Penjelasan Algoritma Backpropagation

Hendra Bunyamin

April 16, 2020

Lecture notes ini hendak memberikan penjelasan tambahan untuk video *Backpropagation Algorithm* (Stanford ML, 2020a) dari Week 5 MOOC *Stanford Machine Learning* (Stanford ML, 2020b).

Penjelasan ini akan dimulai dengan *logistic regression* (LR). *Logistic regression* sebenarnya adalah *neural networks* dengan 2 layer, yaitu *input layer* dan *output layer*.

Logistic regression

Contoh *logistic regression* digambarkan pada Figure 1. Model *logistic regression* ini mempunyai 2 *features*, yaitu x_1 (*neuron* berwarna hijau) dan x_2 (*neuron* berwarna hijau). Ada juga θ_1 (tanda panah yang menghubungkan x_1 dengan $a_1^{(2)}$ (*neuron* berwarna merah muda)), dan θ_2 (tanda panah yang menghubungkan x_2 dan $a_1^{(2)}$). Terdapat juga *bias term* (x_0) yang tidak tergambar secara eksplisit dan θ_0 (tanda panah yang menghubungkan x_0 dan $a_1^{(2)}$).

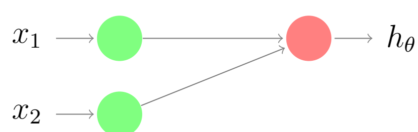

Figure 1: *Logistic regression* dengan 2 *feature*, yaitu x_1 dan x_2

Model *logistic regression* (h_θ) memiliki bentuk

$$h_\theta(x_1, x_2) = \frac{1}{1 + e^{-(\theta_0 + \theta_1 x_1 + \theta_2 x_2)}} \tag{1}$$

atau bentuk *vectorized*-nya adalah

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}} \tag{2}$$

dengan $\theta = \begin{bmatrix} \theta_0 & \theta_1 & \theta_2 \end{bmatrix}^T$ dan $x = \begin{bmatrix} 1 & x_1 & x_2 \end{bmatrix}^T$.

Gradient dari Logistic Regression untuk satu training instance

Misalkan terdapat satu *training instance*, yaitu $x = \begin{bmatrix}1 & x_1 & x_2\end{bmatrix}^T$ dan y. Akan dihitung *gradient* dari model *logistic regression* tersebut. *Gradient* ini sudah dihitung di perkuliahan sebelumnya, yaitu:

$$\frac{\partial J(\theta)}{\partial \theta_0} = (h_\theta(x) - y), \tag{3}$$

$$\frac{\partial J(\theta)}{\partial \theta_1} = (h_\theta(x) - y)x_1, \text{ dan} \tag{4}$$

$$\frac{\partial J(\theta)}{\partial \theta_2} = (h_\theta(x) - y)x_2. \tag{5}$$

Marilah kita mendefinisikan notasi berikut:

$$\delta_1^{(2)} = (h_\theta(x) - y). \tag{6}$$

$\delta_1^{(2)}$ ini dibaca *delta* pada $a_1^{(2)}$ dan merupakan *selisih prediksi model dengan nilai sebenarnya*.

Dengan menggunakan notasi $\delta_1^{(2)}$ pada Persamaan (6), Persamaan (3), (4), dan (5) dapat ditulis menjadi

$$\frac{\partial J(\theta)}{\partial \theta_0} = \delta_1^{(2)}, \tag{7}$$

$$\frac{\partial J(\theta)}{\partial \theta_1} = \delta_1^{(2)} x_1, \text{ dan} \tag{8}$$

$$\frac{\partial J(\theta)}{\partial \theta_2} = \delta_1^{(2)} x_2. \tag{9}$$

OK, yang perlu diperhatikan adalah bahwa **setiap *neuron* selain *neuron-neuron* di *input layer* memiliki delta ($\boldsymbol{\delta}$)**. Lebih lanjut,

$$\boxed{\text{gradient} = \delta \times \text{input.}}$$

Gradient dari LR untuk banyak training instance

Diketahui m *training instances* dan *label*nya, yaitu

$$X = \begin{bmatrix} 1 & x_1^{(1)} & x_2^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} \\ \vdots & \vdots & \vdots \\ 1 & x_1^{(m)} & x_2^{(m)} \end{bmatrix} \text{ dan } y = \begin{bmatrix} y^{(1)} & y^{(1)} & \cdots & y^{(m)} \end{bmatrix}^T$$

