FUNDAMENTAL NOTES #13: CARA TERMUDAH MEMAHAMI THE BASICS OF AI UNTUK PARA JURUTERA

(AND A TRIBUTE TO HOPFIELD AND HINTON)

by Ir Dr Airil Yasreen Mohd Yassin

PRELUDE

*Siapa baca sampai habis, insyaAllah dia akan dapat kefahaman fundamental on the basic concepts of AI that is, down to the maths.

*Dia juga kemudiannya, boleh nampak kenapa dunia berebut bina AI Data Centre merata dan kenapa wujudnya GPU war antara say, NVIDIA vs AMD, dan chip war antara US, CHINA, TSMC, Samsung etc.

*Untuk engineer2 pula, siapa baca sampai habis, dia boleh ada measure on how AI akan affect engineering works dan how should we prepare.

*Note ini telah jadi full-blown artikel so saya dah PDF kan. Tapi untuk meneruskan siri fundamentals note sebelum ini, saya share juga di FB ini. Siapa yang ingin baca PDF, boleh click link di bawah ni untuk download. Kalau rasa bermanfaat, boleh share PDF tu dengan kawan2 dan kenalan.

https://github.com/msnm-official/ai_modules

INTRODUCTION

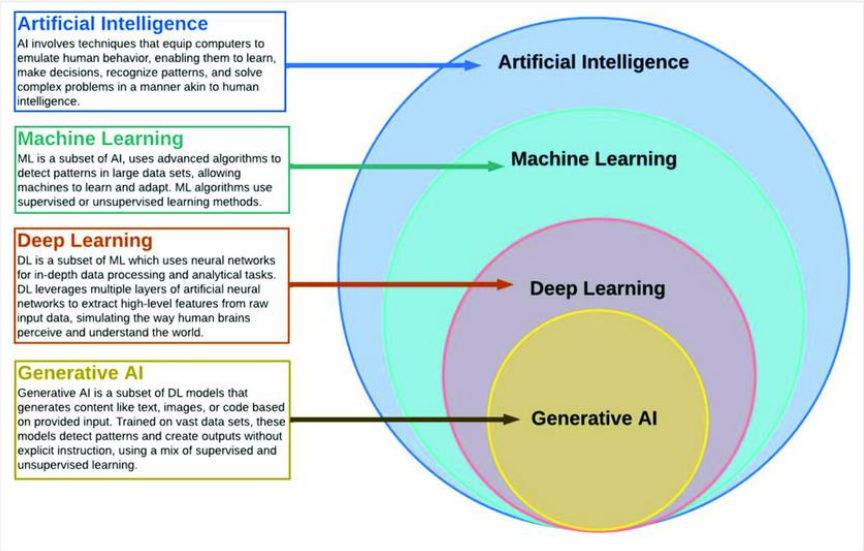
Dalam post Sabtu lepas, saya ada janjikan untuk buat satu fundamentals note se"simple" mungkin on AI untuk sesiapa yang ingin memahami the basic concepts and the innerworking of AI supaya nanti dapat berikan input yang relevan kepada masyarakat (bawah theme Engineers in Society). Ini note nya.

Tapi seperti yang saya selalu katakan, untuk memahami sains dan engineering, kita tidak boleh elak dari lihat equations dan gambarajah. Saya akan guna equations yang se"simple" mungkin dan tunjukkan juga, how these equations tidak lah asing sebenarnya, they are just other forms of equation2 yang kita already familiar as engineers.

So, note ni supposed nya "simple", ia cuma perlu dibaca perlahan2.

Also, kerana AI ni ada banyak jenis, untuk "simple"kan discussion, first thing to do, kita mesti focus kan scopenya. Kita akan focus kepada jenis yang paling vibe sekarang, iaitu yang menjadi asas kepada ChatGPT, Google Translate dll. Kita akan focus kepada Deep Learning (DL).

Refer Foto 1, saya ambil dua carta tersebut dari sources yang berbeza (references diberikan di ruangan komen). Kita akan nampak, AI adalah payung yang besar. Di bawahnya ada Machine Learning (ML) yang menjadi payung pula kepada Deep Learning. Dan akhir sekali, AI seperti ChatGPT, adalah Generative AI yang berada di bawah payung Deep Learning. So kita akan focus kepada Deep Learning.



