

ลอกให้ถูกชั้น

มนุษย์เลียนแบบธรรมชาติจนได้ปัญญาประดิษฐ์ได้อย่างไร — ตั้งแต่หามติด
ชนหมา ถึงแม่เหล็กที่คิดแทนเรา

ห้องสมุด AICE · copy-the-right-layer

14 บท · ฉบับภาษาเดียว

เขียนครั้งแรก 20 กันยายน 2026 · ปรับปรุงล่าสุด 20 กันยายน 2026

ทุกตัวเลข ชื่อ และปีในเล่มนี้ตรวจย้อนไปยังแหล่งที่อ้างอิงไว้ได้

ลอกให้ถูกชั้น

มนุษย์เลียนแบบธรรมชาติจนได้ปัญญาประดิษฐ์ได้อย่างไร — ตั้งแต่หนามติดขนหมาถึงแม่เหล็กที่คิดแทนเรา

ปี 2024 คณะกรรมการโนเบลมอบรางวัลสาขา**ฟิสิกส์**ให้กับคนสองคนที่ไม่ได้ทำฟิสิกส์มาสี่สิบปี

คนหนึ่งเป็นนักชีวฟิสิกส์ที่หันไปสนใจความจำของมนุษย์ อีกคนเป็นนักจิตวิทยาการรู้คิดที่กลายเป็นนักวิทยาศาสตร์คอมพิวเตอร์ ทั้งคู่ใช้ชีวิตส่วนใหญ่อยู่ในแผนกที่ไม่มีคำว่าฟิสิกส์ติดอยู่ที่ประตู และสิ่งที่พวกเขาทำจนได้รางวัลคือของที่ทุกวันนี้เรียกว่าปัญญาประดิษฐ์

เสียงบ่นตามมาทันที ว่าเป็นการยกย่องรางวัลฟิสิกส์ให้วิศวกรรมคอมพิวเตอร์

เล่มนี้จะพาไปดูว่าทำไมเสียงบ่นนั้นเข้าใจผิด และคำอธิบายที่ถูกต้องนั้นตรงไปตรงมากกว่าที่คิดมาก — เพราะกลไกที่อยู่ข้างในโปรแกรมแชทที่ตอบคำถามเป็นภาษาคนได้ เป็นสมการเดียวกันกับที่นักฟิสิกส์ใช้อธิบายว่าทำไมตะปูถึงติดแม่เหล็ก ไม่ใช่คล้ายกัน ไม่ใช่เปรียบเทียบได้ แต่เป็นสมการเดียวกัน

เล่มนี้ไม่ใช่อะไร

ไม่ใช่ประวัติศาสตร์ AI แบบไล่ปี ถ้าอยากได้ไหมไลน์ครบ มีคนเขียนไว้ดีกว่านี้แล้วหลายเล่ม เล่มนี้เดินตามคำถามข้อเดียว คือ **เวลามนุษย์ลอกธรรมชาติ เราลอกอะไรของมันกันแน่** แล้วใช้คำถามนั้นไล่ตั้งแต่หนามติดขนหมาไปจนถึงชิปที่ยังสร้างไม่เสร็จ

ไม่ใช่เล่มที่ต้องมีพื้นฐาน ไม่มีสมการสักรตัวในเล่มนี้ ตั้งแต่หน้าแรกถึงหน้าสุดท้าย แต่การที่ไม่มีสมการไม่ได้แปลว่าไม่มีของจริง ตรงไหนที่ของจริงมีเงื่อนไข เงื่อนไขนั้นจะถูกพิมพ์ไว้

และไม่ใช่เล่มที่อ่านแล้ววาง ทุกบทมีของหนึ่งอย่างให้ลองเอง ทำได้วันนี้ ด้วยของที่มีอยู่แล้วในบ้าน ไม่ต้องซื้อ ไม่ต้องลงโปรแกรม ตีนตุ๊กแกหนึ่งแผ่น แม่เหล็กติดตู้เย็น สองอัน สวิตช์ไฟหับบันได กล่องกับเหรียญ

ภาพเดียวที่ต้องแบกไปทั้งเล่ม

ลูกบอลกลิ้งลงหุบเขา

แค่นั้น ถ้าคุณเห็นภาพลูกบอลที่ถูกปล่อยลงบนพื้นขรุขระแล้วมันหาทางลงไปอนที่ก้นหุบ คุณมีของครบแล้วสำหรับเล่มนี้ทั้งเล่ม เราจะใช้ภาพนี้อธิบายก่อนแม่เหล็กในบทที่ 5 ใช้อธิบายความจำในบทที่ 6 ใช้อธิบายการเรียนรู้ในบทที่ 7 และใช้อธิบายว่า ChatGPT เลือกคำตอบไปยังไงในบทที่ 10

ของชิ้นเดียว หกบท ไม่เปลี่ยนอุปกรณ์กลางทาง

สิบสองบทข้างหน้า

สิบบทแรกเป็นเรื่องของการลอกที่ยังไม่สำเร็จ — หนาม ปีกนก แล้วนิรอนตัวแรกที่วาดขึ้นมาบนกระดาษในปี 1943 ซึ่งทำอะไรไม่ได้เลยอยู่หลายปี

สิบหกกลางเป็นเรื่องของการลอกที่สำเร็จ และมันสำเร็จตอนที่มีคนเลิกลอกสมองแล้วหันไปลอกก้อนเหล็ก

สิบบทท้ายเป็นเรื่องของวันนี้กับวันข้างหน้า รวมถึงคำถามที่ค้างอยู่ในหัวคนเขียน โปรแกรมรุ่นหนึ่งมานาน ว่าการเขียนให้ประหยัด ที่เราทิ้งไปตอนเครื่องแรงขึ้น มันหายไปไหน และมันกำลังกลับมาหรือเปล่า

คำตอบอยู่ในบทที่ 11 และมันไม่ได้เข้าข้างความรู้สึกที่ทำให้เกิดคำถามนั้นทั้งหมด

ทุกวันที่ ชื่อคน และตัวเลขในเล่มนี้อ้างอิงแหล่งที่ตรวจแล้ว รายการทั้งหมดอยู่ท้ายเล่ม ตรงไหนที่แหล่งขัดกันเอง เล่มนี้พิมพ์ทั้งสอง

1 • หนามที่ติดขนหมา

หนามนั้นไม่ได้เกาะ มันเกี่ยว

ความต่างระหว่างสองคำนี้ดูเล็กมากจนน่าหงุดหงิด แต่มันคือทั้งหมดของเล่มนี้ ของที่เกาะต้องใช้แรงดูด ใช้กาบ ใช้ไฟฟ้าสถิต — ของพวกนี้ละเอียด หמדอายุ แล้วก็ใช้ซ้ำไม่ได้ ส่วนของที่เกี่ยวข้อง ใช้แค่รูปทรง ตะขอหนึ่งอันกับห่วงหนึ่งอัน ไม่มีเคมีเข้ามาเกี่ยวข้องเลย และมันดึงออกแล้วเกี่ยวใหม่ได้เป็นพันครั้ง

George de Mestral วิศวกรชาวสวิส กลับจากเดินป่าล่าสัตว์ในเทือกเขาแอลป์ช่วงต้นทศวรรษ 1940 พร้อมกับหมาของเขาและปัญหาที่ทุกคนที่พาหมาเดินป่าเคยเจอ คือ ขนหมาเต็มไปด้วยหนามพืชที่ดึงออกยาก เรื่องเล่ามาตรฐานหยุดตรงนี้แล้วกระโดดไปที่คำว่า "เขาจึงประดิษฐ์ตีนตุ๊กแก" ซึ่งข้ามส่วนที่น่าสนใจที่สุดไปทั้งหมด

ส่วนที่น่าสนใจคือเขาเอาหนามไปส่องกล้องจุลทรรศน์

นั่นคือการตัดสินใจที่แยกเขาออกจากคนอื่นทุกคนที่เคยดึงหนามออกจากขนหมา คนอื่นเห็น "ของที่ติด" เขาไปดูว่ามันติดด้วยกลไกอะไร และสิ่งที่เห็นในกล้องคือปลายหนามแต่ละอันงอกลับเป็นตะขอเล็ก ๆ ขนสัตว์เป็นวงห่วงพันกัน ตะขอเข้าห่วง ล็อกจบ

ทีนี้ลองสังเกตว่าเขาไม่ได้ลอกอะไรบ้าง เขาไม่ได้ทำของที่เป็นสี่เหลี่ยม ไม่ได้ทำของที่กลม ไม่ได้ทำของที่เติบโตได้ ไม่ได้สนใจว่าพืชชนิดนั้นชื่ออะไร อยู่ในวงศ์ไหน หนามมันมีไว้เพื่อกระจายพันธุ์ยังไง ทั้งหมดนั้นคือหนามจริง ๆ และทั้งหมดนั้นเขาทิ้ง

เขาเอามาชิ้นเดียว คือ ตะขอเจ็ดห่วงแล้วล็อก

แล้วเขาก็ใช้เวลาต่อจากนั้นอีกแปดปี

แปดปีนั้นไม่ได้หมดไปกับการคิดไอเดีย ไอเดียจบตั้งแต่วันแรกที่ส่องกล้อง แปดปีหมดไปกับคำถามว่าจะทำตะขอกับห่วงจากอะไร ฝ้ายอ่อนเกินไป ตะขอคล้ายตัวหลังใช้ไม่

ก็ครั้ง คำตอบมาจากไนลอนซึ่งเพิ่งมีใช้ในตอนนั้น และมาจากการค้นพบว่าถ้าเอา ไนลอนไปอบใต้แสงอินฟราเรดแล้วตัดปลายห้วง มันจะได้ตะขอที่คืนรูปเองได้ เขาจด สิทธิบัตรในสวิตเซอร์แลนด์ปี 1951 ได้รับอนุมัติปี 1955 และตั้งชื่อของชิ้นนั้นจากคำ ฝรั่งเศสสองคำ — velours กำมะหยี่ กับ crochet ตะขอ

Velcro

เรื่องนี้ถูกเล่าซ้ำจนกลายเป็นตัวอย่างมาตรฐานของคำว่า biomimicry แต่สิ่งที่มันสอน จริง ๆ ไม่ใช่ว่า "ธรรมชาติมีคำตอบอยู่แล้ว" ธรรมชาติไม่เคยมีคำตอบสำเร็จรูป สิ่งที่มีมันคือกองของที่ใช้งานได้ ปนกับกองของที่ไม่เกี่ยวข้องเลย วางทับกันอยู่เป็นชั้น ๆ งานของคนลอกคือรู้ว่าจะเอาชั้นไหน

และงานนั้นยากกว่าที่ใคร ๆ ยอมรับกัน เพราะชั้นที่ผิดมักจะเป็นชั้นที่เห็นง่ายที่สุด

หนามที่เห็นด้วยตาเปล่าคือของสีเขียว ๆ กลม ๆ มีขน ส่วนตะขอที่เป็นคำตอบจริงต้องใช้กล้องถึงจะเห็น ของที่เด่นที่สุดกับของที่สำคัญที่สุด แทบไม่เคยเป็นของชิ้นเดียวกัน

บทต่อไปเป็นเรื่องของคนกลุ่มหนึ่งที่ลอกชั้นที่เห็นง่ายที่สุด แล้วตกลงมาตาย

ลองเอง: หาตีนตุ๊กแกในบ้าน — รองเท้าเด็ก สายรัดของ ปกกระเป๋า แยกสองด้านออกจากกันแล้วเอานิ้วลูบ ด้านหนึ่งจะสากเหมือนกระดาษทราย อีกด้านจะนุ่มเหมือนผ้า สักหลาด เอาด้านสากไปแตะผ้าขนหนู แล้วลองดึง — มันจะติด ทั้งที่ผ้าขนหนูไม่ใช่ ของคู่กัน นั่นเพราะตะขอไม่เลือกห้วง มันแค่ต้องการห้วง ซึ่งเป็นเหตุผลเดียวกับที่ หนามเกาะขนหมาได้ทั้งที่พิชไม่เคยรู้จักหมา

บทนี้จะผิดเมื่อ: มีหลักฐานร่วมสมัยว่า de Mestral ทำงานจากความเข้าใจระดับเคมี ของผิวหนาม ไม่ใช่ระดับรูปทรง หรือมีเอกสารว่าไนลอนถูกเลือกด้วยเหตุผลอื่นที่ไม่ใช่การคืนรูปของตะขอ

2 • คนที่กระพือปีก

คนที่ลอกการกระพือปีกตกตาย คนที่ลอกแรงยกได้บิน

ประโยชน์นี้สันจนดูเหมือนคำคม แต่มันคือบันทึกตรง ๆ ของสิ่งที่เกิดขึ้นในช่วงสามสิบปีสุดท้ายก่อนเครื่องบินลำแรก และถ้าจะมีบทเรียนข้อเดียวที่ต้องแบกไปตลอดเล่มนี้ มันอยู่ตรงนี้ ไม่ใช่ตรงไหนอื่น

ย้อนกลับไปดูโลกก่อนหน้านั้นก่อน คนที่อยากบินมีตัวอย่างอยู่ตรงหน้าเต็มไปหมด นักทำได้ทุกวัน ทุกเช้า ฟรี ๆ และสิ่งที่ทุกคนเห็นเมื่อมองนกคือ**มันขยับปีกขึ้นลง** นั่นคือส่วนที่เคลื่อนไหว ส่วนที่ดังที่สุด ส่วนที่ตาจับได้ ใครก็ตามที่วาดรูปนกให้เด็กดู จะวาดปีกกาง ๆ แล้วบอกว่ามันกระพือ

คนจำนวนมากจึงสร้างเครื่องที่กระพือปีก มีคำเรียกเฉพาะด้วยซ้ำ — ornithopter ตั้งแต่สมัย Leonardo da Vinci ลากมาจนถึงศตวรรษที่ 19 มีคนกระโดดลงจากหอคอยพร้อมปีกที่ขยับได้หลายสิบคน และผลที่ได้ก็เหมือนกันเกือบทุกครั้ง

ที่น่าสนใจกว่าคือคนที่ทำถูก **ครั้งเดียว**

Otto Lilienthal วิศวกรชาวเยอรมัน ทำสิ่งที่ไม่เคยมีใครทำมาก่อนคือหยุดเดาแล้วไปวัด เขาศึกษาปีกนกอย่างจริงจัง ตีพิมพ์หนังสือชื่อ *Der Vogelflug als Grundlage der Fliegekunst* ในปี 1889 — การบินของนกในฐานะรากฐานของศิลปะการบิน — และสิ่งที่เขาดึงออกมาจากนกไม่ใช่การกระพือ เขาดึง**รูปโค้งของปีก**ออกมา ปีกนกไม่ได้แบน มันโค้งขึ้นด้านบน และความโค้งนั้นทำให้อากาศที่ไหลผ่านด้านบนกับด้านล่างไม่เท่ากัน ซึ่งดันปีกขึ้น

