5. Simple Models

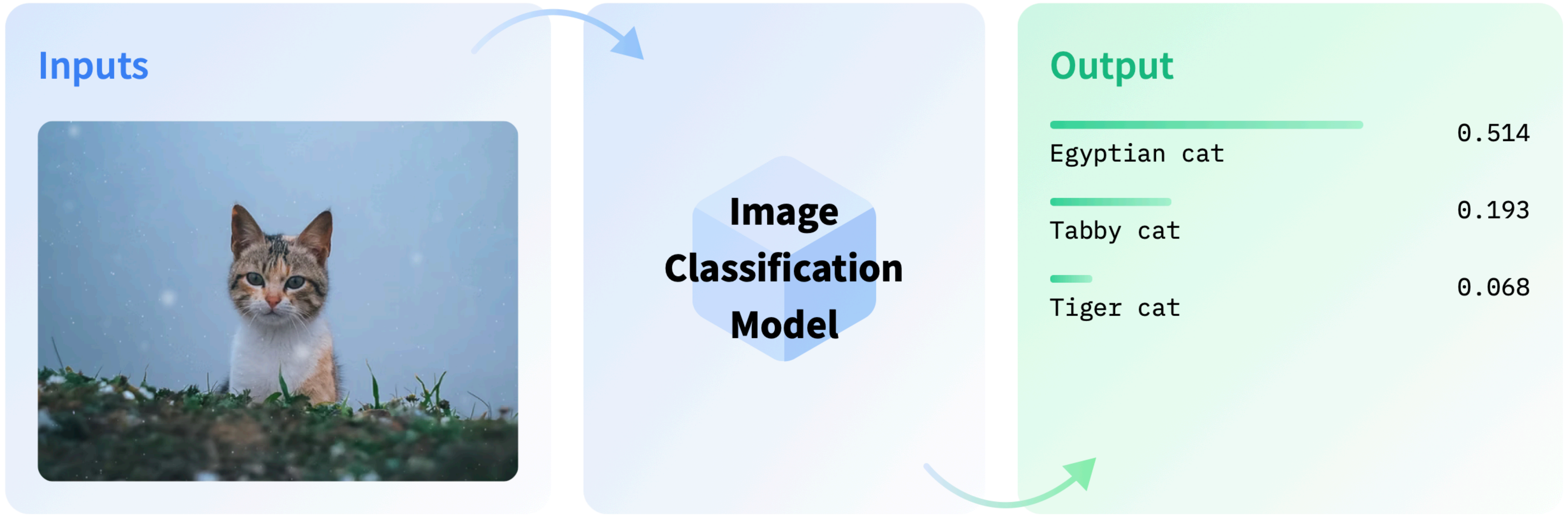
EECE454 Introduction to Machine Learning Systems

2023 Fall, Jaeho Lee

Naïve Bayes

Task

- We consider ***classification***—
  - Predict an output $Y \in \{1, \ldots, K\}$ (called "class") given the input X.



* image source: HuggingFace

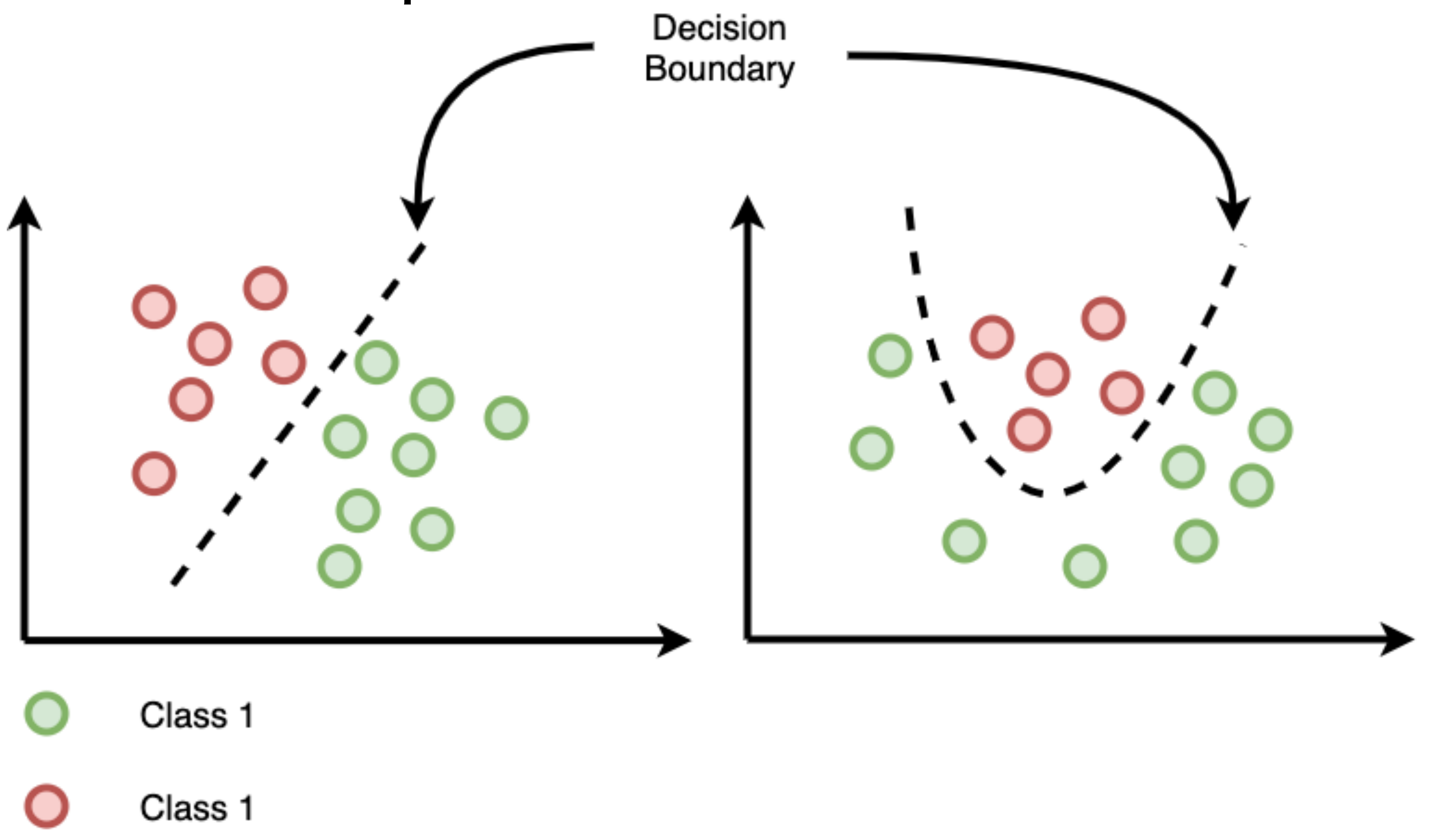
Terminology

- Suppose that there are two classes: 0, 1 (binary classification)
- Any classifier can be viewed as:

$$f(x) = \begin{cases} 0 & \cdots \quad x \in \mathscr{R}_0 \\ 1 & \cdots \quad x \in \mathscr{R}_1 \end{cases}$$

