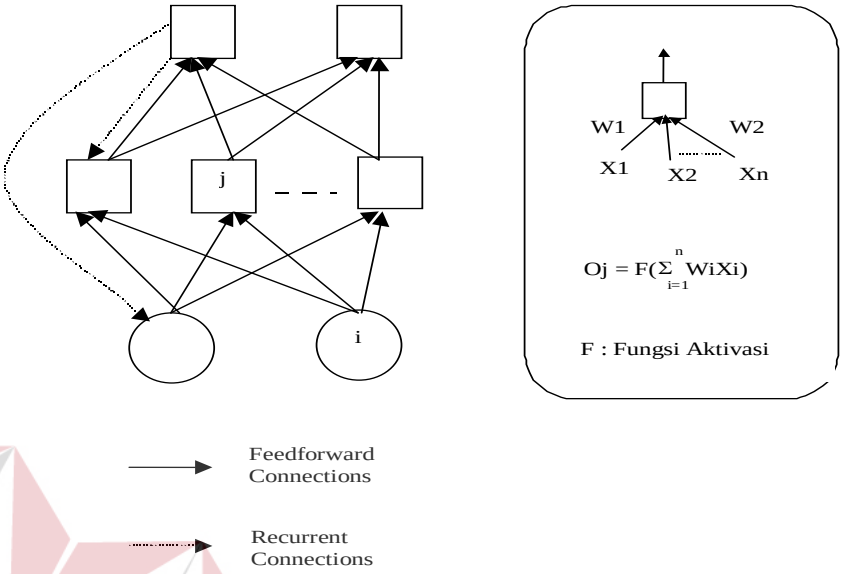
2.1.3 Konsep dasar Neural Network

Neural network mempunyai arsitektur distribusi parallel yang terdiri dari banyak node-node dan penghubung. Antara titik penghubung dari satu node ke lainnya dihubungkan dengan suatu bobot. Untuk membuat kontruksi dari suatu neural network meliputi aturan yang terdiri dari :

Menentukan properti network : topologi network (hubungannya), tipe hubungan, order dari hubungan, dan batas bobot.

Menentukan properti node : nilai aktivasi dan fungsi aktivasi

Menentukan system dinamik : menyusun inisialisasi bobot, formula penghitungan aktivasi dan aturan pembelajaran.



Gambar 2.1. Model sederhana dari struktural neural network.

Terdapat beberapa hal yang berkaitan dengan Neural network yaitu :

Properti Network

Topologi dari neural network mengacu pada *framework* sebagaimana skema interkoneksi. Framework biasanya sering terdirir dari beberapa layer (slabs) dan beberapa node per layer. Tipe dari layer terdiri dari :

Layer *Input*. Node dalam hal ini dinamakan dengan *input units*, yang mana encode pada suatu kasus ditampilkan pada network untuk diproses. Sebagai contoh, tiap intput unit mungkin di disain oleh nilai atribut acak (random) pada suatu kasus.

Layer Hidden. Node dalam hal ini dinamakan dengan *hidden untis*, yang mana tidak secara langsung kelihatan dan dampak hidden.. layer hidden menyediakan nonlinier untuk network.

Layer *Output*. Node dalam hal ini dinamakan dengan *output* units, encode mungkin berupa konsep (nilai) yang dialokasikan pada suatu kasus. Contohnya, tiap output unit mewakili sebuah *class* pada suatu obyek.

Input unit tidak memproses informasi, tetapi hanya mendistribusikan informasi ke units yang lain. Secara skema, input units digambarkan sebagai lingkaran dari suatu proses elemen seperti units hidden dan units output yang digambarkan juga sebagai segiempat.

Properti Node

Level aktivasi dari node dapat dipisahkan (0 dan 1) atau dilanjutkan sesuai dengan jangkauan (0,1) atau batasan yang dibuat. Hal ini tergantung dari fungsi aktivasi (transfer) yang dipilih. Jika ada pembatasan fungsi, maka level aktivasi adalah 0 (atau -1) dan 1. untuk fungsi *sigmoid* level aktivasi dibatasi pada jangkauan real (0,1)

Dinamik Sistem

Skema inisialisasi bobot di khususkan untuk model neural network khusus yang dipilih. Meskipun, dalam banyak kasus inisialisasi bobot hanya digunakan nilai random untuk memperkecil angka real.

Aturan pembelajaran adalah salah satu dari kebanyakan atribut penting, untuk menspesifikasikan suatu neural network Pada aturan pembelajaran menjelaskan bagaimana menyesuaikan bobot koneksi dalam tingkatan untuk

mengoptimalkan kinerja network. Ini mengindikasikan bagaimana menghitung bobot penyesuaian selama siklus pelatihan. Meskipun, aturan disesuaikan setelah pelatihan selesai.

Ketika neural network digunakan untuk menyelesaikan suatu masalah, penyelesaian didapatkan dalam level aktivasi pada units output. Sebagi contoh, tujuan neural network digunakan untuk mengklasifikasikan buah-buahan kedalam lemon, jeruk dan apel. Network mempunyai tiga unit output untuk merepresentasikan tiga jenis klasifikasi. penentuan karakteristik informasi yang diterima oleh layer input dan *propagated forward.* Jika unit output sesuai dengan class apel dan mencapai aktivasi maksimum, maka class ditandai untuk buah apel. Pada contoh ini, kesimpulan tingkah laku dari neural network termasuk perhitungan level aktivasi yang berlaku di network. Satu hal yang harus diperhatikan bahwa pelatihan dalam neural network meliputi perhitungan untuk mengetahui level aktivasi aktual dan level aktivasi yang dibutuhkan untuk mengetahui kesalahan yang kemudian digunakan untuk penyesuaian bobot.