เขาสร้างเครื่องร่อนตามรูปร่างนั้นและบินจริง ประมาณ 2,000 เที่ยวบินระหว่างปี 1891 ถึง 1896 ภาพถ่ายของเขาห้อยตัวอยู่ใต้ปีกผ้ากางกว้างถูกตีพิมพ์ไปทั่วโลกและเป็นสิ่งที่จุดให้คนรุ่นถัดไปเริ่มลงมือ

วันที่ 9 สิงหาคม 1896 ลมกระโชกพลิกเครื่องของเขาขณะบิน เขาตกจากความสูงราว 15 เมตร กระตุกสันหลังหัก และเสียชีวิตในวันรุ่งขึ้น

Lilienthal ลอกถูกชั้นเรื่องแรงยก และเขาไม่ได้ลอกชั้นที่สองที่จำเป็นพอ ๆ กัน คือการ**ควบคุม** วิธีบังคับเครื่องของเขาคือเหวี่ยงน้ำหนักตัว ซึ่งใช้ได้ตอนลมนิ่งและใช้ไม่ได้ตอนลมกระโชก นกไม่ได้บังคับตัวเองด้วยการเหวี่ยงน้ำหนัก มันบิดขมปลายปีก

แต่ตรงนั้นเป็นส่วนที่มองด้วยตาเปล่าไม่เห็น เพราะมันเล็กและเร็ว เหมือนตะขอบนหนามที่ต้องส่องกล้องถึงจะเจอ

สองพี่น้องที่ร้านจักรยานในเมือง Dayton อ่านข่าวการตายนั้นและทำสองอย่างที่ต่างจากทุกคน

อย่างแรก พวกเขาไปนั่งดูนกแรงร้อน — ไม่ใช่ดูว่ามันกระพืออย่างไร แต่ดูว่าตอนมันเอียงตัวแล้ว **อะไรบนตัวมันขยับ** สิ่งที่เขาเห็นคือปลายปีกข้างหนึ่งบิดขึ้น อีกข้างบิดลง พวกเขาออกซันนั้นมาเป็นระบบที่เรียกว่า wing warping คือดึงลวดให้ปีกผ้าบิดตัว ซึ่งเป็นบรรพบุรุษของปีกเล็กที่อยู่ปลายปีกเครื่องบินทุกลำในทุกวันนี้

อย่างที่สองสำคัญกว่า และไม่เกี่ยวกับนกเลย พวกเขาเอาตารางค่าแรงยกของ Lilienthal มาลองใช้แล้วพบว่าเครื่องร่อนของตัวเองยกไม่ขึ้นตามที่ตารางบอก แทนที่จะโทษเครื่องตัวเอง พวกเขาสงสัยตาราง จึงสร้างอุโมงค์ลมขนาดเล็กขึ้นในปี 1901 แล้ววัดปีกรูปต่าง ๆ ใหม่ทั้งหมดด้วยตัวเอง ตัวเลขที่ได้ต่างจากของ Lilienthal จริง และเครื่องที่สร้างจากตัวเลขใหม่คือเครื่องที่บินขึ้นจากเนินทรายที่ Kitty Hawk วันที่ 17 ธันวาคม 1903 นาน 12 วินาที ไกล 120 ฟุต

สังเกตให้ดูว่าอุโมงค์ลมไม่ได้ลอกอะไรจากธรรมชาติเลยสักนิดเดียว มันคือเครื่องมือสำหรับ**ตรวจสอบ**สิ่งที่ลอกมา และมันคือสิ่งที่ทำให้การลอกครั้งนั้นต่างจากการกระโดดลงจากหอคอย

เครื่องบินทุกลำที่บินอยู่เหนือหัวคุณตอนนี้ ไม่มีลำไหนกระพือปีก ทั้งหมดใช้ปีกนิ่งกับความโค้งที่ Lilienthal วัดมาจากนก บวกกับแรงขับจากเครื่องยนต์ที่นกไม่มี นี่ไม่ใช่การลอกนกแบบแย ๆ มันคือการลอกที่ถูกชั้นพอดี — เอากฎ ทั้งรูปร่าง

และตรงนี้แหละที่เรื่องของปัญญาประดิษฐ์เริ่มต้น เพราะคำถามเดียวกันเป๊ะกำลังจะถูกลอกกับของอีกอย่างหนึ่งที่ทุกคนมีอยู่ในกะโหลก

ลองเอง: นั่งรถแล้วยื่นฝ่ามือออกนอกหน้าต่างตอนรถวิ่ง (ที่ที่ปลอดภัย ถนนโล่ง) ตั้งมือให้แบนขนานพื้นก่อน จะรู้สึกแค่ลมดันถอยหลัง จากนั้นเอียงปลายนิ้วขึ้นนิดเดียว — สิบถึงสิบห้าองศา ไม่ต้องมาก — มือจะถูกดันขึ้นทันทีอย่างแรง นั่นคือแรงยก มือคุณไม่ได้กระพือ ไม่ได้มีขน ไม่ได้มีอะไรเหมือนนกเลย มันแค่ทำมุมกับอากาศที่ไหล

ผ่าน และเท่านั้นก็พอแล้ว เอียงมากเกินไปแล้วลองอีกครั้ง มือจะสั่นแล้วแรงยกหายไป นั่นคือ stall ซึ่งเป็นสิ่งที่ฆ่านักบินมาตลอดศตวรรษ

บทนี้จะผิดเมื่อ: มีเครื่องบินโดยสารเชิงพาณิชย์ที่ใช้การกระพือปีกเป็นวิธีสร้างแรงยกหลัก หรือมีหลักฐานว่าพี่น้อง Wright สร้างอุโมงค์ลมด้วยเหตุผลอื่นที่ไม่ใช่ความไม่เชื่อตารางของ Lilienthal

3 · นิรอนที่วาดบนกระดาษ

สิ่งที่ McCulloch กับ Pitts วาดขึ้นในปี 1943 เป็นภาพวาดของกฎ ไม่ใช่ภาพวาดของเซลล์

และนั่นคือทั้งเหตุผลที่มันใช้ได้ และเหตุผลที่มันเรียนรู้ไม่ได้ยู่สิบห้าปี

ย้อนไปดูโลกตอนนั้นก่อน ปี 1943 ยังไม่มีคอมพิวเตอร์แบบที่เราเข้าใจกัน สิ่งที่มีคือคำถามที่ค้างมานานว่าความคิดเกิดขึ้นได้อย่างไรในก้อนเนื้อ นักประสาทวิทยารู้แล้วว่าสมองประกอบด้วยเซลล์ที่เรียกว่านิรอน รู้ว่ามันส่งสัญญาณไฟฟ้าหากัน แต่ระหว่าง "เซลล์ส่งไฟฟ้า" กับ "คนคิดเลขได้" มีช่องว่างที่ไม่มีใครข้าม

Warren McCulloch เป็นหมอกับนักประสาทวิทยา Walter Pitts เป็นเด็กหนุ่มที่ไม่เคยเรียนจบอะไรเลยและเก่งตรรกศาสตร์อย่างที่คนรอบตัวอธิบายไม่ถูก ทั้งสองเขียนบทความชื่อ A Logical Calculus of the Ideas Immanent in Nervous Activity ลงใน Bulletin of Mathematical Biophysics และสิ่งที่พวกเขาทำคือการลอกแบบที่บทที่แล้วพูดถึง

พวกเขาทิ้งเกือบทุกอย่างเกี่ยวกับเซลล์จริง

นิรอนจริงมีสารเคมีสืบทอดกัน มีเยื่อหุ้มที่ปล่อยไอออนเข้าออก มีรูปร่างแตกแขนงที่ไม่ซ้ำกันสักเซลล์ มีจังหวะการยิงที่เข้ารหัสข้อมูลด้วยความถี่ มีเซลล์ที่เลี้ยงคอยดูแลรอบ ๆ ทั้งหมดนั้นคือชั้นที่พวกเขาไม่เอา

สิ่งที่เอามาคือประโยคเดียว: รับสัญญาณเข้ามาหลายทาง รวมกันได้ถึงเกณฑ์ก็ส่งต่อ ไม่ถึงก็เงียบ

เท่านั้นจริง ๆ เปิดหรือปิด หนึ่งหรือศูนย์ ไม่มีกลาง

และสิ่งที่พวกเขาพิสูจน์ต่อจากนั้นคือส่วนที่ทำให้บทความนี้ยังถูกอ้างถึงอยู่จนทุกวันนี้ — ถ้าเอาของง่าย ๆ แบบนั้นมาต่อกันเป็นโครงข่าย มันทำงานตรรกะได้ทุกอย่างที่คอมพิวเตอร์ทำได้ "และ" "หรือ" "ไม่" ต่อกันเป็นชั้น ๆ ได้ทั้งหมด ความคิดไม่ต้องการสารเคมีพิเศษอะไรเลย มันต้องการแค่ของที่ตัดสินใจเปิดปิดได้ ต่อกันเยอะ ๆ

นั่นคือการลอกที่ถูกลืม เหมือนที่ Wright ลอกแรงยกไม่ใช่ชน

แต่มันมีปัญหาข้อหนึ่งที่ใหญ่มาก และใช้เวลาอีกสิบห้าปีกว่าจะมีคนแก้

โครงข่ายของ McCulloch กับ Pitts ไม่เรียนรู้ คนเป็นคนตั้งคำว่าสัญญาณไหนสำคัญแค่ไหน ตั้งเกณฑ์ว่าเท่าไรถึงจะยิง ทุกอย่างถูกตั้งด้วยมือ มันเป็นวงจรที่ฉลาดแต่ตายตัว เหมือนนาฬิกา สวยงามและไม่เคยเปลี่ยนใจ

ของที่ขาดไปไม่ใช่จำนวนนิวรอน มันคือ**ปุ่มปรับ**

Donald Hebb นักจิตวิทยาชาวแคนาดา เขียนสิ่งที่กลายเป็นคำตอบไว้ในหนังสือ The Organization of Behavior ปี 1949 ข้อเสนอมันคือเวลาเซลล์สองตัวทำงานพร้อมกันบ่อย ๆ การเชื่อมต่อระหว่างมันจะแข็งแรงขึ้น เส้นทางที่ใช้บ่อยจะเด่นขึ้น เส้นทางที่ไม่ใช้จะจางไป — เหมือนทางเดินในสนามหญ้าที่ชัดขึ้นเพราะคนเดินซ้ำ

ตรงนี้ต้องพูดให้ชัดหนึ่งข้อ ประโยคติดหูที่ว่า "เซลล์ที่ยิงพร้อมกัน เชื่อมต่อกัน" ไม่ใช่คำที่ Hebb เขียน มันเป็นคำสรุปที่มีคนตั้งขึ้นทีหลังราวปี 1992 และมันสรุปกระชับกว่าที่เขาเขียนจริงพอสมควร

Frank Rosenblatt เอาความคิดของ Hebb มาใส่ในเครื่องจริงในปี 1958 เขาเรียกมันว่า **perceptron** และแต่ละเส้นเชื่อมมีตัวเลขกำกับอยู่ตัวหนึ่ง เรียกว่า **weight** หรือน้ำหนัก ถ้าน้ำหนักสูง สัญญาณจากเส้นนั้นสำคัญมาก ถ้าต่ำ สำคัญน้อย ถ้าติดลบ สัญญาณนั้นคัดค้าน

วิธีสอนมันตรงไปตรงมาจนน่าประหลาดใจ — ให้มันทาย ถ้าทายถูก ไม่ต้องทำอะไร ถ้าทายผิด ขยับน้ำหนักของเส้นที่พาไปผิดลงนิดหนึ่ง แล้วให้ทายใหม่ ทำแบบนี้ซ้ำหลายพันรอบ น้ำหนักจะค่อย ๆ เข้าที่เอง

Mark I Perceptron ที่ Cornell Aeronautical Laboratory ต่อกับกล่องความละเอียด 400 จุด และแยกรูปสามเหลี่ยมกับสี่เหลี่ยมได้โดยไม่มีใครบอกมันว่าสามเหลี่ยมหน้าตาอย่างไร มันหาเองจากตัวอย่าง นี่คือการครั้งแรกที่เครื่องจักรเรียนรู้จากข้อมูลแทนที่จะทำตามคำสั่ง

แล้วในปี 1969 Marvin Minsky กับ Seymour Papert ตีพิมพ์หนังสือชื่อ Perceptrons ซึ่งพิสูจน์ด้วยคณิตศาสตร์ว่า perceptron ชั้นเดียวทำโจทย์ข้อหนึ่งไม่ได้ และโจทย์ข้อนั้นเล็กมาก

มันคือ "อย่างใดอย่างหนึ่ง แต่ไม่ใช่ทั้งสอง" — XOR

อยู่ในสวิตช์ไฟที่บ้านคุณ

หนังสือเล่มนั้นบอกด้วยว่าถ้าเอา perceptron มาซ้อนกันหลายชั้น ข้อจำกัดนี้จะหายไป ซึ่งเป็นความจริง แต่ตอนนั้นไม่มีใครรู้วิธีสอนของที่ซ้อนกันหลายชั้น เงินวิจัยหดหาย คนย้ายสาขา ช่วงนั้นถูกเรียกในภายหลังว่าฤดูหนาวของ AI ครั้งแรก

และสิ่งที่ขวางอยู่จริง ๆ ไม่ใช่ XOR มันคือคำถามที่บดท้อไปทั้งบทพูดถึง

ลองเอง: ไปที่บ้านใดบ้านหรือทางเดินที่มีสวิตช์ไฟสองตัว ตัวหนึ่งข้างบน ตัวหนึ่งข้างล่าง ลองสังเกตว่าไฟติดเมื่อไร — สวิตช์ทั้งสองตัวอยู่คนละท่ากัน ไฟติด อยู่ท่าเดียวกัน ไฟดับ ไม่มีสวิตช์ตัวไหนเป็น "ตัวเปิด" เลยสักตัว ความหมายของแต่ละตัวขึ้นกับอีกตัวเสมอ นั่นคือ XOR และนั่นคือเหตุผลที่กฎแบบ "ให้น้ำหนักสวิตช์แต่ละตัวแล้วบวกกัน" อธิบายมันไม่ได้ ไม่ว่าจะเลือกน้ำหนักเก่งแค่ไหน

บทนี้จะผิดเมื่อ: พบว่าคำว่า "cells that fire together wire together" ปรากฏในงานเขียนของ Hebb เอง หรือพบว่า Mark I Perceptron ถูกตั้งค่าน้ำหนักเริ่มต้นด้วยมือในลักษณะที่ทำให้การแยกรูปไม่ใช่การเรียนรู้จากตัวอย่างจริง

4 • ใครผิด

ปัญหาไม่ใช่ว่ามันตอบผิด ปัญหาคือไม่มีใครรู้ว่าต้องไปโทษใคร

นี่คือกำแพงที่ขวางงานด้านนี้อยู่เกือบยี่สิบปี และมันไม่ใช่กำแพงทางเทคนิคแบบที่คิดกัน — ไม่ใช่เครื่องมือไม่แรงพอ ไม่ใช่ข้อมูลไม่พอ มันเป็นกำแพงทางความคิดที่ฝังดูแทบไม่เหมือนปัญหาเลยตอนพูดออกมาครั้งแรก