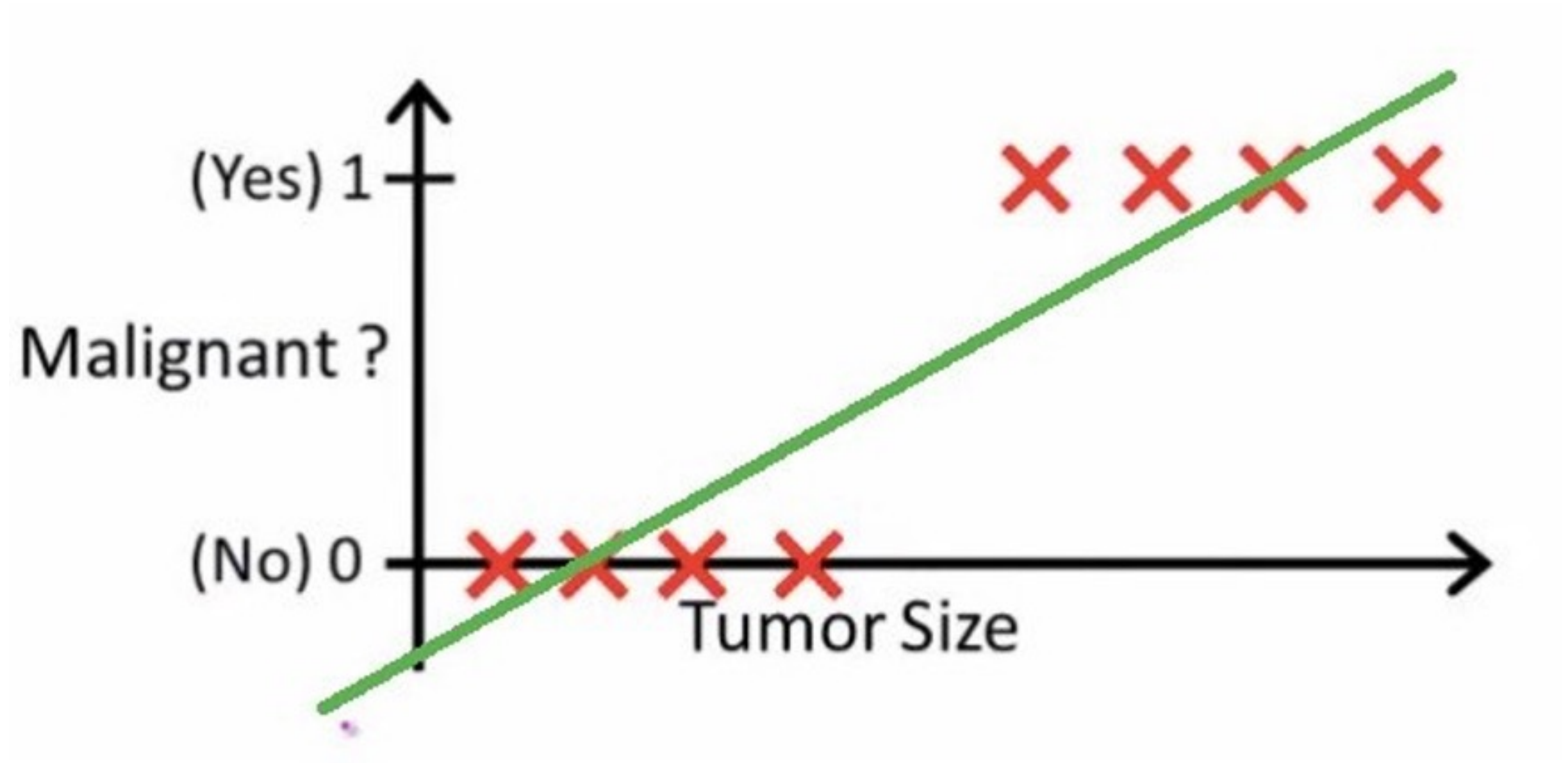
for some decision region $\mathscr{R}_0, \mathscr{R}_1$

- These are separated by the decision boundary.



Linear Regression for Classification?

- One can use linear regression for classification
  - ... but this is a bad choice.
- **Reason.** Very sensitive to outliers.
  - *Example*. Tumor malignancy prediction.



Naïve Bayes

- **Setting.** We have $\{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{n} \sim P_{XY}, \quad \mathbf{x}^{(i)} \in \mathbb{R}^d, y^{(i)} \in \{0,1\}$

- **Assumption.** Entries of $\mathbf{x}$ are conditionally independent given y.

$$p(\mathbf{x} \mid y) = \prod_{i=1}^{n} p(x_i \mid y)$$

  - Can be true for tabular data, but definitely wrong for images.
    - *naïve assumption ;(*
  - From now on, we let $d = 1$ WLOG.

Hypothesis

- Based on some human knowledge, we manually design two things:
  - ***likelihood models***: $p(x \mid y)$
  - ***priors***: $p(y)$
    - _Example._ ***Gaussian Likelihood*** has two parameters $\mu, \sigma \in \mathbb{R}$ for each y.

$$p(x \mid y) = \frac{1}{\sigma_y \sqrt{2\pi}} \exp\left(-\frac{(x - \mu_y)^2}{2\sigma_y^2} \right)$$

Hypothesis

- Then, our predictor is the **MAP estimator** which maximizes the posterior probability (MAP = maximum a posteriori)

$$f(\mathbf{x}) = \arg\max_y p(y \mid \mathbf{x})$$

$$= \arg\max_y \color{red}{p(y)}\color{blue}{p(\mathbf{x} \mid y)}$$

$$= \arg\max_y \left(p(y) \prod_{i=1}^{d} p(x_i \mid y) \right)$$

Hypothesis Space

- The hypothesis space is constructed by selecting parameters for:
  - likelihood model $p(x \mid y)$
  - prior distribution $p(y)$

- *Example.* Gaussian Likelihood $\Rightarrow$ select $\mu_0, \mu_1, \sigma_0, \sigma_1 \in \mathbb{R}$

  Bernoulli prior $\Rightarrow$ select $p \in [0,1]$

Fitting the parameters

- To fit the parameters, we maximize the joint probability:

$$\max_{\theta} p_\theta(\mathbf{x}_1, \ldots, \mathbf{x}_n, y_1, \ldots, y_n) \quad = \quad \max_{\theta_\ell, \theta_p} \prod_{i=1}^{n} p_{\theta_\ell}(\mathbf{x}_i \,|\, y_i) p_{\theta_p}(y_i)$$

- Equivalent to solving ERM, with

$$= \min_{\theta_\ell, \theta_p} \sum_{i=1}^{n} \left(\underbrace{- \log p_{\theta_\ell}(\mathbf{x}_i \,|\, y_i)}_{\uparrow} - \log p_{\theta_p}(y_i) \right)$$

So-called ***negative log-likelihood (NLL) loss***

Fitting the parameters

- Again, equivalent to solving ***two optimizations separately***:

$$\min_{\theta_\ell} \sum_{i=1}^{n} \Big(-\log p_{\theta_\ell}(\mathbf{x}_i \,|\, y_i) \Big) \longleftarrow$$

