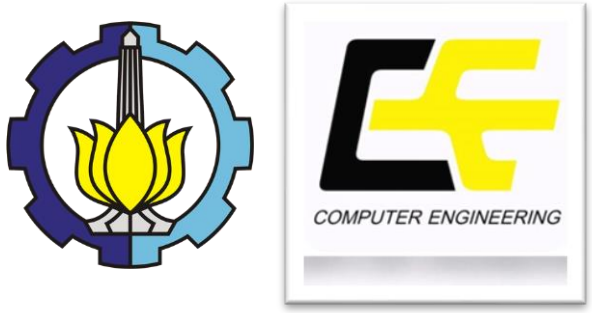
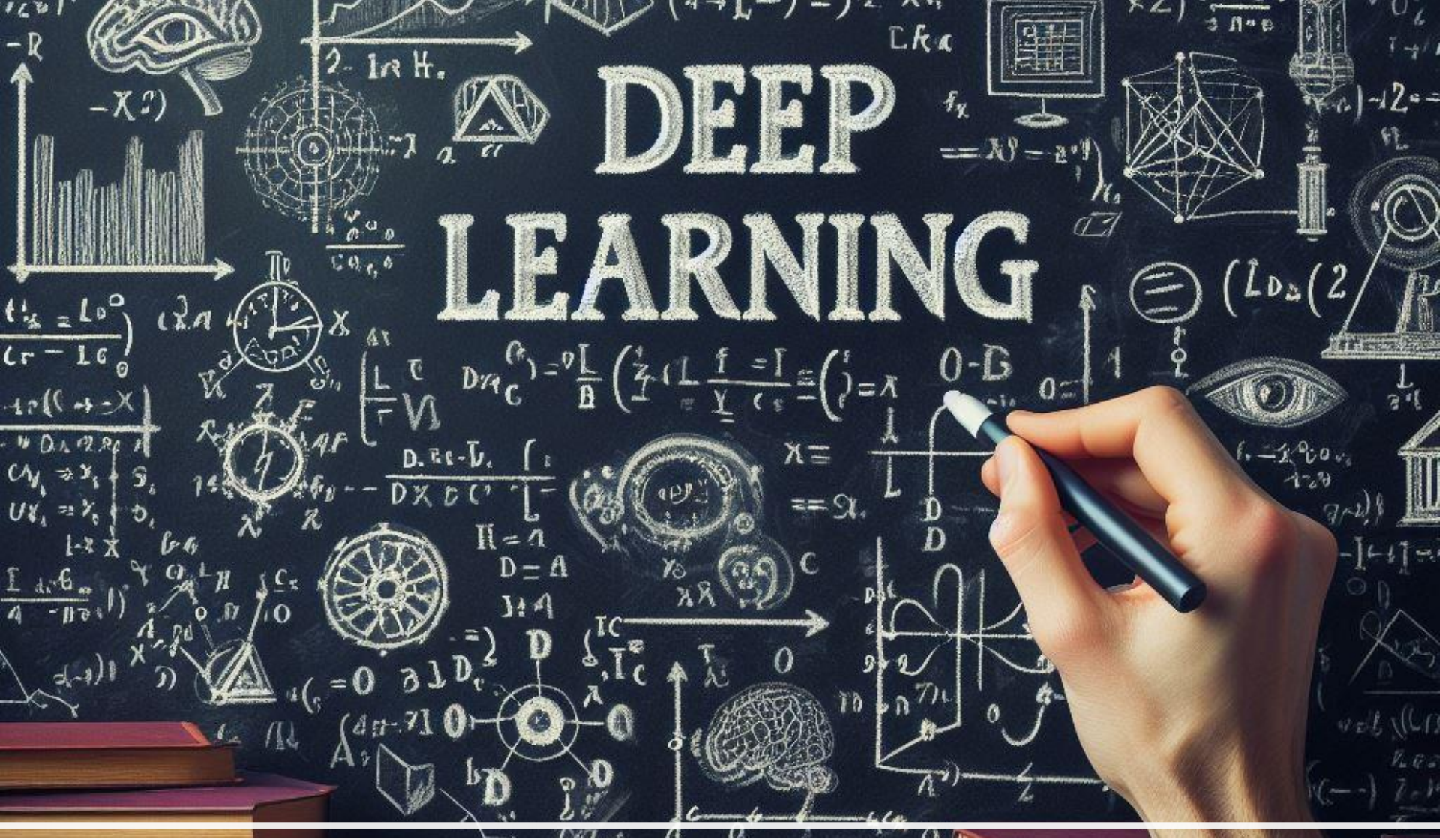
Intro – DL for Multimedia

Reza Fuad Rachmadi





Today: **Basic Deep Learning**

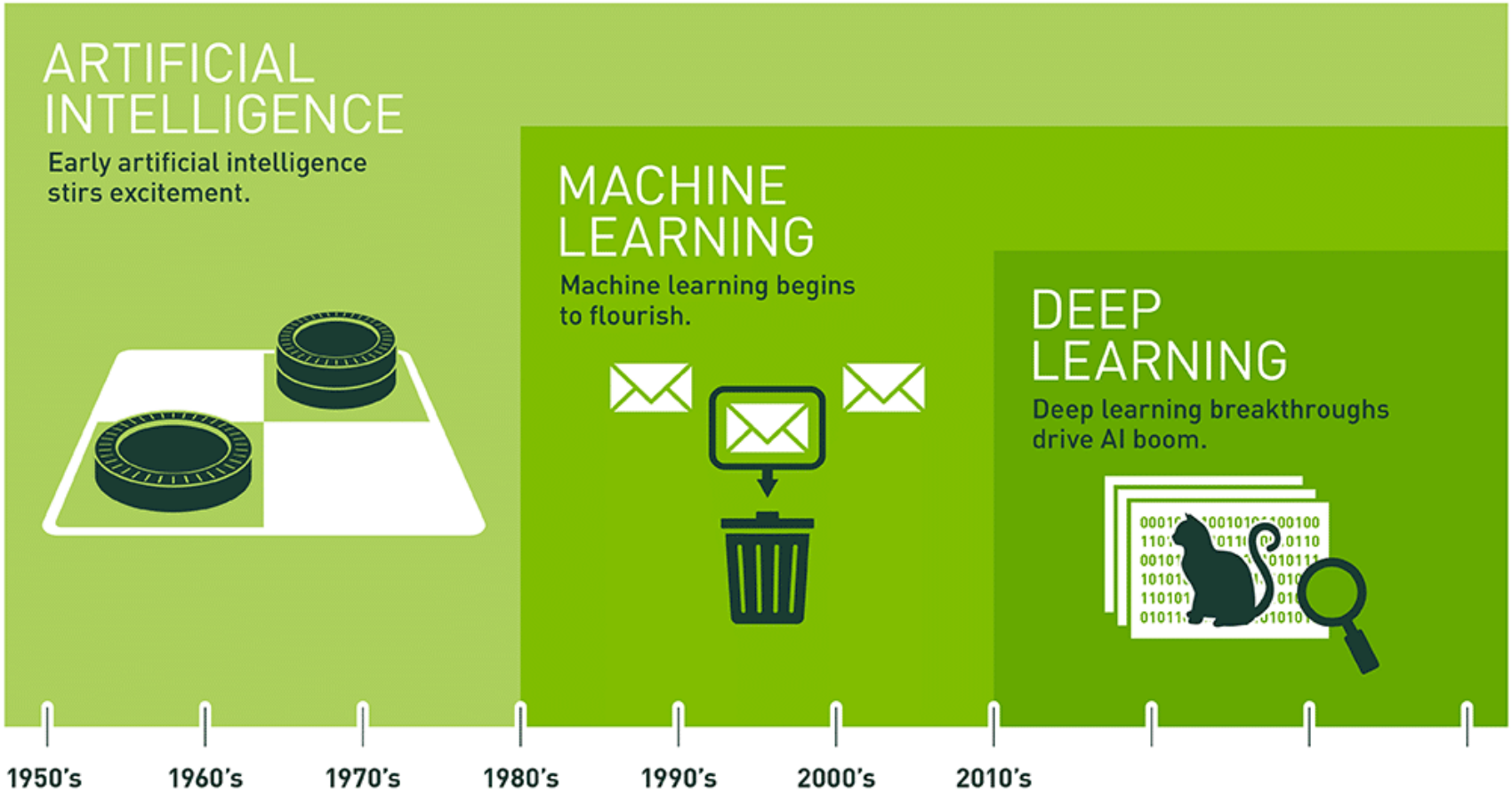
Outline

- What is deep learning?
- Deep learning application
- Simple neural network classifier
- Deep learning classifier

Outline

- **What is deep learning?**
- Deep learning application
- Simple neural network classifier
- Deep learning classifier

Basic concept



Source: NVIDIA

What is deep learning

- From Wikipedia:
  - **Deep learning** is the subset of machine learning methods based on artificial neural networks with representation learning. The adjective "deep" refers to the use of multiple layers in the network. Methods used can be either supervised, semi-supervised or unsupervised.
- From IBM:
  - **Deep learning** is a subset of machine learning that uses neural networks with three or more layers to simulate the behavior of the human brain, allowing it to learn from large amounts of data.

What is deep learning

- From MathWorks:
  - **Deep learning** is a machine learning technique that teaches computers to learn by example. It uses neural networks with many layers to perform classification tasks from images, text, or sound.
- In summary:
  - **Deep learning** is a branch of machine learning that uses multi-layered neural networks to learn from data and perform tasks. It can handle structured and unstructured data, and it automates feature extraction, which reduces the need for human expertise.

What is deep learning

- In brief:
  - Hierarchical representation
    - Cascade of non-linear transformation
    - Multiple layers of representations
  - End-to-End Learning
    - Learning representations
    - Learning to extract features

What is deep learning

- Example of hierarchical representation

Vision

pixels → edge → texton → motif → part → object

NLP

character → word → NP/VP/.. → clause → sentence → story

General

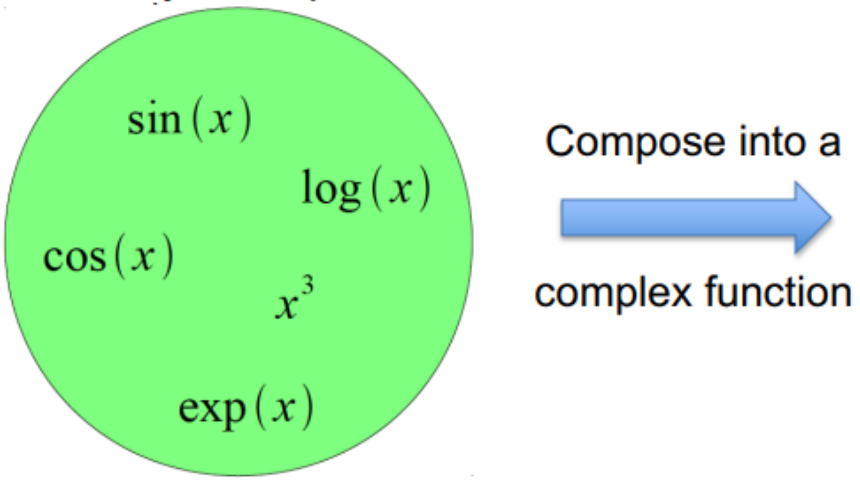
simple function → complex function

Slide Credit: Marc'Aurelio Ranzato, Yann LeCun

What is deep learning

- Building a complicated function

Given a library of simple functions

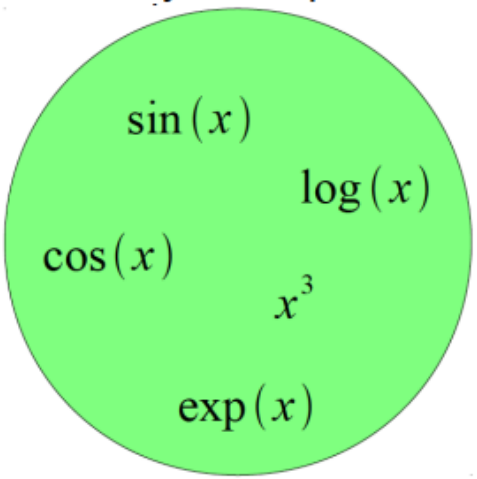


Slide Credit: Marc'Aurelio Ranzato, Yann LeCun

What is deep learning

- Building a complicated function

Given a library of simple functions



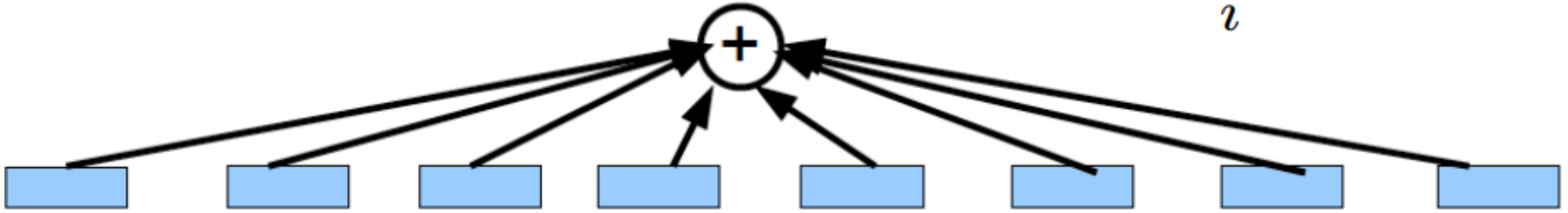
Compose into a

complex function

Idea 1: Linear Combinations

- Boosting
- Kernels
- …

$$f(x) = \sum_i \alpha_i g_i(x)$$

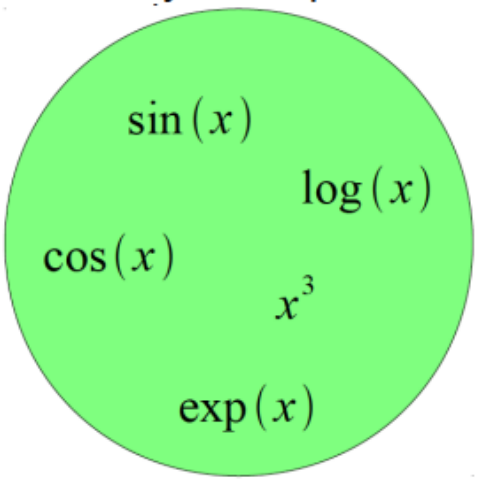


Slide Credit: Marc'Aurelio Ranzato, Yann LeCun

What is deep learning

- Building a complicated function

Given a library of simple functions



Compose into a

complex function

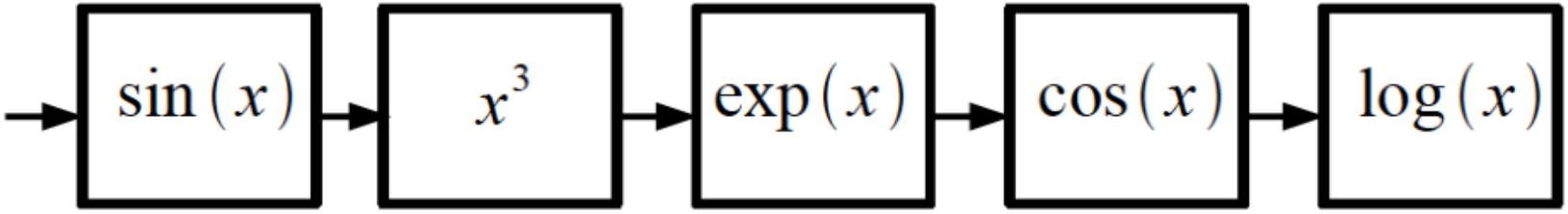
Idea 2: Compositions

Compose a set of functions (layers) through which the input data get transformed.

More layers = "Deeper"

Can we make it more expressive?

$$f(x) = \log(\cos(\exp(\sin^3(x))))$$

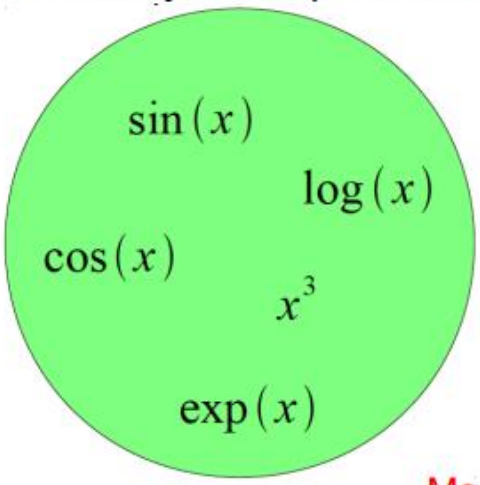


Slide Credit: Marc'Aurelio Ranzato, Yann LeCun

What is deep learning

- Building a complicated function

Given a library of simple functions



Compose into a

complex function

Idea 2: Compositions

Compose a set of functions (layers) through which the input data get transformed.

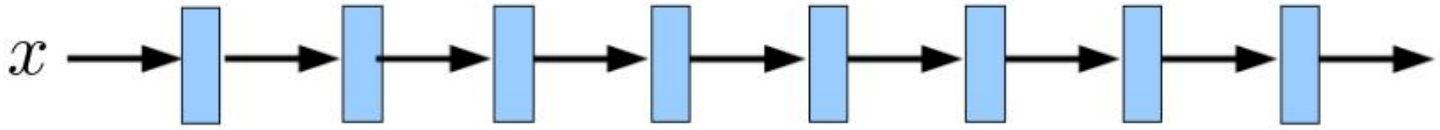
More layers = "Deeper"

Yes! Parametric functions

Modern DNNs have huge # of parameters, on the orders of bn's

Parametric functions

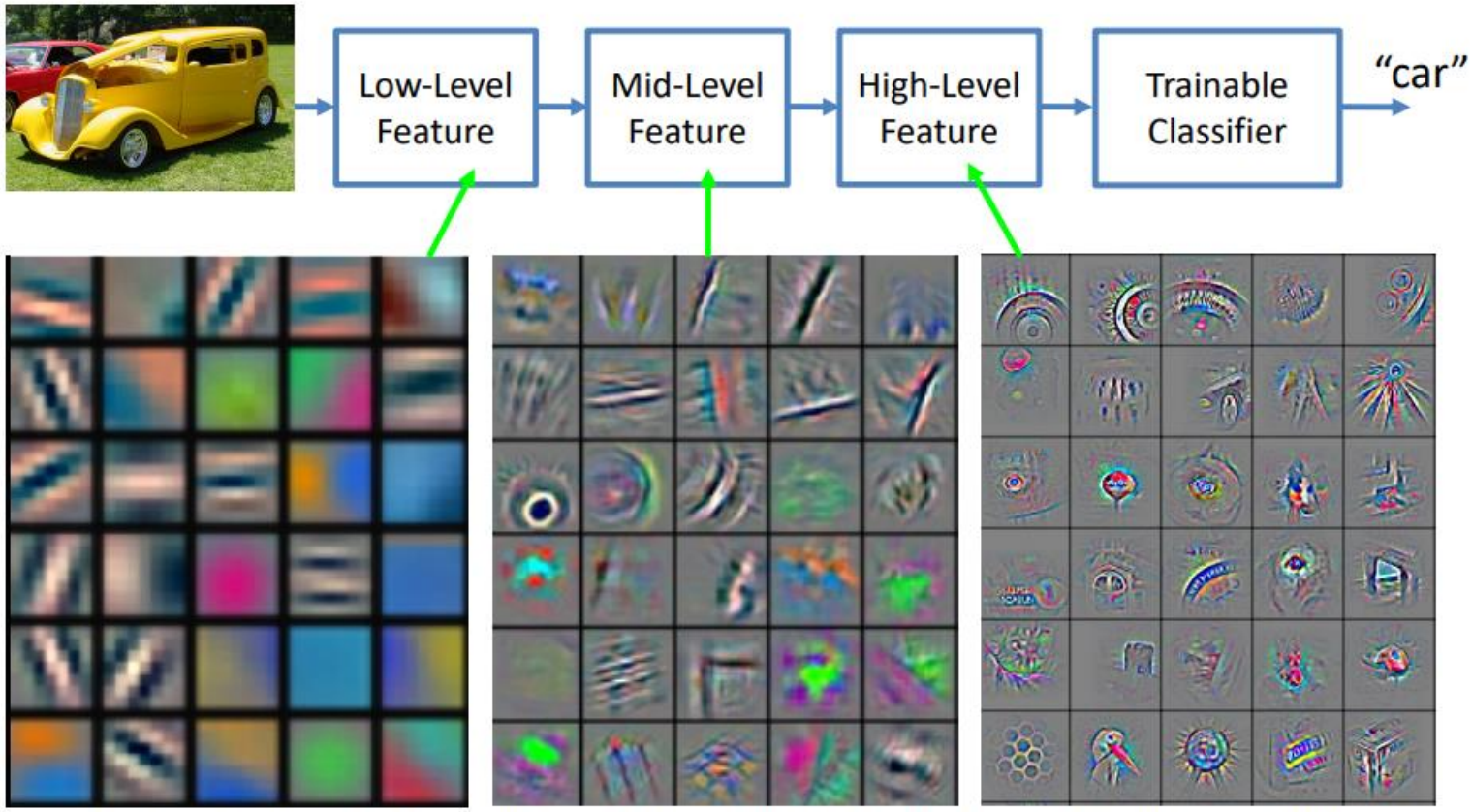
$$f_\theta(x) = g_{\theta_n}(\ldots g_{\theta_2}(g_{\theta_1}(x)\ldots))$$



Slide Credit: Marc'Aurelio Ranzato, Yann LeCun

What is deep learning

- Deep Learning = Hierarchical Compositionality

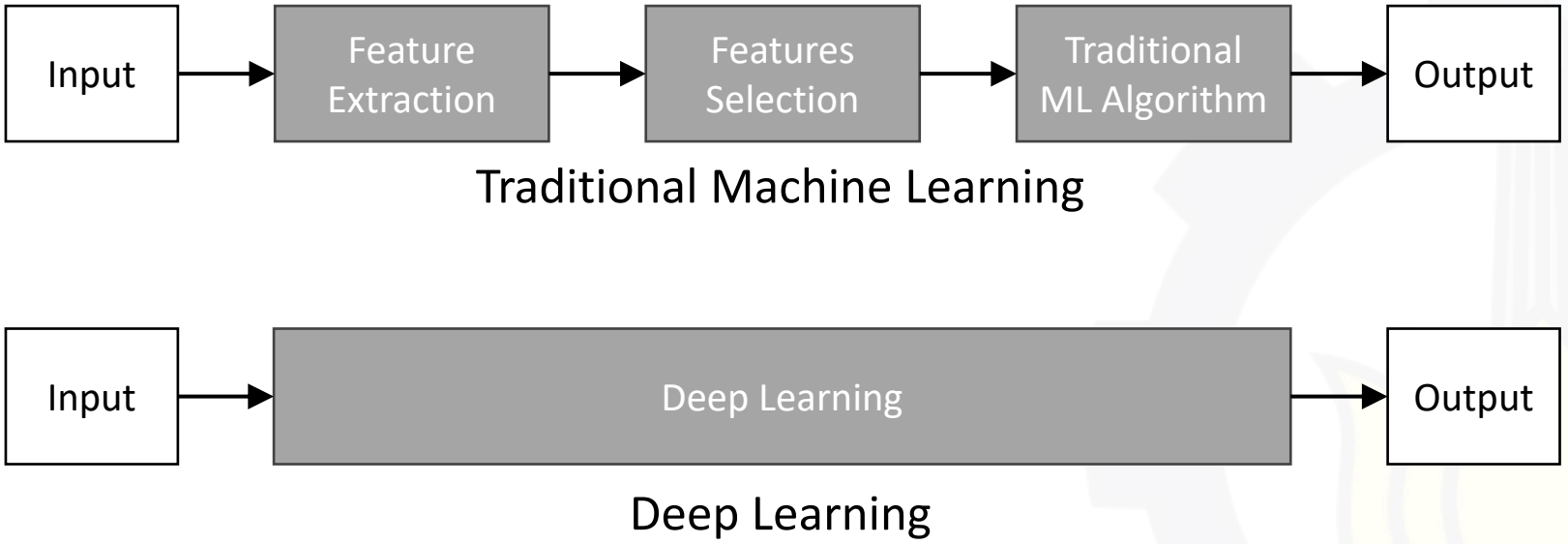


Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Slide Credit: Marc'Aurelio Ranzato, Yann LeCun

What is deep learning

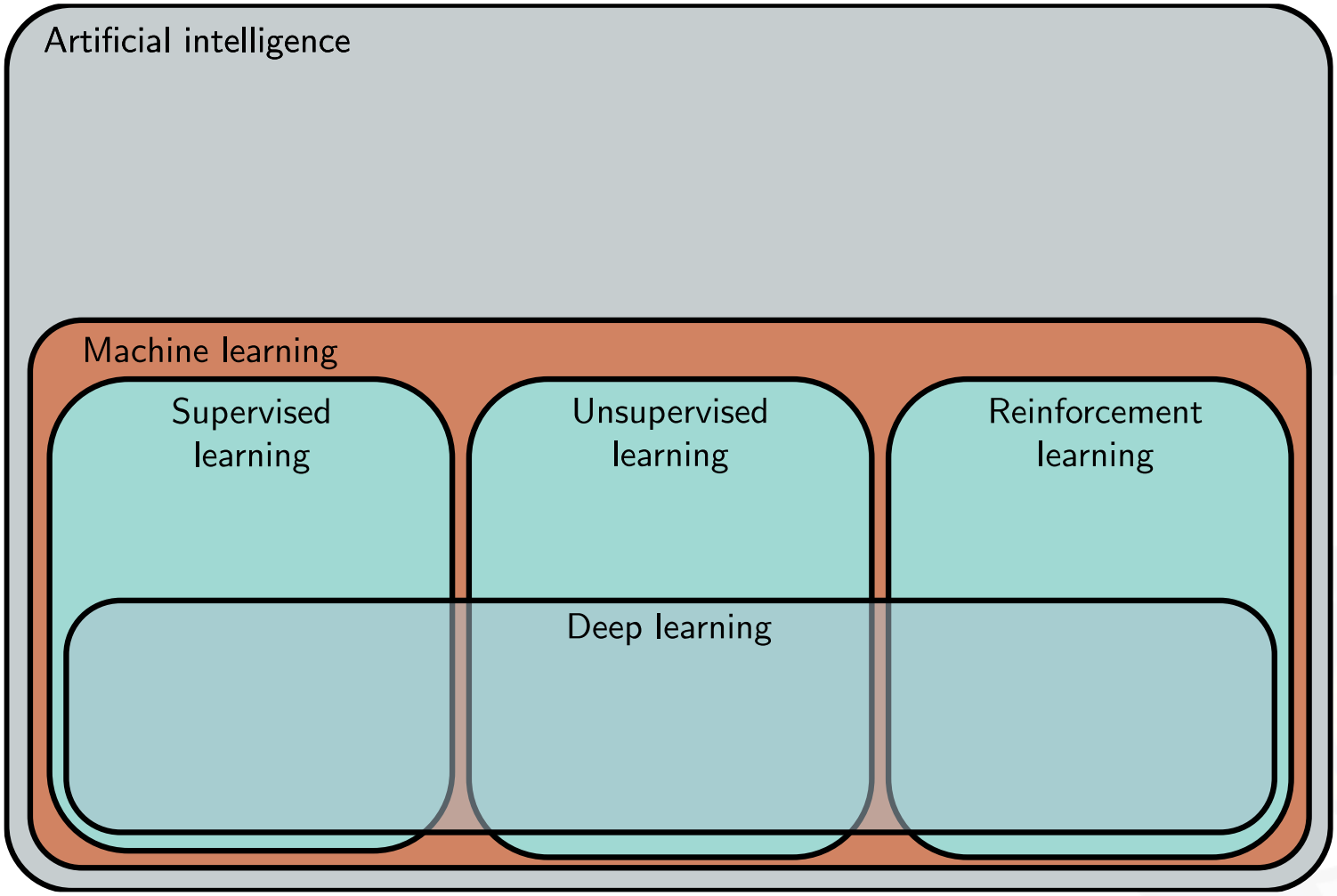
- End-to-End Learning



Outline

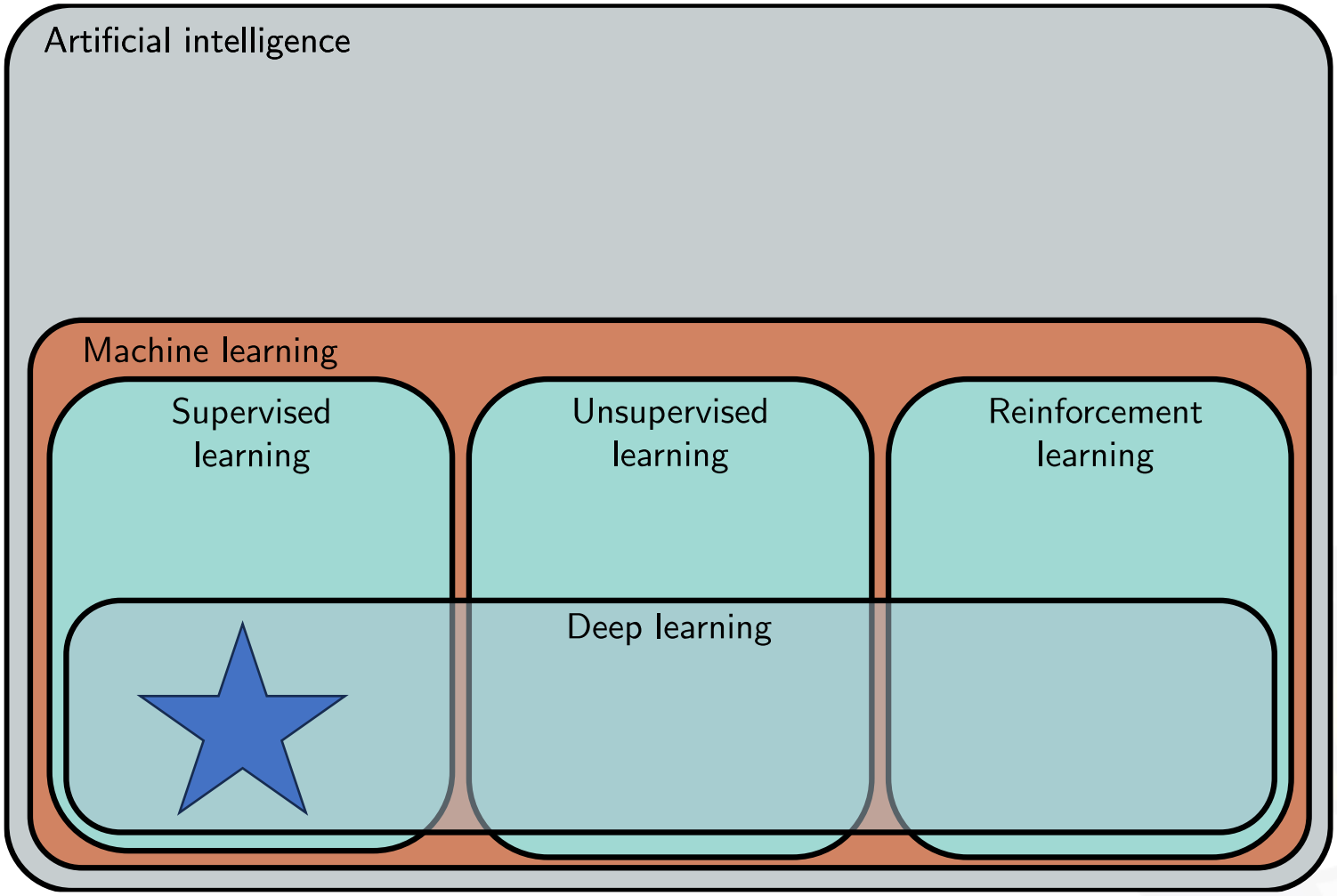
- What is deep learning?
- **Deep learning application**
- Simple deep learning classifier
- Detail of deep learning classifier

Deep Learning Position (Recap)



