Machine Learning and Data Mining

Lecture 7.2: Advanced Neural Networks & Decision Trees

CS 434

Backpropagation Again!

Element-wise Logistic with 3 Inputs:

Forward:

$$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \sigma\left(\begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix}\right) = \begin{bmatrix} 1/(1+e^{-z_1}) \\ 1/(1+e^{-z_2}) \\ 1/(1+e^{-z_3}) \end{bmatrix}$$

Backward:

$$\frac{\partial \boldsymbol{a}}{\partial \boldsymbol{z}} = \begin{bmatrix} \frac{\delta a_1}{\delta z_1} & \frac{\delta a_1}{\delta z_2} & \frac{\delta a_1}{\delta z_3} \\ \frac{\delta a_2}{\delta z_1} & \frac{\delta a_2}{\delta z_2} & \frac{\delta a_2}{\delta z_3} \\ \frac{\delta a_3}{\delta z_1} & \frac{\delta a_3}{\delta z_2} & \frac{\delta a_3}{\delta z_3} \end{bmatrix}$$

$$\frac{\delta a_i}{\delta z_i} = a_i(1-a_i)$$

$$\frac{\delta a_i}{\delta z_j} = 0$$

$$\frac{\partial \boldsymbol{a}}{\partial \boldsymbol{z}} = \begin{bmatrix} a_1(1-a_1) & 0 & 0 \\ 0 & a_2(1-a_2) & 0 \\ 0 & 0 & a_3(1-a_3) \end{bmatrix}$$

Backpropagation Again!

Linear layer with 5 inputs and 3 outputs:

Forward:

$$\begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} = \sigma\left(\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}\right) = W\boldsymbol{x} + \boldsymbol{b} = \begin{bmatrix} w_{11} & w_{12} & w_{13} & w_{14} & w_{15} \\ w_{21} & w_{22} & w_{23} & w_{24} & w_{25} \\ w_{31} & w_{32} & w_{33} & w_{34} & w_{35} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

Backward: (Output with respect to input)

$$\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{x}} = \begin{bmatrix} \frac{\partial z_1}{\partial x_1} & \frac{\partial z_1}{\partial x_2} & \frac{\partial z_1}{\partial x_3} & \frac{\partial z_1}{\partial x_4} & \frac{\partial z_1}{\partial x_5} \\ \frac{\partial z_2}{\partial x_1} & \frac{\partial z_2}{\partial x_2} & \frac{\partial z_2}{\partial x_3} & \frac{\partial z_2}{\partial x_4} & \frac{\partial z_2}{\partial x_5} \\ \frac{\partial z_3}{\partial x_1} & \frac{\partial z_3}{\partial x_2} & \frac{\partial z_3}{\partial x_3} & \frac{\partial z_3}{\partial x_4} & \frac{\partial z_3}{\partial x_5} \end{bmatrix}$$

$$\frac{\delta z_i}{\delta x_j} = w_{ij}$$

$$\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{x}} = \begin{bmatrix} w_{11} & w_{12} & w_{13} & w_{14} & w_{15} \\ w_{21} & w_{22} & w_{23} & w_{24} & w_{25} \\ w_{31} & w_{32} & w_{33} & w_{34} & w_{35} \end{bmatrix} = W$$

Backpropagation Again!

Linear layer with 5 inputs and 3 outputs:

Forward:

$$\begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} = \sigma\left(\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}\right) = W\boldsymbol{x} + \boldsymbol{b} = \begin{bmatrix} w_{11} & w_{12} & w_{13} & w_{14} & w_{15} \\ w_{21} & w_{22} & w_{23} & w_{24} & w_{25} \\ w_{31} & w_{32} & w_{33} & w_{34} & w_{35} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

Backward: (Output with respect to bias)

$$\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{b}} = \begin{bmatrix} \frac{\partial z_1}{\partial b_1} & \frac{\partial z_1}{\partial b_2} & \frac{\partial z_1}{\partial b_3} \\ \frac{\partial z_2}{\partial b_1} & \frac{\partial z_2}{\partial b_2} & \frac{\partial z_2}{\partial b_3} \\ \frac{\partial z_3}{\partial b_1} & \frac{\partial z_3}{\partial b_2} & \frac{\partial z_3}{\partial b_3} \end{bmatrix}$$

$$\frac{\delta z_i}{\delta b_i} = 1$$

$$\frac{\delta z_i}{\delta b_{j \neq i}} = 0$$

$$\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{b}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = I$$

Backpropagation Again!