such θ_ℓ is the ***maximum likelihood estimate (MLE)***

$$\min_{\theta_p} \sum_{i=1}^{n} \Big(-\log p_{\theta_p}(y_i) \Big)$$

Fitting the parameters

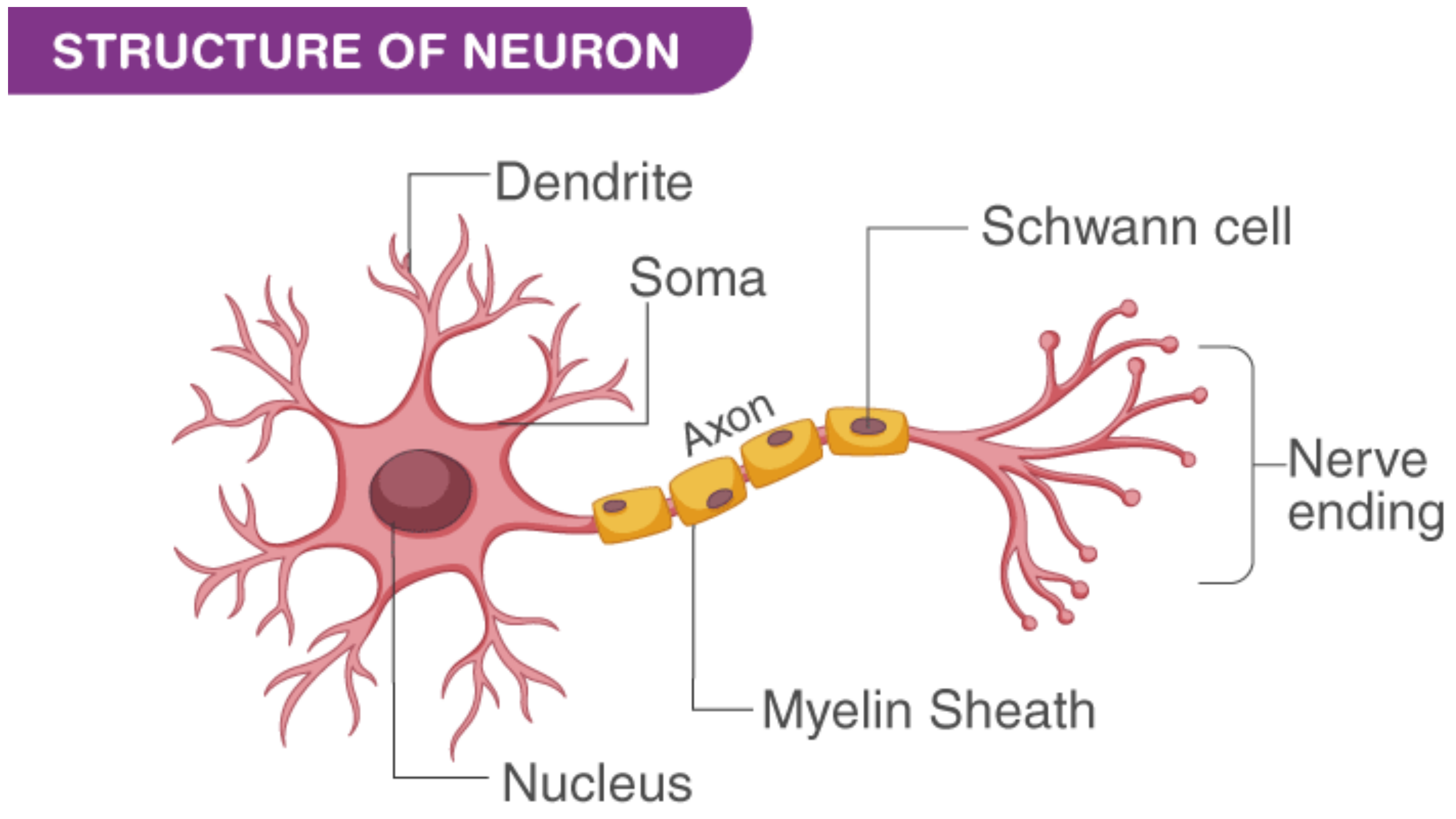
- ERM solutions are usually simple:
  - *Example.* Gaussian Likelihood
    - Use ***class-wise sample mean*** and ***classwise sample variance*** for $\mu_0, \mu_1, \sigma_0^2, \sigma_1^2$
  - *Example.* Bernoulli Prior
    - Simply use the ***frequency***

$$p = \frac{\text{\#1s in dataset}}{n}$$

Perceptron & Logistic Regression

Perceptron

- The first "neural network" by Rosenblatt (1958).



