

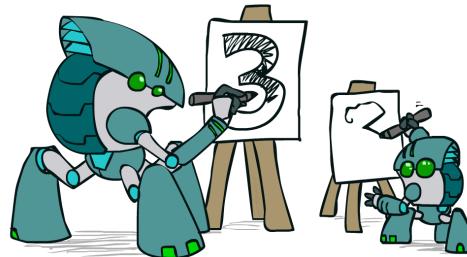
ปัญญาประดิษฐ์เบื้องต้น

บทบรรยายที่ 7: การเรียนรู้ของเครื่องและโครงข่ายประสาทเทียม

ผศ. ดร. อิทธิพล ฟองแก้ว
[ittipon@g.sut.ac.th]



เนื้อหาวันนี้



การเรียนรู้จากข้อมูลเป็นองค์ประกอบสำคัญของปัญญาประดิษฐ์ ในบทบรรยายนี้ เราจะแนะนำหลักการของ:

- การเรียนรู้ของเครื่อง (Machine learning)
- โครงข่ายประสาทเทียม (Neural networks)

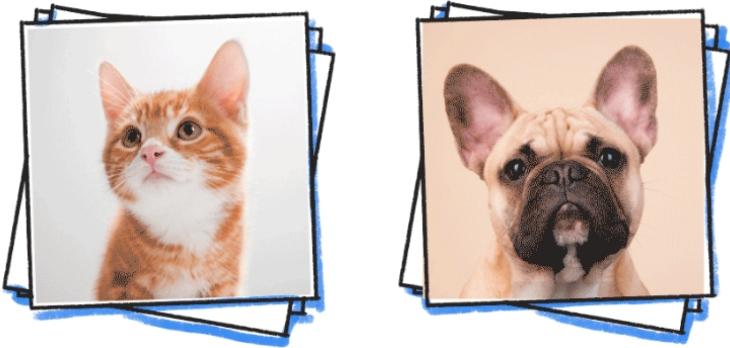
เอเจนต์ที่เรียนรู้ได้ (Learning agents)

จะเกิดอะไรขึ้นถ้าสภาพแวดล้อม **ไม่เป็นที่รู้จัก**?

- การเรียนรู้เป็นวิธีการอัตโนมัติในการปรับเปลี่ยนกลไกการตัดสินใจภายในของเอเจนต์เพื่อปรับปรุงประสิทธิภาพของตัวเอง
- เป็นการให้อเจนต์ได้สัมผัสถึงความเป็นจริงแทนที่จะพยายามเขียนโค้ดความเป็นจริงลงในโปรแกรมของเอเจนต์

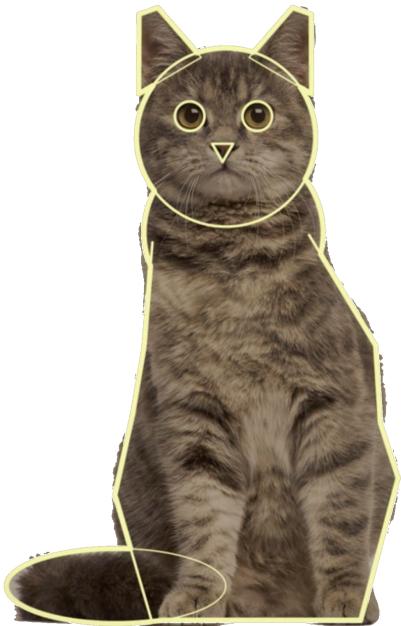
โดยทั่วไปแล้ว การเรียนรู้มีประโยชน์สำหรับงานใดๆ ที่ยากต่อการเขียนโปรแกรมเพื่อทำงานนั้น แต่สามารถหาตัวอย่างของพฤษิตกรรมที่ต้องการได้ง่าย

การเรียนรู้ของเครื่อง (Machine learning)



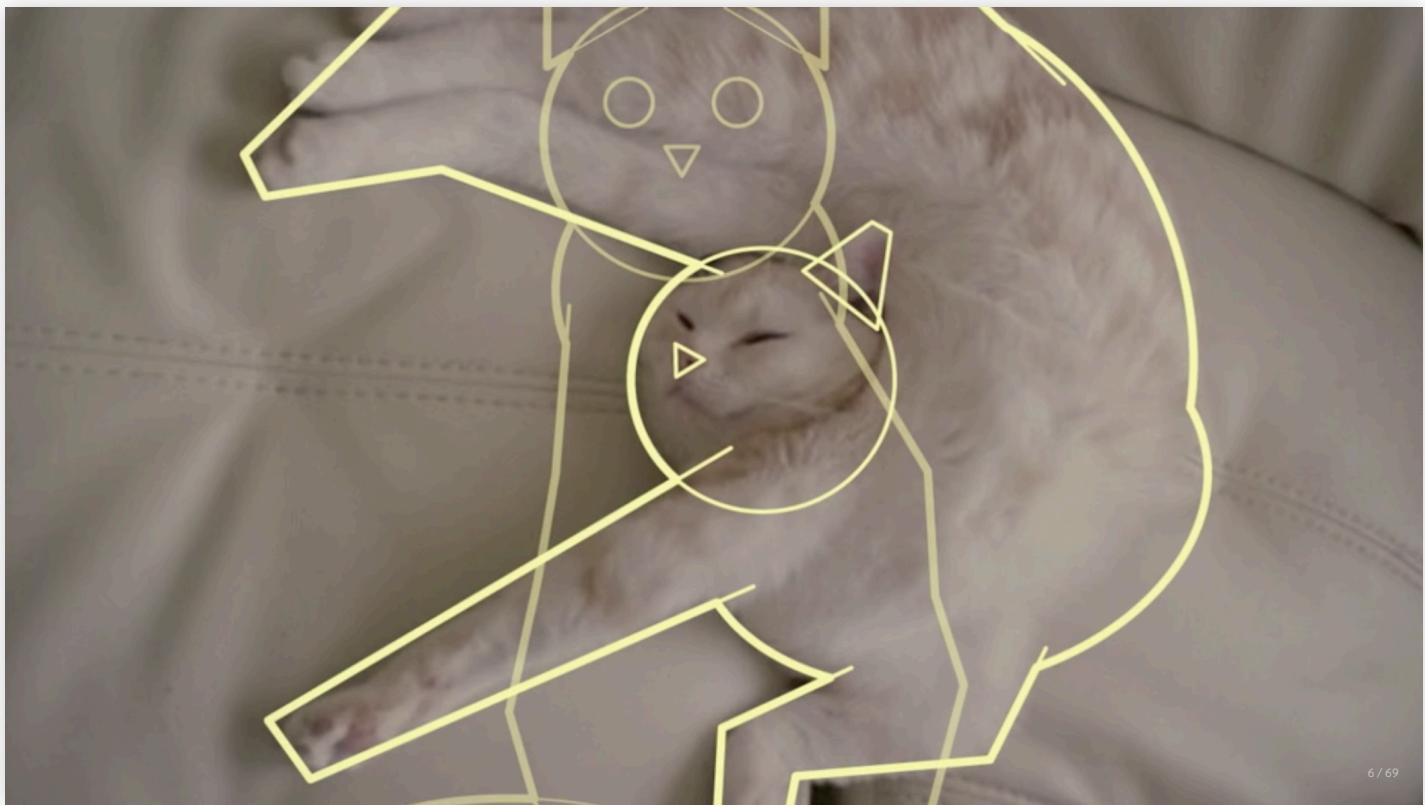
คุณจะเขียนโปรแกรมคอมพิวเตอร์ที่จำแนกแมวออกจากสุนัขได้อย่างไร?