Linear layer with 5 inputs and 3 outputs:

Forward:

$$\begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} = \sigma\left(\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}\right) = W\boldsymbol{x} + \boldsymbol{b} = \begin{bmatrix} w_{11} & w_{12} & w_{13} & w_{14} & w_{15} \\ w_{21} & w_{22} & w_{23} & w_{24} & w_{25} \\ w_{31} & w_{32} & w_{33} & w_{34} & w_{35} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

Backward: (Output with respect to weights)

$$\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{W}} = \begin{bmatrix} \frac{\partial z_1}{\partial w_{11}} & \frac{\partial z_1}{\partial w_{12}} & \cdots & \frac{\partial z_1}{\partial w_{35}} \\ \frac{\partial z_2}{\partial w_{11}} & \frac{\partial z_2}{\partial w_{12}} & \cdots & \frac{\partial z_2}{\partial w_{35}} \\ \frac{\partial z_3}{\partial w_{11}} & \frac{\partial z_3}{\partial w_{12}} & \cdots & \frac{\partial z_3}{\partial w_{35}} \end{bmatrix}$$

$$\frac{\partial z_i}{\partial w_{ik}} = x_k$$

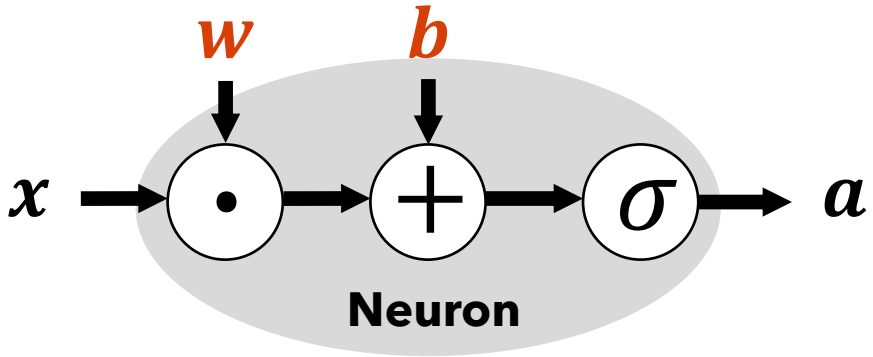
$$\frac{\partial z_i}{\partial w_{j \neq i,k}} = 0$$

$$\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{W}} = \begin{bmatrix} x^T & 0 \cdots 0 & 0 \cdots 0 \\ 0 \cdots 0 & x^T & 0 \cdots 0 \\ 0 \cdots 0 & 0 \cdots 0 & x^T \end{bmatrix}$$

RECAP

From Last Lecture

Neuron



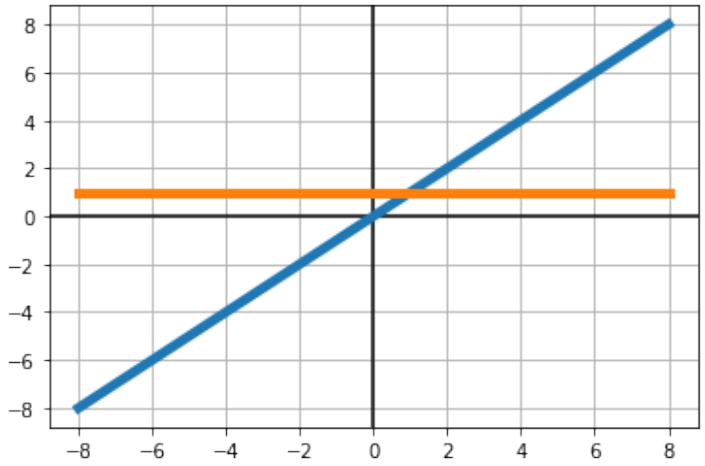
Neuron is a linear function of its input followed by a (typically) non-linear activation function to produce output. $a = \sigma\left(\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x} + b\right)$

Hyperparameters : activation function (σ)

