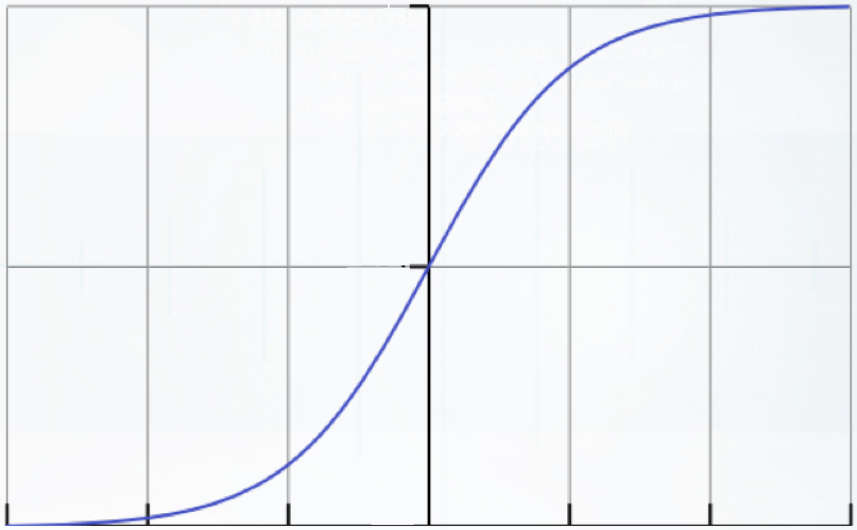
Multilayer perceptron

In this video

- The basic principle of “deep learning”
- The one you will be applying to all problems hereinafter
- Absolutely essential for all future material

