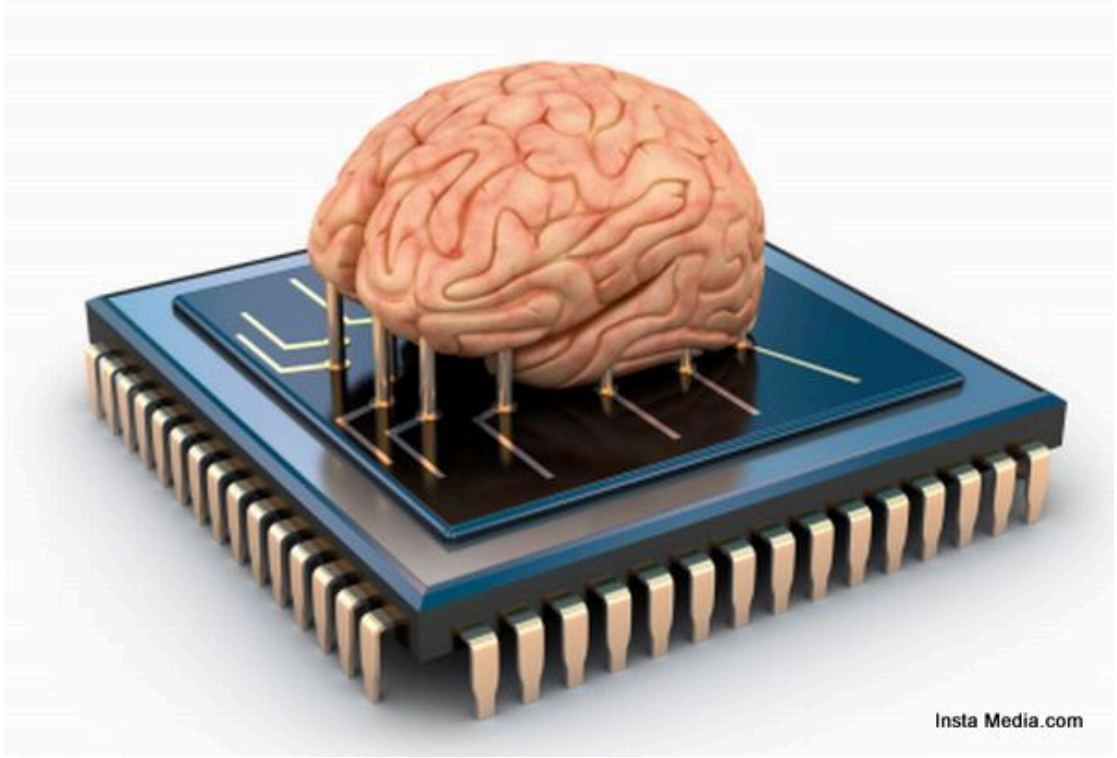
# Capacity Scaling of Artificial Neural Networks

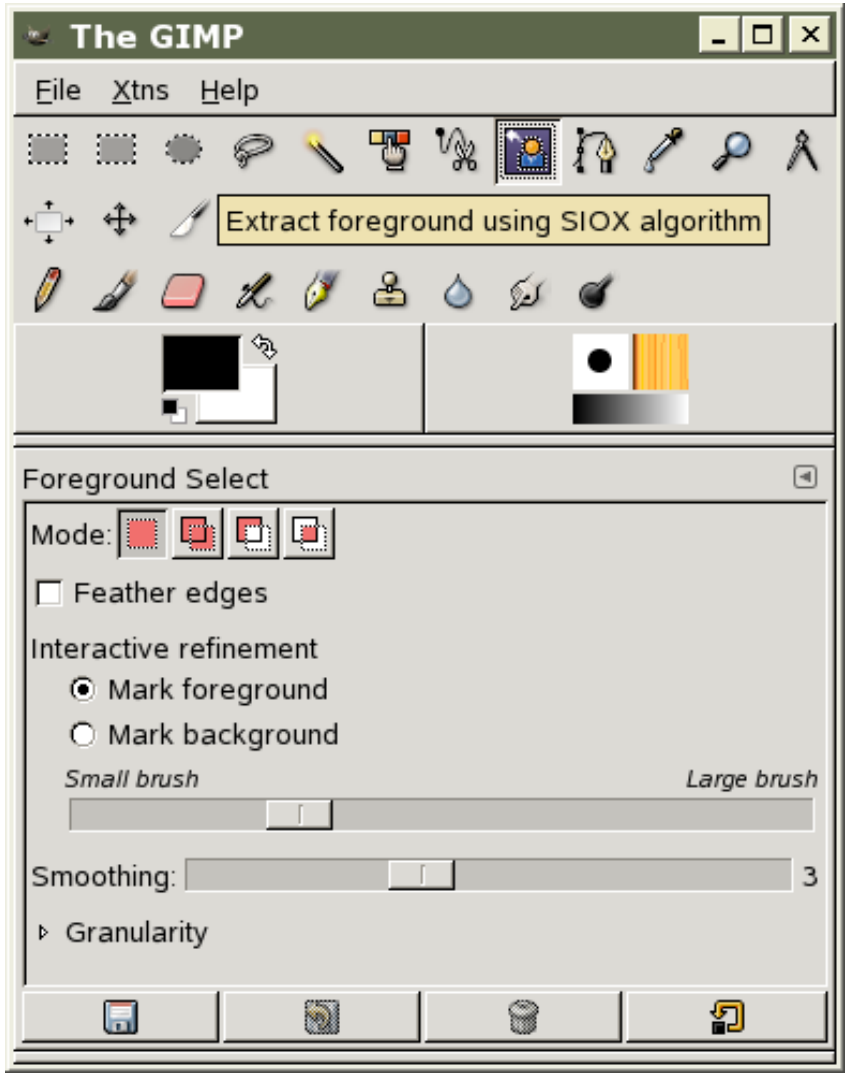


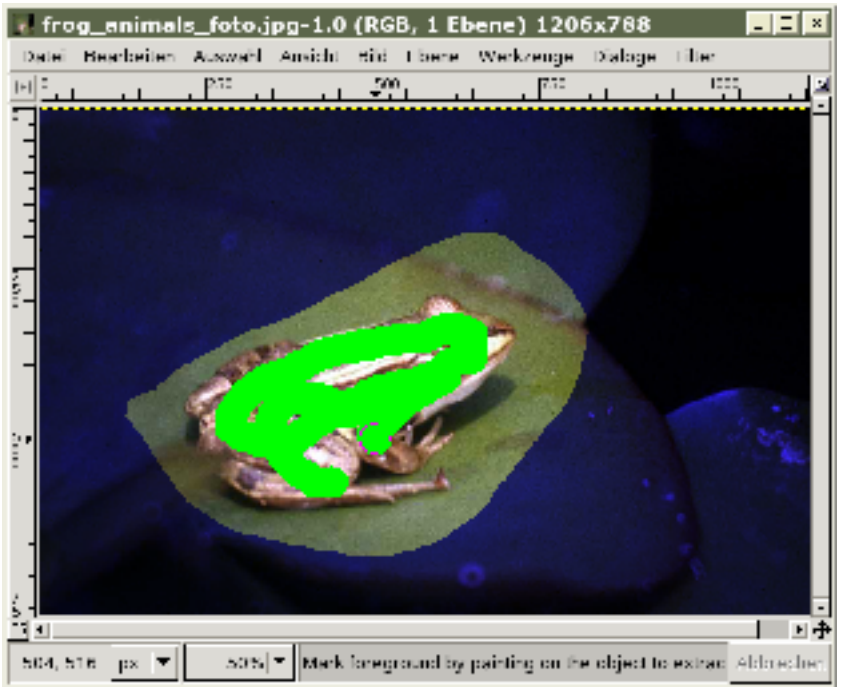

*Gerald Friedland, Mario Michael Krell*

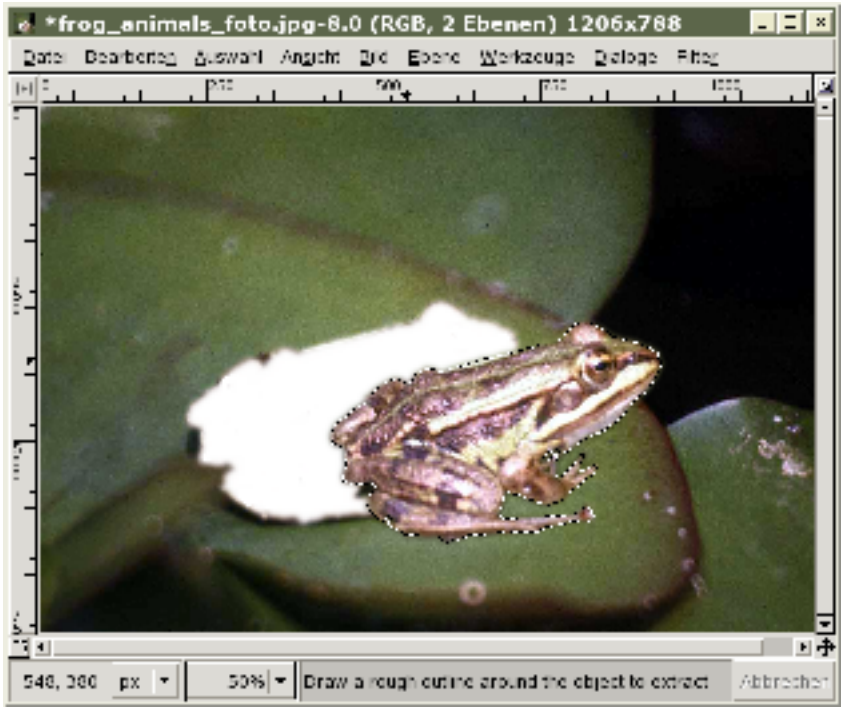
fractor@eecs.berkeley.edu

http://arxiv.org/abs/1708.06019

# Prior work

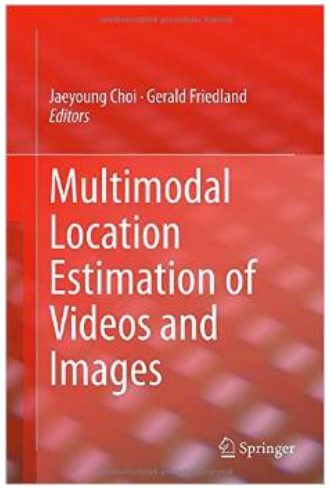



G. Friedland, K. Jantz, T. Lenz, F. Wiesel, R. Rojas: *A Practical Approach to Boundary-Accurate Multi-Object Extraction from Still Images and Videos*, to appear in Proceedings of the IEEE International Symposium on Multimedia (ISM2006), San Diego (California), December, 2006

# Multimodal Location Estimation



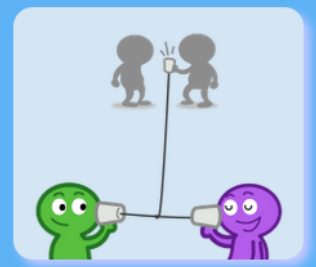

http://mmle.icsi.berkeley.edu

# Ten Principles for Online Privacy

You're Leaving Footprints

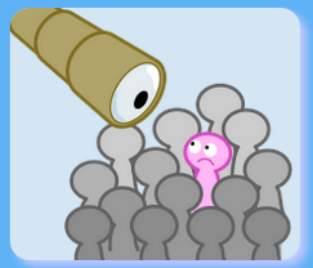
There's No Anonymity

Information Is Valuable



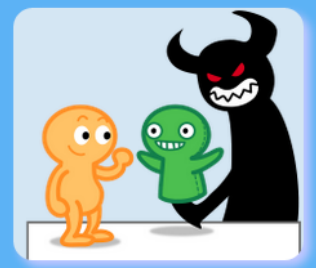
Someone Could Listen

Sharing Releases Control



Search Is Improving

Online Is Real



Identity Isn't Guaranteed

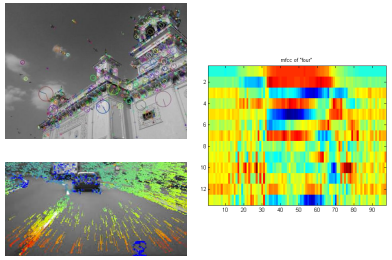
You Can't Escape

Privacy Requires Work

http://teachingprivacy.org

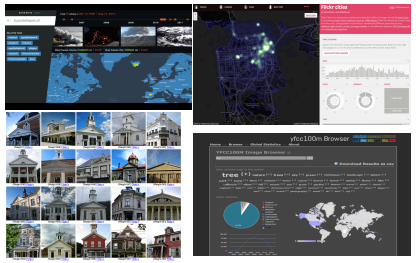
# The Multimedia Commons (YFCC100M)

100.2M Photos
800K Videos



Features for Machine Learning
(Visual, Audio, Motion, etc.)

