BAB 10
Perceptron dan ANN (Artificial Neural Network)

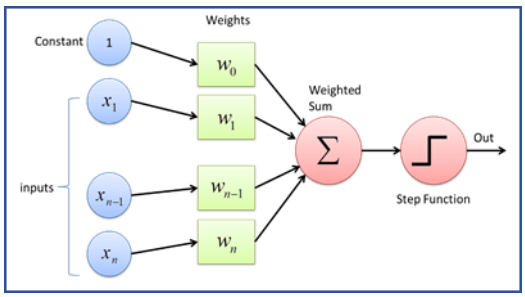
10.1 Pengertian Perceptron

Algoritma Perceptron adalah jenis Artificial Neural Network yang paling sederhana. Nama lain percepton adalah Single-Layer Neural Network, FeedForward Network. Perceptron merupakan model neuron tunggal yang dapat digunakan untuk binary classification dan juga dasar untuk mengembangkan "network" yang jauh lebih besar. Perceptron biasanya digunakan untuk mengklasifikasikan suatu tipe pola tertentu yang sering dikenal dengan pemisahan secara linear. Pada dasarnya, perceptron pada jaringan syaraf dengan satu lapisan memiliki bobot yang bisa diatur dan suatu nilai ambang (threshold).

Perceptron pertama kali diciptakan oleh Frank Rosenblatt pada akhir tahun 1950-an. Hal yang mendasari penciptaan perceptron adalah mengadopsi kemampuan otak manusia. Misalkan dalam proses penglihatan, maka apa yang kita lihat adalah data. Data itu kemudian diproses oleh saraf dan dikirim ke otak. Otak kemudian mengenali data tersebut berdasarkan karakteristik hal-hal yang terlihat. Otak terdiri dari sel-sel yang disebut *neuron* yang berfungsi untuk memproses informasi, Koneksi antar neuron disebut *sinapsis*, dimana sinapsis berfungsi untuk melakukan transmisi informasi. Otak manusia diperkirakan terdiri dari 100 miliar *neuron* dan 100 triliun *sinapsis*. Komponen utama neuron adalah:

- Dendrit ~ menerima sinyal listrik dari neuron lain
- Body ~ memproses sinyal
- Akson ~ mengirimkan sinyal ke neuron lain

Pada algoritma perceptron komponen utaman neuron diadopsi seperti pada gambar 10.1. Neuron individu dapat dianggap sebagai unit komputasi yang memproses satu atau lebih input untuk menghasilkan output. Fungsi perceptron analog dengan neuron. Perceptron menerima satu atau lebih data masukan, kemudian memprosesnya, dan mengembalikan output.



Gambar 10. 1 Ilustrasi Perceptron

Perceptron bisa dibagi menjadi empat bagian dasar: input, bobot dan bias, *Weighted Sum*, dan fungsi aktivasi. Input adalah data masukan dari perceptron tersebut. Bobot dan bias merepresentasikan tingkat signifikansi dari tiap input. Input yang telah dikalikan bobot dan ditambah bias dijumlah menjadi *Weighted Sum*. Hasilnya akan menjadi parameter dari fungsi aktivasi. Fungsi aktivasi ini yang mementukan hasil output dari perceptron tersebut.

Input

Input dari sebuah perceptron adalah bahan baku yang akan diolah perceptron..

Bobot dan Bias

Pada sebuah Perceptron, tidak semua input memiliki signifikansi yang sama. Karena itu, input diberi bobot dan bias yang berbeda untuk menentukan seberapa penting input ini berpengaruh terhadap output yang diinginkan. Bobot akan diupdate secara otomatis melalui banyak algoritma pembelajaran pada saat proses pelatihan.

Weighted Sum

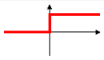
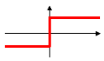
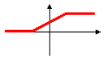
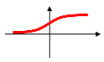
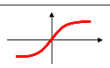
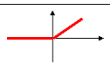
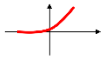
Masing-masing input akan dikalikan bobot masing-masing dan ditambah bias. Hasilnya akan dijumlah semua menjadi *weighted sum*. Angka ini nantinya akan menjadi parameter untuk fungsi aktivasi. Formulanya adalah seperti ini

$$\sum_{n=1}^{n}(x_n w_n) + b$$

Dengan x_n adalah input ke-n yang memiliki bobot w_n dan bias b.

Fungsi aktivasi

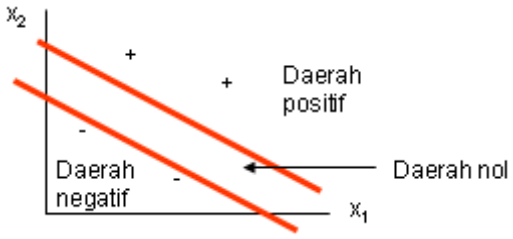
Hasil dari *Weighted Sum* akan menjadi parameter dari sebuah fungsi aktifasi. Fungsi inilah yang menentukan hasil dari perceptron tersebut. Output dari fungsi ini bisa menjadi input dari perceptron lain atau menjadi output dari ANN secara keseluruhan. Perceptron mengklasifikasikan instance dengan memproses kombinasi linier fitur dan parameter model menggunakan fungsi aktivasi. Ada banyak jenis fungsi aktivasi, masing-masing memiliki kegunaan dan fungsi sendiri-sendiri.