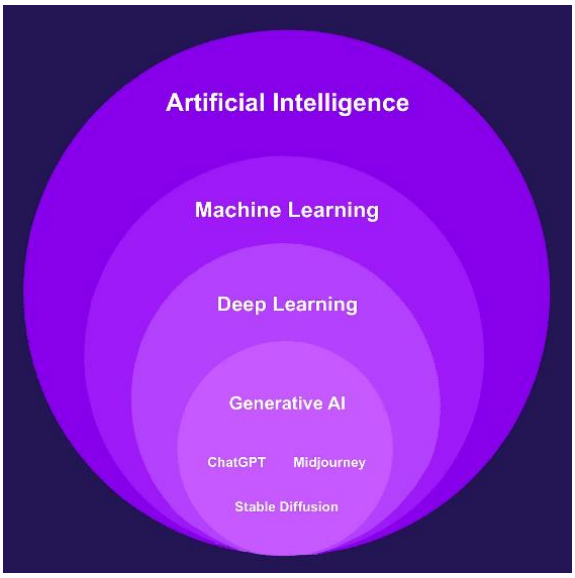


Foto 1: Type and sets of Deep Learning [1, 2]

Deep learning pula adalah multi-layered Neural Network dan ia ada beberapa forms, antaranya FFNN, CNN, RNN, Attention, dan Tranformer. Tetapi asasnya adalah Feedforwad Neural Network (FFNN). So kita akan focus kepada FFNN.

So far ok tak? Baca pelan2 k dan tak payah risau nama2 kata atas ni. Yang penting points di bawah ni je, iaitu:

i. AI ada banyak forms

ii. AI yang hot sekarang basically originated from the tree of Machine Learning/Deep Learning

iii. Deep Learning asasnya Feedforward Neural Network (FFNN)

iv. Kalau kita faham sedikit sebanyak FFNN, insyaAllah kita faham sedikit sebanyak the whole vibes of AI sekarang ni.

By the way, Deep Learning is so big right now, Nobel Prize winners in physics yang diumumkan kelmarin pun adalah pengasasnya, John Hopfield dan Geoffrey Hinton. So saya juga ambil peluang menulis write-up ini sebagai tribute kepada kejayaan mereka.

Ok let's proceed.

THE NETWORK ARCHITECTURE

So, apakah the main idea of Deep Learning sehingga it has worked very succesfully especially dalam image processing and large-language model (LLM)?

The main ideanya adalah menghubungkan input dan output secara mathematics. Once hubungan ini dapat diestablishedkan, kita boleh buat prediction iaitu...apakah outputnya, jika diberikan satu input yang baru.

Sound familiarkan? It sounds like any other scientific or engineering activities. Memang pun.

Ok, sekarang kita refer kepada Foto 2. Ia tunjukkan the general architecture of Deep Learning. Tapi jangan risau figure yang "berserabut" ni. Yang ni saya nak tunjuk the general terms dan components je. Nanti, untuk discuss detailsnya, saya akan guna arrangement yang jauh lebih mudah (dalam "Worked Example" nanti) tapi boleh represent the idea, say up to 70-80% of the general architecture. So jangan risau.

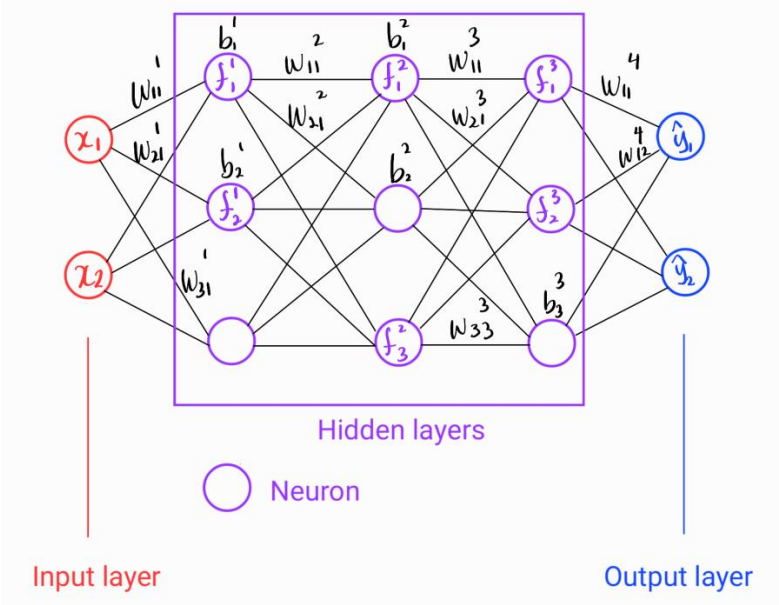


Foto 2: The network architecture

Dalam Foto 2, kita boleh nampak layer of input (bulat merah) dan layer of output (bulat biru).

Input ni, contohnya, boleh jadi pixels value gambar2 binatang. Output pula adalah label atau ID kepada binatang2 tu, kucing ke, harimau ke, lembu ke and so on.

Layer input dan output ni pula disambungkan oleh hidden layers (kotak purple). Haaa...hidden layers ni la the "mathematics" yang hubungkan input dan output, dan ini la yang kita nak establishkan.

Selepas kita dapat establishkan "mathematics" atau hidden layers tu, itu la waktunya nanti bila masuk gambar kucing (as input), network atau system kita terus tahu, itu kucing (as output)!

Ok, this is the main purpose of any Deep Learning, that is, to predict output for a given input.

Tapi macamana nak establishkan "mathematics" tu? Kata lainnya, macamana nak establishkan hidden layers tu? Ini persoalannya sekarang.