ลองนึกภาพนี้ก่อน

เครือข่ายชั้นเดียวง่ายๆ มีสัญญาณเข้าสิบเส้น มีคำตอบออกหนึ่งอัน ทายผิดปั๊บ รู้ทันทีว่าต้องขยับน้ำหนักเส้นไหน เพราะทุกเส้นต่อตรงเข้าคำตอบ มันเหมือนหัวหน้าที่มีลูกน้องสิบคนนั่งอยู่ในห้องเดียวกัน งานพลาด เดินไปคุยกับคนที่ทำได้เลย

ทีนี้เอาสามชั้นมาซ้อนกัน สัญญาณเข้าที่ชั้นล่าง ส่งขึ้นชั้นกลาง ชั้นกลางส่งขึ้นชั้นบน ชั้นบนตอบ คำตอบผิด

ใครผิด

ชั้นบนอาจจะผิดเอง หรือชั้นบนอาจจะถูกต้องสมบูรณ์แบบแต่ได้ของเสียมาจากชั้นกลาง หรือชั้นกลางอาจจะไม่ผิดเลยแต่ได้ของเสียจากชั้นล่าง หรือทั้งสามชั้นผิดนิดหน่อยคนละทาง แล้วความผิดไปรวมกันตอนท้าย หรือ — อันนี้แย่ที่สุด — ชั้นล่างผิดไปทางหนึ่ง ชั้นกลางผิดไปอีกทางหนึ่งที่บังเอิญกลบความผิดอันแรกได้ครึ่งหนึ่ง

นี่คือปัญหาที่เรียกว่า **credit assignment** การจ่ายเครดิต — ทั้งเครดิตความดีความชอบและเครดิตความผิด และมันไม่ใช่ปัญหาของคอมพิวเตอร์เท่านั้น มันเป็นปัญหาของทุกระบบที่มีคนกลางเยอะ ๆ

โรงงานที่ของเสียออกมาตอนท้ายสายพาน บริษัทที่ยอดตก ทีมฟุตบอลที่แพ้ ทุกกรณีมีสิ่งเดียวกันคือ ผลลัพธ์อยู่ปลายทาง สาเหตุกระจายอยู่ต้นทาง และระหว่างสองจุดนั้นมีย้อมมากมายที่จับของชิ้นเดียวกันต่อ ๆ กันมา

วิธีที่คนใช้ตอนนั้นคือการเดา ปรับชั้นนี้ดูก่อน ถ้าดีขึ้นก็ปรับต่อ ไม่ดีขึ้นก็ถอย ซึ่งใช้เวลามหาศาลและไม่มีหลักประกันว่าจะไปถึงไหน พอชั้นเยอะขึ้น จำนวนทางที่ต้องลองก็ระเบิดขึ้นเร็วกว่าเวลาที่มนุษย์มี

คำตอบมาถึงในปี 1986 บนกระดานสีหน้าใน Nature เขียนโดย David Rumelhart, Geoffrey Hinton และ Ronald Williams ชื่อวิธีคือ **backpropagation** — การแพร่ย้อนกลับ

ความคิดของมันเล่าได้ในประโยคเดียว **ส่งความผิดพลาดย้อนกลับทางเดิมที่สัญญาณเดินมา**

ถ้าชั้นบนผิดไปเท่านี้ และชั้นกลางเส้นนี้เป็นคนป้อนของให้ชั้นบนด้วยน้ำหนักเท่านี้ ก็คำนวณได้ว่าชั้นกลางเส้นนั้นต้องรับผิดชอบเท่าไร แล้วส่งตัวเลขความรับผิดชอบนั้นลงไปชั้นถัดไปอีก ทำอย่างนี้ลงไปเรื่อย ๆ จนถึงชั้นล่างสุด ทุกเส้นในเครือข่ายได้รับใบแจ้งหนี้ของตัวเองว่าต้องชดเชยเท่าไร ไปทางไหน

เรื่องนี้มีประวัติที่ควรพูดให้ครบ วิธีคำนวณแบบเดียวกันนี้เคยถูกเขียนไว้ก่อนหน้านี้แล้วอย่างน้อยสองครั้ง — Seppo Linnainmaa ในปี 1970 ในบริบทคณิตศาสตร์ล้วน และ Paul Werbos ในวิทยานิพนธ์ปี 1974 บทความปี 1986 ไม่ใช่การค้นพบครั้งแรก มันคือครั้งที่คนในวงการเห็นและเปลี่ยนวิธีทำงาน ซึ่งเป็นคนละเรื่องกัน และในประวัติศาสตร์วิทยาศาสตร์ สองอย่างนี้มักไม่ตรงกัน

ตรงนี้ขอหยุดชี้จุดหนึ่ง เพราะมันคือหัวใจของเล่มนี้ที่กำลังจะกลับมาในบทหน้า

backpropagation ไม่ได้ลอกมาจากธรรมชาติ

สมองไม่ทำแบบนี้ ไม่มีหลักฐานว่านิวรอนส่งสัญญาณความผิดพลาดย้อนกลับทางแอกซอนแบบที่วิธีนี้ต้องการ นักประสาทวิทยาถกเถียงกันเรื่องนี้มาตลอดสามสิบปีและข้อสรุปที่ตรงไปตรงมาที่สุดคือ กลไกในสมองที่ทำหน้าที่คล้ายกันยังหาไม่เจอ

backpropagation เป็นคณิตศาสตร์ล้วน ๆ เป็นของที่มนุษย์คิดขึ้นเองเพราะมันตอบโจทย์ ไม่ใช่เพราะเห็นใครทำมาก่อน

มันคืออุโมงค์ลมของบทที่ 2 — เครื่องมือที่ไม่มีในธรรมชาติ แต่ขาดไม่ได้สำหรับการทำของที่ลอกมาให้ใช้งานจริง

เครือข่ายหลายชั้นสอนได้แล้ว กำแพงพัง คนกลับเข้ามาในสาขา

และปัญหาถัดไปก็โผล่ขึ้นมาทันที คือ **ความจำ** เครือข่ายที่สอนด้วย backpropagation ทำนายได้ดี แต่มันไม่ได้จำอะไรในความหมายที่เราใช้คำนี้กับตัวเอง บ้อนรูปครึ่งรูปให้มันแล้วขอรูปเต็มคืน มันทำไม่ได้ บ้อนเนื้อเพลงสองคำแล้วขอทั้งท่อน มันทำไม่ได้

คนที่แก้เรื่องนี้ไม่ได้มาจากสาย AI เลย และเขาไม่ได้เริ่มจากการมองสมอง

เรามองก่อนหลัก

ลองเอง: ทำอาหารอะไรก็ได้ที่ใส่ของเค็มมากกว่าหนึ่งอย่าง — ผัดสักจาน ใส่เกลือ น้ำปลา ซีอิ๊ว ชิมแล้วเค็มไป ทีนี้ตอบให้ได้ว่าตัวไหนใส่เกินไป ไม่ใช่ตอบว่า "ก็ของเค็มทั้งหมด" แต่ตอบว่าตัวไหนที่ช้อน คุณจะพบว่าตอบไม่ได้ ทั้งที่คุณเป็นคนใส่เองทั้งสามตัว และถ้าคราวหน้าอยากแก้ไขตรงจุด สิ่งเดียวที่ทำได้คือทำใหม่แล้วใส่ทีละตัว จดไว้ทุกครั้ง — ซึ่งคือสิ่งที่ backpropagation ทำให้เครื่องอัตโนมัติ

บทนี้จะผิดเมื่อ: มีหลักฐานทางประสาทวิทยาว่าเซลล์ประสาทส่งสัญญาณความผิดย้อนกลับในรูปแบบที่ backpropagation ต้องการจริง หรือพบว่างานปี 1986 อ้างอิง Linnainmaa 1970 หรือ Werbos 1974 ไว้แล้วในต้นฉบับ

5 • เชมทิสจิวในก้อนหลัก

ก้อนแม่หลักทั้งก้อนไม่เคยตัดสินใจอะไรเลย เชมทิสแต่ละอันข้างในนั้นแค่มองเพื่อนที่อยู่ติดกัน

นี่คือบทที่หนักที่สุดของเล่ม และถ้าผ่านบทนี้ไปได้ ที่เหลืออีกเจ็ดบทจะกลายเป็นเรื่องเดียวกันเล่าซ้ำ เพราะภาพที่กำลังจะได้จากตรงนี้ คือภาพที่ใช้อธิบาย ChatGPT ได้ทั้งตัว

เริ่มจากคำถามที่ดูไร้เดียงสา — **ทำไมตะปึงถึงติดแม่หลัก แต่ซอนสเดนเลสไม่ติด**

คำตอบที่นักฟิสิกส์ให้ช่วงต้นศตวรรษที่ 20 คือ ข้างในหลักทุกก้อนมีของเล็ก ๆ ที่ทำตัวเหมือนเชมทิสจิวอยู่เต็มไปหมด อะตอมแต่ละตัวมีทิสของมัน ชี้ขึ้นหรือชี้ลง เหนือ

หรือได้ ที่นี้ถ้าเข็มทิศทั้งหมดในก้อนชี้ไปทางเดียวกัน ทิศของมันเป็นบวกกันเป็นพลังที่ดึงดูดได้ ถ้ามันชี้มั่วไปคนละทาง พลังหักล้างกันหมด ก้อนนั้นก็เป็นแค่เหล็กธรรมดา

คำถามที่ตามมาทันทีคือ แล้วมันตกลงกันได้ยังไง

ในก้อนเหล็กขนาดเท่าหัวแม่มือมีอะตอมมากกว่าจำนวนดาวในกาแล็กซีหลายเท่า ไม่มีใครสั่งมัน ไม่มีตัวกลาง ไม่มีใครถือแผนผัง แต่มันตกลงกันได้ทุกครั้ง

คำตอบคือกฎข้อเดียว ที่เข็มทิศทุกอันถืออยู่เหมือนกันหมด: **หันไปทางเดียวกับเพื่อนที่อยู่ติดกัน แล้วจะสบายกว่า**

สบาย ในที่นี้มีความหมายทางฟิสิกส์ตรง ๆ คือใช้พลังงานน้อยกว่า เข็มทิศที่หันสวนกับเพื่อนข้าง ๆ เหมือนคนที่ยืนดันประตูค้างไว้ — มันต้องออกแรงตลอดเวลา ส่วนเข็มทิศที่หันตามเพื่อน เหมือนคนที่ปล่อยประตูปิดเอง ไม่ต้องออกแรงอะไรเลย

ที่นี่ประเด็นสำคัญ **ไม่มีเข็มทิศอันไหนรู้เรื่องทั้งก้อนเลย** แต่ละอันเห็นแค่เพื่อนที่อยู่ติดกันสี่หกอัน ไม่รู้ว่าก้อนนี้ใหญ่แค่ไหน ไม่รู้ว่าอีกฟากทำอะไรรออยู่ ไม่มีเป้าหมายร่วม ไม่มีการประชุม แต่พอทุกอันทำตามกฎเห็นแก่ตัวข้อเดียวกันพร้อม ๆ กัน ก้อนทั้งก้อนก็ไหลไปหาสภาพที่พลังงานรวมต่ำที่สุด เหมือนน้ำที่ไม่มีใครบอกว่าทะเลอยู่ทางไหน แต่ไหลลงทะเลได้ทุกสาย

แบบจำลองคณิตศาสตร์ของเรื่องนี้ชื่อ **Ising model** Wilhelm Lenz เสนอโครงไว้ในปี 1920 และมอบให้ลูกศิษย์ชื่อ Ernst Ising แก่ ซึ่ง Ising แก้ได้ในกรณีที่ย่างที่สุด — เข็มทิศเรียงเป็นเส้นตรงเส้นเดียว — และตีพิมพ์ในปี 1925 ผลที่ได้ทำให้เขาผิดหวัง เพราะในหนึ่งมิติ ระบบไม่เกิดการเปลี่ยนสถานะแบบที่แม่เหล็กจริงเกิด เขาสรุปว่าแบบจำลองนี้ใช้ไม่ได้และออกจากวงการฟิสิกส์ไปเป็นครู

เขาคิดผิด Lars Onsager แก้กรณีสองมิติได้ในปี 1944 และในสองมิติ ระบบเปลี่ยนสถานะจริง ชื่อของ Ising ติดอยู่กับแบบจำลองที่เขาเชื่อว่าลึ้มเหลว และมันกลายเป็นหนึ่งในของที่ถูกใช้หนักที่สุดในฟิสิกส์ทั้งศตวรรษ

มาถึงภาพที่ต้องแบกไปทั้งเล่ม

ถ้าเราลองวาดกราฟ โดยให้แกนนอนคือ "สถานะ" — คือทิศของเข็มทิศทุกอันรวมกัน เป็นชุดหนึ่ง — และให้แกนตั้งคือ "พลังงาน" ของชุดนั้น สิ่งที่ได้คือภูมิประเทศ มีเนิน มีหุบ มีที่ราบ

สถานะที่เข็มทิศทะเลาะกันเยอะคือยอดเขา พลังงานสูง ไม่มั่นคง สถานะที่เข็มทิศเข้ากันได้ดีคือกันหุบเขา พลังงานต่ำ อยู่ได้นาน

และก่อนเหล็ก กลิ้งลงหุบเขาเสมอ มันไม่ได้เลือกทางที่ดีที่สุดอย่างฉลาด มันแค่ไม่เคย กลิ้งขึ้นเนินเอง เหมือนลูกบอลที่ถูกวางลงบนพื้นขรุขระแล้วปล่อยมือ

นี่คือของทั้งหมด ลูกบอล หุบเขา แล้วก็แรงโน้มถ่วง

ในบทหน้าจะมีคนคนหนึ่งมองภาพนี้แล้วถามคำถามที่ไม่มีนักฟิสิกส์คนไหนเคยถาม

ถ้าเราเป็นคนชุดหุบเขาเองล่ะ

ลองเอง: หาแม่เหล็กติดตู้เย็นสองอันที่เป็นแบบแผ่นกลมหรือแท่ง วางบนโต๊ะแล้วดันให้เข้าหากันด้านเดียวกัน — มันจะดันกลับ ยิ่งใกล้ยิ่งดันแรง ปล่อยมือ มันจะกระเด็นออกจากกัน นั่นคือสถานะพลังงานสูง ระบบไม่อยากอยู่ตรงนั้น ทีนี้พลิกอันหนึ่งกลับด้าน แล้วปล่อยให้เข้าใกล้กัน — มันจะกระโดดเข้าหากันเองแล้วติดแน่น ปล่อยไว้ทั้งวันก็ไม่ขยับ นั่นคือกันหุบเขา สังเกตให้ดีว่าคุณไม่ได้ออกแรงในขั้นตอนที่สองเลย ระบบพาตัวเองลงไป และในก้อนเหล็กหนึ่งก้อนมีเหตุการณ์แบบนี้เกิดขึ้นพร้อมกัน เป็นล้านล้านครั้ง ไม่มีใครคุมสักครั้งเดียว

