

Jaringan Saraf Tiruan (Artificial Neural Networks)

Ali Akbar Septiandri

August 4, 2017

Universitas Al Azhar Indonesia

Table of contents

1. Ulasan Regresi Linear
2. Ulasan Regresi Logistik
3. Jaringan Saraf Tiruan (JST)
4. Variasi JST
5. Aplikasi JST

Ulasan Regresi Linear

Regresi linear

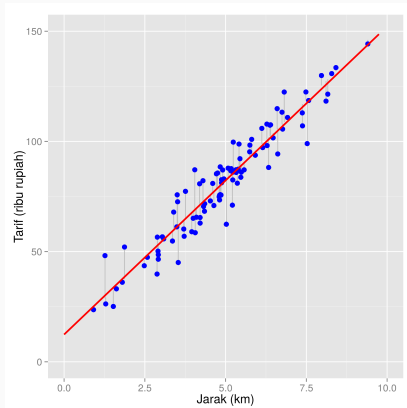


Figure 1: Regresi linear tarif taksi dari jarak tempuh

Regresi Linear Satu Dimensi

$$y = w_0 + w_1 x_1$$

Regresi Linear Multidimensi

$$y = w_0 + w_1x_1 + \dots + w_Dx_D = \sum_{j=0}^D w_jx_j$$

Regresi Linear Multidimensi (notasi vektor)

$$y = \mathbf{w}^T \mathbf{x}$$

- Fungsi error

$$E(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^N (y_i - \mathbf{w} \cdot \phi(x_i))^2$$

- Solusi tertutupnya:

$$\hat{\mathbf{w}} = (\phi^T \phi)^{-1} \phi^T \mathbf{y}$$

- Bagian $(\phi^T \phi)^{-1} \phi^T$ dikenal sebagai *pseudo-inverse*

Ulasan Regresi Logistik

Regresi logistik

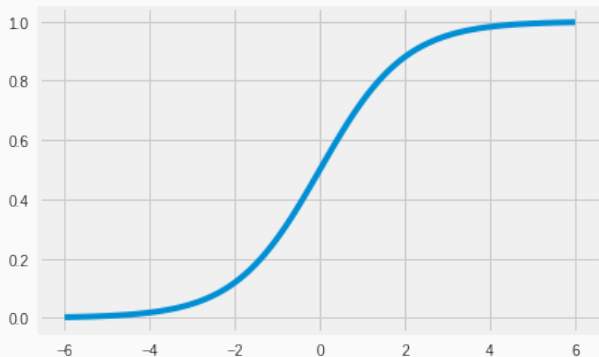


Figure 2: Fungsi logistik $\sigma(z) = \frac{1}{1 + \exp(-z)}$

Probabilitas kelas dengan fungsi logistik

- Fungsi logistik (sigmoid) mengubah nilai z dari $(-\infty, \infty)$ menjadi $[0, 1]$

Probabilitas kelas dengan fungsi logistik

- Fungsi logistik (sigmoid) mengubah nilai z dari $(-\infty, \infty)$ menjadi $[0, 1]$
- Nilai $[0, 1]$ dapat diartikan sebagai probabilitas dari kelas

Probabilitas kelas dengan fungsi logistik

- Fungsi logistik (sigmoid) mengubah nilai z dari $(-\infty, \infty)$ menjadi $[0, 1]$
- Nilai $[0, 1]$ dapat diartikan sebagai probabilitas dari kelas
- Regresi linear + fungsi logistik = regresi logistik