Activation function	Equation	Example	1D Graph
Unit step (Heaviside)	$\phi(z) = \begin{cases} 0, & z < 0, \\ 0.5, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Sign (Signum)	$\phi(z) = \begin{cases} -1, & z < 0, \\ 0, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Linear	$\phi(z) = z$	Adaline, linear regression	
Piece-wise linear	$\phi(z) = \begin{cases} 1, & z \geq \frac{1}{2}, \\ z + \frac{1}{2}, & -\frac{1}{2} < z < \frac{1}{2}, \\ 0, & z \leq -\frac{1}{2}, \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multi-layer NN	
Hyperbolic tangent	$\phi(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$	Multi-layer Neural Networks	
Rectifier, ReLU (Rectified Linear Unit)	$\phi(z) = max(0, z)$	Multi-layer Neural Networks	
Rectifier, softplus	$\phi(z) = \ln(1 + e^{z})$	Multi-layer Neural Networks	

Gambar 10. 2 Jenis-jenis fungsi aktivasi

Sumber : http://rasbt.github.io/mlxtend/user_guide/general_concepts/activation-functions/

Fungsi aktivasi dibuat sedemikian rupa sehingga terjadi pembatasan antara daerah positif dan negatif



- Garis pemisah antara daerah positif dan daerah nol memiliki :

w1x1 + w2x2 + b > q

- Sedangkan garis pemisah antara daerah negatif dengan daerah nol memiliki :

w1x1 + w2x2 + b < -q

10.2 Algoritma Perceptron

Algoritma perceptron dimulai dengan menetapkan bobot awal ke nol, atau ke nilai acak kecil. Kemudian memprediksi kelas untuk instance data training. Perceptron adalah algoritma pembelajaran yang digerakkan oleh error: Jika prediksinya benar, algoritma melanjutkan ke instance berikutnya. Jika prediksi salah, algoritma akan memperbarui bobot.

Tahapan algoritma percepton

Input : sekumpulan data dengan fitur / variable x dan variable y (x1,y1) ,(x2,y2)….(xn,yn)

Output : sebuah perceptron didefinisikan oleh (w0,w1,w2….wd)

1. Menentukan bobot, seringkali bobot(w) diberikan nilai awal = 0. Tentukan juga nilai awal bias (b) biasanya diberi nilai =1
2. Set learning rate: $\boldsymbol{\alpha}$ = 1, ($0 < \boldsymbol{\alpha} \leq 1$)
3. Selama masih terdapat kondisi false dalam artian belum tercapainya convergen, lakukan langkah-langkah sebagai berikut :
    a. Untuk setiap sampel x_i semua i anggota (1….n)
    b. Hitung respon untuk unit output dengan menggunakan *weighted sum* dan fungsi aktivasi
    c. Bandingkan hasil output(y) dengan nilai actual kelas(d), jika output tidak sesuai dengan nilai actual kelas maka dianggap error. Perbaiki bobot jika terjadi error:

    Bobot baru :

    $$\boldsymbol{w_i(t+1) = w_i(t) + \alpha\left(d_j - y_j(t)\right)x_{j,i}}$$ untuk semua fitur $0 \leq i \leq n$

    Dimana

    d_j : Actual kelas untuk data instance j

    y_j : Kelas hasil prediksi untuk data instance *j*

$x_{j,i}$: nilai fitur ke i untuk data instance j
α : a adalah hyperparameter yang mengontrol kecepatan pemelajaran
$\alpha\left(d_j - y_j(t)\right)x_{j,i}$: faktor kenaikan nilai parameter untuk setiap fitur.

Kembali ke Langkah a

4. Algoritma pembelajaran telah convergen ketika menyelesaikan 1kali epoch tanpa adanya kesalahan output prediksi.

10.3 Contoh sederhana perhitungan perceptron

Dataset yang digunakan adalah dataset untuk permasalahan klasifikasi. Misalkan ingin memisahkan kucing dewasa (adult) dari anak kucing(kitten). Diberikan dua variabel penjelas (x) yang tersedia dalam kumpulan data: proporsi hari saat hewan tertidur dan proporsi hari saat hewan itu menggerutu. Data pelatihan terdiri dari empat instances berikut:

Tabel 10. 1 Dataset perceptron

Instance	Proportion of the day spent sleeping	Proportion of the day spent being grumpy	Kitten or adult?
1	*0.2*	*0.1*	*Kitten*
2	*0.4*	*0.6*	*Kitten*
3	*0.5*	*0.2*	*Kitten*
4	*0.7*	*0.9*	*Adult*