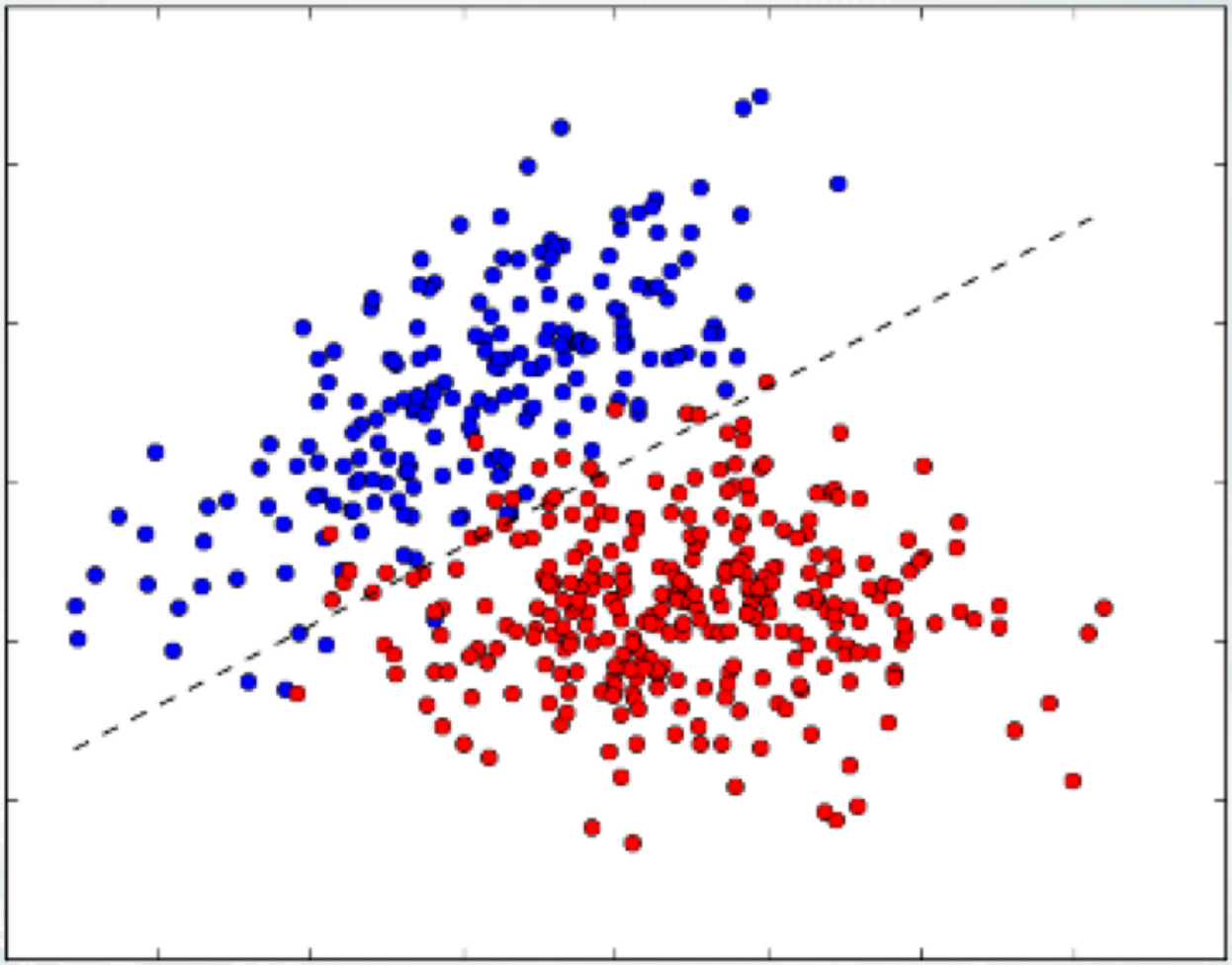
Recap: linear model

$W \cdot x + b$ →  → $P(y = +1|x)$

x - features vector

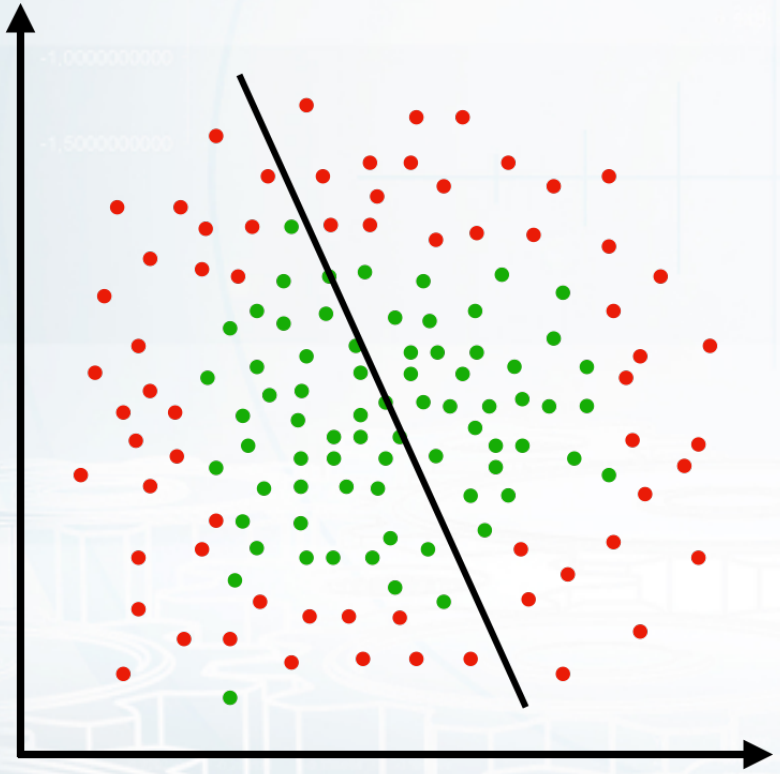
W, b - model slope and intercept

Linear dependency