Fungsi aktivasi diperlukan dalam perhitungan input, unit hidden dan unit output, sesuai dengan tipe fungsi aktivasi yang digunakan. Disediakan fungsi sigmoid, fungsi aktivasi (*Oj*) dari unit *j* dihitung dengan :

$$Oj = 1 / [\ 1 + e^{-(\Sigma i\ WjiXi-\theta j)}\ ] \qquad (2.1)$$

Dimana Xi adalah inputan dari unit i, Wji bobot dari koneksi dari unit i ke unit j, dan θj batas atas dari unit *j*. Dalam kasus fungsi aktivasi yang *hard-limiting* output dari neuron diperoleh dari

$$Oj = \begin{cases} 1, \Sigma_i\ WijXi > \theta j \\ 0, \end{cases} \qquad (2.2)$$

D. Model Neuron

Suatu *neuron* merupakan unit dasar pemroses informasi, untuk mengoperasikan neural network. Ada tiga elemen dasar dari model neuron, yaitu :

1. *Set of synapses* atau *link* koneksi, semua karakteristik yang dimiliki bobot atau *strength.* Khususnya, suatu sinyal x_j pada input *j* yang mengkoneksikan ke neuron *k* dikalikan dengan bobot koneksi $W_{kj.}$ Ini penting untuk membuat catatan dari tingkah laku yang ditulis secara *subscripts* dari koneksi bobot W_{kj}. Subscript yang pertama mengacu pada pertanyaan neuron, subscript kedua mengacu pada input akhir dari koneksi yang diacu oleh bobot. Bobot W_{kj} memiliki nilai positif jika koneksi diasosiasikan sebagai pendukung, dan negatif jika koneksi sebagai penghambat.
2. *Adder* untuk menjumlahkan sinyal input, dengan bobot koneksinya sendiri dari neuron, operasi disini menjelaskan tujuan dari *linier combiner.*
3. *Activation function* digunakan untuk membatasi luas dari output neuron. Fungsi aktivasi juga diacu sebagai literatur *squash function* yang menekan (membatasi) luas sinyal output yang diperbolehkan pada nilai yang terbatas. Biasanya, *range* luas output dari neuron ditulis dalam kurung siku [0,1] atau lainnya [-1,1].

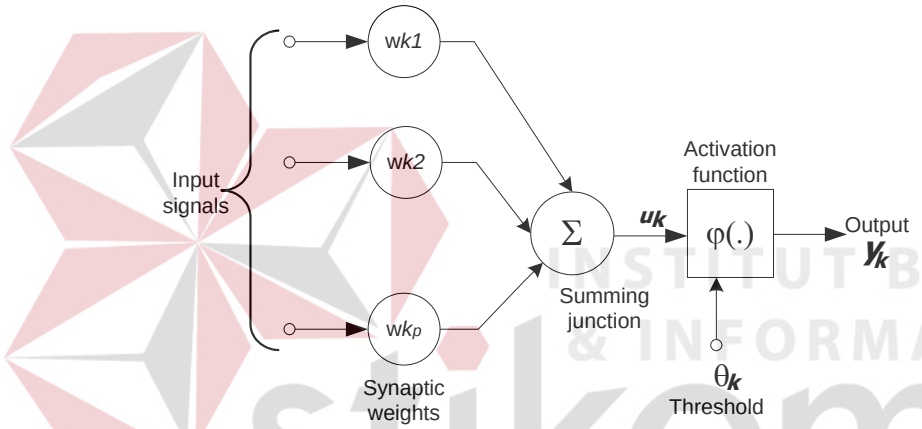
Model neuron yang ditunjukkan pada gambar 2.2, juga memasukkan *threshold* θK terapan eksternal, yang mempunyai efek dari penurunan input jaringan pada fungsi aktivasi. Di lain pihak, jaringan inipun untuk fungsi aktivasi mungkin bisa ditingkatkan dengan membuat sebuah periode bias dibandingkan sebuah threshold, nilai biasnya adalah negatif dari threshold.

Dalam istilah matematika, dapat digambarkan bahwa sebuah neuron ditulis dengan pasangan persamaan sebagai berikut:

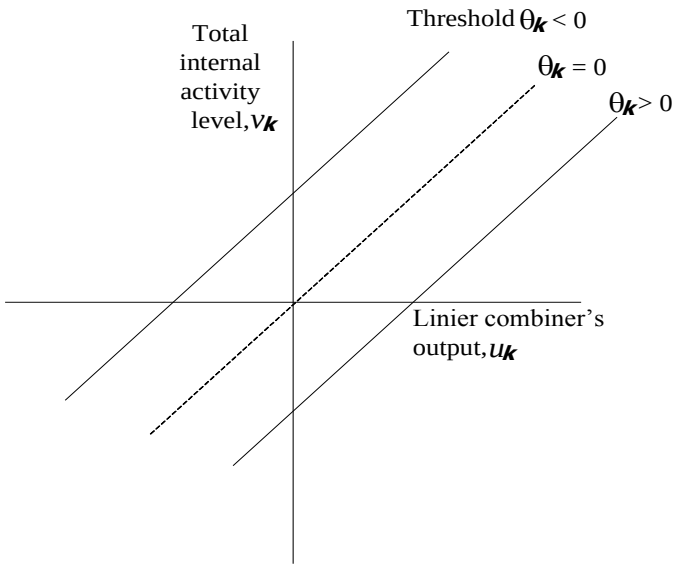
$$u_k = \sum_{j=1}^{p} w_{kj} x_j \qquad (2.3)$$