Slide Credit: Simon Prince

Supervised Learning



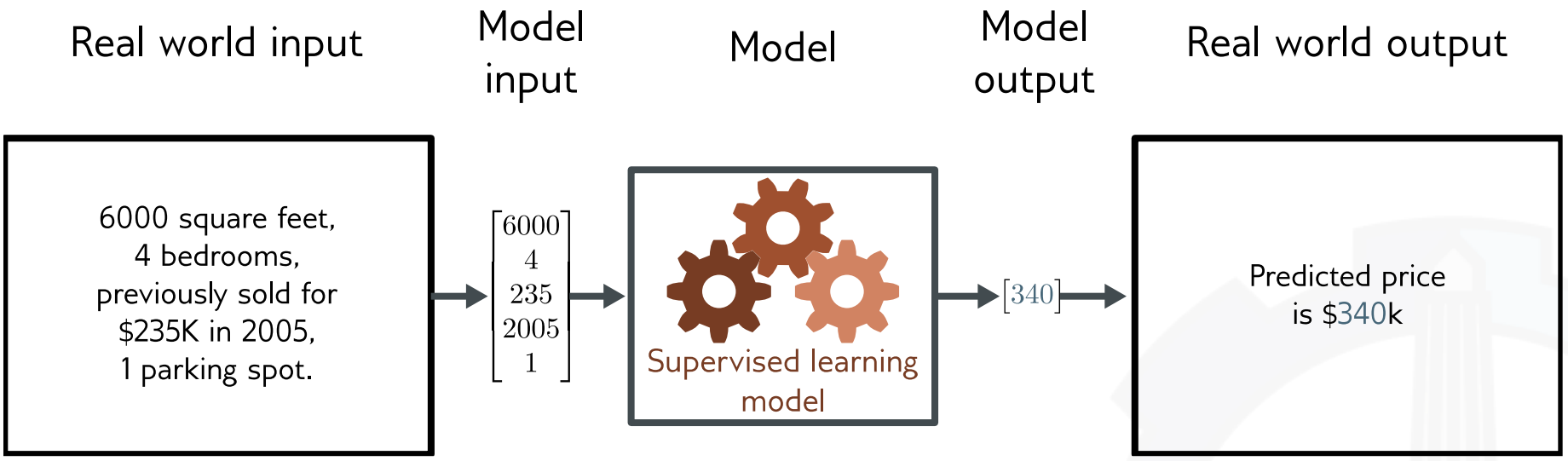
Slide Credit: Simon Prince

Supervised Learning

- Define a mapping from input to output
- Learn this mapping from paired input/output data examples

Slide Credit: Simon Prince

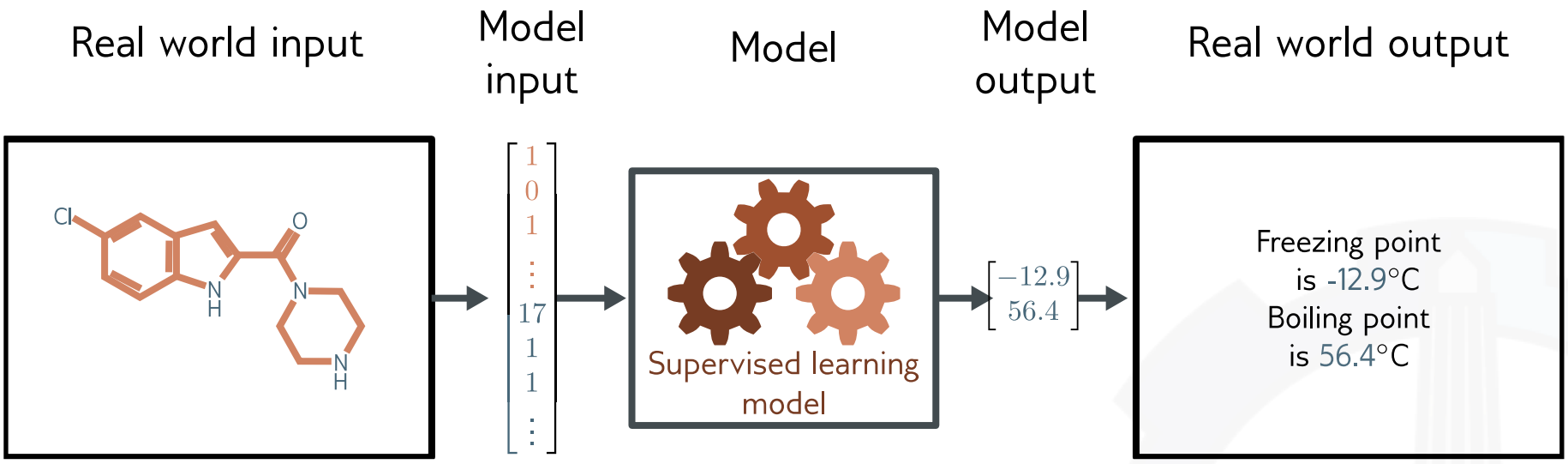
Supervised Learning (Regression)



- Univariate regression problem (one output, real value)
- Fully connected network

Slide Credit: Simon Prince

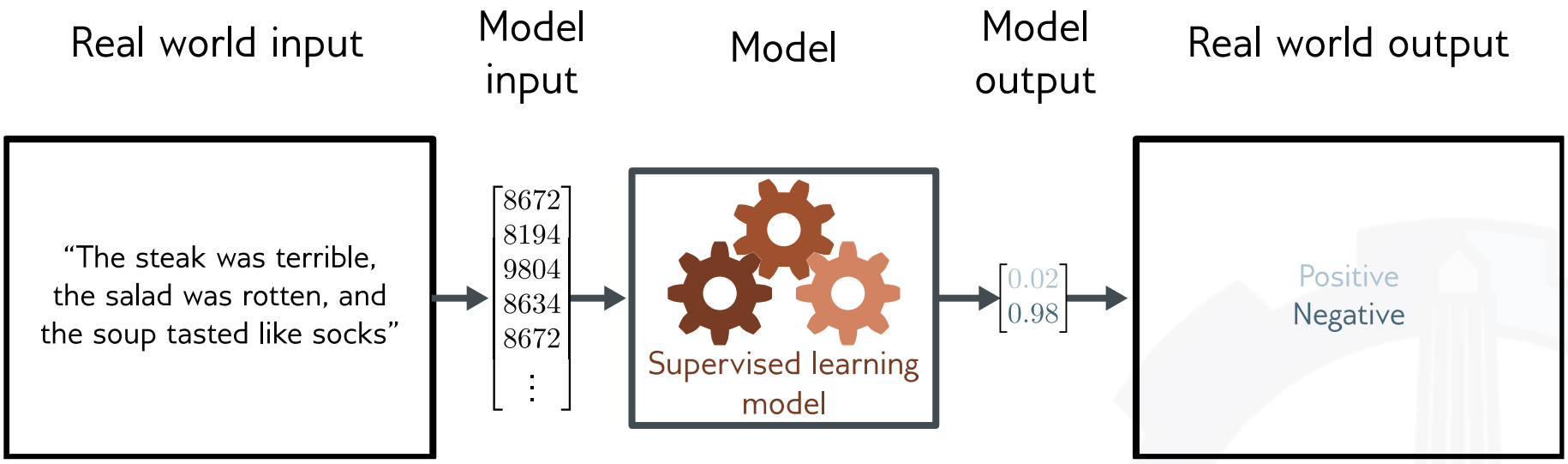
Supervised Learning (Graph Regression)



- Multivariate regression problem (>1 output, real value)
- Graph neural network

Slide Credit: Simon Prince

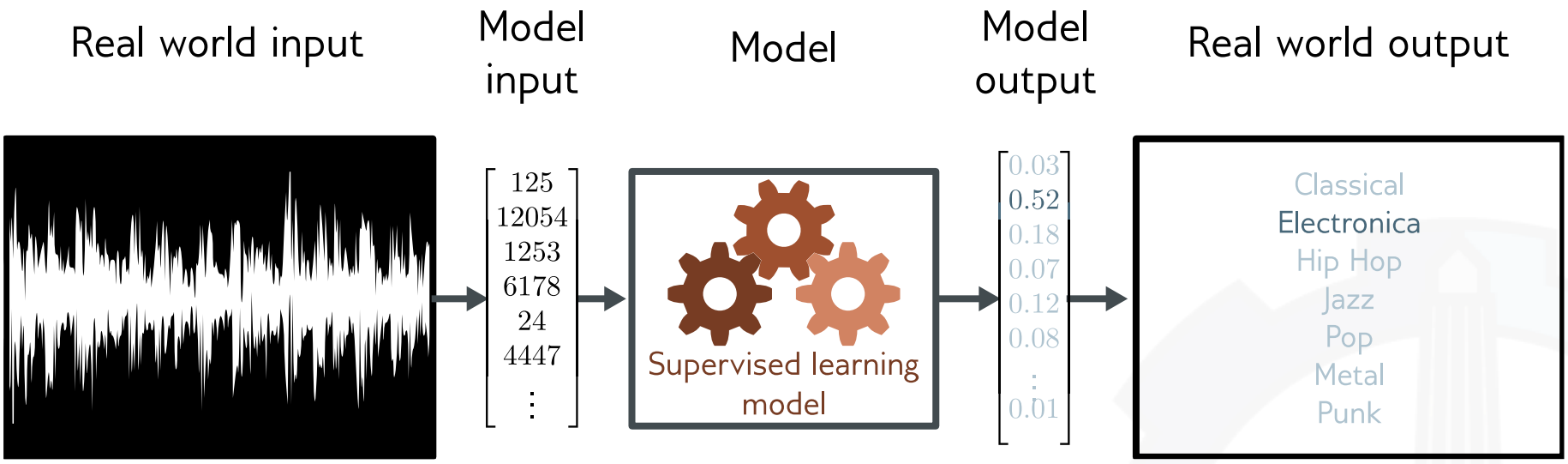
Supervised Learning (Text Classification)



- Binary classification problem (two discrete classes)
- Transformer network

Slide Credit: Simon Prince

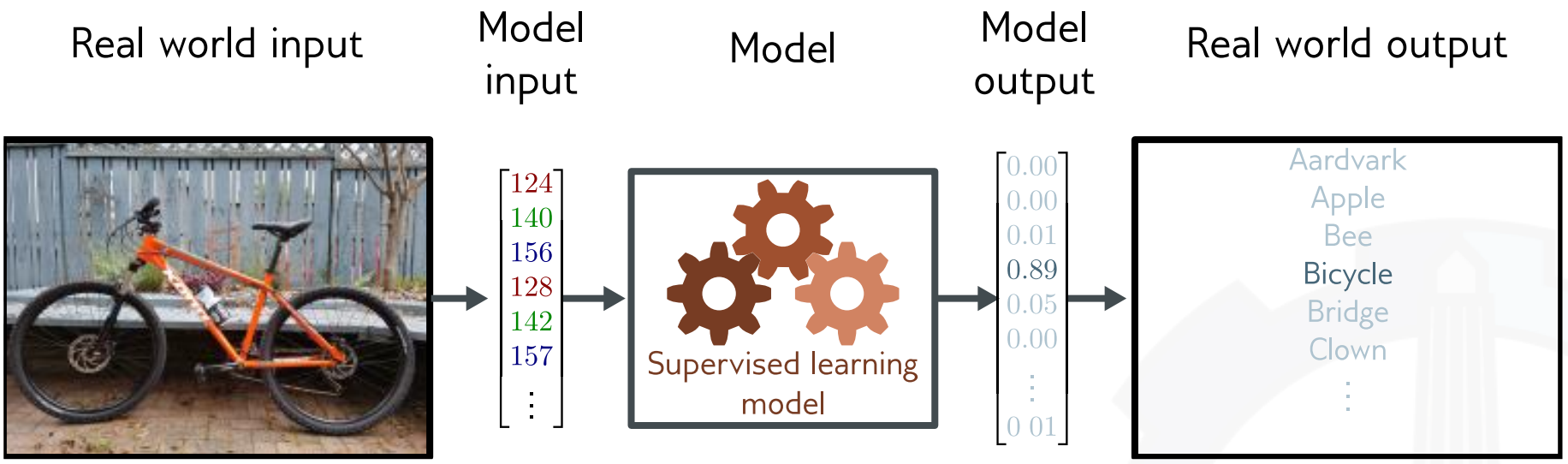
Supervised Learning (Music Genre Classification)



- Multiclass classification problem (discrete classes, >2 possible values)
- Recurrent neural network (RNN)

Slide Credit: Simon Prince

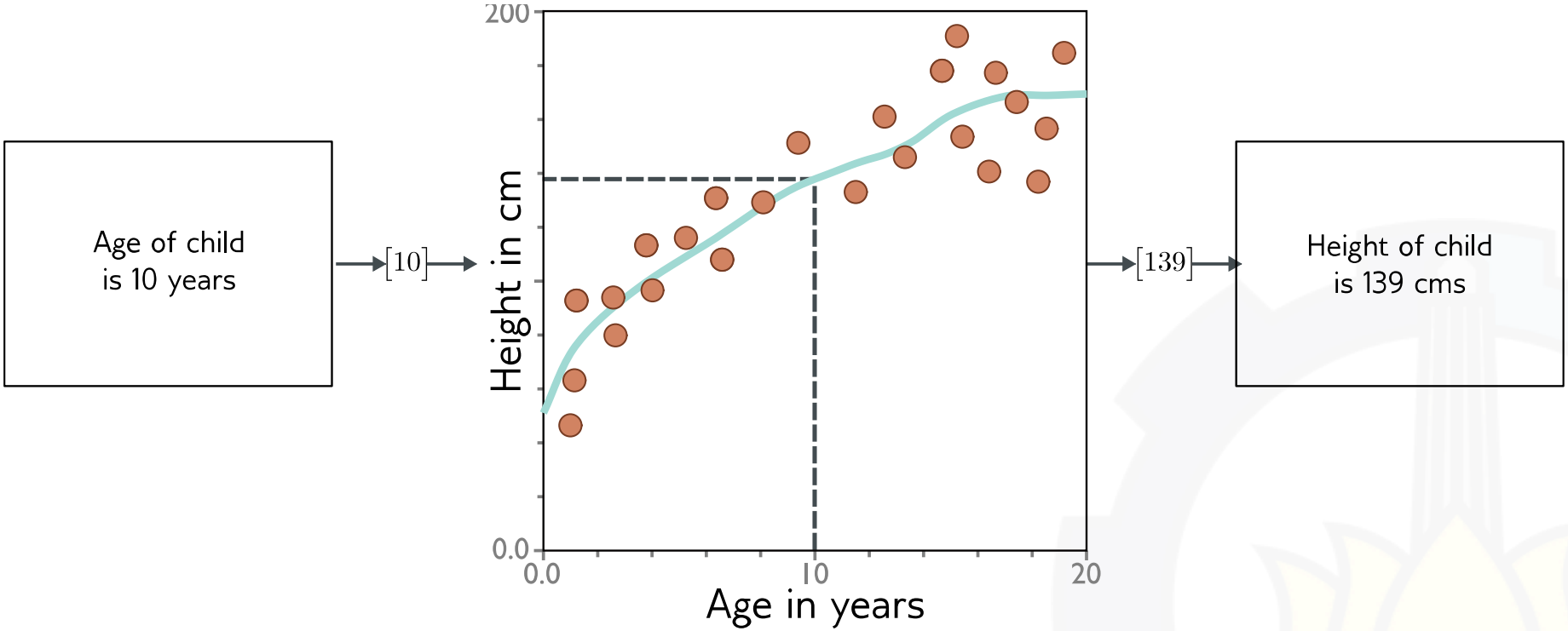
Supervised Learning (Image Classification)



- Multiclass classification problem (discrete classes, >2 possible classes)
- Convolutional network

Slide Credit: Simon Prince

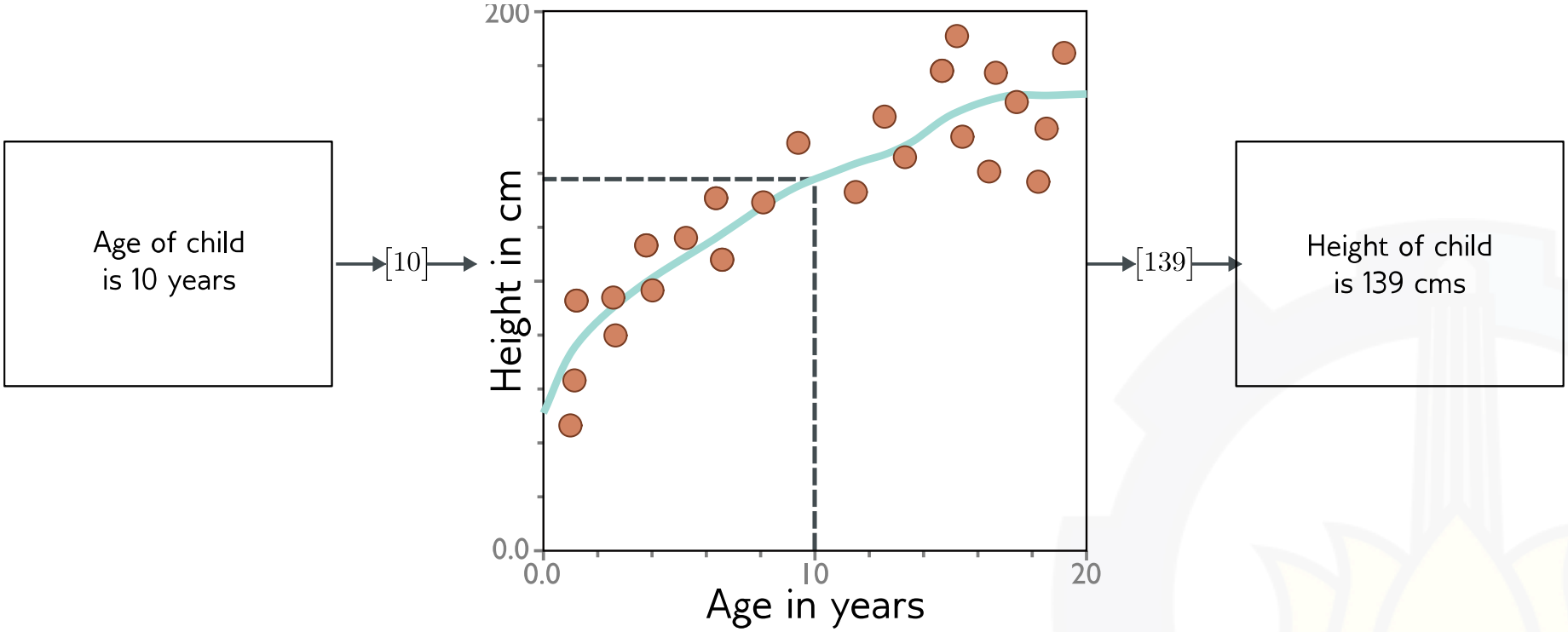
What is a supervised learning model?



- An equation relating input (age) to output (height)
- Search through family of possible equations to find one that fits training data well

Slide Credit: Simon Prince

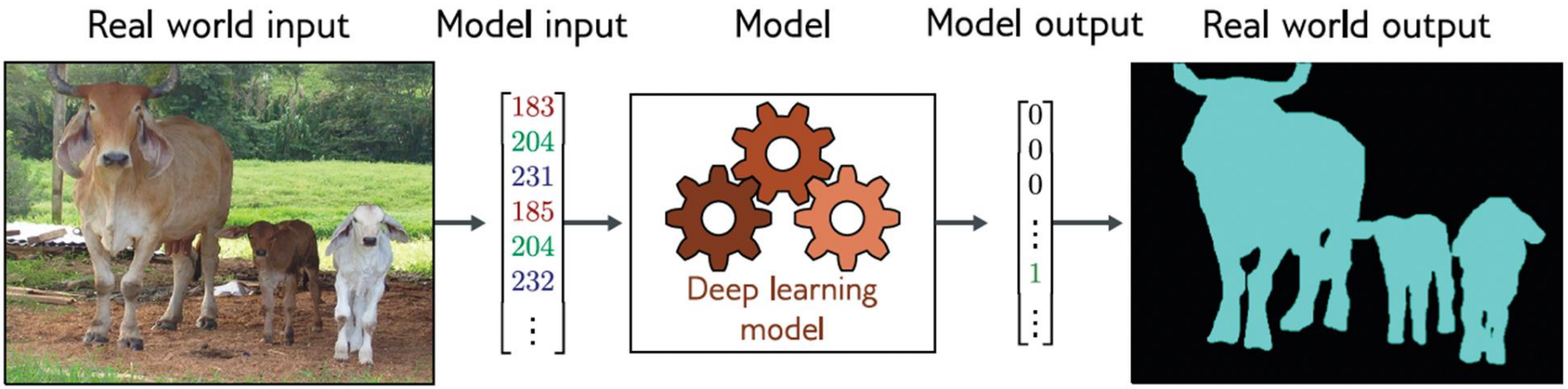
What is a supervised learning model?



- Deep neural networks are just a very flexible family of equations
- Fitting deep neural networks = “Deep Learning”

Slide Credit: Simon Prince

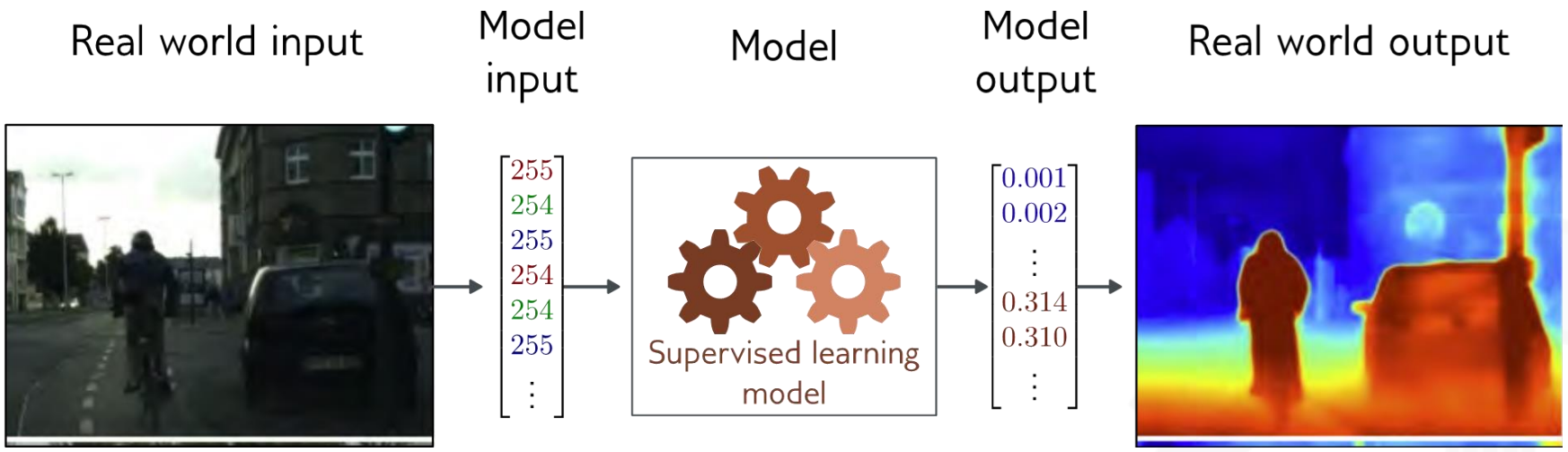
Supervised Learning (Image Segmentation)



- Multivariate binary classification problem (many outputs, two discrete classes)
- Convolutional encoder-decoder network

Slide Credit: Simon Prince

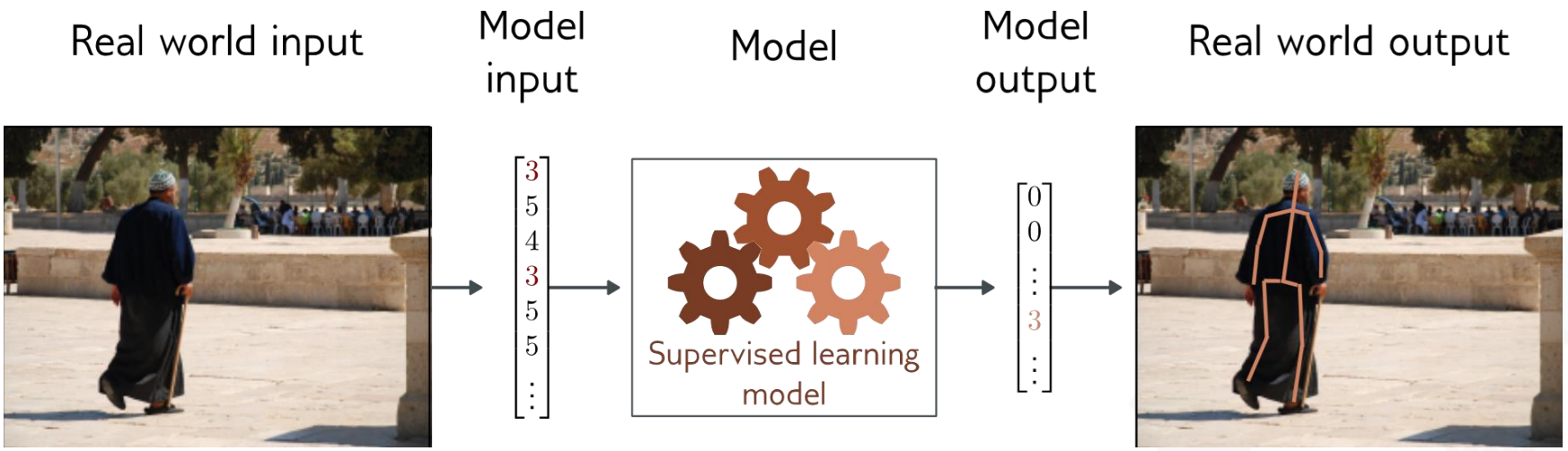
Supervised Learning (Depth Estimation)



- Multivariate regression problem (many outputs, continuous)
- Convolutional encoder-decoder network

Slide Credit: Simon Prince

Supervised Learning (Pose Estimation)



- Multivariate regression problem (many outputs, continuous)
- Convolutional encoder-decoder network

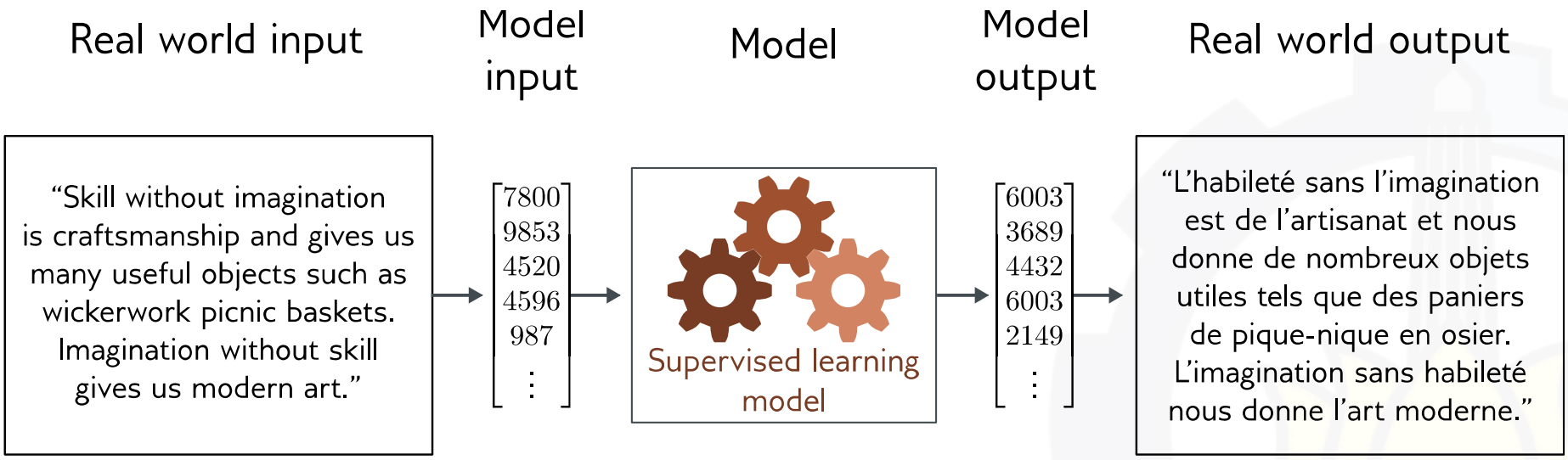
Slide Credit: Simon Prince

Terms

- Regression = continuous numbers as output
- Classification = discrete classes as output
- Two class and multiclass classification treated differently
- Univariate = one output
- Multivariate = more than one output

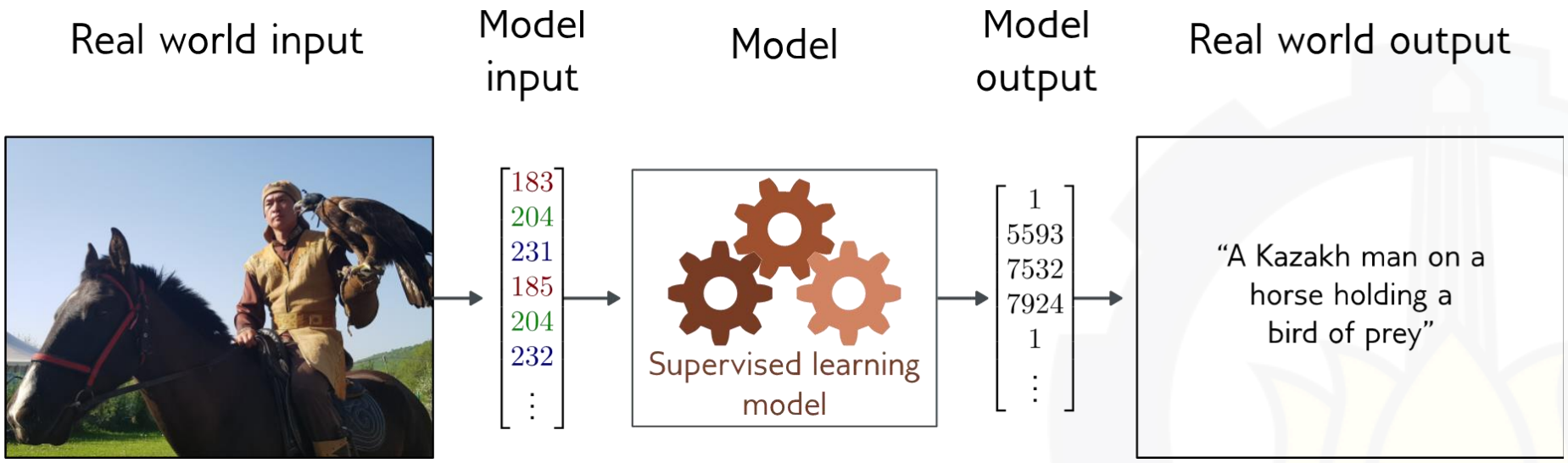
Slide Credit: Simon Prince

Supervised Learning (Translation)



Slide Credit: Simon Prince

Supervised Learning (Image Captioning)



Slide Credit: Simon Prince

What do these examples have in common?