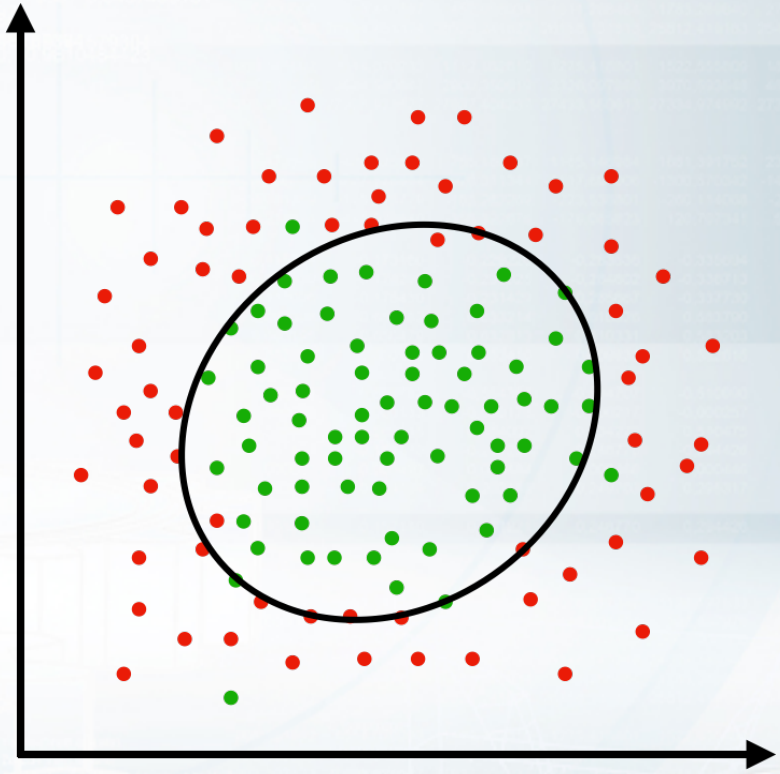


Nonlinear dependencies

What we have



What we want



Somewhat nonlinear dependencies

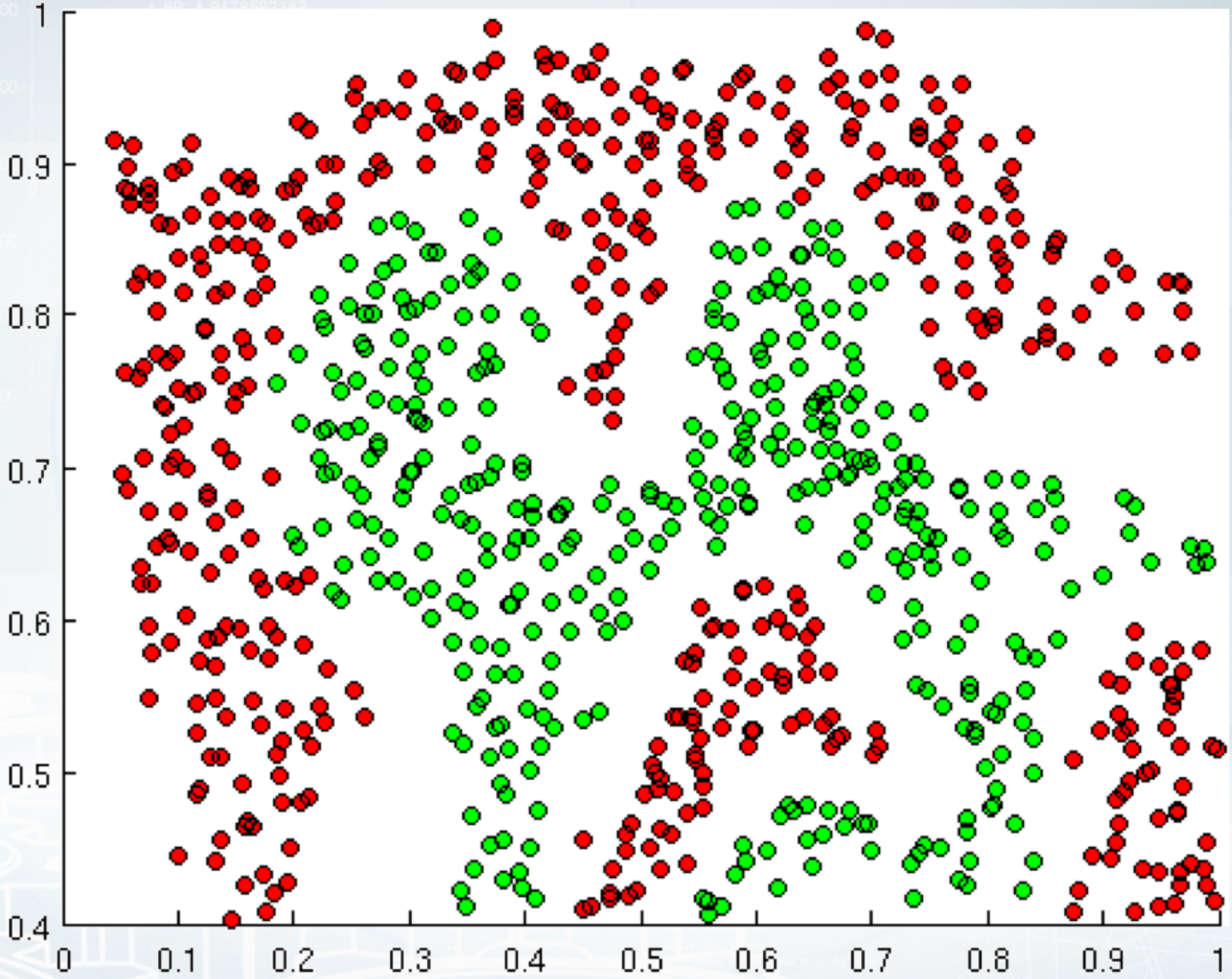
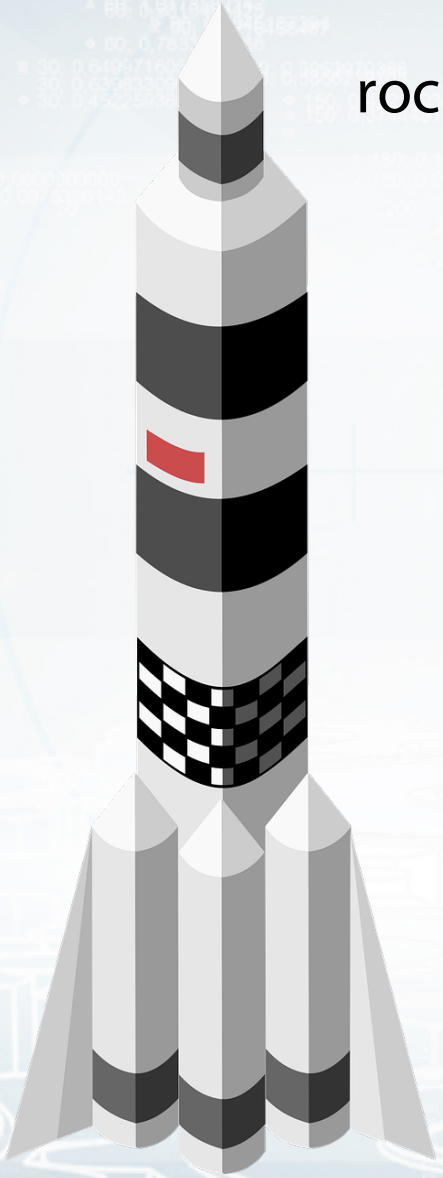