Seperti yang dibahas di perkuliahan sebelumnya, *gradient* dari model *logistic regression* dengan banyak *training instance* ini adalah

$$\frac{\partial J(\theta)}{\partial \theta_0} = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}), \tag{10}$$

$$\frac{\partial J(\theta)}{\partial \theta_1} = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_1^{(i)}, \text{ dan} \tag{11}$$

$$\frac{\partial J(\theta)}{\partial \theta_2} = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_2^{(i)}. \tag{12}$$

Lebih lanjut lagi, kita perkenalkan notasi berikut:

$$\delta_1^{(2)(i)} = (h_\theta(x^{(i)}) - y^{(i)}). \tag{13}$$

$\delta_1^{(2)(i)}$ adalah *selisih prediksi model dengan nilai sebenarnya* untuk *instance* ke-i, mirip dengan notasi delta di Persamaan (6).

Dengan menggunakan notasi $\delta_1^{(2)(i)}$ di Persamaan (13), Persamaan (10), (11), dan (12) menjadi

$$\frac{\partial J(\theta)}{\partial \theta_0} = \frac{1}{m} \sum_{i=1}^{m} \delta_1^{(2)(i)}, \tag{14}$$

$$\frac{\partial J(\theta)}{\partial \theta_1} = \frac{1}{m} \sum_{i=1}^{m} \delta_1^{(2)(i)} x_1^{(i)}, \text{ dan} \tag{15}$$

$$\frac{\partial J(\theta)}{\partial \theta_2} = \frac{1}{m} \sum_{i=1}^{m} \delta_1^{(2)(i)} x_2^{(i)}. \tag{16}$$

Kembali kita menggunakan notasi baru yaitu

$$\Delta_{kj}^{(l)} = \sum_{i=1}^{m} \delta_k^{(l+1)(i)} x_j^{(i)} \tag{17}$$

dengan

$$\Delta_{kj}^{(l)} \quad = \quad \text{Total penjumlahan } \delta_k^{(l+1)(i)} x_j^{(i)} \text{ untuk instance ke-} i = 1, \ldots, m$$

sehingga Persamaan (14), (15), (16) menjadi

$$\frac{\partial J(\theta)}{\partial \theta_0} = \frac{1}{m}\Delta_{10}^{(1)}, \tag{18}$$
$$\frac{\partial J(\theta)}{\partial \theta_1} = \frac{1}{m}\Delta_{11}^{(1)}, \text{ dan} \tag{19}$$
$$\frac{\partial J(\theta)}{\partial \theta_2} = \frac{1}{m}\Delta_{12}^{(1)} \tag{20}$$

Dengan definisi baru ini, algoritma *gradient descent* menjadi

$$\theta_0 = \theta_0 - \alpha \times \frac{1}{m}\Delta_{10}^{(1)}, \tag{21}$$
$$\theta_1 = \theta_1 - \alpha \times \frac{1}{m}\Delta_{11}^{(1)}, \text{ dan} \tag{22}$$
$$\theta_2 = \theta_2 - \alpha \times \frac{1}{m}\Delta_{12}^{(1)} \tag{23}$$

Dengan pemahaman ini, selanjutnya kita akan menghitung gradient dari *neural network*.

Artificial Neural Network (ANN)

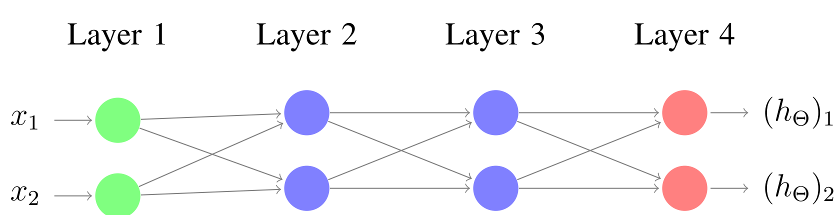

Figure 2: *Artificial neural network* dengan 2 *hidden layer*

Figure 2 menggambarkan ANN dengan 2 *hidden layer* dan Algorithm 1 menjelaskan detil dari algoritma *backpropagation* yang dijelaskan di video.