- Very complex relationship between input and output
- Sometimes may be many possible valid answers
- But outputs (and sometimes inputs) obey rules

"A Kazakh man on a horse holding a bird of prey"

Language obeys grammatical rules

Natural images also have "rules"

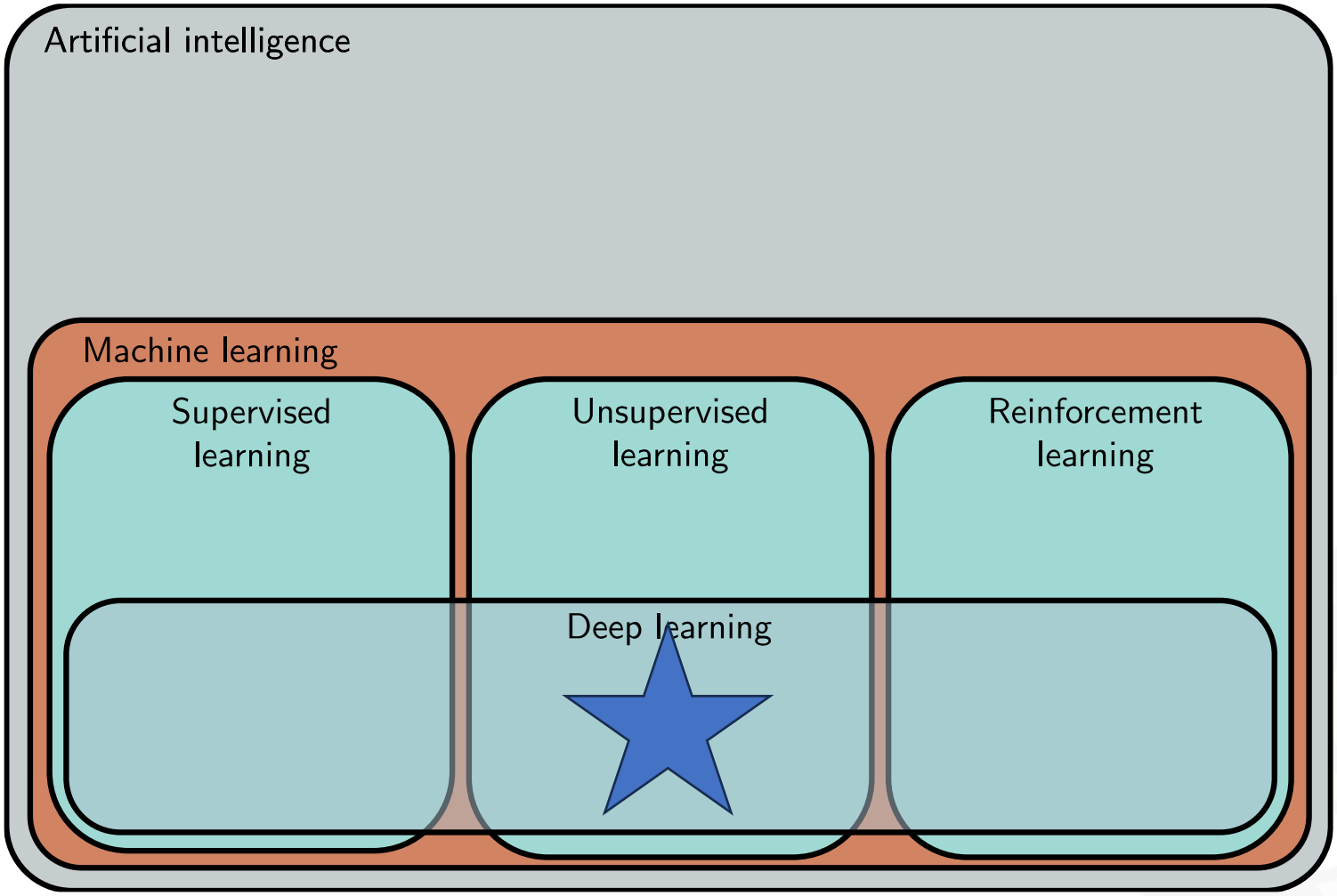
Slide Credit: Simon Prince

Idea

- Learn the “grammar” of the data from unlabeled examples
- Can use a gargantuan amount of data to do this (as unlabeled)
- Make the supervised learning task earlier by having a lot of knowledge of possible outputs

Slide Credit: Simon Prince

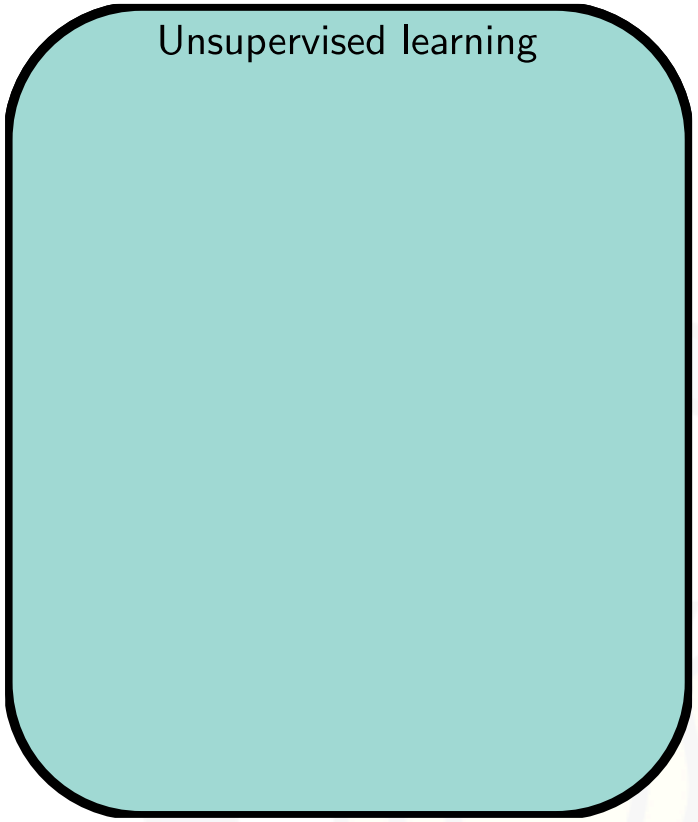
Unsupervised Learning



Slide Credit: Simon Prince

Unsupervised Learning

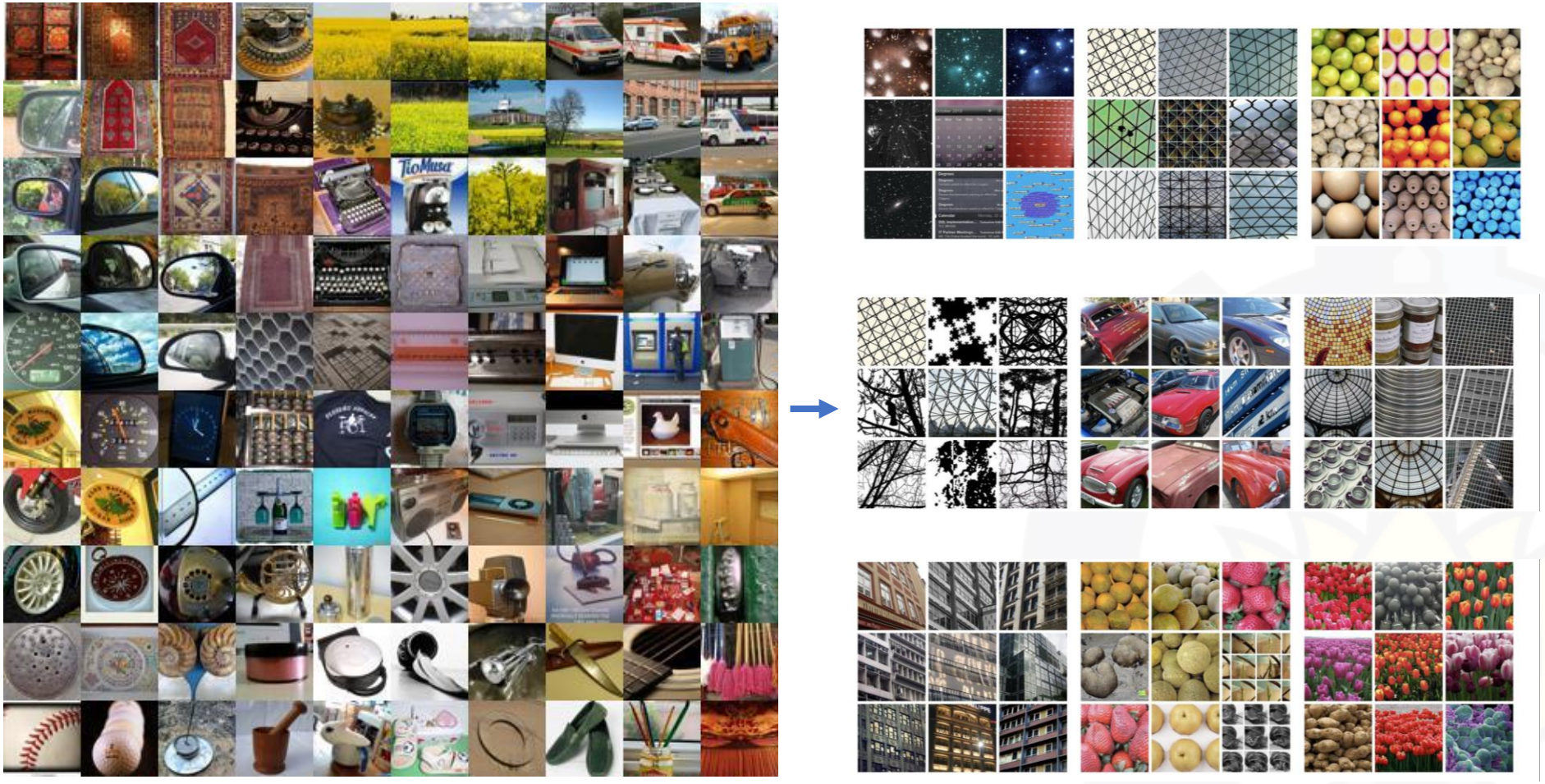
- Learning about a dataset without labels
  - Clustering
  - Finding outliers
  - Generating new examples
  - Filling in missing data



Slide Credit: Simon Prince

DeepCluster: Deep Clustering for Unsupervised Learning of Visual Features (Caron et al., 2018)

Slide Credit: Simon Prince

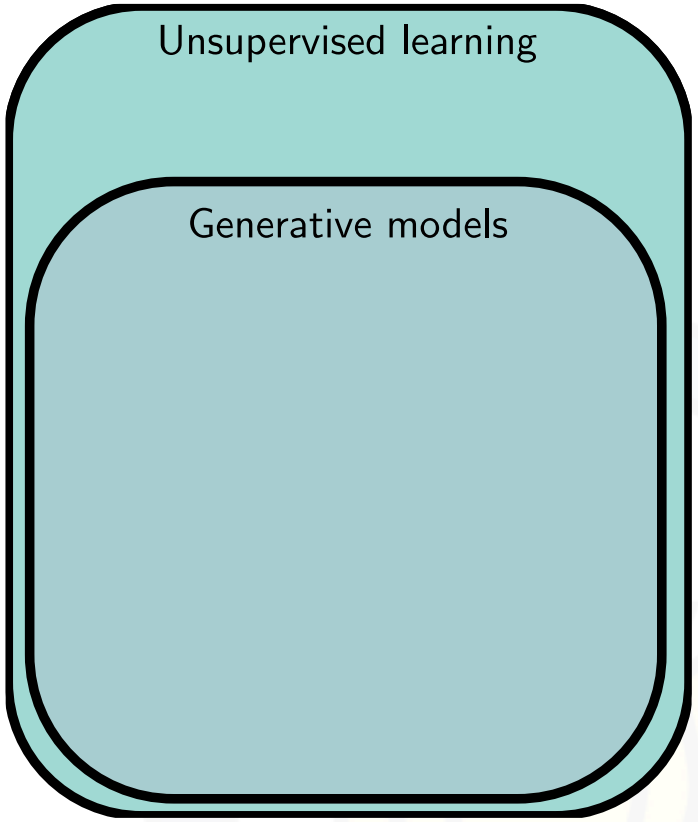


DeepCluster: Deep Clustering for Unsupervised Learning of Visual Features (Caron et al., 2018)

Slide Credit: Simon Prince

Unsupervised Learning

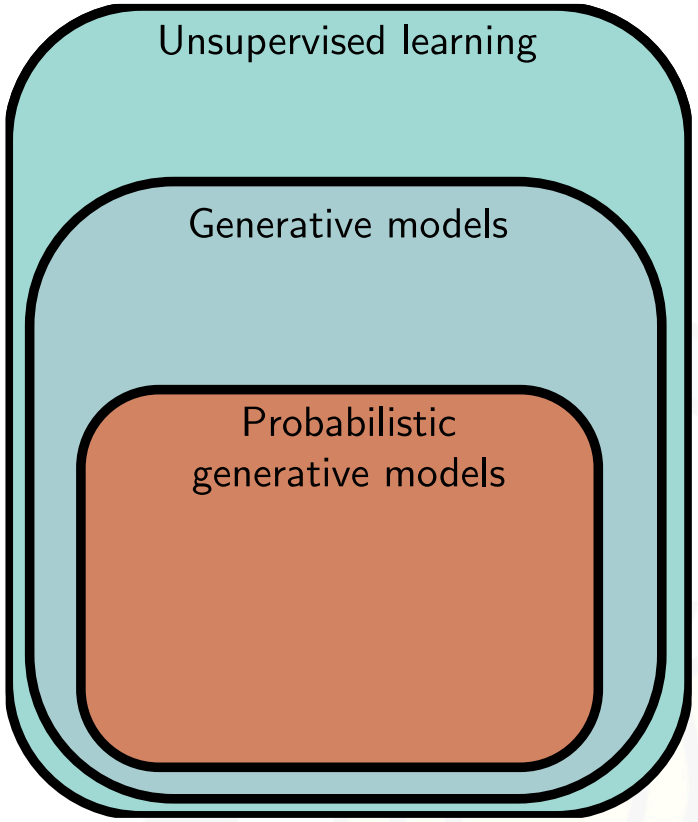
- Learning about a dataset without labels
  - e.g., clustering
- Generative models can create examples
  - e.g., generative adversarial networks



Slide Credit: Simon Prince

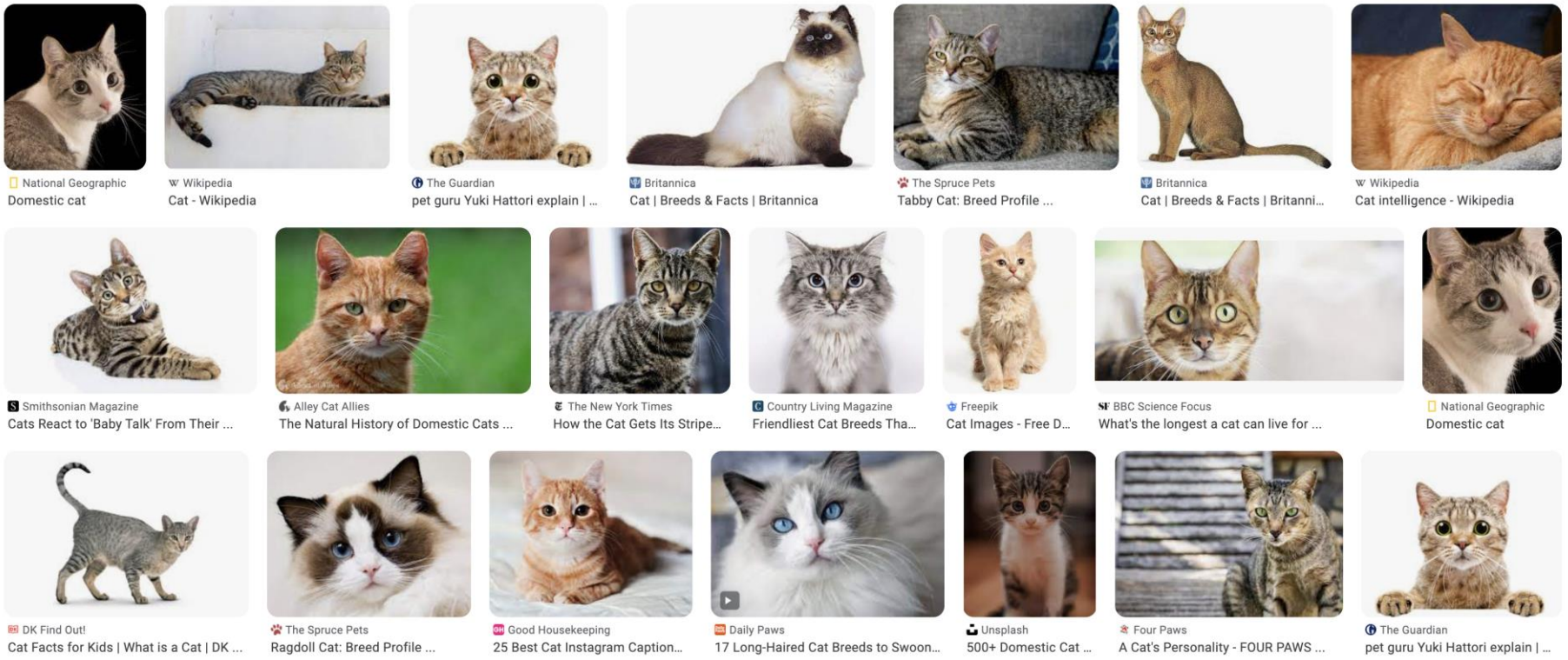
Unsupervised Learning

- Learning about a dataset without labels
  - e.g., clustering
- Generative models can create examples
  - e.g., generative adversarial networks
- PGMs learn distribution over data
  - e.g., variational autoencoders,
  - e.g., normalizing flows,
  - e.g., diffusion models



Slide Credit: Simon Prince

Generative models

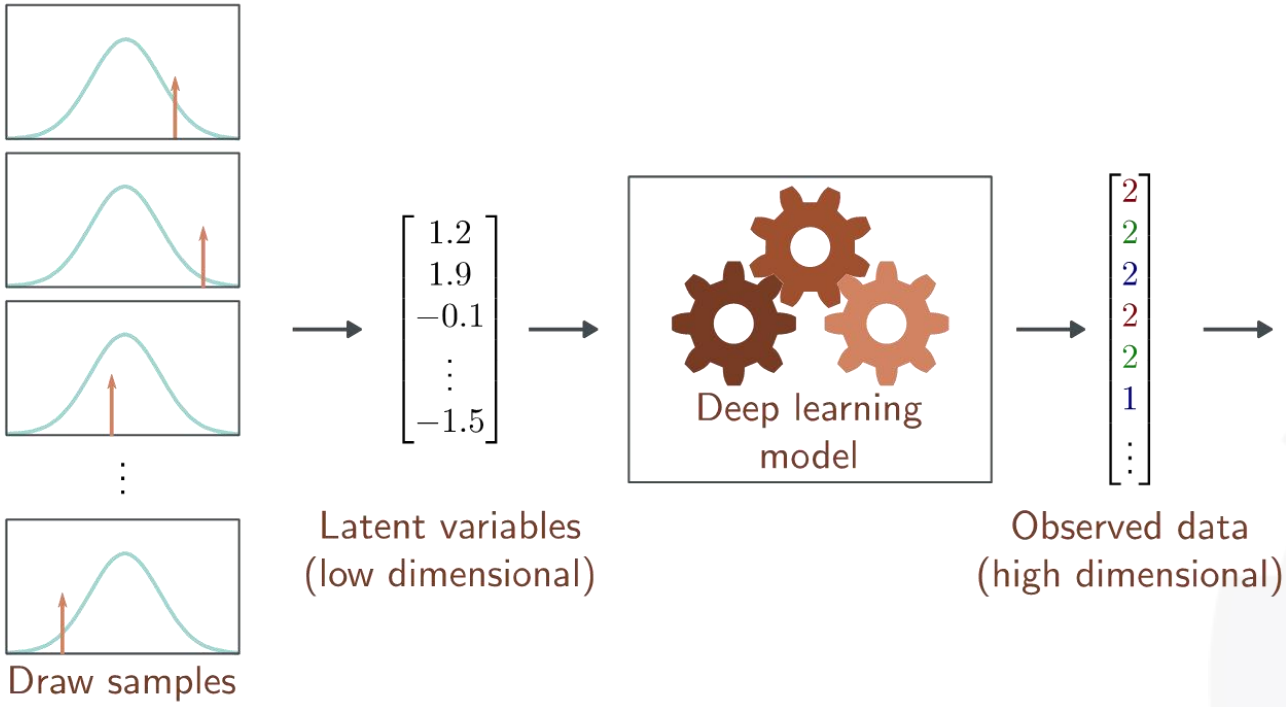


Slide Credit: Simon Prince

Generative models

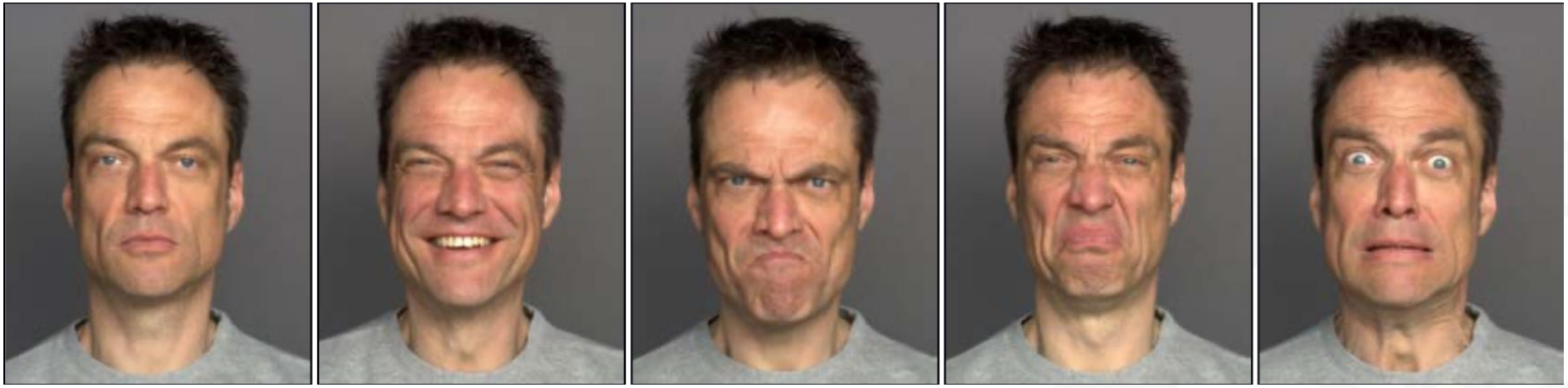
Slide Credit: Simon Prince

Latent variables



Slide Credit: Simon Prince

Why should this work?



Slide Credit: Simon Prince

Interpolation

Slide Credit: Simon Prince

Conditional synthesis

Slide Credit: Simon Prince

I was a little nervous before my first lecture at the University of Bath. It seemed like there were hundreds of students and they looked intimidating. I stepped up to the lectern and was about to speak, when something bizarre happened.

Suddenly, the room was filled with a deafening noise, like a giant roar. It was so loud that I couldn't hear anything else and I had to cover my ears. I could see the students looking around, confused and frightened. Then, as quickly as it had started, the noise stopped and the room was silent again.

I stood there for a few moments, trying to make sense of what had just happened. Then I realized that the students were all staring at me, waiting for me to say something. I tried to think of something witty or clever to say, but my mind was blank. So I just said, "Well, that was strange," and then I started my lecture.

I was a little nervous before my first lecture at the University of Bath. It seemed like there were hundreds of students and they looked intimidating. I stepped up to the lectern and was about to speak, when something bizarre happened.

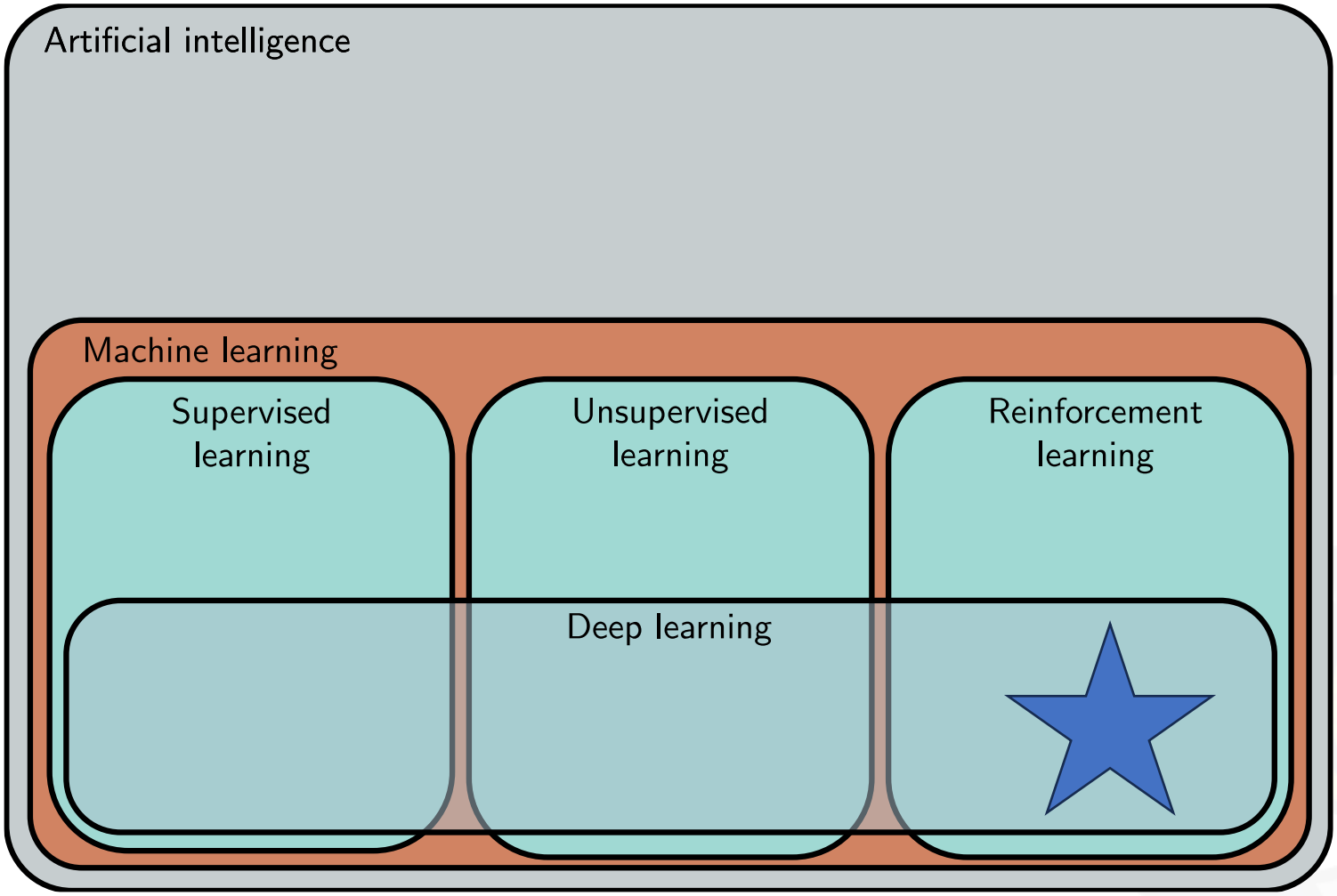
Suddenly, a giant rabbit ran into the lecture hall! The students started screaming and running around in panic. I was so shocked that I couldn't move. The rabbit ran up to me and hopped onto the lectern. Then, in a booming voice, it said:

"I am the Easter Bunny! I have come to give you all a special gift!"

The students were so surprised that they stopped screaming and listened to the Easter Bunny. Then, the Easter Bunny started handing out chocolate eggs to everyone in the lecture hall. The students were so happy that they started cheering and clapping. I was so relieved that the Easter Bunny had saved my lecture! After that, I was able to continue and the students paid attention for the rest of the hour. It was a great success!

Slide Credit: Simon Prince

Unsupervised Learning



Slide Credit: Simon Prince

Reinforcement Learning

- A set of states
- A set of actions
- A set of rewards

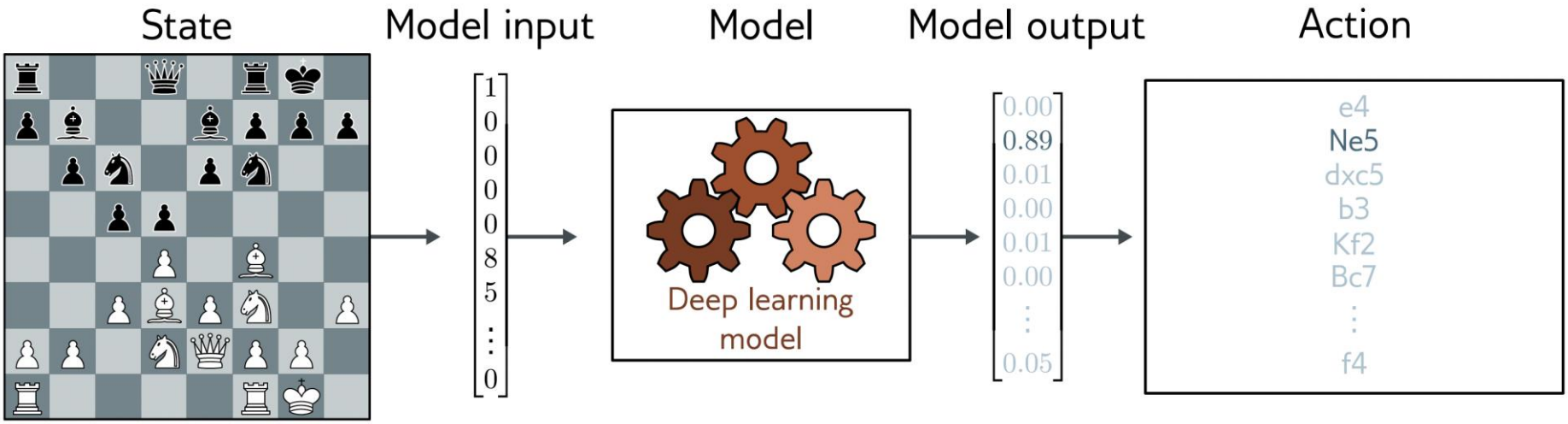
- Goal: take actions to change the state so that you receive rewards

- You don't receive any data – you have to explore the environment yourself to gather data as you go

Slide Credit: Simon Prince

Example: chess

- States are valid states of the chess board
- Actions at a given time are valid possible moves
- Positive rewards for taking pieces, negative rewards for losing them



Slide Credit: Simon Prince

Why is this difficult?

- Stochastic
  - Make the same move twice, the opponent might not do the same thing
  - Rewards also stochastic (opponent does or doesn't take your piece)
- Temporal credit assignment problem
  - Did we get the reward because of this move? Or because we made good tactical decisions somewhere in the past?
- Exploration-exploitation trade-off
  - If we found a good opening, should we use this?
  - Or should we try other things, hoping for something better?