Image: Andrew Ng

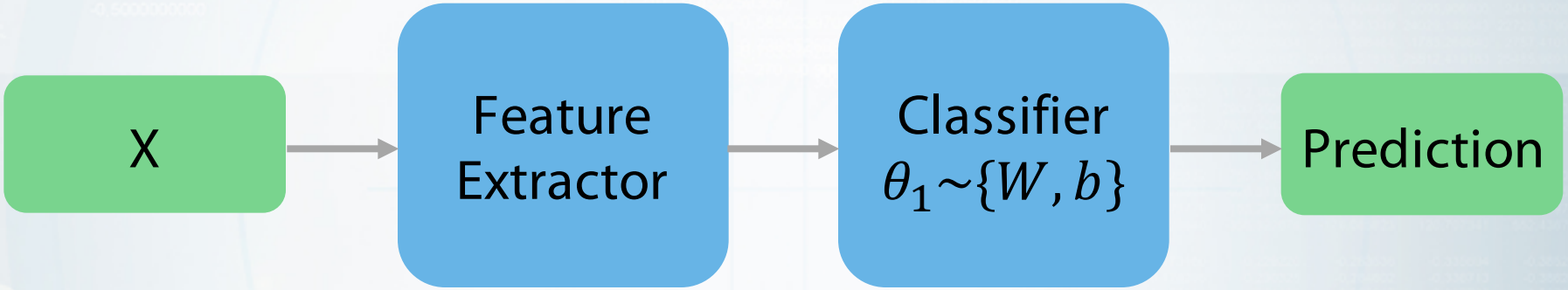
Extremely nonlinear dependencies

rocket



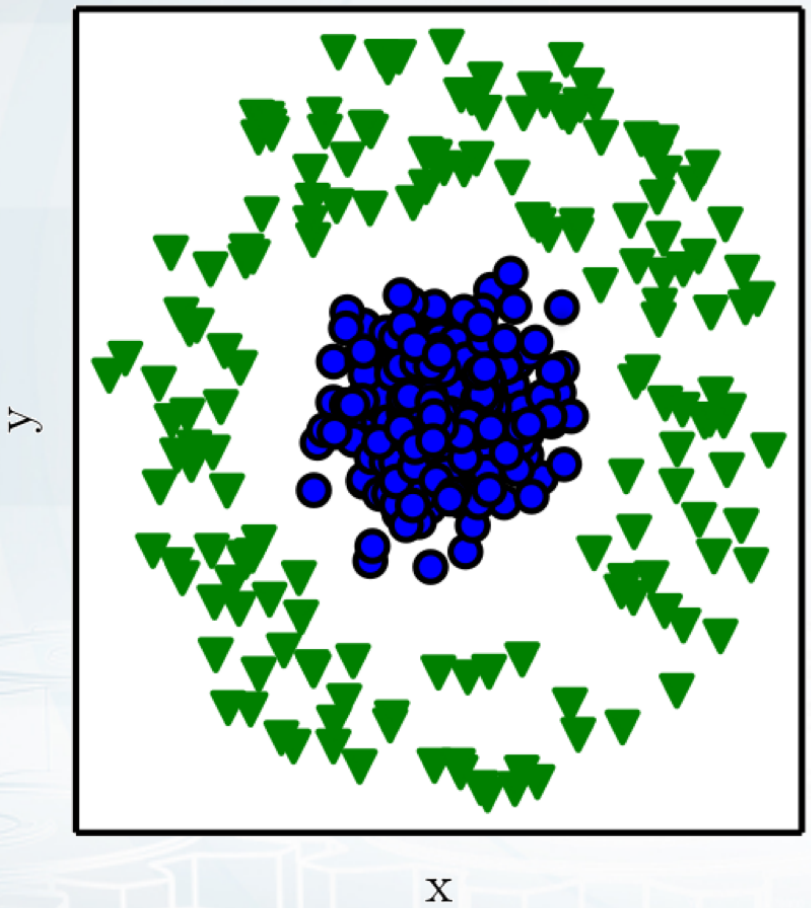
cat

Extremely nonlinear dependencies



Feature extraction

Cartesian coordinates



Polar coordinates

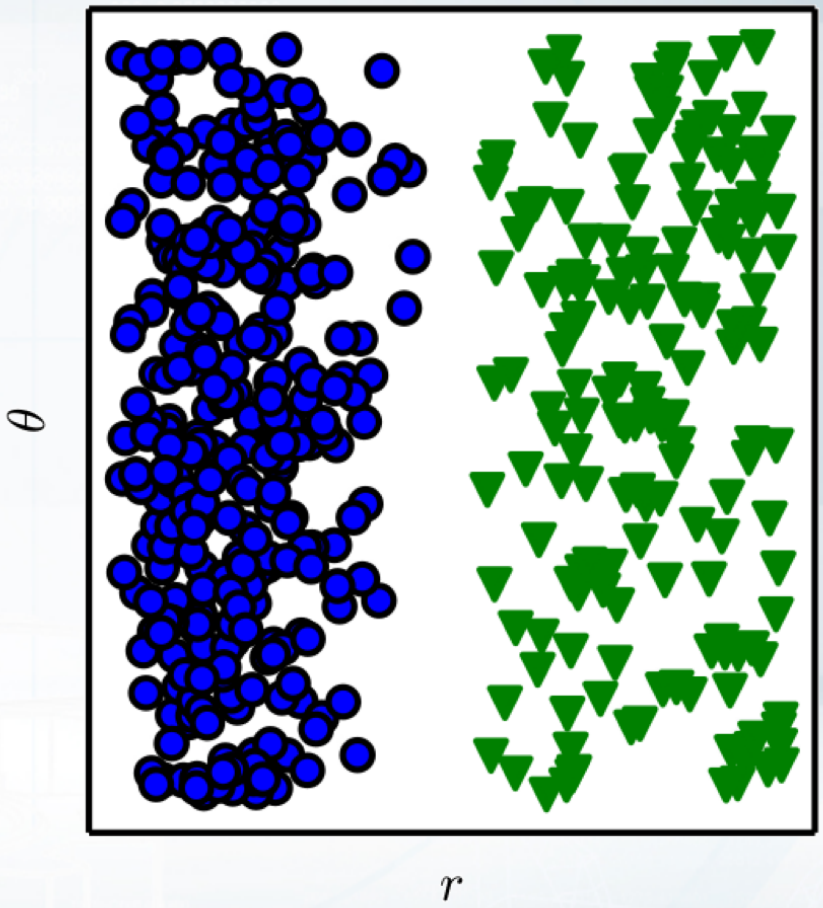