Berdasarkan tabel 10.1 maka kelas kitten akan menjadi kelas positif(1) dan kelas adult akan menjadi kelas negative(0). Algoritma perceptron pada kasus ini memiliki tiga buah input yaitu X1 untuk bias, x2 untuk fitur proportion of the day spent sleeping dan x3 untuk fitur proportion of the day being grumpy. Fungsi aktivasi yang digunakan pada contoh ini adalah unit step yaitu dengan aturan sebagai berikut :

$$g(x) = \begin{cases} 1, & \text{if } x > 0 \\ 0, & \text{elsewhere} \end{cases}$$

Epoch 1
bobot(w) diberikan nilai awal = 0, bias =1, dan learning rate: $\boldsymbol{\alpha}$ = 1

Tabel 10. 2 Pelatihan pada epoch 1

iterasi	W	x1,x2,x3	weighted sum	Prediction	Target/actual	is correct?	updated weight
1	0, 0, 0;	1.0, 0.2, 0.1;	(1*0)+(0,2*0)+(0,1*0)=0	0	1	FALSE	new W1= 0+1(1-0)*1=1; new W2= 0+1(1-0)*0,2=0,2 ; new W3= 0+1(1-0)*0,1=0,1 ;
2	1;0.2;0.1	1.0, 0.4, 0.6;	1,14	1	1	TRUE	no update
3	1;0.2;0.1	1.0, 0.5 , 0.2	1,12	1	1	TRUE	no update

4	1;0.2;0.1	1.0, 0.7 , 0.9	1,23	1	0	FALSE	new W1= 1+1(0-1)*1=0;
							new W2= 0,2+1(0-1)*0,7=-0,5;
							new W3= 0,1+1(0-1)*0,9=-0,8;

Pada epoch 1 terjadi dua kali perbaikan bobot yaitu pada iterasi 1 dan pada iterasi 4. Perbaikan bobot terjadi karena adanya misclassification yaitu prediksi klasifikasi yang tidak sesuai dengan actual data / variable target. Pada epoch 1 Algoritma pembelajaran belum mencapai convergen sehingga dilanjutkan ke epoch 2.
Epoch 2

Tabel 10. 3 Pelatihan pada epoch 2

iterasi	W	x1,x2,x3	weighted sum	Prediction	Target / actual	is correct?	updated weight
1	0;-0.5;-0,8	1.0, 0.2, 0.1;	-0,18	0	1	FALSE	1;-0,3;-0,7
2	1;-0,3;-0,7	1.0, 0.4, 0.6;	0,46	1	1	TRUE	no update
3	1;-0,3;-0,7	1.0, 0.5 , 0.2	0,71	1	1	TRUE	no update
4	1;-0,3;-0,7	1.0, 0.7 , 0.9	0,16	1	0	FALSE	0;-1;-1,6

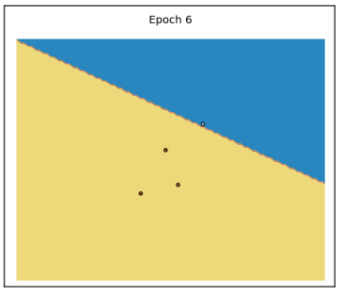
Pada epoch 2terjadi dua kali perbaikan bobot yaitu pada iterasi 1 dan pada iterasi 4. Perbaikan bobot terjadi karena adanya misclassification yaitu prediksi klasifikasi yang tidak sesuai dengan actual data / variable target. Pada epoch 2 Algoritma pembelajaran belum mencapai convergen sehingga dilanjutkan ke epoch 3.

Lanjutkan pelatihan sampai mencapai convergen yaitu dimana pada keseluruhan data pelatihan / data training hasil prediksi memiliki nilai yang sama dengan actual data / variable target. Pada contoh kasus diatas convergen akan dicapai pada epoch 6. Hasil yang didapatkan ditunjukkan pada tabel 10.4

Tabel 10. 4 Pelatihan pada epoch 6

Instance	Initial Weights; x; Activation	Prediction, target	Is correct?	Updated weights
0	*2, -1, -1.5* *1.0, 0.2, 0.1* *1.0*2 + 0.2*-1 + 0.1*-1.5 = 1.65*	*1, 1*	*True*	*2, -1, -1.5*
1	*2, -1, -1.5* *1.0, 0.4, 0.6* *1.0*2 + 0.4*-1 + 0.6*-1.5 = 0.70*	*1, 1*	*True*	*2, -1, -1.5*
2	*2, -1, -1.5* *1.0, 0.5, 0.2* *1.0*2 + 0.5*-1 + 0.2*-1.5 = 1.2*	*1, 1*	*True*	*2, -1, -1.5*
3	*2, -1, -1.5* *1.0, 0.7, 0.9* *1.0*2 + 0.7*-1 + 0.9*-1.5 = -0.05*	*0, 0*	*True*	*2, -1, -1.5*

Visualisasi decision boundary mulai dari epoch dapat dilihat pada gambar berikut ini.