Slide Credit: Simon Prince

Landmarks in Deep Learning

- 1958 Perceptron (Simple `neural' model)
- 1986 Backpropagation (Practical Deep Neural networks)
- 1989 Convolutional networks (Supervised learning)
- 2012 AlexNet Image classification (Supervised learning)
- 2014 Generative adversarial networks (Unsupervised learning)
- 2014 Deep Q-Learning -- Atari games (Reinforcement learning)
- 2016 AlphaGo (Reinforcement learning)
- 2017 Machine translation (Supervised learning)
- 2019 Language models ((Un)supervised learning)
- 2022 Dall-E2 Image synthesis from text prompts ((Un)supervised learning)
- 2022 ChatGPT ((Un)supervised learning)
- 2023 GPT4 Multimodal model ((Un)supervised learning)

Slide Credit: Simon Prince

2018 Turing award winners

Slide Credit: Simon Prince

Outline

- What is deep learning?
- Deep learning application
- **Simple neural network classifier**
- Deep learning classifier

We Start With

- Supervised Learning → simple and straightforward
- Supervised learning model = mapping from one or more inputs to one or more outputs
- Model is a mathematical equation

- Computing the inputs from the outputs = inference

- Example:
  - Input is age and milage of secondhand Toyota Prius
  - Output is estimated price of car

Slide Credit: Simon Prince

We Start With

- Supervised Learning → simple and straightforward
- Supervised learning model = mapping from one or more inputs to one or more outputs
- Model is a mathematical equation

- Computing the inputs from the outputs = inference
- Model also includes parameters
- Parameters affect outcome of equation

Slide Credit: Simon Prince

We Start With

- Supervised Learning → simple and straightforward
- Supervised learning model = mapping from one or more inputs to one or more outputs
- ~~Model is a mathematical equation~~
- Model is a family of equations
- Computing the inputs from the outputs = inference
- Model also includes parameters
- Parameters affect outcome of equation

Slide Credit: Simon Prince

We Start With

- Supervised Learning → simple and straightforward
- Supervised learning model = mapping from one or more inputs to one or more outputs
- Model is a family of equations

- Computing the inputs from the outputs = inference
- Model also includes parameters
- Parameters affect outcome of equation
- Training a model = finding parameters that predict outputs "well" from inputs for a training dataset of input/output pairs

Slide Credit: Simon Prince

Notation:

- Input:

$$\mathbf{x}$$

- Output:

$$\mathbf{y}$$

- Model:

$$\mathbf{y} = \mathbf{f}[\mathbf{x}]$$

Variables always Roman letters

Normal = scalar
Bold = vector
Capital Bold = matrix

Functions always square brackets

Normal = returns scalar
Bold = returns vector
Capital Bold = returns matrix

Slide Credit: Simon Prince

Notation:

- Input:

$$\mathbf{x} = \begin{bmatrix} \text{age} \\ \text{mileage} \end{bmatrix}$$

Structured or tabular data

- Output:

$$y = [\text{price}]$$

- Model:

$$y = \text{f}[\mathbf{x}]$$

Slide Credit: Simon Prince

Model

- Parameters:

$$\boldsymbol{\phi}$$

Parameters always Greek letters

- Model:

$$\mathbf{y} = \mathbf{f}[\mathbf{x}, \boldsymbol{\phi}]$$

Slide Credit: Simon Prince

Loss Function

- Training dataset of I pairs of input/output examples:

$$\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{I}$$

- Loss function or cost function measures how bad model is:

$$L\Big[\underbrace{\boldsymbol{\phi}, \mathrm{f}[\mathbf{x}, \boldsymbol{\phi}]}_{\text{model}}, \underbrace{\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{I}}_{\text{train data}}\Big]$$

or for short: $L[\boldsymbol{\phi}]$ ← Returns a scalar that is smaller when model maps inputs to outputs better

Slide Credit: Simon Prince

Training

- Loss function:

$$L\left[\boldsymbol{\phi}\right]$$

Returns a scalar that is smaller when model maps inputs to outputs better

- Find the parameters that minimize the loss:

$$\hat{\boldsymbol{\phi}} = \underset{\boldsymbol{\phi}}{\operatorname{argmin}}\Big[\mathrm{L}\left[\boldsymbol{\phi}\right]\Big]$$

Slide Credit: Simon Prince

Testing

- To test the model, run on a separate test dataset of input / output pairs

- See how well it generalizes to new data

Slide Credit: Simon Prince

Example: 1D Linear regression model

- Model:

$$y = \mathrm{f}[x, \boldsymbol{\phi}]$$
$$= \phi_0 + \phi_1 x$$

- Parameters

$$\boldsymbol{\phi} = \begin{bmatrix} \phi_0 \\ \phi_1 \end{bmatrix}$$

ϕ_0 ← y-offset

ϕ_1 ← slope

Slide Credit: Simon Prince

Example: 1D Linear regression model

- Model:

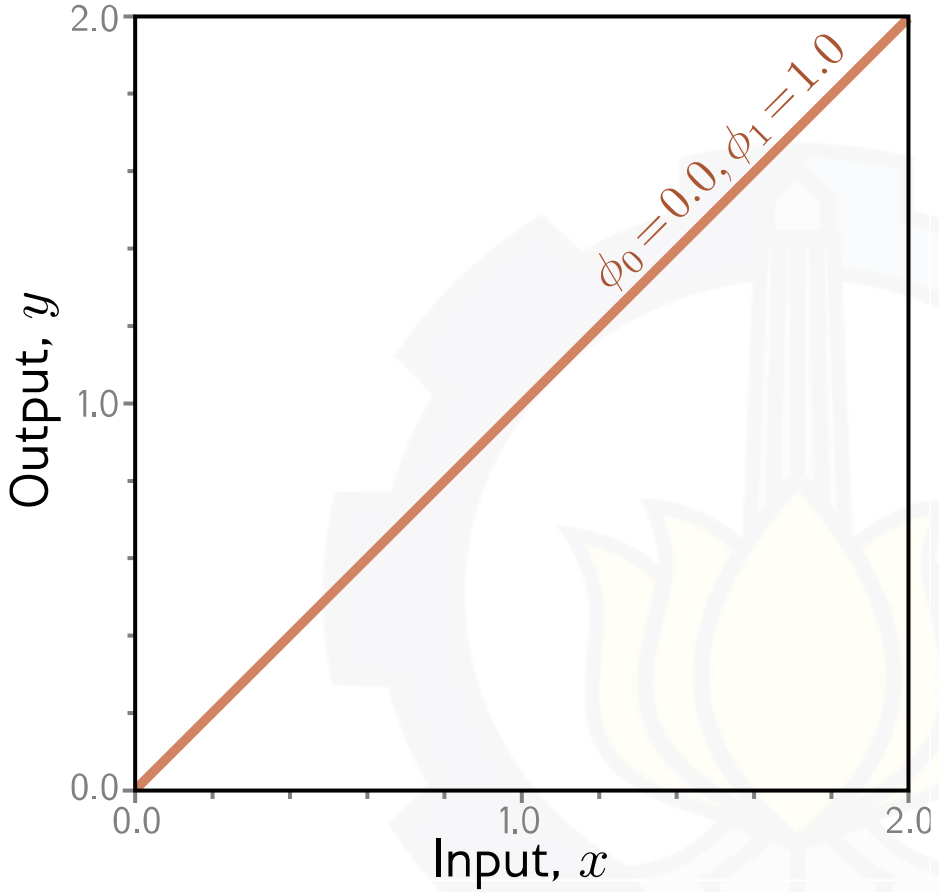
$$y = \mathrm{f}[x, \boldsymbol{\phi}]$$
$$= \phi_0 + \phi_1 x$$

- Parameters

$$\boldsymbol{\phi} = \begin{bmatrix} \phi_0 \\ \phi_1 \end{bmatrix}$$

ϕ_0 ← y-offset

ϕ_1 ← slope



Slide Credit: Simon Prince

Example: 1D Linear regression model

- Model:

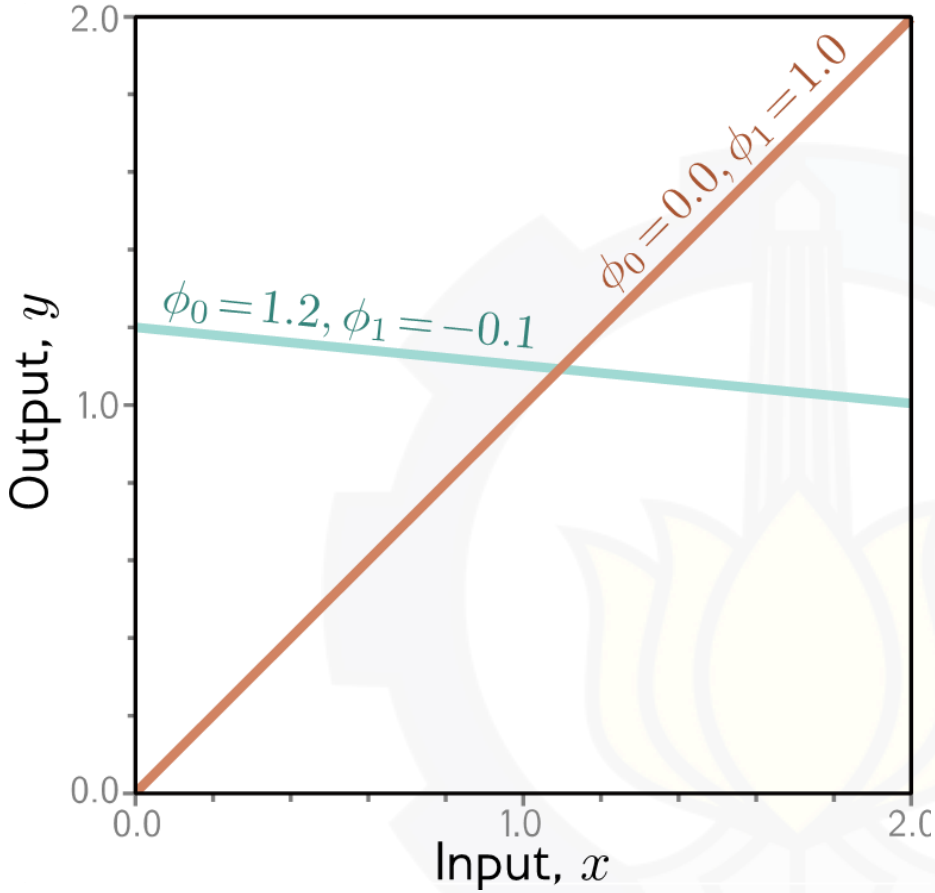
$$y = \mathrm{f}[x, \boldsymbol{\phi}]$$
$$= \phi_0 + \phi_1 x$$

- Parameters

$$\boldsymbol{\phi} = \begin{bmatrix} \phi_0 \\ \phi_1 \end{bmatrix}$$

ϕ_0 ← y-offset

ϕ_1 ← slope



Slide Credit: Simon Prince

Example: 1D Linear regression model

- Model:

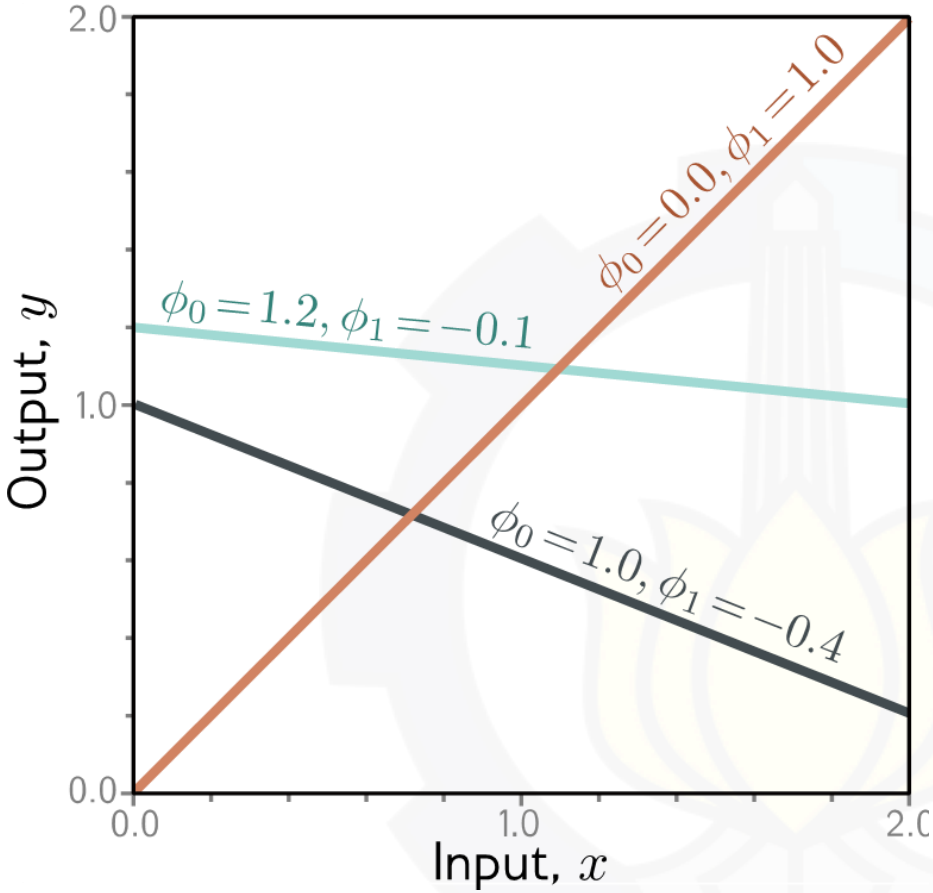
$$y = \mathrm{f}[x, \boldsymbol{\phi}]$$
$$= \phi_0 + \phi_1 x$$

- Parameters

$$\boldsymbol{\phi} = \begin{bmatrix} \phi_0 \\ \phi_1 \end{bmatrix}$$

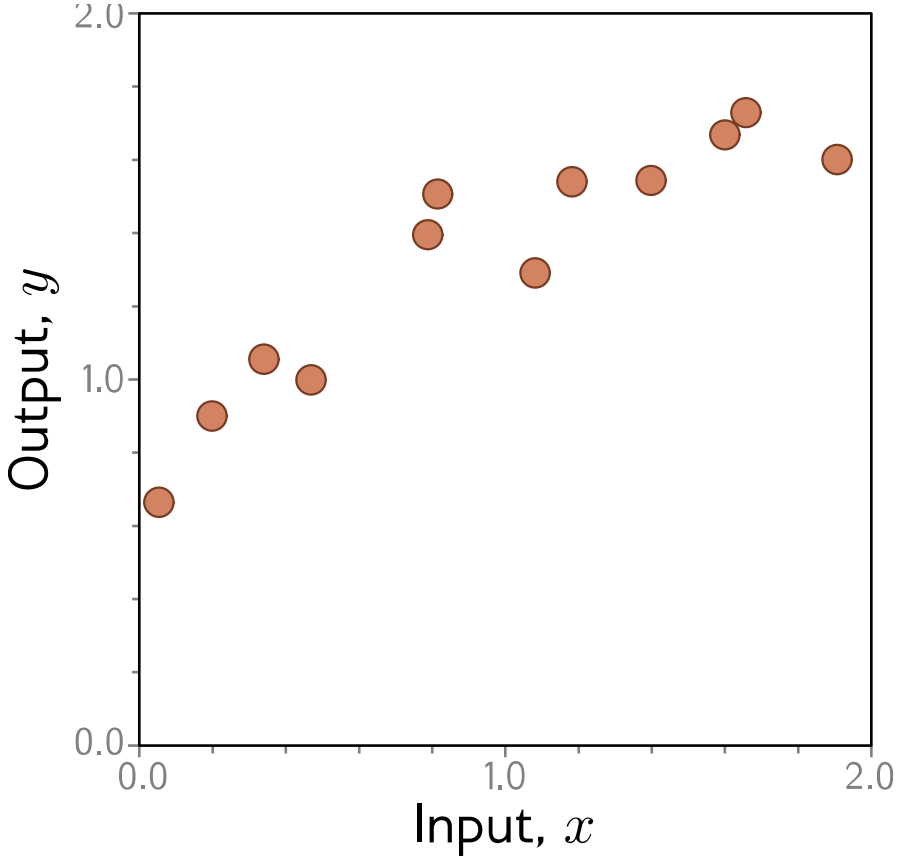
ϕ_0 ← y-offset

ϕ_1 ← slope



Slide Credit: Simon Prince

Example: 1D Linear regression model



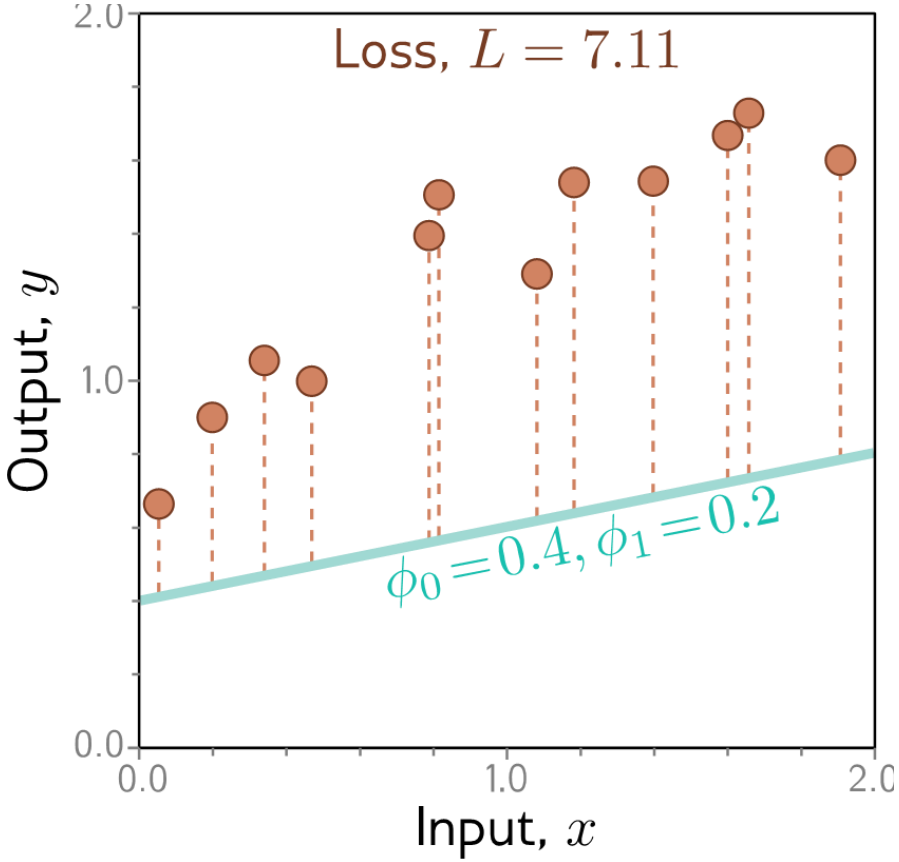
Loss function:

$$L[\boldsymbol{\phi}] = \sum_{i=1}^{I} (\mathrm{f}[x_i, \boldsymbol{\phi}] - y_i)^2$$

$$= \sum_{i=1}^{I} (\phi_0 + \phi_1 x_i - y_i)^2$$

"Least squares loss function"

Example: 1D Linear regression model



Loss function:

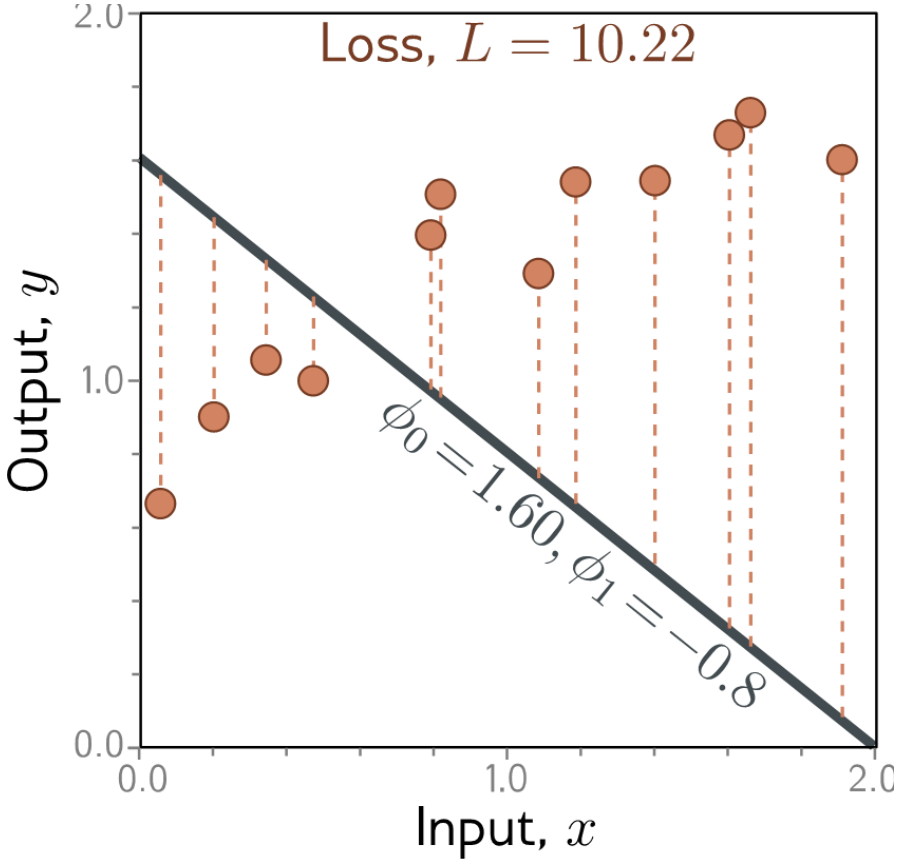
$$L[\boldsymbol{\phi}] = \sum_{i=1}^{I} (\mathrm{f}[x_i, \boldsymbol{\phi}] - y_i)^2$$

$$= \sum_{i=1}^{I} (\phi_0 + \phi_1 x_i - y_i)^2$$

"Least squares loss function"

Slide Credit: Simon Prince

Example: 1D Linear regression model



Loss function:

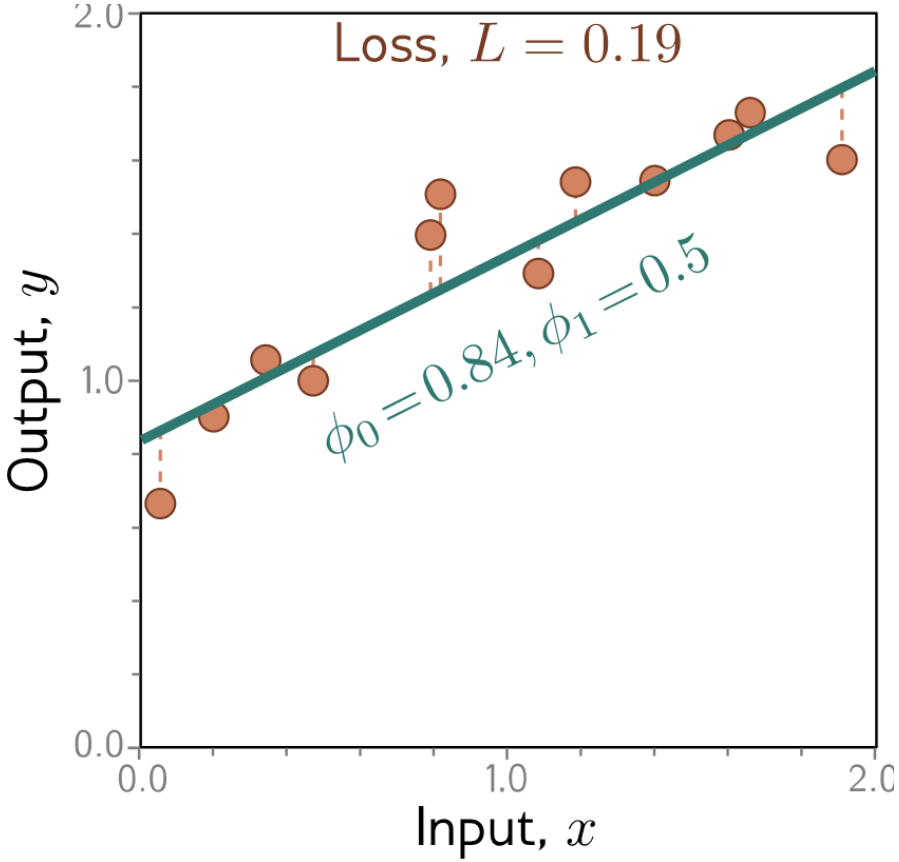
$$L[\boldsymbol{\phi}] = \sum_{i=1}^{I} (\mathrm{f}[x_i, \boldsymbol{\phi}] - y_i)^2$$

$$= \sum_{i=1}^{I} (\phi_0 + \phi_1 x_i - y_i)^2$$

"Least squares loss function"

Slide Credit: Simon Prince

Example: 1D Linear regression model



Loss function:

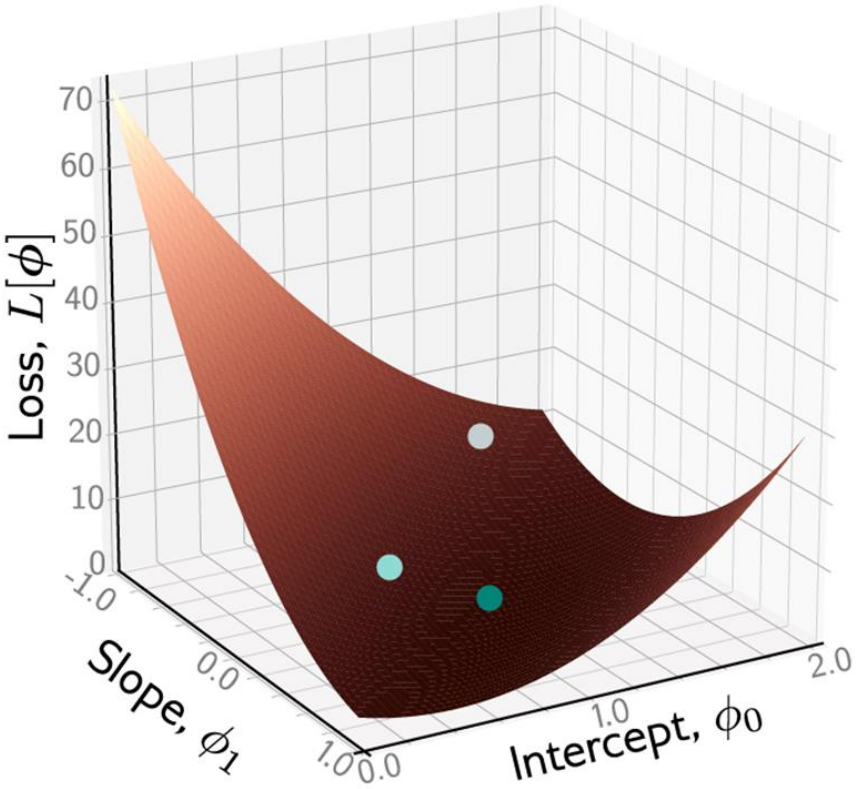
$$L[\boldsymbol{\phi}] = \sum_{i=1}^{I} (\mathrm{f}[x_i, \boldsymbol{\phi}] - y_i)^2$$

$$= \sum_{i=1}^{I} (\phi_0 + \phi_1 x_i - y_i)^2$$

"Least squares loss function"

Slide Credit: Simon Prince

Example: 1D Linear regression model



Loss function:

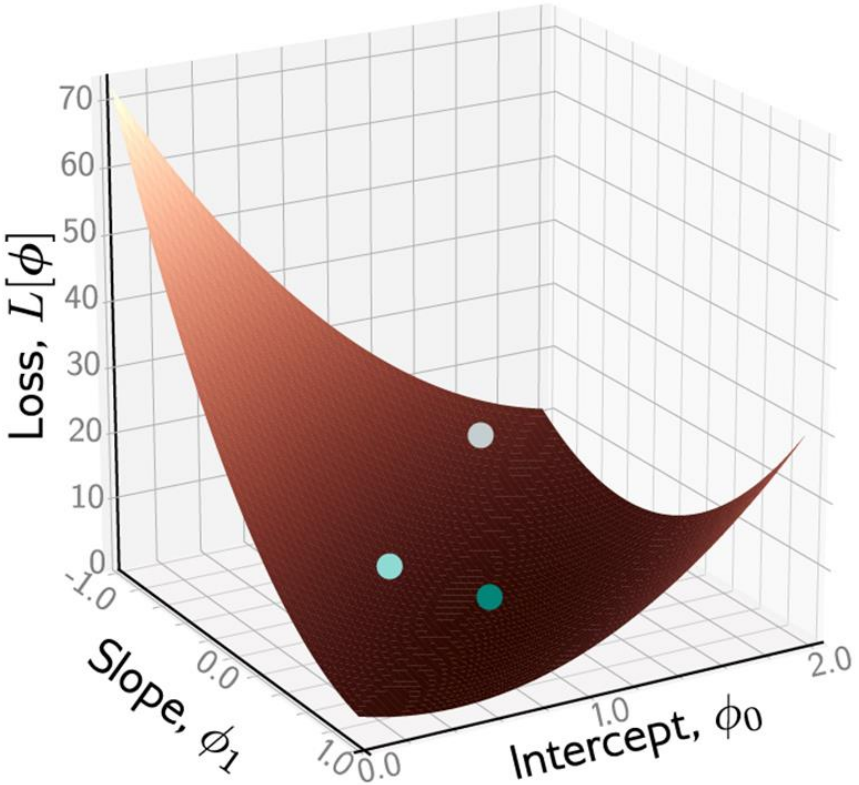
$$L[\boldsymbol{\phi}] = \sum_{i=1}^{I} (\mathrm{f}[x_i, \boldsymbol{\phi}] - y_i)^2$$

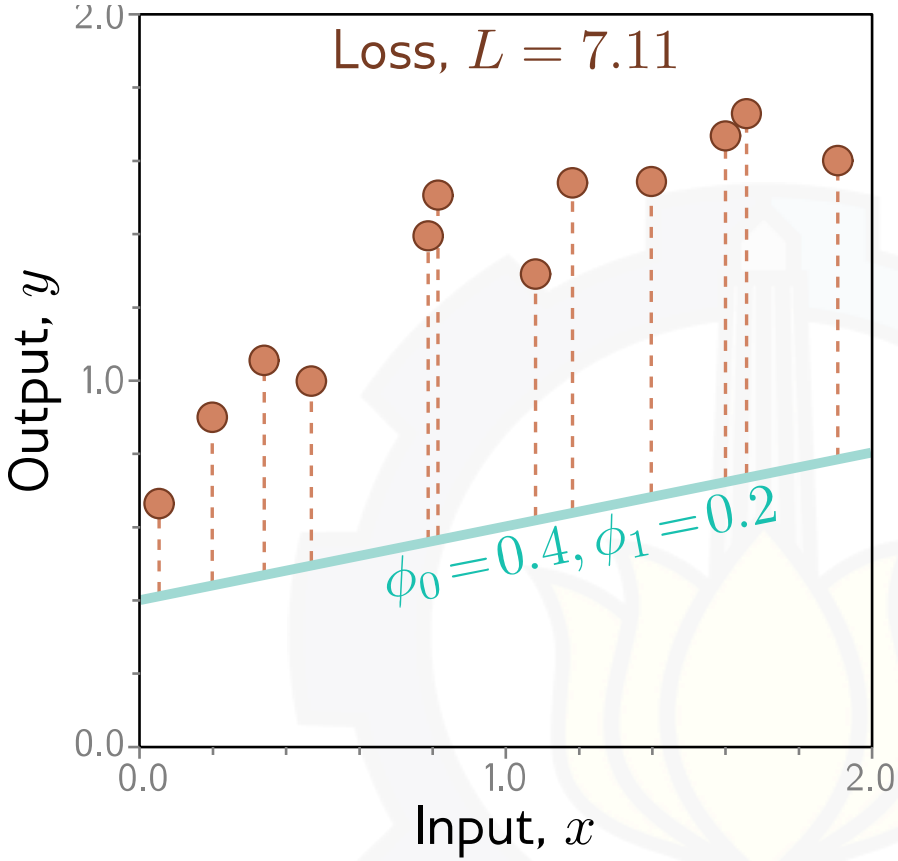
$$= \sum_{i=1}^{I} (\phi_0 + \phi_1 x_i - y_i)^2$$

"Least squares loss function"