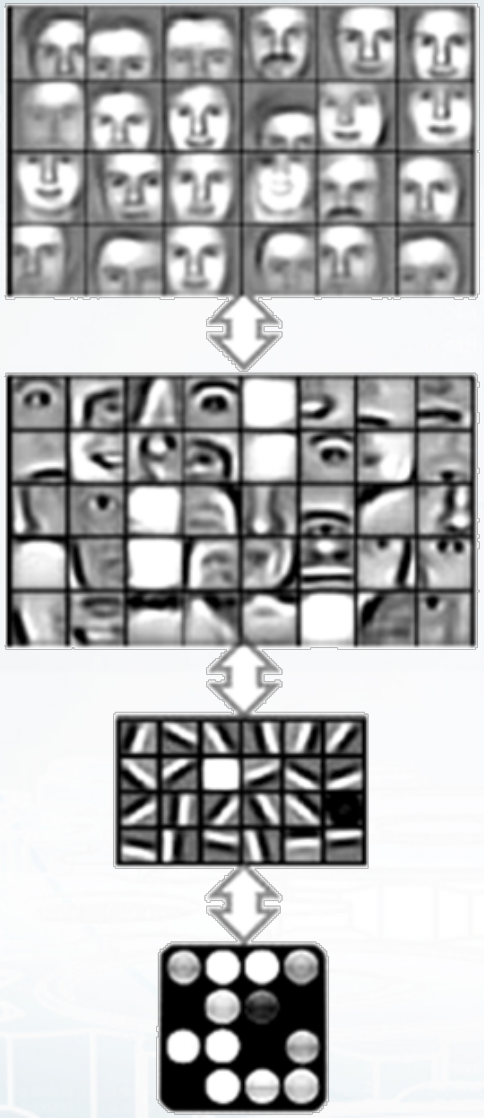


Image: Ian Goodfellow et al.



Discrete Choices

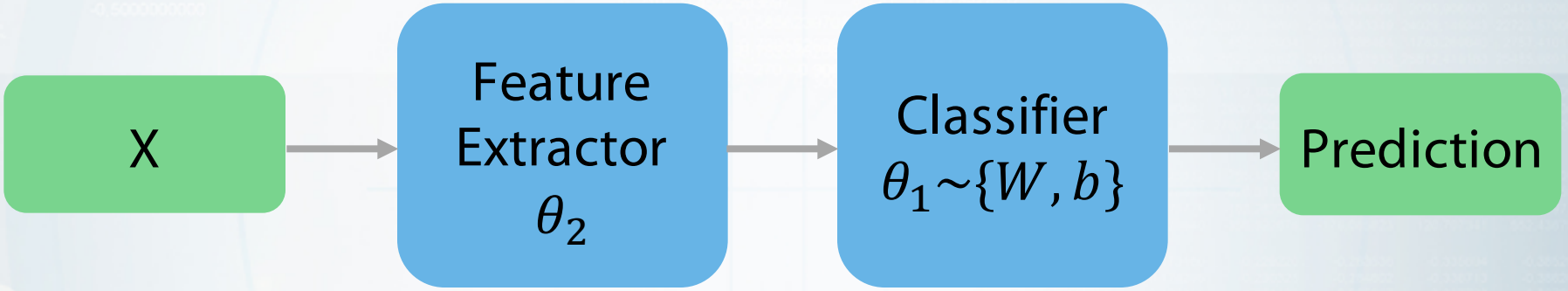
Layer 2 Features

Layer 1 Features

Original Data

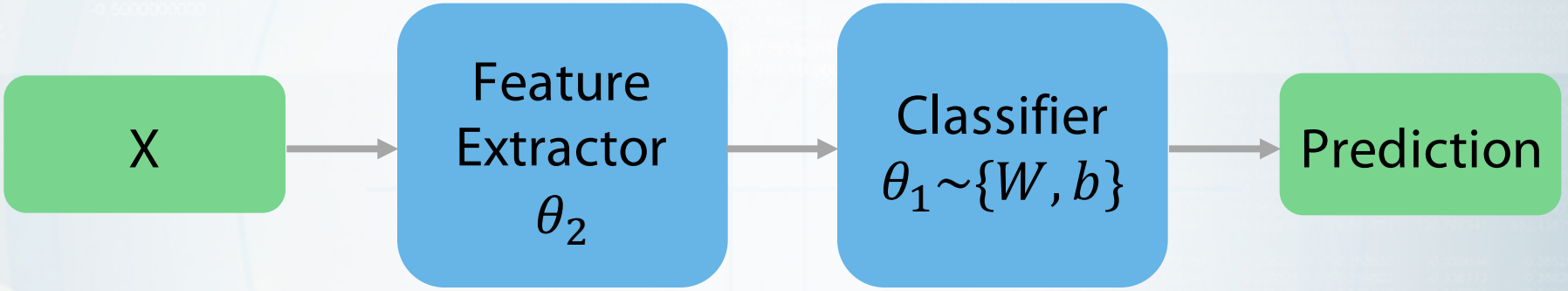
Image: Alex Burnap et al.