บทนี้จะผิดเมื่อ: พบว่าระบบสปินแบบ Ising ในธรรมชาติเคลื่อนไปหาสถานะพลังงานสูงได้เองโดยไม่มีพลังงานจากภายนอก หรือพบว่า Ising ตีพิมพ์ผลในมิติมากกว่าหนึ่งไว้ในงานปี 1925

6 • ความจำคือกันหุบ

ความจำไม่ใช่ของที่เก็บไว้ที่ไหนสักแห่ง มันคือกันหุบ

ประโยคนี้คือสิ่งที่ John Hopfield เสนอในปี 1982 และมันพลิกคำถามทั้งคำถาม เพราะก่อนหน้านี้ทุกคนถามว่า "สมองเก็บข้อมูลไว้ตรงไหน" ซึ่งเป็นคำถามที่หาที่อยู่

ให้ของ เหมือนหาแฟ้มในตู้ ส่วนคำถามของ Hopfield คือ "สมองทำให้ตัวเองลี้กกลับไปที่ได้มันได้อย่างไร" ซึ่งเป็นคำถามคนละชนิด

ลองดูปัญหาก่อนว่าทำไมมันถึงยาก

คอมพิวเตอร์จำของด้วยที่อยู่ ข้อมูลชิ้นหนึ่งอยู่ที่ตำแหน่งหนึ่ง อยากได้ต้องรู้ตำแหน่ง ถ้าบอกตำแหน่งผิดไปหนึ่งช่อง ได้ของผิดกลับมาทั้งชิ้น ไม่ใช่ของที่ใกล้เคียง — ของที่ไม่เกี่ยวข้องกันเลย

คนไม่ได้ทำงานแบบนั้น คนได้ยินเสียงกีตาร์สี่ตัวโน้ตแล้วนึกออกทั้งเพลง เห็นหลังคนที่เดินอยู่ไกล ๆ แล้วรู้ว่าเป็นใคร ได้กลิ่นอะไรสักอย่างแล้วเห็นภาพห้องครัวบ้านยาย ทั้งห้องในทุกกรณี สิ่งที่ป้อนเข้าไปคือ**เศษเสี้ยว** และสิ่งที่ได้กลับมาคือ**ของทั้งชิ้น** ไม่มีใครบอกที่อยู่ให้เลย

Hopfield เป็นนักฟิสิกส์ที่ย้ายมาสนใจชีววิทยา ซึ่งแปลว่าเขามีของในมือที่นักประสาทวิทยาไม่มี — เขารู้จัก Ising model

สิ่งที่เขาทำคือการลอกที่ถูกชั้นที่สุดในเล่มนี้ทั้งเล่ม **เขาไม่ได้ลอกสมอง เขาลอกสมการของก้อนเหล็ก**

ภาพจากบทที่แล้วยังอยู่ครบ เชื่อมทิศจั่วกลายเป็นนิรอน เปิดหรือปิดแทนที่จะเหนือหรือใต้ ความสัมพันธ์ระหว่างเชื่อมทิศที่อยู่ติดกันกลายเป็นน้ำหนักของเส้นเชื่อมระหว่างนิรอน ภูมิภาคประเทศพลังงานยังเป็นภูมิภาคประเทศพลังงานเหมือนเดิม และระบบยังกลิ้งลงหุบเขาเหมือนเดิม

แล้วเขาก็ถามคำถามที่นักฟิสิกส์ไม่ถาม เพราะสำหรับนักฟิสิกส์ ภูมิภาคประเทศนั้นเป็นของที่ธรรมชาติกำหนดมาแล้ว

ถ้าเราเลือกหุบเขาเองได้ล่ะ

ถ้าจะให้ระบบจำรูปตัว A ก็ตั้งน้ำหนักของเส้นเชื่อมทั้งหมดให้สถานะที่ตรงกับรูปตัว A กลายเป็นกันหุบ นั่นแหละคือการ "จำ" — ไม่ใช่การเขียนรูป A ลงในช่องเก็บของ แต่คือการ**ตัดภูมิภาคประเทศให้มีหลุมตรงที่รูป A อยู่**

จากนั้นเวลาเรียกคืน ป้อนรูป A ที่ขาดไปครึ่งหนึ่งเข้าไป ระบบจะพบว่าตัวเองยืนอยู่บนไหล่เขาใกล้ ๆ หลุมนั้น และเพราะระบบกำลังลงเสมอ มันก็กำลังลงไปในอนที่ก้นหลุม ซึ่งคือรูป A เต็ม ๆ

ไม่มีใครค้นหาอะไรเลย ไม่มีการเทียบกับของที่เก็บไว้ ไม่มีที่อยู่ ระบบแค่ตกลงไปในที่ที่มันตกลงไปได้

นี่คือเหตุผลที่ความจำแบบนี้ทนต่อความเสียหาย ขาดไปสามสิบเปอร์เซ็นต์ก็ยังกลงหลุมเดิม มีจุดรวบรวนกียังลง เพราะเรื่องทั้งหมดไม่ได้ขึ้นกับความถูกต้องของข้อมูลที่ป้อนเข้าไป มันขึ้นกับว่าคุณยืนอยู่ในเขตของหลุมไหน

และมันอธิบายเรื่อง que everyone เคยเจอด้วย — จำผิดเป็นคนอื่นที่หน้าคล้ายกัน สองหลุมที่ขุดใกล้กันเกินไปจะเริ่มกินเขตกัน ขอบระหว่างมันเลือนลง เศษเสี้ยวที่ควรพาไปหลุมหนึ่งก็ลงไปลงอีกหลุม สมองก็ทำแบบนี้ และ Hopfield network ก็ทำแบบนี้ ด้วยเหตุผลเดียวกันเป๊ะ

ข้อจำกัดของมันวัดได้ชัดเจน และนี่คือสิ่งที่จะกลับมาสำคัญมากในบทที่ 10 —

เครือข่ายแบบ Hopfield ดั้งเดิมเก็บของได้น้อยมากเมื่อเทียบกับจำนวนนิรอน

ประมาณการมาตรฐานคือราว 0.14 ของจำนวนนิรอน นิรอนหนึ่งพันตัวเก็บรูปได้ประมาณหนึ่งร้อยสี่สิบบิต เกินกว่านั้นหลุมจะทับกันจนพัง แล้วระบบจะเริ่มกลิ้งลงไปในอนในหลุมที่ไม่ใช่ของใครเลย — สถานะลูกผสมที่ไม่เคยถูกสอนให้จำ

บทความปี 1982 ชื่อ Neural networks and physical systems with emergent collective computational abilities ตีพิมพ์ใน PNAS และคำที่สำคัญที่สุดในชื่อนั้นคือ **emergent** — ความสามารถในการคำนวณไม่ได้อยู่ในนิรอนตัวไหนเลย มันโผล่ขึ้นมาจากการทำงานของโง่ ๆ จำนวนมากทำตามกฎง่าย ๆ ข้อเดียวกันพร้อมกัน

อันเดียวกับเคมีในก้อนเหล็กทุกประการ

แต่มีปัญหาใหญ่ข้อหนึ่งที่ Hopfield network แก้ไม่ได้ และมันคือปัญหาที่ลูกบอลทุกลูกในโลกนี้มี

ลูกบอลไม่รู้ว่ามีมันนอนอยู่ในหลุมที่ถูกหรือเปล่า มันรู้แค่ว่ารอบตัวสูงกว่ามันหมด อาจจะมียุคที่ลึกกว่านี้อยู่ถัดไปอีกสองเนิน มันไม่มีทางรู้ และมันไม่มีแรงจะปีนออกไปดู

ทางออกของเรื่องนี้มาจากข้างใต้เหล็ก

ลองเอง: นึกเพลงที่คุณชอบสมัยมัธยม แล้วเริ่มร้องในใจจากท่อนกลาง ๆ ที่จำได้ไม่ครบ ท่อนที่จำได้แค่สี่ห้าคำ คุณจะพบว่าส่วนที่เหลือมาเอง มันไม่ได้มาเรียงจากต้นเพลง และคุณไม่ได้ "ค้นหา" มัน สังเกตอีกข้อ — ถ้าจำผิดคำหนึ่งแล้วร้องต่อ ทำนองมักจะยังถูก เพราะสิ่งที่พาคุณไปไม่ใช่ตัวคำ มันคือรูปรวมที่คุณกลิ้งลงไปอยู่แล้ว

บทนี้จะผิดเมื่อ: มีหลักฐานว่าความจุของ Hopfield network ดั้งเดิมไม่ได้แปรตามจำนวนนิเวศอย่างมีประสิทธิภาพมาตรฐานระบุ หรือมีการทดลองที่แสดงว่าการเรียกคืนจากเศษเสี้ยวในเครือข่ายแบบนี้ต้องอาศัยการค้นหาแบบเทียบทีละรายการ

7 • เขย่ามันซะ

ถ้าลูกบอลติดอยู่ในหลุมผิด อย่าไปดึงมันขึ้น — เขย่าโต๊ะ

นี่เป็นคำตอบที่ดูไม่ฉลาดเลยตอนได้ยินครั้งแรก และมันมาจากอาชีพที่เก่าแก่กว่าคอมพิวเตอร์หลายพันปี

ช่างตีเหล็กรู้เรื่องนี้มานานแล้ว เวลาจะทำให้โลหะแข็งแรงและไม่เปราะ เขาไม่ได้ทุบมันให้เข้าที่ เขาเผาให้ร้อนแล้วปล่อยให้เย็นลงช้า ๆ กระบวนการนี้เรียกว่า annealing และเหตุผลที่มันได้ผลคือของที่ร้อนสั่นแรง อะตอมกระโดดไปมาได้ พอค่อย ๆ เย็นลงอะตอมก็ค่อย ๆ ตกลงเข้าที่ของมัน ถ้าปล่อยให้เย็นเร็วเกินไป อะตอมจะค้างอยู่ในตำแหน่งมั่ว ๆ ที่มันบังเอิญอยู่ตอนนั้น โลหะจะเปราะและแตกง่าย

แปลเป็นภาษาของบทที่แล้ว: ร้อน แปลว่าลูกบอลกระโดดได้ เย็น แปลว่าลูกบอลกลิ้งลงอย่างเดียว

ลูกบอลที่เย็นสนิทติดหลุมแรกที่เจอเสมอ ลูกบอลที่ร้อนจัดกระโดดไปทั่วจนไม่ยอมนอนที่ไหนเลย ของที่ต้องการคือเริ่มร้อนแล้วค่อย ๆ เย็น — ตอนร้อนมันสำรวจได้

กว้าง หลุมตื้น ๆ กักมันไว้ไม่อยู่ พอเย็นลงเรื่อย ๆ มันจะหลุดออกจากหลุมตื้นได้แต่
ออกจากหลุมลึกไม่ได้ สุดท้ายมันจะนอนอยู่ในหลุมที่ลึกที่สุดเท่าที่เดินไปเจอ

วิธีนี้ถูกเขียนเป็นอัลกอริทึมในปี 1983 โดย Scott Kirkpatrick, Daniel Gelatt และ
Mario Vecchi ตีพิมพ์ใน Science ชื่อของมันคือ **simulated annealing** — การอบ
จำลอง ซึ่งเป็นชื่อที่ชื่อสตั้มมาก มันคือการเอาวิธีของช่างตีเหล็กมาใช้กับตัวเลข

ที่นี่ Geoffrey Hinton กับ Terry Sejnowski เอาความคิดนี้ไปใส่ใน Hopfield network
ในปี 1983 และตั้งชื่อของที่ได้ว่า **Boltzmann machine** ตามชื่อ Ludwig Boltzmann
นักฟิสิกส์ชาวออสเตรียที่เขียนกฎว่าระบบที่มีอุณหภูมิหนึ่ง ๆ จะพบอยู่ในสถานะไหน
ด้วยความน่าจะเป็นเท่าไร

แต่สิ่งที่พวกเขาเพิ่มเข้าไปอีกอย่างสำคัญกว่าอุณหภูมิ และมันคือสิ่งที่คณะกรรมการ
โนเบลอ้างถึงในปี 2024

พวกเขาแบ่งนิวรอนออกเป็นสองพวก

พวกแรกเรียกว่า **visible units** คือหน่วยที่ได้เห็นข้อมูลจริง รูปที่ป้อนเข้าไปตกลงบน
หน่วยพวกนี้

พวกที่สองเรียกว่า **hidden units** คือหน่วยที่**ไม่ได้เห็นอะไรเลย** ไม่มีข้อมูลตกลงบน
มัน ไม่มีคำตอบที่ถูกให้มันเทียบ มันแค่ต่ออยู่กับหน่วยพวกแรก และมันถูกปล่อยให้หา
เองว่าจะตั้งตัวยังไง

ฟังดูเหมือนของที่ไร้ประโยชน์ที่สุดในเครื่อง และมันคือส่วนที่ทำให้เครื่องเข้าใจอะไร
บางอย่าง

เหตุผลคือถ้าหน่วยที่ซ่อนอยู่ตัวหนึ่งพบว่า ทุกครั้งที่หน่วยที่มองเห็นสามตัวนี้ติดพร้อม
กัน มันเองก็ควรติดด้วย หน่วยนั้นก็เพิ่งค้นพบรูปแบบที่ไม่มีใครบอกมัน สมมติป้อนรูป
ตัวอักษรเข้าไปเป็นพัน ๆ รูป หน่วยซ่อนตัวหนึ่งอาจจะกลายเป็นคนที่ติดทุกครั้งที่มี
เส้นตั้งอยู่ตรงกลาง อีกตัวติดทุกครั้งที่มีวงกลมปิด ไม่มีใครสอนคำว่าเส้นตั้งหรือ
วงกลมให้มันเลย มันหาเองเพราะรูปแบบนั้นทำให้พลังงานรวมของระบบต่ำลง

นี่คือสิ่งที่คำประกาศรางวัลปี 2024 เรียกว่า "วิธีที่สามารถค้นพบคุณสมบัติในข้อมูลได้ด้วยตัวเอง"

มันคือการเรียนรู้แบบที่ไม่มีใครเฉลย เรียกในวงการว่า unsupervised learning และมันใกล้เคียงกับสิ่งที่เด็กทำมากกว่าการเรียนรู้แบบมีเฉลย เด็กที่เห็นหมาสามสิบตัวไม่ได้ถูกสอนนิยามของหมา เขาสร้างหมวดของตัวเองขึ้นมาจากของที่เห็นซ้ำ ๆ แล้วค่อยมีคนมาติดป้ายชื่อทีหลัง