$y_k = \varphi\,(u_k - \theta_K)$

Dimana *x1, x2,…,xp* adalah sinyal input ; *wk1, wk2,…,wkp* adalah bobot synapsi dari neuron *k*; *uk* adalah output kombiner yang linear; *θK* adalah *threshold*; φ(.) adalah fungsi aktivasi, dan *yk* adalah sinyal output dari neuron.



Gambar 2.2. Model nonlinier dari sebuah neuron



Gambar 2.3. Hasil transformasi yang dihadirkan oleh threshold

Penggunaan *threshold* θk mempunyai efek dari penerapan transformasi pada output *uk* dari kombiner linier dalam model gambar 2.2, yang ditunjukkan oleh persamaan

$$vk = uk - \theta k \qquad (2.5)$$

Secara khusus, bergantung pada nilai θk adalah positif atau negatif, hubungan diantara fungsi aktifitas internal yang efektif atau potensial aktivasi *vk* dari neuron *k* dan output kombiner linier adalah dimodifikasi dalam perilaku yang digambarkan dalam gambar 2.3. Perhatikan bahwa hasil dari transaformasi ini, graph dari *vk* dibandingkan *uk* tidak lagi dilewatkan sebagaimana nilai aslinya.

Threshold data *k* adalah sebuah parameter ekternal dari artificial neuron *k*. dimana perhitungan dapat ditunjukkan dalam persamaan 1.2. Sebanding dengan formula yang dikombinasikan dalam persamaan 1.1 dan 1.2 sebagimana yang ditunjukkan :

$$v_k = \sum_{j=1}^{p} w_{kj} x_j \qquad (2.6)$$

$$y_k = \varphi (v_k)$$

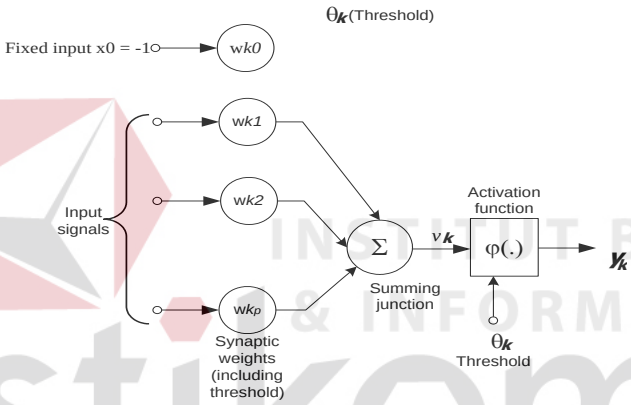
Dalam persamaan 1.4 yang telah ditambahkan synapsi baru, inputnya adalah

$$x0 = -1 \qquad (2.8)$$

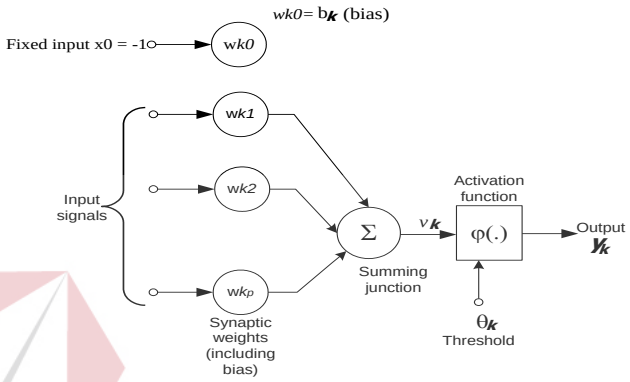
dan bobotnya adalah

$$wk0 = \theta k \qquad (2.9)$$

Pembuatan formulasi dari model neuron *k* sebagaimana yang ditunjukkan pada gambar 2.4a. Dalam gambar tersebut efek dari threshold ditunjukkan oleh perlakuan dua hal : (1) menambahkan input sinyal baru pada nilai –1, dan (2) menambahkan synapsi baru dengan bobot yang sama pada threshold θk. Jalan alternatif, yang dapat membuat model neuron sebagimana ditunjukkan dalam gambar 2.4b, dimana kombinasi konstan input adalah $x0 = +1$, dan bobot $wk0 = bk$ terhitung secara bias untuk *bk*. Meskipun model dari gambar 2.1 dan 2.4 adalah berbeda dalam tampilan, tetapi secara matematis adalah sama.



(a)



(b)

Gambar 2.4. Dua model nonlinier lain dari sebuah neuron

E. Tipe Fungsi Aktivasi

Fungsi aktivasi yang akan digunakan pada backpropagation harus mempunyai beberapa karakteriktik penting sebagai berikut : harus berkelanjutan, bersifat beda, dan bersifat mengurangi kejenuhan. Selanjutnya, untuk efisiensi perhitungan fungsi ini sangat perlu dikembangkan untuk memudahkan komputasi. Pada umumnya digunakan fungsi aktivasi dengan nilai yang dapat dikembangkan (nilai aktual dari variabel bebas) dalam kondisi nilai dari fungsi. Biasanya fungsi adalah diharapkan untuk *saturate* seperti pendekatan perhitungan maksimum dan minimum nilai asimtot.