Regresi Logistik

$$y = f(\mathbf{x}) = \sigma(\mathbf{w}^T \mathbf{x})$$

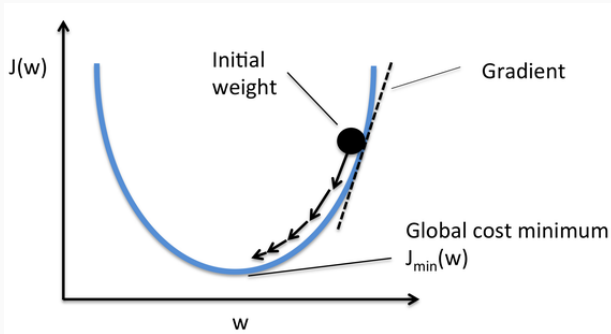


Figure 3: Menuruni lembah fungsi error $J(w)$ [Raschka, 2015]

Jaringan Saraf Tiruan (JST)

Jaringan saraf manusia

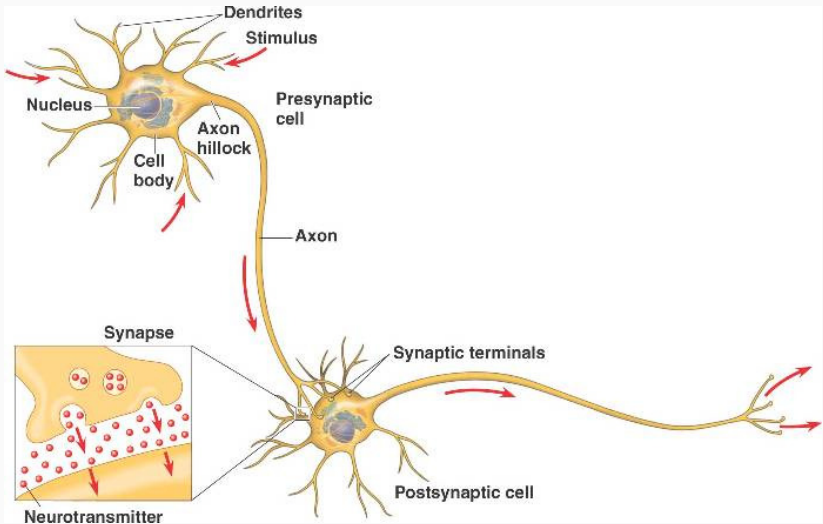


Figure 4: Neuron pembentuk jaringan saraf [NicerWeb, 2016]

- Bagaimana kalau kita menumpuk neuron-neuron yang ada [McCulloch dan Pitts, 1943]?

- Bagaimana kalau kita menumpuk neuron-neuron yang ada [McCulloch dan Pitts, 1943]?
- Hasil **keluaran dari suatu neuron** dapat dijadikan sebagai **masukan dari neuron yang lain**

- Bagaimana kalau kita menumpuk neuron-neuron yang ada [McCulloch dan Pitts, 1943]?
- Hasil **keluaran dari suatu neuron** dapat dijadikan sebagai **masukan dari neuron yang lain**
- Neuron paling akhir lah yang akan melakukan prediksi

Jaringan saraf tiruan

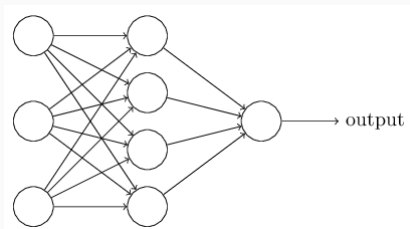


Figure 5: Jaringan saraf tiruan [Nielsen, 2016]

Jaringan saraf tiruan

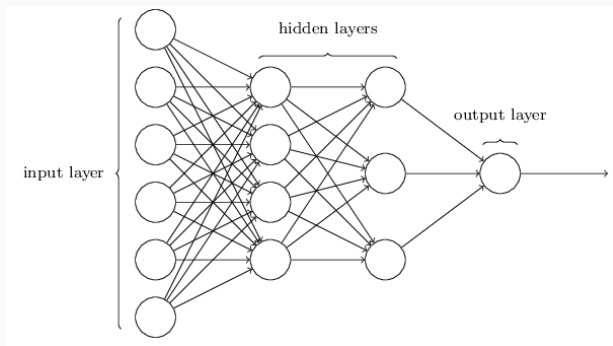


Figure 6: Lapisan jaringan saraf tiruan [Nielsen, 2016]