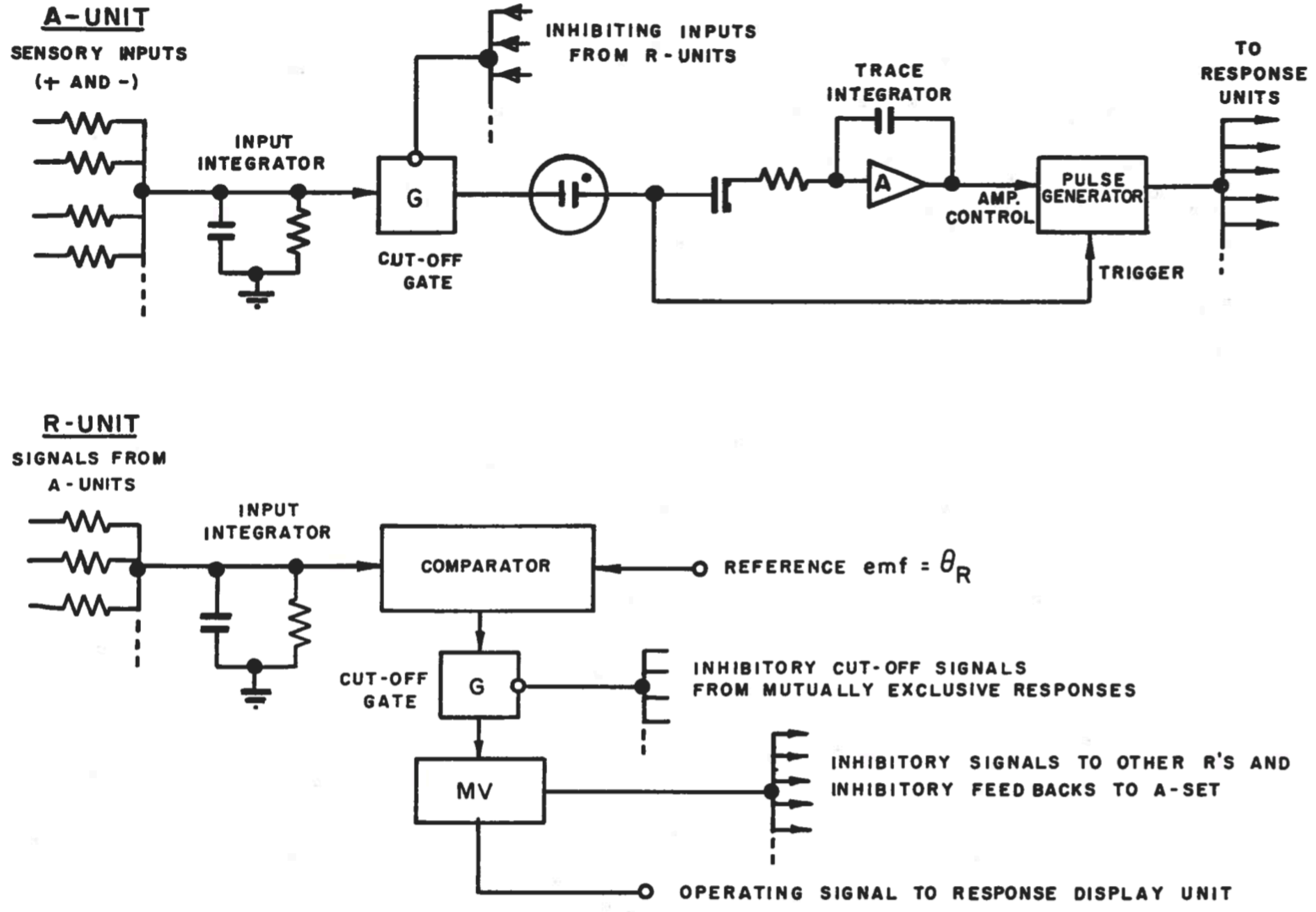
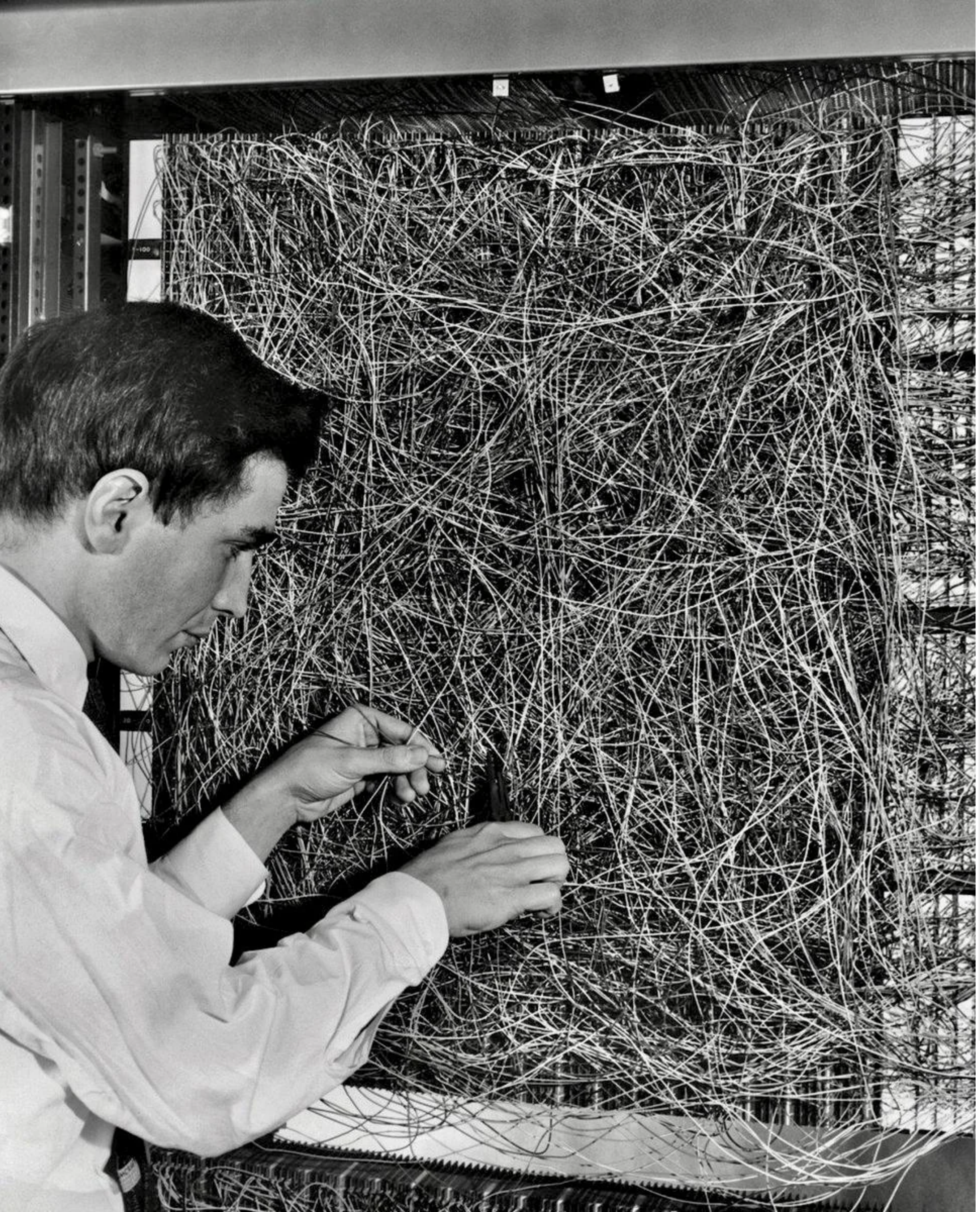
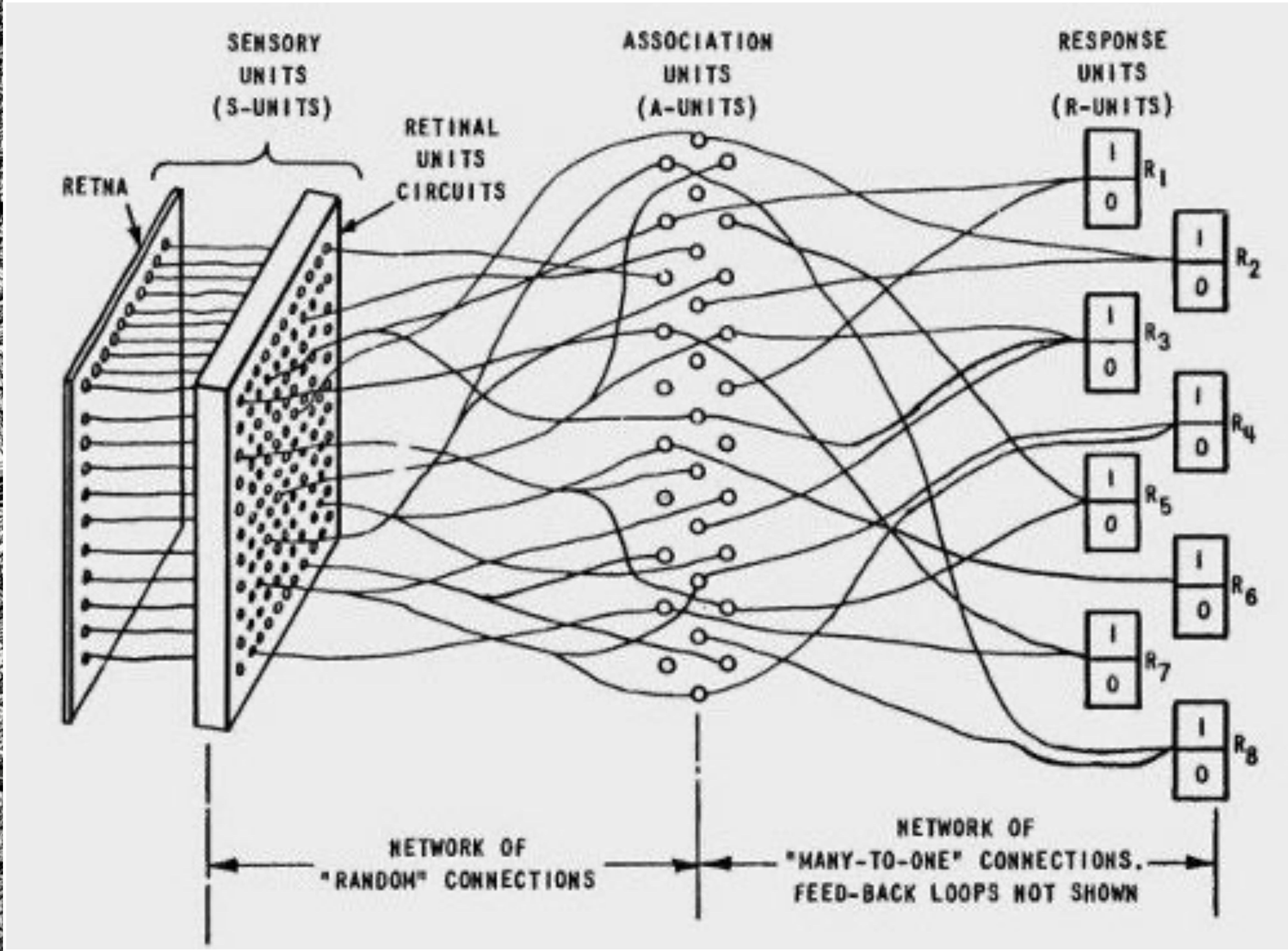

