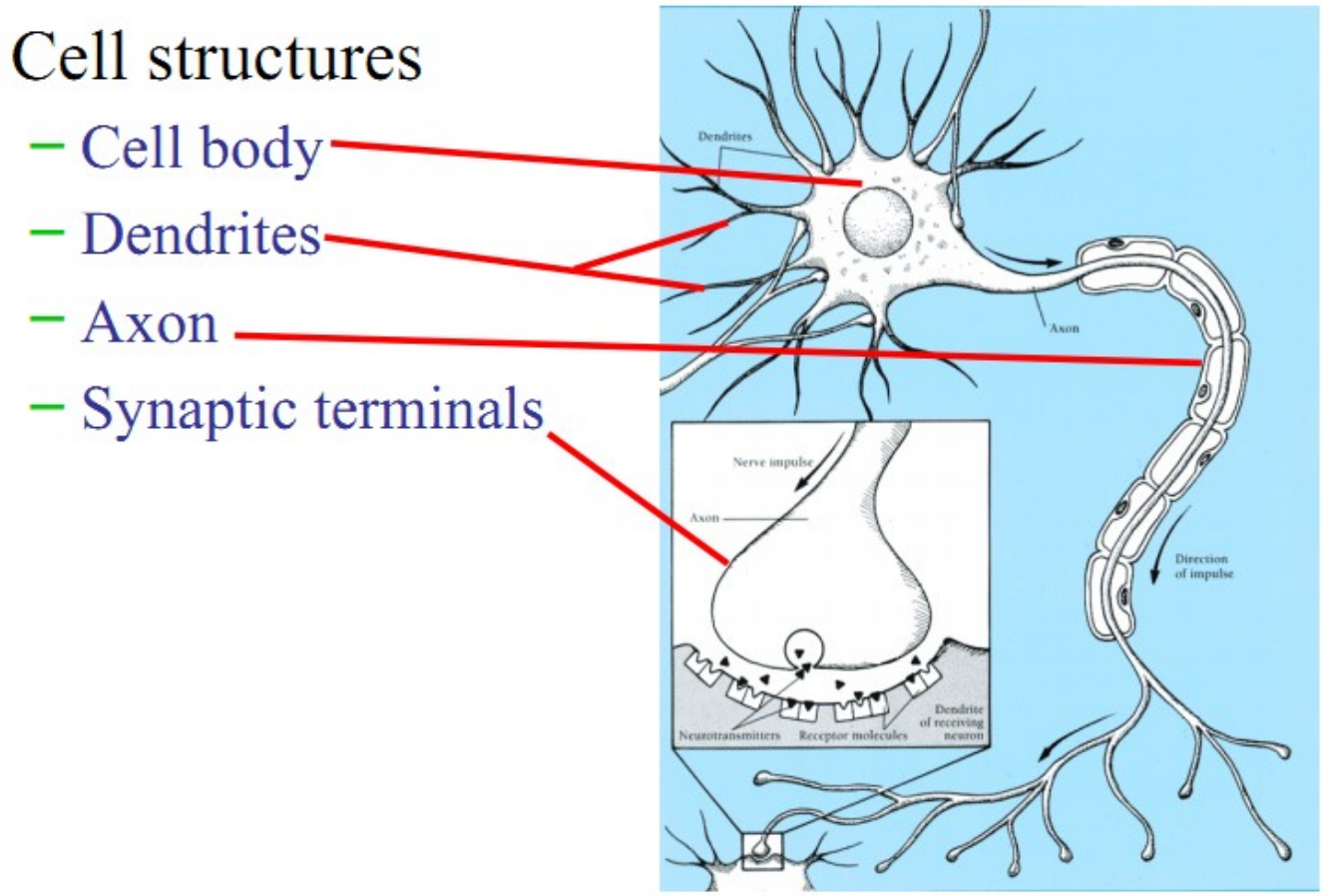
Pertemuan 6

Neural Network

NN (neural network) atau dikenal dengan nama lainnya **ANN (artificial neural network)** dan **JST (jaringan syaraf tiruan)** adalah sebuah model machine learning yang berdasarkan syaraf manusia.

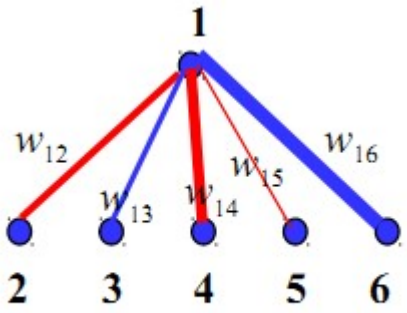
Awalnya dikembangkan dari perceptron, sebuah algoritma sederhana yang meniru cara kerja syaraf. Syaraf manusia bekerja dari impuls-impuls yang diberikan oleh neuron sebelahnya.