Slide Credit: Simon Prince

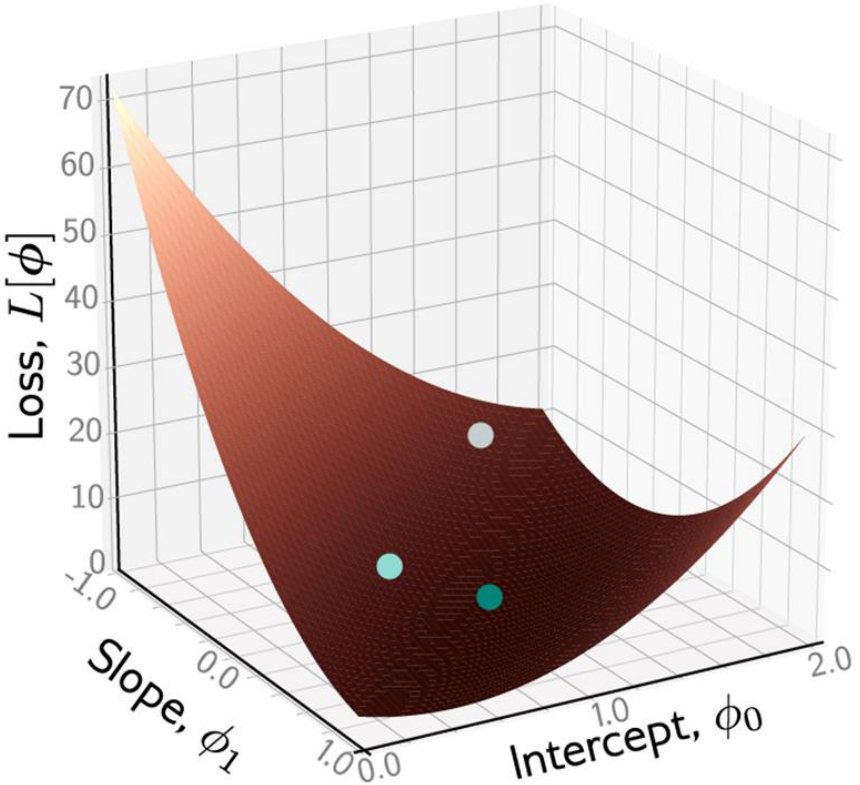
Example: 1D Linear regression model

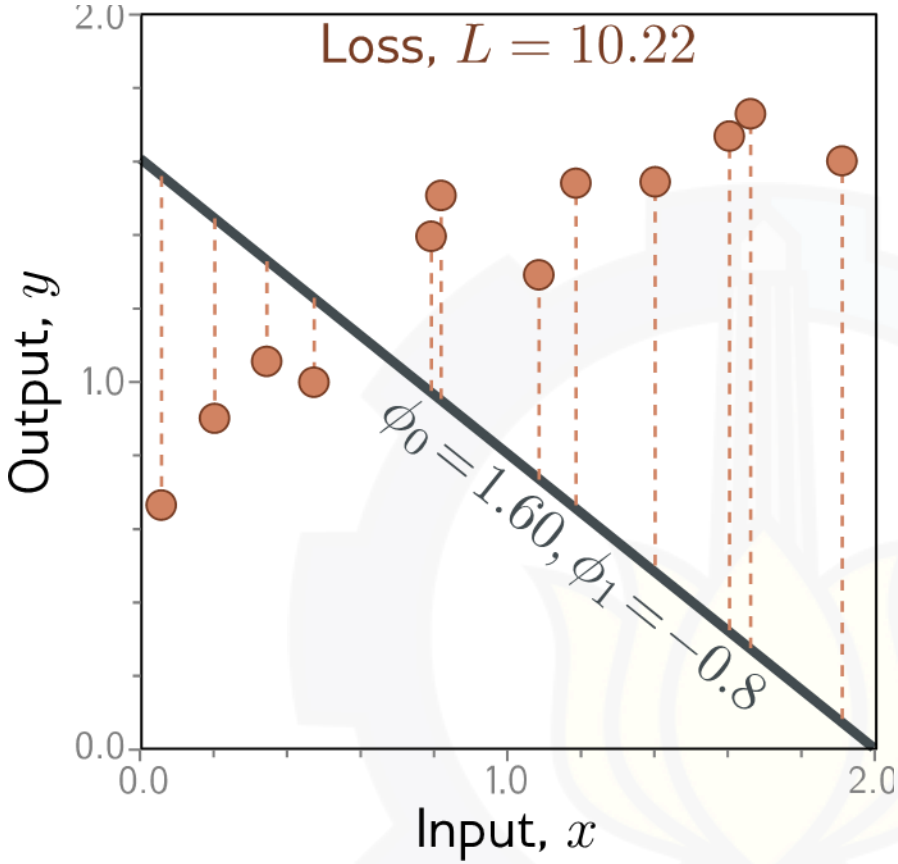




Slide Credit: Simon Prince

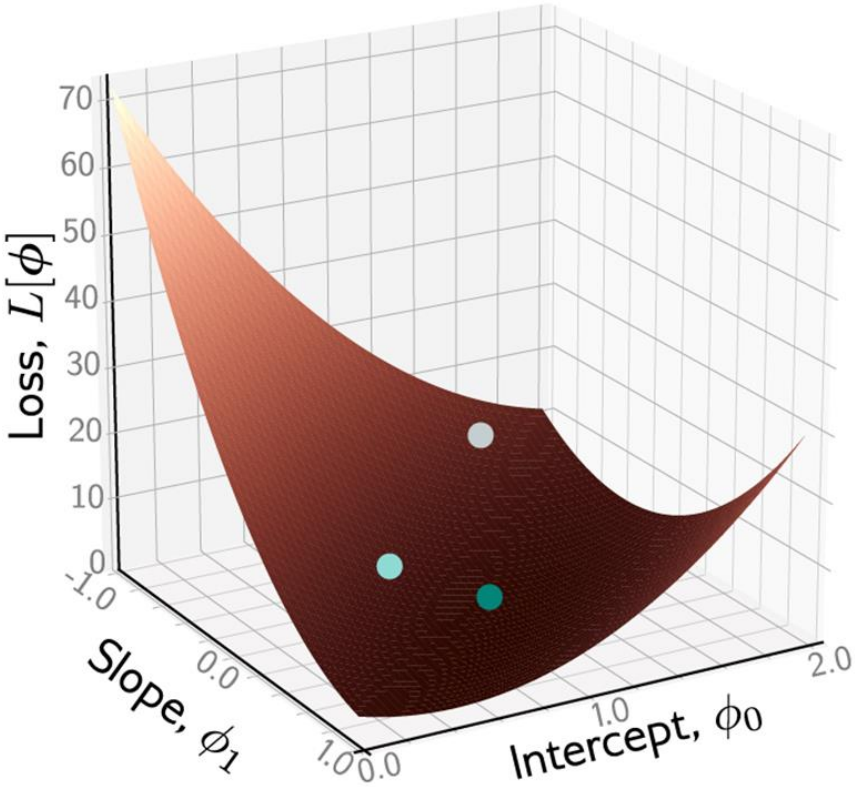
Example: 1D Linear regression model

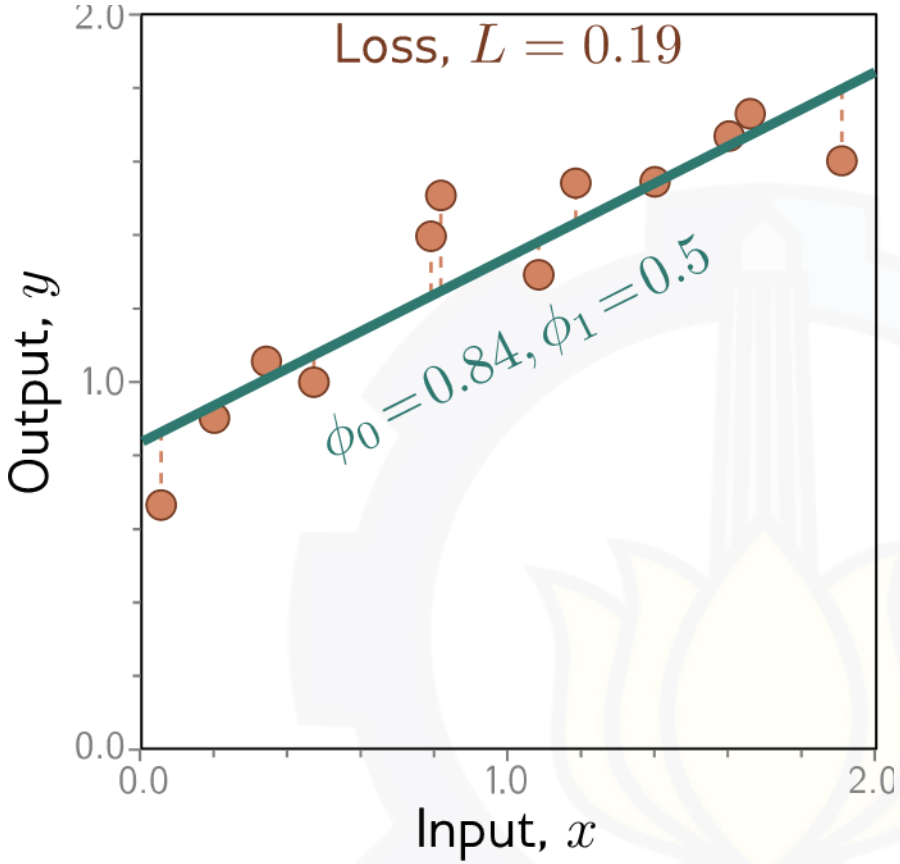




Slide Credit: Simon Prince

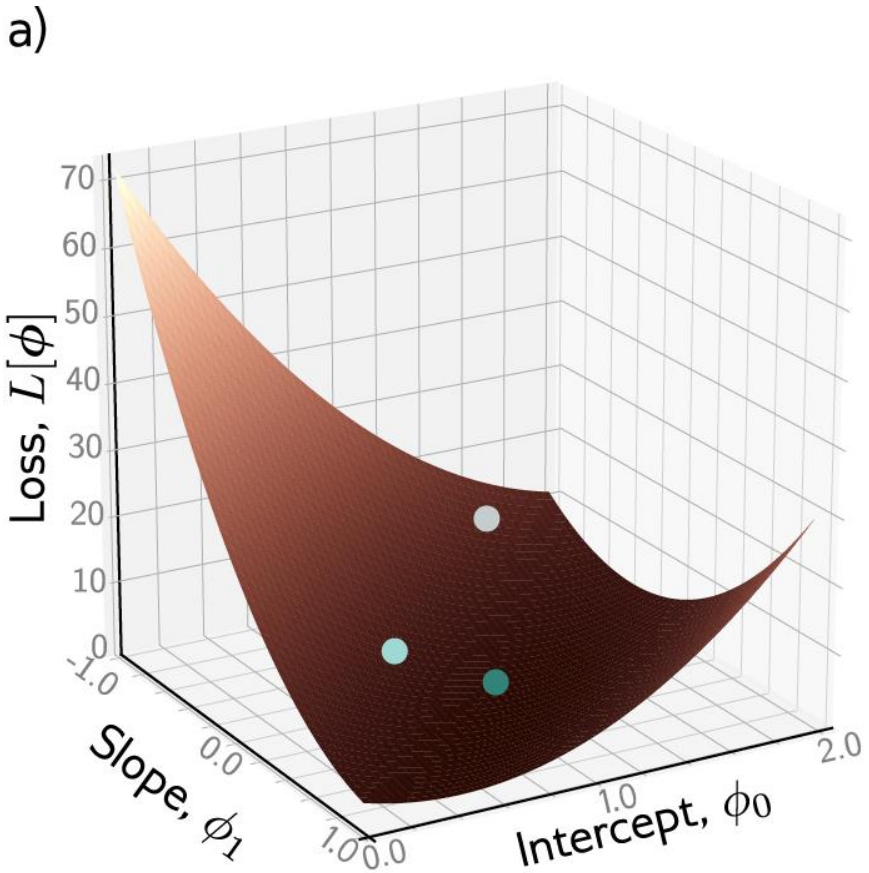
Example: 1D Linear regression model

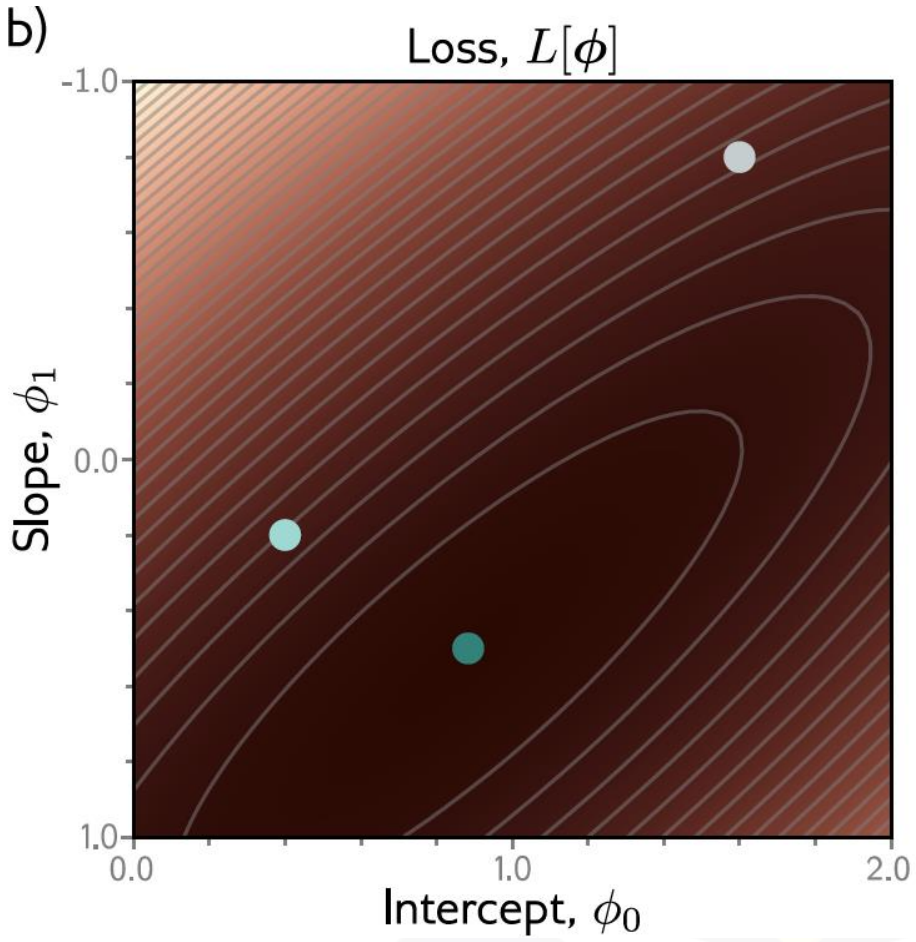




Slide Credit: Simon Prince

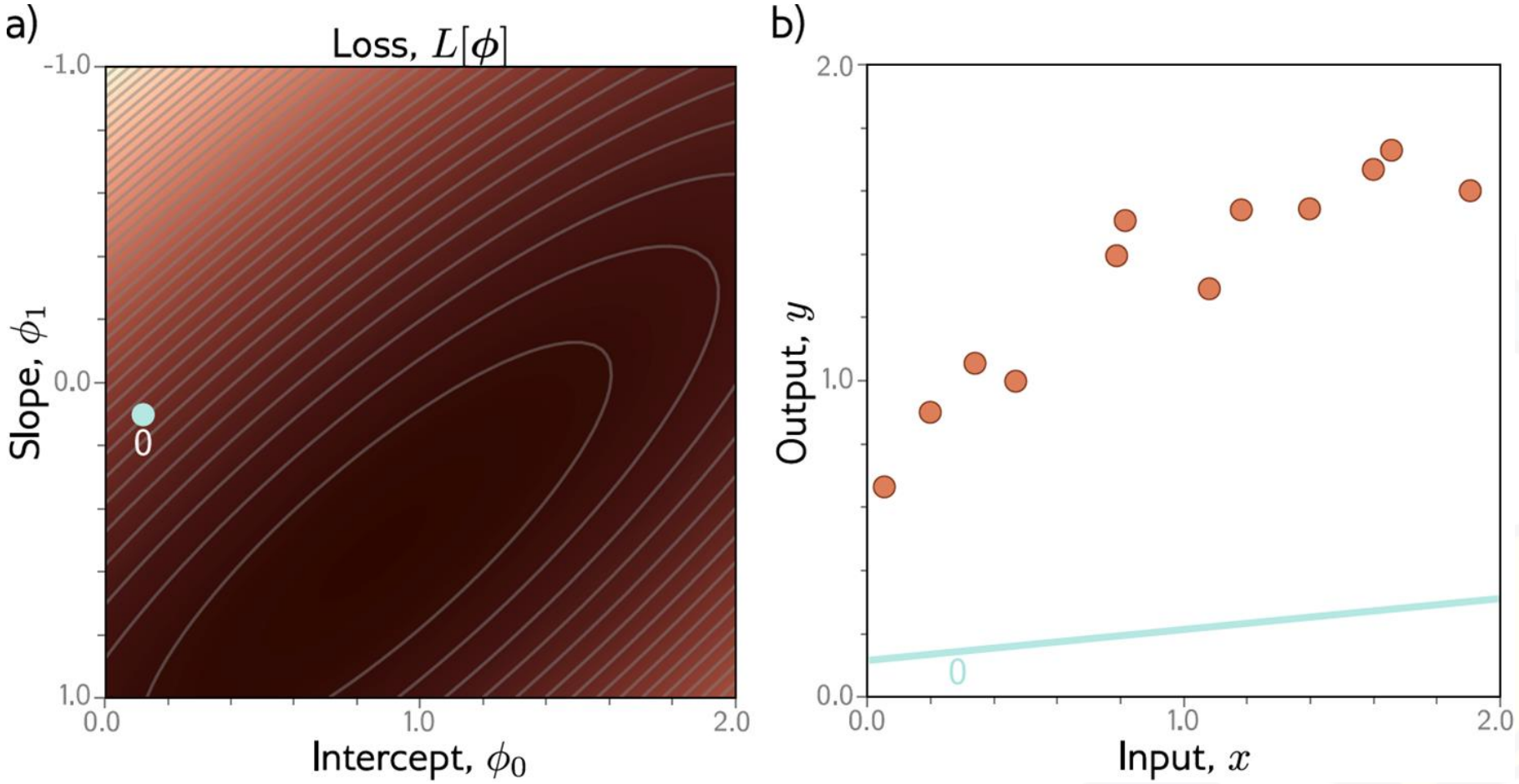
Example: 1D Linear regression model





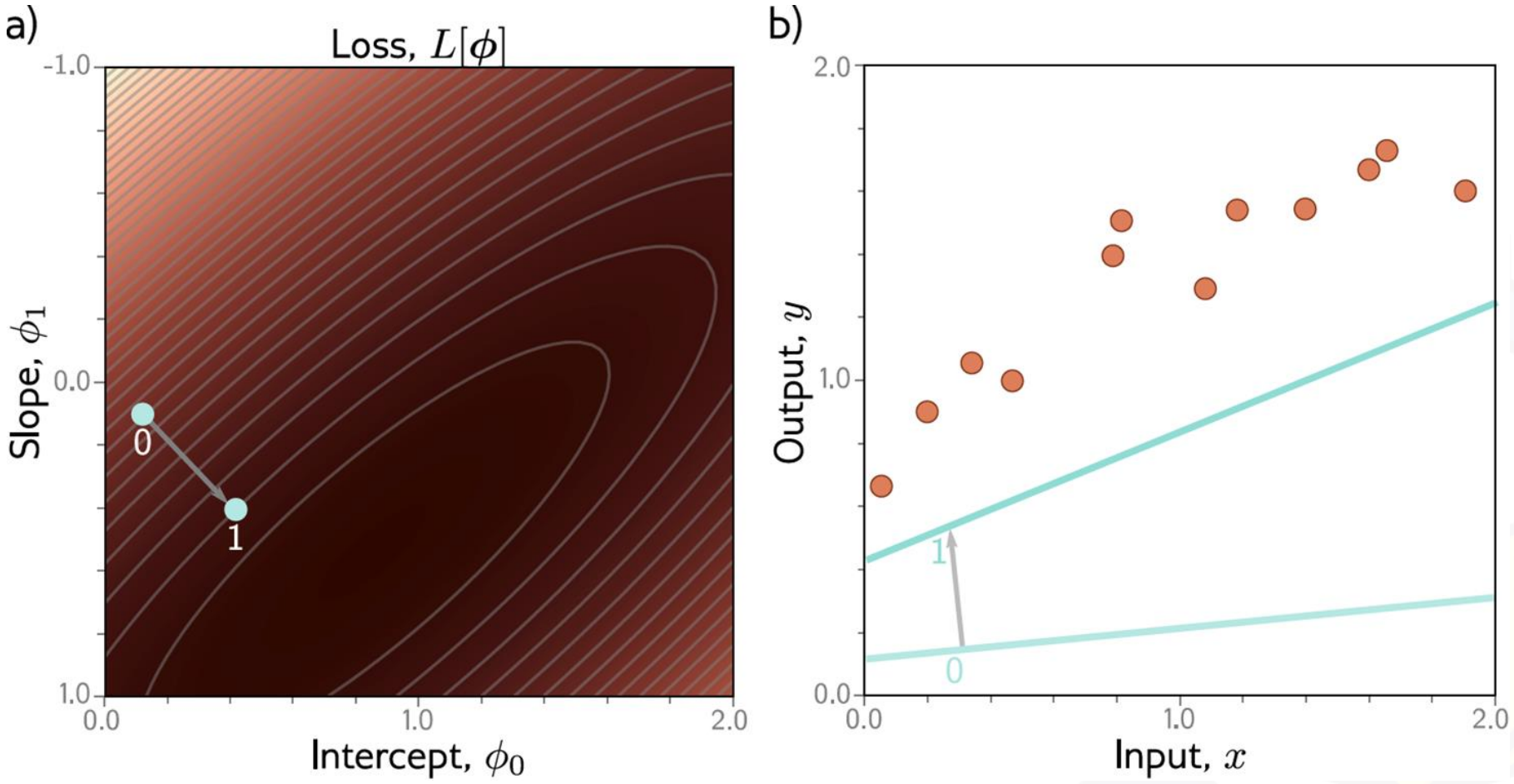
Slide Credit: Simon Prince

Example: 1D Linear regression model



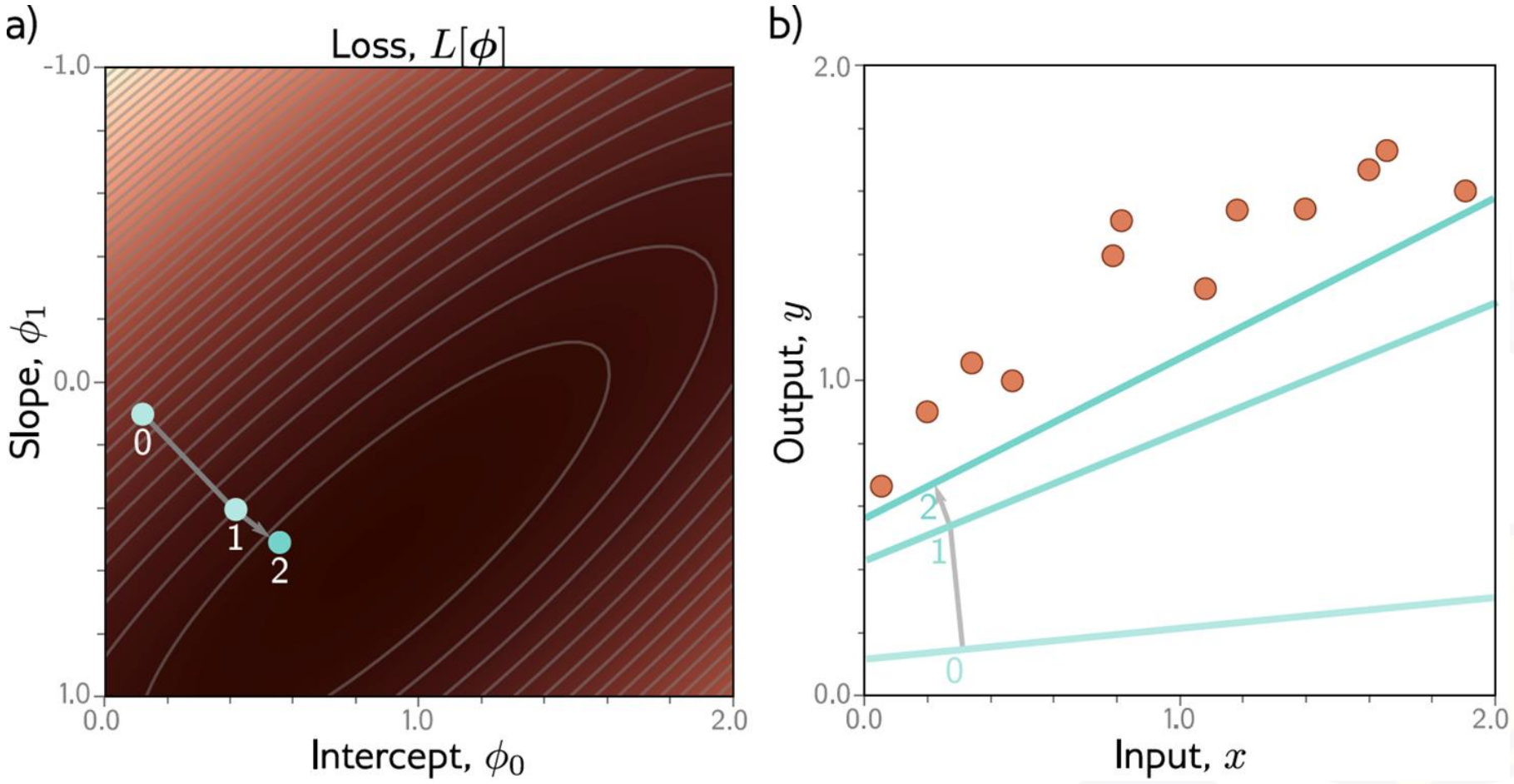
Slide Credit: Simon Prince

Example: 1D Linear regression model



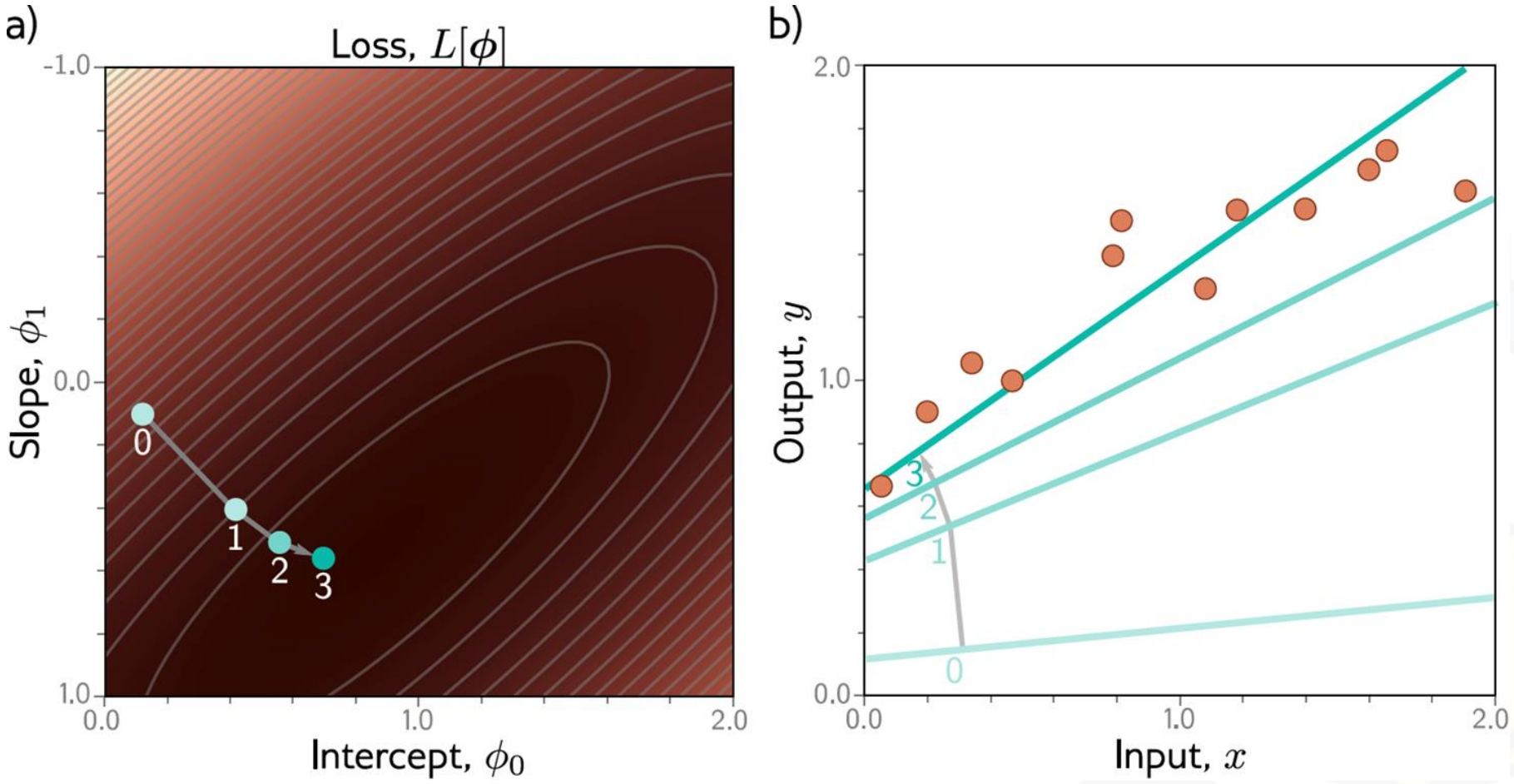
Slide Credit: Simon Prince

Example: 1D Linear regression model



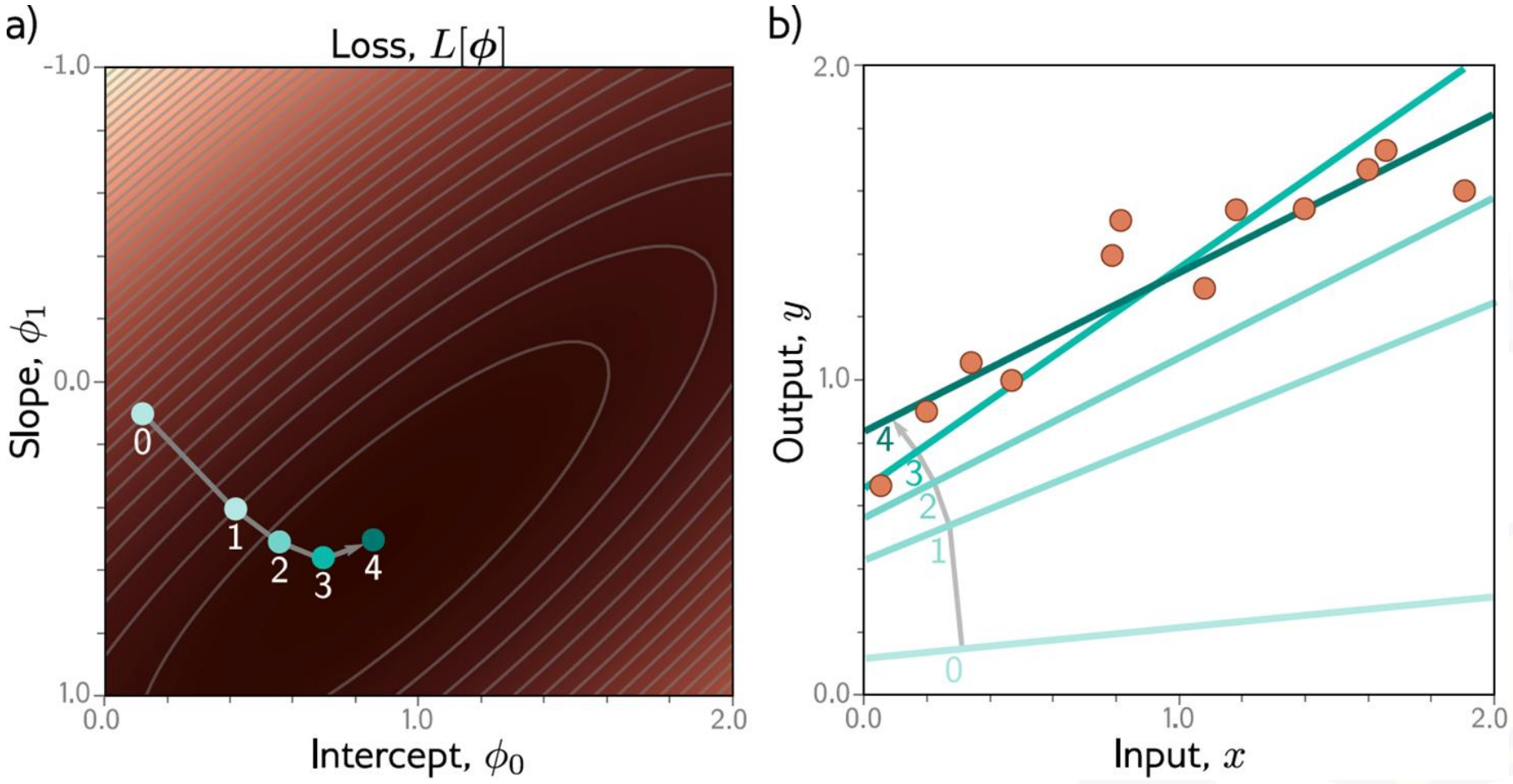
Slide Credit: Simon Prince

Example: 1D Linear regression model



Slide Credit: Simon Prince

Example: 1D Linear regression model



Slide Credit: Simon Prince

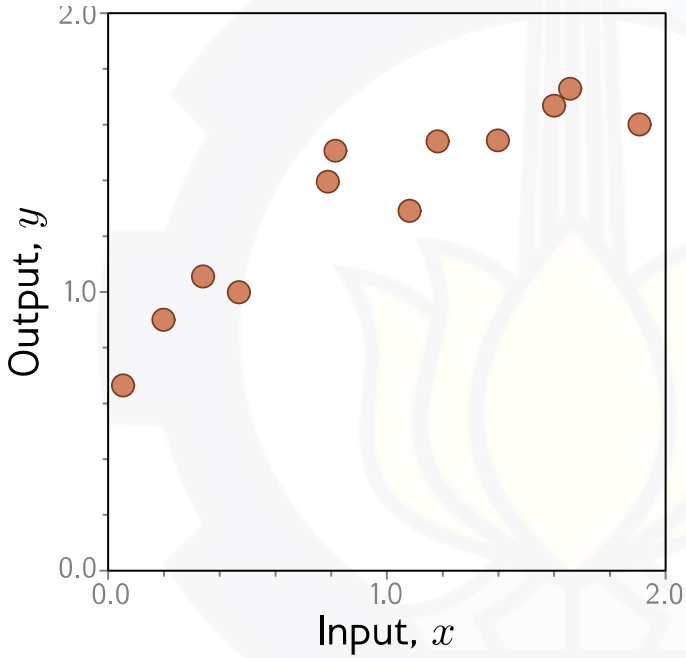
Possible objections

- But you can fit the line model in closed form!
  - Yes – but we won't be able to do this for more complex models
- But we could exhaustively try every slope and intercept combo!
  - Yes – but we won't be able to do this when there are a million parameters

