Jaringan Saraf Tiruan
Artificial Neural Networks

Ali Akbar Septiandri

Universitas Al-Azhar Indonesia

aliakbars@live.com

May 3, 2020

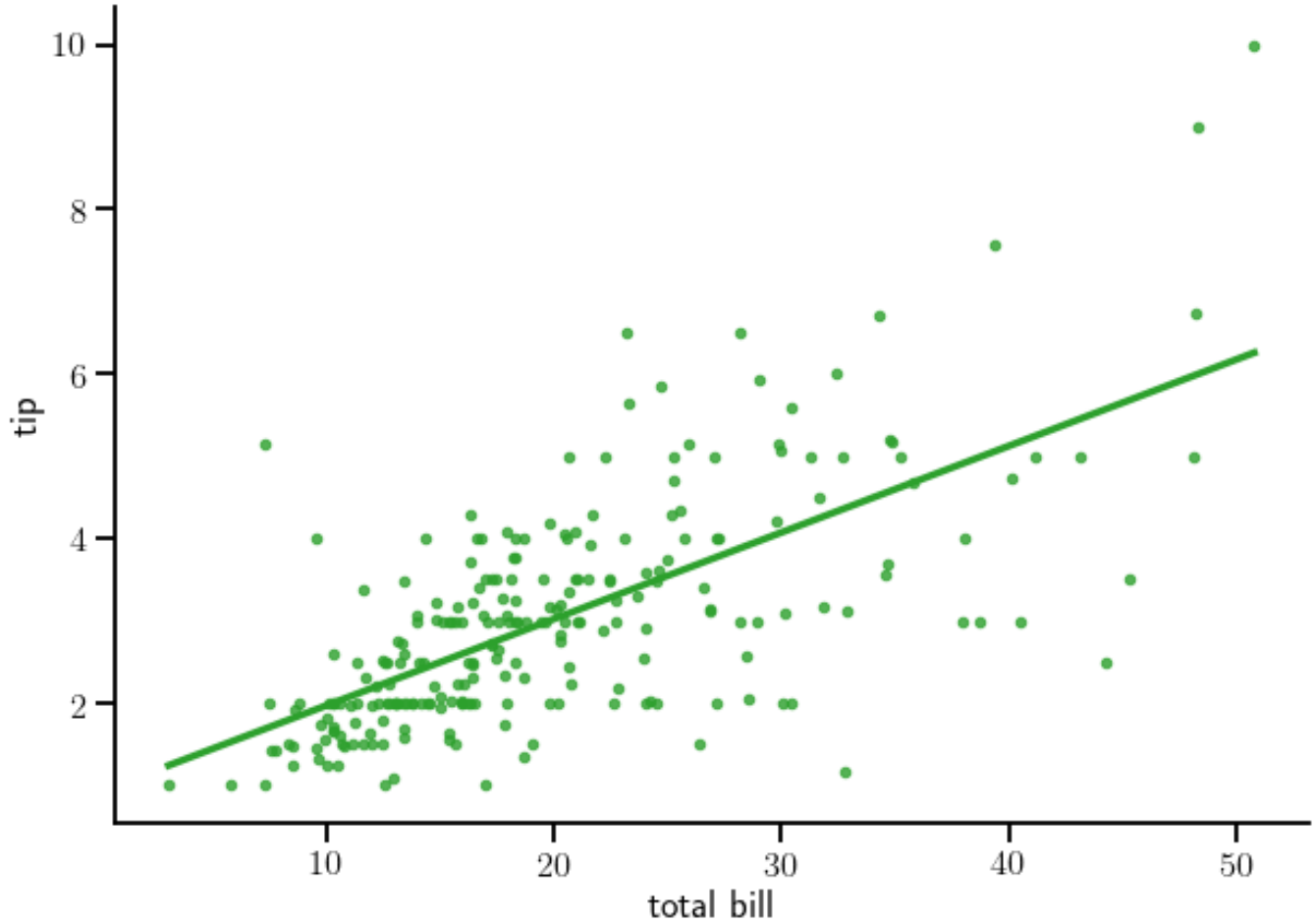
Selayang Pandang

Bahan Bacaan

1. Nielsen, M. A. (2015). *Neural networks and deep learning*. San Francisco, CA, USA: Determination press.
2. Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press.
3. LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature, 521*(7553), 436-444.