Fungsi aktivasi yang disimbolkan dengan φ(.) mendifinisikan output dari neuron dalam periode pada level aktifitas di input tersebut.

Fungsi *Threshold*. Untuk jenis dari fungsi aktivasi ini digambarkan dalam gambar 2.5a yang memiliki persamaan

$$\varphi(v) = \begin{cases} 1 & \text{if } v \geq 0 \\ 2 & \text{if } v < 0 \end{cases} \quad (2.10)$$

dimana hubungan persamaan pada output neuron *k* sebagaimana fungsi threshold yang ditulis dalam bentuk

$$v_k = \begin{cases} 1 & \text{if } v_k \geq 0 \\ 2 & \text{if } v_k < 0 \end{cases} \quad (2.11)$$

dimana *vk* adalah level aktifitas internal dari neuron, yaitu :

$$v_k = \sum_{j=1}^{p} w_{kj} x_j - \theta_k \quad (2.12)$$

Sebuah neuron yang dihubungkan dalam literatur seperti model *McCulloch-Pitts*, dalam mengenali pekerjaan awal yang dilakukan McCulloch dan Pits (1943). Dalam model ini, output dari neuron mengambil nilai 1 jika total aktivitas internal dari neuron tersebut adalah non negatif dan 0 jika yang lain. Pernyataan ini menggambarkan semua properti atau tidak satupun dari model McCulloch-Pitts.

Fungsi linier *Piecewise*. Untuk fungsi ini dapat digambarkan dalam gambar 2.5b, sehingga menghasilkan persamaan :

$$\varphi(v) = \begin{cases} 1, & v = 0 \\ v, & -\frac{1}{2} > v > \frac{1}{2} \\ 0, & v = \frac{1}{2} \end{cases}$$

(2.13)

Dimana faktor amplifikasi didalam area operasi linier diasumsikan menjadi kesatuan. Fungsi aktivasi dalam sebuah form dapat dilihat sebagai aproksimasi pada sebuah amplier non linier. Dua situasi dibawah ini dapat dilihat sebagai form khusus dari fungsi linier piecewise :

Sebuah kombiner linier yang timbul jika operasi daerah linier diatur tanpa menjalankan satu rasi.

Fungsi linier Piecewise mengurangi fungsi threshold jika faktor amplifikasi dari daerah linier dibuat secara terbatas.

Fungsi sigmoid adalah bentuk paling umum dari fungsi aktivasi yang dipergunakan dalam konstruksi jaringan neural network. Hal itu didefinisikan sebagai sebuah fungsi peningkatan yang cukup nyata dengan menunjukkan kehalusan dan properti asymtot. Sebuah contoh dari sigmoid adalah fungsi logistik yang didefiniskan sebagai berikut :

$$f(x) = \frac{1}{1 + \exp(-ax)}$$

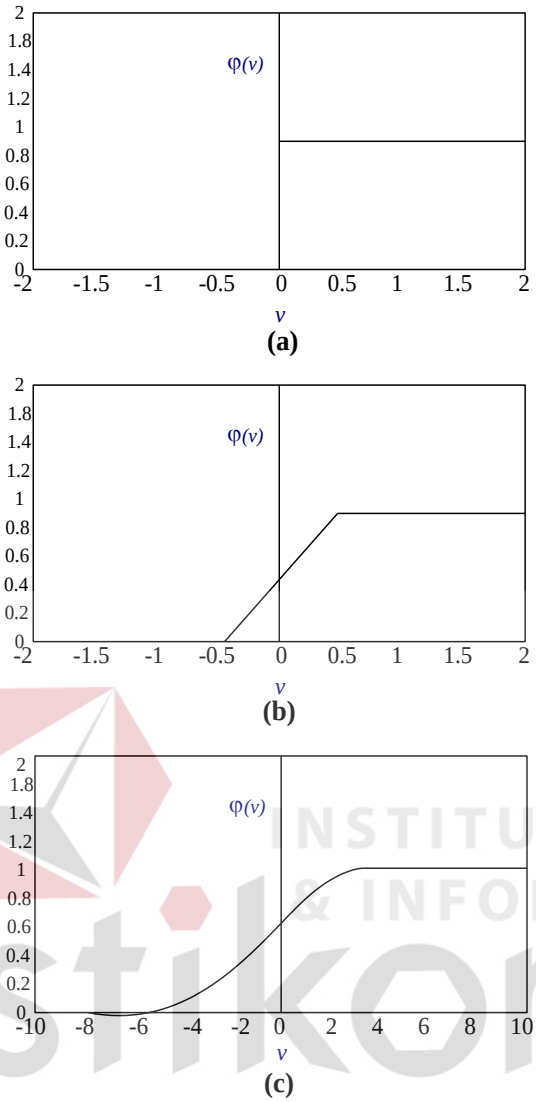
(2.14)

Dimana f mewakili fungsi aktivasi yang digunakan, x adalah variable input yang akan dimasukkan fungsi dan a adalah *slope paramater* dari fungsi sigmoid dengan melakukan variasi pada parameter a yang dapat melakukan fungsi sigmoid

dari slope yang berbeda, sebagaimana ditunjukkan dalam gambar 2.5c. Dalam kenyataan slope aslinya dibandingkan dengan *a*/4. Pada batasan, slope sebagai parameter mendekati ketidakterbatasan, fungsi sigmoid menyederhanakan fungsi *threshold*. Dimana fungsi *threshold* mengasumsikan nilai 0 atau 1, sebuah fungsi sigmoid mengasumsikan daerah nilai dari 0 sampai 1, dimana fungsi sigmoid juga mengasumsikan area yang kontinyu dari nilai 0 sampai 1. Perhatikan juga bahwa fungsi sigmoid dapat dibedakan, dimana fungsi threshold tidak dapat melakukannya. (Kemampuan ini dapat dibedakan sebagai sebuah fitur yang sangat penting dari teori neural network).