User-Supplied Metadata
and New Annotations



Tools for Searching,
Processing, and Visualizing

**100M videos and images,** and a growing pool of **tools** for research with **easy access** through **Cloud Computing**

**Collaboration Between Academia and Industry:**

ICSI International Computer Science Institute

Lawrence Livermore National Laboratory

Yahoo! Labs

amazon web services

Berkeley University of California

**Benchmarks & Grand Challenges:**



ACM MM

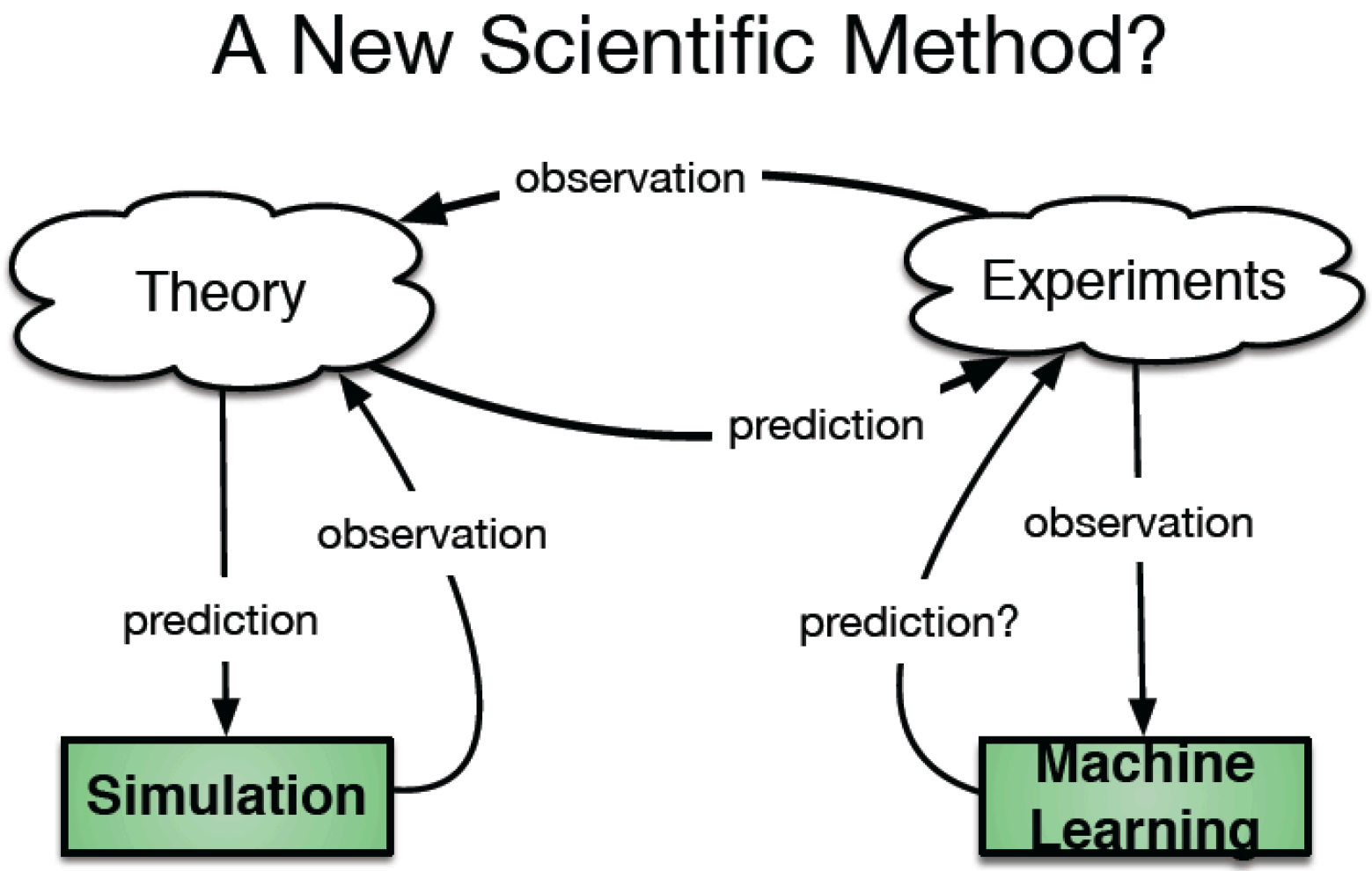
MediaEval Benchmark

Creative Commons or Public Domain

NSF

Supported in part by NSF Grant 1251276
"BIGDATA: Small: DCM: DA: Collaborative Research:
SMASH: Scalable Multimedia content AnalysiS in a High-level language"

# Data Science



# Neural Networks

What we <u>think</u> we know:

- Neural Networks can be trained to be more intelligent than humans e.g., beat Go masters
- Deep Learning is better than „shallow“ Learning
- Neural Networks are like the brain
- AI is going to take over the world soon
- Let’s pray to AI!

It is what we think we know already that often prevents us from learning.

*Claude Bernard*

# Occam's razor

Among competing hypotheses, the one with the fewest assumptions should be selected.

For each accepted explanation of a phenomenon, there may be an extremely large, perhaps even incomprehensible, number of possible and more complex alternatives, because one can always burden failing explanations with ad hoc hypotheses to prevent them from being falsified; therefore, simpler theories are preferable to more complex ones because they are more testable. (Wikipedia, Sep. 2017)

Source: Wikipedia

# Neural Networks

What we actually know:

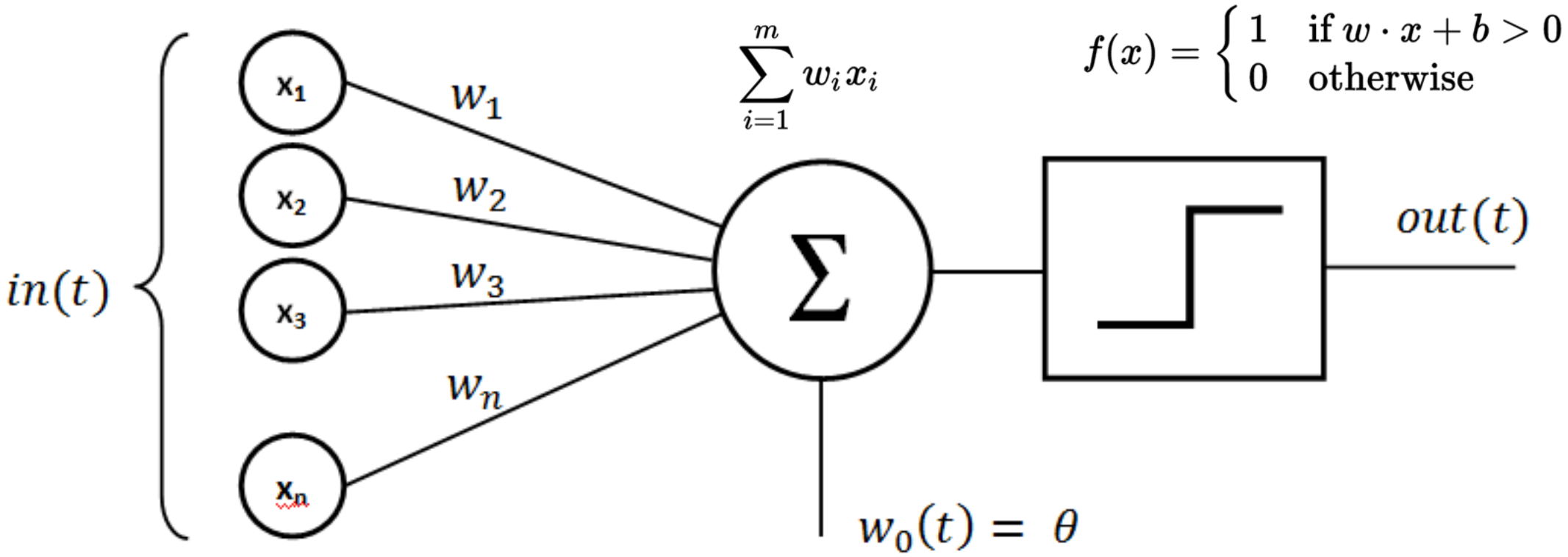
- Neural networks were created as memory (Memistor, Widrow 1962)
- Backpropagation is NP complete (Blum & Rivest 1989)
- Perceptron Learning is NP complete (Amaldi 1991)
- Knowing what function is implemented by a given network is at least NP complete (Cook & Levin 1971)

# By the end of this talk…

You will have learned that:

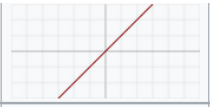
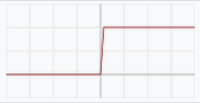
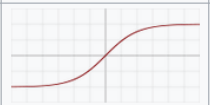
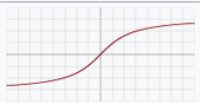
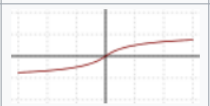
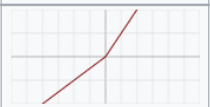
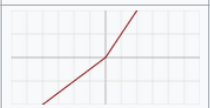
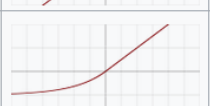
- Machine Learners have a capacity that is measurable
- Artificial Neural Networks with gating functions (Sigmoid, ReLU, etc.)
  - have a capacity that is analytically provable: 1 bit per parameter.
  - have 2 critical points that define their behavior (phase transitions): Lossless Memory Dimension and MacKay Dimension, scaling linearly with the number of weights, independent of the network architecture.
- Predicting and measuring these two critical points allows task-independent optimization of a concrete network architecture, learning algorithm, convergence tricks, etc…

# The Perceptron (Base Unit)




Source: Wikipedia

# Gating Functions… (too many)

| | | | | | |
|---|---|---|---|---|---|
| Identity |  | $f(x) = x$ | $f'(x) = 1$ | $(-\infty, \infty)$ | $C^\infty$ |
| Binary step |  | $f(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ 1 & \text{for} \quad x \geq 0 \end{cases}$ | $f'(x) = \begin{cases} 0 & \text{for} \quad x \neq 0 \\ ? & \text{for} \quad x = 0 \end{cases}$ | $\{0, 1\}$ | $C^{-1}$ |
| Logistic (a.k.a. Soft step) |  | $f(x) = \frac{1}{1 + e^{-x}}$ | $f'(x) = f(x)(1 - f(x))$ | $(0, 1)$ | $C^\infty$ |
| TanH |  | $f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$ | $f'(x) = 1 - f(x)^2$ | $(-1, 1)$ | $C^\infty$ |
| ArcTan |  | $f(x) = \tan^{-1}(x)$ | $f'(x) = \frac{1}{x^2 + 1}$ | $\left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$ | $C^\infty$ |
| Softsign [7][8] |  | $f(x) = \frac{x}{1 + \lvert x \rvert}$ | $f'(x) = \frac{1}{(1 + \lvert x \rvert)^2}$ | $(-1, 1)$ | $C^1$ |
| Rectified linear unit (ReLU)[9] |  | $f(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ x & \text{for} \quad x \geq 0 \end{cases}$ | $f'(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ 1 & \text{for} \quad x \geq 0 \end{cases}$ | $[0, \infty)$ | $C^0$ |
| Leaky rectified linear unit (Leaky ReLU)[10] |  | $f(x) = \begin{cases} 0.01x & \text{for} \quad x < 0 \\ x & \text{for} \quad x \geq 0 \end{cases}$ | $f'(x) = \begin{cases} 0.01 & \text{for} \quad x < 0 \\ 1 & \text{for} \quad x \geq 0 \end{cases}$ | $(-\infty, \infty)$ | $C^0$ |
| Parameteric rectified linear unit (PReLU)[11] |  | $f(\alpha, x) = \begin{cases} \alpha x & \text{for} \quad x < 0 \\ x & \text{for} \quad x \geq 0 \end{cases}$ | $f'(\alpha, x) = \begin{cases} \alpha & \text{for} \quad x < 0 \\ 1 & \text{for} \quad x \geq 0 \end{cases}$ | $(-\infty, \infty)$ | $C^0$ |
| Randomized leaky rectified linear unit (RReLU)[12] |  | $f(\alpha, x) = \begin{cases} \alpha x & \text{for} \quad x < 0 \\ x & \text{for} \quad x \geq 0 \end{cases}$ [1] | $f'(\alpha, x) = \begin{cases} \alpha & \text{for} \quad x < 0 \\ 1 & \text{for} \quad x \geq 0 \end{cases}$ | $(-\infty, \infty)$ | $C^0$ |
| Exponential linear unit (ELU)[13] |  | $f(\alpha, x) = \begin{cases} \alpha(e^x - 1) & \text{for} \quad x < 0 \\ x & \text{for} \quad x \geq 0 \end{cases}$ | $f'(\alpha, x) = \begin{cases} f(\alpha, x) + \alpha & \text{for} \quad x < 0 \\ 1 & \text{for} \quad x \geq 0 \end{cases}$ | $(-\alpha, \infty)$ | $C^1$ when $\alpha = 1$, otherwise $C^0$ |

Source: Wikipedia

# What is the purpose of a Neural Network?

Neural Networks memorize and optimize a function from some data to some labeling.