Gambar 10. 3 decision boundary pada epoch 6

Klasifikasi dokumen menggunakan algoritma perceptron.

```
from sklearn.datasets import fetch_20newsgroups
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import Perceptron
from sklearn.metrics import f1_score, classification_report

categories = ['rec.sport.hockey', 'rec.sport.baseball', 'rec.autos']
newsgroups_train = fetch_20newsgroups(subset='train', categories=categories, remove=('headers', 'footers', 'quotes'))
newsgroups_test = fetch_20newsgroups(subset='test', categories=categories, remove=('headers', 'footers', 'quotes'))

vectorizer = TfidfVectorizer()
X_train = vectorizer.fit_transform(newsgroups_train.data)
X_test = vectorizer.transform(newsgroups_test.data)
clf = Perceptron(random_state=11)
clf.fit(X_train, newsgroups_train.target )
predictions = clf.predict(X_test)
print(classification_report(newsgroups_test.target, predictions))
```

Output :

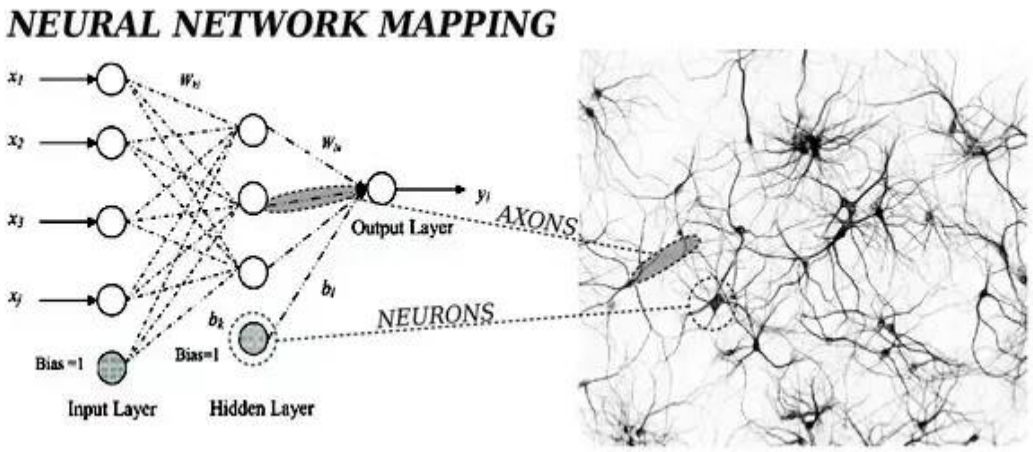
```
Downloading 20news dataset. This may take a few minutes.
Downloading dataset from https://ndownloader.figshare.com/files/5975967 (14 MB)
              precision    recall  f1-score   support

           0       0.88      0.88      0.88       396
           1       0.82      0.83      0.83       397
           2       0.88      0.87      0.87       399

    accuracy                           0.86      1192
   macro avg       0.86      0.86      0.86      1192
weighted avg       0.86      0.86      0.86      1192
```

Dataset yang digunakan pada kode program diatas adalah 20newsgroup yang terdiri dari sekitar 20.000 dokumen. Scikit-learn bahkan menyediakan fungsi yang memberikan kemudahan untuk mengunduh dan membaca kumpulan dataset dengan menggunakan sklearn.datasets. pada kode program diatas Perceptron mampu melakukan klasifikasi multikelas; strategi yang digunakan adalah one-versus-all untuk melakukan pelatihan untuk setiap kelas dalam data training. Dokumen teks memerlukan ekstraksi fitur salah satunya adalah bobot tf-idf pada kodeprogram diatas digunakan tfidf-vectorizer.

10.4 Artificial Neural Network (ANN

Jika perceptron dianalogikan dengan neuron, maka pada ANN atau jaringan saraf tiruan(JST) dapat dianalogikan sebagai otak yang terdiri dari kumpulan neuron. Pada otak manusia terdiri dari milyaran neuron dengan triliunan sinapsis, pada ANN direpesrentasikan dengan graph terarah dari neuron-neuron buatan. Edged ari graph diberi nilai bobot(w) ; bobot tersebut merupakan parameter model yang harus

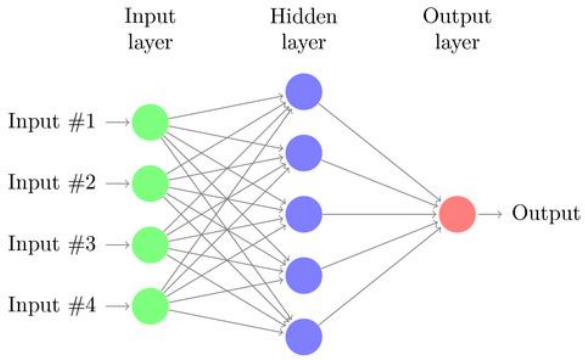
dipelajari. Pada gambar 10.4 merupakan pemetaan neural network dari otak manusia menjadi neural network yang ada pada machine learning. setiap neuron dinotasikan dengan node dan akson dinotasikan dengan edge.