Ulasan

Regresi Linear



Gambar: Mencari hubungan $\mathbf{y} = X\mathbf{w}$ dengan meminimalkan $E(\mathbf{w}) = \sum_{i=1}^{N}(y_i - \mathbf{w}^T\mathbf{x}_i)^2$

Regresi Linear Multidimensi

$$y = w_0 + w_1 x_1 + ... + w_D x_D = \sum_{j=0}^{D} w_j x_j$$

Regresi Linear Multidimensi (notasi vektor)

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

Optimasi analitis

- Fungsi error

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

- Solusi tertutupnya:

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

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

Probabilitas kelas dengan fungsi logistik

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

Probabilitas kelas dengan fungsi logistik

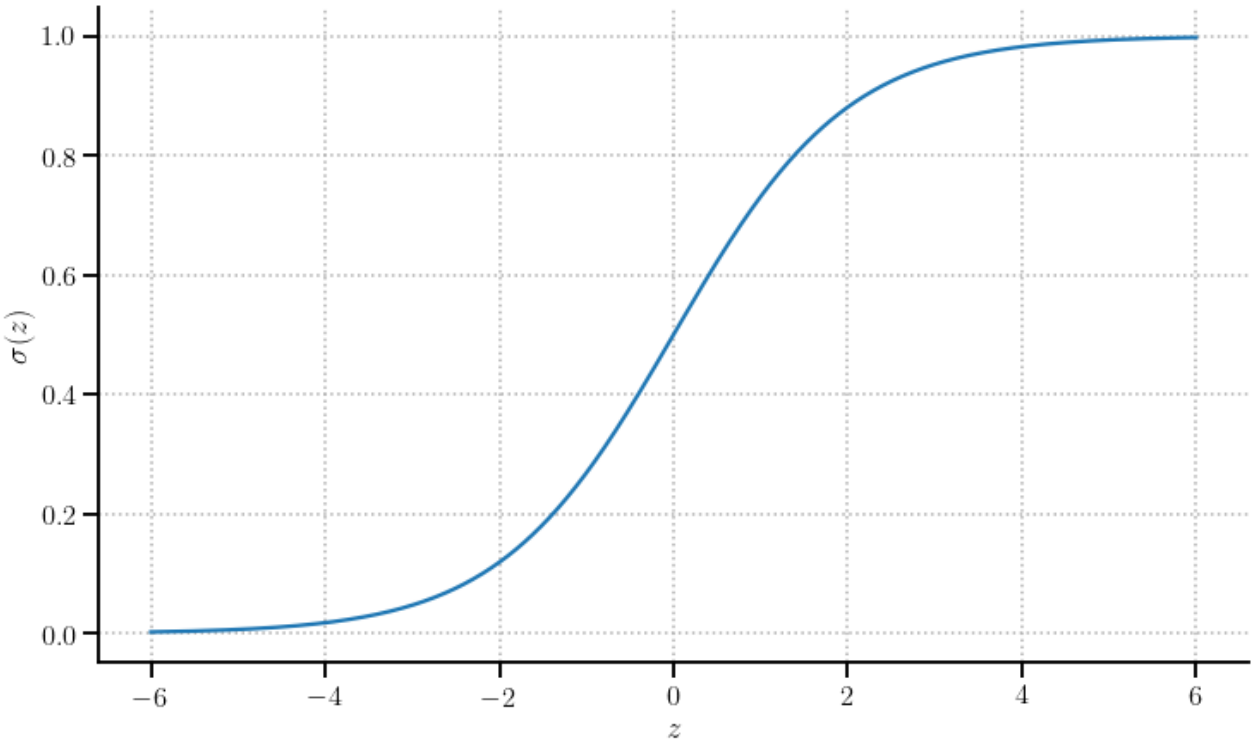
- Fungsi logistik (sigmoid) mengubah nilai z dari $(-\infty, \infty)$ menjadi $[0, 1]$
- Nilai $[0, 1]$ dapat diartikan sebagai probabilitas dari kelas