Slide Credit: Simon Prince

Example: 1D Linear regression testing

- Test with different set of paired input/output data
  - Measure performance
  - Degree to which this is same as training = generalization
- Might not generalize well because
  - Model too simple
  - Model too complex
    - fits to statistical peculiarities of data
    - this is known as overfitting



Slide Credit: Simon Prince

Shallow NN vs 1D Linear Regression

- 1D Linear Regression

$$y = \mathrm{f}[x, \boldsymbol{\phi}]$$
$$= \phi_0 + \phi_1 x$$

- Shallow Network

$$y = \mathrm{f}[x, \boldsymbol{\phi}]$$
$$= \phi_0 + \phi_1 \mathrm{a}[\theta_{10} + \theta_{11} x] + \phi_2 \mathrm{a}[\theta_{20} + \theta_{21} x] + \phi_3 \mathrm{a}[\theta_{30} + \theta_{31} x]$$

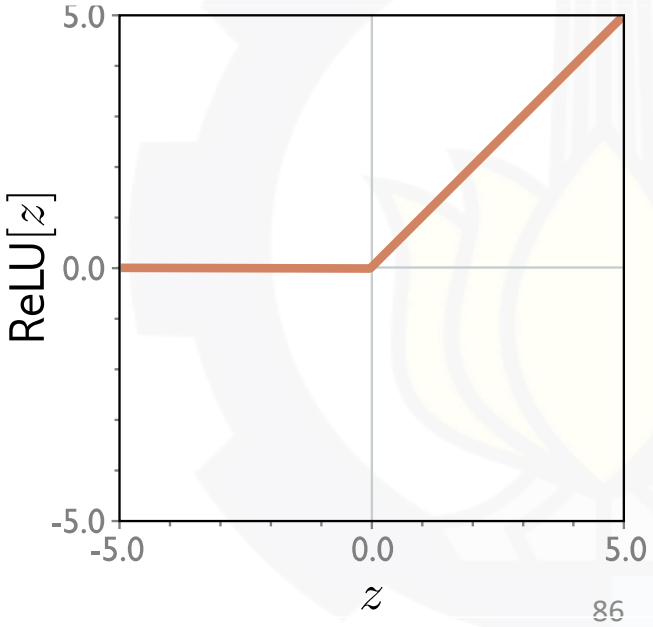
Slide Credit: Simon Prince

Example shallow network

Activation function

$$y = \mathrm{f}[x, \boldsymbol{\phi}]$$
$$= \phi_0 + \phi_1 \mathrm{a}[\theta_{10} + \theta_{11}x] + \phi_2 \mathrm{a}[\theta_{20} + \theta_{21}x] + \phi_3 \mathrm{a}[\theta_{30} + \theta_{31}x]$$

$$\mathrm{a}[z] = \mathrm{ReLU}[z] = \begin{cases} 0 & z < 0 \\ z & z \geq 0 \end{cases}.$$



Rectified Linear Unit

(particular kind of activation function)

Slide Credit: Simon Prince

Example shallow network

$$y = \mathrm{f}[x, \boldsymbol{\phi}]$$
$$= \phi_0 + \phi_1 \mathrm{a}[\theta_{10} + \theta_{11} x] + \phi_2 \mathrm{a}[\theta_{20} + \theta_{21} x] + \phi_3 \mathrm{a}[\theta_{30} + \theta_{31} x]$$

This model has 10 parameters:

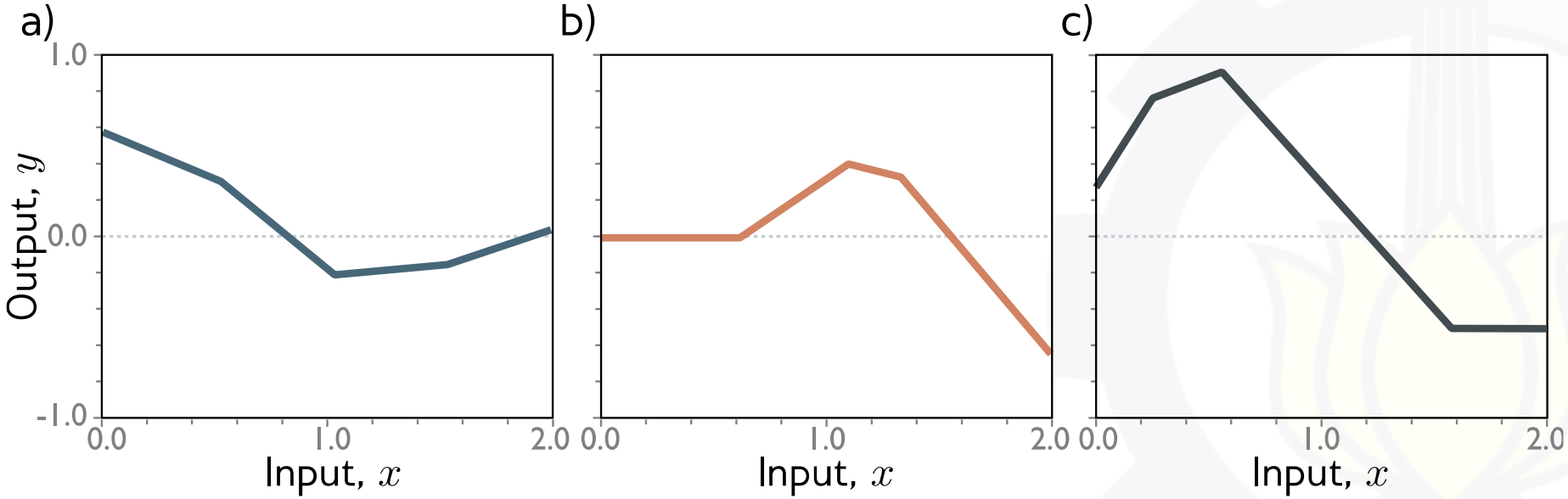
$$\boldsymbol{\phi} = \{\phi_0, \phi_1, \phi_2, \phi_3, \theta_{10}, \theta_{11}, \theta_{20}, \theta_{21}, \theta_{30}, \theta_{31}\}$$

- Represents a family of functions
- Parameters determine particular function
- Given parameters can perform inference (run equation)
- Given training dataset
- Define loss function (least squares)
- Change parameters to minimize loss function

Slide Credit: Simon Prince

Example shallow network

$$y = \phi_0 + \phi_1 \mathrm{a}[\theta_{10} + \theta_{11}x] + \phi_2 \mathrm{a}[\theta_{20} + \theta_{21}x] + \phi_3 \mathrm{a}[\theta_{30} + \theta_{31}x].$$



Piecewise linear functions with three joints

Slide Credit: Simon Prince

Hidden units

$$y = \phi_0 + \phi_1 \mathrm{a}[\theta_{10} + \theta_{11}x] + \phi_2 \mathrm{a}[\theta_{20} + \theta_{21}x] + \phi_3 \mathrm{a}[\theta_{30} + \theta_{31}x].$$

Break down into two parts:

$$y = \phi_0 + \phi_1 h_1 + \phi_2 h_2 + \phi_3 h_3$$

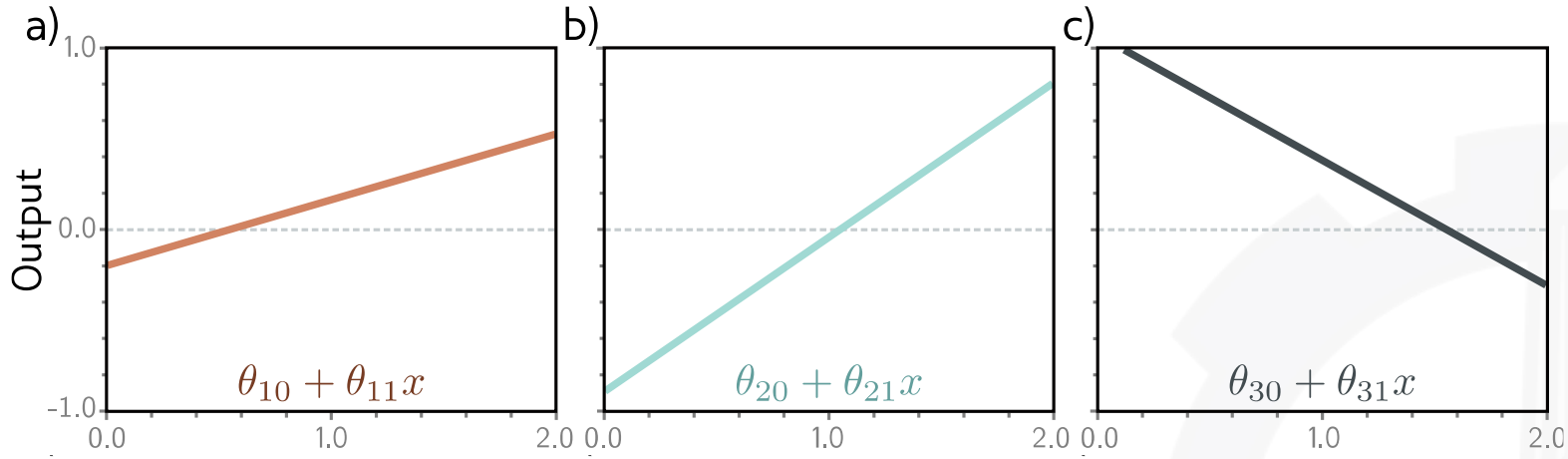
where:

Hidden units

$$\begin{aligned} h_1 &= \mathrm{a}[\theta_{10} + \theta_{11}x] \\ h_2 &= \mathrm{a}[\theta_{20} + \theta_{21}x] \\ h_3 &= \mathrm{a}[\theta_{30} + \theta_{31}x] \end{aligned}$$

Slide Credit: Simon Prince

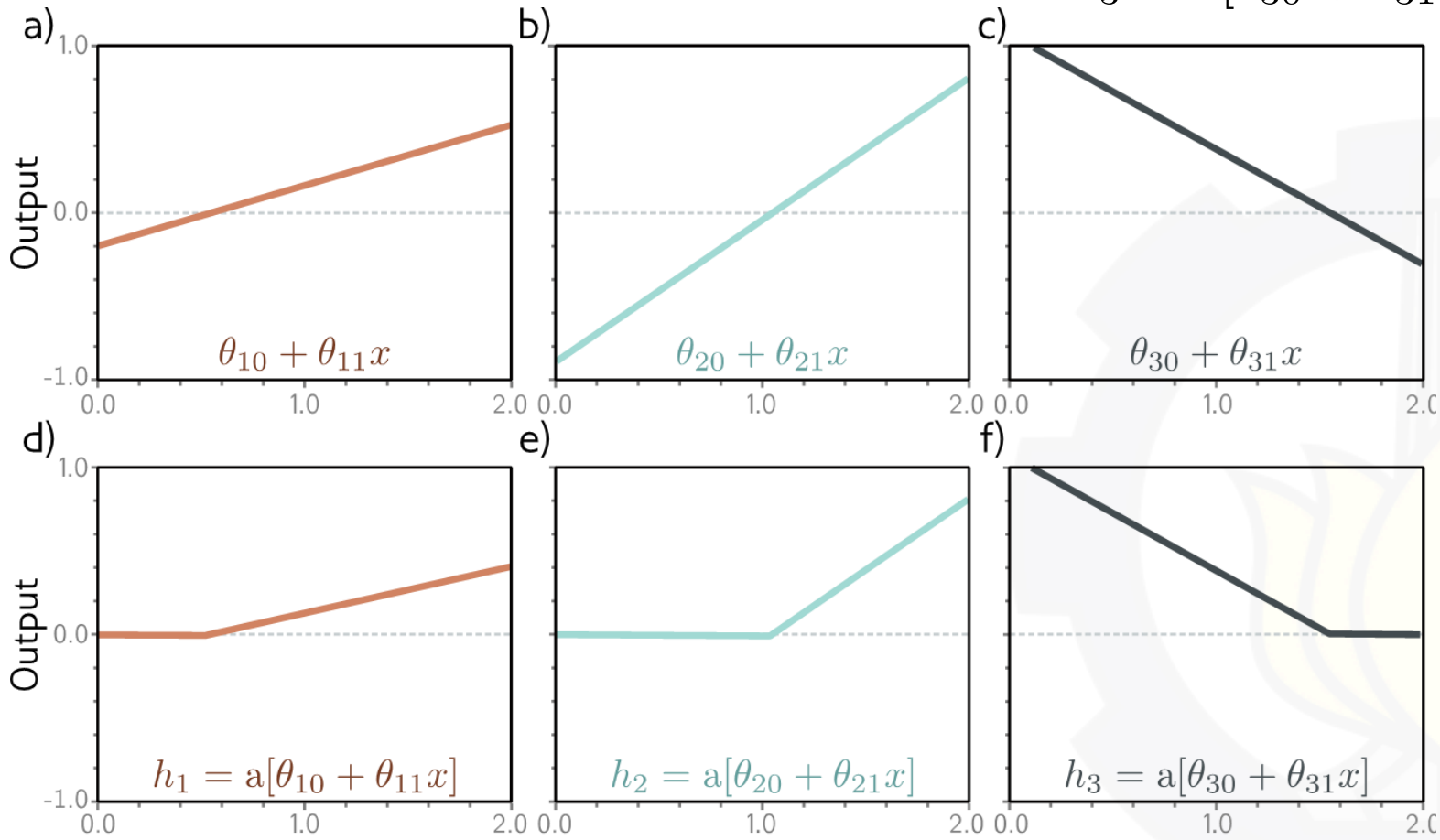
1. compute three linear functions



Slide Credit: Simon Prince

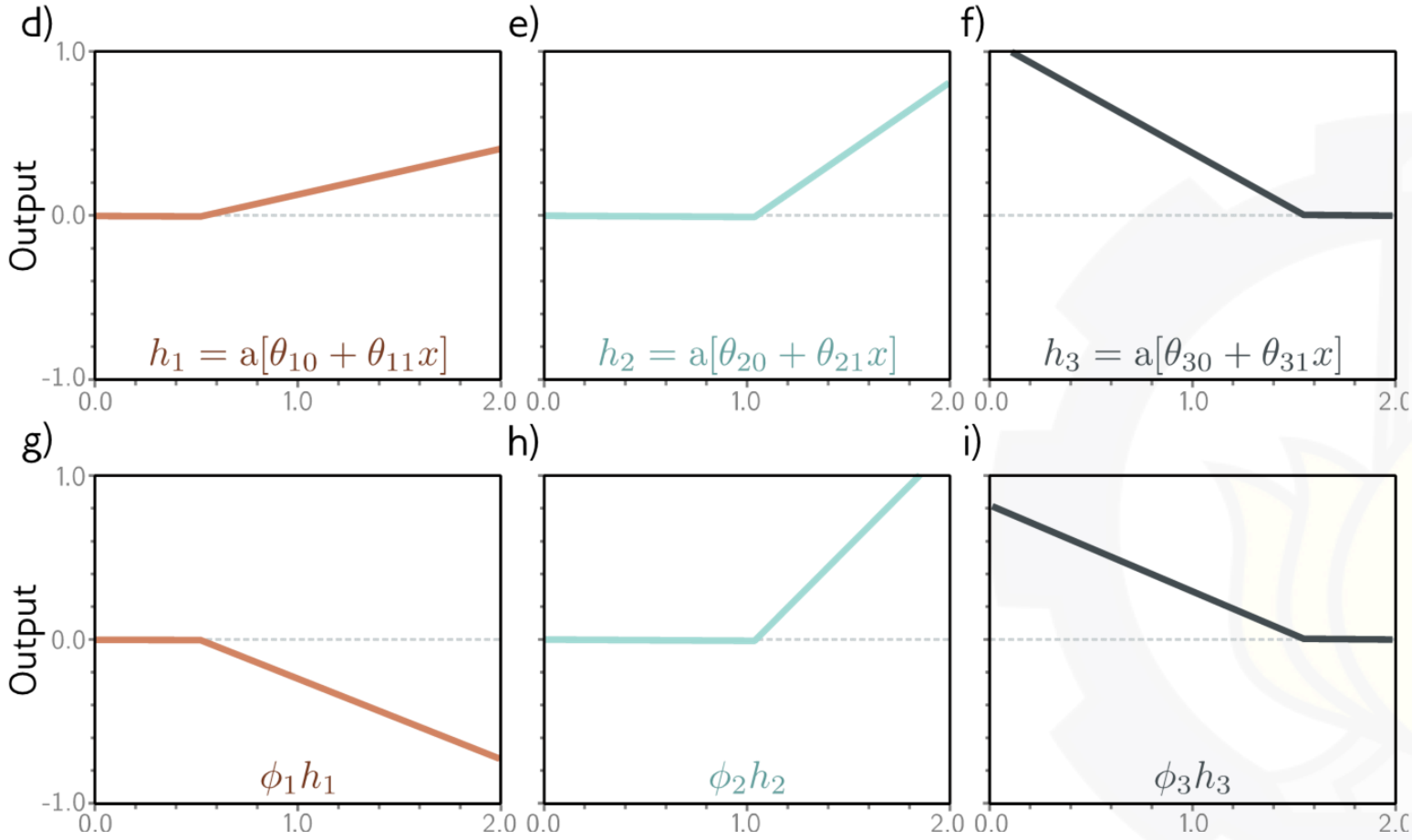
$$h_1 = \mathrm{a}[\theta_{10} + \theta_{11}x]$$
$$h_2 = \mathrm{a}[\theta_{20} + \theta_{21}x]$$
$$h_3 = \mathrm{a}[\theta_{30} + \theta_{31}x],$$

2. Pass through ReLU functions (creates hidden units)



Slide Credit: Simon Prince

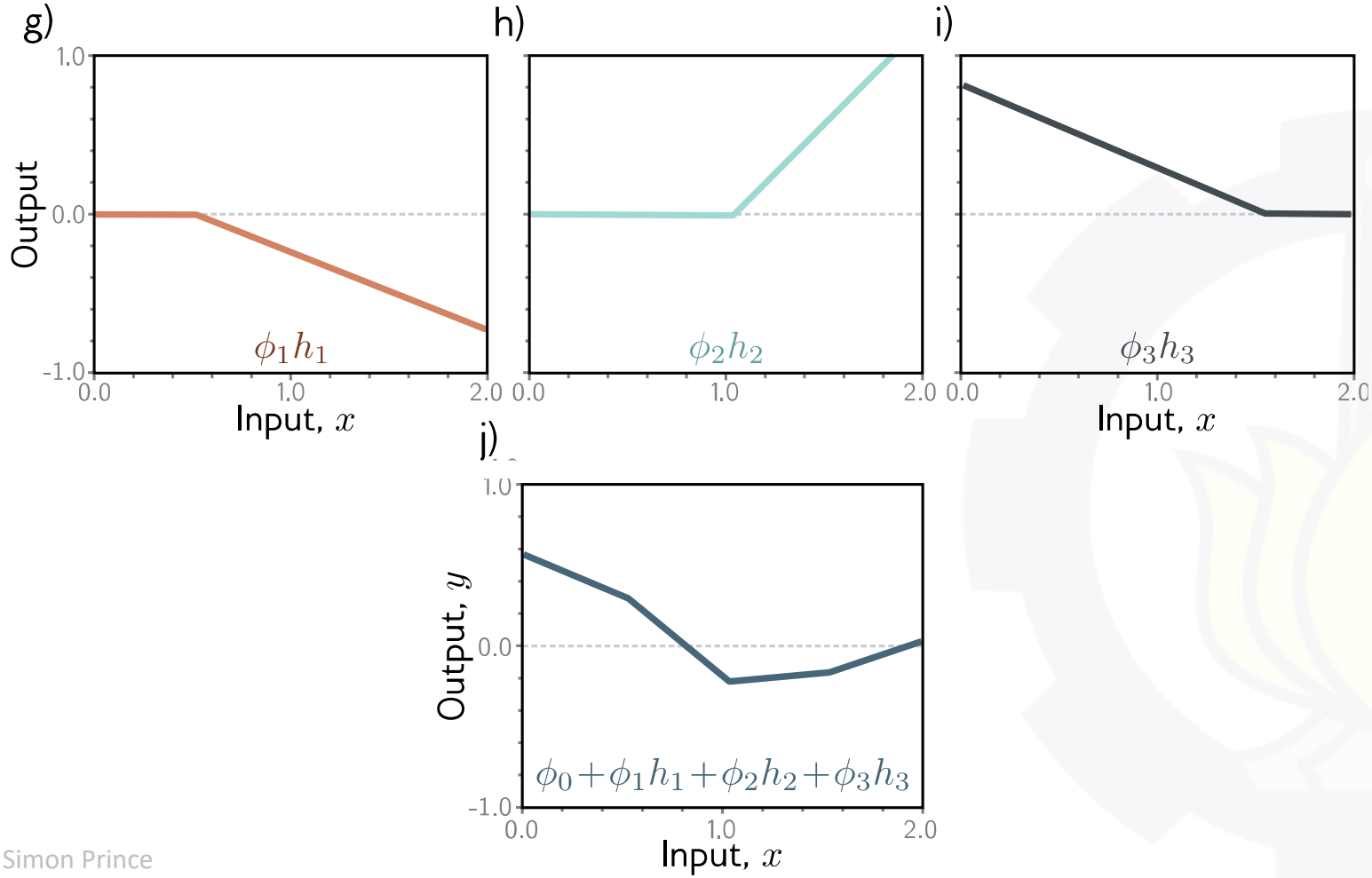
3. Weight the hidden units



Slide Credit: Simon Prince

4. Sum the weighted hidden units to create output

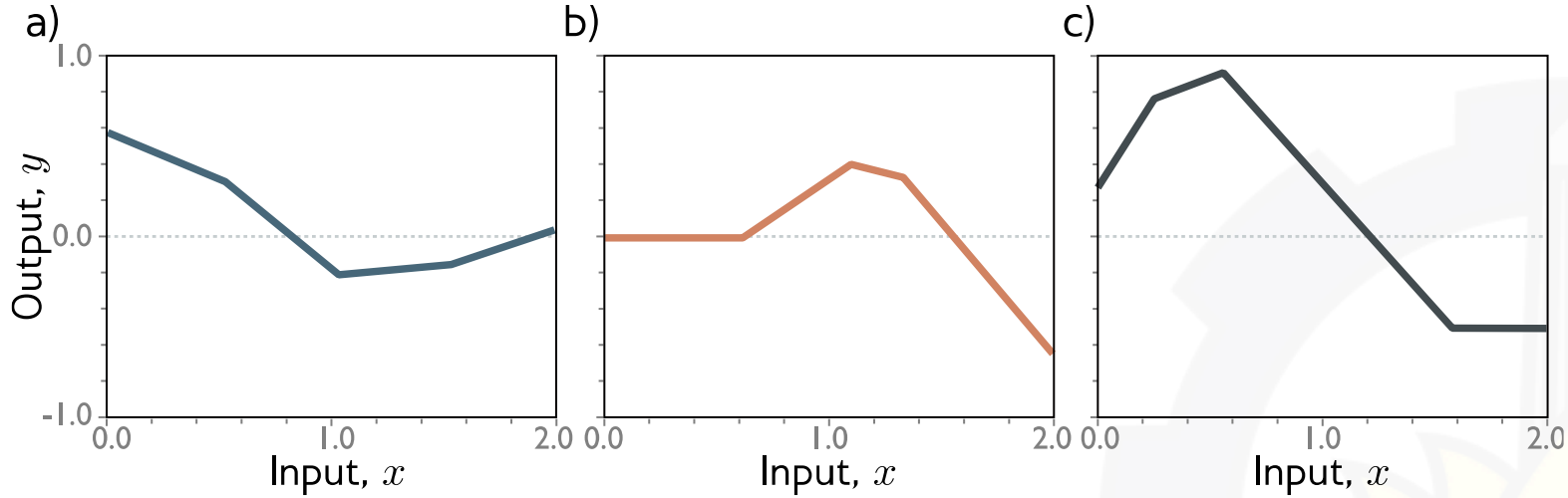
$$y = \phi_0 + \phi_1 h_1 + \phi_2 h_2 + \phi_3 h_3$$



Slide Credit: Simon Prince

Example shallow network

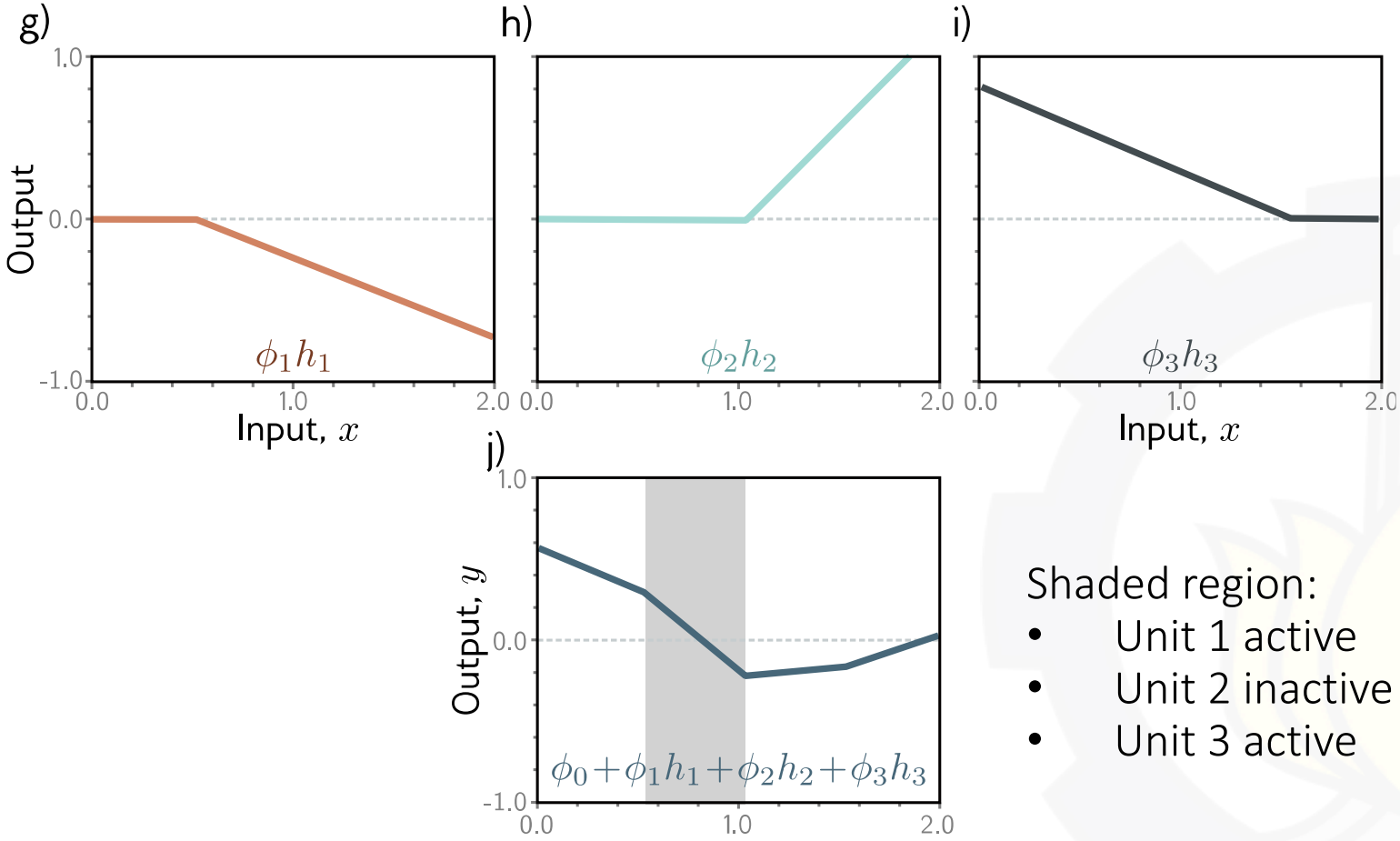
$$y = \phi_0 + \phi_1 \mathrm{a}[\theta_{10} + \theta_{11}x] + \phi_2 \mathrm{a}[\theta_{20} + \theta_{21}x] + \phi_3 \mathrm{a}[\theta_{30} + \theta_{31}x].$$



Example shallow network = piecewise linear functions
1 “joint” per ReLU function

Slide Credit: Simon Prince

Activation pattern = which hidden units are activated



Shaded region:

- Unit 1 active
- Unit 2 inactive
- Unit 3 active

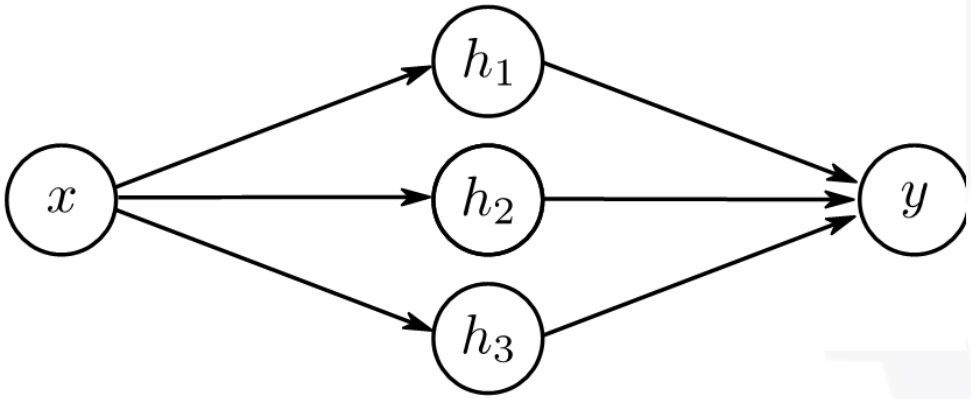
Slide Credit: Simon Prince

Depicting neural networks

$$h_1 = \mathrm{a}[\theta_{10} + \theta_{11}x]$$
$$h_2 = \mathrm{a}[\theta_{20} + \theta_{21}x]$$
$$h_3 = \mathrm{a}[\theta_{30} + \theta_{31}x]$$

$$y = \phi_0 + \phi_1 h_1 + \phi_2 h_2 + \phi_3 h_3$$

b)



Slide Credit: Simon Prince

Shallow Neural Network

- With 3 hidden units:

$$h_1 = \mathrm{a}[\theta_{10} + \theta_{11}x]$$
$$h_2 = \mathrm{a}[\theta_{20} + \theta_{21}x]$$
$$h_3 = \mathrm{a}[\theta_{30} + \theta_{31}x]$$

$$y = \phi_0 + \phi_1 h_1 + \phi_2 h_2 + \phi_3 h_3$$

- With D hidden units:

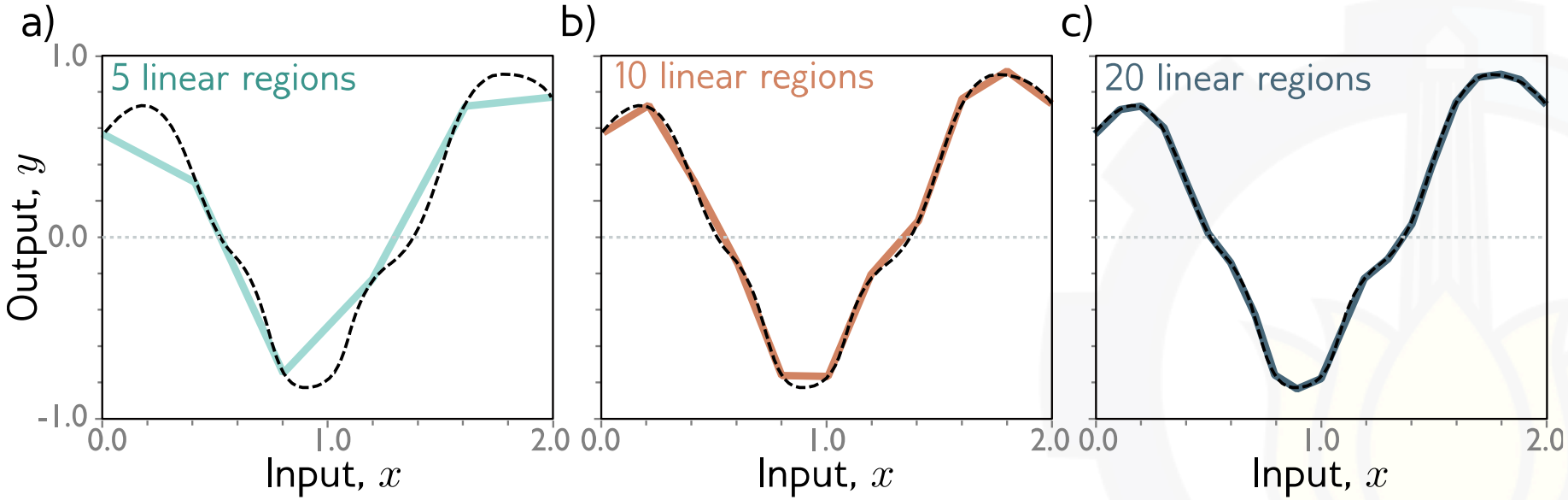
$$h_d = \mathrm{a}[\theta_{d0} + \theta_{d1}x]$$

$$y = \phi_0 + \sum_{d=1}^{D} \phi_d h_d$$

Slide Credit: Simon Prince

With enough hidden units…

… we can describe any 1D function to arbitrary accuracy



Slide Credit: Simon Prince

Universal approximation theorem

"a formal proof that, with enough hidden units, a shallow neural network can describe any continuous function on a compact subset of $\mathbb{R}^D$ to arbitrary precision"

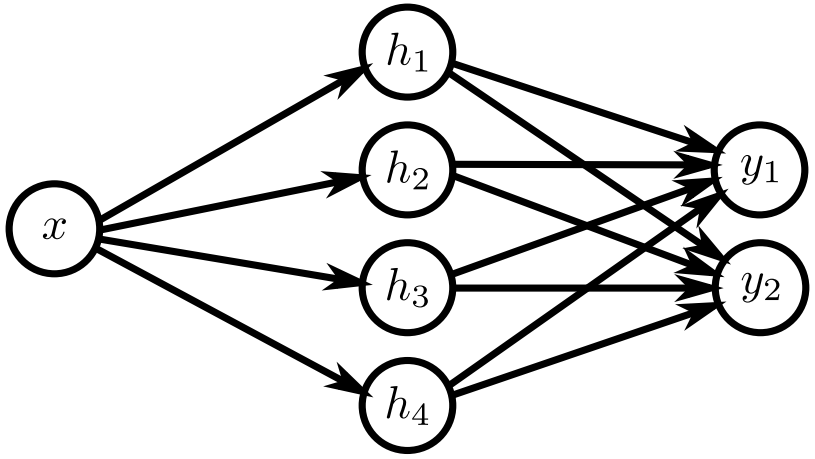
Slide Credit: Simon Prince

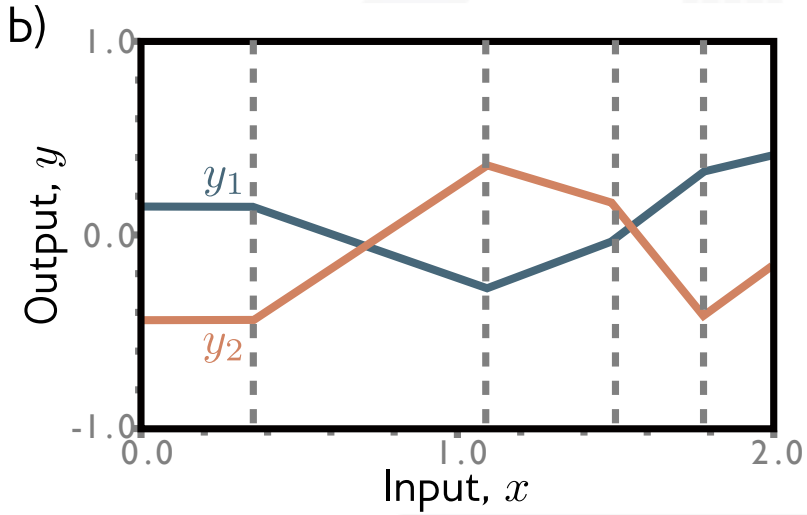
Two outputs

- 1 input, 4 hidden units, 2 outputs

$$h_1 = \mathrm{a}[\theta_{10} + \theta_{11}x]$$
$$h_2 = \mathrm{a}[\theta_{20} + \theta_{21}x]$$
$$h_3 = \mathrm{a}[\theta_{30} + \theta_{31}x]$$
$$h_4 = \mathrm{a}[\theta_{40} + \theta_{41}x]$$

$$y_1 = \phi_{10} + \phi_{11}h_1 + \phi_{12}h_2 + \phi_{13}h_3 + \phi_{14}h_4$$
$$y_2 = \phi_{20} + \phi_{21}h_1 + \phi_{22}h_2 + \phi_{23}h_3 + \phi_{24}h_4$$





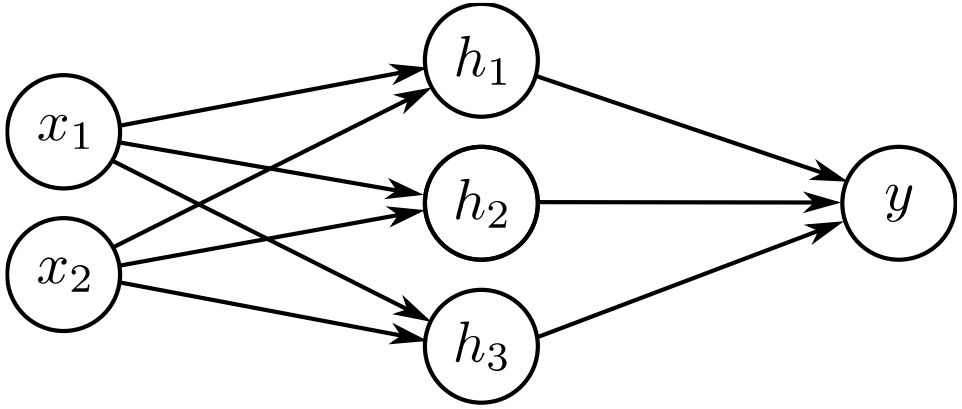
Slide Credit: Simon Prince

Two inputs

- 2 inputs, 3 hidden units, 1 output

$$h_1 = \mathrm{a}[\theta_{10} + \theta_{11}x_1 + \theta_{12}x_2]$$
$$h_2 = \mathrm{a}[\theta_{20} + \theta_{21}x_1 + \theta_{22}x_2]$$
$$h_3 = \mathrm{a}[\theta_{30} + \theta_{31}x_1 + \theta_{32}x_2]$$

$$y = \phi_0 + \phi_1 h_1 + \phi_2 h_2 + \phi_3 h_3$$



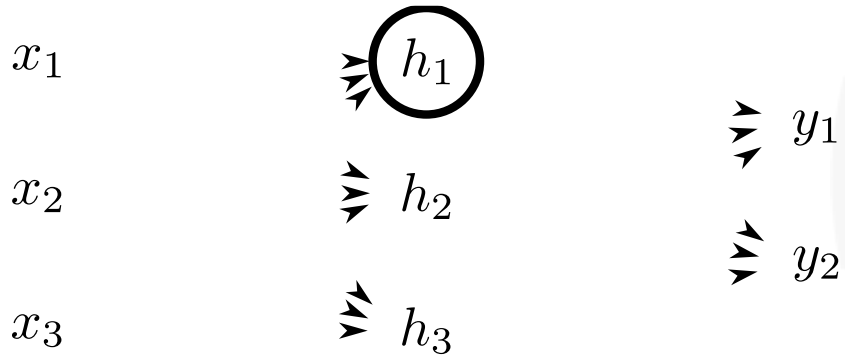
Slide Credit: Simon Prince

Arbitrary inputs, hidden units, outputs

- D_o Outputs, D hidden units, and D_i inputs

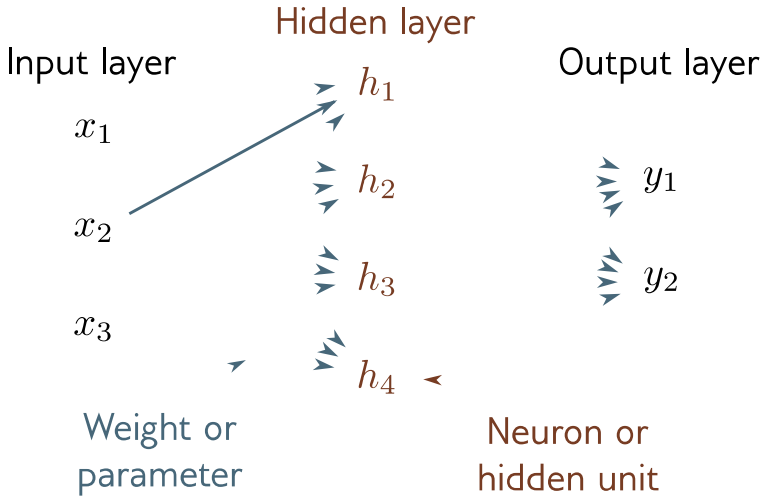
$$h_d = \mathrm{a}\left[\theta_{d0} + \sum_{i=1}^{D_i} \theta_{di} x_i\right] \qquad y_j = \phi_{j0} + \sum_{d=1}^{D} \phi_{jd} h_d$$

- e.g., Three inputs, three hidden units, two outputs



Slide Credit: Simon Prince

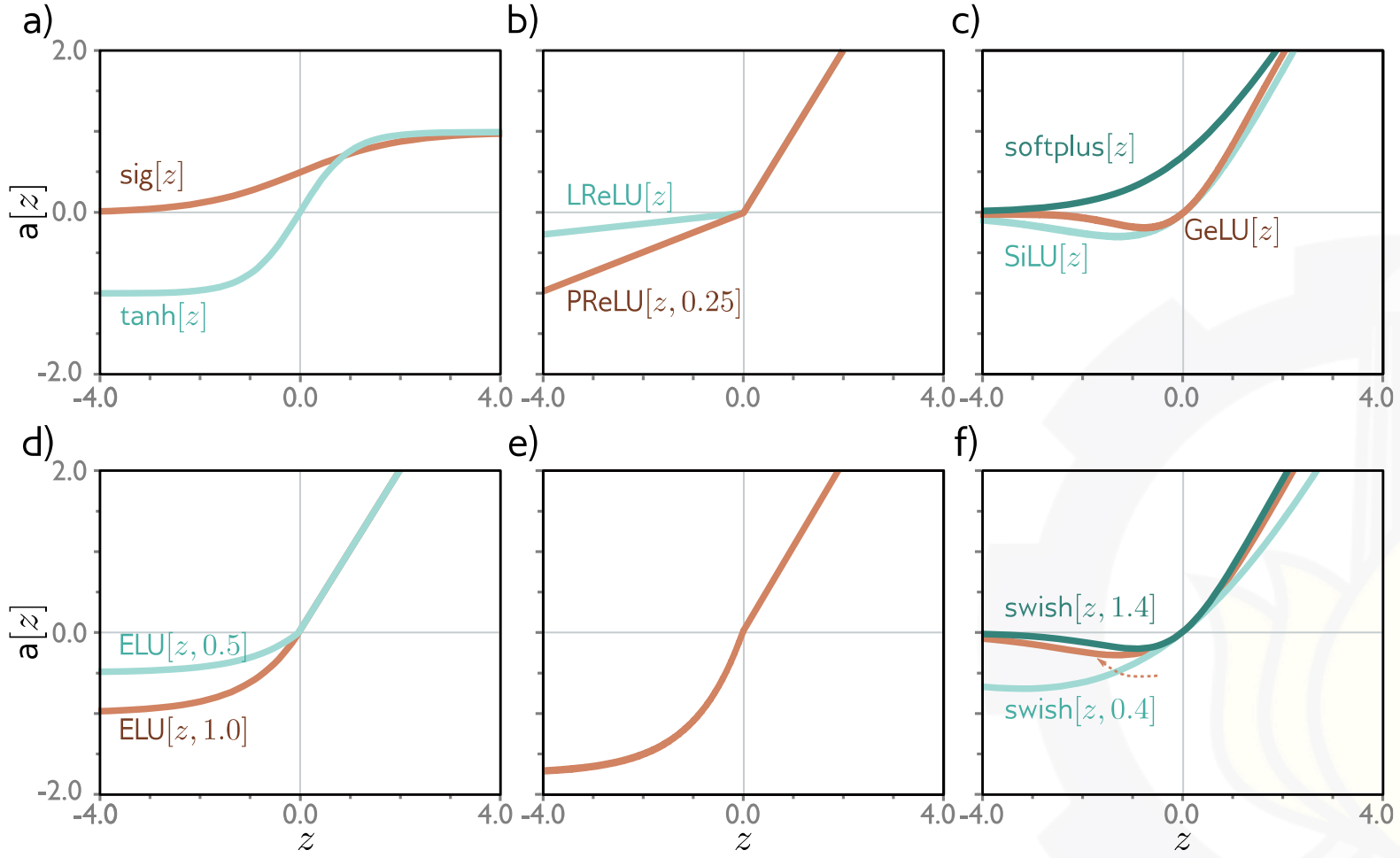
Nomenclature



- Y-offsets = biases
- Slopes = weights
- Everything in one layer connected to everything in the next = fully connected network
- No loops = feedforward network
- Values after ReLU (activation functions) = activations
- Values before ReLU = pre-activations
- One hidden layer = shallow neural network
- More than one hidden layer = deep neural network
- Number of hidden units ≈ capacity

Slide Credit: Simon Prince

Other activation functions



Slide Credit: Simon Prince

Outline

- What is deep learning?
- Deep learning application
- Simple neural network classifier
- **Deep learning classifier**

Deep Neural Network

- Networks with more than one hidden layer
- Intuition becomes more difficult

Slide Credit: Simon Prince

Composing two networks.

Network 1:

$$h_1 = \mathrm{a}[\theta_{10} + \theta_{11}x]$$
$$h_2 = \mathrm{a}[\theta_{20} + \theta_{21}x]$$
$$h_3 = \mathrm{a}[\theta_{30} + \theta_{31}x]$$

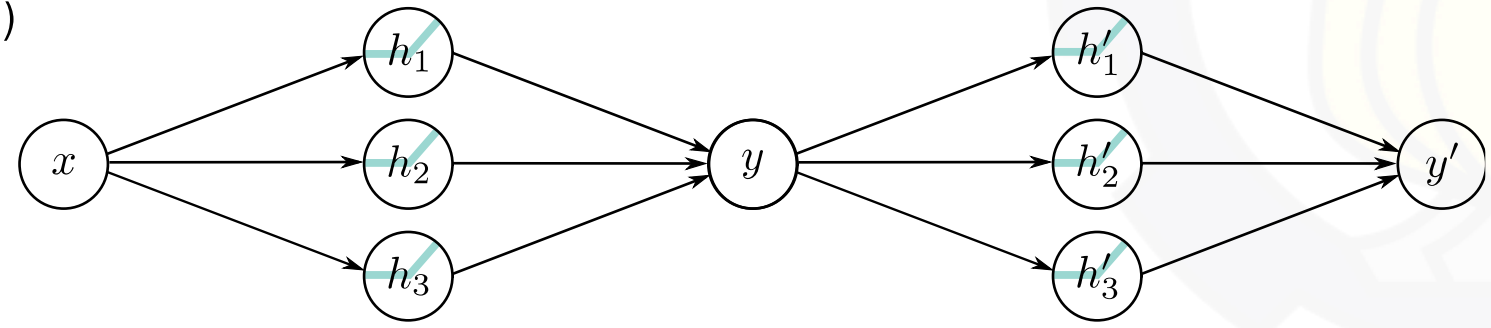
$$y = \phi_0 + \phi_1 h_1 + \phi_2 h_2 + \phi_3 h_3$$

Network 2:

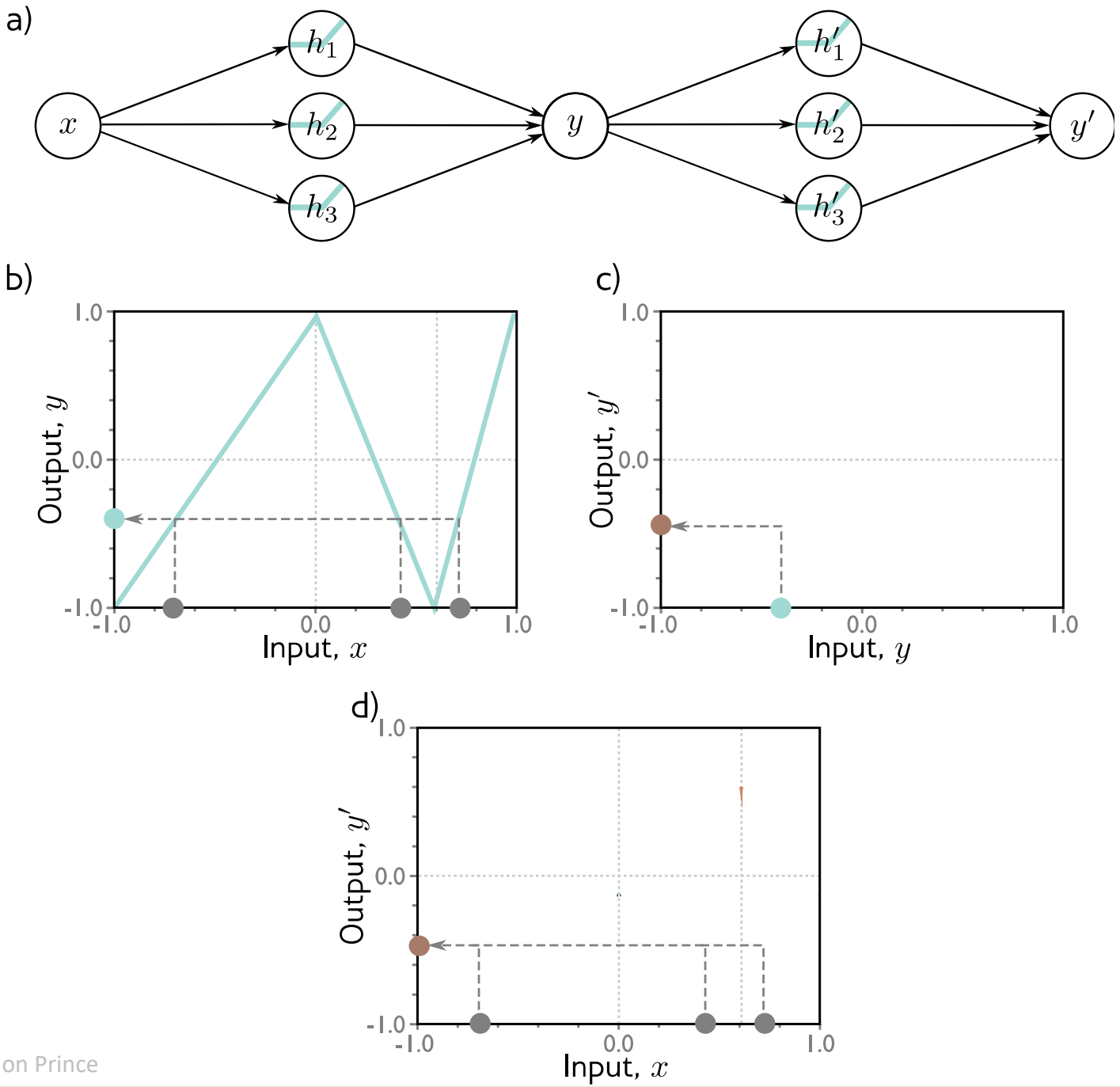
$$h'_1 = \mathrm{a}[\theta'_{10} + \theta'_{11}y]$$
$$h'_2 = \mathrm{a}[\theta'_{20} + \theta'_{21}y]$$
$$h'_3 = \mathrm{a}[\theta'_{30} + \theta'_{31}y]$$

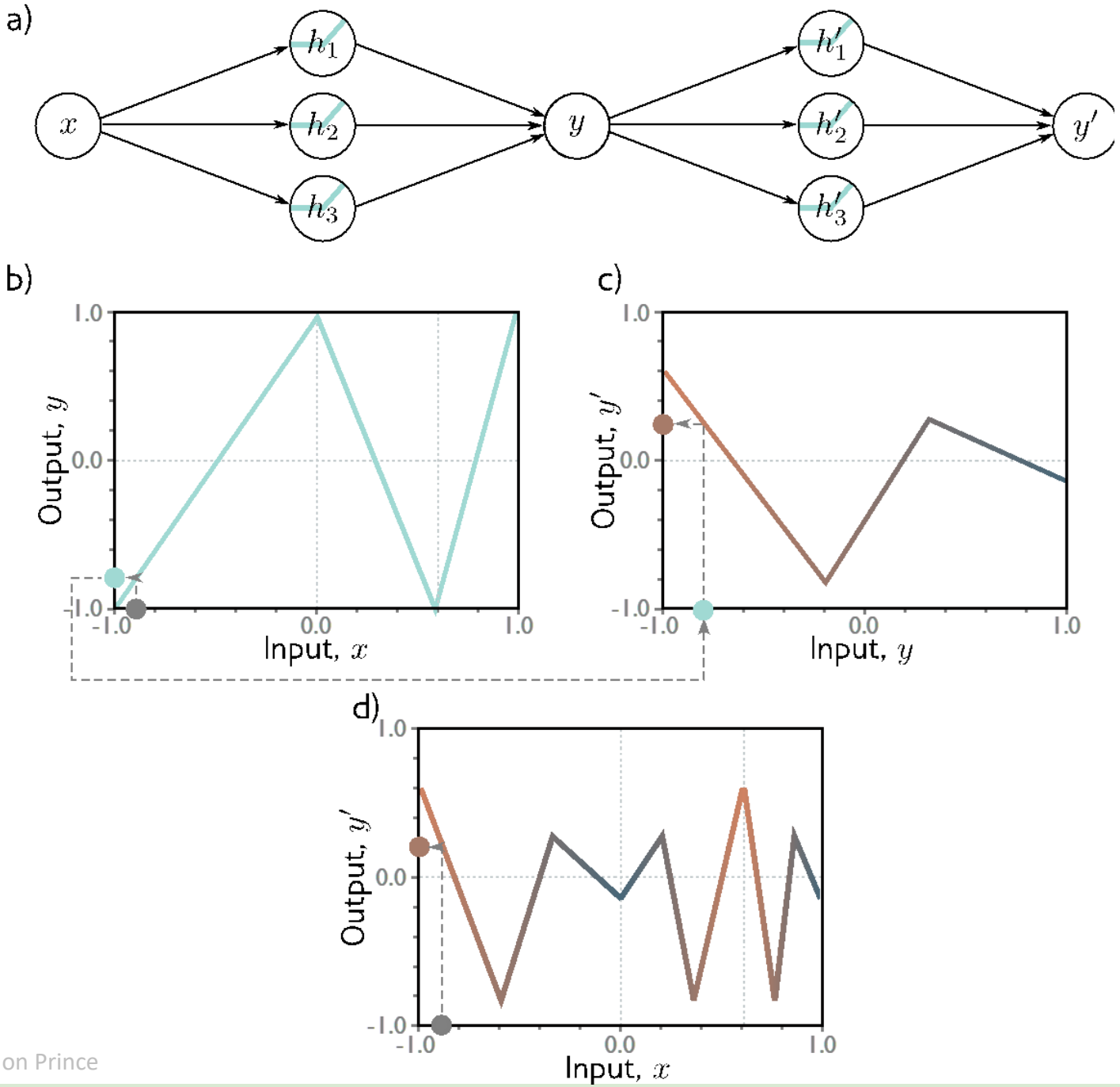
$$y' = \phi'_0 + \phi'_1 h'_1 + \phi'_2 h'_2 + \phi'_3 h'_3$$

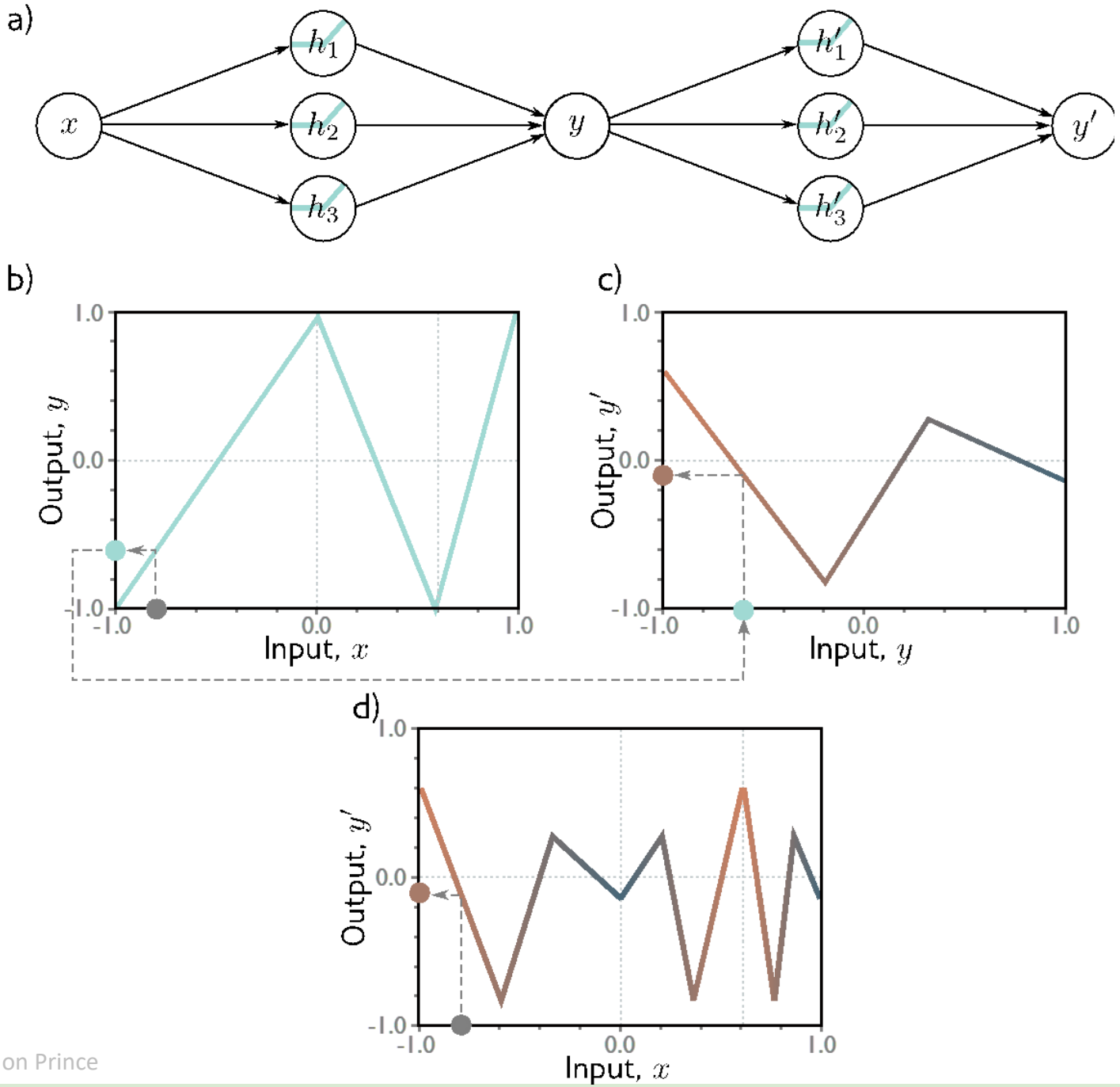
a)

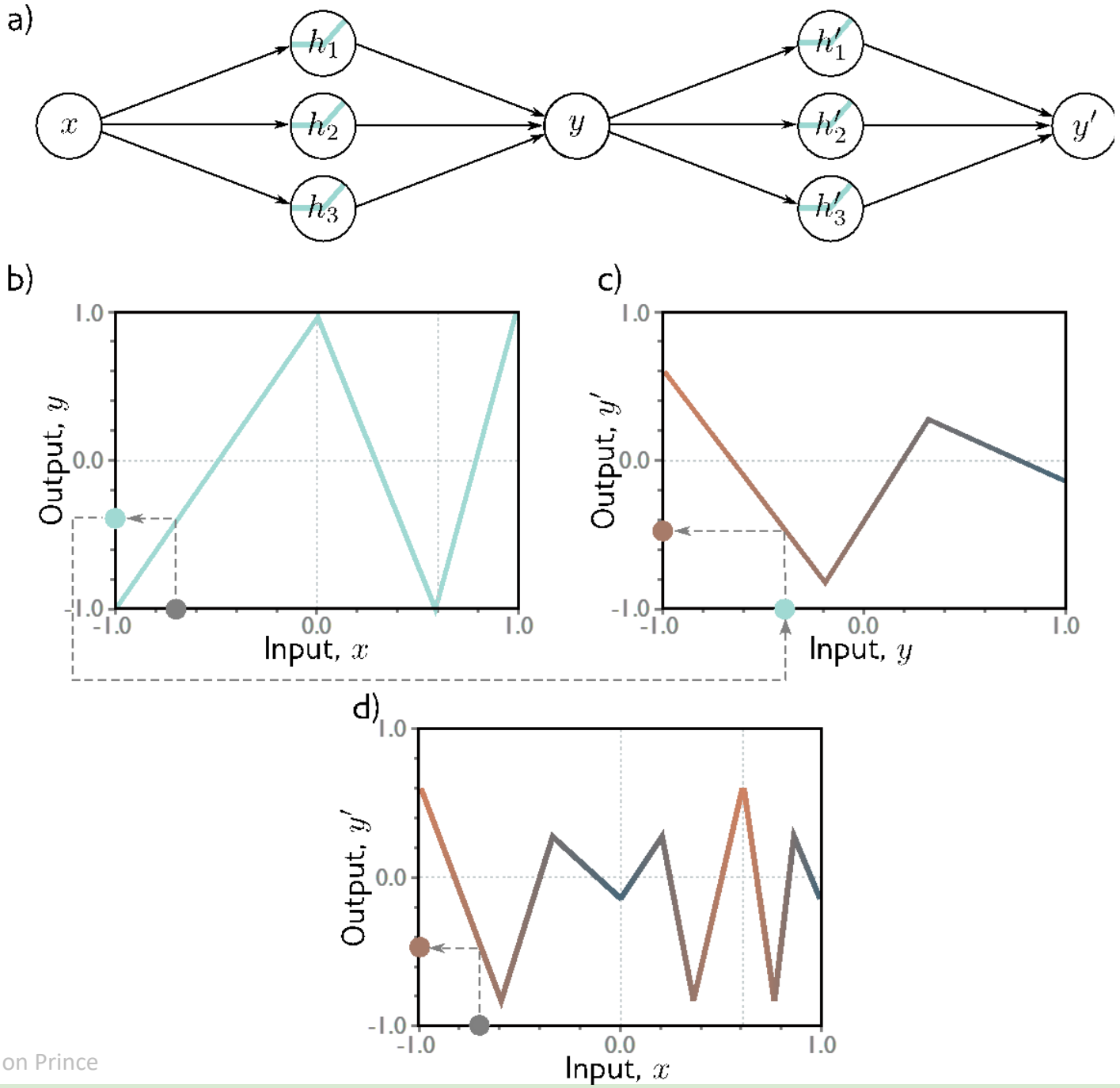


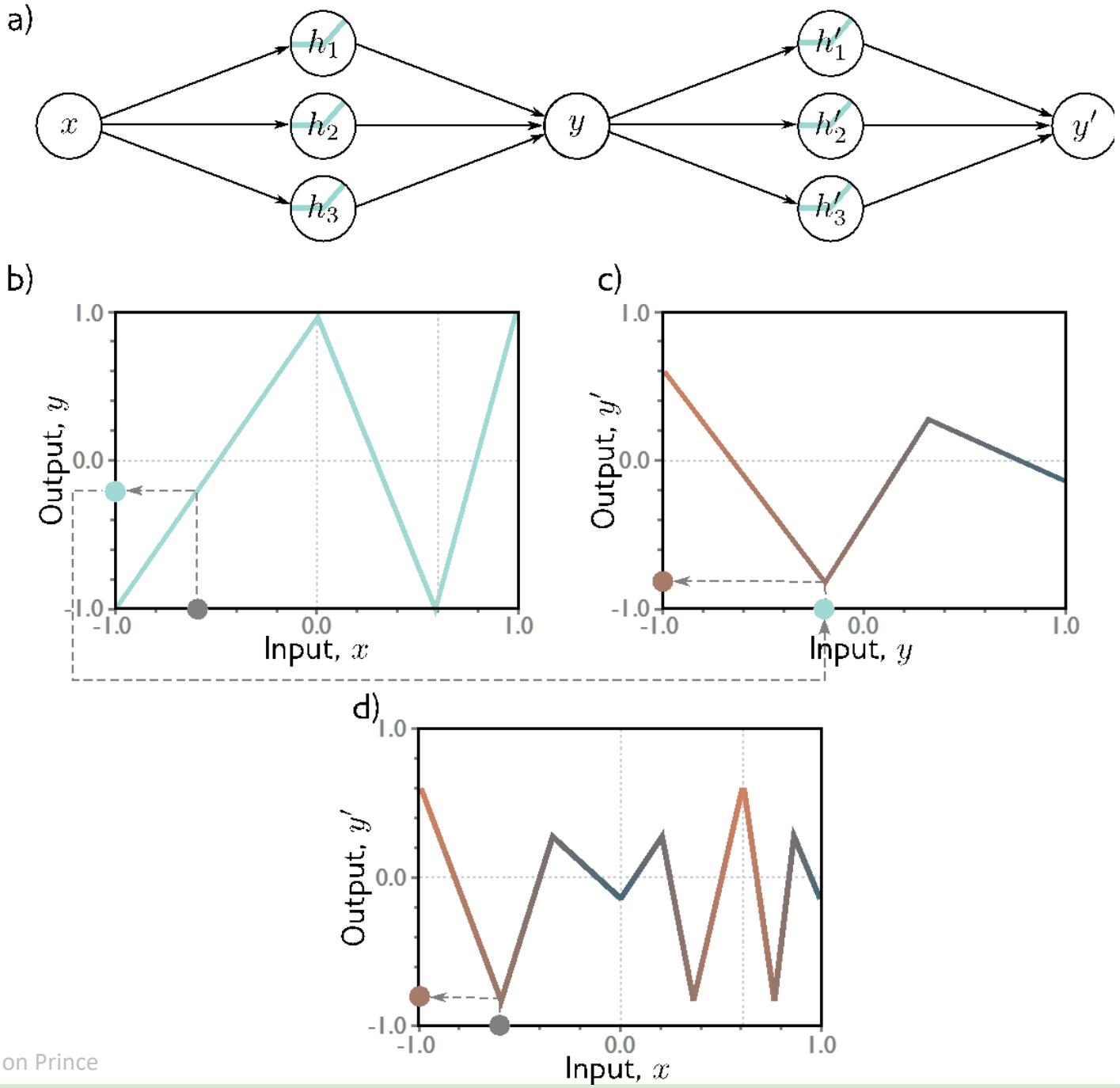
Slide Credit: Simon Prince


a)
x
h_1
h_2
h_3
y
h'_1
h'_2
h'_3
y'
b)
Output, y
Input, x
1.0
0.0
-1.0
c)
Output, y'
Input, y
d)
Output, y'
Input, x


a)
x
h_1
h_2
h_3
y
h'_1
h'_2
h'_3
y'
b)
Output, y
1.0
0.0
-1.0
-1.0
0.0
1.0
Input, x
c)
Output, y'
1.0
0.0
-1.0
-1.0
0.0
1.0
Input, y
d)
Output, y'
1.0
0.0
-1.0
-1.0
0.0
1.0
Input, x


a)
x
h_1
h_2
h_3
y
h'_1
h'_2
h'_3
y'
b)
Output, y
Input, x
c)
Output, y'
Input, y
d)
Output, y'
Input, x
1.0
0.0
-1.0


a)
x
h_1
h_2
h_3
y
h'_1
h'_2
h'_3
y'
b)
Output, y
Input, x
1.0
0.0
-1.0
c)
Output, y'
Input, y
d)
Output, y'
Input, x


a)
x
h_1
h_2
h_3
y
h'_1
h'_2
h'_3
y'
b)
Output, y
Input, x
1.0
0.0
-1.0
c)
Output, y'
Input, y
d)
Output, y'
Input, x