Gambar 10. 4 Gambaran Pemetaan Neural Network

10.5 Multi-Layer Perceptron

Multi-layer perceptron(MLP) merupakan salah satu algoritma Artificial Neural Network yang memiliki beberapa layer dari neuron yang menyerupai perceptron. Pada MLP memiliki setidaknya 3 layer ataupun lebih dengan setiap layer akan terhubung dengan layer berikutnya. Layer minimal yang ada pada MLP adalah 3 layer yaitu input layer, satu buah hidden layer, dan output layer. Setiap lapisan atau layer memiliki banyak perceptron yang menerima input dari lapisan sebelumnya dan mengeluarkan output aktivasi untuk lapisan selanjutnya. Struktur MLP dengan satu hidden layer dapat dilihat pada gambar 10.5



Gambar 10. 5 Struktur MLP

Input Layer

Lapisan ini adalah lapisan yang menerima input langsung dari data yang dimasukkan. Jumlah node input yang ada pada input layer tegantung pada banyaknya fitur yang

digunakan untuk pelatihan. Output dari lapisan input ini akan menjadi input dari hidden layer.

Hidden Layer

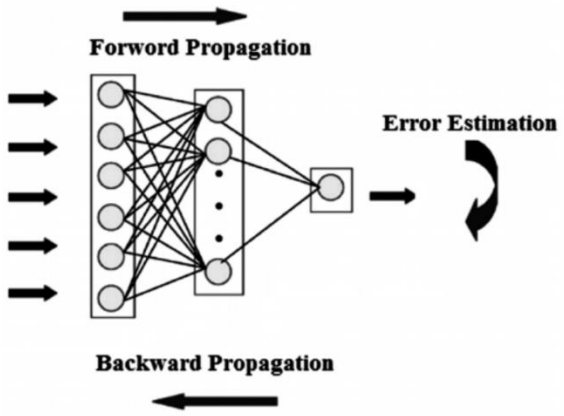
Hidden Layer merepresentasikan variabel laten; layer ini adalah layer yang memproses input sebelum diteruskan ke layer output. Semakin hidden layer, umumnya akurasi dari ANN juga semakin bagus, namun hal ini juga dapat menyembabkan peningkatan biaya komputasi. Hidden layer pada lapisan terakhir akan terhubung dengan output.

Output Layer

Layer ini adalah layer yang akan menjadi luaran dari MLP yang dibuat. Perceptron pada lapisan ini menerima input dari hidden layer. Jumlah perceptron dari lapisan ini bergantung pada alternatif output yang diinginkan, diubah dalam bentuk biner. Misal jika hanya prediksi benar salah, maka hanya diperlukan satu perceptron pada lapisan output yang menghasilkan luaran 0 dan 1.

10.6 Pelatihan pada ANN

Setelah membuat struktur dari ANN baik berupa perceptron maupun Multi-layer Perceptron, proses selanjutnya yang dilakukan adalah melakukan pelatihan. ANN perlu dilatih agar dapat menghasilkan output yang diinginkan. Proses pelatihan ini ada dua tahap yang dilakukan secara berulang-ulang sampai menghasilkan ANN yang diinginkan. Dua tahapan ini adalah **forward propagation (propagasi maju)** dan **backward propagation (propagasi mundur)**. Ilustrasi proses pelatihan JST bisa dilihat pada gambar berikut.



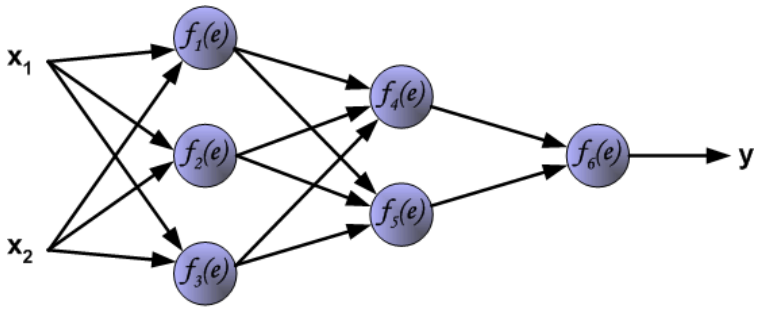
Gambar 10. 6 Pelatihan ANN
(Sumber : https://medium.com/@purnasaigudikandula/a-beginner-intro-to-neural-networks-543267bda3c8)

ANN yang saat ini banyak digunakan adalah gabungan dari forward propagation dan backward propagation hal ini didasari karena jika hanya menggunakan forward propagation akan sangat bergantung pada inisialisasi awal pemberian nilai pada bobot.

Permasalahan biasanya muncul Ketika data training yang mempunyai banyak fitur yang artinyya node input juga banyak hal ini akan membuat banyanya bobot yang harus di inisialisai. Sehingga pada prakteknya saat ini banyak digunakan gabungan gabungan iterasi berulang dari forward propagation dan backward propagation yang biasa disebut dengan backpropragation. Backpropagation biasanya digunakan bersamaan dengan algoritma optimasi seperti gradient descent.