ราคาของมันคือเวลา Boltzmann machine ฉบับดั้งเดิมสอนช้ามาก เพราะทุกครั้งที่จะปรับน้ำหนักสักที ต้องปล่อยให้ระบบเขย่าตัวเองจนเข้าที่ก่อน แล้ววัด แล้วทำซ้ำ ในเครื่องยุคทศวรรษ 1980 นี่แปลว่าโจทย์ขนาดจริงจังใช้เวลานานเกินกว่าจะรอไหว

ของที่แก้เรื่องนี้เกิดขึ้นจากการตัดสายออก ไม่ใช่การเพิ่มอะไรเข้าไป และมันได้ชื่อว่า Restricted Boltzmann Machine

สิบปีถัดมามันคือสิ่งที่ปลุก deep learning ขึ้นมาจากหลุม แล้วมันก็หายไปเงียบ ๆ

บทหน้าเป็นเรื่องของมัน

ลองเอง: หาเหรียญสักสิบเหรียญกับกล่องหรือถาดที่มีขอบ วางเหรียญคว่ำหงายมั่ว ๆ แล้วตั้งเป้าว่าอยากให้หงายทั้งหมด เขย่าเบา ๆ — แทบไม่มีเหรียญไหนพลิก ระบบติดอยู่ที่เดิม เขย่าแรงเต็มที่ — เหรียญพลิกมั่วไปหมด เขย่าหยุดแล้วก็ยังมั่วเหมือนเดิม ทีนี้ลองเขย่าแรงก่อนแล้วค่อย ๆ เบาลงจนหยุด จะยังไม่ได้หงายครบสิบ เพราะเหรียญไม่มีเหตุผลทางพลังงานที่จะชอบหงายมากกว่าคว่ำ แต่คุณจะรู้สึกได้ด้วยมือว่าของที่คุยกำลังคุมอยู่คือ **ความกว้างของการสำรวจ** ไม่ใช่ทิศทางของมัน นั่นคือสิ่งเดียวที่อุณหภูมิทำ

บทนี้จะผิดเมื่อ: มีหลักฐานว่า Boltzmann machine ค้นพบรูปแบบในข้อมูลได้โดยไม่ต้องมี hidden units หรือพบว่าคำประกาศรางวัลโนเบลปี 2024 ไม่ได้อ้างถึงความสามารถในการค้นพบคุณสมบัติด้วยตัวเองว่าเป็นผลงานของ Hinton

8 · บันไดที่ถูกถอดออก

บันไดที่ดีที่สุดคือบันไดที่ถอดออกแล้วตึกไม่ล้ม

RBM คือบันไดแบบนั้น และเรื่องของมันถูกเล่าผิดบอว่าเป็นเทคโนโลยีที่ล้ำสมัย ความจริงตรงข้าม มันหายไปเพราะมันทำงานสำเร็จ

ย้อนไปที่ปัญหาค้างจากบทที่แล้วก่อน Boltzmann machine สอนช้าเกินกว่าจะใช้งานจริง เหตุผลคือทุกหน่วยต่ออยู่กับทุกหน่วย รวมถึงหน่วยที่อยู่ชั้นเดียวกัน แปลว่าเวลาจะคำนวณว่าหน่วยไหนควรติดหรือดับ ต้องรอดูหน่วยอื่นในชั้นเดียวกันด้วย ซึ่งหน่วยอื่นนั้นก็รอดูหน่วยนี้อยู่เหมือนกัน ทุกอย่างพึ่งพากันเป็นวง ทางเดียวที่จะหาคำตอบคือปล่อยให้มันเขย่าตัวเองไปเรื่อย ๆ จนนิ่ง แล้วค่อยวัด

ทางแก้คือการตัดสายออก ไม่ใช่เพิ่มอะไร

Restricted Boltzmann Machine ตัดเส้นเชื่อมภายในชั้นเดียวกันทั้งหมด หน่วยที่มองเห็นต่อกับหน่วยที่ซ่อนอยู่เท่านั้น ไม่มีหน่วยไหนในชั้นเดียวกันคุยกันเอง คำว่า restricted มาจากตรงนี้ — มันคือ Boltzmann machine ที่ถูกจำกัด

ผลของการตัดสายชุดนั้นเปลี่ยนทุกอย่าง เพราะพอไม่มีใครในชั้นเดียวกันรอกันอยู่ หน่วยทั้งชั้นก็คำนวณพร้อมกันได้ทีเดียว จากการรอให้ระบบนิ่ง กลายเป็นการคำนวณสองรอบ ขึ้นแล้วลง

Hinton เขียนวิธีสอนมันในปี 2002 ชื่อ **contrastive divergence** และความคิดของมันหน้าด้านอย่างน่าชื่นชม — ทฤษฎีบอกว่าต้องปล่อยให้ระบบเขย่าจนเข้าที่ก่อนวัด เขาถามว่าถ้าไม่รอละ ถ้าเขย่าแค่ก้าวเดียวแล้ววัดเลยละ คำตอบทางทฤษฎีคือค่าที่ได้จะไม่ถูกต้อง คำตอบทางปฏิบัติคือมันผิดในทิศทางที่ไม่ทำลายอะไร และมันเร็วขึ้นหลายร้อยเท่า

ปี 2006 คือปีที่บันไดนี้ถูกใช้

Hinton กับ Simon Osindero และ Yee Whye Teh ตีพิมพ์วิธีที่เรียกว่า **greedy layer-wise pretraining** ความคิดคือแทนที่จะพยายามสอนเครือข่ายลึก ๆ ทั้งก้อน

พร้อมกัน — ซึ่งตอนนั้นยังทำไม่สำเร็จ เพราะสัญญาณความผิดที่ส่งย้อนกลับจากบทที่ 4 จางลงจนแทบเป็นศูนย์ก่อนถึงชั้นล่าง — ให้สอนทีละชั้น

สอนชั้นล่างสุดด้วย RBM ตัวเดียวจนมันเจอรูปแบบง่าย ๆ ในข้อมูลดิบ แล้วแข่งชั้นนั้นไว้ เอาสิ่งที่ชั้นล่างเห็นเป็นข้อมูลเข้าของชั้นถัดไป สอนชั้นนั้นด้วย RBM อีกตัว ทำอย่างนี้ขึ้นไปทีละชั้น แต่ละชั้นเรียนของที่เป็นนามธรรมขึ้นกว่าชั้นใต้มัน เสร็จแล้วค่อยเอา backpropagation มาจนทั่วทั้งก้อนอีกที

ที่มันได้ผลเพราะการสอนแบบนี้ไม่ได้เริ่มจากน้ำหนักรั่ว ๆ มันเริ่มจากน้ำหนักที่ผ่านการจัดระเบียบตามโครงสร้างของข้อมูลมาแล้ว ลูกบอลไม่ได้ถูกปล่อยลงมาจากที่ไหนก็ไม่ว่า มันถูกวางไว้ใกล้หุบที่ถูกต้องตั้งแต่แรก

ปีเดียวกันนั้น Hinton กับ Ruslan Salakhutdinov ตีพิมพ์ใน Science แสดงว่าเครือข่ายลึกที่สอนวิธีนี้บีบข้อมูลได้ดีกว่าวิธีมาตรฐานที่ใช้กันมาสามสิบปี

นั่นคือช่วงที่คำว่า deep learning เริ่มมีน้ำหนัก และคณะกรรมการโนเบลอ้างถึงงานปี 2006 ชุดนี้โดยตรงในคำประกาศรางวัลปี 2024

ทีนี้ส่วนที่ต้องเล่าให้ครบ

ประมาณปี 2010 ถึง 2012 มีของสามอย่างมาพร้อมกัน

อย่างแรก มีคนพบว่าถ้าเลือกวิธีตั้งน้ำหนักเริ่มต้นให้ถูกขนาด สัญญาณความผิดจะเดินทางลงไปถึงชั้นล่างได้โดยไม่จางหายไปก่อน

อย่างที่สอง มีการเปลี่ยนฟังก์ชันที่นิวรอนใช้ตัดสินใจว่าจะติดหรือดับ จากเส้นโค้งรูปตัว S มาเป็นเส้นหักมุมง่าย ๆ ที่เรียกว่า ReLU ซึ่งไม่ทำให้สัญญาณจางลงเวลาเดินย้อนกลับ

อย่างที่สาม GPU ราคาผู้บริโภคแรงพอที่จะสอนเครือข่ายใหญ่ ๆ ได้ในเวลาไม่กี่วัน

ปี 2012 AlexNet ชนะการแข่งขัน ImageNet ด้วยความผิดพลาดที่ต่ำกว่าอันดับสองอย่างมาก มันถูกสอนบนการ์ดจอสำหรับเล่นเกมสองใบ

และมันไม่ได้ใช้ RBM pretraining เลย

ข้อสรุปที่ว่าการลงตัวในช่วงหลังจากนั้นคือ การเตรียมน้ำหนักแบบไม่มีเฉลยไม่ใช่สิ่งจำเป็นสำหรับการสอนเครือข่ายลึก มันช่วยได้จริงในยุคที่ยังไม่มีของสามอย่างข้างบน และเมื่อของสามอย่างนั้นมาครบ การสอนแบบมีเฉลยตรง ๆ ทำได้ดีกว่า

ตรงนี้ขอชี้ให้ชัด เพราะมันเป็นสิ่งที่คนเล่าผิดบ่อยที่สุดเกี่ยวกับ RBM

RBM ไม่ได้แพ้ มันชนะจนตัวเองไม่จำเป็น

งานของมันคือพาเครือข่ายลึกข้ามช่วงที่ยังไม่มีใครสอนมันได้ มันทำงานนั้นสำเร็จ คนจึงกล้าลงทุนกับเครือข่ายลึกต่อ แล้วของที่ดีกว่าก็ถูกสร้างขึ้นมาในช่วงที่ได้เวลานั้นมา มันได้ถูกถอดออกเพราะตึกยืนได้แล้ว ไม่ใช่เพราะมันได้หัก

นี่คือรูปแบบที่จะเจออีกครั้งในบทที่ 11 และเป็นเหตุผลที่คำถามว่า "ของเก่าจะกลับมาไหม" มักถูกถามผิดข้อ ของที่หายไปเพราะพ่ายแพ้อาจกลับมาได้ถ้าเงื่อนไขเปลี่ยน ส่วนของที่หายไปเพราะทำงานเสร็จ จะกลับมาอีกต่อเมื่อมีงานใหม่ที่หน้าตาเหมือนงานเดิม

และหุบเขากับลูกบอลก็ยังไม่ได้หายไปไหนเลย มันกำลังจะกลับมาในบทที่ 10 ในรูปที่ไม่มีใครคาดคิด

ลองเอง: เปิดแอปรูปภาพในมือถือ แล้วดูว่ามันจัดกลุ่มหน้าคนให้คุณเอง — ไม่มีใครสอนเครื่องว่าใบหน้าคนนี่ชื่ออะไรก่อนที่มันจะรวมรูปเป็นกอง มันรวมเพราะรูปพวกนั้นเหมือนกันในแบบที่มันหาเจอเอง แล้วรอให้คุณมาติดชื่อทีหลัง นั่นคือรูปเดียวกันกับที่ hidden units ทำในบทที่แล้ว ต่างกันที่เครื่องมือข้างใน ไม่ใช่ที่ความคิด

บทนี้จะผิดเมื่อ: พบว่า AlexNet ปี 2012 ใช้การเตรียมน้ำหนักแบบไม่มีเฉลย หรือพบผลการทดลองยืนยันว่าการเตรียมน้ำหนักด้วย RBM ยังให้ผลดีกว่าการสอนแบบมีเฉลยตรง ๆ อย่างสม่ำเสมอในเครือข่ายขนาดใหญ่ยุคปัจจุบัน

9 • แล้วเราก็เลิกประหยัด

Super Mario Bros ทั้งเกมคือ 40 กิโลไบต์ และ 8 ใน 40 นั้นเป็นรูป

ตัวเลขชุดนี้แยกได้ชัดเจนเพราะตลับ NES แบ่งหน่วยความจำเป็นสองก้อนคนละชนิด — PRG-ROM 32 กิโลไบต์ เก็บโค้ดกับข้อมูลของเกม และ CHR-ROM 8 กิโลไบต์ เก็บภาพตัวละครกับฉากทั้งหมด ยี่สิบเปอร์เซ็นต์ของตลับคือศิลปะ ตั้งแต่ปี 1985

ตัวเลขนี้เป็นของที่ถูยกยกมาบอยในบทสนทนาเรื่อง "สมัยก่อนเขาเขียนโปรแกรมเก่งกว่า" และก่อนจะไปต่อ ต้องแยกสองอย่างออกจากกันก่อน เพราะมันคนละเรื่อง

เรื่องแรกเป็นความจริงและน่าสนใจมาก **ข้อจำกัดผลิตความสว่าง** เครื่อง NES มีหน่วยความจำสำหรับทำงานอยู่ 2 กิโลไบต์ สองพันกว่าไบต์ สำหรับทุกอย่างที่เกมต้องจำระหว่างเล่น ตลับมีราคาต่อกิโลไบต์ที่แพงจริงในเชิงการผลิต คนที่เขียนเกมในสภาพนั้นไม่ได้ประหยัดเพราะมีคุณธรรม เขาประหยัดเพราะมีกำแพง และกำแพงนั้นบังคับให้เกิดวิธีที่ฉลาดจริง ๆ — ฉากที่สร้างจากกฎแทนการเก็บเป็นภาพ ตัวละครที่ใช้รูปเดียวกันพลิกกลับด้าน เมฆกับพุ่มไม้ใน Super Mario Bros ที่เป็นรูปเดียวกันคนละสี

เรื่องที่สองเป็นสิ่งที่ต้องแย้ง **ของที่บวมขึ้นในเกมสมัยนี้ไม่ใช่โค้ด**

เกมขนาดร้อยกิกะไบต์ไม่ได้มีตรรกะมากกว่า Super Mario Bros เป็นล้านเท่า มันมีพื้นผิวความละเอียดสูง เสียงพากย์หลายภาษา วิดีโอ โมเดลสามมิติ แผนที่ขนาดใหญ่ สิ่งเหล่านี้คือ asset และ asset ขยายตามความละเอียดของจอกับหูของคนเล่น ไม่ได้ขยายตามความซับซ้อนของโปรแกรม ส่วนเอนจินเกมสมัยใหม่ที่ต้องดันภาพทหสิบเฟรมต่อวินาทีโดยไม่กระตุก คือโค้ดที่ถูกไล้บีบอย่างหนักที่สุดประเภทหนึ่งเท่าที่มนุษย์เขียนกัน

ตรงนี้ต้องบอกสิ่งที่ยังบอกไม่ได้ด้วย — **สัดส่วนโค้ดต่อ asset ของเกมสมัยใหม่ เล่มนี้ไม่มีตัวเลขที่ตรวจแล้ว** ของที่ยืนยันได้คือ 32 ต่อ 8 ของตลับ NES และคำอธิบายเชิงกลไกว่าทำไมสองฝั่งถึงโตคนละอัตรา ใครก็ตามที่มีตัวเลขจริงของฝั่งสมัยใหม่ ขอให้ส่งมา แล้วเล่มนี้จะแก้