Fungsi aktivasi yang didefinisikan dalam persamaan (2.10), (2.13), dan (2.14) area dari 0 sampai +1. Terkadang terdapat keinginan untuk memiliki fungsi aktivasi dimana area dari –1 sampai +1, dalam hal ini fungsi aktivasi diasumsikan sebuah bentuk antisimetrik dengan memperhatikan nilai aslinya. Secara khusus, fungsi threshold dari persamaan 2.10 dapat didefiniskan sebagai berikut:

$$\varphi(v) = \begin{cases} 1 & \text{if } v > 0 \\ 0 & \text{if } v = 0 \\ -1 & \text{if } v < 0 \end{cases} \qquad (2.15)$$



Gambar 2.5. (a) Fungsi threshold.(b) fungsi Piecewise-linier.(c) fungsi sigmoid

Secara umum mengacu pada fungsi signum. Untuk sebuah sigmoid dimungkinkan menggunakan fungsi tangen hiperbolik, yang didefinisikan sebagai berikut :

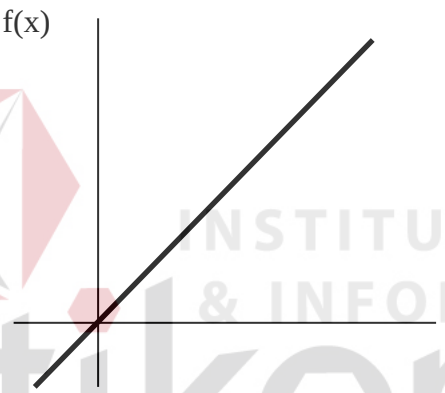
$$\varphi(v) = \tanh\left[\frac{v}{2}\right] = \frac{1 - \exp(-v)}{1 + \exp(-v)} \qquad (2.16)$$

Fungsi aktivasi dari tipe sigmoid diperbolehkan untuk mengasumsikan nilai negatif sebagaimana yang digambarkan persamaan 1.14 mempunyai keuntungan analitik. Lebih jauh, psikologi neuro membuktikan bahwa sifat eksperimental (Eekman dan Freema, 1986), secara kasar dengan antisimetri yang sempurna tentang keaslian karakteristik dari fungsi tangen hiperbolik.

F. Tipe fungsi Aktivasi Identitas

Fungsi identitas merupakan fungsi aktivasi untuk semua *input unit*.