A-UNIT
SENSORY INPUTS (+ AND -)
INPUT INTEGRATOR
G
CUT-OFF GATE
INHIBITING INPUTS FROM R-UNITS
TRACE INTEGRATOR
A
AMP. CONTROL
PULSE GENERATOR
TRIGGER
TO RESPONSE UNITS
R-UNIT
SIGNALS FROM A-UNITS
INPUT INTEGRATOR
COMPARATOR
REFERENCE emf = θ_R
CUT-OFF GATE
G
INHIBITORY CUT-OFF SIGNALS FROM MUTUALLY EXCLUSIVE RESPONSES
MV
INHIBITORY SIGNALS TO OTHER R'S AND INHIBITORY FEED BACKS TO A-SET
OPERATING SIGNAL TO RESPONSE DISPLAY UNIT

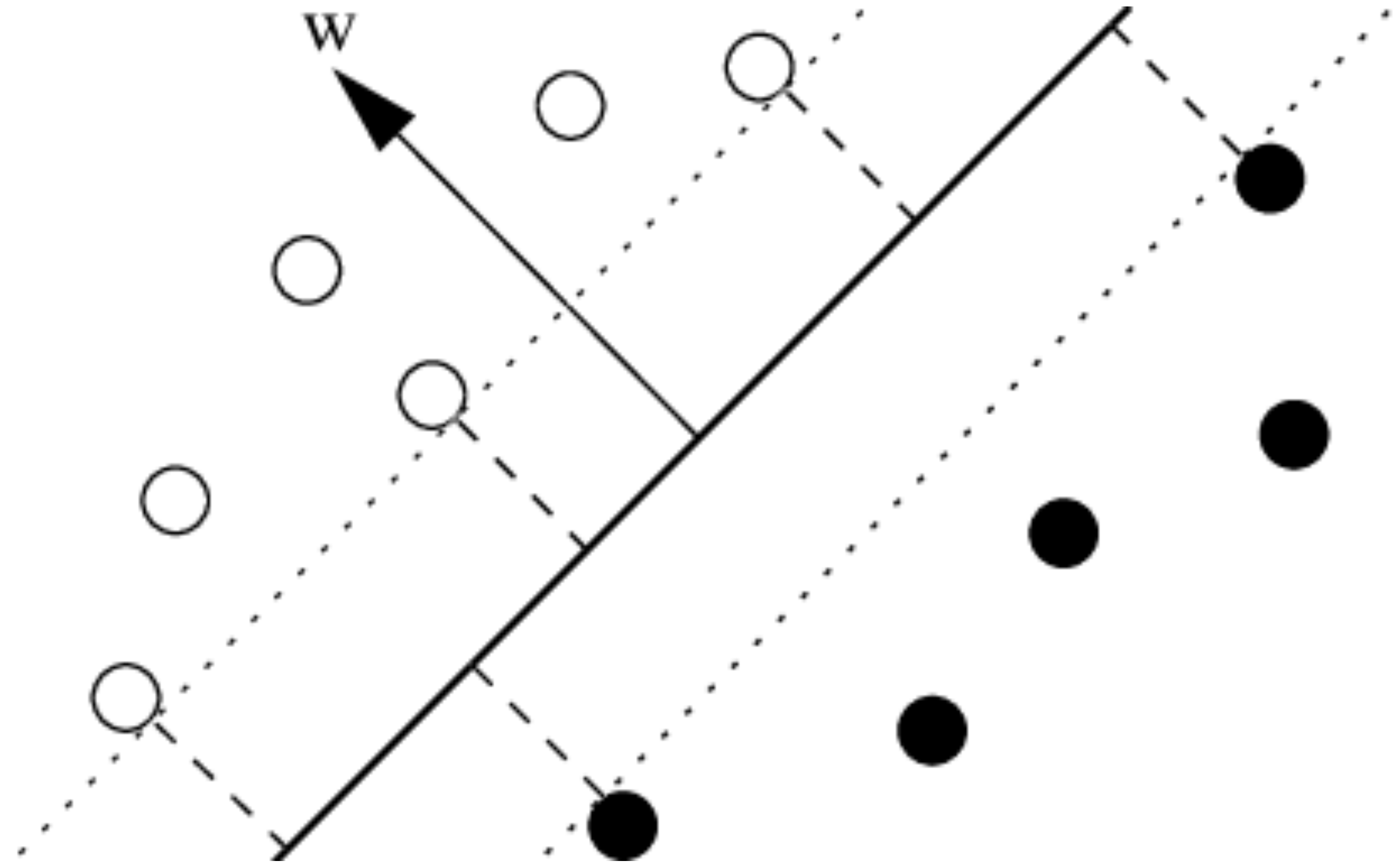
FIGURE 5
DESIGN OF TYPICAL UNITS

Rosenblatt, "The Perceptron: A Perceiving and Recognizing Automaton," 1957




SENSORY UNITS (S-UNITS)
ASSOCIATION UNITS (A-UNITS)
RESPONSE UNITS (R-UNITS)
RETNA
RETINAL UNITS CIRCUITS
R1
R2
R3
R4
R5
R6
R7
R8
I
O
NETWORK OF "RANDOM" CONNECTIONS
NETWORK OF "MANY-TO-ONE" CONNECTIONS. FEED-BACK LOOPS NOT SHOWN

Perceptron



- Mathematically, quite simple—
  - We use the ***sign of linear models*** as our hypothesis space.

$$\left\{ f_\theta(\,\cdot\,) \;\middle|\; f_\theta(\mathbf{x}) = \mathbf{1}\left[\theta_{\mathbf{1}}{}^\top \mathbf{x} + \theta_0 > 0\right] \right\}$$

$$= \left\{ f_\theta(\,\cdot\,) \;\middle|\; f_\theta(\mathbf{x}) = \mathbf{1}\left[\theta^\top \tilde{\mathbf{x}} > 0\right] \right\}$$

- **Problem.** Taking derivatives w.r.t. $\mathbf{1}[\,\cdot\,]$ is nasty.

  (indicator function; 1 if the bracketed event is true, 0 if false)

Loss

- To optimize, we use the loss

$$\ell(y, f_\theta(\mathbf{x})) = (f_\theta(\mathbf{x}) - y) \cdot \theta^\top \mathbf{x}$$

  - That is, we have loss $|\theta^\top \mathbf{x}|$ when wrong. (penalize confidence?)
    0 when correct.

  - **Note.** It is common to use loss functions different from the performance criterion (e.g., cross entropy loss vs. accuracy)

    These are called ***surrogate loss***.

  - **Note.** If $\theta = 0$?

Optimization

- The original perceptron paper assumes that data comes one-by-one. (called online learning)
    - The gradient is $\nabla_\theta \ell(y, f_\theta(\mathbf{x})) = (f_\theta(\mathbf{x}) - y)\mathbf{x}$
        - If wrong for a sample with $y = 1$.

$$\theta^{(i+1)} = \theta^{(i)} + \eta \cdot \mathbf{x}$$

        - If wrong for a sample with $y = 0$

$$\theta^{(i+1)} = \theta^{(i)} - \eta \cdot \mathbf{x}$$

        - If correct, no change.