6/69



นิยามปัญหา (Problem statement)

ให้ $\mathbf{d} \sim p(\mathbf{x}, y)$ เป็นชุดข้อมูลของคู่ตัวอย่างอินพุต-เอาต์พุต N คู่

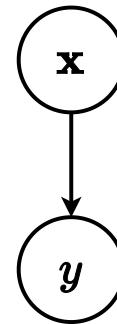
$$\mathbf{d} = (\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_N, y_N),$$

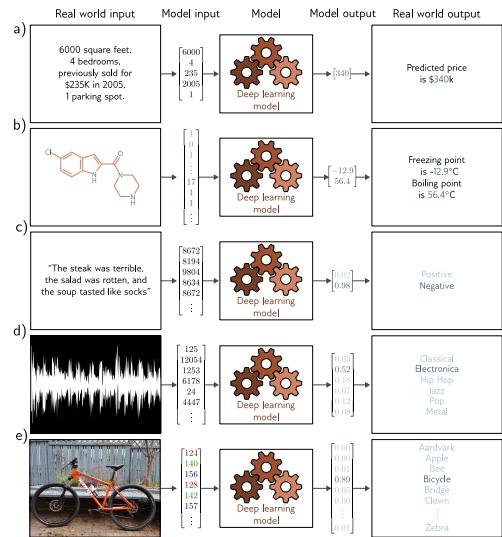
โดยที่ $\mathbf{x}_i \in \mathbb{R}^d$ คือเวกเตอร์ d มิติที่แสดงค่าอินพุต และ $y_i \in \mathcal{Y}$ คือค่าเอาต์พุตที่สอดคล้องกัน

จากข้อมูลนี้ เราต้องการระบุแบบจำลองความน่าจะเป็น

$$p_{\theta}(y|\mathbf{x})$$

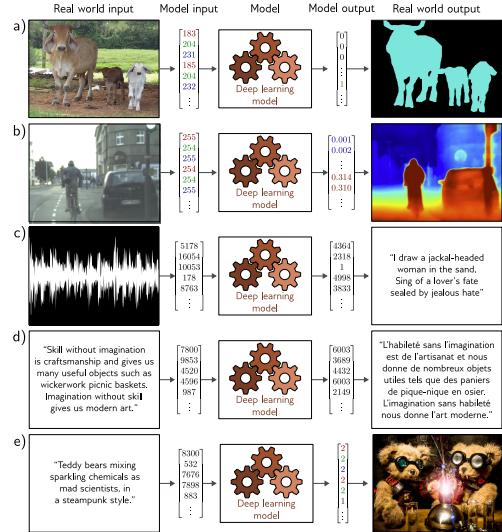
ที่อธิบายข้อมูลได้ดีที่สุด





ปัญหาการถดถอย (Regression) ($y \in \mathbb{R}$) และการจำแนกประเภท (classification) ($y \in \{0, 1, \dots, C-1\}$)

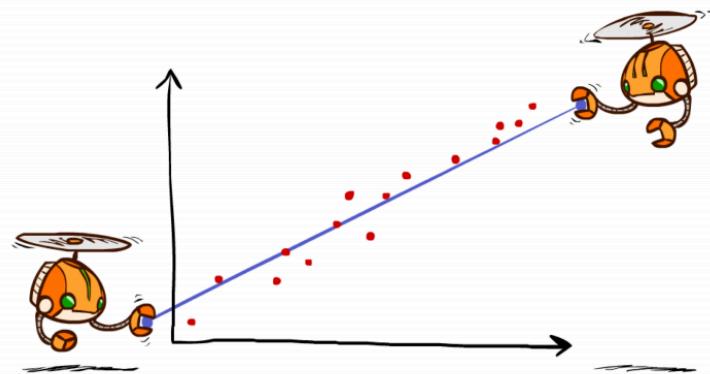
เครดิต: Simon J.D. Prince, 2023.

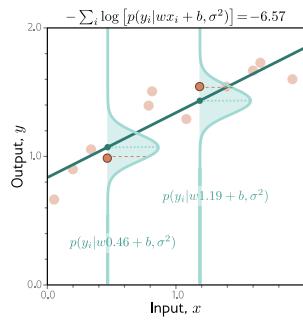
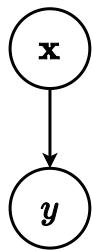


การเรียนรู้แบบมีผู้สอนพร้อมผลลัพธ์ที่มีโครงสร้าง (structured outputs) ($y \in \mathcal{Y}$)

Linear Regression

สมมติว่า $y \in \mathbb{R}$ ก่อน





Linear regression พิจารณาแบบจำลองเชิงเส้นแบบเกาส์เชียน (linear Gaussian model) ที่มีพารามิเตอร์สำหรับแบบจำลองพารามิเตอริกของ $p(y|x)$ นั่นคือ

$$p(y|x) = \mathcal{N}(y|\mathbf{w}^T \mathbf{x} + b, \sigma^2),$$

โดยที่ \mathbf{w} และ b เป็นพารามิเตอร์ที่ต้องหาค่า

เพื่อเรียนรู้การแจกแจงแบบมีเงื่อนไข $p(y|\mathbf{x})$ เราจะทำการหาค่าสูงสุดของ

$$p(y|\mathbf{x}) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2} \frac{(y - (\mathbf{w}^T \mathbf{x} + b))^2}{\sigma^2}\right)$$

เทียบกับ \mathbf{w} และ b บนข้อมูล $\mathbf{d} = (\mathbf{x}_j, y_j)$

เพื่อเรียนรู้การแจกแจงแบบมีเงื่อนไข $p(y|\mathbf{x})$ เราจะทำการหาค่าสูงสุดของ

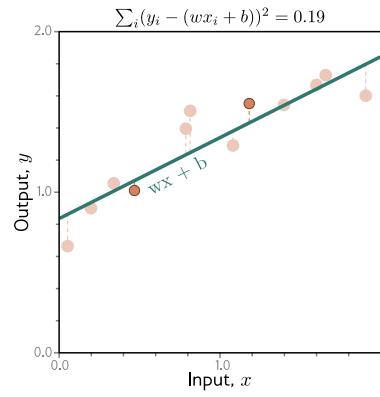
$$p(y|\mathbf{x}) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2} \frac{(y - (\mathbf{w}^T \mathbf{x} + b))^2}{\sigma^2}\right)$$

เทียบกับ \mathbf{w} และ b บนข้อมูล $\mathbf{d} = (\mathbf{x}_j, y_j)$

โดยการบังคับให้อันพันธ์ของ log-likelihood เป็น 0 เราจะไปถึงปัญหาของการหาค่าต่ำสุดของ

$$\mathcal{L}(\mathbf{w}, b) = \sum_{j=1}^N (y_j - (\mathbf{w}^T \mathbf{x}_j + b))^2.$$

ดังนั้น การหาค่าต่ำสุดของผลรวมของกำลังสองของความคลาดเคลื่อน (sum of squared errors) จะสอดคล้องกับการแก้ปัญหา MLE สำหรับการประมาณเชิงเส้น โดยสมมติว่ามีสัญญาณรบกวนแบบ gauss เนื่องที่มีความแปรปรวนคงที่



การหาค่าตัวสูดของ negative log-likelihood ของแบบจำลอง linear Gaussian model ลดรูปลงเป็นการหาค่าตัวสูดของผลรวมของกำลังสองของส่วนที่เหลือ (sum of squared residuals)

ถ้าเรารวมพจน์ใบแอก b เข้าไปในเวกเตอร์น้ำหนัก \mathbf{w} โดยการเพิ่มคุณลักษณะคงที่ $x_0 = 1$ เข้าไปในเวกเตอร์อินพุต \mathbf{x}

ผลเฉลย \mathbf{w}^* จะสามารถหาได้ในรูปแบบวิเคราะห์ (analytically) ดังนี้:

$$\mathbf{w}^* = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$$

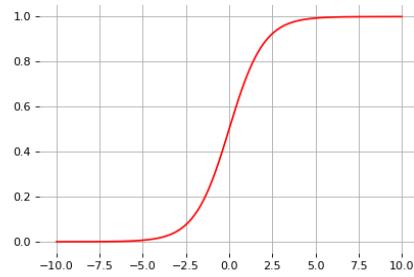
โดยที่:

- \mathbf{X} คือเมตริกซ์อินพุตที่ประกอบด้วยเวกเตอร์อินพุต \mathbf{x}_j (รวมถึงคุณลักษณะคงที่)
- \mathbf{y} คือเวกเตอร์เอาต์พุตที่ประกอบด้วยค่าเอาต์พุต y_j

Logistic Regression

ตอนนี้ให้เราสมมติว่า $y \in \{0, 1\}$





ฟังก์ชัน sigmoid $\sigma(z) = \frac{1}{1+e^{-z}}$ จะจับคู่ค่าใดๆ บนเส้นจำนวนจริงกับช่วง $(0, 1)$

A minimizer of the approximation $\hat{\mathcal{L}}(\epsilon; \theta_0)$ is given for

$$\begin{aligned}\nabla_{\epsilon} \hat{\mathcal{L}}(\epsilon; \theta_0) &= 0 \\ &= \nabla_{\theta} \mathcal{L}(\theta_0) + \frac{1}{\gamma} \epsilon,\end{aligned}$$

which results in the best improvement for the step $\epsilon = -\gamma \nabla_{\theta} \mathcal{L}(\theta_0)$.