Ok, seperti yang kita boleh lihat, hidden layers dalam kotak purple tu ada layer2 of neurons.

Setiap neuron ni akan keluarkan (atau spit out) what we called, activation functions (i.e., $f_1, f_2 \ldots$). Kenapa dia keluarkan activation function ni?

Bukan ke tadi kita kata nak establishkan the "mathematics" antara input dan output? So "mathematics" ni mesti la dalam bentuk functions. Pasal tu lah neuron2 ni akan keluarkan functions, ok?

Tapi siapa yang decide functions tu? Developer atau engineers lah (mereka yang bangunkan network tu) yang akan decide. Dia boleh pilih macam2 activation functions, antaranya sigmoid, tanh, ReLU, softmax dll.

Tapi, tak kisah la functions mana yang dipilih, yang penting kita tahu, niat kita adalah untuk establishkan mathematical functions antara input dan output, pasal tu neuron2 akan keluarkan ("spit out") function. Ok tak ni?

Now, semua mathematical functions dalam dunia ni perlukan coefficients, kan? Contohnya polynomial di bawah ni:

$$y = a_1 + a_2 x + a_2 x^2 + \cdots \qquad (1)$$

di mana a1, a2, a3 dan seterusnya adalah coefficients.

Dalam neural network (refer Foto 2), w_{11}, w_{12}, b_1, b_2 dan seterusnya, itu lah coefficientsnya. Mereka akan berada dalam activation functions tu nanti.

Dalam terminologi DL, w (e.g., w_{11}, w_{12}, etc) dipanggil weights dan b (e.g., b_1, b_2, etc) dipanggil biases. Dan secara amnya, instead of panggil coefficients, weights dan biases ni dipanggil parameters atau network parameters.

Ok so far kita dah sebut atau bincang terminologi2 penting dalam Deep Learning. Nanti bila baca atau sembang tentang DL, kita dah familiar dengan terminologi2 penting berikut:

- network architecture

- input dan out layers

- hidden layers

- neurons

- activation functions (sigmoid, ReLU etc.)

- parameters (weights and biases)

Good job! Let's proceed tapi baca pelan2 k. Sekarang kita bincang process nya. First sekali, seperti yang kita tahu, dalam AI ni, kita perlu ada banyak sets of input dan output. Kenapa?

Sebab, seperti yang AKAN kita bincangkan, establishing the "mathematics" (atau hidden layers) juga bermaksud, kita kena "train" the network. "Train" atau "training" ini pun satu terminologi penting dalam DL atau AI.

Banyak AI data centre dibina di seluruh dunia adalah untuk process "training" ni, so kena ingat dan faham terminologi ni.

So, saya buatkan satu sub-heading untuknya seperti di bawah.

TRAINING THE NETWORK

Training ni bermula bila kita insert the first set of input kedalam network. Input ni akan masuk ke hidden layer yang pertama dimana ia akan didarabkan dengan weight, kemudian akan ditambah dengan bias kemudian akan masuk kedalam activation function neuron2 pada layer pertama.

Kemudian, activation functions dari neuron layer pertama akan melalui process yang sama.

Dia akan keluar dari hidden layer pertama dan masuk ke hidden layer kedua dimana ia akan didarabkan dengan weight lagi, kemudian ditambah dengan bias lagi dan masuk ke dalam activation function neuron2 layer kedua.

Process ini akan diteruskan bergantung kepada sebanyak mana hidden layers di dalam network tersebut sehingga lah ia sampai ke output layer dan akhirnya mengeluarkan output.

Ok, tapi untuk process ini mengeluarkan output dalam bentuk numerical (selalunya antara 0 hingga 1), value2 numerical perlu diberikan terlebih dahulu kepada parameters (weights and biases).

Secara general, initial values untuk parameters ini, kita "teka" je dulu, atau kita decide randomly (tapi in practice, we should have some measures in what range selalunya values of weights and biases ni berada). Formal term untuk "teka" ni kita panggil "initializing the value of the parameters". Antara algorithm nya adalah Xavier atau Kaiming.

Now, disebabkan value2 parameters ni kita "teka" je in the first run, output yang terhasil bukan lah output yang tepat atau yang sebenar. Maka, akan ada ERROR antara output yang kita kira from the first run, berbanding output yang sebenar (yang kita memang dah ada dalam possession kita).

Error inilah yang perlu kita buang (atau kurangkan), dan process untuk buang/kurangkan error ini lah yang kita panggil "training".

Ok, ini yang menarik. Untuk buang/kurangkan error ini, kita akan guna variational approach, sama seperti kita guna waktu derive Finite Element based on minimum principle of total potential energy. Bermaksud, sesiapa yang biasa dengn FEM based on variational formulation akan cepat faham discussion DL ni.

(*Haa...ini la pentingnya syllabus engineering uni2 kita diubah supaya penekanan modern based on computational mechanics diberikan seperti di Imperial College, macam posting saya sebelum2 ni, nampak kan?).