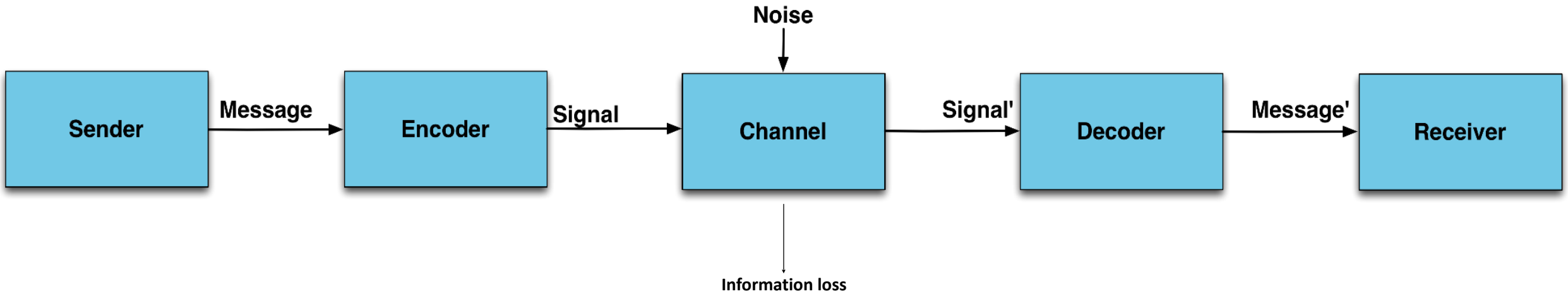
**f(data) -> labels**

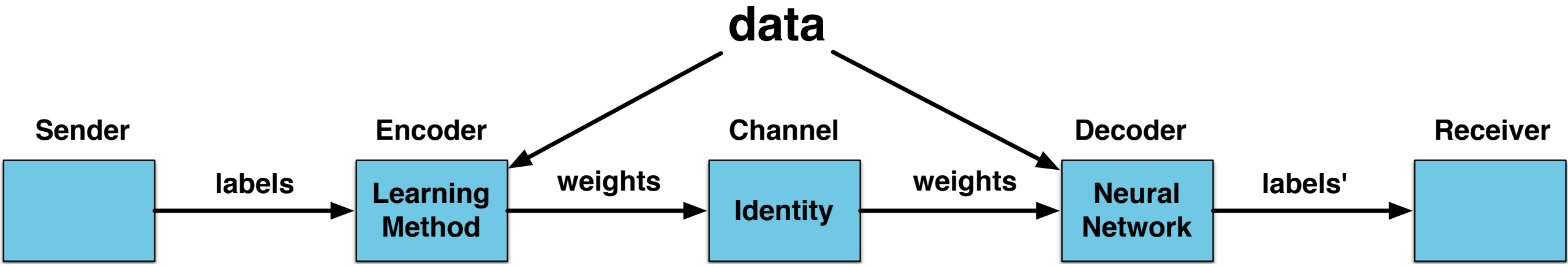
Question 1: How well can a function be memorized?

Question 2: What is minimum amount of parameters to memorize that function?

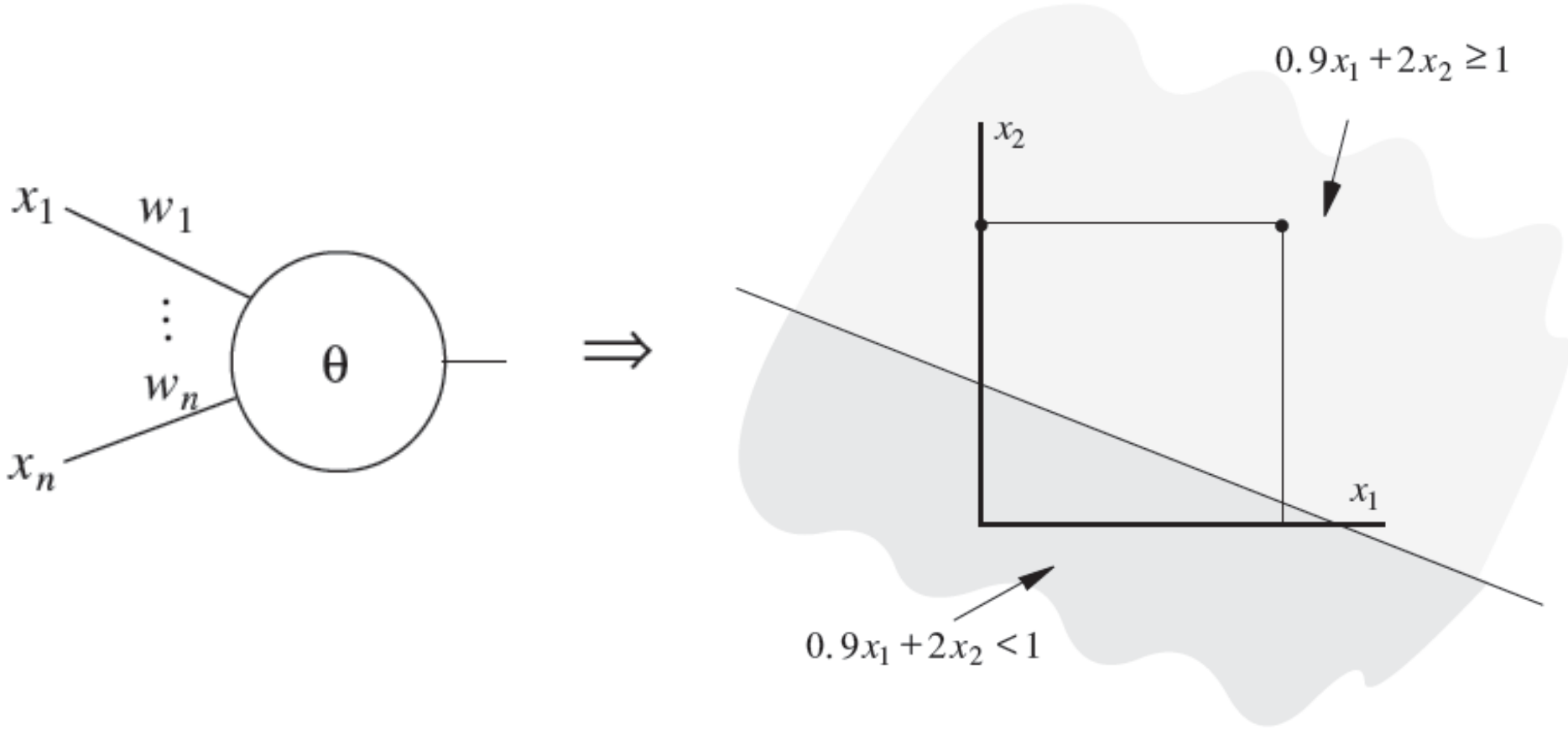
Question 3: Does my function generalize to other data?

# Machine Learning as Encoder/Decoder



# How good is the Perceptron as an Encoder?

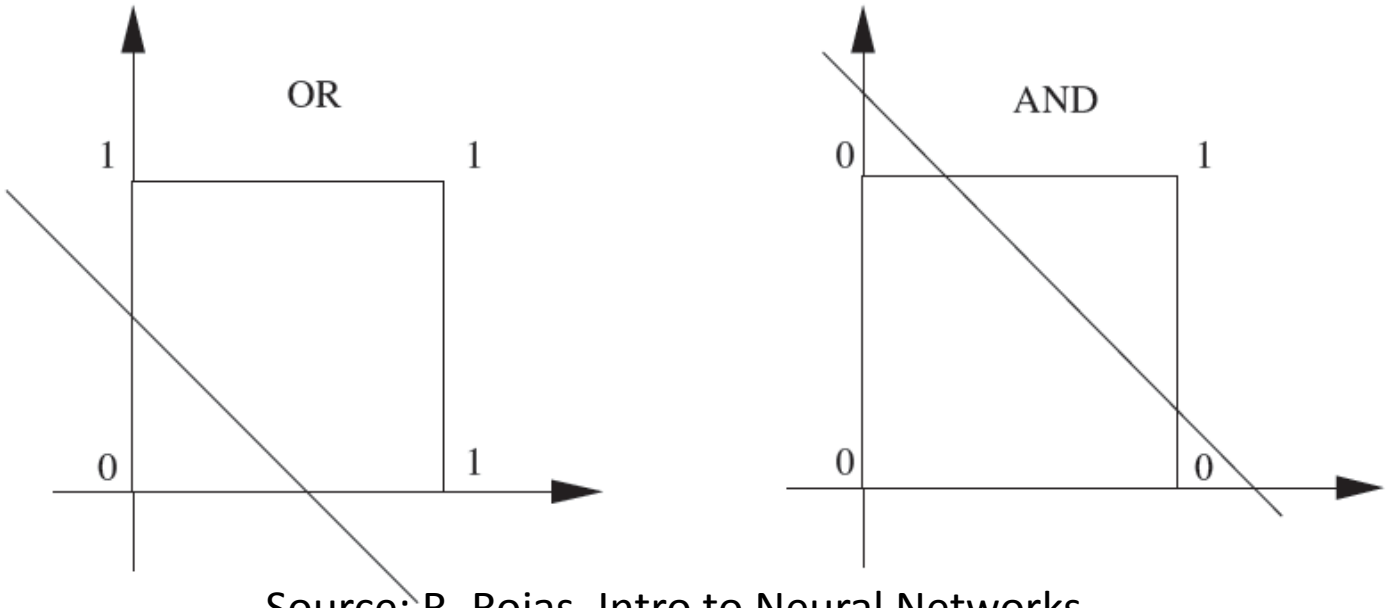



Source: R. Rojas, Intro to Neural Networ

N points => input space $2^N$ labels.