f(x) = x untuk semua x. Bentuk fungsi identitas terdapat pada Gambar 2.6



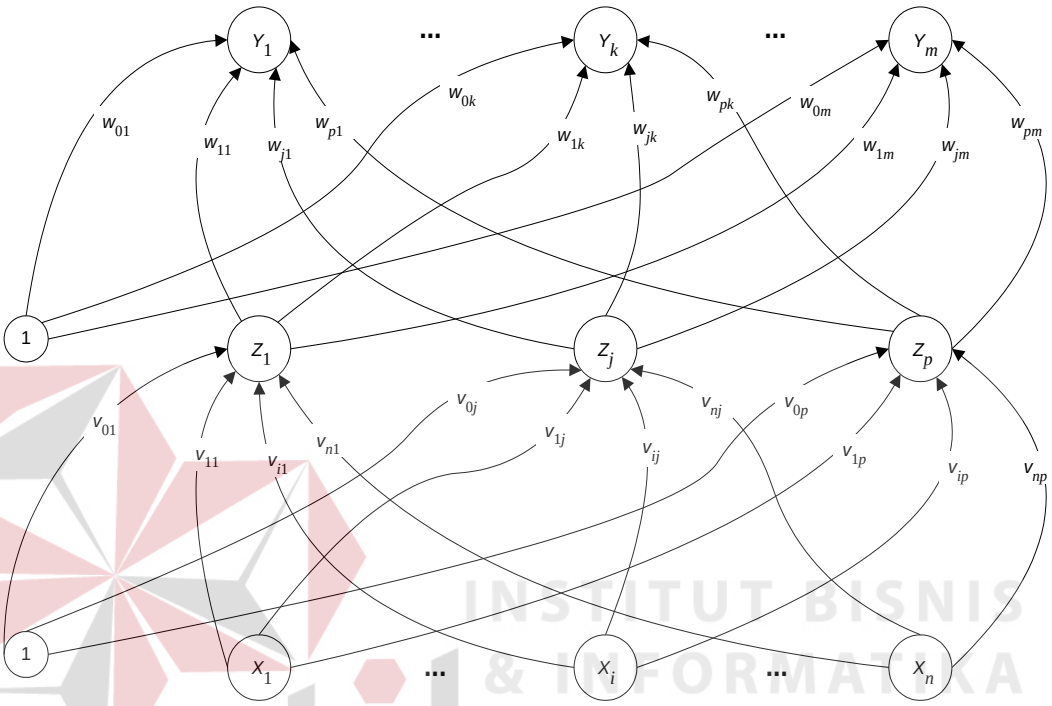
Gambar 2.6. Fungsi Identitas

2.1.4 *Backpropagation Neural Net*

A. Arsitektur

Jaringan neural multi layer dengan satu hidden layer (unit Z) ditunjukkan pada gambar 2.7 Unit output (unit *Y*) dan unit hidden juga mungkin mempunyai pembiasan (sebagaimana yang ditunjukkan pada gambar). Bias pada unit ouput *Yk* dinotasikan dengan *wok*; bias pada unit hidden Z*j* dinotasikan dengan *voj*. Bias ini berindak seperti bobot ada koneksi dari unit yang outpunya selalu satu. (Unit ini

ditunjukkan pada gambar 2.7 tapi biasanya tidak ditunjukkan secara eksplisit). Hanya arah dari alur informasi untuk fase operasi feedforward yang ditunjukkan. Selama fase pembelajaran, sinyal dikirim dalam arah berlawanan.



Gambar 2.7. Backpropagation neural network dengan satu hidden layer

B. Algoritma Backpropagation

Pelatihan network dengan *backpropagation* melibatkan tiga tahap : *feedforward* dari pola pelatihan input, backpropagation dari error yang terhubung, dan *penyesuaian* bobot.

Selama feedforward tiap unit input (*Xi*) menerima sebuah sinyal input dan menyebarkan sinyal ini pada tiap unit hidden *Z1, ..., Zp*. Tiap unit hidden kemudian mengkomputasi aktivasi (*Yk)* dan mengirimkan sinyal tersebut untuk

output unit. Tiap unit output mengkomputasi aktivasi dirinya pada bentuk respon dari jaringan yang memberikan pola input.

Selama pelatihan, tiap unit ouput membandingkan aktivasi komputasi *Yk* dengan nilai *target tk* untuk menentukan error yang berhubungan pada pola dengan unit tersebut. Berdasarkan error ini, faktor δk (k = 1,…, *m*) dilakukan komputasi. δk igunakan untuk mendistribusikan kesalahan pada unit output *Yk* kembali pada semua unit dalam layer sebelumnya (unit hidden yang berhubungan ke *Yk*). Hal ini juga digunakan untuk melakukan update bobot diantara output dan hidden layer. Dalam perilaku yang serupa faktor δj (j = 1,…, p) dikomputasi untuk tiap hidden unit Z*j*. Adalah tidak diperlukan untuk melakukan pembelajaran terhadap error pada layer input, tapi δj digunakan untuk melakukan update bobot diantara hidden layer dan input layer.

Setelah semua factor δ telah ditentukan bobot semua layer disesuaikan secara simultan. Penyesuaian terhadap bobot *wjk* (dari unti hidden Z*j* pada unit output *Yk*) didasarkan pada faktor δk dan aktivasi *zj* dari unit hidden Z*j*. Penyesuaian pada bobot *vij* (dari unit input X*i* pada unit hidden Z*j*) didasarkan pada factor δj dan aktivasi *xi* dari unit input. Tata aturan penulisan yang digunakan dalam algoritma pelatihan jaringan backpropagation mengikuti aturan :

x Input vektor pelatihan : x = (*x1,..., xi,... xn).*

t Output vaktor target : t = (*t1,..., ti,... tn).*

δk Bagian dari penyesuaian bobot koreksi error untuk *wjk* yang dikarenakan suatu error dari unit output *Yk* ; jadi informasi tentang error di unit *Yk* yang dikembalikan ke hidden unit itu dimasukkan kedalam unit *Yk*.

δj Bagian dari penyesuaian bobot koreksi error untuk v*ij* yang dikarenakan informasi yang salah backpropagation dari layer output ke unit hidde Z*j*.

α Rata-rata pembelajaran

Xi Unit input *i :* untuk sebuah unit input, sinyal input dan sinyal output adalah sama, penamaan, *xi*.

v*oj* Bias pada unit hidden *j*.

Zj Unit hidden *j :* Input net ke *Zj* dinotasikan dengan *z_inj* :

$z\_in_j = v_{oj} + \Sigma\, x_i\, v_{ij}$

Sinyal output (aktivasi) dari Z*j* dinotasikan dengan *zj* :

$zj = f(z\_in_j)$.

wok Bias pada unit output *k*.

Yk Unit output *k* : Input net ke *Yk* dinotasikan dengan *y_ink* :

$y\_in_k = w_{ok} + \Sigma\, z_i\, w_{jk}$

Sinyal output (aktivasi) dari *Yk* dinotasikan dengan *yk*:

$y_k = f(y\_in_k)$.

Secara lebih rinci algoritma backpropagation adalah sebagai berikut :

Step 0. Inisialisasi bobot.

(Set ke nilai random kecil).

Step 1. Ketika kondisi berhenti adalah salah, kerjakan langkah 2-9.

Step 2. Untuk tiap pasang pelatihan, kerjakan langkah 3-8.

Step 3. Tiap unit input (*Xi, i* = 1,…, *n*) menerima sinyal input *xi* dan meneruskan sinyal ini ke semua unit pada layer yang bersangkutan (unit hidden).