แล้วคำว่า "เราเลิกประหยัด" มันผิดทั้งหมดหรือเปล่า

ไม่ผิด แต่มันเกิดคนละที่กับที่คนชี้

สิ่งที่หายไปจริงคือ การประหยัดในที่ที่ไม่มีกำแพงอีกแล้ว เว็บแอปที่โหลดไลบรารีท่าแมกะไบต์เพื่อแสดงข้อความสามบรรทัด แอปที่กินแรมเป็นกิกะไบต์เพื่อทำงานของโปรแกรมที่เคยกินหนึ่งเมกะ สิ่งเหล่านี้เกิดขึ้นเพราะเครื่องรับไหว และเมื่อเครื่องรับไหว การประหยัดกลายเป็นต้นทุนที่ไม่มีใครจ่าย — เวลาของโปรแกรมเมอร์แพงกว่าแรมเสมอ ตราบใดที่แรมยังพอ

ในสาขา AI เรื่องเดียวกันเกิดขึ้นชัดกว่าที่ไหน

ตั้งแต่ราวปี 2012 ถึงต้นทศวรรษ 2020 คำตอบที่ได้ผลที่สุดกับเกือบทุกโจทย์คือ **ทำให้ใหญ่ขึ้น** โมเดลใหญ่ขึ้น ข้อมูลเยอะขึ้น เครื่องแรงขึ้น วิธีการเดิมทั้งหมดยังใช้ได้ แค่ขยายขนาด และสิ่งที่วัดได้คือผลดีขึ้นอย่างต่อเนื่องในแบบที่คาดการณ์ได้ ซึ่งเป็นเรื่องผิดคาดสำหรับคนในวงการเอง

ในสภาพแบบนั้น งานที่ใช้สมองไปกับการทำให้ของเล็กลงคืองานที่ให้ผลตอบแทนแย่มากที่สุด ใช้เวลาหกเดือนบีบโมเดลให้เร็วขึ้นสองเท่า หรือรอหกเดือนแล้วซื้อการ์ดรุ่นใหม่ที่เร็วขึ้นสองเท่า คำตอบตรงไปตรงมาและไม่มีใครต้องเถียงกัน

RBM หายไปในช่วงนี้ด้วยเหตุผลเดียวกัน มันคือวิธีที่ฉลาดสำหรับโลกที่ยังไม่มีของมากพอ พอของมีมากพอ ความฉลาดของมันเล็กเป็นข้อได้เปรียบ

และนี่คือจุดที่ควรหยุดคิดสักนิด เพราะรูปแบบนี้ไม่ได้เกิดกับซอฟต์แวร์เท่านั้น

ทุกครั้งที่ทรัพยากรมีเหลือเพื่อ วิธีที่ประหยัดทรัพยากรนั้นจะถูกทิ้ง และมันถูกทิ้งด้วยเหตุผลที่ถูกต้องในตอนนั้น ไม่ใช่เพราะคนเสื่อมลง ความสง่างามไม่เคยเป็นคุณสมบัติของยุคสมัย มันเป็นผลของกำแพง

แปลว่าคำถามที่ควรถามไม่ใช่ "การประหยัดจะกลับมาไหม"

คำถามที่ควรถามคือ **กำแพงถัดไปอยู่ตรงไหน**

คำตอบอยู่ในบทที่ 11 และมันมีตัวเลขกำกับ

ลองเอง: เปิดที่เก็บข้อมูลในเครื่องหรือในมือถือแล้วเรียงแอปตามขนาด เลือกตัวที่ใหญ่ที่สุดสักตัว แล้วถามตัวเองว่าสิ่งที่มันทำได้ต่างจากโปรแกรมที่ทำเรื่องเดียวกันเมื่อสิบปีก่อนมากพอ ๆ กับขนาดที่ต่างกันหรือเปล่า ถ้าเป็นเกมหรือแอปรูปภาพ คำ

ตอบมักจะสมเหตุสมผล เพราะของที่โตคือภาพกับเสียง ถ้าเป็นแอปที่แสดงตัวหนังสือเป็นหลัก คำตอบมักจะไม่สมเหตุสมผล และตรงนั้นคือที่ที่กำแพงหายไปจริง ๆ

บทนี้จะผิดเมื่อ: มีการวัดที่แสดงว่าส่วนที่โตขึ้นของขนาดติดตั้งเกมสมัยใหม่เป็นโค้ดปฏิบัติการมากกว่าเป็น asset หรือพบว่าตลับ Super Mario Bros ไม่ได้แบ่ง 32 กิโลไบต์ PRG กับ 8 กิโลไบต์ CHR ตามที่ระบุ

10 · หลุมที่ลึกและแคบ

สมการที่โมเดลภาษาใช้เลือกคำถัดไป คือสมการเดียวกับที่ก้อนเหล็กใช้หาทิศ

ประโยชน์ไม่ใช่คำเปรียบเทียบ มันคือผลทางคณิตศาสตร์ที่มีคนพิสูจน์ไว้ในปี 2020 และมันเป็นเหตุผลที่รางวัลโนเบลสาขาฟิสิกส์ปี 2024 ตกกับงานที่ทุกคนเรียกว่า AI กลับไปที่ปัญหาค้างจากบทที่ 6 ก่อน

Hopfield network ดั้งเดิมเก็บของได้น้อย ราวสิบสี่เปอร์เซ็นต์ของจำนวนนิวรอน ปัญหาของมันมองเห็นได้ชัดถ้านึกถึงภูมิประเทศ หุบเขาที่ขุดด้วยวิธีดั้งเดิมนั้นกว้างและตื้น พอขุดหลายหลุมใกล้กัน ไหล่เขาของแต่ละหลุมก็เริ่มกินกัน จนสุดท้ายแทนที่จะได้หลุมสิบหลุม กลายเป็นแอ่งใหญ่แอ่งเดียวที่ไม่ตรงกับของที่จำไว้เลยสักชิ้น

ลูกบอลยังกลิ้งลงเหมือนเดิม มันแค่ไปนอนในที่ที่ไม่มีความหมาย

Dmitry Krotov กับ John Hopfield กลับมาที่ปัญหานี้ในปี 2016 ด้วยงานชื่อ Dense Associative Memory for Pattern Recognition และสิ่งที่พวกเขาเปลี่ยนคือรูปร่างของภูมิประเทศ ไม่ใช่จำนวนนิวรอน

ถ้าเปลี่ยนกฎที่คำนวณพลังงาน ให้ความต่างระหว่าง "ตรงกับของที่จำไว้" กับ "เกือบตรง" ถูกขยายให้ห่างกันมากขึ้น หุบเขาจะเปลี่ยนรูปจากแอ่งกว้าง ๆ กลายเป็นหลุมลึกและแคบ เหมือนเจาะบ่อแทนการขุดแอ่ง

และหลุมลึกแคบวางใกล้กันได้มากกว่าแอ่งกว้างมหาศาล

ผลคือความจุกระโดดขึ้นหลายระดับ จากที่เคยแปรตามจำนวนนิวรอนแบบตรง ๆ กลายเป็นแปรตามกำลัง และในงานรุ่นถัดมาที่ใช้รูปแบบเอกซ์โพเนนเชียล ความจุโตเร็วกว่านั้นอีก

ปี 2020 กลุ่มของ Hubert Ramsauer ที่มหาวิทยาลัย Linz ตีพิมพ์งานชื่อ Hopfield Networks is All You Need ซึ่งตั้งชื่อล้อบทความที่สร้างสถาปัตยกรรม Transformer ในปี 2017 ที่ชื่อ Attention Is All You Need และสิ่งที่พวกเขาพิสูจน์คือของสองอย่างนั้นเป็นของเดียวกัน

เนื้อหาของงานนั้นมีสามข้อ

ข้อแรก Hopfield network รุ่นใหม่ที่สถานะเป็นค่าต่อเนื่องแทนที่จะเป็นเปิดปิด เก็บรูปแบบได้จำนวนมหาศาล ด้วยความผิดพลาดในการเรียกคืนที่เล็กลงอย่างรวดเร็ว

ข้อสอง มันเรียกคืนของได้ใน**การอัปเดตครั้งเดียว** ลูกบอลไม่ต้องกลิ้งทีละนิดจนถึงกันหลุมอีกแล้ว มันกระโดดถึงในทีเดียว

ข้อสาม ซึ่งเป็นข้อที่สำคัญที่สุด — **กฎการอัปเดตครั้งเดียวนั้น คือกลไก attention ของ Transformer**

ไม่ใช่คล้าย ไม่ใช่เทียบเคียงได้ เป็นสมการเดียวกัน

แปลเป็นภาษาคอนอ่าน: เวลาโมเดลภาษาอ่านประโยคหนึ่ง ทุกคำจะถามว่าคำไหนในประโยคนี้เกี่ยวกับฉันบ้าง แล้วดึงความหมายจากคำเหล่านั้นมาผสมเป็นความเข้าใจของตัวเอง กลไกที่ทำเรื่องนี้ชื่อ attention และสิ่งที่งานปี 2020 แสดงคือ การถามแบบนั้นหนึ่งครั้ง **มีค่าเท่ากับการที่ลูกบอลกลิ้งลงหุบเขาในภูมิประเทศพลังงานจนถึงกัน** คำที่ถูกดึงออกมาคือสิ่งที่อยู่กันหลุม

และงานนั้นยังจำแนกไว้ด้วยว่าหลุมที่ลูกบอลลงไปในอนมีสามแบบ — แบบที่เฉลี่ยทุกอย่างในประโยคเข้าด้วยกัน แบบที่เฉลี่ยเฉพาะกลุ่มย่อยกลุ่มหนึ่ง และแบบที่ลงไปเจาะจงที่ของขึ้นเดียว สิ่งที่วัดได้คือขั้นต้น ๆ ของโมเดลมักทำแบบเฉลี่ยรวม ส่วนขั้นสูงขึ้นไปทำแบบเจาะจงกลุ่มย่อย

นั่นคือคำอธิบายเชิงกลไกว่าโมเดลภาษา "เข้าใจบริบท" ยังไง มันไม่ได้เข้าใจ มันกลิ้งลงหลุม แต่หลุมถูกขุดไว้ดีพอจนผลที่ออกมาใช้แทนกันได้

นี่เรื่องรางวัล

คำอ้างอิงอย่างเป็นทางการของรางวัลโนเบลสาขาฟิสิกส์ปี 2024 คือ "สำหรับการค้นพบและสิ่งประดิษฐ์พื้นฐานที่ทำให้การเรียนรู้ของเครื่องด้วยโครงข่ายประสาทเทียมเป็นไปได้" โดยระบุแยกว่า Hopfield สร้างโครงสร้างที่เก็บและสร้างข้อมูลกลับขึ้นมาใหม่ได้ ส่วน Hinton คิดค้นวิธีที่ค้นพบคุณสมบัติในข้อมูลได้ด้วยตัวเอง

เสียงคัดค้านมีจริงและควรพิมพ์ไว้ตรง ๆ ฝ่ายที่ไม่เห็นด้วยบอกว่านี่คือวิทยาการคอมพิวเตอร์ ไม่ใช่ฟิสิกส์ และหมวดรางวัลถูกกำหนดไว้ในพินัยกรรมของ Nobel จะยึดไม่ได้ตามใจ

ฝ่ายที่เห็นด้วยตอบด้วยสิ่งที่เล่ามาเก่าบทนี้ — เครื่องมือที่ใช้คือกลศาสตร์สถิติ ภูมิประเทศพลังงานคือของที่นักฟิสิกส์ใช้อธิบายแก้ว แม่เหล็ก และวัสดุที่จัดตัวเองไม่ลงตัว สมการที่อยู่ข้างใน attention คือสมการของระบบสปิน และความรู้ที่ได้ก็เดินทางกลับเข้าไปในฟิสิกส์ด้วย ไม่ได้ไหลออกทางเดียว

ข้อสรุปที่ซื่อสัตย์ที่สุดคือทั้งสองฝ่ายพูดถึงคนละคำถาม ฝ่ายหนึ่งถามว่าผลงานนี้เกิดจากฟิสิกส์หรือไม่ ซึ่งคำตอบชัดเจนว่าใช่ อีกฝ่ายถามว่าผลงานนี้เป็นฟิสิกส์หรือไม่ ซึ่งเป็นคำถามเรื่องเส้นแบ่งของสาขา และคำถามแบบนั้นไม่มีการทดลองไหนตัดสินให้ได้ สิ่งที่ตัดสินได้คือข้อนี้ — ถ้าไม่มีคนที่รู้จัก Ising model นั่งอยู่ในห้อง ปี 1982 จะไม่มีใครถามว่าจะขุดหลุมเขาเองได้ไหม

ลองเอง: ค้นคำอ้างอิงจริงของรางวัลโนเบลสาขาฟิสิกส์ปี 2024 จากเว็บไซต์ nobelprize.org แล้วอ่านสองบรรทัด สังเกตว่าคำว่า physics ไม่ปรากฏในคำอ้างอิงเลย ปรากฏแต่ชื่อหมวด นั่นคือข้อเท็จจริงที่ทั้งสองฝ่ายของข้อถกเถียงใช้ได้ทั้งคู่ และการที่หลักฐานขึ้นเดียวกันรับใช้สองฝ่ายได้ คือสัญญาว่าคำถามที่ถามกันอยู่ยังไม่ใช่คำถามที่ตัดสินได้

บทนี้จะผิดเมื่อ: มีการพิสูจน์ว่ากฎการอัปเดตของ modern Hopfield network ในงานปี 2020 ไม่ได้เทียบเท่ากับกลไก attention ของ Transformer หรือมีหลักฐานว่าความจุของ dense associative memory ไม่ได้เพิ่มขึ้นตามที่งานปี 2016 ระบุ

11 · กำแพงย้ายที่ การประหยัดก็ย้ายตาม

การประหยัดกลับมาแล้ว และมันไม่ได้กลับมาเป็นของเดิม

คำถามที่พาเรามาถึงบทนี้คือ ของที่เราทิ้งไปตอนฮาร์ดแวร์ดีขึ้น จะฟื้นขึ้นมาอีกไหม คำตอบสั้นคือฟื้นแล้ว ตั้งแต่ราวปี 2023 และเหตุผลที่มันฟื้นไม่ใช่เพราะใครนึกถึงคุณค่าของความประหยัดขึ้นมาได้ มันฟื้นเพราะกำแพงย้ายที่

บทที่ 9 ปิดด้วยคำถามว่ากำแพงถดไปอยู่ตรงไหน คำตอบมีสามข้อ และข้อที่คนคาดถึงน้อยที่สุดคือข้อที่ปีบนหนักที่สุด

กำแพงที่หนึ่ง — จำนวนการคำนวณ นี่คือน้ำหนักที่ทุกคนคิดถึงเวลาพูดว่าเครื่องไม่แรงพอ มันเคยเป็นกำแพงจริงในยุค 2012 และมันคลายลงเยอะหลังจากนั้น