Regresi logistik

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

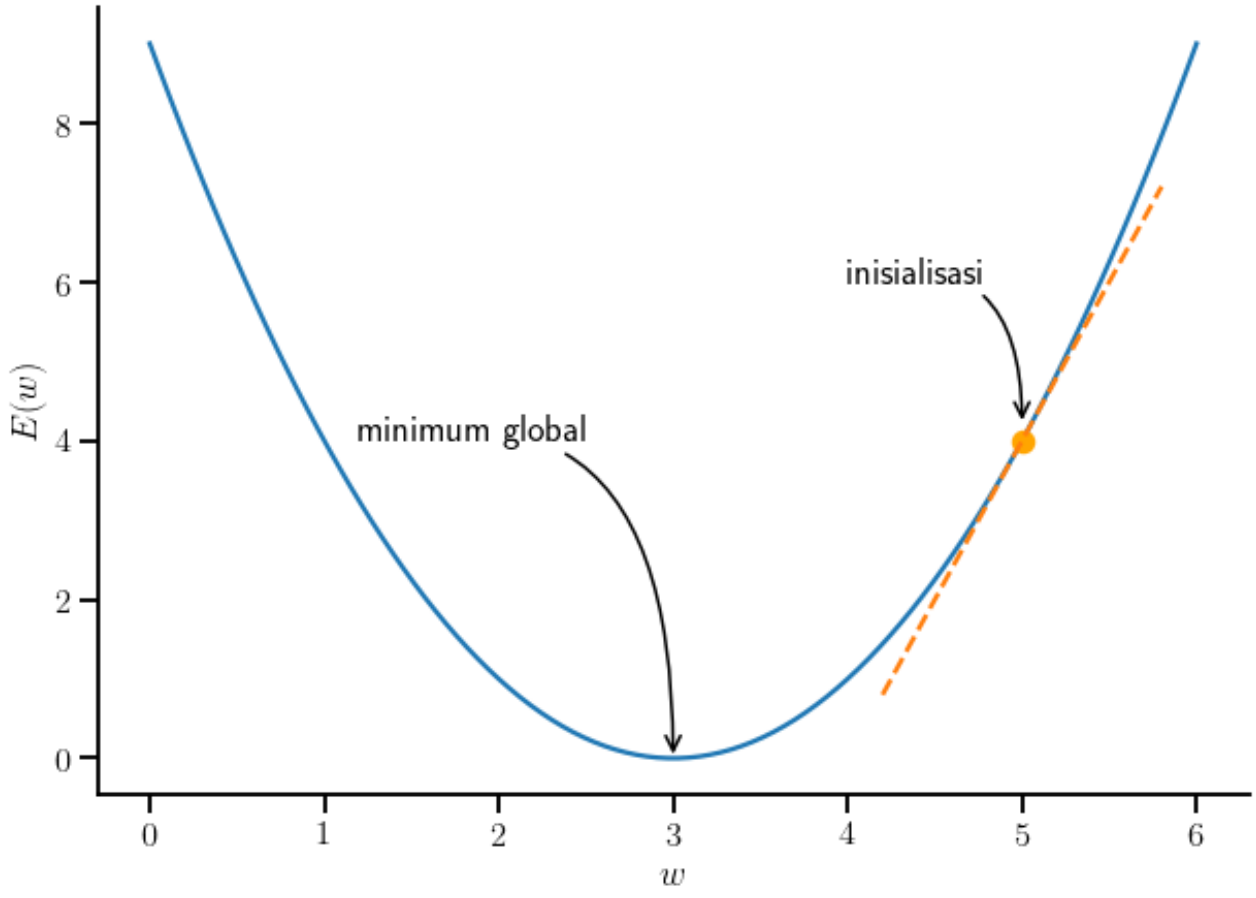
Regresi logistik



Gambar: Fungsi sigmoid/logistik $\sigma(z) = \frac{1}{1+exp(-z)}$