Untuk buang/kurangkan error ni based on variational approach, kita kena wujudkan functional. Senang je nak wujudkan functional. Contoh, kita runkan network tu say, 100 kali. Setiap kali run,

kita square kan (kuasa dua kan) Error yang kita dapati. Then kita tambahkan kesemua seratus squared error ini dan average kan (i.e., bahagikan dengan seratus).

Bila kita buat macam ni, kita akan dapat functional berbentuk Mean Squared Error atau MSE. Tapi ada bentuk2 lain, boleh check out elsewhere.

Dalam DL, functional ini kita panggil Loss function atau Cost function. Secara persamaan, ia boleh diberikan seperti berikut:

$$Loss\ function, L = \frac{1}{n}\sum Error^2 \qquad (2)$$

where n is the number of the sets of input, juga dipanggil batch atau mini-batch.

Loss/Cost Function dan MSE ni pun terminologies yang penting dalam DL. Bila baca buku atau artikel lain nanti, akan selalu jumpa term2 ini.

Dan ingat ye, Error tak sama dengan Loss Function. Loss Functions adalah Error yang di square kan dan di average kan. Ianya adalah satu functional.

Now being a functional, the minimum of Loss function ini lah yang akan memberikan kita the "correct" value of the parameters (weights and biases).

Refer Foto 3 untuk tengok concept variational approach ni. Saya yakin, pembaca familiar dengan graph sebegini sebab dah belajar waktu sekolah menengah lagi. Then, sapa yang buat FEM based on minimum potential energy, lagi lah akan familiar.

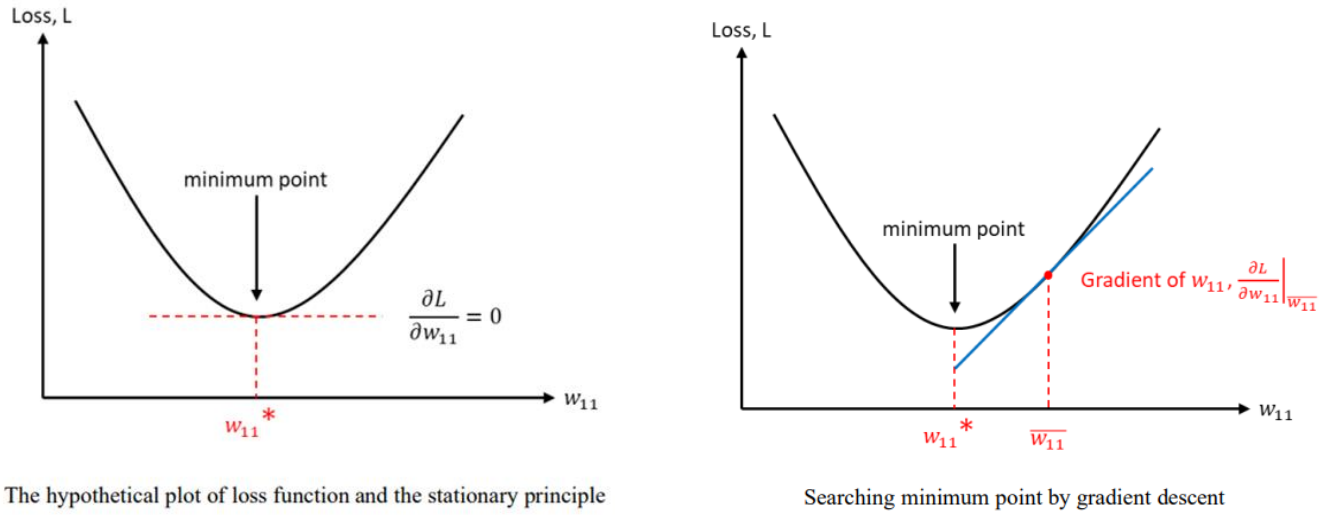


Foto 3: Variational approach and minimum principle

Jika kita dapat minimum point ini, maka network kita akan completely trained dan ia akan predict output yang betul seperti yang kita kehendaki. Contoh, dia boleh predict gambar2 binatang tu dengan tepat.

Cumanya, kita tak boleh directly dapatkan minimum point by setting the gradient to zero (e.g., $\frac{\partial Loss}{dw} = 0$ atau $\frac{\partial L}{dw} = 0$) mcm kita buat dalam FEM (atau waktu sekolah menengah dahulu).

The main reasons nya:

i. Loss Function kita melibatkan beratus ribu malah billions even trillion of parameters

Note: ChatGPT 4 sekarang ada 1.76 trillion of parameters, 120 layers!

ii. Loss Function kita juga berbentuk discrete (bukan continuous). So lebih senang kita gunakan approach yang diterangkan di bawah ni.