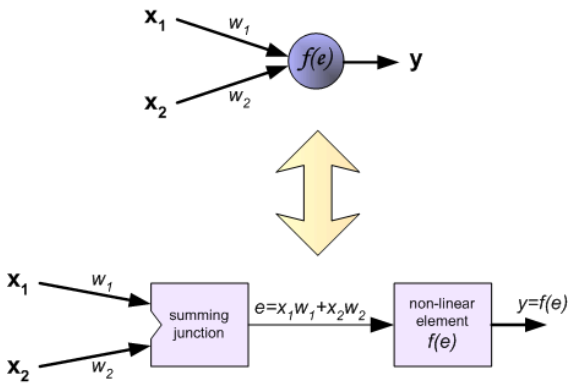
Ilustrasi **Backpropagation**

Untuk mengambarkan bagaimana backpropagation digunakan struktur MLP dengan 3 layer seperti pada gambar berikut :



Gambar 10. 7 Struktur MLP 3 layer

Setiap edge yang menghubungkan baik input ke hidden layer ataupun hidden layer ke output memiliki bobotnya masing-masing. Ilustrasi bobot :

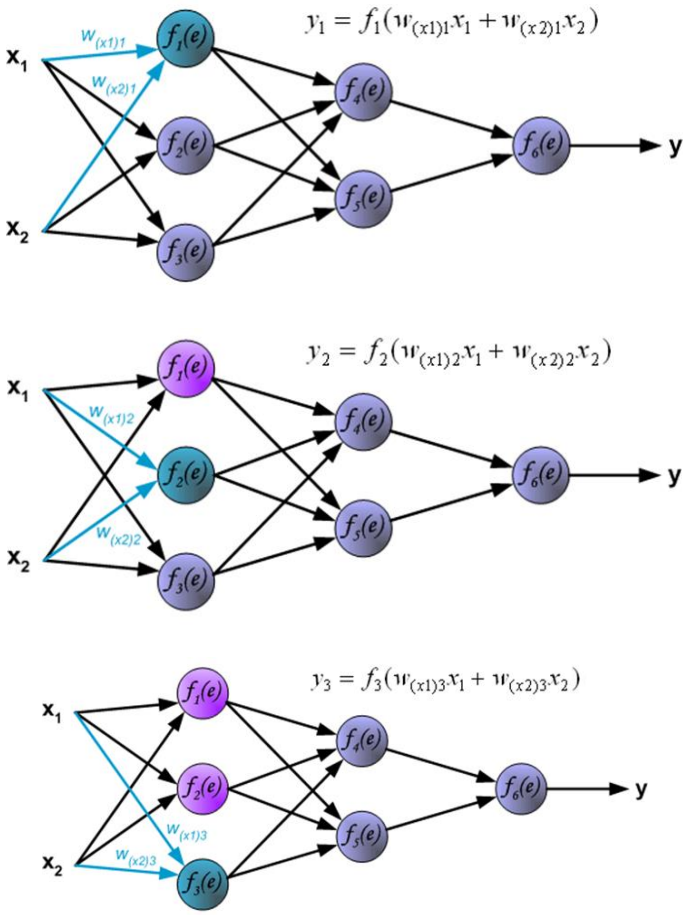


Pada gambar 10.7 dapat dilihat bahwa data inputan memiliki dua buah fitur yaitu X_1, dan X_2.

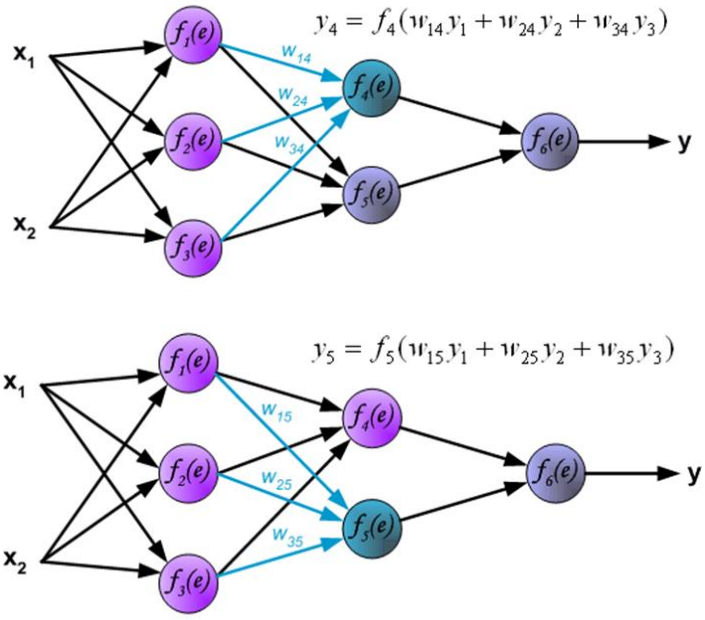
Proses **Forward propagation**

Gambar dibawah ini merupakan ilustrasi bagaimana proses maju dari input ke output melalui network, Simbol w (xm) n mewakili bobot koneksi antara input jaringan xm dan neuron n pada lapisan input. Simbol yn merepresentasikan sinyal keluaran neuron n.