Therefore, model parameters can be updated iteratively using the update rule

$$\theta_{t+1} = \theta_t - \gamma \nabla_{\theta} \mathcal{L}(\theta_t),$$

where

- θ_0 are the initial parameters of the model,
- γ is the learning rate.

ตามหลักการของ maximum likelihood estimation เราจะได้:

$$\arg \max_{\mathbf{w}, b} P(\mathbf{d} | \mathbf{w}, b)$$

$$\begin{aligned} &= \arg \max_{\mathbf{w}, b} \prod_{i=1}^n P(Y=y_i | \mathbf{x}_i, \mathbf{w}, b) \\ &= \arg \max_{\mathbf{w}, b} \prod_{i=1}^n \sigma(\mathbf{w}^T \mathbf{x}_i + b)^{y_i} (1 - \sigma(\mathbf{w}^T \mathbf{x}_i + b))^{1-y_i} \end{aligned}$$

การหา log และเปลี่ยนจาก max เป็น min ของ negative log-likelihood:

$$= \arg \min_{\mathbf{w}, b} \sum_{i=1}^n \ell(y_i, \hat{y}(\mathbf{x}_i; \mathbf{w}, b))$$

โดยที่:

$$\ell_i = -y_i \log \sigma(\mathbf{w}^T \mathbf{x}_i + b) - (1 - y_i) \log(1 - \sigma(\mathbf{w}^T \mathbf{x}_i + b))$$

สมการนี้คือฟังก์ชัน loss $\mathcal{L}(\mathbf{w}, b) = \sum_i \ell(y_i, \hat{y}(\mathbf{x}_i; \mathbf{w}, b))$

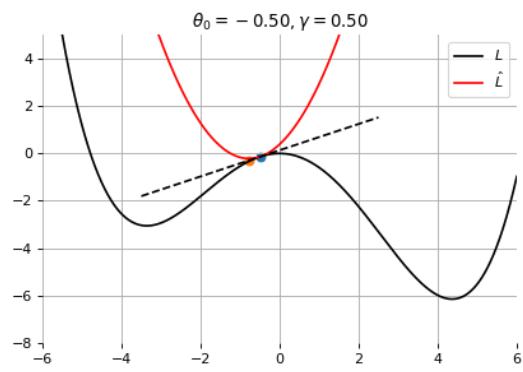
Loss นี้เป็นตัวประมาณค่าของ cross-entropy:

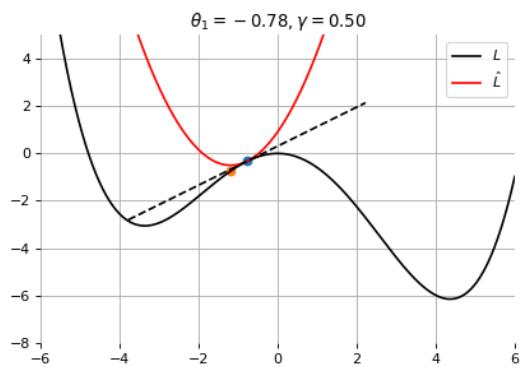
$$H(p, q) = \mathbb{E}_p[-\log q]$$

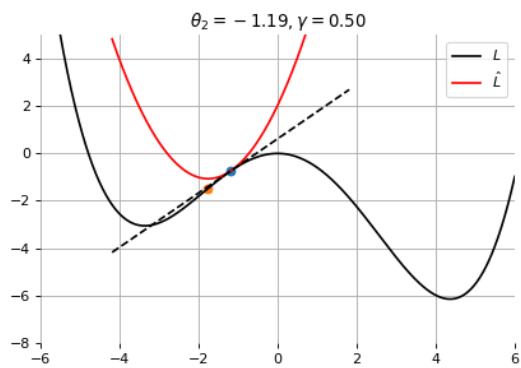
สำหรับ $p = Y | \mathbf{x}_i$ และ $q = \hat{Y} | \mathbf{x}_i$

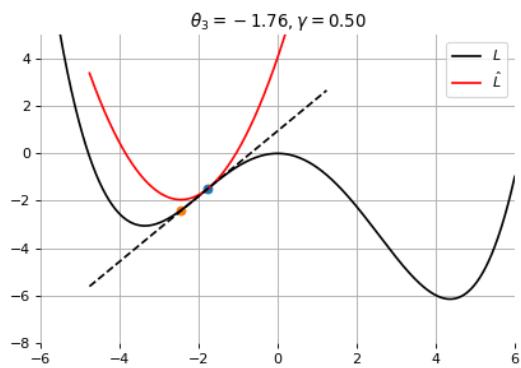
นำเสียดายที่ไม่มีผลเฉลยในรูปแบบปิด (closed-form solution) สำหรับ MLE ของ \mathbf{w} และ b