Jadi, instead of dapatkan directly minimum point tu, kita dapatkanya secara "sikit"2, yakni secara "updating". Kata lain, kita update "sikit"2 value of the parameters (weights and biases) sehingga kita jumpa minimum point tu. Untuk process updating ni, salah satu caranya adalah dengan guna formula dibawah:

$$w_{new} = w_{old} - \mu \frac{\partial Loss}{\partial w_{old}}\Big|_{w_{old}} \qquad (3)$$

dimana w_{new} tu adalah nilai pameters yang baru (updated) dan w_{old} tu value parameters yang lama. Gradient $\frac{\partial Loss}{\partial w_{old}}\Big|_{w_{old}}$ tu kita dapatkan guna chain-rule. μ tu pula dipanggil learning rate, kononnya untuk cepatkan training tapi kena "trial-and-error" valuenya.

Updating guna Eq. (3) ni kita panggil Stochastic Gradient Descent (SGD). Selain ini, kita ada options lain seperti Adam, AdaGrad, RMSProp. All these are important optimizers or minimizers hence important terminologies dalam DL.

Value parameters yang baru ni (yang been updated by Eq. (3)), kita akan masukkan semula kedalam network, kita replacekan pada value yang lama.

Then, kita ulang semula process yang sama. Kita masukkan input yang baru kedalam network dan kira semula Error dan Loss-functionnya tapi based on input dan value parameters yang baru. Kemudian, guna Eq. (3) untuk update lagi value parameters tersebut.

So kita akan ulang, ulang, ulang process iteration ini (ini la kita yang kita panggil training the network) sampai Error itu hilang atau jadi sangat2 kecil. Bila ini tercapai, maknanya:

i. Network kita sudah completely trained

ii. Value parameters (weights and biases) sudah betul

iii. Kita sudah boleh guna network kita untuk buat prediction.

- masuk gambar kucing, network akan kata itu kucing

- masuk gambar plate kereta, network terus baca berapa nombor plate nya

- bila kita masuk aliran di hilir sungai (inflow hydrograph), network boleh predict outflow hydrographnya dan banjir atau tidak (kalau nak contoh kegunaan AI dalam civil engineering).

Walla! This is the basic idea on how and why AI works!

Oh, before that, ada dua last terminologies yang penting (yang akan dijumpa bila kita baca atau belajar pasal DL), iaitu Backpropagation dan Forward-pass.

Process dari Input sampai terhasilnya Output kita panggil Feedforward (atau forward propagation or forward pass).

Process dari kira Error, kemudian kira Loss-function, kemudian kira gradients, kemudian updating, kita panggil Backpropagation (or backward pass).

So, process "training" ini akan melibatkan iteration of forward-pass dan backward pass ini sampai converged (sampai Error become zero atau very2 small). Sound very familiar kan? It sounds exactly macam nonlinear FEM. Memang pun.

Ok, tak lengkap kalau saya tak tunjukkan satu "worked example" untuk nail kan our understanding. Tapi, pada sesiapa yang rasa dah cukup discussion di atas, dia boleh skip "worked example" ini dan go straight to the concluding remarks di mana saya highlightkan beberapa isu semasa berkaitan AI Data Centre, GPU war dan chip wars. Saya juga ada highlight kan my views on what would be the possible futures for our engineering fields due to the influence of AI.

Ok sapa yang nak go through dulu dengan the "worked example" here we go. Sapa nak skip, boleh skip.

"WORKED EXAMPLE"

Untuk "worked example" ni, kita create the simplest network seperti dalam Foto 4. Ia cuma ada satu input (x) dan satu ouput ($\hat{y}$), yang di hubungkan oleh dua layers of a single neuron.

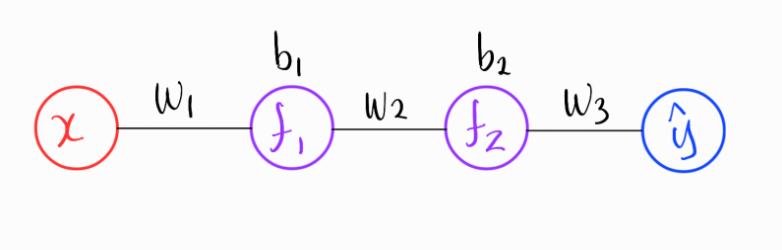


Foto 4: The simplest network

Untuk simplest network ini, kita cuma ada 5 parameters iaitu weights, w_1, w_2, w_3 dan biases, b_1 dan b_2.

Bila kita masukkan input, (x) kedalam hidden layer yang pertama, ia akan didarabkan dengan w_1 kemudian ditambah dengan b_1 sebelum dimasukkan kedalam activation function neuron pertama, f_1. Ini akan di "spit out" by neuron pertama sebagai:

$f_1(xw_1 + b_1)$ yang bermaksud, activation dari neuron pertama.

Kemudian, ia akan masuk ke hidden layer kedua dimana ia akan didarabkan dengan w_2 dan ditambah dengan bias b_2 sebelum dimasukkan kedalam activation function neuron kedua, f_2. Ini akan di "spit out" by neuron kedua sebagai:

$f_2(w_2(f_1(xw_1 + b_1)) + b_2)$ yang bermaksud, activation dari neuron kedua.