กำแพงที่สอง — การขนตัวเลขไปกลับ นี่คือน้ำหนักที่ปีบจริงในวันนี้ และคนนอกวงการแทบไม่เคยได้ยินถึง

ปัญหาคือในชิปสมัยใหม่ หน่วยคำนวณอยู่ที่หนึ่ง หน่วยความจำอยู่อีกที่หนึ่ง และระหว่างสองที่นั่นมีท่อ ตัวเลขทุกตัวที่จะถูกคุณต้องเดินทางผ่านท่อนั้น ของที่เกิดขึ้นคือหน่วยคำนวณเร็วขึ้นเร็วกว่าท่อกว้างขึ้นมาก จนถึงจุดที่หน่วยคำนวณนั่งรอของเป็นส่วนใหญ่ของเวลา

นี่คือเหตุผลที่วิธีชื่อ FlashAttention ที่ออกมาในปี 2022 ถึงดังมาก มันไม่ได้ลดจำนวนการคูณลงเลยสักครั้งเดียว คำตอบที่ได้เหมือนเดิมเป๊ะ สิ่งที่มีมันทำคือจัดลำดับการทำงานใหม่ให้ข้อมูลเดินทางผ่านท่อน้อยลง มันคือการประหยัดที่ไม่เกี่ยวกับคณิตศาสตร์เลย มันเกี่ยวกับการเดินของ

กำแพงที่สาม — ไฟฟ้า ศูนย์ข้อมูลสมัยนี้ถูกจำกัดด้วยกำลังไฟที่ทำได้ในพื้นที่ ไม่ใช่ด้วยเงินที่ซื้อชิปได้ และไฟฟ้าเป็นกำแพงชนิดที่ซื้อทางออกไม่ได้ด้วยความเร็ว

แล้วมีเรื่องที่ไม่ใช่เทคนิคแต่หนักที่สุดข้อหนึ่ง — **สอนครั้งเดียว ใช้ตลอดไป**

การสอนโมเดลหนึ่งตัวเกิดขึ้นครั้งเดียว ส่วนการเรียกใช้เกิดขึ้นทุกครั้งที่มีคนพิมพ์อะไรสักอย่าง วันละหลายร้อยล้านครั้ง ในสมการนั้น การทำให้เร็วขึ้นสองเท่าไม่ได้ประหยัดแค่สองเท่าของงานหนึ่งชิ้น มันประหยัดสองเท่าของทุกชิ้นตลอดไป ซึ่งเป็นเหตุผลทางธุรกิจล้วน ๆ ที่ทำให้งานบีบของกลับมาเป็นงานที่มีคนเก่ง ๆ อยากทำ นี่คือของที่กลับมา และชื่อของมันไม่มีคำว่า RBM อยู่เลย

quantization — ลดความละเอียดของตัวเลข เดิมน้ำหนักแต่ละตัวในโมเดลเก็บเป็นทศนิยมความละเอียดสูง ถ้าวัดลงเหลือความละเอียดต่ำ ของทั้งก้อนเล็กลง เดินผ่านท่อได้เร็วขึ้น ของที่สดใต่งที่สุดในสายนี้คือ BitNet b1.58 ของ Microsoft ซึ่งบังคับให้น้ำหนักทุกตัวเป็นได้แค่สามค่า คือ ลบหนึ่ง ศูนย์ หรือ หนึ่ง ผลที่รายงานไว้คือประหยัดหน่วยความจำได้เกินเก้าสิบเปอร์เซ็นต์เทียบกับความละเอียดมาตรฐาน และไลบรารีของมันวัดความเร็วบนชิพ ARM ได้ 1.37 ถึง 5.07 เท่า พร้อมลดการใช้พลังงาน 55.4 ถึง 70.0 เปอร์เซ็นต์ โมเดลทางการตัวแรกขนาดสองพันล้านพารามิเตอร์ออกเมื่อเมษายน 2025

สังเกตว่าของนี้ไม่ได้ฉลาดขึ้น มันหยาบลง และมันเร็วขึ้นเพราะมันหยาบลง

distillation — ให้โมเดลเล็กเรียนจากโมเดลใหญ่ แทนที่จะสอนโมเดลเล็กด้วยข้อมูลดิบ สอนมันด้วยคำตอบของโมเดลใหญ่ ซึ่งมีข้อมูลมากกว่าคำตอบที่ถูกต้องอย่างเดียว เพราะมันบอกด้วยว่าคำตอบอื่นใกล้เคียงแค่ไหน

Mixture of Experts — เปิดใช้แค่บางส่วน โมเดลมีพารามิเตอร์จำนวนมาก แต่ทุกครั้งที่ประมวลผลคำหนึ่งคำ มีแค่ส่วนน้อยที่ถูกเรียกใช้ ที่เหลือนอนเฉย ๆ ของทั้งก้อนใหญ่ แต่ค่าใช้จ่ายต่อครั้งเท่าโมเดลเล็ก

speculative decoding — ให้ตัวเล็กเดาก่อน โมเดลเล็กเดาคำถัดไปห้าคำรวดเดียว โมเดลใหญ่ตรวจทั้งห้าคำพร้อมกันในรอบเดียว เดาก็ได้ห้าคำในราคาหนึ่งรอบ เดาก็ทิ้ง

LoRA — ปรับเฉพาะส่วนเล็ก ๆ เวลาจะปรับโมเดลใหญ่ให้เข้ากับงานเฉพาะทาง ไม่ต้องขยับน้ำหนักทุกตัว ขยับชุดเล็ก ๆ ที่แปะเพิ่มเข้าไปพอ

ทั้งห้าอย่างนี้คือสิ่งเดียวกันกับที่คนเขียนเกมบนตลับ NES ทำ — หาว่าอะไรที่ทิ้งได้ โดยที่ผลลัพธ์ยังใช้ได้ ต่างกันที่ของที่ขาดแคลน สมัยนั้นขาดพื้นที่เก็บ สมัยนี้ขาด ความกว้างของท่อกับไฟฟ้า

ทีนี้ส่วนที่ต้องตอบตรง ๆ เพราะเป็นคำถามที่ถามมา — **RBM จะกลับมาไหม**

ในฐานะวิธีสอน คำตอบคือไม่น่า และเหตุผลอยู่ในบทที่ 8 แล้ว งานที่ RBM เคยทำคือ พาเครือข่ายลึกลับข้ามช่วงที่สอนไม่ได้ งานนั้นเสร็จแล้ว และวิธีที่มาแทนก็ดีกว่าในทุก ด้านที่วัดได้ ของที่หายไปเพราะทำงานเสร็จ ไม่กลับมาเพราะคนคิดถึงมัน มันจะกลับมา ก็ต่อเมื่อมีงานใหม่ที่หน้าตาเหมือนงานเดิม

แต่ถ้าถามถึงความคิดที่อยู่ข้างใน RBM คำตอบกลับข้างทันที

ความคิดนั้นคือการมองข้อมูลเป็นภูมิประเทศพลังงาน แล้วให้ระบบกลิ้งลงหาที่ต่ำ และความคิดนั้นไม่เคยหายไปเลยสักปี มันคือสิ่งที่อยู่ใน attention ตามบทที่แล้ว และมันคือสิ่งที่อยู่ในโมเดลสร้างภาพรุ่นปัจจุบัน ซึ่งทำงานด้วยการเริ่มจากภาพที่เป็น จตุรบกวนล้วน ๆ แล้วค่อย ๆ ไล่มันลงไปหาที่ต่ำจนกลายเป็นภาพ — ซึ่งคือการอบ แบบย้อนกลับของบทที่ 7 ทำซ้ำเป็นร้อยชั้น

เครื่องมือตาย ฟิสิกส์ไม่ตาย

และนั่นคือรูปแบบที่ใช้ได้กับคำถามว่าของเก่าจะกลับมาไหมเกือบทุกครั้ง — ถามว่า ของนั้นเคยแก้ปัญหาอะไร ถ้าปัญหายังอยู่ ของมีสิทธิ์กลับ ถ้าปัญหาหายไป ของก็ไปกับมัน และถ้าสิ่งที่ยังอยู่คือกฎที่อยู่ใต้ของชิ้นนั้น กฎนั้นจะโผล่ขึ้นมาอีกในชื่ออื่น ในที่ ที่ไม่มีใครมองหา

เหมือนที่สมการของกอนแม่เหล็กโผล่ขึ้นมาอีกครั้งในกล่องข้อความของโปรแกรมแชทที่คนทั้งโลกใช้กันอยู่ทุกวันนี้

ลองเอง: เปิดรูปถ่ายในเครื่องสกรูป แล้วส่งส่งต่อให้คนอื่น — เครื่องมักถามว่าจะส่ง ขนาดไหน เล็ก กลาง หรือเต็ม ลองเลือกขนาดเล็กแล้วดูตัวเลขขนาดไฟล์ที่มันบอก จะ เหลือไม่กี่เปอร์เซ็นต์ของเดิม จากนั้นเปิดรูปเล็กนั้นดูเต็มจอ ถ้าไม่ได้เอาไปขยายพิมพ์ จะแทบแยกไม่ออกจากตัวเต็ม นั่นคือ quantization ทั้งดุ้น — โยนความละเอียดที่ตา

ไม่ได้ใช้ทิ้งไป แล้วเก็บของที่เหลือไว้ครบ โมเดลภาษาถูกบีบด้วยความคิดเดียวกันเป๊ะ ต่างกันแค่สิ่งที่ถูกโยนทิ้งคือทศนิยมท้าย ๆ ของตัวเลขข้างใน ไม่ใช่เม็ดสี

บทนี้จะผิดเมื่อ: มีการวัดที่แสดงว่าความกว้างของช่องทางหน่วยความจำไม่ใช่คอขวดหลักของการเรียกใช้โมเดลภาษาขนาดใหญ่ หรือมีผลที่แสดงว่าการเตรียมน้ำหนักแบบ RBM กลับมาให้ผลดีกว่าวิธีปัจจุบันในงานขนาดใหญ่

12 · เอาแม่เหล็กมาคิดจริง ๆ

เราลอกฟิสิกส์มาทำความคิด ตอนนี้กำลังเอาความคิดกลับไปใส่ฟิสิกส์

วงกลมนี้ปิดตัวเองในอีกสิบสองบทที่เพิ่งอ่านจบ และมันปิดในที่ที่ตรงไปตรงมาจนเกือบจะน่าขำ

ย้อนกลับไปดูว่าเราทำอะไรมา ก่อนเหล็กมีเข็มทิศจิวที่หาสถานะพลังงานต่ำสุดได้เอง โดยไม่มีใครสั่ง เราเอากฎข้อนั้นออกมา เขียนเป็นสมการ แล้วเอาสมการนั้นไปรันบนคอมพิวเตอร์ที่ทำงานด้วยหลักการคนละอย่างสิ้นเชิง — คอมพิวเตอร์ที่สลับสวิตซ์ที่ละตัวตามลำดับคำสั่ง

คำถามที่ตามมาจึงตรงไปตรงมา ถ้าก่อนเหล็กทำเรื่องนี้ได้เองอยู่แล้ว ทำไมเราต้องจำลองมัน

นี่คือความคิดที่อยู่เบื้องหลังของที่เรียกว่า **Ising machine** — ฮาร์ดแวร์ที่ไม่ได้คำนวณคำตอบทีละขั้น แต่ปล่อยให้ฟิสิกส์ของตัวเองไหลลงไปหาคำตอบ ตั้งปัญหาให้อยู่ในรูปของภูมิประเทศพลังงาน สร้างระบบจริงที่มีภูมิประเทศหน้าตาแบบนั้น แล้วปล่อยมือ ระบบจะกลิ้งลงกันหลุมเองตามธรรมชาติของมัน คำตอบคือที่ที่มันไปนอน

มีหลายแบบที่สร้างกันอยู่ บางแบบใช้แสง โดยให้พัลส์เลเซอร์ที่วิ่งวนในเส้นใยแก้วนำแสงทำหน้าที่เป็นเข็มทิศ บางแบบใช้วงจรอิเล็กทรอนิกส์ธรรมดาที่ออกแบบให้เลียนพฤติกรรมของสปิน บางแบบใช้ชิปดิจิทัลเฉพาะทางที่ยังคำนวณเหมือนเดิมแต่ออกแบบมาเพื่อโจทย์รูปนี้โดยเฉพาะ

และตรงนี้ต้องระวังมาก เพราะสาขานี้มีตัวเลขที่ถูกอ้างเกินจริงบ่อยกว่าค่าเฉลี่ย

สิ่งที่หลักฐานปัจจุบันบอกมีสามข้อ

ข้อแรก จากบทความทบทวนใน Nature Reviews Physics ปี 2022 — ฮาร์ดแวร์ Ising ที่ใช้เทคโนโลยีดิจิทัลแบบคลาสสิกทำผลงานได้ดีที่สุดในโจทย์มาตรฐานที่ใช้เทียบกัน ไม่ใช่แบบที่ใช้แสงหรือแบบที่อ้างความเป็นควอนตัม

ข้อสอง ผลขึ้นกับรูปของโจทย์เป็นอย่างมาก ไม่มีเครื่องไหนชนะทุกโจทย์ เครื่องที่ชนะขาดในโจทย์รูปหนึ่งอาจแพ้ซอฟต์แวร์ธรรมดาในอีกรูปหนึ่ง ตัวเลขที่วัดได้ล่าสุดตัวหนึ่งคือ Digital Annealer รุ่นสองของ Fujitsu ทดสอบบนโจทย์ Max-Cut ที่มีตัวแปรถึง 53,000 ตัว โดยวัดเป็นเวลาจริงบนนาฬิกา ได้ผลดีกว่าวิธีที่ดีที่สุดที่มีอยู่ในราว 69.2 เปอร์เซ็นต์ของชุดที่ทดสอบ — ซึ่งแปลว่ามีอีกราวสามสิบเปอร์เซ็นต์ที่มันไม่ชนะ

ข้อสาม และข้อนี้สำคัญที่สุดสำหรับคนที่ได้ยินคำว่าควอนตัมมาพร้อมกัน — การอ้างว่าเครื่องหนึ่งเร็วกว่าอีกเครื่องเป็นเรื่องที่ถูกโต้แย้งกันอยู่จริง มีการแสดงว่าเครื่องแบบ coherent Ising machine ทำได้ดีกว่าเครื่องของ D-Wave ในโจทย์บางประเภท ซึ่งทีม D-Wave โต้แย้ง และที่ต้องพิมพ์ไว้คือ ในเครื่องแบบ coherent นั้น การเชื่อมต่อแสงเข้าด้วยกันผ่านการวัดกับการป้อนกลับทางอิเล็กทรอนิกส์ จึงไม่มีการพัวพันเชิงควอนตัมระหว่างพัลส์ ความเร็วที่วัดได้จากเครื่องแบบนี้ ไม่ใช่ความเร็วที่มาจากควอนตัม ไม่ว่าจะเขียนข่าวยังไงก็ตาม