Beberapa terminologi yang digunakan:

- *Input, hidden, output layers*
- Tiap *layer* terdiri dari neuron atau lebih sering disebut sebagai *unit*
- Terkadang satu unit dikenal juga dengan nama *perceptron*
- Fungsi sigmoid (σ) pada tiap unit merupakan salah satu contoh dari *fungsi aktivasi*

Hidden layers

- Jumlah *hidden layers* dapat ditentukan sendiri

Hidden layers

- Jumlah *hidden layers* dapat ditentukan sendiri
- Jaringan saraf tiruan merupakan penghampiran universal (*universal approximator*), i.e. dapat meniru berbagai fungsi kontinu dengan akurasi tertentu

Hidden layers

- Jumlah *hidden layers* dapat ditentukan sendiri
- Jaringan saraf tiruan merupakan *penghampiran universal* (*universal approximator*), i.e. dapat meniru berbagai fungsi kontinu dengan akurasi tertentu
- Penghampiran tersebut dapat dicapai dengan menggunakan dua *hidden layers saja* [Cybenko, 1988]!

Bagaimana cara menentukan *weight*-nya?

Weight matrix

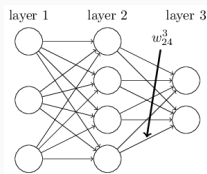


Figure 7: Penulisan *weight* dalam skalar [Nielsen, 2016]

- Karena keluarannya menjadi masukan dari beberapa *units*, maka $y_k = \sum_{j=0}^D w_{kj}x_j$
- Dalam notasi matriks-vektor, keluarannya di tiap *layer* menjadi $\mathbf{y}^l = \mathbf{W}^l \mathbf{x}^{l-1}$

Fungsi Aktivasi

Untuk tiap *layer*, \mathbf{y} menggunakan fungsi aktivasi sehingga sering diganti dengan notasi \mathbf{a} dan \mathbf{z} , maka formulanya menjadi

$\mathbf{z}^l = \mathbf{W}^l \mathbf{a}^{l-1}$ dan $\mathbf{a}^l = g(\mathbf{z}^l)$, dengan $g(\cdot) = \sigma(\cdot)$

- Masalahnya, kita belum tahu nilai **W**!

- Masalahnya, kita belum tahu nilai **\mathbf{W}** !
- Seperti regresi logistik, tidak ada solusi bentuk tertutup

- Masalahnya, kita belum tahu nilai **W**!
- Seperti regresi logistik, tidak ada solusi bentuk tertutup
- Digunakanlah metode optimasi numerik, e.g. *gradient descent*

- Diperlukan fungsi galat '*error function*' $E(\mathbf{w})$ yang dapat diminimalkan

- Diperlukan fungsi galat '*error function*' $E(\mathbf{w})$ yang dapat diminimalkan
- Galatnya adalah perbedaan prediksi dengan nilai sebenarnya

- Diperlukan fungsi galat '*error function*' $E(\mathbf{w})$ yang dapat diminimalkan
- Galatnya adalah perbedaan prediksi dengan nilai sebenarnya
- Pembelajaran \equiv menuruni permukaan fungsi galat

Gradient descent

- Diperlukan fungsi galat '*error function*' $E(\mathbf{w})$ yang dapat diminimalkan
- Galatnya adalah perbedaan prediksi dengan nilai sebenarnya
- Pembelajaran \equiv menuruni permukaan fungsi galat
- Akan sangat bergantung kepada **inisialisasi** nilai \mathbf{w} !