Finally, ia akan sampai ke Output layer. Tapi sebelum sampai, ia akan didarabkan dahulu dengan w3. So, Output network ini boleh diberikan sebagai

$$\hat{y} = w_3 f_2(w_2(f_1(xw_1 + b_1)) + b_2) \qquad (4)$$

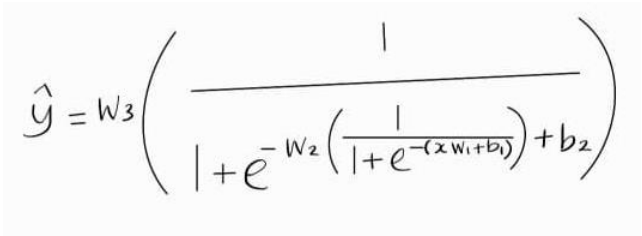
Tadaa...kita telah relate kan Input (x) kepada Output ($\hat{y}$) secara "mathematics"!

Ini juga yang dimaksudkan,

"Hidden layers adalah functions yang menyambungkan input kepada output"

Dan, apabila kita dapat value ŷ dari Eq. (4), itu bermakna kita telah completekan satu forward-pass.

Note: Refer Foto 5 dan tengok Eq. (4a). Di situ saya tunjukkan apa akan jadi pada Eq. (4) kalau saya gunakan the actual expression of activation function, say sigmoid, untuk f_1 dan f_2.



(4a)

Foto 5: Eq. (4) when the sigmoid is used for the activation function

Dan kita boleh lihat, dalam hubungan mathematics itu (Eq. (4) or (4a)), terkandung coefficients atau parameters weights (e.g., w_1, w_2, w_3) dan biases (e.g., b_1, b_2) yang valuenya kita perlu "train" terlebih dahulu supaya menjadi value yang betul.

Bila values weights dan biases itu betul, barulah equation antara Input (x) dan Output ($\hat{y}$) tu jadi betul juga. Dan barulah kita boleh predict apakah value $\hat{y}$ apabila kita ada input x yang baru.

So, selagi Eq. (4) itu belum betul lagi, ia akan produce Error. Error ini kita perlu buang atau kurangkan dengan train parameters kita. Error tersebut boleh diberikan sebagai

$$Error, E = \hat{y} - \mathrm{y} \qquad (5)$$

where, $\hat{y}$ adalah output yang dikira guna Eq. (4) atau 4(a) dan y adalah value yang betul (labelled output). So perbezaan antara output yang dikira oleh network (yang masih belum complete trainingnya) dan output yang betul, itu lah Error.

Selepas kita dapat Error, kita dapatkan Loss function nya pula guna MSE. Ia diberikan seperti berikut:

$$Loss\ function, L = \frac{1}{n}\sum Error^2 = \frac{1}{n}\sum(\hat{y} - \mathrm{y}\ )^2 \qquad (6)$$

Now kita dah ready untuk update setiap parameters dan since ini "worked example" dan since cuma ada 5 parameters sahaja, saya akan berikan updating untuk setiap parameters supaya senang nampak, seperti berikut:

$$w_{1.new} = w_{1,old} - \mu \frac{\partial Loss}{\partial w_{1,old}}\bigg|_{w_{1,old}} \qquad (7)$$

$$w_{2.new} = w_{2,old} - \mu \frac{\partial Loss}{\partial w_{2,old}}\bigg|_{w_{2,old}} \qquad (8)$$

$$w_{3.new} = w_{3,old} - \mu \frac{\partial Loss}{\partial w_{3,old}}\bigg|_{w_{3,old}} \quad (9)$$

$$b_{1.new} = b_{1,old} - \mu \frac{\partial Loss}{\partial b_{1,old}}\bigg|_{b_{1,old}} \quad (10)$$

$$b_{2.new} = b_{2,old} - \mu \frac{\partial Loss}{\partial b_{2,old}}\bigg|_{b_{2,old}} \quad (11)$$

That's it, dengan updatednya value2 parameters ini, settled backward pass. Sekarang tinggal masukkan value parameters yang baru updated ni kedalam network, then masukkan set input yang baru dan ulang Eq. (4) hingga Eq. (11).

Then, the next round kita update parameters lagi, kita masukkan input baru lagi ke dalam network dan ulang.

Ulang dan ulang (iterate) sampailah Error pada Eq. (5) jadi sekecil2 nya (sekecil yang kita boleh terima). Kata lain, iterate sampai converged.

Bila dah converged, boleh la guna network tu untuk predict gambar ikan ke, gambar plate kereta, predict bahasa ke, predict orang bercakap ke, predict aliran sungai ke, predict apa2 lah.

Settleddd....

Again, inilah the basic idea of AI Deep Learning. Syabas kepada sesiapa yang masih mampu baca sampai ke sini...haha, anda insan yang hebat :).