Feature extraction



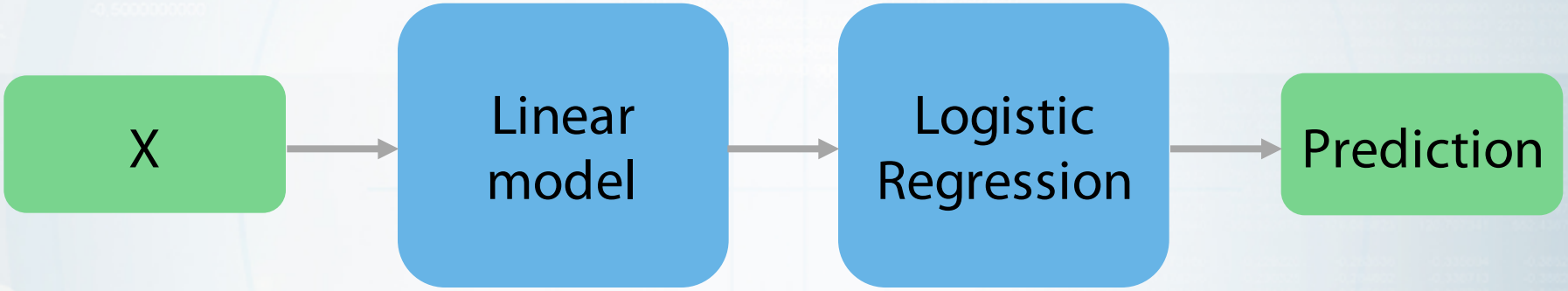
Training: manual $\underset{\theta_1}{\operatorname{argmin}} L\big(y, P(y \mid x)\big)$

Can it be done automatically?



Training: ? $\underset{\theta_1}{\operatorname{argmin}} L\big(y, P(y \mid x)\big)$

Try linear



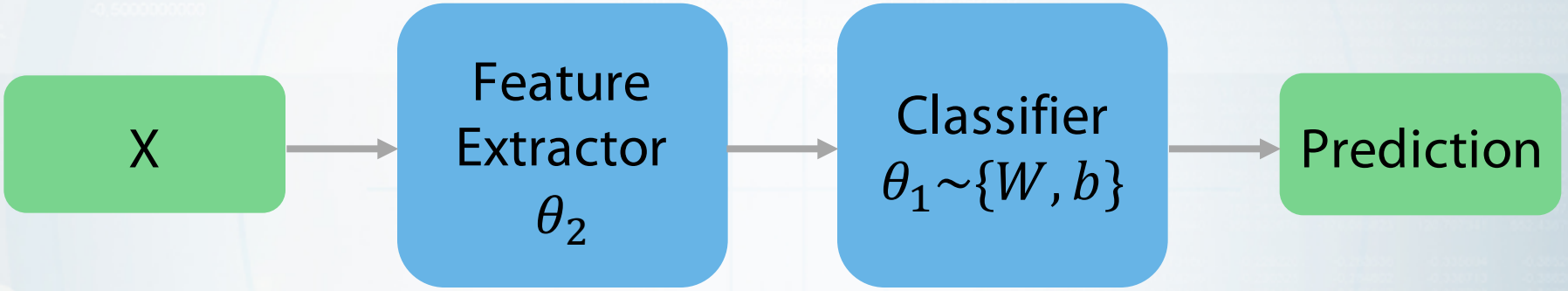
$$h_j = \sum_i w_{ij}^h x_i + b_j^h \qquad y_{\text{pred}} = \sigma\left(\sum_j w_j^o h_j + b^o\right)$$

$$j \in \{1,2,\ldots,n\}$$

Training:

$$\underset{w^h, w^0, b^h, b^0}{\text{argmin}} \; L\big(y, P(y \mid x)\big)$$

Can it be done automatically?



Training: ? $\underset{\theta_1}{\operatorname{argmin}} L\big(y, P(y \mid x)\big)$

In-video Question (технический слайд)

Will stacking linear functions improve quality?

Answer: No

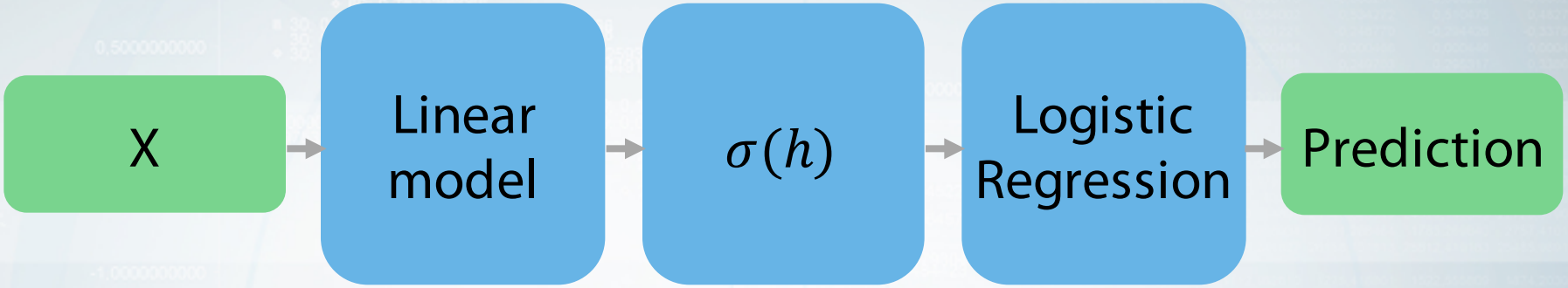
A combination of linear models is a linear model