# Example: Boolean Function

| $x_1$ | $x_2$ | $f_0$ | $f_1$ | $f_2$ | $f_3$ | $f_4$ | $f_5$ | $f_6$ | $f_7$ | $f_8$ | $f_9$ | $f_{10}$ | $f_{11}$ | $f_{12}$ | $f_{13}$ | $f_{14}$ | $f_{15}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |




Source: R. Rojas, Intro to Neural Networks

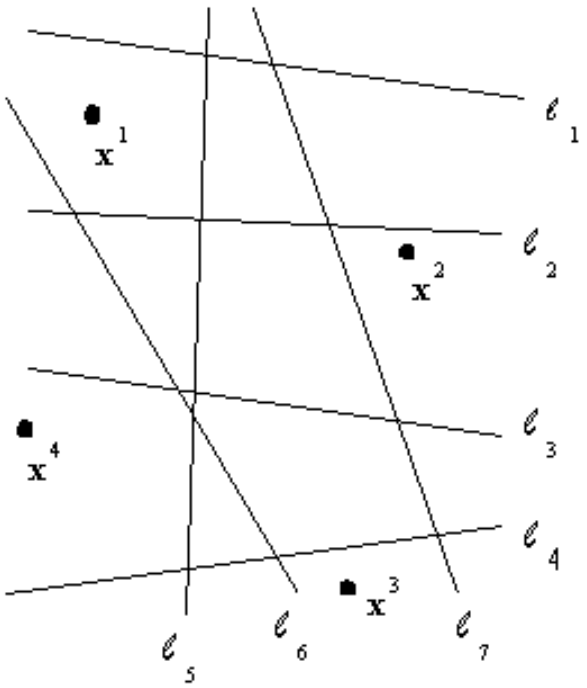
- $2^{2^v}$ functions of $v$ boolean variables
- $2^v$ labelings of $2^v$ points.
- For $v=2$, all but 2 functions work: XOR, NXOR

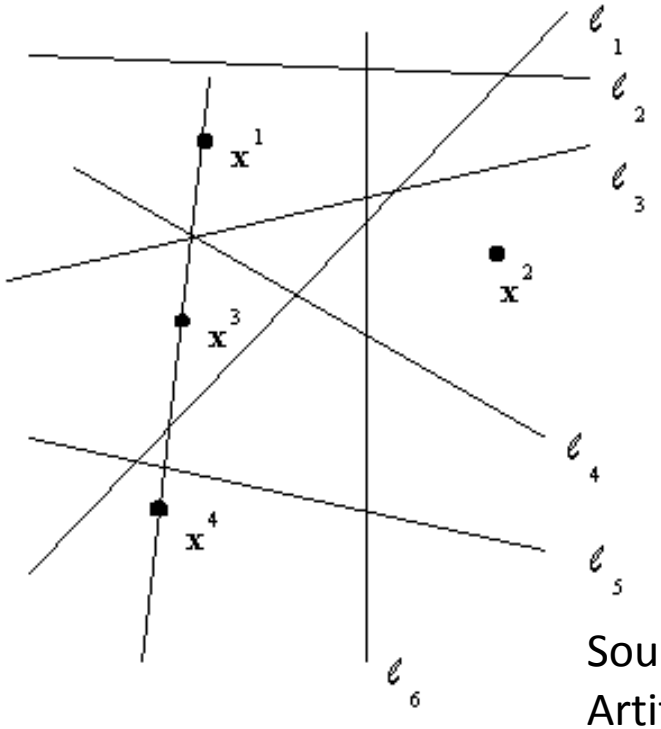
# Vapnik-Chervonenkis Dimension

**Definition 3.1** (VC Dimension [47]). The VC dimension $D_{VC}$ of a hypothesis space $f$ is the maximum integer $D = D_{VC}$ such that *some dataset* of cardinality $D$ can be shattered by $f$. Shattered by $f$ means that any arbitrary labeling can be represented by a hypothesis in $f$. If there is no maximum, it holds $D_{VC} = \infty$.

# General Position (from Linear Algebra)

**Definition 40.1** *A set of points* $\{\mathbf{x}_n\}$ *in* $K$*-dimensional space are in* general position *if any subset of size* $\leq K$ *is linearly independent, and no* $K+1$ *of them lie in a* $(K-1)$*-dimensional plane.*




Source: Mohamad H. Hassoun: Fundamentals of Artificial Neural Networks (MIT Press, 1995)

# How many points can we label in general?

Formula by Schlaefli (1852):

$$T(n,k) = T(n-1,k) + T(n-1,k-1), \quad (3)$$

where $T(n,1) = T(1,k) = 2$ or iteratively:

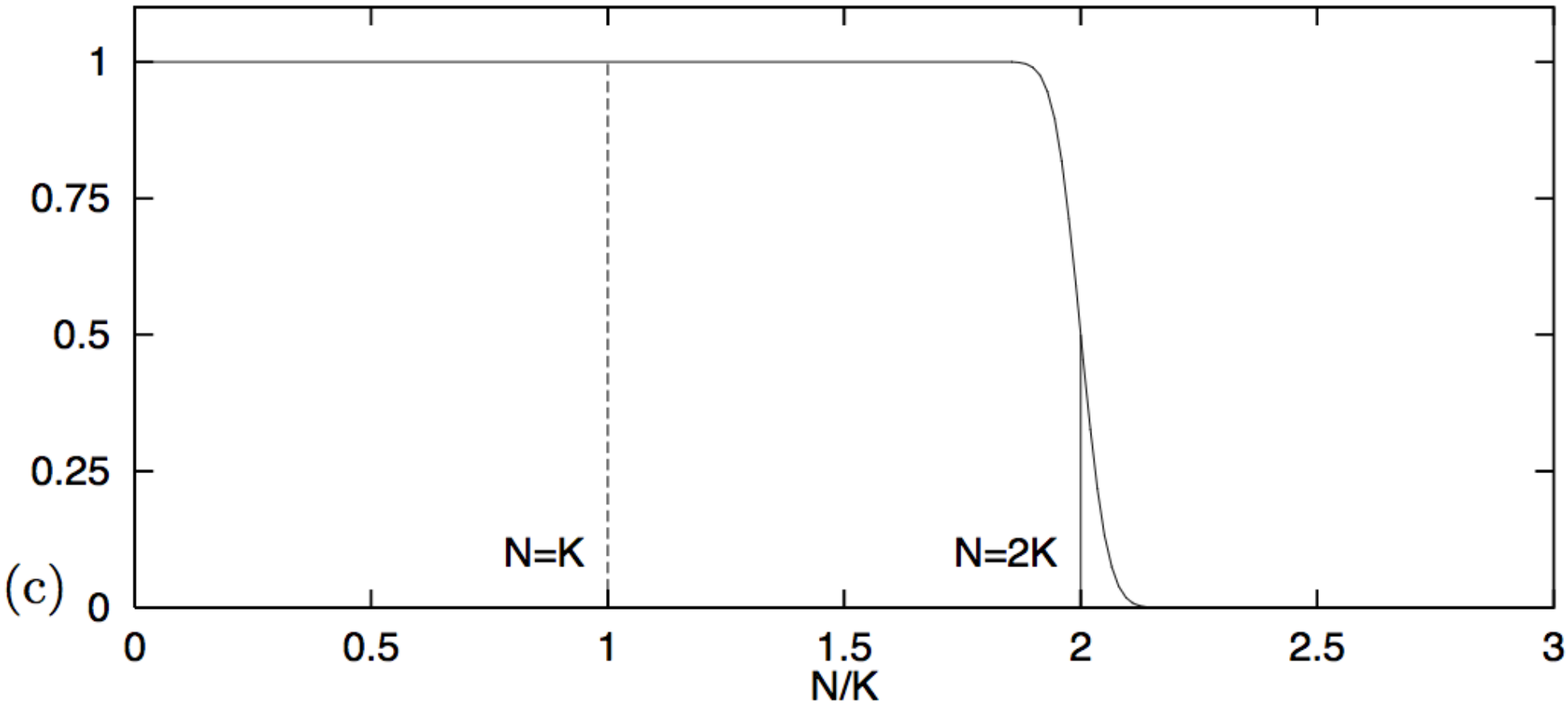
$$T(n,k) = 2\sum_{l=0}^{k-1} \binom{n-1}{l} \quad (4)$$

| $n\backslash k$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | **2** | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| 2 | 2 | **4** | 4 | 4 | 4 | 4 | 4 | 4 |
| 3 | 2 | 6 | **8** | 8 | 8 | 8 | 8 | 8 |
| 4 | 2 | 8 | 14 | **16** | 16 | 16 | 16 | 16 |
| 5 | 2 | 10 | 22 | 30 | **32** | 32 | 32 | 32 |
| 6 | 2 | 12 | 32 | 52 | 62 | **64** | 64 | 64 |
| 7 | 2 | 14 | 44 | 84 | 114 | 126 | **128** | 128 |
| 8 | 2 | 16 | 58 | 128 | 198 | 240 | 254 | **256** |

Table 1: Some values of the $T(n,k)$ function indicating the number of distinct threshold functions on $n$ points in general position in $k$ dimensions as defined by [22].

$$T(n,k) = 2^n \text{ for } k >= n.$$

# Critical points (1 Perceptron)




Source: D. MacKay: Information Theory, Inference and Learning

N=K: VC Dimension

# Generalizing from the Perceptron…

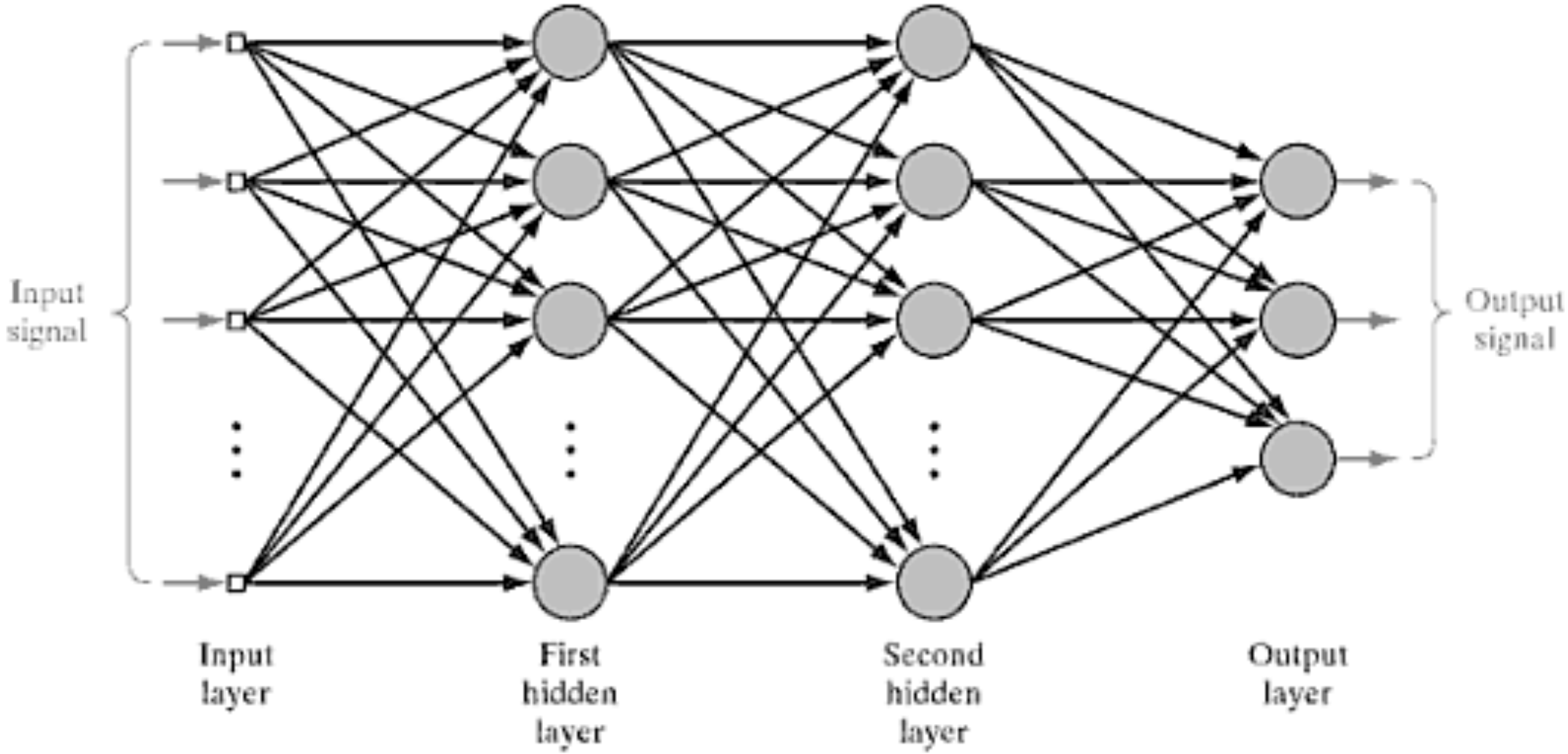
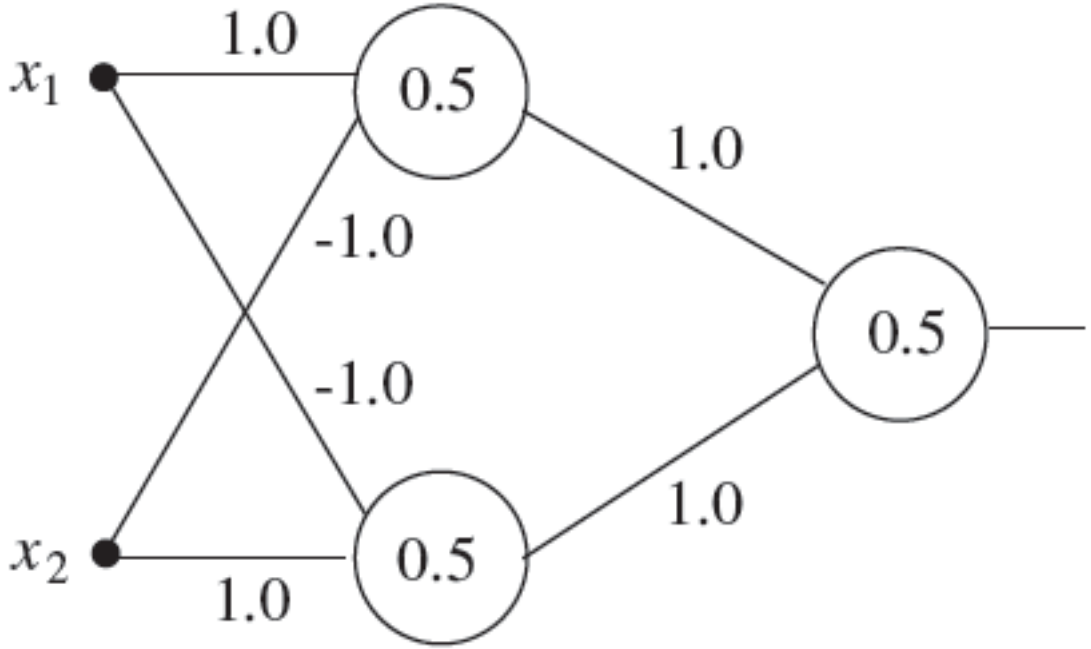