Marilah kita mulai membahas proses di dalam algoritma *backpropagation* pada Figure 2. Setelah *forward propagation* dilakukan, nilai-nilai dari $a_1^{(2)}$, $a_2^{(2)}$, $a_1^{(3)}$, $a_2^{(3)}$, $a_1^{(4)}$, dan $a_2^{(4)}$ diketahui. Algoritma *backpropagation* dimulai dari layer terakhir, dalam hal ini adalah layer 4. Di layer 4 dapat dihitung

$$\delta_1^{(4)} = ((h_\Theta(x))_1 - y_1) \tag{24}$$
$$\delta_2^{(4)} = ((h_\Theta(x))_2 - y_2) \tag{25}$$

Algorithm 1 Detil algoritma *Backpropagation* dari Video Andrew Ng

procedure ALGORITMA-BACKPROPAGATION
 ▷ Our Training Set: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})\}$
 Set $\Delta_{ij}^{(l)} = 0$ (for all l, i, j)
 for $i = 1$ to m **do**
  Set $a^{(1)} = x^{(i)}$
  Perform forward propagation to compute $a^{(l)}$ for $l = 2, 3, \ldots, L$
  Using $y^{(i)}$, compute $\delta^{(L)} = a^{(L)} - y^{(i)}$
  Compute $\delta^{(L-1)}, \delta^{(L-2)}, \ldots, \delta^{(2)}$
  $\Delta_{ij}^{(l)} := \Delta_{ij}^{(l)} + a_j^{(l)} \delta_i^{(l+1)}$
 end for
 $D_{ij}^{(l)} := \frac{1}{m}\Delta_{ij}^{(l)} + \lambda\Theta_{ij}^{(l)}$ if $j \neq 0$
 $D_{ij}^{(l)} := \frac{1}{m}\Delta_{ij}^{(l)}$ if $j = 0$
 ▷ Catatan: $\frac{\partial J(\Theta)}{\partial \Theta_{ij}^{(l)}} = D_{ij}^{(l)}$
end procedure

Persamaan (24) dan Persamaan (25) dapat dijadikan bentuk *vectorized* menjadi

$$\delta^{(4)} = \begin{bmatrix} (h_\Theta(x))_1 - y_1 \\ (h_\Theta(x))_2 - y_2 \end{bmatrix} \tag{26}$$

Selanjutnya di layer 3 dapat dihitung

$$\delta_0^{(3)} = (\Theta_{10}^{(3)}\delta_1^{(4)} + \Theta_{20}^{(3)}\delta_2^{(4)})a_0^{(3)} \times (1 - a_0^{(3)}) \tag{27}$$
$$\delta_1^{(3)} = (\Theta_{11}^{(3)}\delta_1^{(4)} + \Theta_{21}^{(3)}\delta_2^{(4)})a_1^{(3)} \times (1 - a_1^{(3)}) \tag{28}$$
$$\delta_2^{(3)} = (\Theta_{12}^{(3)}\delta_1^{(4)} + \Theta_{22}^{(3)}\delta_2^{(4)})a_2^{(3)} \times (1 - a_2^{(3)}) \tag{29}$$

Persamaan (27), (28), dan (29) dapat dijadikan vektor menjadi

$$\underbrace{\begin{bmatrix} \delta_0^{(3)} \\ \delta_1^{(3)} \\ \delta_2^{(3)} \end{bmatrix}}_{\delta^{(3)}} = \underbrace{\begin{bmatrix} \Theta_{10}^{(3)} & \Theta_{20}^{(3)} \\ \Theta_{11}^{(3)} & \Theta_{21}^{(3)} \\ \Theta_{12}^{(3)} & \Theta_{22}^{(3)} \end{bmatrix}}_{(\Theta^{(3)})^T} \underbrace{\begin{bmatrix} \delta_1^{(4)} \\ \delta_2^{(4)} \end{bmatrix}}_{\delta^{(4)}} .* \underbrace{\begin{bmatrix} a_0^{(3)} \\ a_1^{(3)} \\ a_2^{(3)} \end{bmatrix}}_{a^{(3)}} .* \underbrace{\begin{bmatrix} 1 - a_0^{(3)} \\ 1 - a_1^{(3)} \\ 1 - a_2^{(3)} \end{bmatrix}}_{1 - a^{(3)}} . \tag{30}$$

Kemudian elemen $\delta_0^{(3)}$ pada $\delta^{(3)}$ kita buang karena $\delta_0^{(3)}$ adalah error atau selisih pada *bias* $a_0^{(3)}$ yang bukan neuron sebenarnya. Oleh karena itu,

$$\delta^{(3)} = \begin{bmatrix} \delta_1^{(3)} \\ \delta_2^{(3)} \end{bmatrix} \tag{31}$$