$$P(y \mid x) = \sigma\left(\sum_j w_j^o \left(\sum_i w_{ij}^h x_i + b_j^h\right) + b^o\right)$$

$$w'_i = \sum_j w_j^o w_{ij}^h \qquad b' = \sum_j w_j^o b_j^h + b^o$$

$$P(y \mid x) = \sigma\left(\sum_i w'_i x_i + b'\right)$$

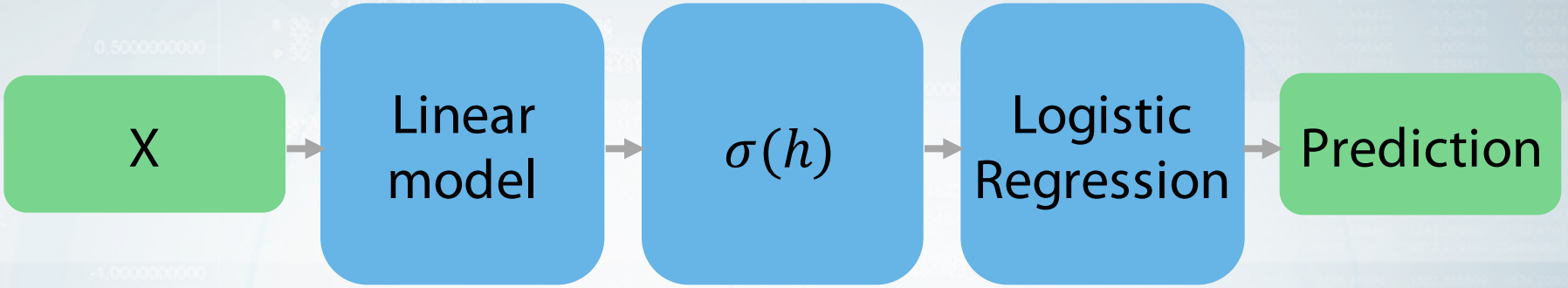
Nonlinearity



$$h_j = \sigma\left(\sum_i w_{ij}^h x_i + b_j^h\right) \qquad y_{\text{pred}} = \sigma\left(\sum_j w_j^o h_j + b^o\right)$$

$$j \in \{1,2,\ldots,n\}$$

Nonlinearity

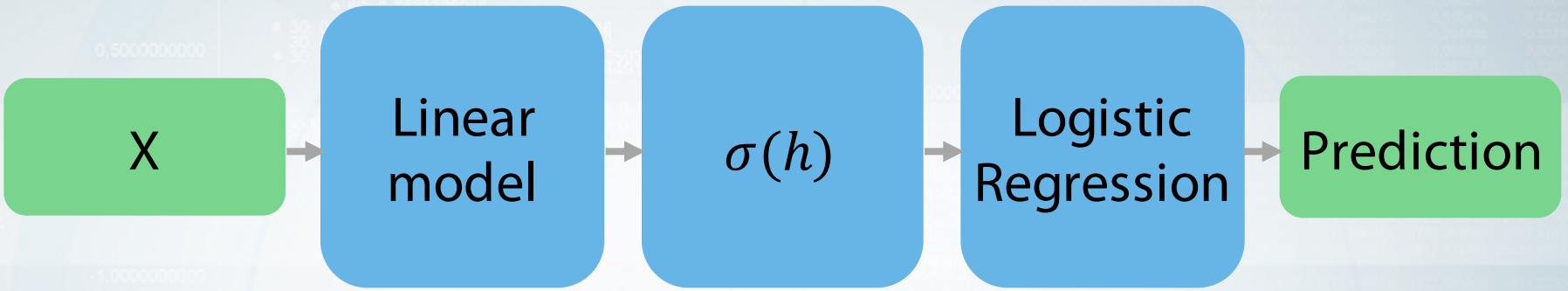


$$h_j = \sigma\left(\sum_i w_{ij}^h x_i + b_j^h\right) \qquad y_{\text{pred}} = \sigma\left(\sum_j w_j^o h_j + b^o\right)$$