Cross-entropy error function

Aturan rantai turunan

$$\frac{\partial E^n}{\partial w_j} = \frac{\partial E^n}{\partial y^n} \frac{\partial y^n}{\partial a^n} \frac{\partial a^n}{\partial w_j}$$

Fungsi galat entropi-silang (cross-entropy error function)

$$E^n = -(t^n \ln(y^n) + (1 - t^n) \ln(1 - y^n))$$

Gradient descent

$$\frac{\partial E^n}{\partial w_j} = (y^n - t^n) x_j$$

Squared error function

Fungsi galat kuadrat (squared error function)

$$E^n = \frac{1}{2}(y^n - t^n)^2$$

Fungsi rataaan galat kuadrat (mean squared error (MSE) function)

$$E(\mathbf{w}) = \frac{1}{2n} \sum_n (y^n - t^n)^2$$

Stochastic gradient descent

begin

 Inisialisasi **W** dengan nilai yang kecil

 Acak urutan data latih **X**

while *not converged* **do**

for $n \leftarrow 1, N$ **do**

for $k \leftarrow 1, K$ **do**

$$y_k^n \leftarrow \sum_{j=0}^D w_{kj} x_j^n$$

$$\delta_k^n \leftarrow y_k^n - t_k^n$$

for $j \leftarrow 1, D$ **do**

$$w_{kj} \leftarrow w_{kj} - \eta \cdot \delta_k^n \cdot x_j^n$$

end

end

end

end

end

Gradient descent

$$\delta_k^n = \frac{\partial E_k^n}{\partial a_k^n} = \frac{\partial E_k^n}{\partial y_k^n} \frac{\partial y_k^n}{\partial a_k^n}$$

Learning rate

η (terkadang juga ditulis sebagai α) disebut juga sebagai *learning rate* yang biasanya di-assign dengan nilai yang kecil (< 1)

Backpropagation

- Bagusnya, metode latihan tersebut dapat diterapkan untuk tiap lapisan sebelumnya juga!

Backpropagation

- Bagusnya, metode latihan tersebut dapat diterapkan untuk tiap lapisan sebelumnya juga!
- Dikenal dengan nama *backpropagation*

Backpropagation

- Bagusnya, metode latihan tersebut dapat diterapkan untuk tiap lapisan sebelumnya juga!
- Dikenal dengan nama *backpropagation*
- Perlu disesuaikan dengan fungsi aktivasi yang digunakan pada lapisan tersebut

Backpropagation

- Bagusnya, metode latihan tersebut dapat diterapkan untuk tiap lapisan sebelumnya juga!
- Dikenal dengan nama *backpropagation*
- Perlu disesuaikan dengan fungsi aktivasi yang digunakan pada lapisan tersebut
- Sayangnya, algoritma ini mungkin terjebak pada solusi *optimum lokal*

Variasi JST

- Yang sudah kita pelajari dikenal juga sebagai *feedforward neural networks*, karena jaringannya berupa *graf berarah asiklik* '*directed acyclic graph*'

- Yang sudah kita pelajari dikenal juga sebagai *feedforward neural networks*, karena jaringannya berupa graf berarah asiklik '*directed acyclic graph*'
- Jika keluaran dari suatu neuron dijadikan masukan kembali untuk neuron tersebut (siklik) → *recurrent neural networks*

- Yang sudah kita pelajari dikenal juga sebagai *feedforward neural networks*, karena jaringannya berupa *graf berarah asiklik* '*directed acyclic graph*'
- Jika keluaran dari suatu neuron dijadikan masukan kembali untuk neuron tersebut (siklik) → *recurrent neural networks*
- *Recurrent neural networks* biasa digunakan untuk tugas yang *berhubungan dengan urutan*, e.g. *natural language processing, speech recognition*

Recurrent neural networks (RNN; non-examinable)