Step 4. Tiap unit hidden (*Z, j* = 1,…, *p*) dijumlahkan dengan bobot sinyal input.

$$z\_in_j = v_{oj} + \Sigma\, x_i\, v_{ij}$$

pergunakan fungsi aktivasi ini untuk menghitung sinyal output,

$$zj = f(z\_in_j),$$

dan mengirimkan sinyal ini ke seluruh unit pada layer yang bersangkutan (unit output).

Step 5. Tiap unit output (*Yk, k* = 1,…, *m*) dijumlahkan dengan bobot sinyal input,

$$y\_in_k = w_{ok} + \Sigma\, z_i\, w_{jk}$$

dan pergunakan fungsi aktivasi ini untuk menghitung sinyal output,

$$y_k = f(y\_in_k).$$

Error dari backpropagation :

Step 6. Tiap unit output (*Yk, k* = 1,…, *m*) menerima satu pola target yang cocok untuk pola pelatihan input, syarat perhitungan informasi error ini,

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

syarat penghitungan bobot koreksi ini (digunakan untuk meng-update w_{jk} nantinya),

$$\Delta w_{jk} = \alpha \delta_k\, zj,$$

syarat penghitungan bias koreksi ini (digunakan untuk meng-update w_{ok} nantinya),

$$\Delta w_{ok} = \alpha \delta_k,$$

dan mengirimkan δ_k ke unit pada layer sebelumnya.

Step 7. Tiap unit hidden (*Yk, k* = 1,…, *p*) menjumlahkan ke input delta (dari unit pada layer sesudahnya),

$$\delta\_in_j = \Sigma\ \delta_k\, w_{jk,}$$

dikalikan dengan nilai dari fungsi aktivasi untuk menghitungnya syarat informasi error,

$$\delta_j = \delta\_in_j\, f'(z\_in_j),$$

hitung koreksi bobot syaratnya (nanti digunakan untuk meng-update v_{ij}

$$\Delta v_{ij} = \alpha\delta_j\, x_i,$$

dan hitung koreksi bias syaratnya (nanti digunakan untuk meng-update v_{oj}),

$$\Delta v_{oj} = \alpha\delta_j.$$

Meng-upadate bobot dan bias :

Step 8. Tiap unit output (*Yk, k* = 1,…, *m*) update bias dan bobot(*j* = 0,.., *p*) :

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

Tiap unit hidden (*Zj, j* = 1,…, *p*) update bias dan bobot(*i* = 0,…, *n*):

$$v_{ij} \text{ (baru)} = v_{ij} \text{ (lama)} + \Delta v_{ij.}$$

Step 9. Kondisi pemberhentian test.

Setelah pelatihan, suatu jaringan neural digunakan hanya pada fase *feedforward* algoritma pelatihan. Prosedur aplikasinya meliputi antara lain :

Step 1. Inisialisasi bobot (dari algoritma pelatihan)

Step 2. Untuk tiap vektor input, kerjakan step 2-4.

Step 2. Untuk *i* = 1,…, *n*; set aktivasi dari unit input *xi;*

Step 3. Untuk j = 1,…, p :

$z\_in_j = v_{oj} + \Sigma\, x_i\, v_{ij;}$

$zj = f(z\_in_j),$

Step 4. Untuk k = 1,…, m :

$y\_in_k = w_{ok} + \Sigma\, z_j\, w_{jk;}$

$y_k = f(y\_in_k).$

2.1.5 *Mean Square Error (MSE)*

MSE adalah fungsi risiko, sesuai dengan nilai yang diharapkan dari hilangnya kesalahan squared atau kerugian kuadratik. Digunakanya MSE untuk membandingkan ketepatan perhitungan. MSE mengukur rata-rata kuadrat dari "kesalahan." Kesalahan adalah jumlah yang estimator yang berbeda dari jumlah yang akan diestimasi.

2.2 *Defence Of The Ancients Allstars* (DOTA)

Defense of the Ancients (atau disingkat DotA) adalah sebuah peta buatan (*custom map*) untuk permainan komputer buatan Blizzard berjudul Warcraft III : Frozen Throne, yang dibuat berdasarkan peta ”Aeon of Strife” dari game Blizzard lainya, Starcraft. DotA merupakan permainan strategi tim yang berpusat pada pertempuran antar *hero,* dimana setiap *hero* memiliki kemampuan yang berbeda-beda. Permainan ini berfokus pada strategi dan kerjasama kelompok, dan bisa dimainkan sampai 10 orang bersama, dan dibagi menjadi 2 kelompok yaitu;

Sentinel dan Scourge. Setiap pemain akan mendapat pilihan seorang hero dari jumlah total 96 hero tokoh pahlawan (Setiawan, 2009).

Tujuan utama permainan ini adalah untuk menghancurkan markas musuh bersama-sama dengant im dan anak buah petarung yang dikontrol oleh komputer (Creeps). Mirip dengan game permainan perang atau biasa disebut Role Playing Game (RPG), pemain dapat meningkatkan level tokoh pahlawan mereka dan membeli peralatan untuk memperkuat serangan dan pertahananya. DotA dikembangkan menggunakan World Editor dari game Warcraft III: Reign of Chaos. Blizzard sendiri mengakui akan popularitas DotA yang melebihi permainan Warcraft, sehingga DotA telah masuk dalam *hall of fame battlenet Blizzard.*

2.3 *Hero*

Hero adalah karakter di DotA yang dijalankan oleh pemain, dan setiap pemain hanya dapat menjalankan satu karakter saja. Total terakhir ada 96 karakter di versi DotA 66.7c dan bisa bertambah di setiap versi. Setiap karakter memiliki jenis, atribut kemampuan, kelebihan dan kekurangan masing-masing (Setiawan, 2009).