Selanjutnya, di layer 2 dapat dihitung

$$\delta_0^{(2)} = (\Theta_{10}^{(2)}\delta_1^{(3)} + \Theta_{20}^{(2)}\delta_2^{(3)})a_0^{(2)} \times (1 - a_0^{(2)}) \tag{32}$$

$$\delta_1^{(2)} = (\Theta_{11}^{(2)}\delta_1^{(3)} + \Theta_{21}^{(2)}\delta_2^{(3)})a_1^{(2)} \times (1 - a_1^{(2)}) \tag{33}$$

$$\delta_2^{(2)} = (\Theta_{12}^{(2)}\delta_1^{(3)} + \Theta_{22}^{(2)}\delta_2^{(3)})a_2^{(2)} \times (1 - a_2^{(2)}) \tag{34}$$

Persamaan (32), (33), dan (34) dapat dijadikan vektor menjadi

$$\underbrace{\begin{bmatrix}\delta_0^{(2)}\\ \delta_1^{(2)}\\ \delta_2^{(2)}\end{bmatrix}}_{\delta^{(2)}} = \underbrace{\begin{bmatrix}\Theta_{10}^{(2)} & \Theta_{20}^{(2)}\\ \Theta_{11}^{(2)} & \Theta_{21}^{(2)}\\ \Theta_{12}^{(2)} & \Theta_{22}^{(2)}\end{bmatrix}}_{(\Theta^{(2)})^T} \underbrace{\begin{bmatrix}\delta_1^{(3)}\\ \delta_2^{(3)}\end{bmatrix}}_{\delta^{(3)}} .* \underbrace{\begin{bmatrix}a_0^{(2)}\\ a_1^{(2)}\\ a_2^{(2)}\end{bmatrix}}_{a^{(2)}} .* \underbrace{\begin{bmatrix}1-a_0^{(2)}\\ 1-a_1^{(2)}\\ 1-a_2^{(2)}\end{bmatrix}}_{1-a^{(2)}}. \tag{35}$$

Kembali elemen $\delta_0^{(2)}$ pada $\delta^{(2)}$ kita buang karena $\delta_0^{(2)}$ adalah error atau selisih pada *bias* $a_0^{(2)}$ yang bukan neuron sebenarnya.

Selanjutnya, kita definisikan

$$\Delta^{(3)} = \begin{bmatrix}\Delta_{10}^{(3)} & \Delta_{11}^{(3)} & \Delta_{12}^{(3)}\\ \Delta_{20}^{(3)} & \Delta_{21}^{(3)} & \Delta_{22}^{(3)}\end{bmatrix} = \begin{bmatrix}0 & 0 & 0\\ 0 & 0 & 0\end{bmatrix} \tag{36}$$

Kemudian untuk setiap *train instance* dari $i = 1, 2, \dots, m$, $\Delta^{(3)}$ akan mengakumulasi gradient ($\delta \times$ input) dari setiap *train instance* sebagai berikut:

$$\underbrace{\begin{bmatrix}\Delta_{10}^{(3)} & \Delta_{11}^{(3)} & \Delta_{12}^{(3)}\\ \Delta_{20}^{(3)} & \Delta_{21}^{(3)} & \Delta_{22}^{(3)}\end{bmatrix}}_{\Delta^{(3)}} = \begin{bmatrix}\Delta_{10}^{(3)} & \Delta_{11}^{(3)} & \Delta_{12}^{(3)}\\ \Delta_{20}^{(3)} & \Delta_{21}^{(3)} & \Delta_{22}^{(3)}\end{bmatrix} + \begin{bmatrix}\delta_1^{(4)} & \delta_1^{(4)}a_1^{(3)} & \delta_1^{(4)}a_2^{(3)}\\ \delta_2^{(4)} & \delta_2^{(4)}a_1^{(3)} & \delta_2^{(4)}a_2^{(3)}\end{bmatrix} \tag{37}$$

$$= \underbrace{\begin{bmatrix}\Delta_{10}^{(3)} & \Delta_{11}^{(3)} & \Delta_{12}^{(3)}\\ \Delta_{20}^{(3)} & \Delta_{21}^{(3)} & \Delta_{22}^{(3)}\end{bmatrix}}_{\Delta^{(3)}} + \underbrace{\begin{bmatrix}\delta_1^{(4)}\\ \delta_2^{(4)}\end{bmatrix}}_{\delta^{(4)}} \underbrace{\begin{bmatrix}1 & a_1^{(3)} & a_2^{(3)}\end{bmatrix}}_{(a^{(3)})^T} \tag{38}$$