By the way, seperti yang saya informed dalam posting sebelum ini, saya dan team (Dr Razin and Dr Halinawati of UTM, Dr Haizad dan Dr Al-Akhbar of UNISEL) telah tulis satu module yang proper untuk membincangkan topic ini.

Di dalam nya kami tunjukkan all the proper maths and formulations termasuk coding yang kami tulis from scratch guna Matlab (takda guna apa2 Toolbox) supaya pembaca boleh belajar everything from scratch juga (even dapatkan gradients with respect to parameters pun kami tunjukkan derivation chain rulenya dan tulis scriptnya from scratch).

So sesiapa yang berminat nak baca module kami tu dan tengok codenya, boleh download dengan click link bawah ni:

https://github.com/msnm-official/ai_modules

Ok lah, kita dah boleh conclude.

CONCLUDING REMARKS

1. The basics of AI Deep Learning (the summary):

- To relate Input to Output mathematically through the hidden layers
- The "mathematical coefficients" of the hidden layers are the parameters, weights and biases

- These parameters must be "trained" until they attain their correct values. Once their values are correct, the network can start making the correct prediction and produce the desired output (i.e.., robot that can speak and communicate with human)
- The training process involves the minimization of the Loss-Function which is the functional of Error
- The training is an iteration of forward and backward passes until converge (the Error becomes very small)

2. Selepas memahami (sedikit sebanyak) how and why AI works, sekarang kita boleh lebih memahami kesan dan kemungkinan of AI.

- Pertamanya, AI memerlukan data yang sangat banyak untuk train parameters tu.
- Ketika ini, data yang banyak tersedia adalah berkaitan image (pixels), linguistic (text, language), finance (fintech) dan medical. Kerana ini kita lihat technology of AI seperti ChatGPT, Google Translate, banyak mempengaruhi bidang2 language, entertainment, report writing dan literature, journalism, communications, graphic design, management, banking and finance serta medical kerana data2 mereka banyak tersedia dalam simpanan persendirian syarikat dan institution atau di internet. Dan bidang2 ini lah yang sangat terkesan dengan AI ketika ini.
- Tetapi ini tidak bermaksud bidang lain tidak akan terkesan. Dalam bidang kejuruteraan contohnya, mana2 aktiviti yang menyimpan banyak data boleh dan in fact, sudah terkesan pun dengan AI, seperti robotics dalam perkilangan dan pembuatan dan signal processing.
- Atau, jika mahukan engineering activity tersebut terkesan dengan AI, jurutera perlu memikirkan bagaimana hendak mendapatkan data2 yang mencukupi. Sebagai contoh, dalam bidang hydrology, data hujan dan data aliran banyak terkumpul. Maka tugas hydrologists adalah memikirkan bagaimana data2 ini boleh diexploit secara AI.
- Kerana kekurangan data dalam engineering juga lah, usaha untuk menghasilkan Deep Learning yang baharu sedang rancak di jalankan di seluruh dunia. DL yang baharu ini dipanggil dengan pelbagai nama, antaranya Physics-Informed Neural Network (PINN), Physics-Based Deep Learning (PBDL) dan Physics-Guided Machine Learning (PGML).
- Dalam AI yang baharu ini, physics governing equations atau mechanics dimasukkan kedalam network untuk compensate data yang kurang. Jika AI yang baharu ini berjaya disempurnakan, definitely, bidang kejuruteraan juga akan “overwhelmed” by AI.
- Kalau masih ingat lagi, dalam posting saya sebelum ini, saya ada share code2 PINN yang saya tulis dengan Dr Razin. Boleh cari semula posting tersebut kalau nak tengok codingnya. Coding tu bagus untuk belajar the basics sebab kami tulis from scratch, takda pakai Toolbox, so boleh nampak semua sekali.

3. Dengan memahami the basics of AI, kita juga dapat appreciate how AI ini sangat consuming dari segi computer resources dan energy. Untuk menyimpan data2, ia memerlukan storage yang sangat besar. Untuk training the network, ia memerlukan infrastructure yang besar juga. Bayangkan resources yang diperlukan kalau network dalam Foto 2 dikembangkan sehingga mempunyai trillion neurons!

- Kerana ini lah, banyak AI Data Centre di dirikan saat ini di seluruh dunia termasuk di Malaysia. Cuma satu persoalan berkenaan AI Data Centre di Malaysia.
- Adakah ia akan jadi the classic of "negara kita cuma menyediakan tanah2 dan kilang2 untuk assembly product luar" seperti industri perkilangan 90an atau kita mampu involve secara langsung dalam development of AI technologies dan industries dan menjadi world player?
- Pada saya, jika kita mahukan the latter, kita mesti beri penekanan kepada skills of maths, physics, dan coding dalam graduan, engineers dan industri kita. Especially nanti apabila kita nak involve dengan engineering industries yang "overwhelmed" by AI. Our future engineers must be ready for this. Cuba tengok discussion yang saya berikan dalam posting ni, Deep Learning is highly mathematics kan?
- Yes, Jensen Huang (CEO NVIDIA) ada kata coding tak diperlukan sebab AI boleh buatkan. Tapi kita kena faham statement ini in its context. Pertama, kita cuma boleh prompt AI untuk buat code, kalau kita sendiri tahu code apa yang kita mahukan dan ini memerlukan kita ada ilmu dan pengalaman terlebih dahulu terhadap coding.
- Keduanya, Jensen Huang mungkin referred kepada DBMS type of coding, belum engineering codes lagi. Saya pernah prompted ChatGPT untuk buatkan code FEM. Saya kena prompt banyak kali (bagitau ChatGPT untuk buat berperingkat2), baru dia dapat buat seperti yang saya mahukan. So nak tak nak, maths dan engineering fundamentals mesti strong. Memang dari dulu prinsipnya, to code is easy, what to code is the challenge.