Ada potensial aksi (action potential) berupa sinyal listrik yang mengalir dari satu sel ke sel lainnya melalui sel syaraf. Awalnya dari sel (misalnya sel sensorik seperti kulit), jika mendapatkan rangsangan dia akan merespon dengan memunculkan lonjakan sinyal (spike). Sinyal ini mengalir melalui akson (axon) dan menghasilkan neurotransmitter. Neurotransmitter ini diterima oleh sel lain dan jika melewati batas tertentu, maka sel penerimanya akan menghasilkan potensial aksi yang baru dan diteruskan ke sel lainnya lagi.

Sel syaraf belajar dari distribusi neurotransmitter ini dan menyesuaikan diri. Makanya jika kita sering menerima rangsangan tertentu, toleransi kita jadi lebih tinggi. Misal, sering makan makanan panas, jadinya mulut dan lidah lebih kuat nahan panas. Sering menerima rangsangan otak misalnya diharuskan memecahkan puzzle atau soal, maka akan jadi terbiasa dan lebih cerdas.

Skema ini dimodelkan dalam perhitungan komputer dalam bentuk lapisan lapisan perseptron. Sebuah perseptron adalah model seperti gambar di bawah ini



tiap titik mewakili sebuah sel syaraf dan garis penghubungnya adalah jaringannya (yang menghubungkan antara satu sel dengan sel lainnya) yang mengalirkan sebuah nilai (mewakili neurotransmitter). Jika nilai input pada suatu sel melebihi batas tertentu, maka sel tersebut akan mendapatkan sebuah nilai (action potential).

Input pada suatu sel dihitung dari jumlah nilai sel sel lain yang menuju ke sel tersebut.

$$net_j = \sum_i w_{ji} o_i$$

Wji (w adalah weight = bobot) adalah nilai jaringan syaraf yang menghubungkan sel i ke sel j (sel tujuannya dituliskan di index yang depan). Nilai ini yang menunjukkan respon jaringan syaraf kita terhadap hal tertentu, inilah **model** yang nanti akan kita dapatkan di akhir training. JST bekerja dengan menyesuaikan weight ini ke data training kita.

Oi adalah action potential yang dihasilkan oleh sel i. Hasil perkalian w dan o mewakili neurotransmitter yang dihasilkan oleh sel tertentu ke sel tujuannya. Nah, total dari semua inputan inilah (net j) yang akan menentukan apakah sel tujuan akan merespon dengan memberikan sinyal ke sel selanjutnya atau tidak. Sama seperti sel yang menghasilkan action potential baru jika input neurotransmitternya melewati batas tertentu (threshold).

Action potentialnya ditentukan dengan sebuah fungsi lain yang disebut **activation function**.

$$o_j = \begin{matrix} 0 \text{ if } net_j < T_j \\ 1 \text{ if } net_i \geq T_j \end{matrix}$$

Ini adalah contoh activation function yang sederhana, istilahnya step (berbentuk tangga). Jika total inputannya (net j) di bawah batas (Tj) maka tidak ada yang dilakukan oleh sel (outputnya 0). Tapi jika mencapai atau melebihi batas tersebut, outputnya 1. Jadi action potentialnya cuma memiliki 2 buah nilai saja, 0 atau 1.

Selain itu juga ada banyak activation function lainnya, seperti logistic, tanh, atau gaussian. Informasi lebih lanjut bisa dibaca di buku/internet.

Training dilakukan dengan memberikan input (data) dan output (label) ke dalam perseptronnya. Misalnya data inputannya berupa kalimat dengan 4 kata, dan kita akan menentukan apakah kalimat tersebut berrima (seperti pantun atau puisi, kata terakhirnya mirip satu sama lain) atau tidak. Berarti kita butuh data training berupa pasangan antara kalimat 4 kata (variabel x) dan labelnya yang menandakan kalimat tersebut berima atau tidak (variabel y).

contoh datanya :

x	y
jalan pagi sehat hati	(true)
kepala botak kelapa batok	(true)
satu dua tiga empat	(false)

dst

Weight dialokasikan secara random di awal perhitungan. Perseptron akan menggunakan algoritma yang mengupdate nilai weight tersebut secara iteratif hingga didapatkan nilai yang tepat (mewakili data inputannya)

Weightnya diupdate menggunakan fungsi

$$w_{ji} = w_{ji} + \eta(t_j - o_j)o_i$$

myu (simbol yang mirip huruf n) adalah **learning rate**, yaitu nilai yang kita tentukan untuk training modelnya. tj adalah output yang diberikan oleh data (teacher specified output), oj adalah nilai output yang dihasilkan oleh activation function, dan oi adalah nilai action potential yang diberikan oleh sel asalnya.

Jadi misalnya awalnya kita punya data yang di atas, kita konversi ke sebuah nilai dengan metode tertentu. (dalam kasus ini huruf, kita bisa ubah menggunakan word2vec atau lainnya).