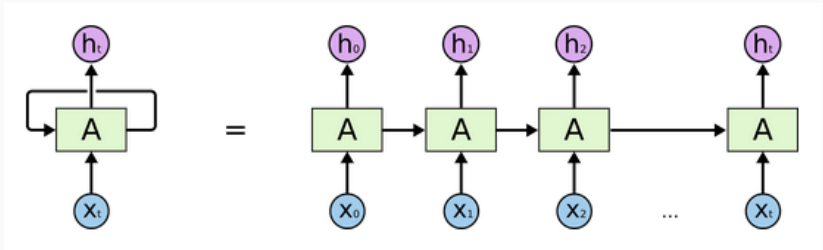


Figure 8: Recurrent neural networks jika dilihat secara sekuensial
[Olah, 2015]

Variasi fungsi aktivasi

- Fungsi sigmoid $\sigma(z)$ memiliki kelemahan, i.e. mudah sekali jenuh

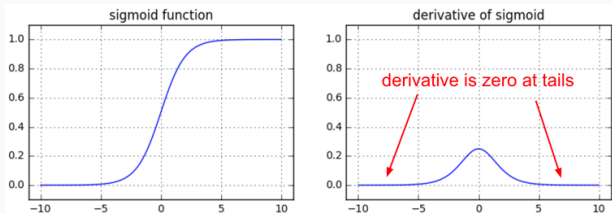


Figure 9: Fungsi sigmoid yang jenuh karena asimtotik [Karpathy, 2016]

Variasi fungsi aktivasi

- Fungsi sigmoid $\sigma(z)$ memiliki kelemahan, i.e. mudah sekali jenuh
- Solusi: inisialisasi \mathbf{W} dengan nilai yang kecil

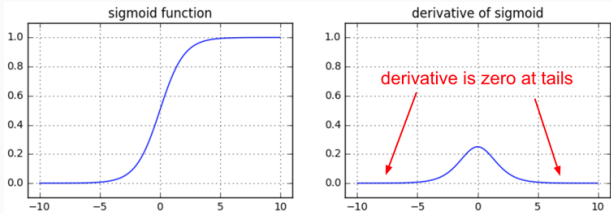


Figure 9: Fungsi sigmoid yang jenuh karena asimtotik [Karpathy, 2016]

Variasi fungsi aktivasi

- Fungsi sigmoid $\sigma(z)$ memiliki kelemahan, i.e. mudah sekali jenuh
- Solusi: inisialisasi \mathbf{W} dengan nilai yang kecil
- Diperkenalkan fungsi aktivasi lain, e.g. rectified linear unit (ReLU) $g(z) = \max(0, z)$

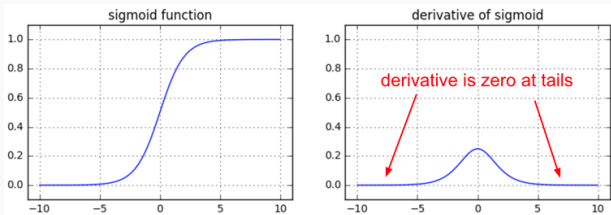


Figure 9: Fungsi sigmoid yang jenuh karena asimtotik [Karpathy, 2016]

Beberapa fungsi aktivasi yang dapat digunakan

- $g(z) = \sigma(z) = \frac{1}{1+e^{-z}}$ — sigmoid
- $g(z) = \tanh(z)$ — $g(z) \in [-1, 1]$ (juga cepat jenuh)
- $g(z) = z$ — linear unit
- $g(z) = \Theta(z)$ — threshold unit
- $g(z) = \max(0, z)$ — rectified linear unit (ReLU)

Pengembangan neural networks (non-examinable)

- Penggunaan *convolutional layer* dan *max pooling layer*: CNN

Pengembangan neural networks (non-examinable)

- Penggunaan *convolutional layer* dan *max pooling layer*: CNN
- Pengembangan metode *gradient descent*, e.g. dengan *momentum* atau *performance-based*

Pengembangan neural networks (non-examinable)

- Penggunaan *convolutional layer* dan *max pooling layer*: CNN
- Pengembangan metode *gradient descent*, e.g. dengan *momentum* atau *performance-based*
- Penggunaan regularisasi L1 dan L2