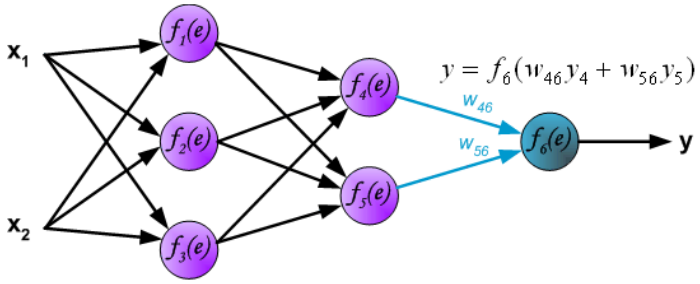
Pada hidden layer 1



Pada hidden layer 2



Pada output Layer

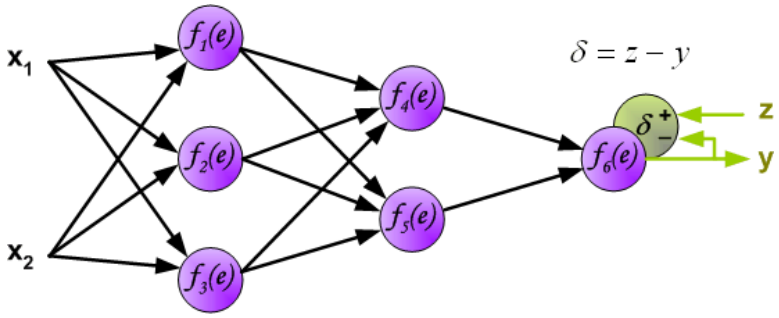


Proses **Backward propagation**

Gambar dibawah ini merupakan ilustrasi bagaimana proses mundur dari output menuju ke input melalui network.

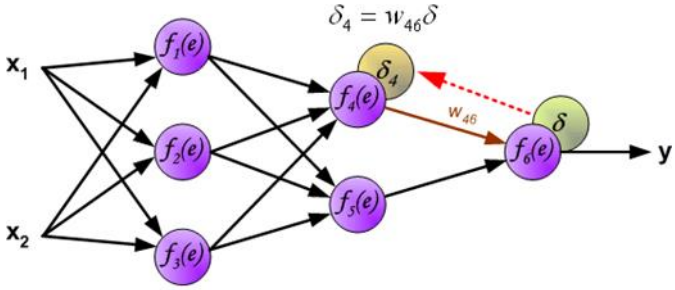
Pada langkah algoritma berikutnya, keluaran jaringan y dibandingkan dengan nilai aktual yang diinginkan (target), nilai target didapatkan dari kumpulan data training. Perbedaan antara tagert dengan output tersebut disebut error (δ) dari neuron lapisan output. Error juga dapat dicari dengan menggunakan squared error cost function (δ) dengan persamaan :

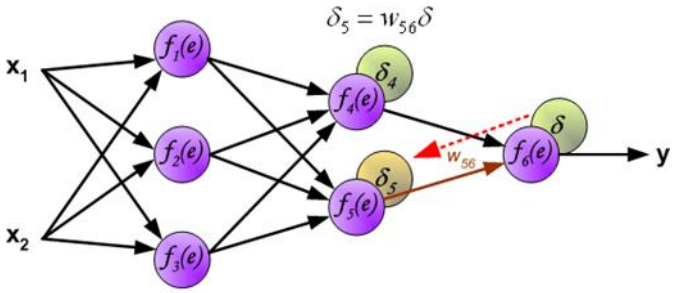
$$E = \frac{1}{2}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$



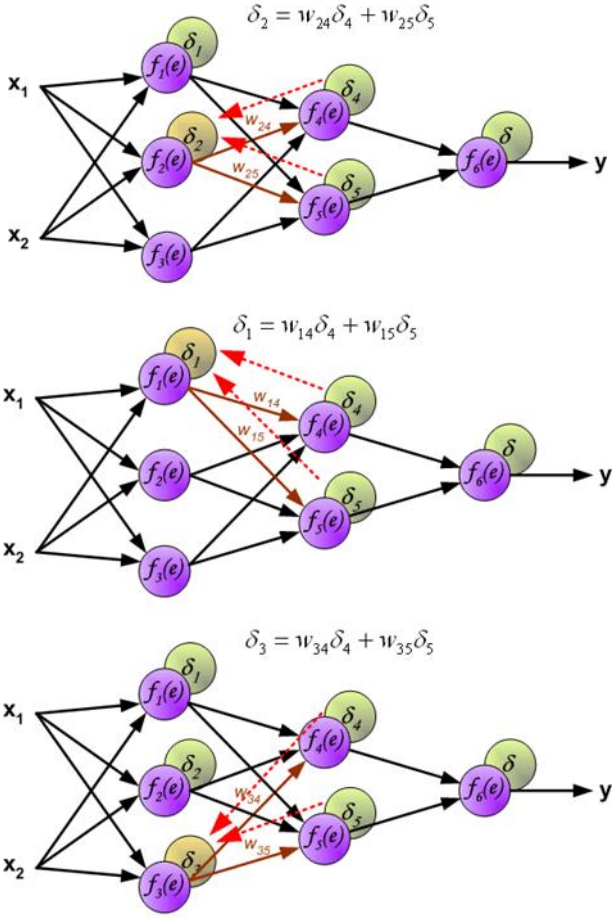
Nilai bobot yang digunakn untuk menghitung setiap node pada hidden layer memiliki nilai yang sama pada saat forward propagation, yang membedakan hanyalah arah perhitungannya dari output menjuju input dan juga bobot yang pada forward propagation dikalikan dengan nilai node setiap layer, pada backward propagation bobot dikalikan dengan error.