Misalnya kalimat pertama jadi vektor (10, 4, 27, 8), kalimat kedua jadi (6, 1, 5, 2), dan yang ketiga jadi (19, 22, 7, 11). Weight awal dirandom ternyata nilainya 1 semua. Model yang kita buat berbentuk graf star dengan **sel 5 adalah sel tujuannya** dan **sel 1 s.d. 4 adalah inputannya** , maka w51 s.d. w54 nilainya 1.

perceptron akan menghitung nilai dari inputan dikalikan weightnya. Dengan data pertama misalnya
net 5 = 10*1 + 4*1 + 27*1 + 8*1 = 49

lalu dia akan memeriksa apakah nilainya melewati batas (threshold) yang ditentukan. Misalnya Tj = 30. Jika melewati maka kita dapatkan nilai outputnya berupa o5 = 1.

Nilai ini dibandingkan dengan nilai yang diberikan oleh data (kalimat pertama nilai y nya true = 1, jadi t1 = 1). Karena hasil perhitungannya sama dengan label datanya, maka tidak ada perubahan (modelnya dianggap sudah tepat untuk data tersebut).

Selanjutnya dia akan menggunakan data kedua. Total nilai inputannya dihitung (6*1+1*1+5*1+2*1 = 14), lalu dimasukkan ke activation function (14 < 30, di bawah threshold, maka nilai outputnya 0), lalu dibandingkan dengan label inputnya (karena kalimat 2 dianggap berrima, t2 = true = 1).

Jika ada perbedaan antara nilai output yang dihitung (oj) dengan label datanya (tj) maka koreksi dilakukan. Jika nilainya terlalu besar akan dikecilkan bobotnya (weight). Jika nilainya terlalu kecil akan dibesarkan.

Misal kita set learning ratenya = 0.5 . Maka koreksi akan dilakukan terhadap seluruh weight yang menuju sel 5 mengikuti aturan di atas.

Wji = wji + myu * (tj-oj) * oi

w51 akan berubah dari 1 menjadi 1.5
1 + 0.5 * (1-0) * 1 = 1.5

karena perhitungannya tadi di bawah data sebenarnya (output yang dihitung = 0 sedangkan datanya menuntut 1), maka weightnya diperbesar. weight yang lain pun demikian. Akan diupdate sesuai nilai koreksinya.

Thresholdnya juga perlu diperbaiki untuk mengkompensasi perubahan weight ini.

$$T_j = T_j - \eta(t_j - o_j)$$

Dan seterusnya hingga didapatkan model yang cukup konvergen. Jadi weight yang kita miliki di perceptronnya (model) bisa mewakili sebagian besar data inputannya (akurasinya tinggi).

Satu kali iterasi ke seluruh data training disebut satu **epoch**. Training biasanya diset sampai sejumlah epoch tertentu agar datanya tidak overfitting (bagus pada saat dites ke data yang sudah ada, tapi untuk data yang baru akurasinya sangat buruk).

Neural network biasanya tidak hanya menggunakan sebuah lapisan (layer) saja seperti penjelasan perceptron di atas. Tetapi perhitungan di tiap lapisan dan tiap selnya kurang lebih sama. Yang membedakan adalah penentuan metode yang digunakan (algoritma, activation function, dll) dan hyper parameternya (learning rate, threshold, dll).

Kasus training menggunakan neural network bisa dilihat juga sebagai kasus gradient descend (pencarian nilai minimum) atau hill climbing (pencarian nilai maksimum).

JST bisa menggunakan beberapa hidden layer (layer diantara layer input dan layer output) untuk menghitung bagian dari modelnya. Banyak sel pada layer inputnya disesuaikan dengan besarnya vektor inputan (bukan banyaknya input). Seperti pada contoh di atas, besar vektor inputnya 4, maka layer inputnya kita set untuk memasukkan 4 data tersebut. Layer outputnya disesuaikan dengan besarnya vektor output (ada berapa parameter pada labelnya). Pada contoh di atas outputnya hanya ada 1 parameter, yaitu berima atau tidak, karena itu layer outputnya memiliki hanya 1 buah sel. Hidden layer dapat kita modifikasi sesuai kebutuhan.

Makin banyak sel pada suatu layer dan makin banyak layernya belum tentu menghasilkan model yang lebih bagus. Bisa saja redundan karena datanya terlalu sederhana. Tapi untuk kasus yang besar biasanya layer yang digunakan banyak agar bisa memeriksa fitur-fitur yang tidak terlihat manusia biasa. Konsekwensinya adalah perhitungan yang lebih banyak dan lama. Lebih lanjut bisa baca tentang deep learning.

Silahkan baca lebih jauh terkait neural networknya beserta variannya (dari slide atau sumber lain) dan didiskusikan di grup kalau ada pertanyaan.