Terdapat 2 Jenis hero, berdasarkan jarak serangan yaitu *Melee* untuk hero yang memiliki jarak serang dekat dan *Ranged* untuk hero yang memiliki jarak serang menengah sampai jauh. Berdasarkan attributes yang paling besar pada hero dibagi menjadi tiga tipe yaitu, *Strength, Agility* dan *Intelligence*.

Secara garis besar hero memiliki tugas dan peran masing-masing (Setiawan, 2010), tugas tersebut antara lain :

1. *Tanker*

Seorang *Tanker* bertugas menerima kerusakan sebanyak-banyaknya dari musuh sebagai pengalih perhatian, sehingga *hero* yang lain dapat menjalankan tugasnya masing-masing.

2. *Ganker*

Hal yang penting dari *Ganker* adalah mobilitas, berkeliling di seluruh map dan membunuh *hero* musuh secara tiba-tiba adalah tugasnya.

3. *Initiator*

Initiator dalam DotA adalah seorang kunci pada tim, bertugas pertama membuka serangan dan memberikan kerusakan, supaya hero yang lain bisa masuk dan menjalankan tugasnya. Karena itu biasanya seorang *Initiator* memiliki jurus ultimate kuat.

4. *Pusher*

Tugas dari seorang *Pusher* adalah membersihkan pasukan musuh secara cepat dengan skill area yang dimilikinya.

5. *Disabler*

Disabler bertugas melumpuhkan musuh, memberikan gangguan, dan menghentikan musuh pada saat mengambil tindakan tertentu dalam sementara waktu.

6. *AOE Stunner*

Area of Effect Stunner atau disingkat AOE *Stunner* bertugas membuat kerusakan dan membuat korban tidak mampu bergerak atau melakukan tindakan dalam jangka waktu tertentu dengan skill area stun yang dimilikinya.

7. *AOE Nuker*

Area of Effect Nuker atau disingkat AOE Nuker tugasnya adalah memberikan damage banyak ke semua musuh yang ada dengan jurus yang dimilikinya. Seorang AOE nuker memiliki cooldown yang agak lama dalam menggunakan jurus yang dipakainya.

8. *Nuker*

Tugas dari seorang *Nuker* adalah memberikan damage sebanyak-banyaknya ke musuh secara cepat dengan jurus atau mantra sihit yang dimilikanya. Seorang nuker memiiliki cooldown waktu tertentu dalam menggunakan jurus atau mantra sihit yang dipakainya.

9. *Support*

Seorang supporter bertugas untuk memberikan status-status positif ke hero teman dan memberikan status-status negatif ke hero musuh.

10. Semi Carry

Semi Carry adalah hero yang memberikan kerusakan secara fisik atau dengan skill yang dimilikinya. Akan tetapi tidak sehebat Carry apabila jadi itemnya, maka dari itu semi carry masi diwajibkan untuk mengikuti pertempuran pada awal permainan.

11. Carry

Seorang *Carry* atau biasa disebut dengan *hitter* adalah hero yang memberikan kerusakan secara fisik dengan natural skillnya atau dengan memperbanyak item di inventori. Kebanyakan hero carry lah yang akan

membawa banyak kemenangan pada tim nya kalau tidak dijaga dan jadi itemnya.

12. Stunner

Tugas dari seorang *Stunner* adalah membuat kerusakan dan membuat korban tidak mampu bergerak atau melakukan tindakan dalam jangka waktu tertentu dengan skill yang dimilikinya. Lama durasi stun sekitar 1 sampai 2 detik.

13. Slower

Slower adalah hero yang memiliki jurus melambatkan jalannya musuh untuk lebih mudah diserang oleh hero yang lain.

2.4 *Captains Mode* (CM)

Captains mode adalah salah satu mode dari sekian banyak mode di DotA, mode ini adalah mode yang sering digunakan pada saat pertandingan resmi dan perlombaan. Pada *Captains mode* posisi Biru dan Merah muda adalah kapten tim. Tim yang pertama mulai memilih hero yang tidak boleh digunakan pada saat eliminasi secara bergantian 1/1/1. Waktu mengeliminasi 40 detik, apabila waktu habis maka akan dianggap tidak mengeliminasi hero apapun. Setelah itu baru kapten tim memilih hero yang akan digunakan secara bergantian 1/2/2/2/2/1. Waktu memilih 60 detik, apabila waktu habis maka akan memasuki ekstra time 45 detik dan apabila masih belum memilih maka akan mendapatkan random hero. Terdapat total 7 fase pemilihan hero pada Dota CM. Segera ketik –CM setelah game berlangsung untuk masuk ke mode Captains mode, apabila kubu Sentinel yang memilih pertama maka ketik –CM 1 setelahnya, sebaliknya kubu Scourge yang memilih pertama maka ketik –CM 2. Hero yang dipilih, di eliminasi, dan

waktu semuanya ada di papan sebelah kanan atas untuk memudahkan melihat. (Setiawan, 2009).