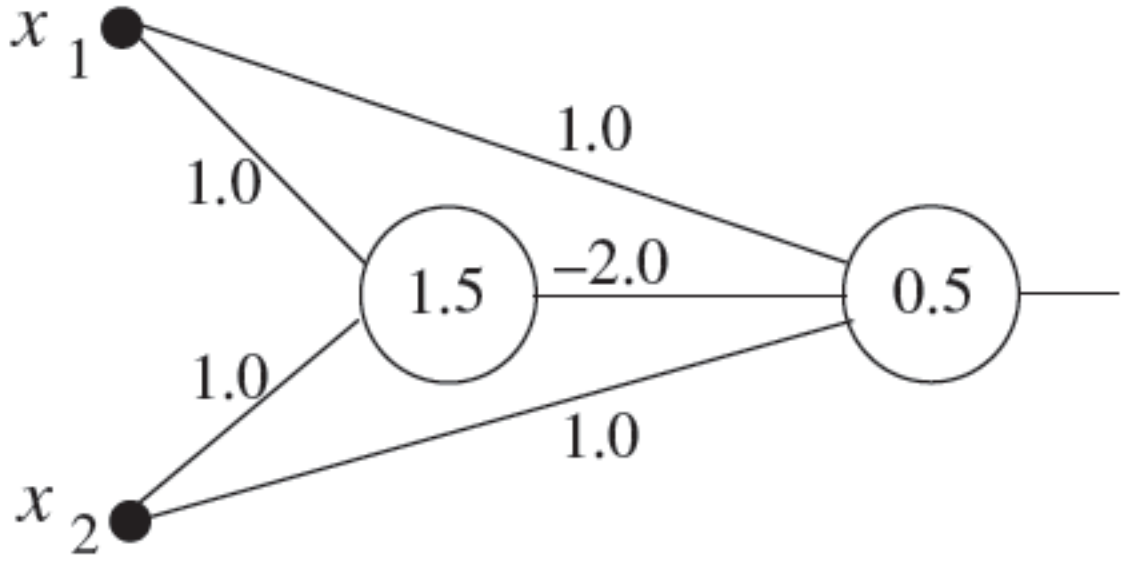

FIGURE 4.1 Architectural graph of a multilayer perceptron with two hidden layers.

Source: Wikipedia

# Example Solutions to XOR




Typical MLP



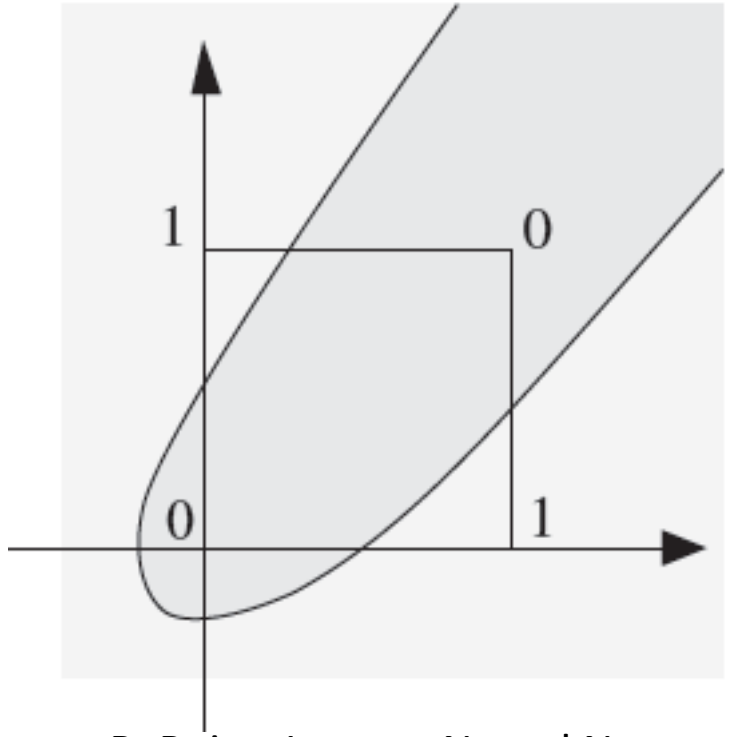

Shortcut

Source: R. Rojas, Intro to Neural Networks

# General Position (from Linear Algebra)

**Definition 40.1** *A set of points* $\{\mathbf{x}_n\}$ *in* $K$*-dimensional space are in* general position *if any subset of size* $\leq K$ *is linearly independent, and no* $K+1$ *of them lie in a* $(K-1)$*-dimensional plane.*

- Good enough for linear separation.
- Not enough for non-linear dependencies!
- pattern+noise != random (see whiteboard)



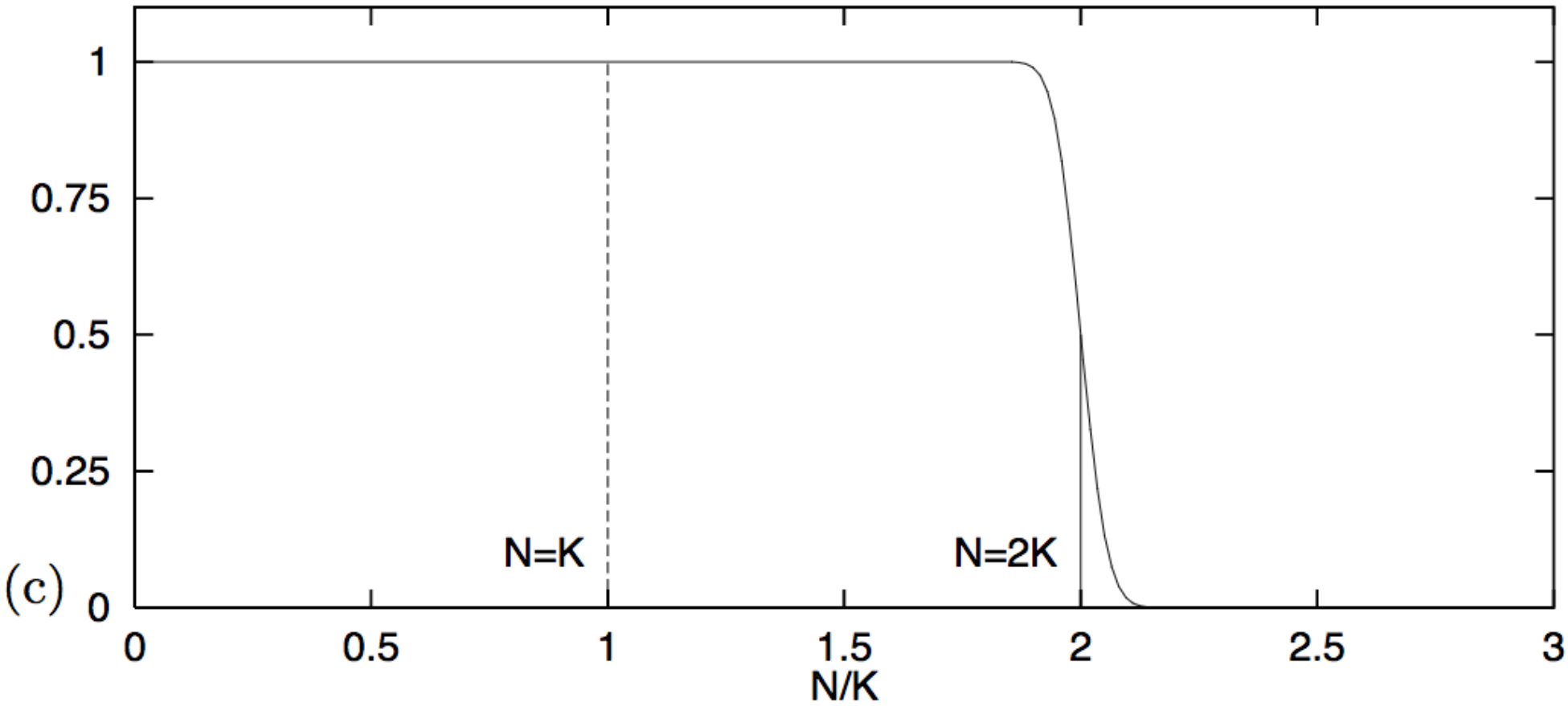

Source: R. Rojas, Intro to Neural Networks

# Random Position

**Definition 4.1** (Random Position)**.** A set of points $\{x_n\}$ in $K$-dimensional space is in random position, if and only if from any subset of size $< n$ it is not possible to infer anything about the positions of the remaining points.

- Random Position => General Position.
- Only valid distribution: Uniform distribution (see Gibbs, 1902)
- Best case learning: Memorization.

# Remember: 1 Perceptron = 2 Critical Points




Source: D. MacKay: Information Theory, Inference and Learning

N=K: LM Dimension

N=2K: MK Dimension

# Lossless Memory Dimension

**Definition 4.2** (Lossless Memory Dimension)**.** The lossless-memory dimension $D_{LM}$ is the maximum integer number $D_{LM}$ such that for any dataset with cardinality $n \leq D_{LM}$ and points in *random position*, *all possible labelings* of this dataset can be represented with a function in the hypothesis space.

- LM Dimension => VC Dimension
- Stricter Definition of VC Dimension with data constraint: “worst case VC dimension“

# 2nd Critical Point: MacKay Dimension

**Definition 4.3** (MacKay Dimension)**.** The MacKay dimension $D_{MK}$ is the maximum integer $D_{MK}$ such that for any dataset with cardinality $n \leq D_{MK}$ and points in random position *at least* 50% of all possible labelings of these datasets can be represented with a function in the hypothesis space [30].

- We will show: MKD = 2*LMD and exactly 50% of correct labelings for perceptron networks.

# Lossless Memory Dimension in Networks

**Just measure in bits!**

- The LM of **any binary classifier** cannot be better than the number of relevant bits in the model (pigeon hole principle, no universal lossless compression).
  This is: *n* bits in the model can *maximally* model *n* bits of data.

- Counting *relevant* bits in a Perceptron: See whiteboard.

# MacKay Dimension in Networks: Induction over T(n,k)

- For a single perceptron $T(n,k)=2^n$ for $n=k$. In other words, when the amount of weights equals the amount of points to label we are perfectly at LM dimension.
- In the best-case network, each weight therefore corresponds to a binary decision for each input point.
- Doubling the amount of points results in two points per individual weight. *T(2n,k) with n=k or T(2n,n)* for each perceptron.
  By induction: *T(2n,n)=0.5*T(2n,2n)* => MK Dimension is twice LM Dimension for each perceptron.
- It follows MK Dimension is twice LM Dimension for a best-case network.

# Result: Network Scaling Law

**Theorem 4.1** (Capacity scaling law)**.**

$$\sum_{j=1}^{l} C(P_j) = C\left(\sum_{j=1}^{l} P_j\right) \tag{6}$$

where $P_j$ is an arbitrary perceptron with $n_j$ inputs including a potential offset weight. The capacity is either $C = D_{MK}$ or $C = D_{LE}$ depending on the targeted phase. $\sum_{j=1}^{l} P_j$ denotes any arbitrary neural network that combines the respective perceptrons in a meaningful structure.