$$j \in \{1,2,\ldots,n\}$$

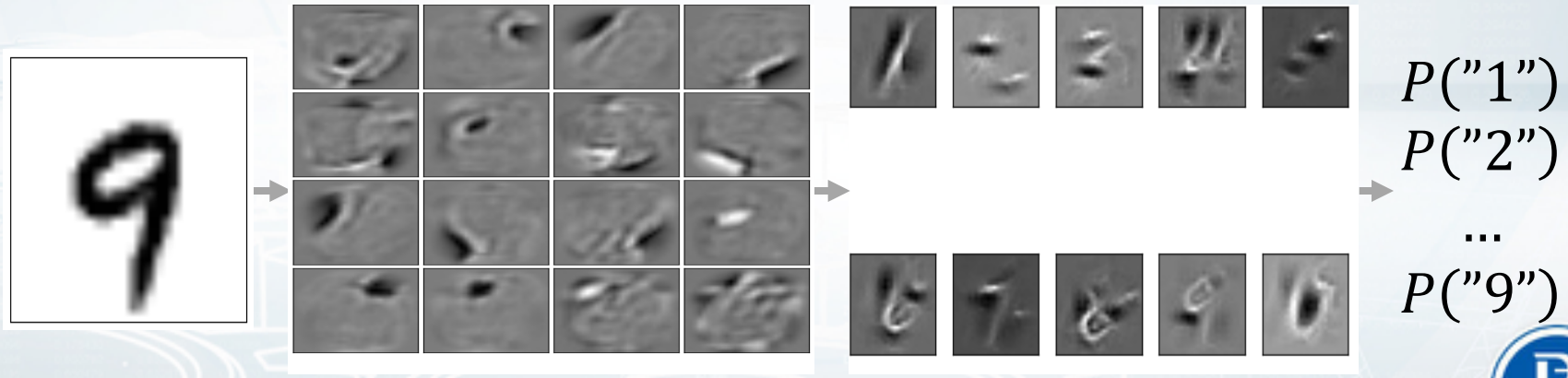
Output: $$P(y \mid x) = \sigma\left(\sum_j w_j^o \sigma\left(\sum_i w_{ij}^h x_i + b_j^h\right) + b^o\right)$$

Effect of nonlinearity



$$h_j = \sigma\left(\sum_i w_{ij}^h x_i + b_j^h\right) \qquad y_{\text{pred}} = \sigma\left(\sum_j w_j^o h_j + b^o\right)$$

$$j \in \{1,2,\ldots,n\}$$



Balakrishnan Prabhu, https://corpocrat.com/2014/10/17/machine-learning-using-restricted-boltzmann-machines/

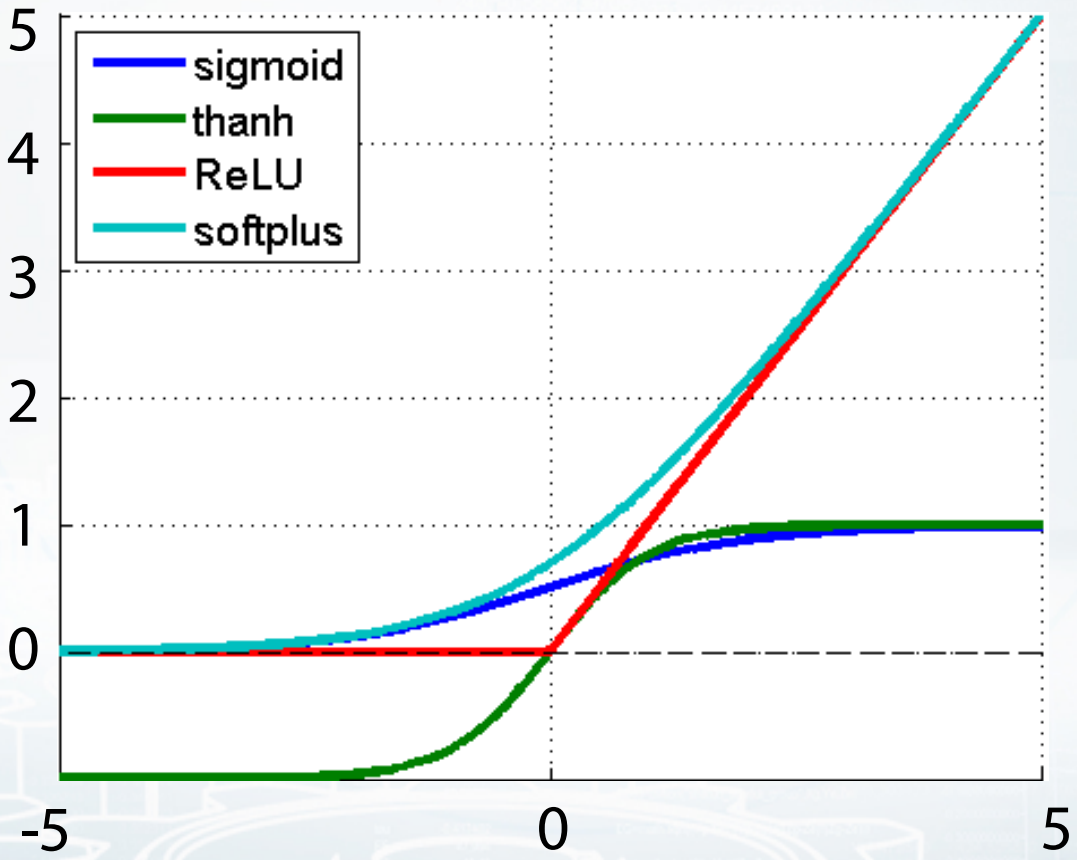
Types of nonlinearity

$$f(a) = \frac{1}{1 + e^{a}}$$

$$f(a) = \max(0, a)$$

$$f(a) = \tanh(a)$$

$$f(a) = \log(1 + e^{a})$$



Wei Di, https://imiloainf.wordpress.com/2013/11/06/rectifier-nonlinearities/

Recap and terminology

- Layer is a building block for neural network:
  - Input layer
  - Dense layer: $f(x) = Wx + b$
  - Nonlinearity layer: $f(x) = \sigma(x)$
  - A few more: we will cover later
  - Output layer

  } Hidden layers

- Activation is layer output
  - i. e. some intermediate signal in the neural network

Potential caveats?

- Hardcore overfitting
- No “golden standard” for architecture
- Computationally heavy

See the next episodes for solutions!