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

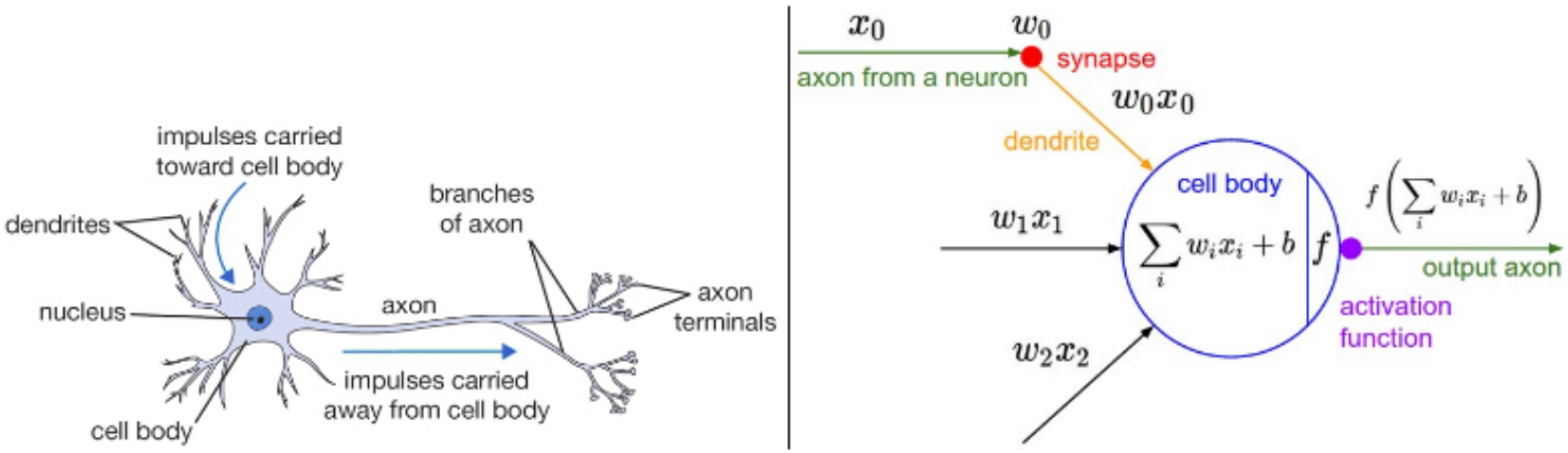
Optimasi numerik



Gambar: Menuruni lembah fungsi error $E(w)$

Jaringan Saraf Tiruan (JST)