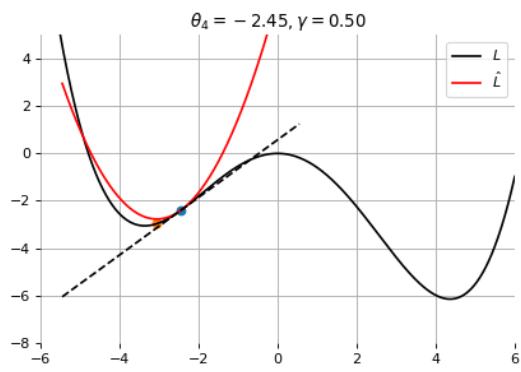
Gradient Descent

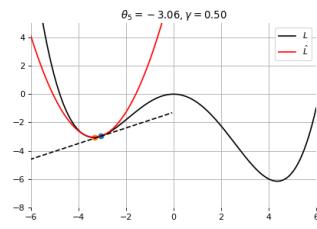


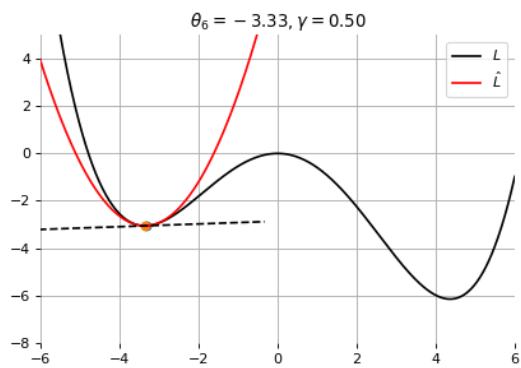


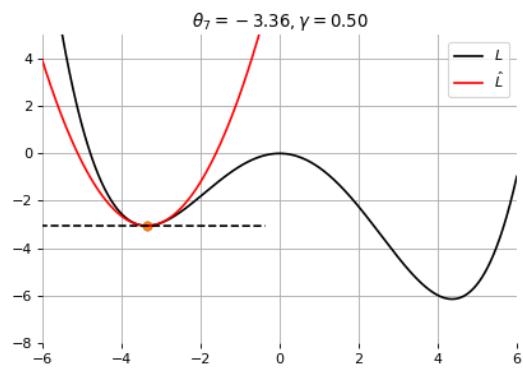


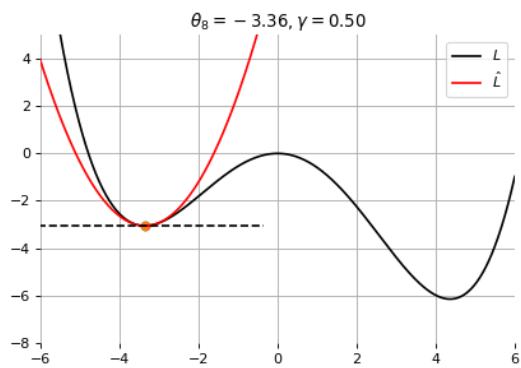


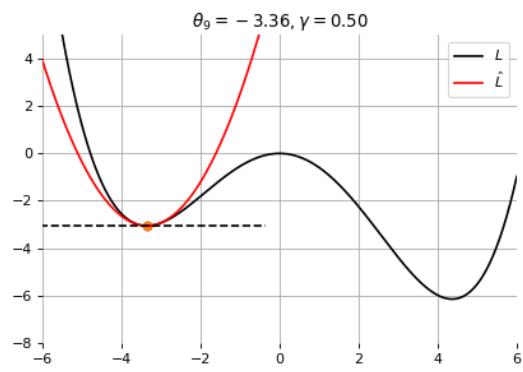








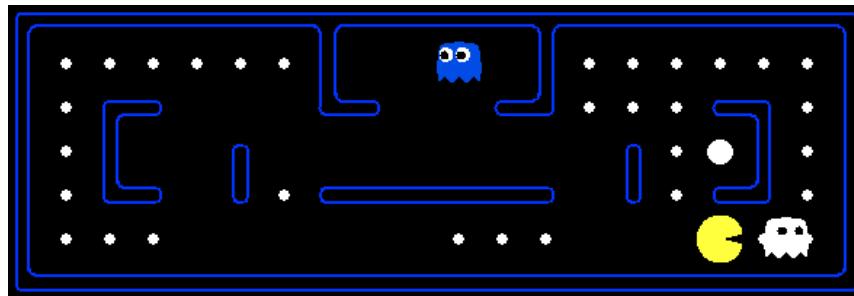


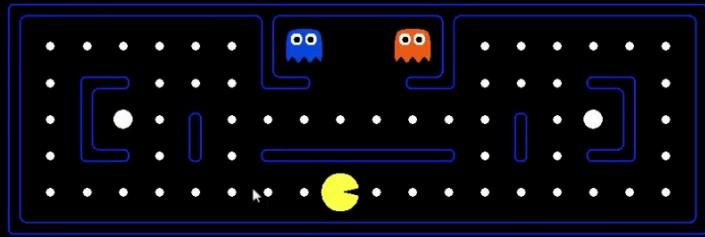


Example: imitation learning in Pacman

Can we learn to play Pacman only from observations?

- Feature vectors $\mathbf{x} = g(s)$ are extracted from the game states s . Output values y corresponds to actions a .
- State-action pairs (\mathbf{x}, y) are collected by observing an expert playing.
- We want to learn the actions that the expert would take in a given situation. That is, learn the mapping $f : \mathbb{R}^d \rightarrow \mathcal{A}$.
- This is a multiclass classification problem that can be solved by combining binary classifiers.

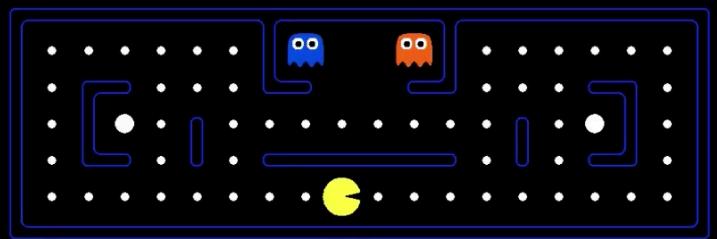




SCORE: 0

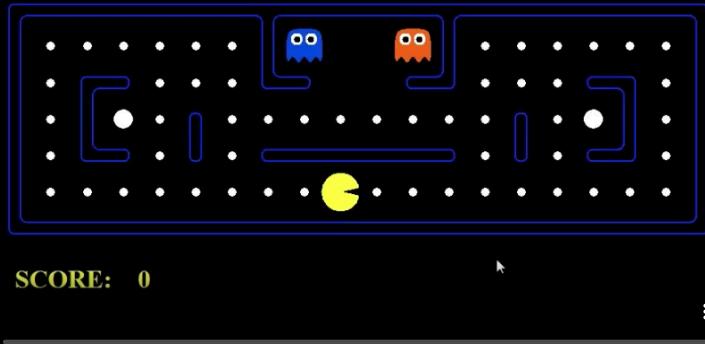
:

The agent observes a very good Minimax-based agent for two games and updates its weight vectors as data are collected.



SCORE: 0

⋮



After two training episodes, the ML-based agents plays.
No more Minimax!

สำหรับ θ_0 การประมาณอันดับหนึ่งรอบ θ_0 สามารถกำหนดได้เป็น

$$\hat{\mathcal{L}}(\epsilon; \theta_0) = \mathcal{L}(\theta_0) + \epsilon^T \nabla_{\theta} \mathcal{L}(\theta_0) + \frac{1}{2\gamma} \|\epsilon\|^2.$$

ตัวหาค่าต่ำสุดของการประมาณ $\hat{\mathcal{L}}(\epsilon; \theta_0)$ จะได้เมื่อ $\begin{aligned} \nabla_{\theta_0} \hat{\mathcal{L}}(\epsilon; \theta_0) &= 0 \\ \nabla_{\theta_0} \hat{\mathcal{L}}(\epsilon; \theta_0) &= \nabla_{\theta_0} (\nabla_{\theta_0} \hat{\mathcal{L}}(\epsilon; \theta_0)) + \frac{1}{\gamma} \nabla_{\theta_0} \hat{\mathcal{L}}(\epsilon; \theta_0) \end{aligned}$ ซึ่งส่งผลให้เกิดการปรับปรุงที่ดีที่สุดสำหรับขั้นตอน $\epsilon = -\gamma \nabla_{\theta_0} \hat{\mathcal{L}}(\epsilon; \theta_0)$

ดังนั้น พารามิเตอร์ของโมเดลสามารถอัปเดตข้าๆ ได้โดยใช้กฎการอัปเดต $\theta_{t+1} = \theta_t - \gamma \nabla_{\theta_t} \hat{\mathcal{L}}(\epsilon; \theta_t)$, โดยที่

- θ_0 คือพารามิเตอร์เริ่มต้นของโมเดล,
- γ คืออัตราการเรียนรู้ (learning rate)

โครงข่ายประสาทเทียม (Neural networks)

Shallow networks

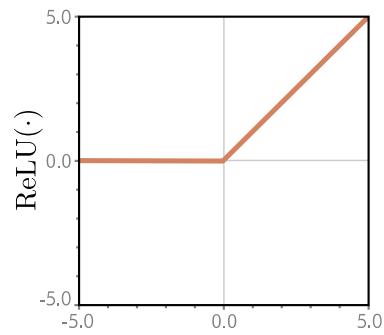
A shallow network is a function

$$f : \mathbb{R}^{d_{\text{in}}} \rightarrow \mathbb{R}^{d_{\text{out}}}$$

that maps multi-dimensional inputs \mathbf{x} to multi-dimensional outputs \mathbf{y} through a hidden layer $\mathbf{h} = [h_0, h_1, \dots, h_{q-1}] \in \mathbb{R}^q$, such that

$$\begin{aligned} h_j &= \sigma \left(\sum_{i=0}^{d_{\text{in}}-1} w_{ji} x_i + b_j \right) \\ y_k &= \sum_{j=0}^{q-1} v_{kj} h_j + c_k, \end{aligned}$$

where w_{ji} , b_j , v_{kj} and c_k ($i = 0, \dots, d_{\text{in}} - 1$, $j = 0, \dots, q - 1$, $k = 0, \dots, d_{\text{out}} - 1$) are the model parameters and σ is an activation function.



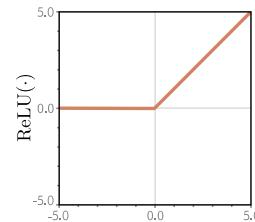
ໂຄຮງໝາຍແບບອືນພູຕເດີຍວ-ເລາຕົ້ມພູຕເດີຍ

Single-input single-output networks

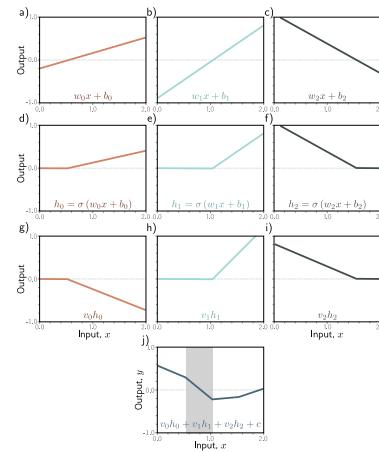
We first consider the case where $d_{\text{in}} = 1$ and $d_{\text{out}} = 1$ for the single-input single-output network

$$y = v_0\sigma(w_0x + b_0) + v_1\sigma(w_1x + b_1) + v_2\sigma(w_2x + b_2) + c$$

where $w_0, w_1, w_2, b_0, b_1, b_2, v_0, v_1, v_2$ and c are the model parameters and where the activation function σ is $\text{ReLU}(\cdot) = \max(0, \cdot)$.