Combine two networks into one

Network 1:

$$h_1 = \mathrm{a}[\theta_{10} + \theta_{11}x]$$
$$h_2 = \mathrm{a}[\theta_{20} + \theta_{21}x]$$
$$h_3 = \mathrm{a}[\theta_{30} + \theta_{31}x]$$

$$y = \phi_0 + \phi_1 h_1 + \phi_2 h_2 + \phi_3 h_3$$

Network 2:

$$h'_1 = \mathrm{a}[\theta'_{10} + \theta'_{11}y]$$
$$h'_2 = \mathrm{a}[\theta'_{20} + \theta'_{21}y]$$
$$h'_3 = \mathrm{a}[\theta'_{30} + \theta'_{31}y]$$

$$y' = \phi'_0 + \phi'_1 h'_1 + \phi'_2 h'_2 + \phi'_3 h'_3$$

Hidden units of second network in terms of first:

$$h'_1 = \mathrm{a}[\theta'_{10} + \theta'_{11}y] = \mathrm{a}[\theta'_{10} + \theta'_{11}\phi_0 + \theta'_{11}\phi_1 h_1 + \theta'_{11}\phi_2 h_2 + \theta'_{11}\phi_3 h_3]$$
$$h'_2 = \mathrm{a}[\theta'_{20} + \theta'_{21}y] = \mathrm{a}[\theta'_{20} + \theta'_{21}\phi_0 + \theta'_{21}\phi_1 h_1 + \theta'_{21}\phi_2 h_2 + \theta'_{21}\phi_3 h_3]$$
$$h'_3 = \mathrm{a}[\theta'_{30} + \theta'_{31}y] = \mathrm{a}[\theta'_{30} + \theta'_{31}\phi_0 + \theta'_{31}\phi_1 h_1 + \theta'_{31}\phi_2 h_2 + \theta'_{31}\phi_3 h_3]$$

Slide Credit: Simon Prince

Create new variables

$$
\begin{aligned}
h_1' = \quad & \mathrm{a}[\theta_{10}' + \theta_{11}' y] \quad = \quad \mathrm{a}[\theta_{10}' + \theta_{11}'\phi_0 + \theta_{11}'\phi_1 h_1 + \theta_{11}'\phi_2 h_2 + \theta_{11}'\phi_3 h_3] \\
h_2' = \quad & \mathrm{a}[\theta_{20}' + \theta_{21}' y] \quad = \quad \mathrm{a}[\theta_{20}' + \theta_{21}'\phi_0 + \theta_{21}'\phi_1 h_1 + \theta_{21}'\phi_2 h_2 + \theta_{21}'\phi_3 h_3] \\
h_3' = \quad & \mathrm{a}[\theta_{30}' + \theta_{31}' y] \quad = \quad \mathrm{a}[\theta_{30}' + \theta_{31}'\phi_0 + \theta_{31}'\phi_1 h_1 + \theta_{31}'\phi_2 h_2 + \theta_{31}'\phi_3 h_3]
\end{aligned}
$$

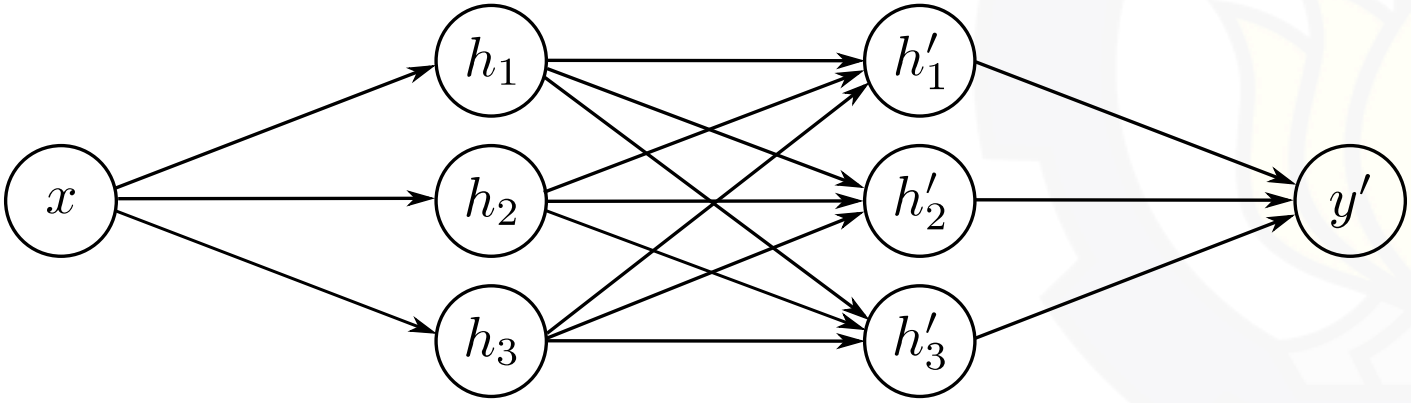
$$
\begin{aligned}
h_1' &= \mathrm{a}[\psi_{10} + \psi_{11} h_1 + \psi_{12} h_2 + \psi_{13} h_3] \\
h_2' &= \mathrm{a}[\psi_{20} + \psi_{21} h_1 + \psi_{22} h_2 + \psi_{23} h_3] \\
h_3' &= \mathrm{a}[\psi_{30} + \psi_{31} h_1 + \psi_{32} h_2 + \psi_{33} h_3]
\end{aligned}
$$

Slide Credit: Simon Prince

Two-layer network

$$h_1 = \mathrm{a}[\theta_{10} + \theta_{11}x] \qquad h'_1 = \mathrm{a}[\psi_{10} + \psi_{11}h_1 + \psi_{12}h_2 + \psi_{13}h_3]$$
$$h_2 = \mathrm{a}[\theta_{20} + \theta_{21}x] \qquad h'_2 = \mathrm{a}[\psi_{20} + \psi_{21}h_2 + \psi_{22}h_2 + \psi_{23}h_3]$$
$$h_3 = \mathrm{a}[\theta_{30} + \theta_{31}x] \qquad h'_3 = \mathrm{a}[\psi_{30} + \psi_{31}h_2 + \psi_{32}h_2 + \psi_{33}h_3]$$

$$y' = \phi'_0 + \phi'_1 h'_1 + \phi'_2 h'_2 + \phi'_3 h'_3$$



Slide Credit: Simon Prince

Two-layer network as one equation

$$h_1 = \mathrm{a}[\theta_{10} + \theta_{11}x] \qquad h'_1 = \mathrm{a}[\psi_{10} + \psi_{11}h_1 + \psi_{12}h_2 + \psi_{13}h_3]$$
$$h_2 = \mathrm{a}[\theta_{20} + \theta_{21}x] \qquad h'_2 = \mathrm{a}[\psi_{20} + \psi_{21}h_2 + \psi_{22}h_2 + \psi_{23}h_3]$$
$$h_3 = \mathrm{a}[\theta_{30} + \theta_{31}x] \qquad h'_3 = \mathrm{a}[\psi_{30} + \psi_{31}h_2 + \psi_{32}h_2 + \psi_{33}h_3]$$

$$y' = \phi'_0 + \phi'_1 h'_1 + \phi'_2 h'_2 + \phi'_3 h'_3$$

$$\begin{aligned} y' = \phi'_0 &+ \phi'_1 \mathrm{a}\left[\psi_{10} + \psi_{11}\mathrm{a}[\theta_{10} + \theta_{11}x] + \psi_{12}\mathrm{a}[\theta_{20} + \theta_{21}x] + \psi_{13}\mathrm{a}[\theta_{30} + \theta_{31}x]\right] \\ &+ \phi'_2 \mathrm{a}[\psi_{20} + \psi_{21}\mathrm{a}[\theta_{10} + \theta_{11}x] + \psi_{22}\mathrm{a}[\theta_{20} + \theta_{21}x] + \psi_{23}\mathrm{a}[\theta_{30} + \theta_{31}x]] \\ &+ \phi'_3 \mathrm{a}[\psi_{30} + \psi_{31}\mathrm{a}[\theta_{10} + \theta_{11}x] + \psi_{32}\mathrm{a}[\theta_{20} + \theta_{21}x] + \psi_{33}\mathrm{a}[\theta_{30} + \theta_{31}x]] \end{aligned}$$

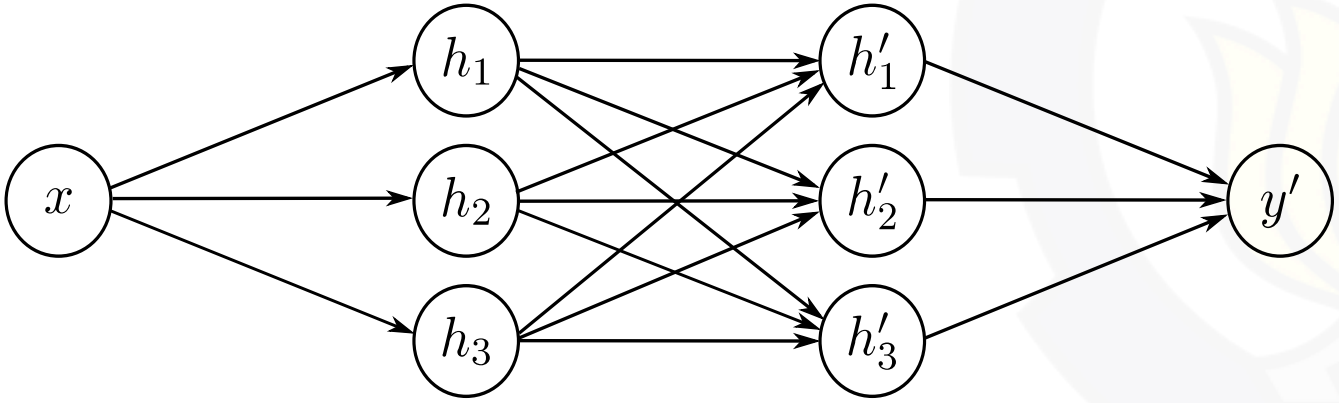
Slide Credit: Simon Prince

Networks as composing functions

$$h_1 = \mathrm{a}[\theta_{10} + \theta_{11}x]$$
$$h_2 = \mathrm{a}[\theta_{20} + \theta_{21}x]$$
$$h_3 = \mathrm{a}[\theta_{30} + \theta_{31}x]$$

$$h'_1 = \mathrm{a}[\psi_{10} + \psi_{11}h_1 + \psi_{12}h_2 + \psi_{13}h_3]$$
$$h'_2 = \mathrm{a}[\psi_{20} + \psi_{21}h_1 + \psi_{22}h_2 + \psi_{23}h_3]$$
$$h'_3 = \mathrm{a}[\psi_{30} + \psi_{31}h_1 + \psi_{32}h_2 + \psi_{33}h_3]$$

Consider the pre-activations at the second hidden units
At this point, it's a one--layer network with three outputs



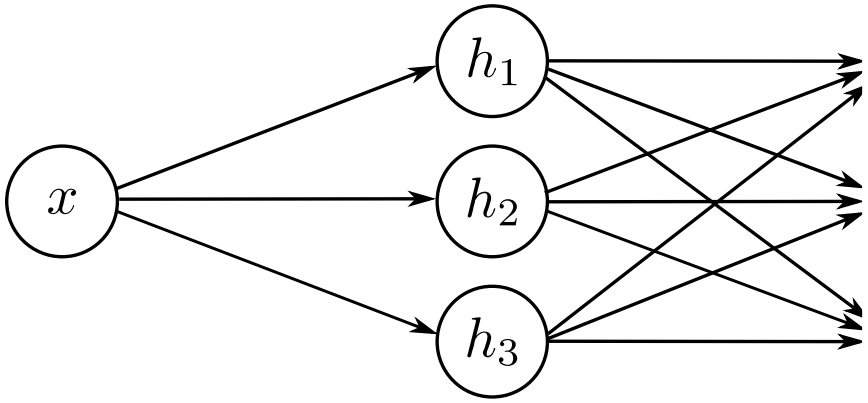
Slide Credit: Simon Prince

Networks as composing functions

$$h_1 = \mathrm{a}[\theta_{10} + \theta_{11}x]$$
$$h_2 = \mathrm{a}[\theta_{20} + \theta_{21}x]$$
$$h_3 = \mathrm{a}[\theta_{30} + \theta_{31}x]$$

$$h'_1 = \mathrm{a}[\psi_{10} + \psi_{11}h_1 + \psi_{12}h_2 + \psi_{13}h_3]$$
$$h'_2 = \mathrm{a}[\psi_{20} + \psi_{21}h_1 + \psi_{22}h_2 + \psi_{23}h_3]$$
$$h'_3 = \mathrm{a}[\psi_{30} + \psi_{31}h_1 + \psi_{32}h_2 + \psi_{33}h_3]$$

Consider the pre-activations at the second hidden units
At this point, it's a one--layer network with three outputs



Slide Credit: Simon Prince

Hyperparameters

- K layers = depth of network
- D_k hidden units per layer = width of network

- These are called hyperparameters – chosen before training the network
- Can try retraining with different hyperparameters – hyperparameter optimization or hyperparameter search

Slide Credit: Simon Prince

Notation change #1

$$
\begin{aligned}
h_1 &= \mathrm{a}[\theta_{10} + \theta_{11}x] \\
h_2 &= \mathrm{a}[\theta_{20} + \theta_{21}x] \\
h_3 &= \mathrm{a}[\theta_{30} + \theta_{31}x]
\end{aligned}
\longrightarrow
\begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix} = \mathbf{a}\left[\begin{bmatrix} \theta_{10} \\ \theta_{20} \\ \theta_{30} \end{bmatrix} + \begin{bmatrix} \theta_{11} \\ \theta_{21} \\ \theta_{31} \end{bmatrix} x\right]
$$

$$
\begin{aligned}
h'_1 &= \mathrm{a}[\psi_{10} + \psi_{11}h_1 + \psi_{12}h_2 + \psi_{13}h_3] \\
h'_2 &= \mathrm{a}[\psi_{20} + \psi_{21}h_1 + \psi_{22}h_2 + \psi_{23}h_3] \\
h'_3 &= \mathrm{a}[\psi_{30} + \psi_{31}h_1 + \psi_{32}h_2 + \psi_{33}h_3]
\end{aligned}
\longrightarrow
\begin{bmatrix} h'_1 \\ h'_2 \\ h'_3 \end{bmatrix} = \mathbf{a}\left[\begin{bmatrix} \psi_{10} \\ \psi_{20} \\ \psi_{30} \end{bmatrix} + \begin{bmatrix} \psi_{11} & \psi_{12} & \psi_{13} \\ \psi_{21} & \psi_{22} & \psi_{23} \\ \psi_{31} & \psi_{32} & \psi_{33} \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix}\right]
$$

$$
y' = \phi'_0 + \phi'_1 h'_1 + \phi'_2 h'_2 + \phi'_3 h'_3 \longrightarrow y' = \phi'_0 + \begin{bmatrix} \phi'_1 & \phi'_2 & \phi'_3 \end{bmatrix} \begin{bmatrix} h'_1 \\ h'_2 \\ h'_3 \end{bmatrix}
$$

Slide Credit: Simon Prince

Notation change #2

$$\begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix} = \mathbf{a}\left[\begin{bmatrix} \theta_{10} \\ \theta_{20} \\ \theta_{30} \end{bmatrix} + \begin{bmatrix} \theta_{11} \\ \theta_{21} \\ \theta_{31} \end{bmatrix} x\right] \longrightarrow \mathbf{h} = \mathbf{a}\left[\boldsymbol{\theta}_0 + \boldsymbol{\theta} x\right]$$

$$\begin{bmatrix} h'_1 \\ h'_2 \\ h'_3 \end{bmatrix} = \mathbf{a}\left[\begin{bmatrix} \psi_{10} \\ \psi_{20} \\ \psi_{30} \end{bmatrix} + \begin{bmatrix} \psi_{11} & \psi_{12} & \psi_{13} \\ \psi_{21} & \psi_{22} & \psi_{23} \\ \psi_{32} & \psi_{32} & \psi_{33} \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix}\right] \longrightarrow \mathbf{h}' = \mathbf{a}\left[\boldsymbol{\psi}_0 + \boldsymbol{\Psi}\mathbf{h}\right]$$

$$y' = \phi'_0 + \begin{bmatrix} \phi'_1 & \phi'_2 & \phi'_3 \end{bmatrix} \begin{bmatrix} h'_1 \\ h'_2 \\ h'_3 \end{bmatrix} \longrightarrow y = \boldsymbol{\phi}'_0 + \boldsymbol{\phi}'\mathbf{h}'$$

Slide Credit: Simon Prince

Notation change #3

Bias vector

Weight matrix

$$\mathbf{h} = \mathbf{a}\left[\boldsymbol{\theta}_0 + \boldsymbol{\theta} x\right] \longrightarrow \mathbf{h}_1 = \mathbf{a}[\boldsymbol{\beta}_0 + \boldsymbol{\Omega}_0 \mathbf{x}]$$

$$\mathbf{h}' = \mathbf{a}\left[\boldsymbol{\psi}_0 + \boldsymbol{\Psi}\mathbf{h}\right] \longrightarrow \mathbf{h}_2 = \mathbf{a}[\boldsymbol{\beta}_1 + \boldsymbol{\Omega}_1 \mathbf{h}_1]$$

$$y = \phi_0' + \boldsymbol{\phi}'\mathbf{h}' \longrightarrow \mathbf{y} = \boldsymbol{\beta}_2 + \boldsymbol{\Omega}_2 \mathbf{h}_2$$

Slide Credit: Simon Prince

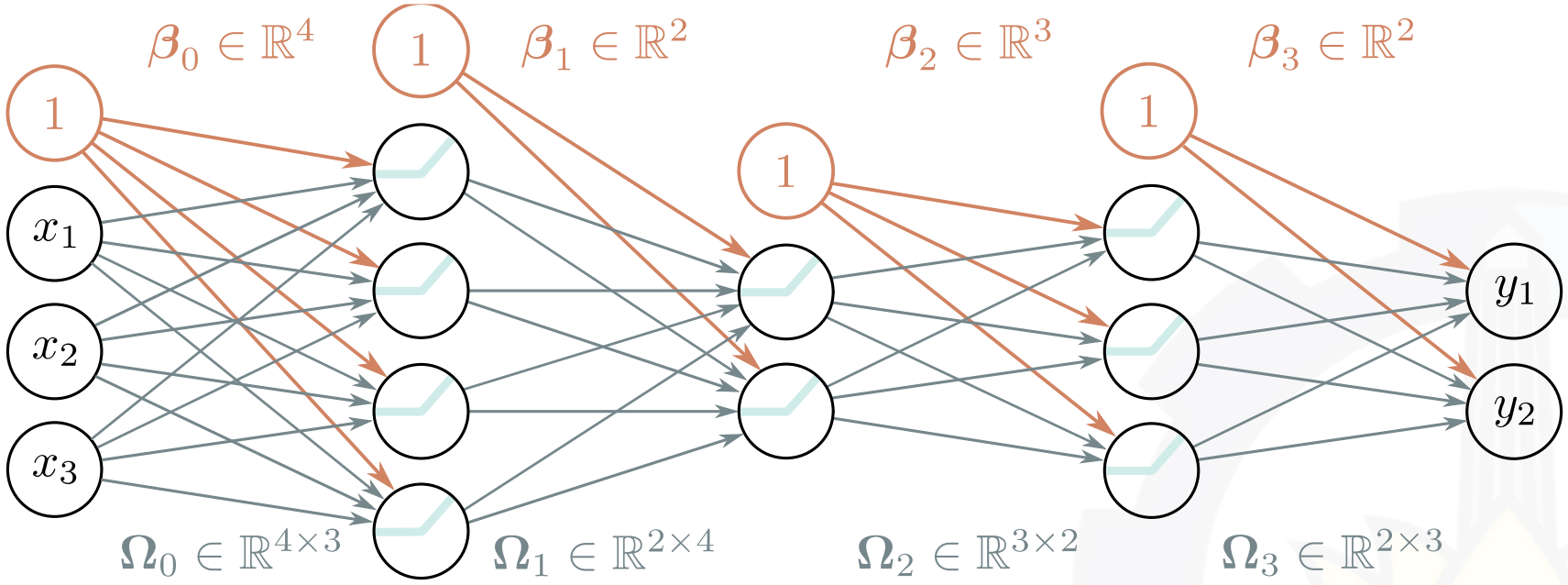
General equations for deep network

$$
\begin{aligned}
\mathbf{h}_1 &= \mathbf{a}[\boldsymbol{\beta}_0 + \boldsymbol{\Omega}_0\mathbf{x}] \\
\mathbf{h}_2 &= \mathbf{a}[\boldsymbol{\beta}_1 + \boldsymbol{\Omega}_1\mathbf{h}_1] \\
\mathbf{h}_3 &= \mathbf{a}[\boldsymbol{\beta}_2 + \boldsymbol{\Omega}_2\mathbf{h}_2] \\
&\vdots \\
\mathbf{h}_K &= \mathbf{a}[\boldsymbol{\beta}_{K-1} + \boldsymbol{\Omega}_{K-1}\mathbf{h}_{K-1}] \\
\mathbf{y} &= \boldsymbol{\beta}_K + \boldsymbol{\Omega}_K\mathbf{h}_K,
\end{aligned}
$$

$$
\mathbf{y} = \boldsymbol{\beta}_K + \boldsymbol{\Omega}_K\mathbf{a}\left[\boldsymbol{\beta}_{K-1} + \boldsymbol{\Omega}_{K-1}\mathbf{a}\left[\ldots\boldsymbol{\beta}_2 + \boldsymbol{\Omega}_2\mathbf{a}\left[\boldsymbol{\beta}_1 + \boldsymbol{\Omega}_1\mathbf{a}\left[\boldsymbol{\beta}_0 + \boldsymbol{\Omega}_0\mathbf{x}\right]\right]\ldots\right]\right]
$$

Slide Credit: Simon Prince

Example



Input, $\mathbf{x}$

$D_i = 3$

Hidden layer, $\mathbf{h}_1$

$D_1 = 4$

Hidden layer, $\mathbf{h}_2$

$D_2 = 2$

Hidden layer, $\mathbf{h}_3$

$D_3 = 3$

Output, $\mathbf{y}$

$D_o = 2$

Slide Credit: Simon Prince

Shallow vs. deep networks

The best results are created by deep networks with many layers.

- 50-1000 layers for most applications
- Best results in
  - Computer vision
  - Natural language processing
  - Graph neural networks
  - Generative models
  - Reinforcement learning

All use deep networks.
But why?

Slide Credit: Simon Prince

Shallow vs. deep networks

1. Ability to approximate different functions?

Both obey the universal approximation theorem.

Argument: One layer is enough, and for deep networks could arrange for the other layers to compute the identity function.

Slide Credit: Simon Prince

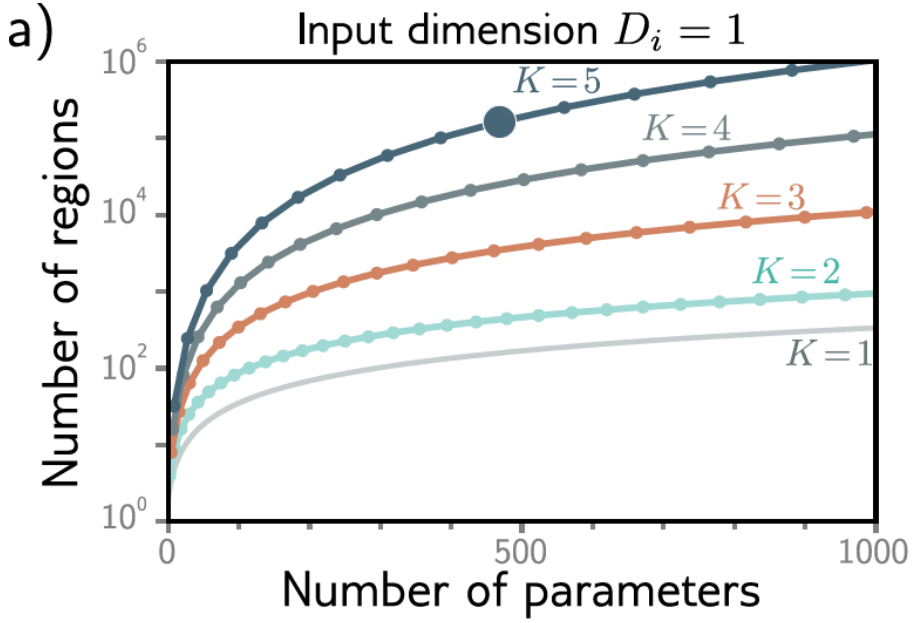
Shallow vs. deep networks

2. Number of linear regions per parameter

Slide Credit: Simon Prince

Number of linear regions per parameter

2. Number of linear regions per parameter



5 layers
10 hidden units per layer
471 parameters
161,501 linear regions

5 layers
50 hidden units per layer
10,801 parameters
$>10^{40}$ linear regions

Slide Credit: Simon Prince

Shallow vs. deep networks

2. Number of linear regions per parameter

- Deep networks create many more regions per parameters
- But there are dependencies between them
  - Think of folding example
  - Perhaps similar symmetries in real-world functions? Unknown

Slide Credit: Simon Prince

Shallow vs. deep networks

3. Depth efficiency

- There are some functions that require a shallow network with exponentially more hidden units than a deep network to achieve an equivalent approximation
- This is known as the depth efficiency of deep networks
- But do the real-world functions we want to approximate have this property? Unknown.

Slide Credit: Simon Prince

Shallow vs. deep networks

4. Large structured networks

- Think about images as input – might be 1M pixels
- Fully connected works not practical
- Answer is to have weights that only operate locally, and share across image
- This leads to convolutional networks
- Gradually integrate information from across the image – needs multiple layers

Slide Credit: Simon Prince

Shallow vs. deep networks

5. Fitting and generalization

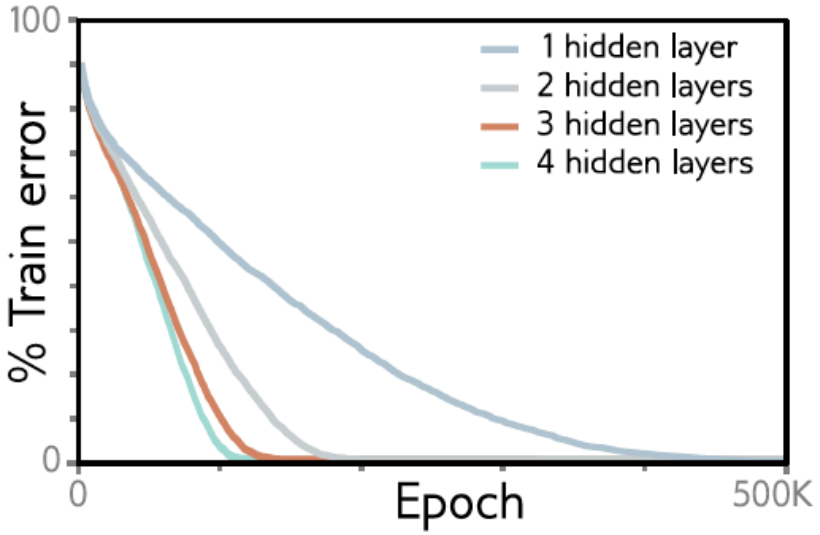
- Fitting of deep models seems to be easier up to about 20 layers
- Then needs various tricks to train deeper networks, so (in vanilla form), fitting becomes harder
- Generalization is good in deep networks. Why?

Slide Credit: Simon Prince

Shallow vs. deep networks

5. Fitting and generalization

Figure 20.2 MNIST-1D training. Four fully connected networks were fit to 4000 MNIST-1D examples with random labels using full batch gradient descent, He initialization, no momentum or regularization, and learning rate 0.0025. Models with 1,2,3,4 layers had 298, 100, 75, and 63 hidden units per layer and 15208, 15210, 15235, and 15139 parameters, respectively. All models train successfully, but deeper models require fewer epochs.



Slide Credit: Simon Prince

Terima kasih