Kembali, kita definisikan

$$\Delta^{(2)} = \begin{bmatrix}\Delta_{10}^{(2)} & \Delta_{11}^{(2)} & \Delta_{12}^{(2)}\\ \Delta_{20}^{(2)} & \Delta_{21}^{(2)} & \Delta_{22}^{(2)}\end{bmatrix} = \begin{bmatrix}0 & 0 & 0\\ 0 & 0 & 0\end{bmatrix} \tag{39}$$

Kemudian kembali untuk setiap *train instance* dari $i = 1, 2, \dots, m$, $\Delta^{(2)}$ akan mengakumulasi gradient ($\delta \times$ input) dari setiap *train instance* sebagai berikut:

$$\underbrace{\begin{bmatrix}\Delta_{10}^{(2)} & \Delta_{11}^{(2)} & \Delta_{12}^{(2)}\\ \Delta_{20}^{(2)} & \Delta_{21}^{(2)} & \Delta_{22}^{(2)}\end{bmatrix}}_{\Delta^{(2)}} = \begin{bmatrix}\Delta_{10}^{(2)} & \Delta_{11}^{(2)} & \Delta_{12}^{(2)}\\ \Delta_{20}^{(2)} & \Delta_{21}^{(2)} & \Delta_{22}^{(2)}\end{bmatrix} + \begin{bmatrix}\delta_1^{(3)} & \delta_1^{(3)}a_1^{(2)} & \delta_1^{(3)}a_2^{(2)}\\ \delta_2^{(3)} & \delta_2^{(3)}a_1^{(2)} & \delta_2^{(3)}a_2^{(2)}\end{bmatrix} \tag{40}$$

$$= \underbrace{\begin{bmatrix}\Delta_{10}^{(2)} & \Delta_{11}^{(2)} & \Delta_{12}^{(2)}\\ \Delta_{20}^{(2)} & \Delta_{21}^{(2)} & \Delta_{22}^{(2)}\end{bmatrix}}_{\Delta^{(2)}} + \underbrace{\begin{bmatrix}\delta_1^{(3)}\\ \delta_2^{(3)}\end{bmatrix}}_{\delta^{(3)}} \underbrace{\begin{bmatrix}1 & a_1^{(2)} & a_2^{(2)}\end{bmatrix}}_{(a^{(2)})^T} \tag{41}$$

Kembali, kita definisikan

$$\Delta^{(1)} = \begin{bmatrix} \Delta_{10}^{(1)} & \Delta_{11}^{(1)} & \Delta_{12}^{(1)} \\ \Delta_{20}^{(1)} & \Delta_{21}^{(1)} & \Delta_{22}^{(1)} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{42}$$

Kemudian kembali untuk setiap *train instance* dari $i = 1, 2, \ldots, m$, $\Delta^{(1)}$ akan mengakumulasi gradient ($\delta \times$ input) dari setiap *train instance* sebagai berikut:

$$\underbrace{\begin{bmatrix} \Delta_{10}^{(1)} & \Delta_{11}^{(1)} & \Delta_{12}^{(1)} \\ \Delta_{20}^{(1)} & \Delta_{21}^{(1)} & \Delta_{22}^{(1)} \end{bmatrix}}_{\Delta^{(1)}} = \begin{bmatrix} \Delta_{10}^{(1)} & \Delta_{11}^{(1)} & \Delta_{12}^{(1)} \\ \Delta_{20}^{(1)} & \Delta_{21}^{(1)} & \Delta_{22}^{(1)} \end{bmatrix} + \begin{bmatrix} \delta_1^{(2)} & \delta_1^{(2)} a_1^{(1)} & \delta_1^{(2)} a_2^{(1)} \\ \delta_2^{(2)} & \delta_2^{(2)} a_1^{(1)} & \delta_2^{(2)} a_2^{(1)} \end{bmatrix} \tag{43}$$