4. For the training process, AI memerlukan hardware yang paling advanced for it to be of industrial and commercial use. Process training ini boleh dilakukan atas CPU atau GPU.

- Cumanya, at current rate, GPU menjadi pilihan untuk process training kerana ia sangat efficient untuk parallel computing. Jadi, selalunya CPU diemployed untuk data processing (input dan output labelling) dan GPU untuk training.
- Kerana inilah kita lihat persaingan yang sangat hebat antara developer2 GPU terutamanya NVIDIA vs AMD, sehingga ia digelar the GPU war. Sesiapa yang keluar dari persaingan atau "peperangan" ini victorious, dia akan menjadi kuasa yang pegang market AI dunia kerana process training akan menggunakan GPUnya.
- Dan, untuk membina GPU tercanggih ini, ia memerlukan chip2 yang termaju juga. Disinilah kita lihat persaingan antara developer2 chip seperti TSMC dan Samsung. AMD kebanyakannya menggunakan chip dari TSMC sementara NVIDIA mengunakan chip dari TSMC dan Samsung.
- TSMC menjangkakan akan berjaya menghasilkan chips bersaiz 1.6 nanometer by 2026
- Sementara itu kita lihat, tekanan antarabangsa dalam bekalan chip ke China telah menggalakan China untuk develop sendiri chip2 AI melalui SMIC.
- Dan inilah chip war nya, "peperangan" untuk menghasilkan AI chip yang akan digunakan oleh industri CPU dan GPU seluruh dunia.
- GPU and Chips Wars inilah, the big-boys game dalam dunia AI sekarang ini. Dan "wars" ini actuallynya boil down to who will conquer the training of AI parameters dalam dunia ni. So that's how big this "training the network" thingy is.
- Cuma persoalannya sekarang, Malaysia nak jadi one of the big-boys in AI or engineering "overwhelmed" by AI atau tidak?

- Kalau kita mahu, kita mesti beri penekanan kepada maths, physics, engineering fundamentals dan first-principles dan coding dalam universiti dan industri kita, dalam pelajar, graduan dan jurutera2 kita. There is no two ways about it.

EPILOGUE

Mungkin ada yang kata, "Eh...kata nak explain se"simple" mungkin, tapi ni panjang berjela".

Hahaha...simple la dah ni, percaya lah.

Lagipun Einstein pun kata, "Everything should be made as simple as possible, but not simpler".

Kalau "simpler" dia akan jadi pseudo, lagi bahaya.

Tapi, kalau masih tak percaya (yang penulisan saya ni cukup simple tapi tak simpler), boleh test.

Lepas ni, just google "Deep Learning" atau "AI" atau "Machine Learning", dan pick mana2 article dan baca. InsyaAllah, you guys (especially the first time learners of the topic) akan dapat baca article2 tersebut dengan "selesa" dan akan rasa macam article2 tu familiar.

Tak pun, watch mana2 youtube video on Deep Learning, AI atau Machine Learning, insyaAllah akan rasa "selesa" dan "familiar" juga.

Perasaan "selesa" dan "familiar" kepada sesuatu ilmu itu lah a good sign untuk menjadi learner kepada ilmu tersebut. Selalunya, bila dah rasa "selesa", kita akan nak belajar lagi benda tu.

Maka, menghidupkan perasaan "selesa" dan "familiar" itulah yang saya cuba wujudkan dengan penulisan saya ini, supaya ramai rakyat dan engineer2 Malaysia ingin tahu dan belajar lebih lanjut berkenaan AI ini (dan ilmu2 kejuruteraan lain yang turut berkembang).

Moga Allah berikan kita semangat untuk terus belajar dan menjadi life-long learner seterusnya menyumbang kepada agama, masyarakat dan negara. InsyaAllah.

Wallahualam (Allah lebih mengetahui)

References:

[1] Zhuhadar, Lily Popova & Lytras, Miltiadis. (2023). The Application of AutoML Techniques in Diabetes Diagnosis: Current Approaches, Performance, and Future Directions, Sustainability. 15. 13484. 10.3390/su151813484

[2] https://blog.nami.gg/p/how-generative-ai-can-charge-web3