โครงข่ายนี้กำหนดตระกูลของฟังก์ชันเชิงเส้นเป็นช่วง (piecewise linear functions) ซึ่งตำแหน่งของข้อต่อ, ความชัน และความสูงของฟังก์ชันถูกกำหนดโดยพารามิเตอร์ 10 ตัว $w_0, w_1, w_2, b_0, b_1, b_2, v_0, v_1, v_2$ และ c



จำนวน q ของหน่วยช่อง h_j เป็นตัววัด ความจุ (*capacity*) ของโครงข่ายตื้น ด้วยฟังก์ชันกระตุ้น ReLU หน่วยช่องจะกำหนดข้อต่อได้ถึง q ข้อในปริภูมิอินพุต ซึ่งจะกำหนดขอบเขตเชิงเส้น $q + 1$ ขอบเขตในปริภูมิเอ้าท์พุต

ເອາະພຸດຫລາຍຕັວແປຣ (Multivariate outputs)

ເພື່ອຂໍາຍໂຄຮງຂ່າຍໄປຢັງເອາະພຸດຫລາຍຕັວແປຣ $\mathbf{y} = [y_0, y_1, \dots, y_{d_{\text{out}}-1}]$ ເຮັດວຽກແດ່ເພີ່ມໜ່ວຍເອາະພຸດມາກີ່ນເປັນກຳນົດໃຈ

ຕົວຢ່າງເຊີ້ນ ໂຄຮງຂ່າຍທີ່ມີໜ່ວຍເອາະພຸດສອງໜ່ວຍ y_0 ແລະ y_1 ອາຈນີໂຄຮງສ້າງດັ່ງຕ່ອໄປນີ້:

$$h_0 = \sigma(w_0x + b_0)$$

$$h_1 = \sigma(w_1x + b_1)$$

$$h_2 = \sigma(w_2x + b_2)$$

$$h_3 = \sigma(w_3x + b_3)$$

$$y_0 = v_{00}h_0 + v_{01}h_1 + v_{02}h_2 + v_{03}h_3 + c_0$$

$$y_1 = v_{10}h_0 + v_{11}h_1 + v_{12}h_2 + v_{13}h_3 + c_1$$

อินพุตหลายตัวแปร (Multivariate inputs)

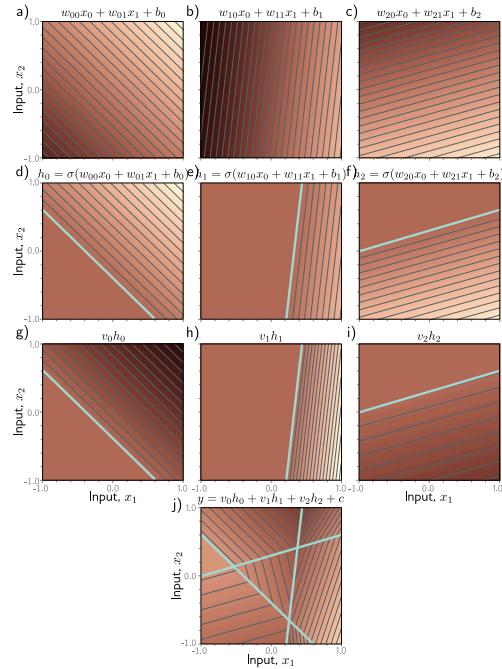
เพื่อขยายโครงข่ายไปยังอินพุตหลายตัวแปร $\mathbf{x} = [x_0, x_1, \dots, x_{d_{\text{in}}-1}]$ เราขยายความสัมพันธ์เชิงเส้นระหว่างอินพุตและหน่วยที่ซ่อนอยู่

ตัวอย่างเช่น โครงข่ายที่มีอินพุตสองตัว $\mathbf{x} = [x_0, x_1]$ อาจมีหน่วยที่ซ่อนอยู่สามหน่วย h_0, h_1 และ h_2 กำหนดเป็น

$$h_0 = \sigma(w_{00}x_0 + w_{01}x_1 + b_0)$$

$$h_1 = \sigma(w_{10}x_0 + w_{11}x_1 + b_1)$$

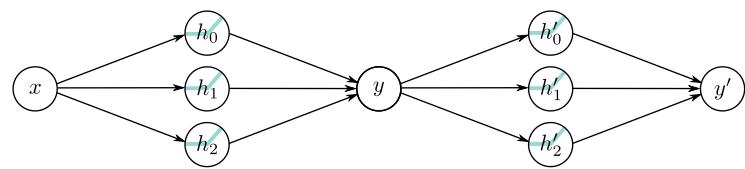
$$h_2 = \sigma(w_{20}x_0 + w_{21}x_1 + b_2).$$

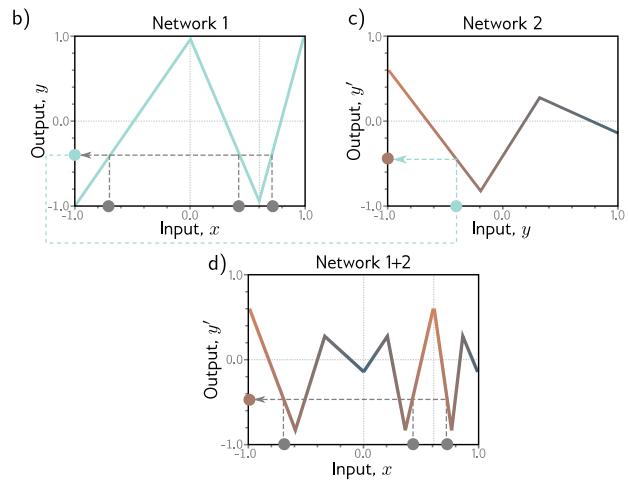


โครงข่ายลึก (Deep networks)

เราจะพิจารณาการประกอบกันของโครงข่ายตื้นสองโครงข่ายก่อน โดยที่เอาร์พุตของโครงข่ายแรกจะถูกป้อนเป็นอินพุตให้กับโครงข่ายที่สองดังนี้

$$\begin{aligned} h_0 &= \sigma(w_0x + b_0) \\ h_1 &= \sigma(w_1x + b_1) \\ h_2 &= \sigma(w_2x + b_2) \\ y &= v_0h_0 + v_1h_1 + v_2h_2 + c \\ h'_0 &= \sigma(w'_0y + b'_0) \\ h'_1 &= \sigma(w'_1y + b'_1) \\ h'_2 &= \sigma(w'_2y + b'_2) \\ y' &= v'_0h'_0 + v'_1h'_1 + v'_2h'_2 + c'. \end{aligned}$$





เมื่อจากการดำเนินการจาก $[h_0, h_1, h_2]$ ไปยัง y เป็นเชิงเส้น และการดำเนินการจาก y ไปยัง $[h'_0, h'_1, h'_2]$ ก็เป็นเชิงเส้นเช่นกัน การประกอบอนุกรมจึงเป็นเชิงเส้น

ดังนั้น การประกอบโครงข่ายต้นสองโครงข่ายจึงเป็นกรณีพิเศษของโครงข่ายลีกที่มีสองชั้นซ่อน โดยที่ชั้นแรกถูกกำหนดเป็น

$$h_0 = \sigma(w_0x + b_0)$$

$$h_1 = \sigma(w_1x + b_1)$$

$$h_2 = \sigma(w_2x + b_2),$$

ชั้นที่สองถูกกำหนดจากเอาต์พุตของชั้นแรกเป็น

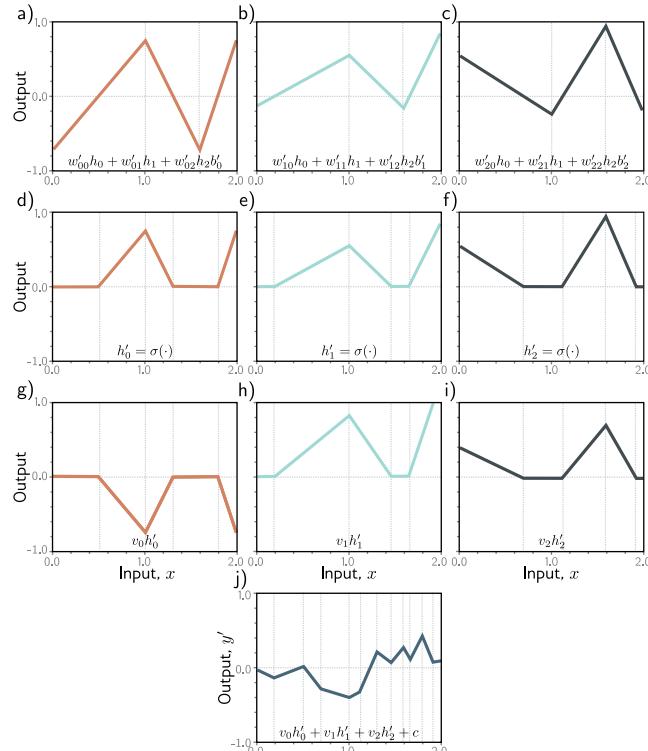
$$h'_0 = \sigma(w'_0h_0 + w'_1h_1 + w'_2h_2 + b'_0)$$