$$= \underbrace{\begin{bmatrix} \Delta_{10}^{(1)} & \Delta_{11}^{(1)} & \Delta_{12}^{(1)} \\ \Delta_{20}^{(1)} & \Delta_{21}^{(1)} & \Delta_{22}^{(1)} \end{bmatrix}}_{\Delta^{(1)}} + \underbrace{\begin{bmatrix} \delta_1^{(2)} \\ \delta_2^{(2)} \end{bmatrix}}_{\delta^{(2)}} \underbrace{\begin{bmatrix} 1 & a_1^{(1)} & a_2^{(1)} \end{bmatrix}}_{(a^{(1)})^T} \tag{44}$$

$$= \underbrace{\begin{bmatrix} \Delta_{10}^{(1)} & \Delta_{11}^{(1)} & \Delta_{12}^{(1)} \\ \Delta_{20}^{(1)} & \Delta_{21}^{(1)} & \Delta_{22}^{(1)} \end{bmatrix}}_{\Delta^{(1)}} + \underbrace{\begin{bmatrix} \delta_1^{(2)} \\ \delta_2^{(2)} \end{bmatrix}}_{\delta^{(2)}} \underbrace{\begin{bmatrix} 1 & x_1 & x_2 \end{bmatrix}}_{x^T} \tag{45}$$

Setelah $\Delta^{(1)}$, $\Delta^{(2)}$, dan $\Delta^{(3)}$ diakumulasi untuk $i = 1, 2, \ldots, m$, saatnya kita menghitung gradient untuk $\Theta^{(1)}$, $\Theta^{(2)}$, dan $\Theta^{(3)}$.

$$D^{(1)} = \begin{bmatrix} \frac{1}{m}\Delta_{10}^{(1)} & \frac{1}{m}\Delta_{11}^{(1)} + \frac{\lambda}{m}\Theta_{11}^{(1)} & \frac{1}{m}\Delta_{12}^{(1)} + \frac{\lambda}{m}\Theta_{12}^{(1)} \\ \frac{1}{m}\Delta_{20}^{(1)} & \frac{1}{m}\Delta_{21}^{(1)} + \frac{\lambda}{m}\Theta_{21}^{(1)} & \frac{1}{m}\Delta_{22}^{(1)} + \frac{\lambda}{m}\Theta_{22}^{(1)} \end{bmatrix} \tag{46}$$

$$D^{(2)} = \begin{bmatrix} \frac{1}{m}\Delta_{10}^{(2)} & \frac{1}{m}\Delta_{11}^{(2)} + \frac{\lambda}{m}\Theta_{11}^{(2)} & \frac{1}{m}\Delta_{12}^{(2)} + \frac{\lambda}{m}\Theta_{12}^{(2)} \\ \frac{1}{m}\Delta_{20}^{(2)} & \frac{1}{m}\Delta_{21}^{(2)} + \frac{\lambda}{m}\Theta_{21}^{(2)} & \frac{1}{m}\Delta_{22}^{(2)} + \frac{\lambda}{m}\Theta_{22}^{(2)} \end{bmatrix} \tag{47}$$

$$D^{(3)} = \begin{bmatrix} \frac{1}{m}\Delta_{10}^{(3)} & \frac{1}{m}\Delta_{11}^{(3)} + \frac{\lambda}{m}\Theta_{11}^{(3)} & \frac{1}{m}\Delta_{12}^{(3)} + \frac{\lambda}{m}\Theta_{12}^{(3)} \\ \frac{1}{m}\Delta_{20}^{(3)} & \frac{1}{m}\Delta_{21}^{(3)} + \frac{\lambda}{m}\Theta_{21}^{(3)} & \frac{1}{m}\Delta_{22}^{(3)} + \frac{\lambda}{m}\Theta_{22}^{(3)} \end{bmatrix} \tag{48}$$

Terakhir, kita akan update $\Theta^{(1)}$, $\Theta^{(2)}$, dan $\Theta^{(3)}$ seperti berikut:

$$\Theta^{(1)} = \Theta^{(1)} - \alpha \times D^{(1)} \tag{49}$$

$$\Theta^{(2)} = \Theta^{(2)} - \alpha \times D^{(2)} \tag{50}$$

$$\Theta^{(3)} = \Theta^{(3)} - \alpha \times D^{(3)} \tag{51}$$

References

Stanford ML (2020a). Backpropagation algorithm. `https://www.coursera.org/learn/machine-learning/supplement/pjdBA/backpropagation-algorithm`. Accessed: 2020-04-09.

Stanford ML (2020b). Neural networks: Learning. `https://www.coursera.org/learn/machine-learning/home/week/5`. Accessed: 2020-04-09.