Logistic Regression

- **Idea.** Solve the classification by regression.
  - **How?** Approximate the quantity

$$\log\left(\frac{p(y=1\,|\,\mathbf{x})}{p(y=0\,|\,\mathbf{x})}\right) \approx \theta^\top \tilde{\mathbf{x}}$$

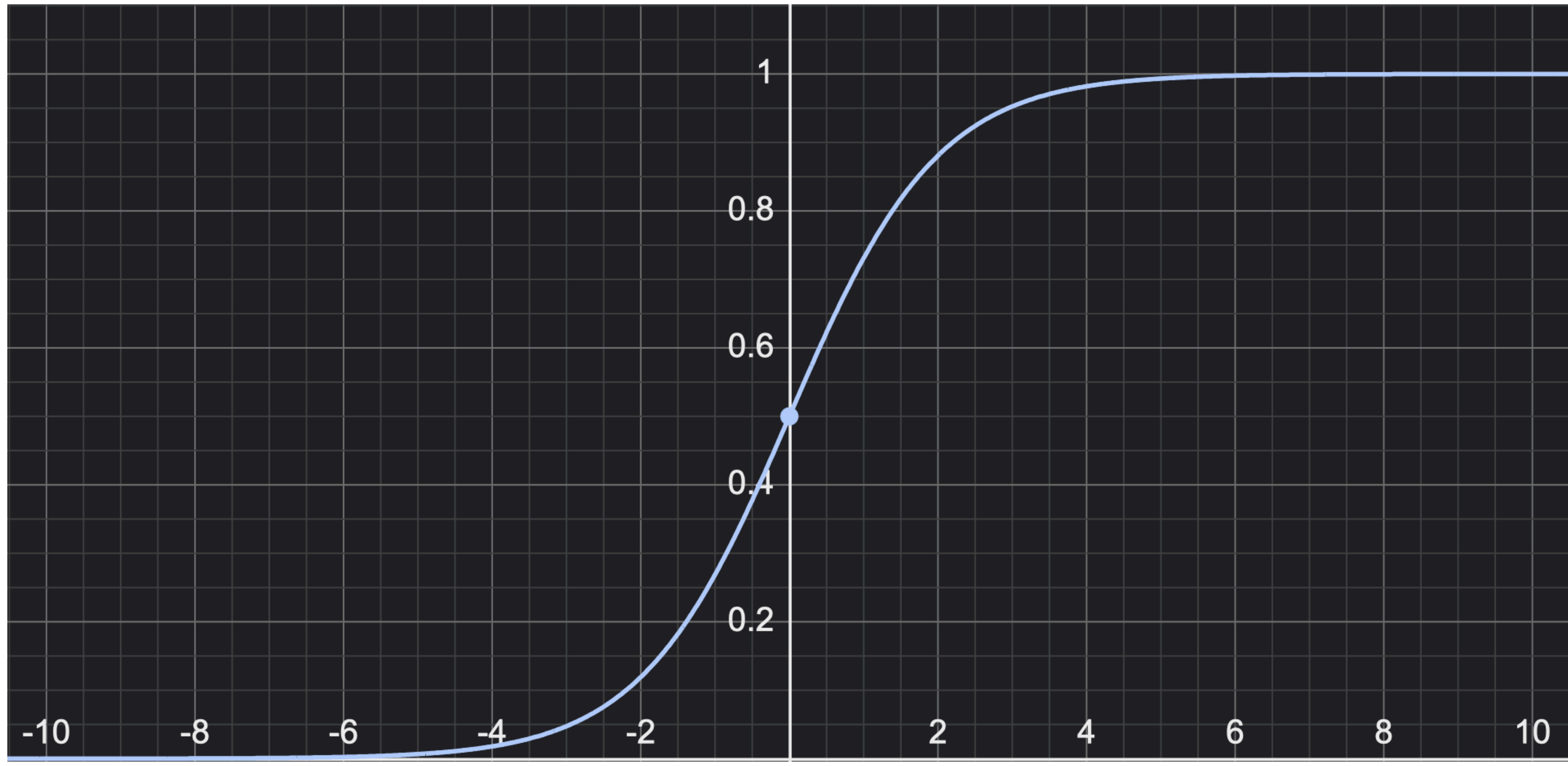
  - **Why not approximate $p(y=1\,|\,\mathbf{x})$?**
    - $p(y=1\,|\,x) \in [0,1]$,   but $\theta^\top \tilde{\mathbf{x}} \in (-\infty, +\infty)$

Logistic Regression

$$\log\left(\frac{p(y=1\,|\,\mathbf{x})}{p(y=0\,|\,\mathbf{x})}\right) \approx \theta^\top \tilde{\mathbf{x}}$$

- This is equivalent to saying that

$$p(y=1\,|\,\mathbf{x}) = \frac{1}{1+\exp(-\theta^\top \tilde{\mathbf{x}})}$$



- $\sigma(t) = 1/1 + \exp(-t)$ is called **logistic function.**

Logistic Regression

- Given the data, we maximize the ***log likelihood***—

$$\max_{\theta} \frac{1}{n} \sum_{i=1}^{n} \log p(y_i \mid \mathbf{x}_i)$$

- Or, minimize the ***NLL loss***—

$$\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \log \left(\frac{1}{p(y_i \mid \mathbf{x}_i)} \right)$$

Logistic Regression

$$\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \log \left(\frac{1}{p(y_i \mid \mathbf{x}_i)} \right)$$

- Again, this is equivalent to the ERM, with:
    - Hypothesis space is $\{f_\theta(\mathbf{x}) = \sigma(\theta^\top \tilde{\mathbf{x}})\}$
    - Loss is the **cross-entropy** $\ell(y, t) = \mathrm{CE}(\mathbf{1}_y, [t, 1 - t])$

$$= \log(t)^{-y} + \log(1 - t)^{y-1}$$

Optimizing

- The training risk can be written more tediously as:

$$\frac{1}{n}\sum_{i=1}^{n}(-y_i)\log(\sigma(\theta^\top \tilde{\mathbf{x}}_i)) + (y_i - 1)\log(1 - \sigma(\theta^\top \tilde{\mathbf{x}}_i))$$

  - Convex, but no general closed-form solution.

- The gradient descent can be written as:

$$\theta^{(\text{new})} = \theta + \eta \cdot \frac{1}{n}\sum_{i=1}^{n}(y_i - \sigma(\theta^\top \tilde{\mathbf{x}}_i))\tilde{\mathbf{x}}_i$$

Note: Similar to perceptron, GD update is proportional to $(y - f_\theta)\mathbf{x}$