Pengembangan neural networks (non-examinable)

- Penggunaan *convolutional layer* dan *max pooling layer*: CNN
- Pengembangan metode *gradient descent*, e.g. dengan *momentum* atau *performance-based*
- Penggunaan regularisasi L1 dan L2
- Inisialisasi dengan pralatih '*pretraining*', e.g. *autoencoder*

Aplikasi JST

Contoh implementasi JST dengan keras.io

1. MNIST: pendeteksian digit yang ditulis tangan (LeCun dan Bengio, 1995)
2. Klasifikasi objek visual (Krizhevsky et al, 2012)
3. Speech recognition (Hinton et al, 2012)
4. Representasi vektor dari kata-kata (Mikolov et al, 2013)



Figure 10: Prisma yang menggunakan JST jenis CNN (Gatys et al, 2015)

Convolutional NNs (non-examinable)

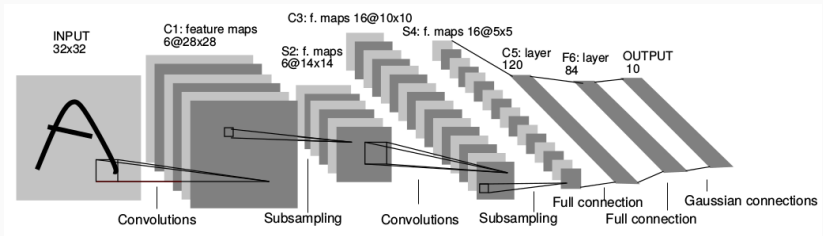


Figure 11: Penggunaan lapisan *convolutional* dan *pooling* pada LeNet [Murphy, 2012] Fig. 16.14

Salindia ini dipersiapkan dengan sangat dipengaruhi oleh:
Chris Williams (2015) dan Steve Renals (2015)

Summary

- Merupakan penghampir universal — sangat mungkin terjadi *overfitting*
- Dapat terjebak dalam *solusi optimum lokal*
- Bisa jadi sangat lambat karena metode *gradient descent*
- *Gradient descent* dapat “diteruskan” ke *layer sebelumnya*, dikenal dengan nama *backpropagation*
- Punya beberapa *alternatif fungsi aktivasi* selain sigmoid
- Sulit diinterpretasi — sangat terbuka untuk dikembangkan

Pertemuan berikutnya

- Nearest neighbours
- Unsupervised learning
- Clustering dengan k-Means



NicerWeb (diakses tanggal 5 Desember 2016)

Neuron

[http://www.nicerweb.com/bio1152/
Locked/media/ch48/neuron.html](http://www.nicerweb.com/bio1152/Locked/media/ch48/neuron.html)



Warren S. McCulloch dan Walter Pitts (1943)

A logical calculus of the ideas immanent in nervous activity

The bulletin of mathematical biophysics 5(4), 115 – 133.



Michael Nielsen (2016)

Neural Networks and Deep Learning

<http://neuralnetworksanddeeplearning.com/>

References



G. Cybenko (1988)

Continuous valued neural networks with two hidden layers are sufficient

Center for Supercomputing Research and Development



Sebastian Raschka (2015)

Single-Layer Neural Networks and Gradient Descent

http://sebastianraschka.com/Articles/2015_singlelayer_neurons.html



Christopher Olah (2015)

Understanding LSTM Networks

<http://colah.github.io/posts/2015-08-Understanding-LSTMs/>



Andrej Karpathy (2016)

Yes you should understand backprop

[https://medium.com/@karpathy/
yes-you-should-understand-backprop-e2f06eab496b#
.70lzt4tw2](https://medium.com/@karpathy/yes-you-should-understand-backprop-e2f06eab496b#.70lzt4tw2)



Kevin P. Murphy (2012)

Machine Learning: a Probabilistic Perspective

MIT Press

Thank you