Jaringan saraf manusia



Gambar: Neuron pembentuk jaringan saraf

Sumber: `http://cs231n.github.io/neural-networks-1/`

Jaringan saraf tiruan

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

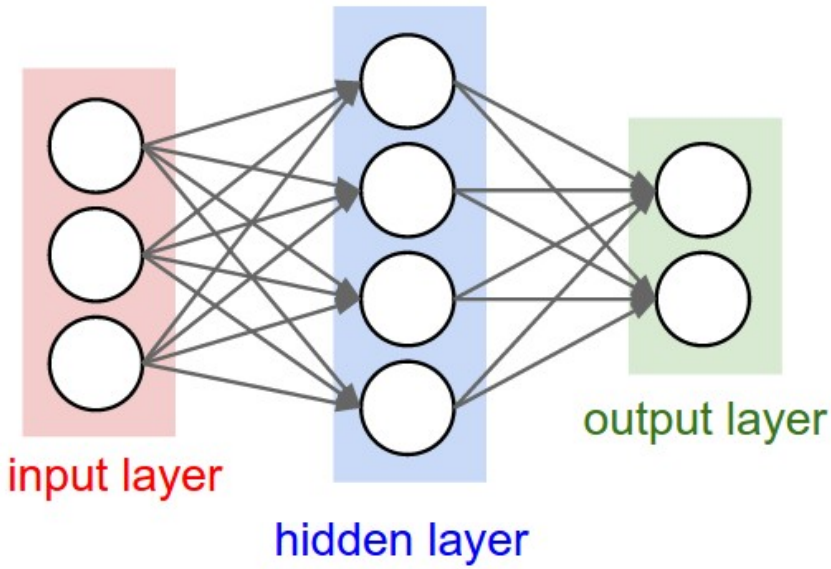
Jaringan saraf tiruan

- 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

Jaringan saraf tiruan

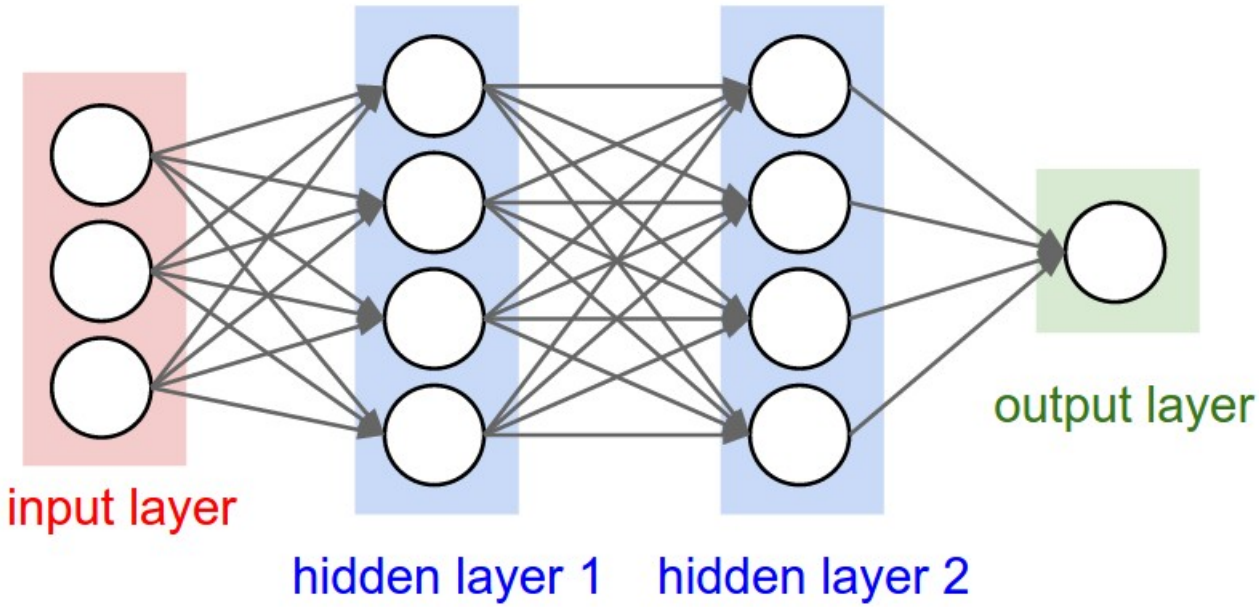
- Bagaimana kalau kita menumpuk neuron-neuron yang ada [McCulloch dan Pitts, 1943]?
- Hasil keluaran dari suatu neuron dapat dijadikan sebagai masukan dari neuron yang lain
- Neuron paling akhir lah yang akan melakukan prediksi

Jaringan saraf tiruan



Gambar: Jaringan saraf tiruan [Karpathy, 2017]

Jaringan saraf tiruan



Gambar: Lapisan jaringan saraf tiruan [Karpathy, 2017]

Terminologi

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

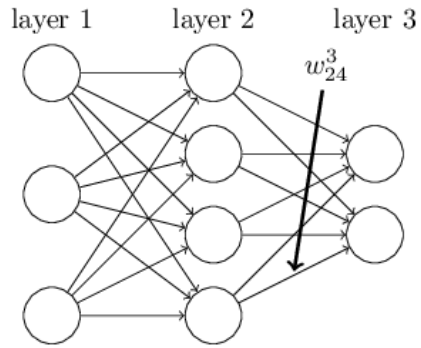
- Jumlah *hidden layers* dapat ditentukan sendiri
- Jaringan saraf tiruan merupakan penghampiran universal (*universal approximator*), i.e. dapat meniru berbagai fungsi kontinu dengan akurasi tertentu

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



Gambar: 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 $\mathbf{y}^l = \mathbf{W}^l \mathbf{x}^{l-1}$

Regresi logistik dalam JST

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 $g(\cdot) = \sigma(\cdot)$

Optimasi solusi

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

Optimasi solusi

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

Optimasi solusi

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

Gradient descent

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

Gradient descent

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

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

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 rataan 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