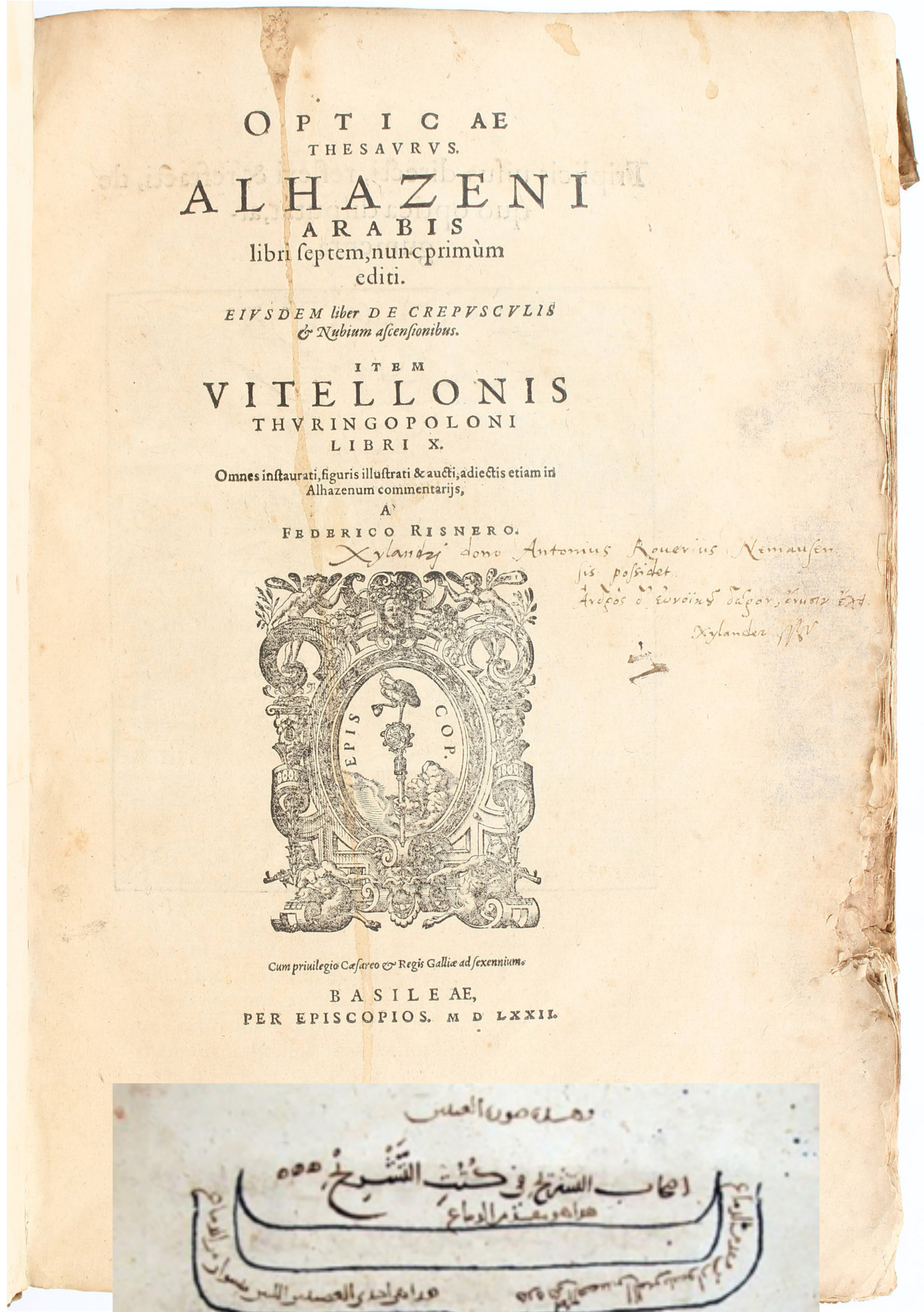
Nearest Neighbors

Nearest Neighbor

- Can be traced back to a book in 1021—
  كتاب المناظر ("The book of optics") by Ibn al-Haytham.

Visual Recognition = Nearest Neighbor

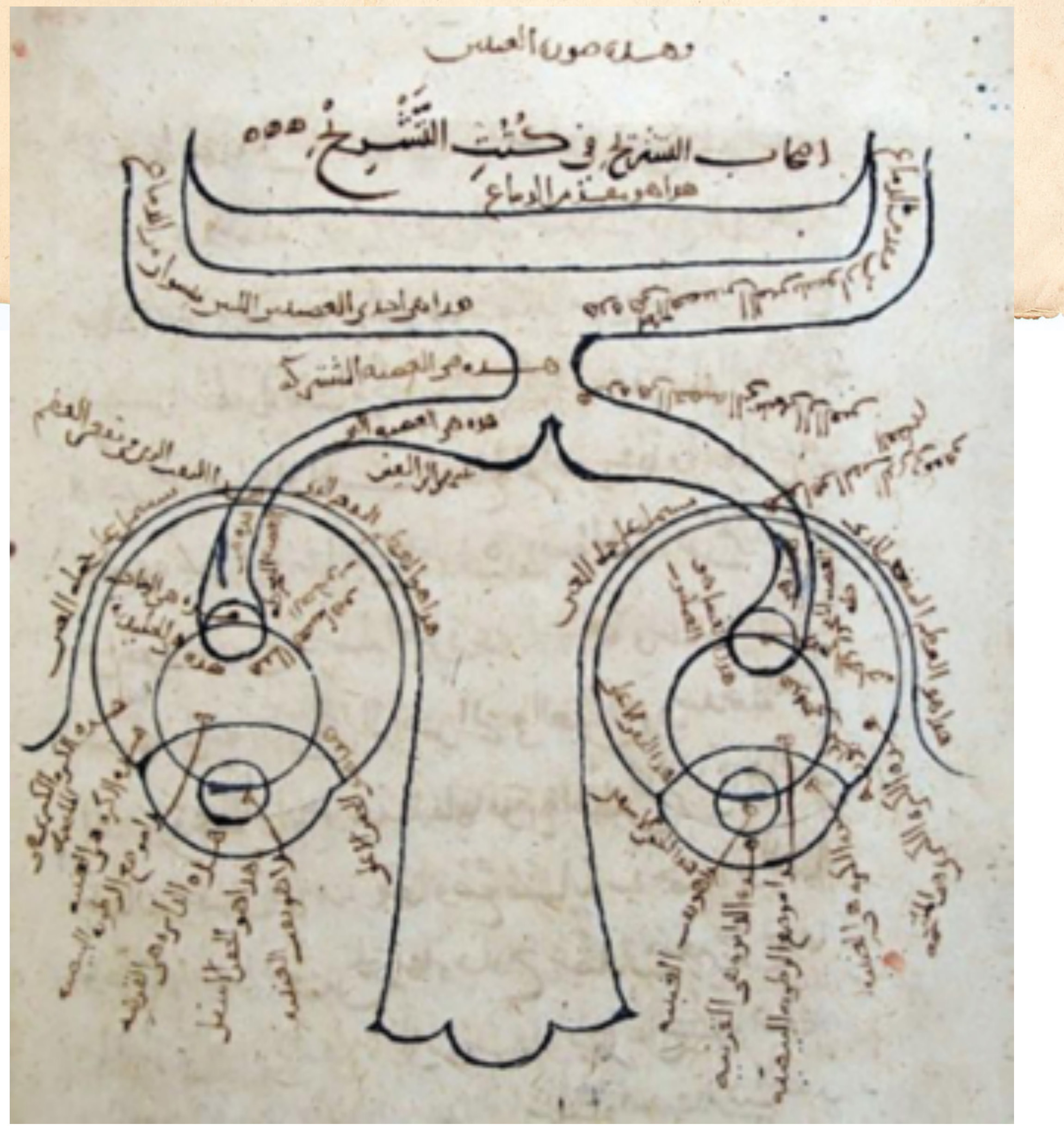
*"Recognition is the perception of similarity
between two forms—i.e., of the form sight
perceives at the moment of recognition
and the form of that visible object, or its like,
that it has perceived one or more times before."*



OPTICAE
THESAVRVS.
ALHAZENI
ARABIS
libri ſeptem, nunc primùm
editi.
EIVSDEM liber DE CREPVSCVLIS & Nubium aſcenſionibus.
ITEM
VITELLONIS
THVRINGOPOLONI
LIBRI X.
Omnes inſtaurati, figuris illuſtrati & aucti, adiectis etiam in Alhazenum commentarijs,
A
FEDERICO RISNERO.