$$h'_1 = \sigma(w'_0h_0 + w'_1h_1 + w'_2h_2 + b'_1)$$

$$h'_2 = \sigma(w'_0h_0 + w'_1h_1 + w'_2h_2 + b'_2),$$

และเอาต์พุตถูกกำหนดเป็น

$$y = v_0h'_0 + v_1h'_1 + v_2h'_2 + c.$$



การคำนวณของชั้นช่องสามารถเขียนในรูปแบบเมทริกซ์ได้ดังนี้

$$\begin{aligned}\mathbf{h} &= \begin{bmatrix} h_0 \\ h_1 \\ \vdots \\ h_{q-1} \end{bmatrix} = \sigma \left(\begin{bmatrix} w_{00} & w_{01} & \cdots & w_{0(d_m-1)} \\ w_{10} & w_{11} & \cdots & w_{1(d_m-1)} \\ \vdots & \vdots & \ddots & \vdots \\ w_{(q-1)0} & w_{(q-1)1} & \cdots & w_{(q-1)(d_m-1)} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{d_m-1} \end{bmatrix} + \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_{q-1} \end{bmatrix} \right) \\ &= \sigma(\mathbf{W}^T \mathbf{x} + \mathbf{b})\end{aligned}$$

โดยที่ $\mathbf{x} \in \mathbb{R}^{d_m}$ คือเวกเตอร์อินพุต, $\mathbf{W} \in \mathbb{R}^{d_m \times q}$ คือเมทริกซ์น้ำหนักของชั้นช่อง และ $\mathbf{b} \in \mathbb{R}^q$ คือเวกเตอร์ไบแอส

ขั้นซ่อนสามารถประกอบกันเป็นอนุกรมเพื่อสร้างโครงข่ายลึกที่มี L ชั้นได้ดังนี้

$$\begin{aligned}\mathbf{h}_0 &= \mathbf{x} \\ \mathbf{h}_1 &= \sigma(\mathbf{W}_1^T \mathbf{h}_0 + \mathbf{b}_1) \\ \mathbf{h}_2 &= \sigma(\mathbf{W}_2^T \mathbf{h}_1 + \mathbf{b}_2) \\ &\vdots \\ \mathbf{h}_L &= \sigma(\mathbf{W}_L^T \mathbf{h}_{L-1} + \mathbf{b}_L) \\ \mathbf{y} &= \mathbf{h}_L,\end{aligned}$$

โดยที่ $\mathbf{W}_\ell \in \mathbb{R}^{q_{\ell-1} \times q_\ell}$ คือเมตริกซ์นำหน้าของชั้นที่ ℓ , $\mathbf{b}_\ell \in \mathbb{R}^{q_\ell}$ คือเวกเตอร์ไบเออสของชั้นที่ ℓ , และ $\mathbf{h}_\ell \in \mathbb{R}^{q_\ell}$ คือเวกเตอร์ซ่อนของชั้นที่ ℓ

- สำหรับการลดด้อย (regression) ความกว้าง q ของชั้นสุดท้าย L จะถูกตั้งค่าเป็นมิติของเอาต์พุต d_{out} และฟังก์ชันกระตุนคือ พังก์ชันเอกลักษณ์ $\sigma(\cdot) = \cdot$ ซึ่งส่งผลให้ได้เวกเตอร์ $\mathbf{h}_L \in \mathbb{R}^{d_{\text{out}}}$
- สำหรับการจำแนกประเภทแบบใบหนารี (binary classification) ความกว้าง q ของชั้นสุดท้าย L จะถูกตั้งค่าเป็น 1 และ พังก์ชันกระตุนคือ sigmoid $\sigma(\cdot) = \frac{1}{1+\exp(-\cdot)}$ ซึ่งส่งผลให้ได้อาต์พุตเดียว $h_L \in [0, 1]$ ที่สร้างแบบจำลองความน่าจะเป็น $p(y = 1|\mathbf{x})$
- สำหรับการจำแนกประเภทหลายคลาส (multi-class classification) ฟังก์ชันกระตุน sigmoid σ ในชั้นสุดท้ายสามารถทำให้ เป็นกรณีที่ว่าไปเพื่อสร้างเวกเตอร์ $\mathbf{h}_L \in \Delta^C$ ของการประมาณค่าความน่าจะเป็น $p(y = i|\mathbf{x})$ ฟังก์ชันกระตุนนี้คือฟังก์ชัน Softmax ซึ่งเอาต์พุตที่ i ของมันถูกกำหนดเป็น

$$\text{Softmax}(\mathbf{z})_i = \frac{\exp(z_i)}{\sum_{j=1}^C \exp(z_j)},$$

สำหรับ $i = 1, \dots, C$

พารามิเตอร์ (เช่น \mathbf{W}_ℓ และ \mathbf{b}_ℓ สำหรับแต่ละชั้น ℓ) ของโครงข่ายลึก $f(\mathbf{x}; \theta)$ จะถูกเรียนรู้โดยการหาค่าต่ำสุดของฟังก์ชันการสูญเสีย $\mathcal{L}(\theta)$ บนชุดข้อมูล $\mathbf{d} = (\mathbf{x}_j, \mathbf{y}_j)$ ของคู่อินพุต-เอาต์พุต

ฟังก์ชันการสูญเสียได้มาจากการ likelihood:

- สำหรับการถดถอย โดยสมมติว่าเป็น Gaussian likelihood การสูญเสียคือค่าเฉลี่ยของความคลาดเคลื่อนกำลังสอง (mean squared error) $\mathcal{L}(\theta) = \frac{1}{N} \sum_{(\mathbf{x}_j, \mathbf{y}_j) \in \mathbf{d}} (\mathbf{y}_j - f(\mathbf{x}_j; \theta))^2$
- สำหรับการจำแนกประเภท โดยสมมติว่าเป็น categorical likelihood การสูญเสียคือ cross-entropy $\mathcal{L}(\theta) = -\frac{1}{N} \sum_{(\mathbf{x}_j, \mathbf{y}_j) \in \mathbf{d}} \sum_{i=1}^C y_{ij} \log f_i(\mathbf{x}_j; \theta)$

MLPs on images?

The MLP architecture is appropriate for tabular data, but not for images.

- Each pixel of an image is an input feature, leading to a high-dimensional input vector.
- Each hidden unit is connected to all input units, leading to a high-dimensional weight matrix.

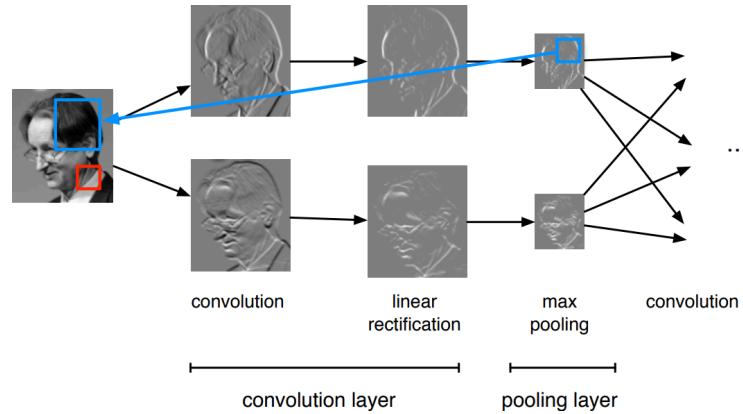
We want to design a neural architecture such that:

- in the earliest layers, the network responds similarly to similar patches of the image, regardless of their location;
- the earliest layers focus on local regions of the image, without regard for the contents of the image in distant regions;
- in the later layers, the network combines the information from the earlier layers to focus on larger and larger regions of the image, eventually combining all the information from the image to classify the image into a category.

Convolutional networks

Convolutional neural networks extend fully connected architectures with

- convolutional layers acting as local feature detectors;
- pooling layers acting as spatial down-samplers.



