

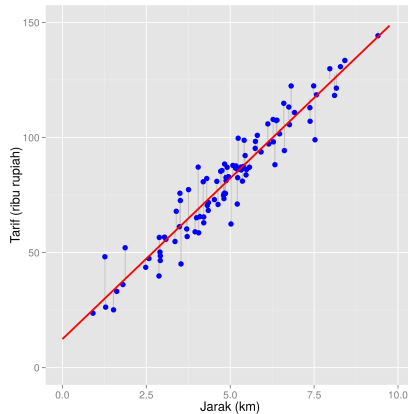
Neural Networks

Ali Akbar Septiandri

Universitas Al Azhar Indonesia

October 8, 2018

Regresi linear



Gambar: Regresi linear tarif taksi dari jarak tempuh

Regresi linear

Regresi Linear Satu Dimensi

$$y = w_0 + w_1 x_1$$

Regresi linear

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)

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

Optimasi analitis

- Fungsi error

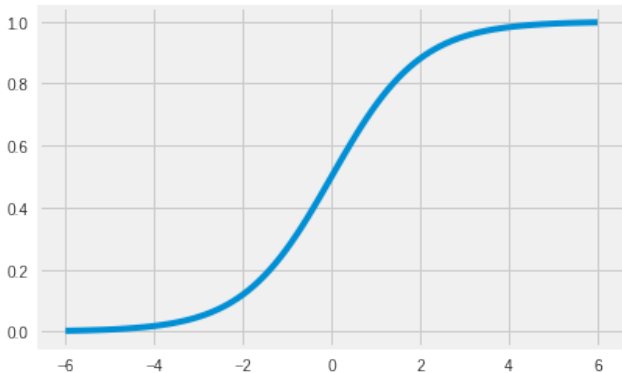
$$E(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^N (y_i - \mathbf{w} \cdot \phi(x_i))^2$$

- Solusi tertutupnya:

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

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

Regresi logistik



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

Regresi logistik

Probabilitas kelas dengan fungsi logistik

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

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 logistik

Probabilitas kelas dengan fungsi logistik

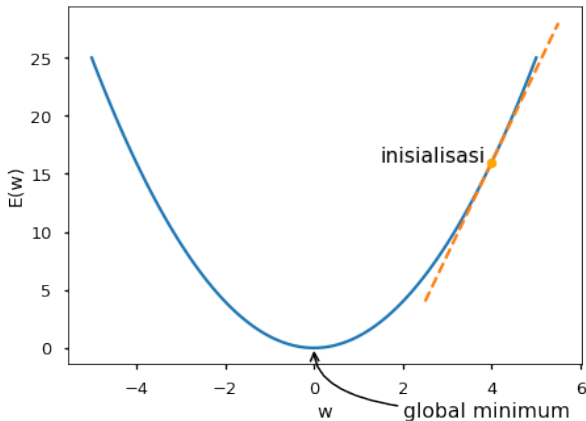
- Fungsi logistik (sigmoid) mengubah nilai z dari $(-\infty, \infty)$ menjadi $[0, 1]$
- Nilai $[0, 1]$ dapat diartikan sebagai probabilitas dari kelas
- Regresi linear + fungsi logistik = regresi logistik

Regresi logistik

Regresi Logistik

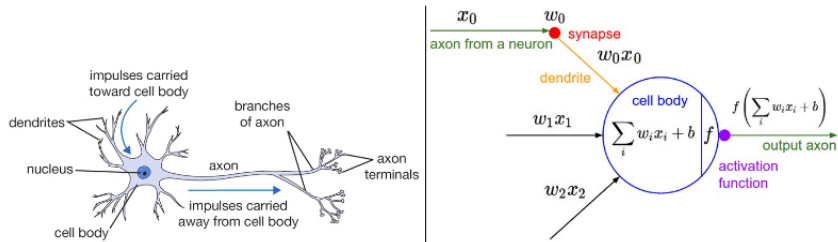
$$y = f(\mathbf{x}) = \sigma(\mathbf{w}^T \mathbf{x})$$

Optimasi numerik



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

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]?

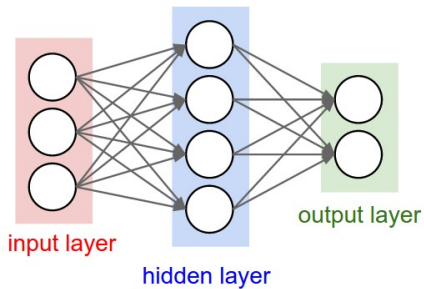
Jaringan saraf tiruan

- 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

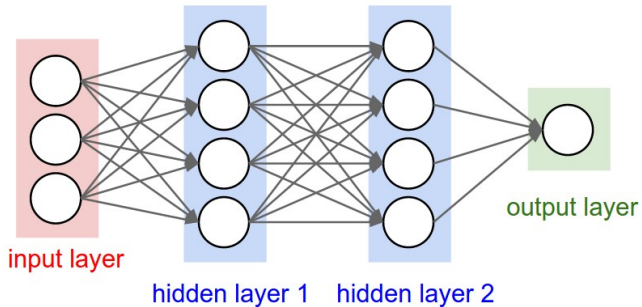
- Bagaimana kalau kita menumpuk neuron-neuron yang ada [McCulloch dan Pitts, 1943]?
- Hasil **keluaran dari suatu neuron** dapat dijadikan sebagai **masukan dari neuron yang lain**
- Neuron paling akhir lah yang akan melakukan prediksi

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

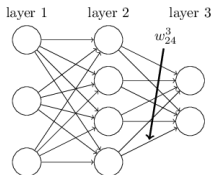
- Jumlah *hidden layers* dapat **ditentukan sendiri**
- Jaringan saraf tiruan merupakan **penghampiran universal** (*universal approximator*), i.e. dapat meniru berbagai fungsi kontinu dengan akurasi tertentu

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

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

Optimasi solusi

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

Optimasi solusi

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

Optimasi solusi

- Masalahnya, kita belum tahu nilai **\mathbf{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 rata-rata 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**

Topologi JST

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

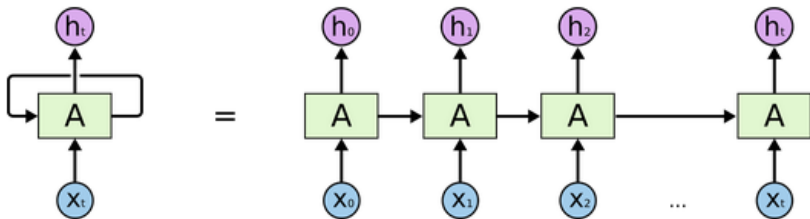
Topologi JST

- 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*

Topologi JST

- 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)



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

Contoh implementasi neural networks dengan Keras

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*

Pertemuan berikutnya

- Multilayer perceptron
- Generalisasi
- Regularisasi

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#.701zt4tw2>



Kevin P. Murphy (2012)

Machine Learning: a Probabilistic Perspective

MIT Press

Terima kasih