Xylandri dono Antonius Rouerius Nemauſenſis poſsidet.
[illegible]
Xylander [illegible]

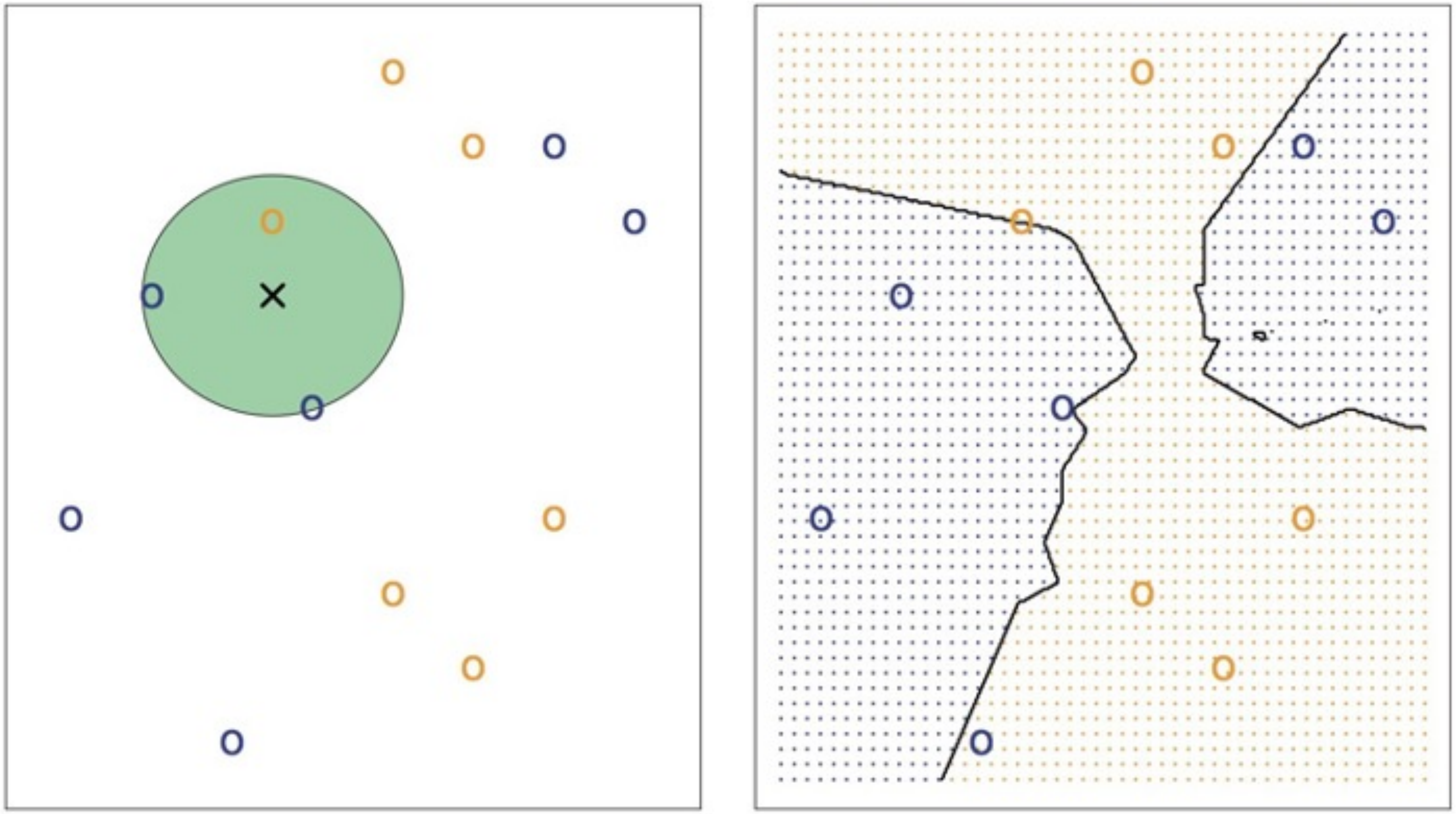
Cum priuilegio Cæſareo & Regis Galliæ ad ſexennium.
BASILEAE,
PER EPISCOPIOS. M D LXXII.



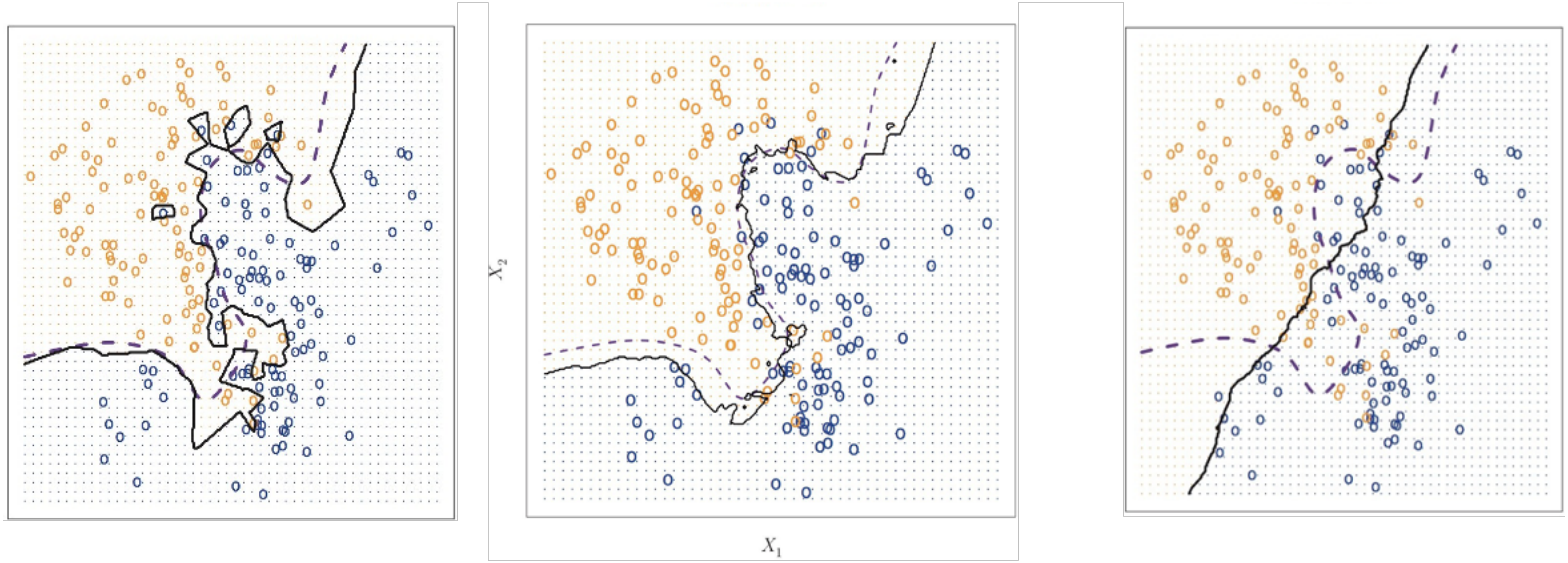
* image source: HuggingFace

K-Nearest Neighbors

- A nonlinear, nonparametric algorithm.
- **Algorithm.**
    - **Dataset.** We have a dataset $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$
    - **Training.** N/A
    - **Testing.** When a new sample $\mathbf{x}$ comes in:
        - Find k samples $\mathbf{x}_{(1)}, \ldots, \mathbf{x}_{(k)}$ in D that has smallest $\|\mathbf{x} - \mathbf{x}_{(i)}\|$.
        - Predict with the majority vote (classification) or averaging (regression).



k-NN with $k = 3$



Small k = More flexibility

Computational Complexity

- K-nearest neighbor is difficult to be scaled to large size.
  - **Good.** Does not take training time.
  - **Bad.** For testing, we need to compute n comparisons.
    - i.e., inference time $\propto$ # data

Parametric vs. Nonparametric

- **Parametric.** Uses a fixed number of parameters.
    - Linear Regression, Logistic Regression, Neural network, …

- **Nonparametric.** Uses flexible number or infinitely many parameters.
    - K-NN, Boosting Trees, Random Forest.

Cheers

- *Next up.* SVM