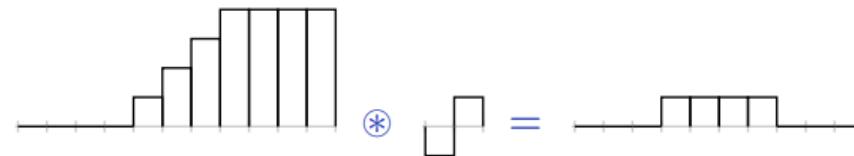
1d convolution

For the one-dimensional input $\mathbf{x} \in \mathbb{R}^W$ and the convolutional kernel $\mathbf{u} \in \mathbb{R}^w$, the discrete convolution $\mathbf{x} \circledast \mathbf{u}$ is a vector of size $W - w + 1$ such that

$$(\mathbf{x} \circledast \mathbf{u})[i] = \sum_{m=0}^{w-1} \mathbf{x}_{m+i} \mathbf{u}_m.$$

Convolutions can implement differential operators:

$$(0, 0, 0, 0, 1, 2, 3, 4, 4, 4) \circledast (-1, 1) = (0, 0, 0, 1, 1, 1, 1, 0, 0, 0)$$



or crude template matchers:

$$(0, 0, 3, 0, 0, 0, 0, 0, 3, 0, 3, 0, 0, 0) \circledast (1, 0, 1) = (3, 0, 3, 0, 0, 0, 3, 0, 6, 0, 3, 0)$$

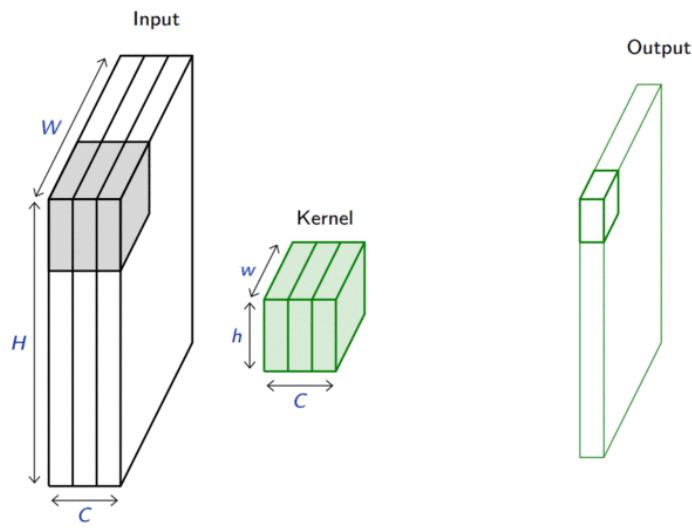


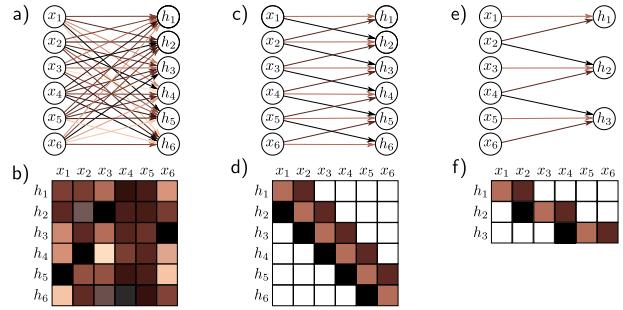
Convolutional layers

A convolutional layer is defined by a set of K kernels \mathbf{u} of size $c \times h \times w$, where h and w are the height and width of the kernel, and c is the number of channels of the input.

Assuming as input a 3D tensor $\mathbf{x} \in \mathbb{R}^{C \times H \times W}$, the output of the convolutional layer is a set of K feature maps of size $H' \times W'$, where $H' = H - h + 1$ and $W' = W - w + 1$. Each feature map \mathbf{o} is the result of convolving the input with a kernel, that is

$$\mathbf{o}_{j,i} = (\mathbf{x} \circledast \mathbf{u})[j, i] = \sum_{c=0}^{C-1} \sum_{n=0}^{h-1} \sum_{m=0}^{w-1} \mathbf{x}_{c,n+j,m+i} \mathbf{u}_{c,n,m}$$





Convolutional layers (c-f) are a special case of fully connected layers (a-b) where hidden units are connected to local regions of the input through shared weights (the kernels).

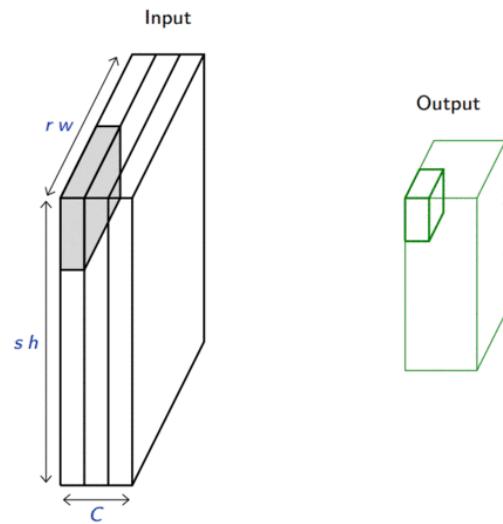
- The connectivity allows the network to learn local patterns in the input.
- Weight sharing allows the network to learn the same patterns at different locations in the input.

Pooling layers

Pooling layers are used to progressively reduce the spatial size of the representation, hence capturing longer-range dependencies between features.

Considering a pooling area of size $h \times w$ and a 3D input tensor $\mathbf{x} \in \mathbb{R}^{C \times (rh) \times (sw)}$, max-pooling produces a tensor $\mathbf{o} \in \mathbb{R}^{C \times r \times s}$ such that

$$\mathbf{o}_{c,j,i} = \max_{n < h, m < w} \mathbf{x}_{c,rj+n, si+m}.$$



(Step-by-step code example)

Recurrent networks

When the input is a sequence $\mathbf{x}_{1:T}$, the feedforward network can be made **recurrent** by computing a sequence $\mathbf{h}_{1:T}$ of hidden states, where \mathbf{h}_t is a function of both \mathbf{x}_t and the previous hidden states in the sequence.

For example,

$$\mathbf{h}_t = \sigma(\mathbf{W}_{xh}^T \mathbf{x}_t + \mathbf{W}_{hh}^T \mathbf{h}_{t-1} + \mathbf{b}),$$

where \mathbf{h}_{t-1} is the previous hidden state in the sequence.

Notice how this is similar to filtering and dynamic decision networks:

- \mathbf{h}_t can be viewed as some current belief state;
- $\mathbf{x}_{1:T}$ is a sequence of observations;
- \mathbf{h}_{t+1} is computed from the current belief state \mathbf{h}_t and the latest evidence \mathbf{x}_t through some fixed computation (in this case a neural network, instead of being inferred from the assumed dynamics).
- \mathbf{h}_t can also be used to decide on some action, through another network f such that $a_t = f(\mathbf{h}_t; \theta)$.

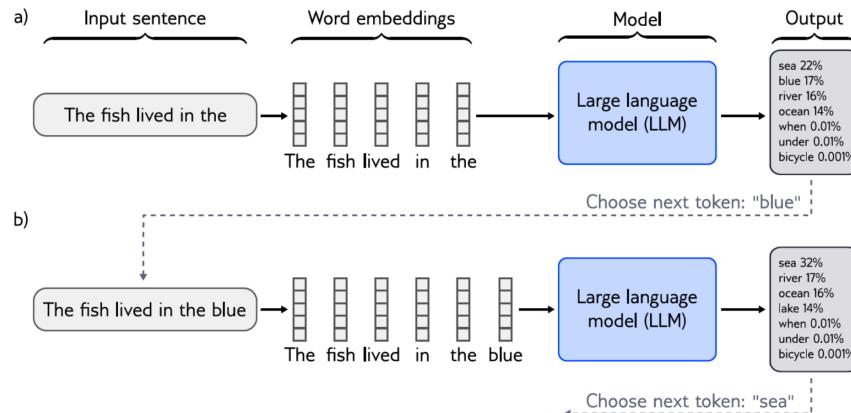
MariFlow - Self-Driving Mario Kart w/Recurrent Neural Network



A recurrent network playing Mario Kart.

Transformers

Transformers are deep neural networks at the core of large-scale language models.

