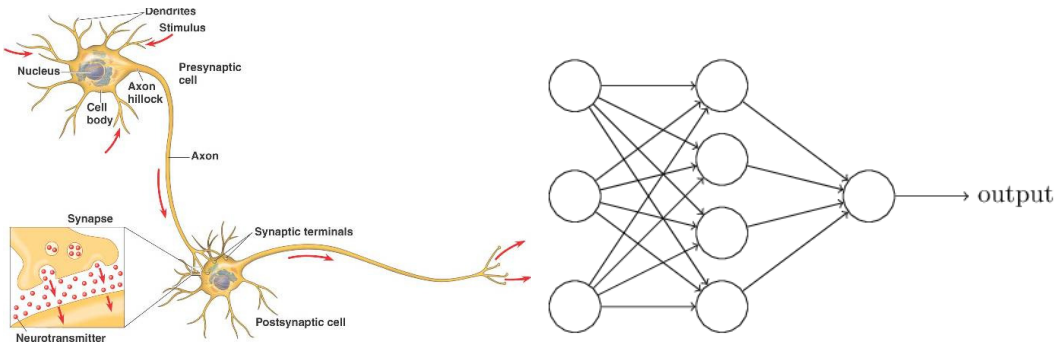
Artificial Intelligence
Kuliah 4: Jaringan Saraf Tiruan

Ali Akbar Septiandri
Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Al Azhar Indonesia
aliakbars@live.com

1 Jaringan Saraf Tiruan

Seperti dijelaskan dalam namanya, *artificial neural networks* atau jaringan saraf tiruan sejatinya adalah usaha para saintis untuk meniru cara kerja jaringan saraf manusia. Idenya, sebuah model regresi logistik dianggap sebagai sebuah **neuron** dan setiap neuron tersebut akan ditumpuk sehingga keluaran dari suatu neuron akan dijadikan sebagai masukan bagi neuron lainnya. Maka, neuron yang berada di bagian paling ujung akan bertindak sebagai neuron untuk prediksi yang dianalogikan seperti aktuator pada sistem saraf manusia. Ilustrasinya dapat dilihat seperti pada Gambar 1. Ide ini pertama kali dicetuskan oleh McCulloch dan Pitts [1].

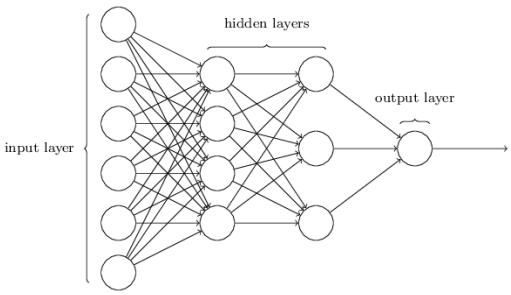


(a) Jaringan saraf manusia (b) Jaringan saraf tiruan

Gambar 1: Model matematis yang dibuat disusun menyerupai jaringan saraf manusia

Dalam perkembangannya, terminologi yang digunakan untuk menggambarkan jaringan saraf tiruan ini menjadi bervariasi. Seperti terlihat pada Gambar 2, sebuah jaringan saraf tiruan terdiri dari **input, hidden,** dan **output layers**. Tiap *layer* atau lapisan ini terdiri dari sejumlah neuron atau unit. Terkadang, satu unit ini dikenal juga dengan nama *perceptron* sehingga jaringan saraf tiruan seperti pada Gambar 2 sering juga disebut sebagai **Multilayer Perceptron** (MLP). Seperti halnya regresi logistik, neuron pada *output layer* menggunakan **fungsi aktivasi** berupa sigmoid $\sigma(z)$.

Model jaringan saraf tiruan ini sangat fleksibel dalam arti bahwa jumlah *hidden layers* dan *hidden units* merupakan nilai yang bisa kita tentukan sendiri sebagai ***hyperparameters***. Bahkan, model ini bisa menjadi *universal approximator* untuk berbagai fungsi kontinu hanya dengan dua *hidden layers* saja [3]. Namun, seperti halnya regresi logistik, permasalahannya ada pada penentuan nilai bobot $\mathbf{w}$.



Gambar 2: Pembagian lapisan pada jaringan saraf tiruan [2]

2 Mencari Nilai Bobot

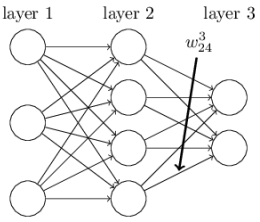
2.1 Gradient Descent

Untuk memudahkan notasi, kita bisa membuat proses perkalian antara dua lapisan dengan bobotnya menjadi

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

dengan w_{kj} adalah bobot dari neuron j di lapisan $l-1$ ke neuron k di lapisan l yang digambarkan dengan garis; x_j adalah masukan neuron j; dan y_k adalah keluaran neuron y_k. Pada Gambar 3, w_{24}^3 berarti bobot dari neuron 4 dari lapisan 2 ke neuron 2 di lapisan 3. Dalam notasi matriks-vektor, keluaran di tiap lapisan menjadi

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



Gambar 3: Notasi penulisan bobot

Seperti halnya regresi logistik, nilai $\mathbf{W}$ hanya bisa dicari dengan metode optimasi numerik. Salah satu cara yang umum digunakan adalah dengan *gradient descent*. Sedikit berbeda dengan regresi logistik, fungsi galat yang dioptimasi terkadang diubah menjadi fungsi galat entropi-silang (***cross-entropy error function***) yang didefinisikan sebagai