**Beware:**
**Architecture ignored!**

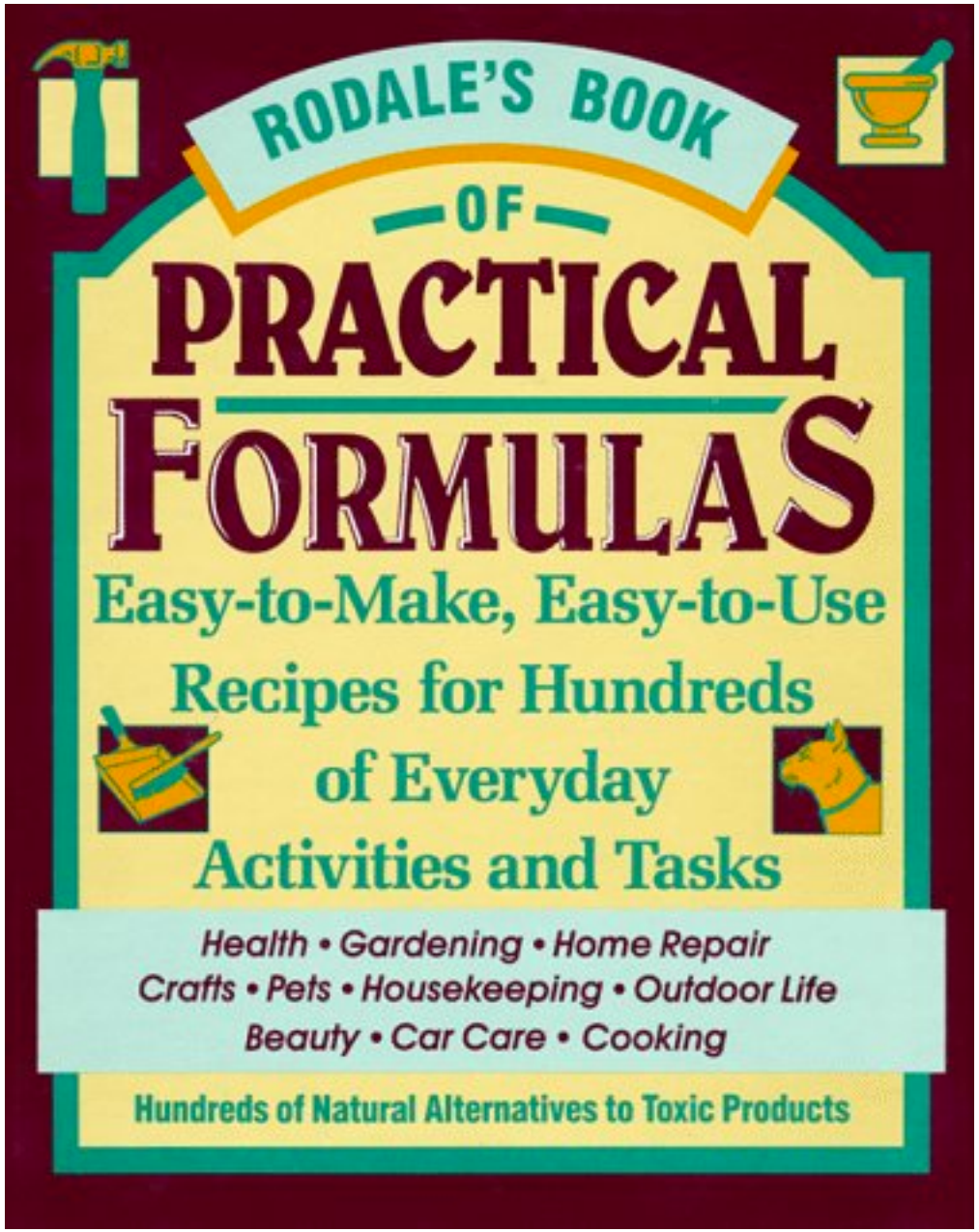
# Practical Formulas

- Capacity of a 3-Layer MLP

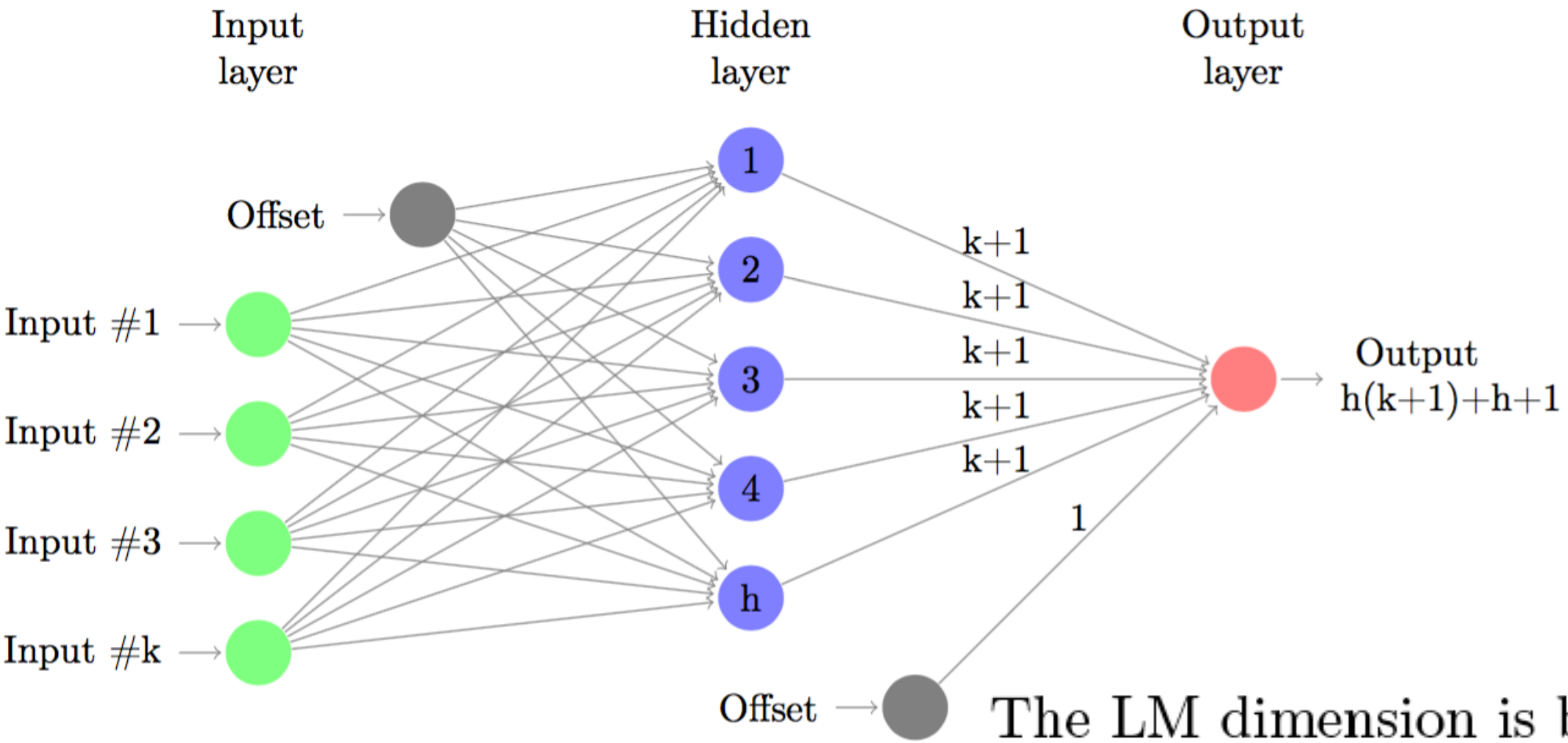
The LM dimension is bounded by

$$D_{LM} = h(k+1) + (h+1)$$

- Unit of measurement: Bits!