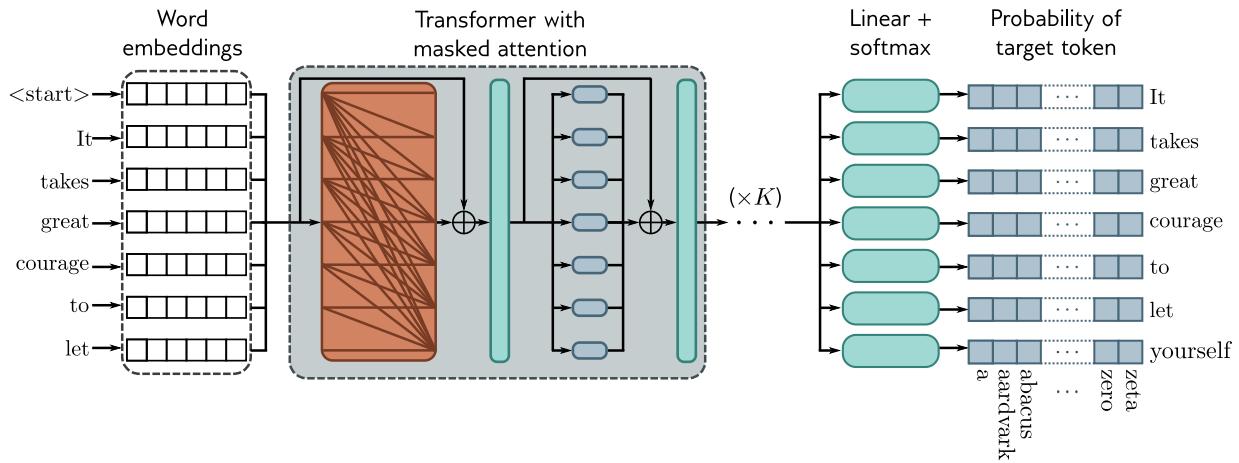


For language modeling, transformers define an **autoregressive model** that predicts the next word in a sequence given the previous words.

Formally,

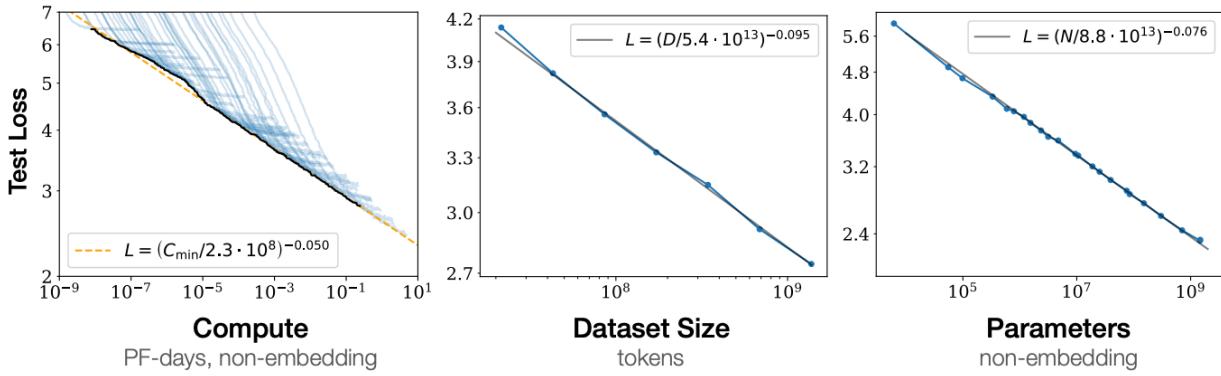
$$p(w_{1:t}) = p(w_1) \prod_{t=2}^T p(w_t | w_{1:t-1}),$$

where w_t is the next word in the sequence and $w_{1:t-1}$ are the previous words.



The decoder-only transformer is a stack of K transformer blocks that process the input sequence in parallel using (masked) self-attention.

The output of the last block is used to predict the next word in the sequence, as in a regular classifier.



Scaling laws

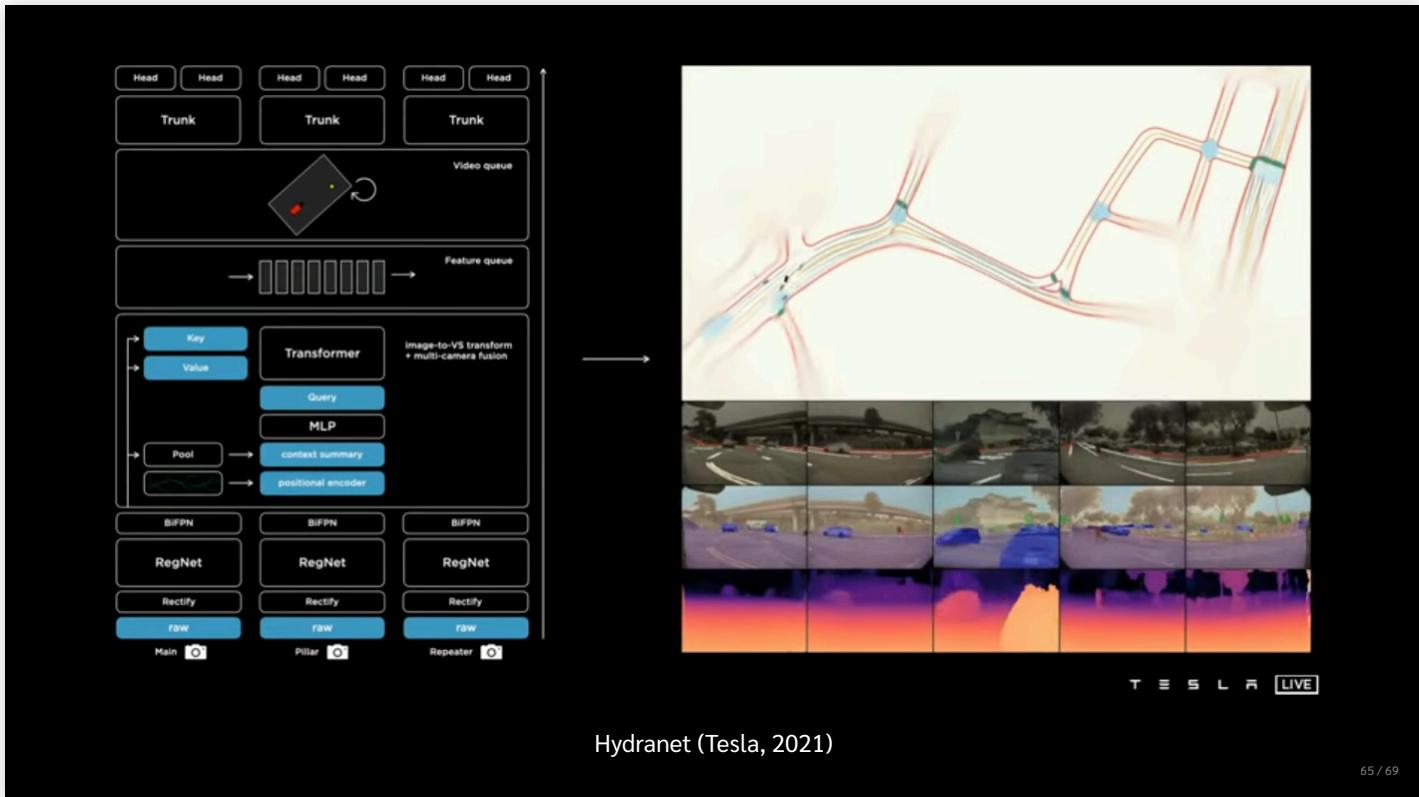
- The more data, the better the model.
- The more parameters, the better the model.
- The more compute, the better the model.

AI beyond Pacman

How AI Helps Autonomous Vehicles See Outside the Box - NVIDIA DR...



How AI Helps Autonomous Vehicles See Outside the Box
(See also other [episodes](#) from NVIDIA DRIVE Labs)



Camels, Code & Lab Coats: How AI Is Advancing Science and M...



How machine learning is advancing medicine (Google, 2018)

สรุป

- Deep learning เป็นเครื่องมือที่ทรงพลังสำหรับการเรียนรู้จากข้อมูล
- โครงข่ายประสาทเทียมประกอบด้วยชั้นของเซลล์ประสาทที่เชื่อมต่อถึงกัน
- น้ำหนักของการเชื่อมต่อจะถูกเรียนรู้โดยการหาค่าต่ำสุดของฟังก์ชันการสูญเสีย
- โครงข่ายสัมภัตนาการ (Convolutional networks) ใช้สำหรับการประมวลผลภาพ
- Transformers ใช้สำหรับการประมวลผลภาษา



"วิธีการเรียนรู้ของเครื่องแบบดั้งเดิมจำเป็นต้องมีผู้ออกแบบที่เป็นมุขย์ที่มีความชำนาญในการออกแบบตัวแยกรุณลักษณ์ (feature extractor) ที่เหมาะสม..."

การเรียนรู้เชิงลึก (deep learning) เป็นเทคนิคการเรียนรู้ของเครื่องที่ช่วยให้คอมพิวเตอร์สามารถเรียนรู้จากประสบการณ์และเข้าใจโลกในแบบที่คำนับชั้นของแนวคิด โดยที่แต่ละแนวคิดถูกกำหนดขึ้นโดยสัมพันธ์กับแนวคิดที่ง่ายกว่า"