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

sehingga nilai gradiennya menjadi

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

2.2 Backpropagation

Bagusnya, metode ini dapat diterapkan bukan hanya pada lapisan terakhir, tetapi juga pada lapisan sebelumnya. Jadi, error yang ditemukan pada lapisan terakhir akan dirambatkan ke lapisan sebelumnya hingga ke lapisan pertama. Nilainya akan disesuaikan dengan fungsi aktivasi yang digunakan pada tiap lapisan. Sebagai contoh, jika fungsi aktivasi yang digunakan adalah sigmoid, maka gradiennya

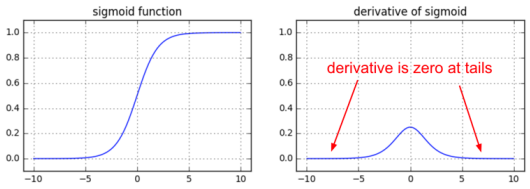
dapat dihitung dengan

$$\frac{\partial E^n}{\partial w_{ji}^{(1)}} = \underbrace{\left(\sum_{c=1}^{k} \delta_c^{(2)} w_{cj}^{(2)}\right) h_j^{(1)}(1 - h_j^{(1)})}_{\delta_j^{(1)}} \cdot x_i$$

dengan $h_j = \sigma(w_{ji}x_i)$ dan $\delta_c = y_c - t_c$ dari

$$\frac{\partial E^n}{\partial w_{kj}^{(2)}} = \underbrace{(y_k - t_k)}_{\delta_k^{(2)}} \cdot h_j^{(1)}$$

Sayangnya, algoritma ini mungkin terjebak pada solusi **optimum lokal** dan terkadang juga mengalami masalah ***vanishing gradients***. Kasus *vanishing gradients* terjadi karena perambatan error ke lapisan awal membuat nilai errornya mengecil sepanjang perambatan. Ini disebabkan fungsi sigmoid yang biasa digunakan sebagai fungsi aktivasi mudah sekali jenuh karena sifatnya yang asimtotik sehingga derivatifnya mendekati nol di "ekornya" [4] seperti pada Gambar 4.



Gambar 4: Fungsi sigmoid yang jenuh karena asimtotik

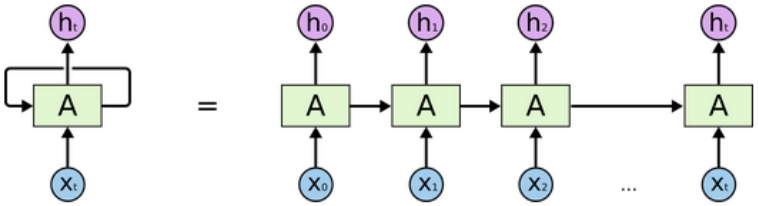
Untuk mengatasi masalah ini, kita sebaiknya menginisialisasi $\mathbf{W}$ dengan nilai yang kecil atau menggunakan fungsi aktivasi yang lain, misalnya ***rectified linear unit*** (ReLU) yang didefinisikan sebagai

$$g(z) = \max(0, z)$$

Namun, fungsi ini juga tidak terlepas dari masalah. Jika keluarannya selalu negatif, maka yang terjadi adalah *dead units*. Oleh karena itu, ada banyak pengembangan lebih lanjut untuk proses *gradient descent* dan *backpropagation* secara umum yang tidak akan dibahas di kuliah ini.

3 Variasi dan Aplikasi Jaringan Saraf Tiruan

Jika diperhatikan kembali, model arsitektur jaringan saraf tiruan di atas merupakan **graf berarah asiklik** (*directed acyclic graph*). Perkalian bobot dengan masukan selalu bergerak "maju" sehingga disebut juga sebagai ***feedforward neural networks***. Hal ini menyebabkan model ini sulit untuk mendapatkan hubungan temporal, misalnya seperti proses memaknai suatu kalimat saat urutan kata menjadi penting. Oleh karena itu, dibuatlah model *recurrent neural networks* yang membuat grafnya menjadi siklik. Ilustrasinya bisa dilihat pada Gambar 5.



Gambar 5: Recurrent neural networks jika dilihat secara sekuensial [5]

Di sisi lain, saat ini *neural networks* yang juga dikenal sebagai *deep learning* dikembangkan untuk menangani data yang bersifat tidak terstruktur, misalnya gambar atau suara. Untuk mengolah

gambar, dikenalkan penggunaan *convolutional layers* dan *max pooling layers*. Arsitektur dengan menggunakan lapisan tersebut dikenal juga dengan nama *convolutional neural networks*. Beberapa contoh aplikasinya dapat dilihat pada [6, 7, 8].

References

[1] McCulloch, W.S. and Pitts, W., 1943. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics, 5*(4), pp.115-133.

[2] Nielsen, M.A., 2015. Neural networks and deep learning.

[3] Cybenko, G., 1988. Continuous valued neural networks with two hidden layers are sufficient.

[4] Karpathy, A., 2016. Yes you should understand backprop. Available at: `https://medium.com/@karpathy/yes-you-should-understand-backprop-e2f06eab496b#.70lzt4tw2`.

[5] Olah, C., 2015. Understanding LSTM Networks. Available at: `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`.

[6] Krizhevsky, A., Sutskever, I. and Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097-1105).

[7] Hinton, G., Deng, L., Yu, D., Dahl, G.E., Mohamed, A.R., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T.N. and Kingsbury, B., 2012. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine, 29*(6), pp.82-97.

[8] Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S. and Dean, J., 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems* (pp. 3111-3119).