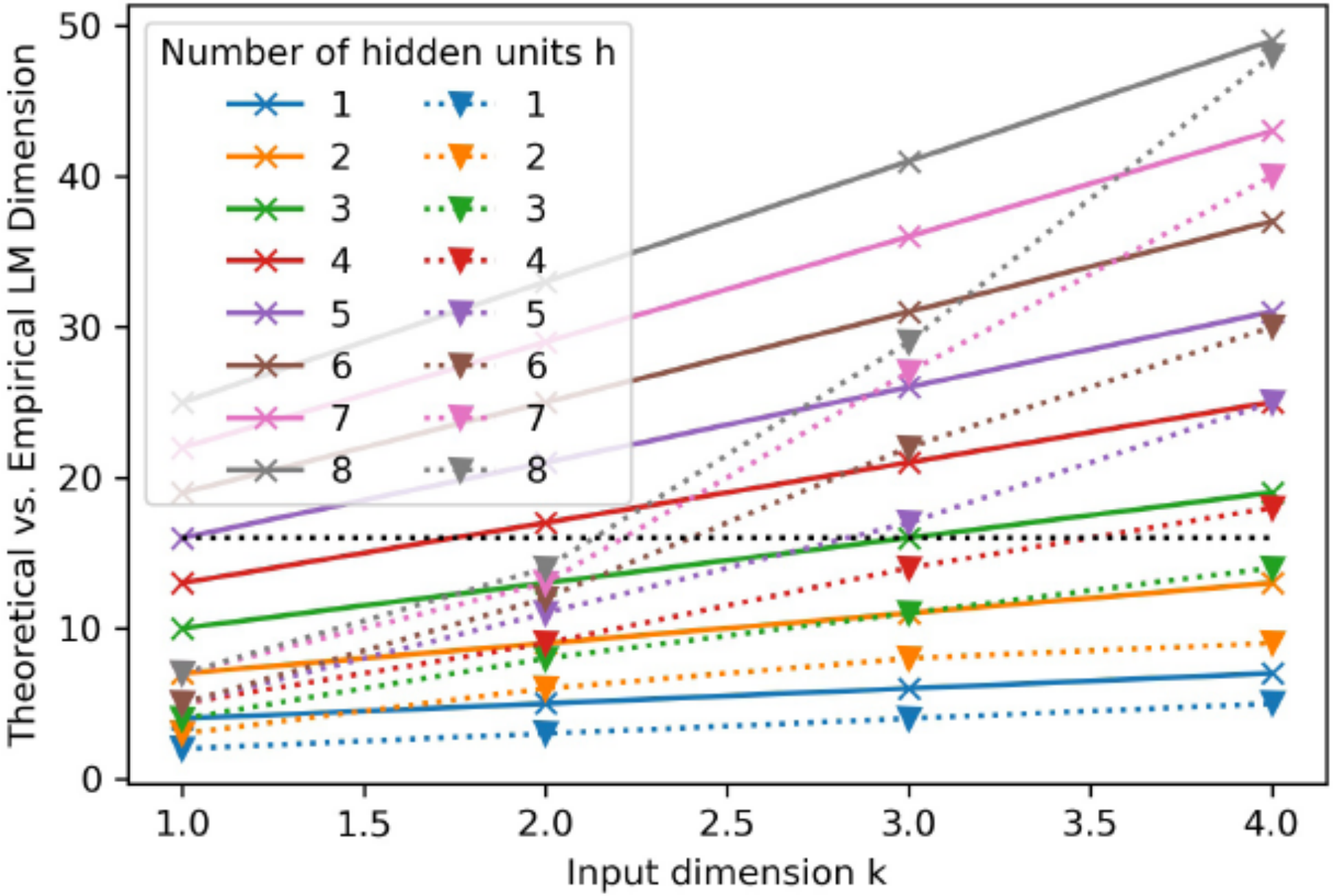
# Capacity Scaling Law: Illustration



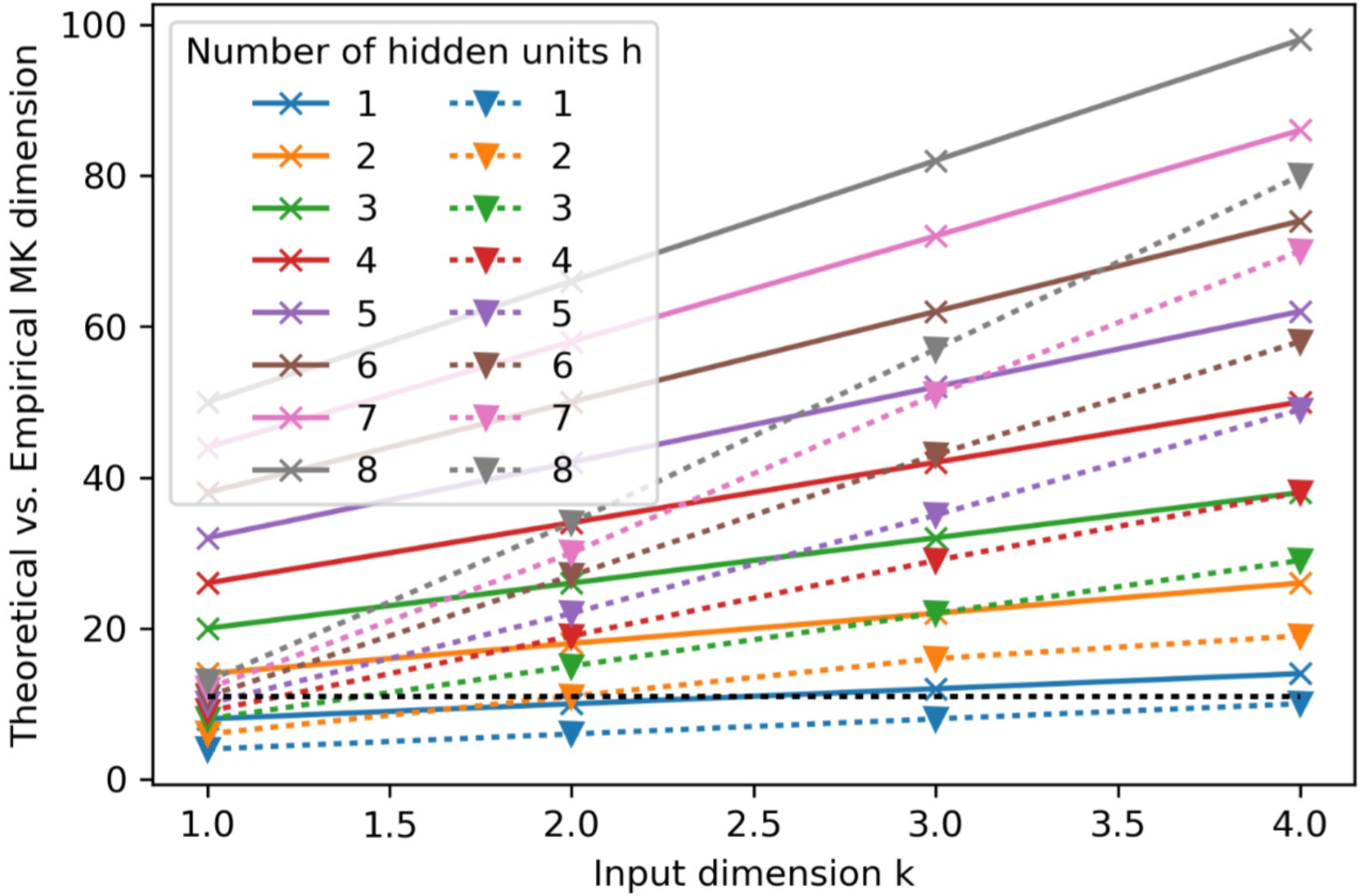

The LM dimension is bounded by

$$D_{LM} = h(k+1) + (h+1)$$

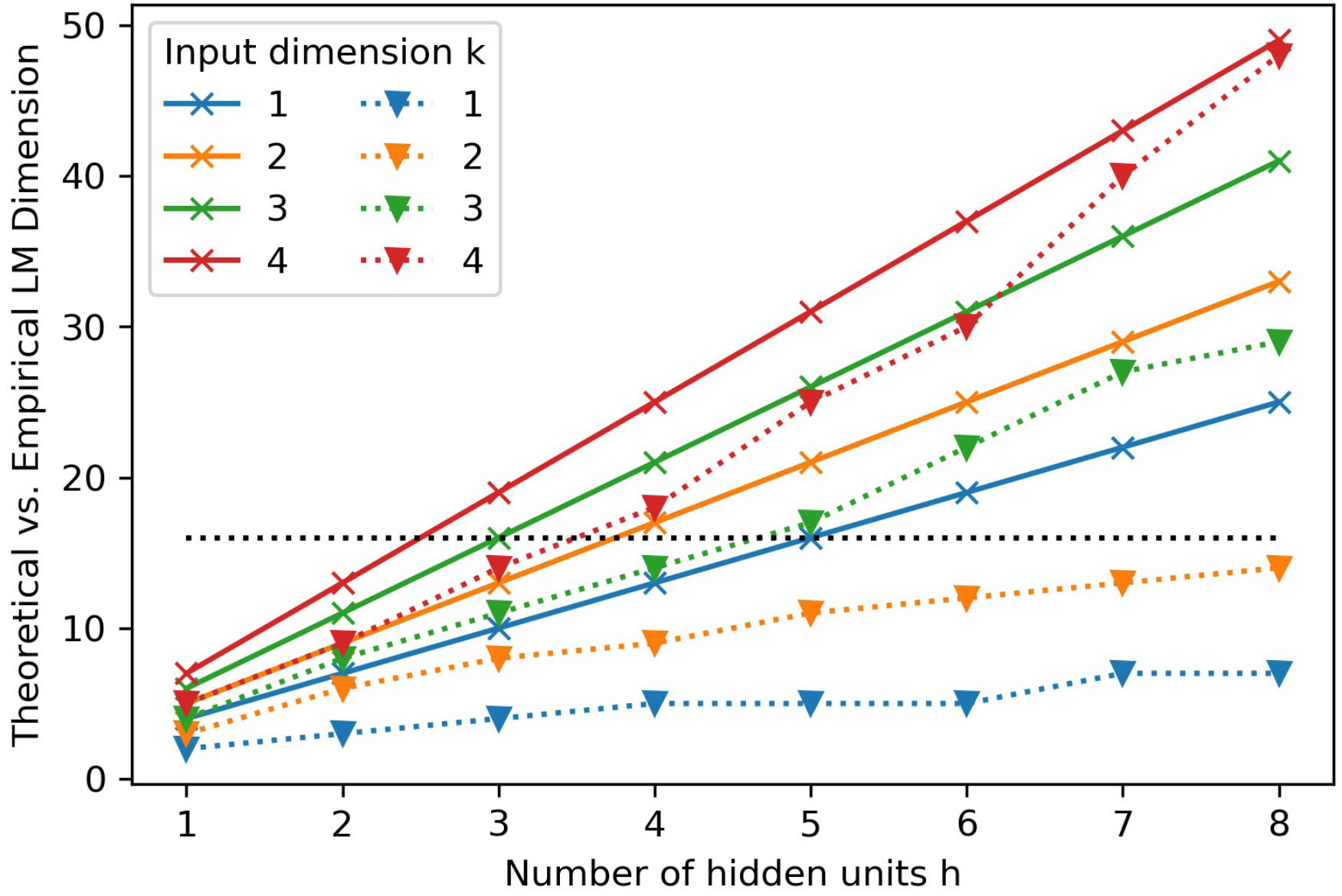
# Experimental Validation: LMD vs Input Dimension



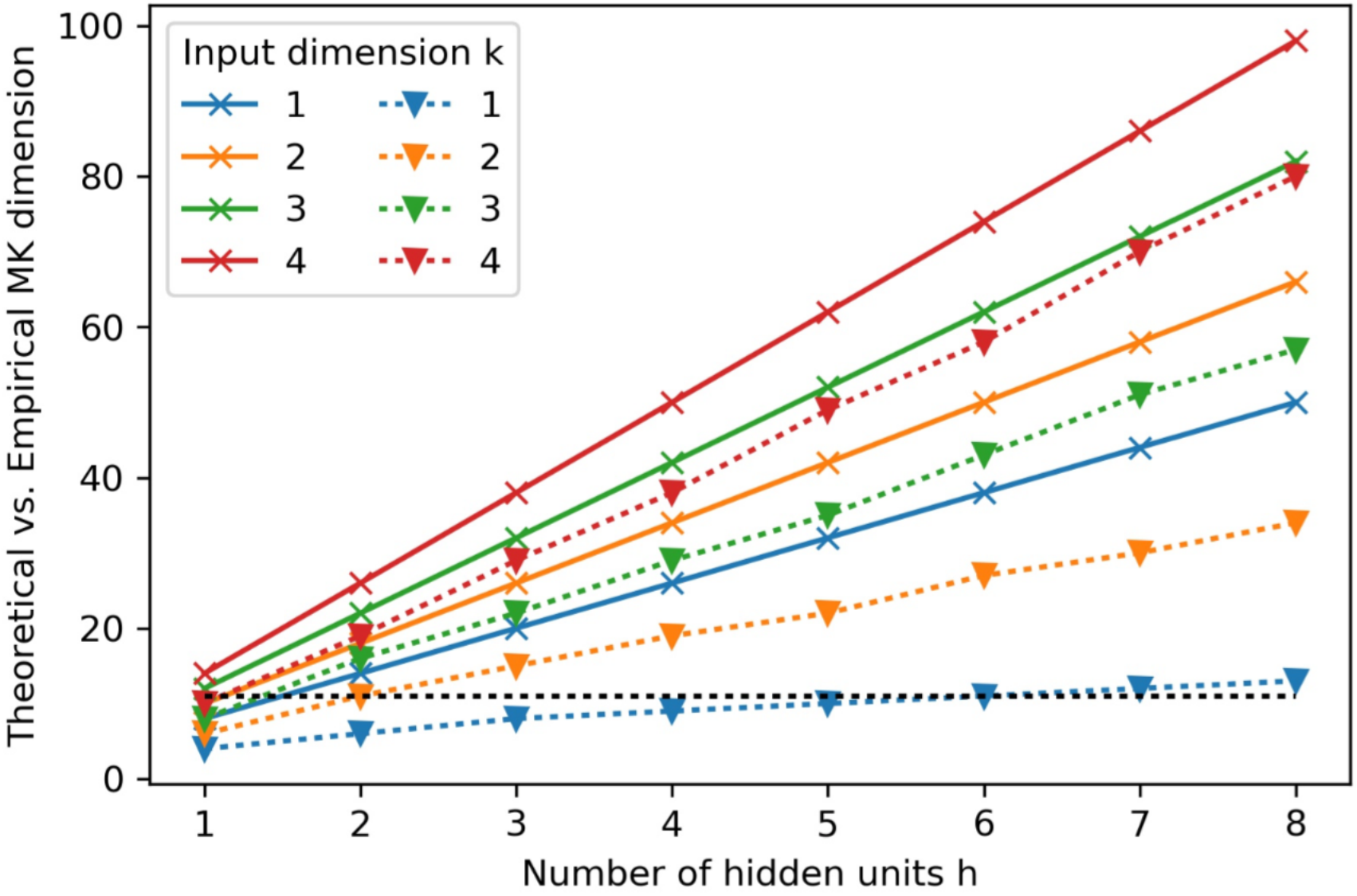
# Experimental Validation: MKD vs Input D