Pada hidden layer 2





Pada hidden layer 1



Update bobot

Ketika error untuk setiap neuron/node telah dihitung, maka bobot dari setiap input node dapat dilakukan modifikasi/perubahan. Dengan menggunakan persamaan

$$w'_{xi} = w_{xi} + \alpha\delta_i \frac{df(e)}{de} x_i$$

Dengan

w'_{xi} = bobot baru

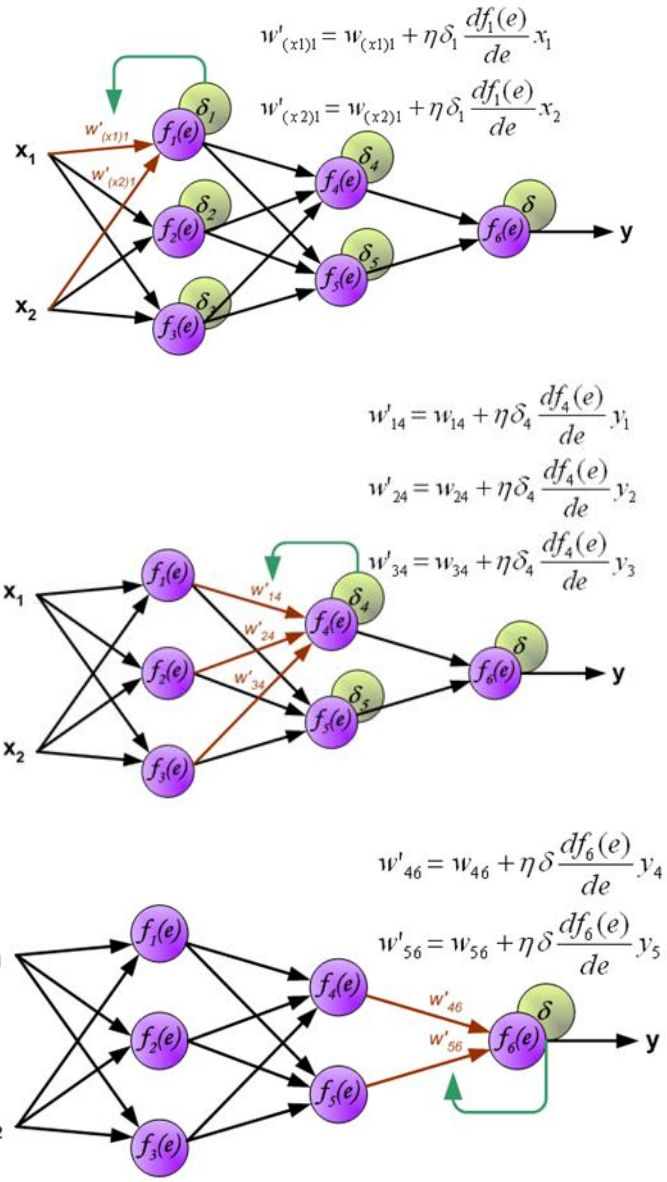
w_{xi} = bobot lama

α $atau$ η = learning rate: $\boldsymbol{\alpha}$ = 1, (0 < $\boldsymbol{\alpha}$ ≤ 1)

x_i = nilai input pada node x

$\frac{df(e)}{de}$ = Derivative activation function yang dapat dicari dengan persamaan berikut

$$\frac{d}{dx} f(x) = f(x)(1 - f(x))$$



10.7 MLP untuk permasalahan XOR

```
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier

y = [0, 1, 1, 0]
X = [[0, 0], [0, 1], [1, 0], [1, 1]]

clf = MLPClassifier(solver='lbfgs', activation='logistic', hidden_layer_sizes=(2,), max_iter=100, random_state=20)
clf.fit(X, y)

predictions = clf.predict(X)
print('Accuracy: %s' % clf.score(X, y))
for i, p in enumerate(predictions[:10]):
    print('True: %s, Predicted: %s' % (y[i], p))
```

```
Accuracy: 1.0
True: 0, Predicted: 0
True: 1, Predicted: 1
True: 1, Predicted: 1
True: 0, Predicted: 0
```