Catatan

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*

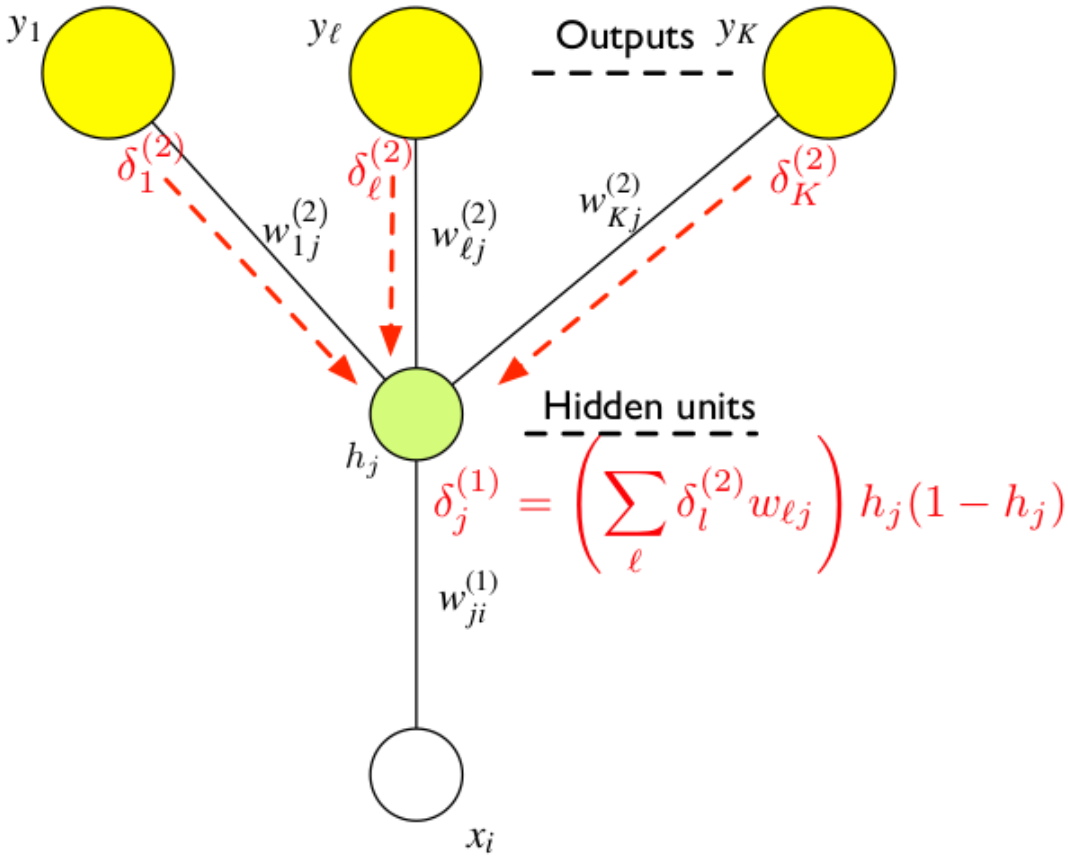
Backpropagation

- 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

Backpropagation

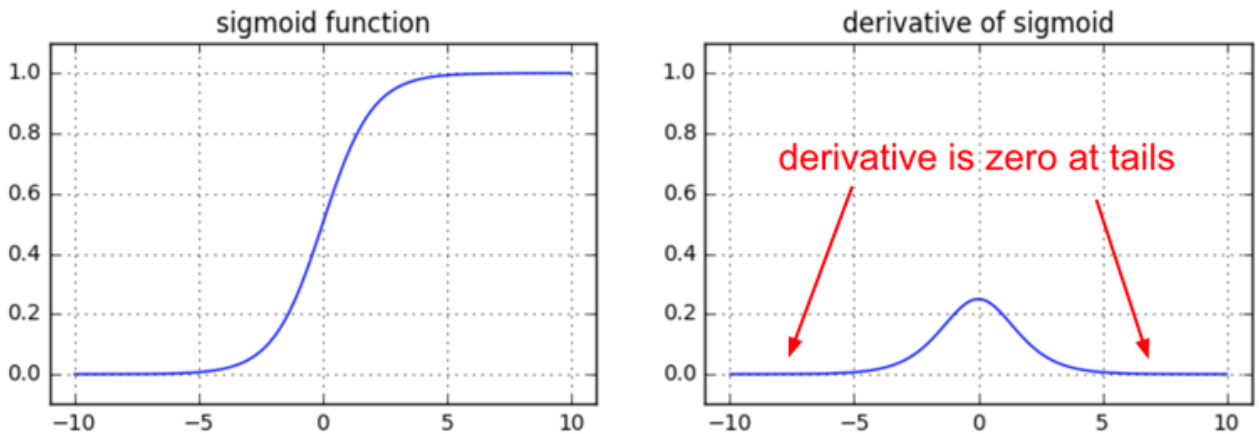


Gambar: Backpropagation untuk error pada *hidden units* (Renals, 2015)