ดังนั้นคำตอบที่ซื่อสัตย์ต่อคำถามว่า "ฮาร์ดแวร์แม่เหล็กจะเร็วกว่าชิปปัจจุบันหลายสิบเท่าไหม" คือ ในบางโจทย์ บางเครื่อง และยังไม่ใช้โจทย์ที่ AI ใช้อยู่ทุกวันนี้

สิ่งที่น่าสนใจกว่าตัวเลขคือรูปของเรื่อง เพราะเรื่องนี้กำลังเดินเข้าบทที่ 2 ทั้งบท

คนที่สร้าง Ising machine วันนี้อยู่ในตำแหน่งเดียวกับ Lilienthal พวกเขามีกฎที่ถูกอยู่ในมือแล้ว และกำลังหาว่าจะทำมันออกมาจากวัสดุอะไร ชั้นที่ยากไม่ใช่ความคิด ความคิดจบตั้งแต่ปี 1982 ชั้นที่ยากคือไนลอนของ de Mestral — วัสดุกับวิศวกรรมที่ทำให้ของที่ถูกในทฤษฎีใช้งานได้จริง แปดปีนั้นเขาไม่ได้ใช้คิด เขาใช้ทำ

ปิดท้ายด้วยสิ่งที่ยังลอกไม่ได้

สิบสองบทนี้เป็นเรื่องของการลอกที่สำเร็จ ทุกครั้งที่สำเร็จ สิ่งที่ลอกได้คือกฎที่มีคนเขียนออกมาเป็นตัวหนังสือได้แล้ว แรงยกเขียนเป็นสมการได้ พลังงานของสปริงเขียนเป็นสมการได้ ตะขอกับห่วงวาดรูปได้

สิ่งที่ยังลอกไม่ได้ในวันนี้ คือของที่ยังไม่มีใครเขียนกฎของมันออกมา

สมองเด็กอายุสามขวบเรียนภาษาจากตัวอย่างไม่กี่ล้านคำ โมเดลภาษาใช้หลายล้านล้านคำ ความต่างนี้ไม่ใช่เรื่องขนาด มันแปลว่ามีกลไกอะไรบางอย่างในนั้นที่เรายังไม่รู้จัก สมองทั้งก้อนกินไฟประมาณเท่าหลอดไฟดวงหนึ่ง ศูนย์ข้อมูลไม่ได้ใกล้เคียงเลย คนเรียนของใหม่ได้โดยไม่ลืมของเก่า ซึ่งเครือข่ายเทียมทำได้ยากมากจนมีชื่อเรียกอาการนี้โดยเฉพาะ

ของพวกนี้ไม่ได้ลอกไม่ได้เพราะมันลึกลับ มันลอกไม่ได้เพราะยังไม่มีใครส่งกล้องเจอตะขอ

และบทเรียนของเล่นนี้ก็ตรงไปตรงมาตามนั้น — เวลาเจอของในธรรมชาติที่ทำงานที่เราอยากทำได้ อย่างรับลอกสิ่งที่ตาเห็น เพราะสิ่งที่ตาเห็นคือปีกที่กระพือ เป็นนามสืเขียวกลม ๆ เป็นเซลล์ที่มีสารเคมีสืบกว่าชนิด

ของที่ต้องเอามาอยู่ลึกลงกว่านั้นหนึ่งชั้นเสมอ และมันมักจะน่าเบื่อว่ามากตอนเจอ

แรงยก ตะขอ ก้นหลุม

บทนี้จะผิดเมื่อ: มี Ising machine ที่แสดงความได้เปรียบเหนือฮาร์ดแวร์ทั่วไปอย่างสม่ำเสมอในโจทย์การเรียกใช้โมเดลภาษาขนาดใหญ่ ไม่ใช่เฉพาะโจทย์เชิงการจัดหมู่ หรือมีการวัดที่ล้นข้อสรุปของบทความทบทวนปี 2022 ว่าฮาร์ดแวร์ดิจิทัลแบบคลาสสิกทำผลงานดีที่สุดในโจทย์มาตรฐาน

แหล่ง และหมายเหตุความมั่นใจ

ทุกวันที่ ชื่อคน และตัวเลขในสิบสองบทข้างต้นมีที่มาอยู่ที่นี่ เนื้อเรื่องจึงเดินได้โดยไม่ต้องแบกเชิงอรรถ และคนที่อยากตรวจก็ตรวจได้ครบในที่เดียว ตรงไหนที่ยืนยันไม่ได้เขียนไว้ตรง ๆ ว่ายืนยันไม่ได้ พร้อมตัวเลขว่ามั่นใจแค่ไหน

บทที่ 1 · นาม

- George de Mestral, สิทธิบัตรสวิส ยื่น 1951 อนุมัติ 1955 · ชื่อ Velcro มาจาก velours + crochet
- **ความมั่นใจ 70%** สำหรับปีที่เดินป่า (แหล่งต่าง ๆ ระบุระหว่าง 1941 ถึง 1948 และเล่มนี้เขียนว่า "ต้นทศวรรษ 1940") · **ความมั่นใจ 60%** สำหรับรายละเอียดกระบวนการอินฟราเรดกับไนลอน ซึ่งมาจากการเล่าซ้ำมากกว่าเอกสารสิทธิบัตรโดยตรง

บทที่ 2 · ปีก

- Otto Lilienthal, Der Vogelflug als Grundlage der Fliegekunst, 1889
- การร่อนราว 2,000 เทียว 1891–1896 · อุบัติเหตุ 9 สิงหาคม 1896 เสียชีวิต 10 สิงหาคม
- Wright brothers, อุโมงค์ลม 1901 · เทียวบิน 17 ธันวาคม 1903 ที่ Kitty Hawk 12 วินาที 120 ฟุต
- **ความมั่นใจ 85%** สำหรับความสูงราว 15 เมตรของการตก

บทที่ 3 · นิวรอนบนกระดาด

- W. McCulloch & W. Pitts, A Logical Calculus of the Ideas Immanent in Nervous Activity, Bulletin of Mathematical Biophysics, 1943
- D. Hebb, The Organization of Behavior, 1949 · วลี "cells that fire together wire together" **ไม่ใช่คำของ Hebb** เป็นคำสรุปที่ตั้งขึ้นภายหลัง มักให้เครดิต Siegrid Löwel ราวปี 1992 — **ความมั่นใจ 80%** สำหรับที่มาของวลี **ความมั่นใจ 95%** ว่าไม่ใช่คำของ Hebb เอง
- F. Rosenblatt, perceptron, 1958 · Mark I Perceptron, Cornell Aeronautical Laboratory, กล้อง 400 จู๊ด
- M. Minsky & S. Papert, Perceptrons, 1969

บทที่ 4 · credit assignment

- D. Rumelhart, G. Hinton & R. Williams, Learning representations by back-propagating errors, Nature, 1986
- ผู้มาก่อน: S. Linnainmaa 1970 (วิทยานิพนธ์ปริญญาโท) · P. Werbos 1974 (วิทยานิพนธ์ปริญญาเอก)
- ข้ออ้างว่าสมองไม่ได้ใช้ backpropagation: **ความมั่นใจ 85%** — สถานะปัจจุบันคือยังไม่พบกลไกที่ทำหน้าที่เทียบเท่า ไม่ใช่การพิสูจน์ว่าไม่มี และข้อถกเถียงยังเปิดอยู่

บทที่ 5 · Ising model

- W. Lenz เสนอ 1920 · E. Ising แกกัณนิหนึ่งมิติ ตีพิมพ์ 1925 · L. Onsager แกกัณนิสองมิติ 1944

- ความมั่นใจ 80% สำหรับรายละเอียดว่า Ising ออกจากวงการฟิสิกส์ไปเป็นครูหลังจากนั้น

บทที่ 6 • Hopfield

- J. Hopfield, Neural networks and physical systems with emergent collective computational abilities, PNAS, 1982
- ความจุราว 0.14 เท่าของจำนวนนิวรอน: ตัวเลขนี้เป็นค่าประมาณมาตรฐานจากการวิเคราะห์เชิงสถิติ ไม่ใช่ค่าตายตัว ขึ้นกับนิยามของการเรียกคืนสำเร็จและความสัมพันธ์ระหว่างรูปแบบที่เก็บ — ความมั่นใจ 85%

บทที่ 7 • Boltzmann machine

- S. Kirkpatrick, C. Gelatt & M. Vecchi, Optimization by Simulated Annealing, Science, 1983
- G. Hinton & T. Sejnowski, Boltzmann machine, 1983 · A Learning Algorithm for Boltzmann Machines, Cognitive Science, 1985
- คำประกาศรางวัลโนเบล 2024 ระบุว่า Hinton "คิดค้นวิธีที่สามารถค้นพบคุณสมบัติในข้อมูลได้ด้วยตัวเอง" — [nobelprize.org](https://www.nobelprize.org/press-releases/2024/physics) · [press release](#)

บทที่ 8 • RBM และบันได

- G. Hinton, contrastive divergence, 2002
- G. Hinton, S. Osindero & Y.-W. Teh, A fast learning algorithm for deep belief nets, Neural Computation, 2006
- G. Hinton & R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science, 2006
- คำประกาศรางวัลโนเบล 2024 อ้างถึงงานปี 2006 ที่ใช้ Boltzmann machine ซ้อนกันเป็นชั้น ร่วมกับ Osindero, Teh และ Salakhutdinov — [nobelprize.org](https://www.nobelprize.org/popular-information) · [popular information](#)
- A. Krizhevsky, I. Sutskever & G. Hinton, AlexNet, ImageNet 2012 · สอนบน GPU สองใบ ไม่ใช่ pretraining
- ข้อสรุปว่าการเตรียมน้ำหนักแบบไม่มีเฉลยไม่จำเป็นสำหรับเครือข่ายลึก เมื่อมีการตั้งค่าเริ่มต้นที่เหมาะสมกับ ReLU: สรุปจากงานทบทวนหลายชิ้น โดยเฉพาะ [arXiv:2107.12521](https://arxiv.org/abs/2107.12521) · [RBM and DBN tutorial and survey](#)

บทที่ 9 • ขนาดของเกม

- Super Mario Bros (NES, 1985) 40 กิโลไบต์ = PRG-ROM 32 KB + CHR-ROM 8 KB — ความมั่นใจ 95%
- NES มี work RAM 2 กิโลไบต์ — ความมั่นใจ 90%

- ไม่มีตัวเลขที่ตรวจแล้ว สำหรับสัดส่วนโค้ดต่อ asset ในเกมสมัยใหม่ เล่นนี้อ้างเฉพาะกลไกว่าทำไมสองฝั่งโคคนละอัตรา ไม่อ้างตัวเลข

บทที่ 10 · modern Hopfield และ attention

- D. Krotov & J. Hopfield, Dense Associative Memory for Pattern Recognition, NeurIPS, 2016
- A. Vaswani และคณะ, Attention Is All You Need, 2017
- H. Ramsauer และคณะ, Hopfield Networks is All You Need, 2020 — กฎการอัปเดตใหม่ คือ กลไก attention ของ Transformer · จุดคงที่สามชนิด (เฉลี่ยรวม · metastable เฉลี่ยกลุ่มย่อย · รูปแบบเดี่ยว) · ขั้นตอนเฉลี่ยรวม ขั้นสูงเฉลี่ยกลุ่มย่อย — [semanticscholar](#)
- คำอ้างอิงรางวัลโนเบลสาขาฟิสิกส์ 2024: "for foundational discoveries and inventions that enable machine learning with artificial neural networks" — [nobelprize.org · summary](#)

บทที่ 11 · กำแพงที่ย้ายที่

- T. Dao และคณะ, FlashAttention, 2022 — วิธีที่ลดการเคลื่อนย้ายข้อมูล ไม่ลดจำนวนการคำนวณ ผลลัพธ์ตรงกับวิธีเดิมทุกประการ
- BitNet b1.58 — น้ำหนักสามค่า $\{-1, 0, +1\}$ · ประหยัดหน่วยความจำเกิน 90% เทียบ FP16 · [bitnet.cpp](#) วัดได้ 1.37–5.07 เท่าบนชิพ ARM พร้อมลดพลังงาน 55.4–70.0% · โมเดลทางการ 2B ออกเมษายน 2025 — [arXiv:2504.12285](#) · [arXiv:2410.16144](#) · [github.com/microsoft/BitNet](#)
- ตัวเลขประสิทธิภาพพลังงานระดับ 71.4 เท่าสำหรับการคูณเมทริกซ์ เป็นค่าประมาณจากแบบจำลองพลังงานบนกระบวนการ 7 นาโนเมตร ไม่ใช้การวัดบนชิปจริงทั้งระบบ — ความมั่นใจ 70% และไม่ควรอ่านเป็นความเร็วรวมของระบบ
- distillation · Mixture of Experts · speculative decoding · LoRA — เล่นน้อยอธิบายกลไก ไม่อ้างตัวเลขประสิทธิภาพของแต่ละวิธี

บทที่ 12 · Ising machine

- N. Mohseni, P. McMahon & T. Byrnes, Ising machines as hardware solvers of combinatorial optimization problems, Nature Reviews Physics, 2022 — ฮาร์ดแวร์ดิจิทัลแบบคลาสสิกทำงานได้ดีที่สุดในโจทย์มาตรฐาน · ผลขึ้นกับรูปของโจทย์ · ข้อโต้แย้งระหว่าง coherent Ising machine กับ D-Wave · ไม่มีการพัวพันเชิงควอนตัมใน CIM เพราะการเชื่อมทำการผ่านการวัดกับการป้อนกลับ — [nature.com/articles/s42254-022-00440-8](#)
- Fujitsu Digital Annealer v2 บน Max-Cut สูงสุด 53,000 ตัวแปร ดีกว่าวิธีคลาสสิกที่ดีที่สุดใน ~69.2% ของชุดที่ทดสอบ เมื่อวัดด้วยเวลาจริง — [arXiv:2507.22117](#)
- ข้ออ้างเรื่องพลังงานของสมองเทียบศูนย์ข้อมูล และเรื่องจำนวนค่าที่เด็กใช้เรียนภาษา: ไม่มีตัวเลขในเล่มนี้ เขียนไว้เป็นทิศทาง ไม่ใช้การวัด เพราะทั้งสองเรื่องมีค่าประมาณที่กระจายกว้างเกินกว่าจะพิมพ์เป็นตัวเลขเดียวได้อย่างมีศักดิ์

เล่มนี้เขียนขึ้นเดือนกันยายน 2026 ทุกบทมีบรรทัด "บทนี้จะผิดเมื่อ" ของตัวเอง
เพราะหนังสือที่บอกไม่ได้ว่าตัวเองจะผิดได้ยังไง คือหนังสือที่ผู้อ่านตรวจไม่ได้ ใครเจอ
ข้อผิดพลาด ของที่ถูกแก้คือตัวเล่ม ไม่ใช่ใบแก้คำผิดท้ายเล่ม