# Experimental Validation: LMD/Hidden Units



# Experimental Validation: MKD/Hidden Units



# Conclusion: Theory Part

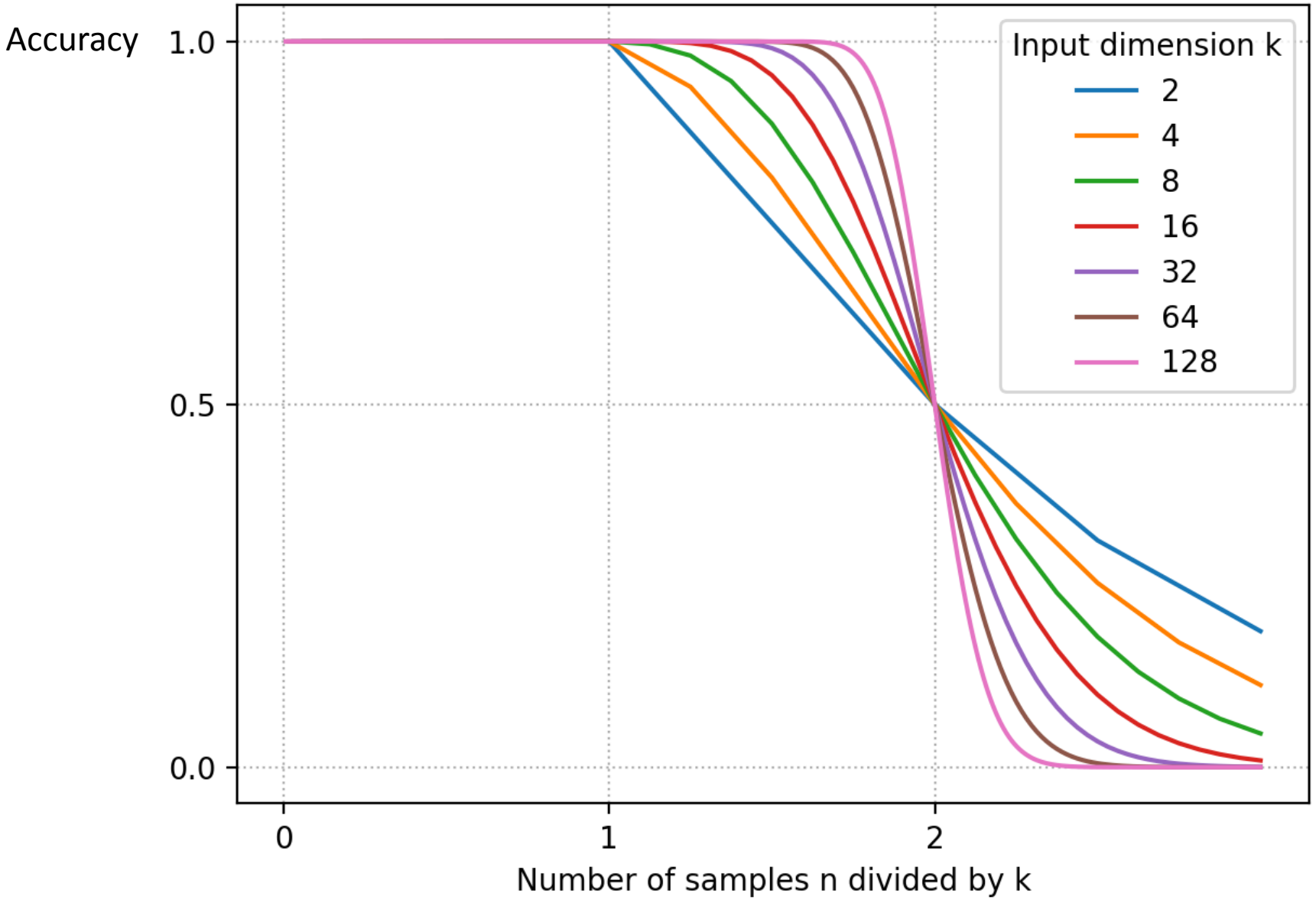
- Neural Networks can be explained as storing a function *f(data)->labels* which requires a certain amount of bits.
- Two critical points (phase transitions) for chaotic position can be scaled linearly
- Code in paper: Repeat our experiments!

http://arxiv.org/abs/1708.06019

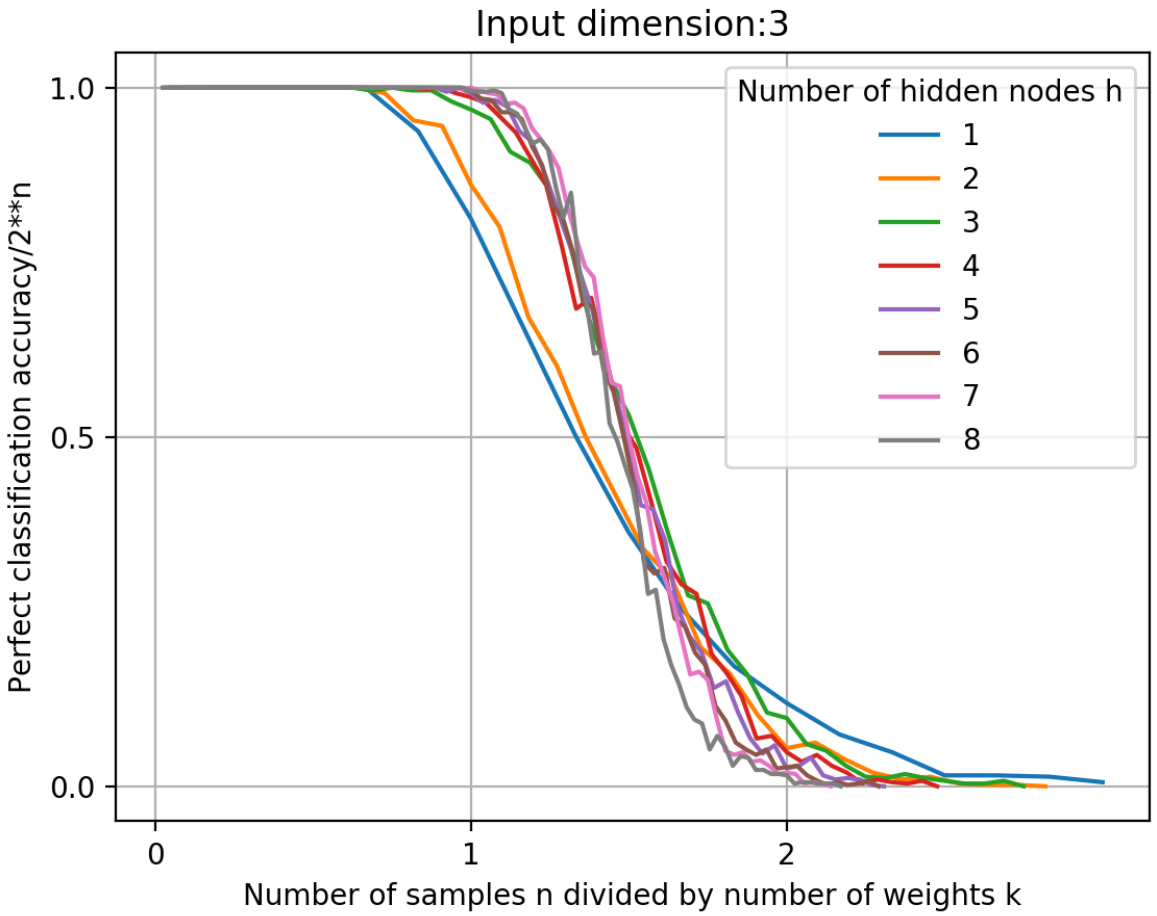
# Practical Implications

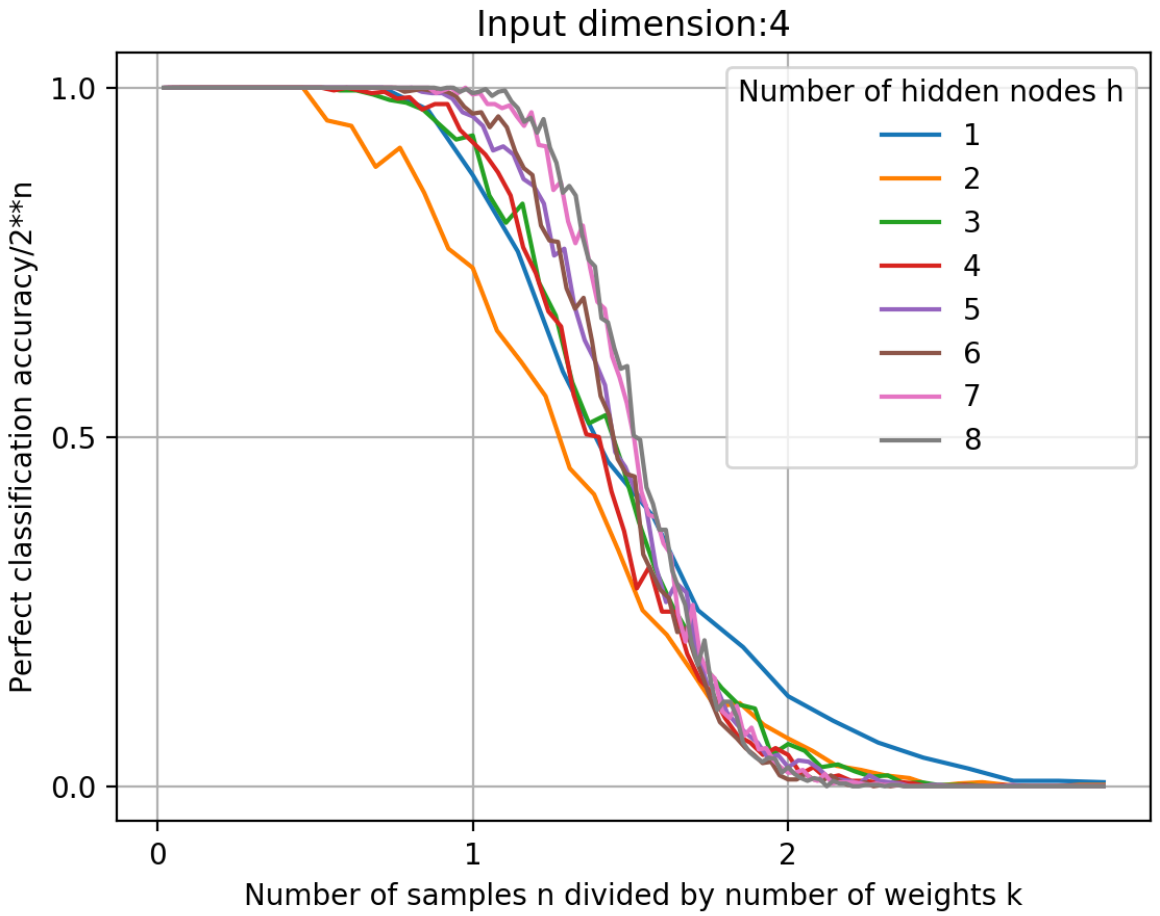
- Upper limit allows for data and task-independent evaluation of
  - Learning algorithms (convergence, efficiency, etc.)
  - Neural Architectures (deep vs. shallow, dropout, etc.)
  - Comparison of networks
  - Estimation of parameters needed for a given dataset
- Idea generalizes to **any** supervised machine learner!

# „Characteristic Curve“ of Neural Network: Theory



# „Characteristic Curve“ of Neural Network: Actual



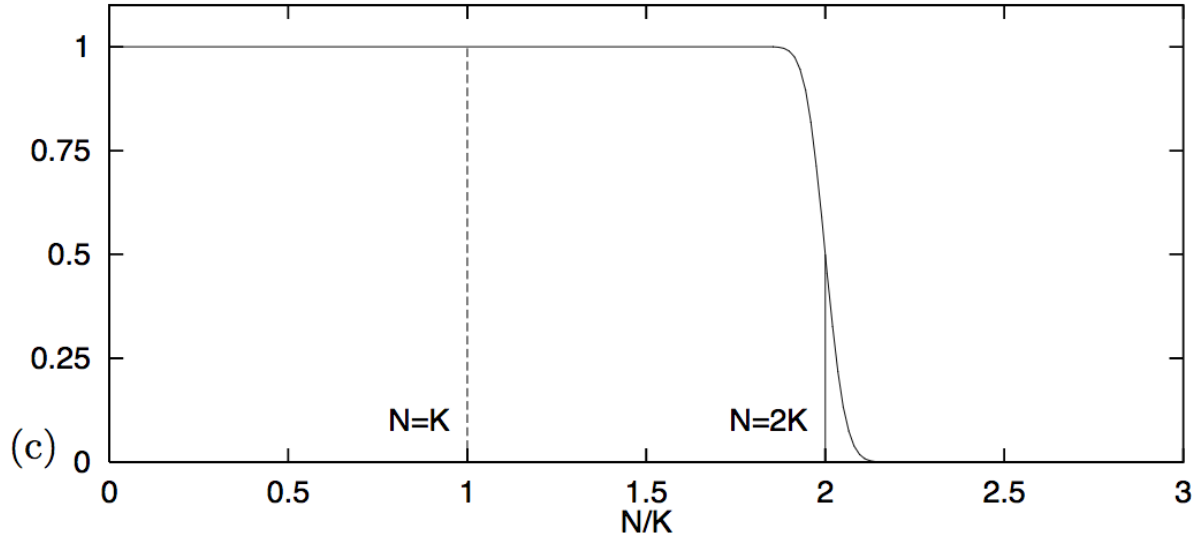

Python scikit-learn, 3-Layer MLP

# Does my function *f(data)->labels* generalize?

- Universally: No.  (Can you predict coin tosses after learning some?)
- In practice: If you learn enough samples from a probability density function (PDF), you maybe able to model it. This is: If your test samples come from the same PDF and it's not flat, you can predict.
- The rules that govern this prediction are investigated in the field of information theory.

# Future Work

- What about more complex activation functions? (RBF, Fuzzy, etc.?) Recursive networks? Convolutional Networks?
- Adversarial examples are connected to capacity!
- Curve looks familiar:
  Exists in EE, chemistry, physics!




Source: D. MacKay: Information Theory, Inference and Learning

# Acknowledgements


- Raul Rojas and Jerry Feldman!
- Bhiksha Raj, Naftali Tishby, Alfredo Metere, Kannan Ramchandran, Jan Hendrik Metzen, Jaeyoung Choi, Friedrich Sommer and Andrew Feit and many others for feedback.
- These slides contain materials from D. MacKay's and Raul Rojas' books. Go buy them! :-)