Variasi JST

Variasi fungsi aktivasi

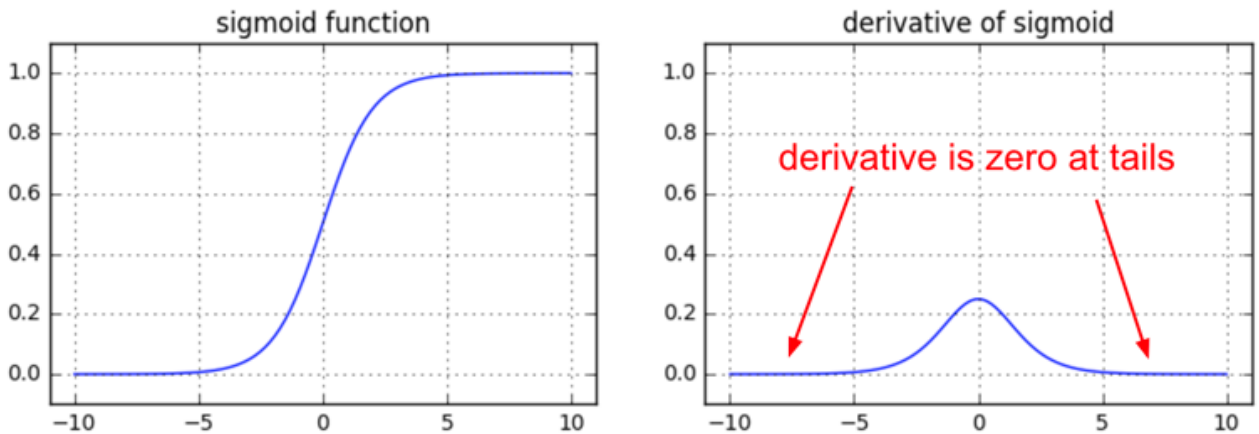
- Fungsi sigmoid $\sigma(z)$ memiliki kelemahan, i.e. mudah sekali jenuh



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

Variasi fungsi aktivasi

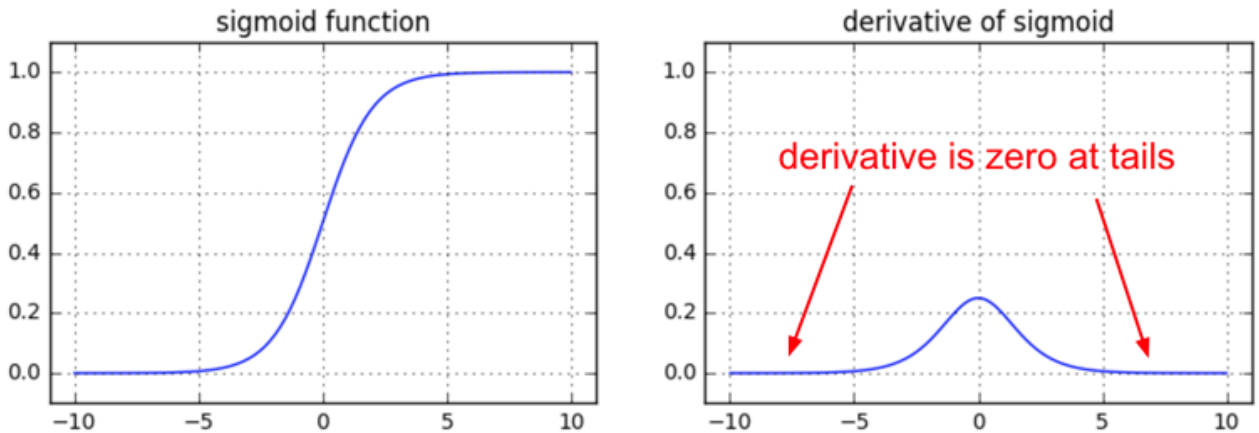
- Fungsi sigmoid $\sigma(z)$ memiliki kelemahan, i.e. mudah sekali jenuh
- Solusi: inisialisasi $\mathbf{W}$ dengan nilai yang kecil



Gambar: 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) $g(z) = max(0, z)$



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

Variasi fungsi aktivasi

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

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

Pengembangan neural networks

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

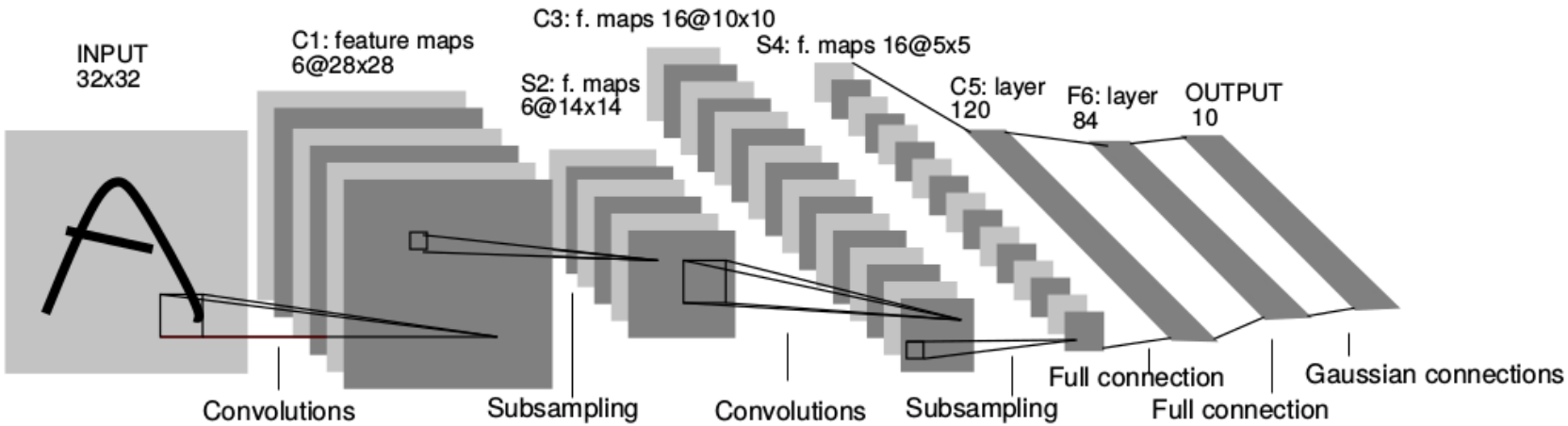
Pengembangan neural networks

- 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

- 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*

Convolutional Neural Networks



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

Recurrent Neural Networks

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

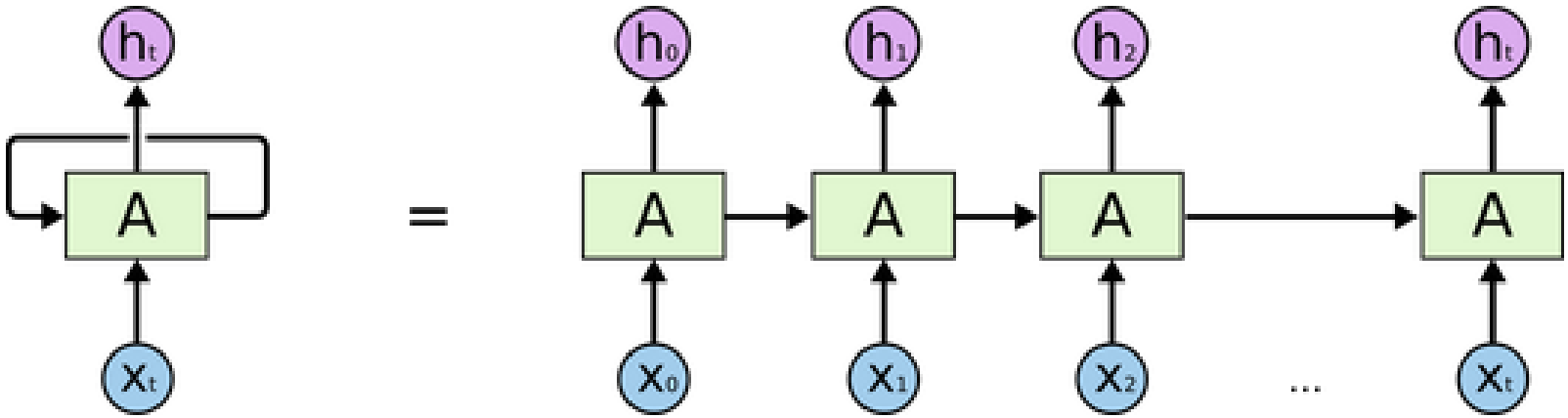
Recurrent Neural Networks

- 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) $\rightarrow$ *recurrent neural networks*

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) $\rightarrow$ *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)



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

Aplikasi JST

Aplikasi JST

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)
5. Pengolahan data teks (Howard dan Ruder, 2018; Devlin et al., 2018)

AlphaGo (Silver et al., 2017)

Contoh implementasi neural networks dengan Keras

1. Multilayer Perceptron
2. Dasar-dasar Keras
3. Convolutional Neural Networks (dengan TPU)

Alternatif yang lebih cepat (fast.ai)

1. Practical Deep Learning for Coders, v3
2. Deep Learning from the Foundations

Neural Networks from Scratch

http://www.wildml.com/2015/09/implementing-a-neural-network-from-scratch/

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

Ikhtisar

- 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*

Referensi

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/

Referensi

Andrej Karpathy (2017)

Neural Networks Part 1: Setting Up the Architecture

http://cs231n.github.io/neural-networks-1/

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

Referensi

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

Kevin P. Murphy (2012)

Machine Learning: a Probabilistic Perspective

MIT Press

Terima kasih