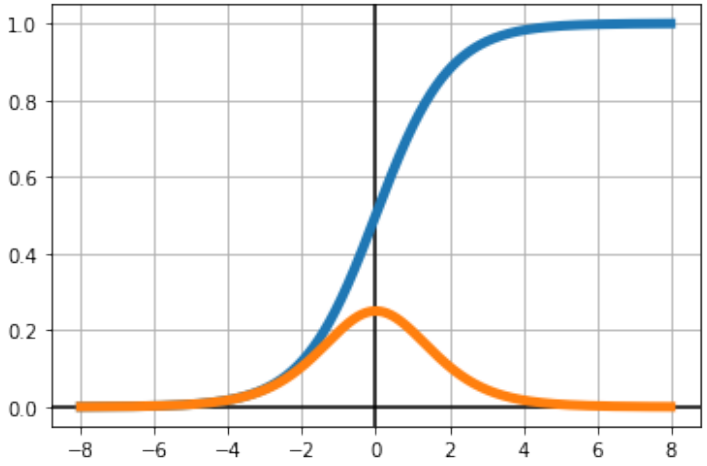
Learnable Parameters: $\mathbf{w} \in \mathbb{R}^d$ and $\mathbf{b} \in \mathbb{R}$

Activation Functions σ

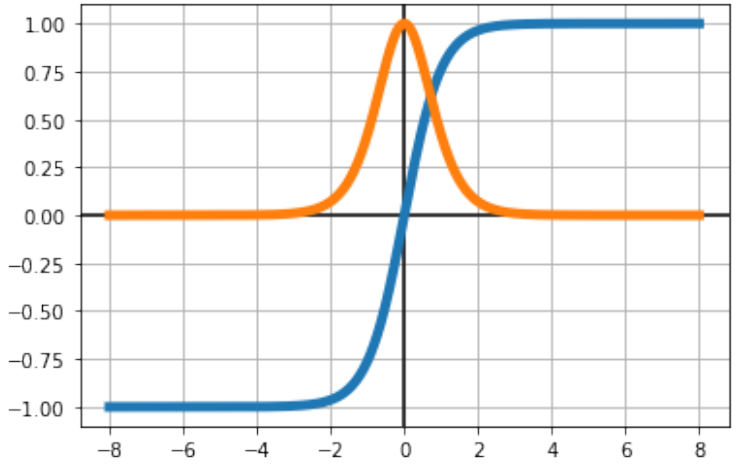
None $\sigma(z) = z$



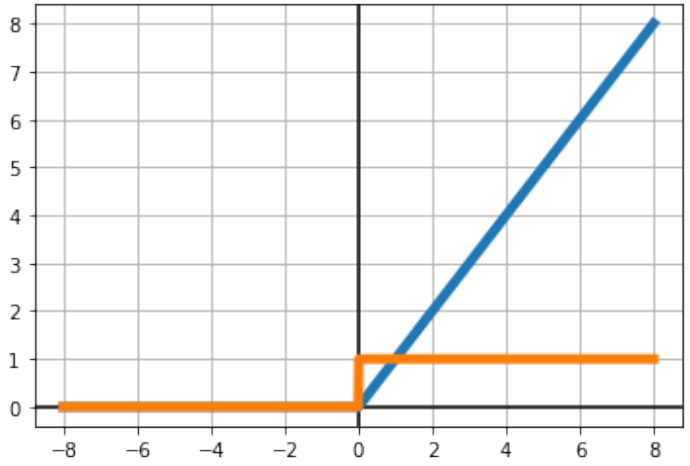
Sigmoid $\sigma(z) = \frac{1}{1+e^{-z}}$



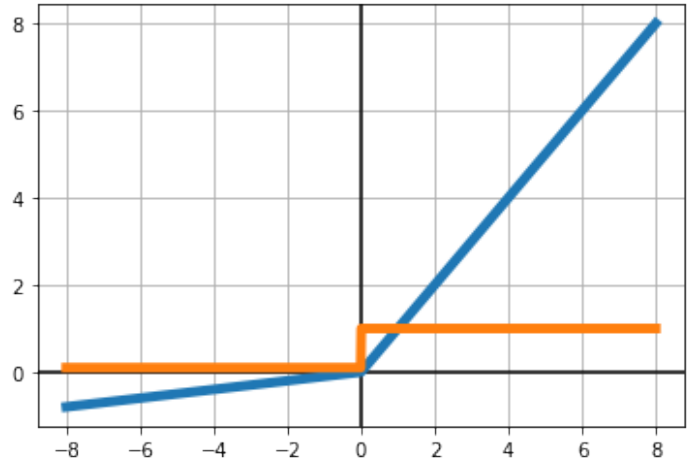
tanh $\sigma(z) = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}$



ReLU $\sigma(z) = \max(0, z)$

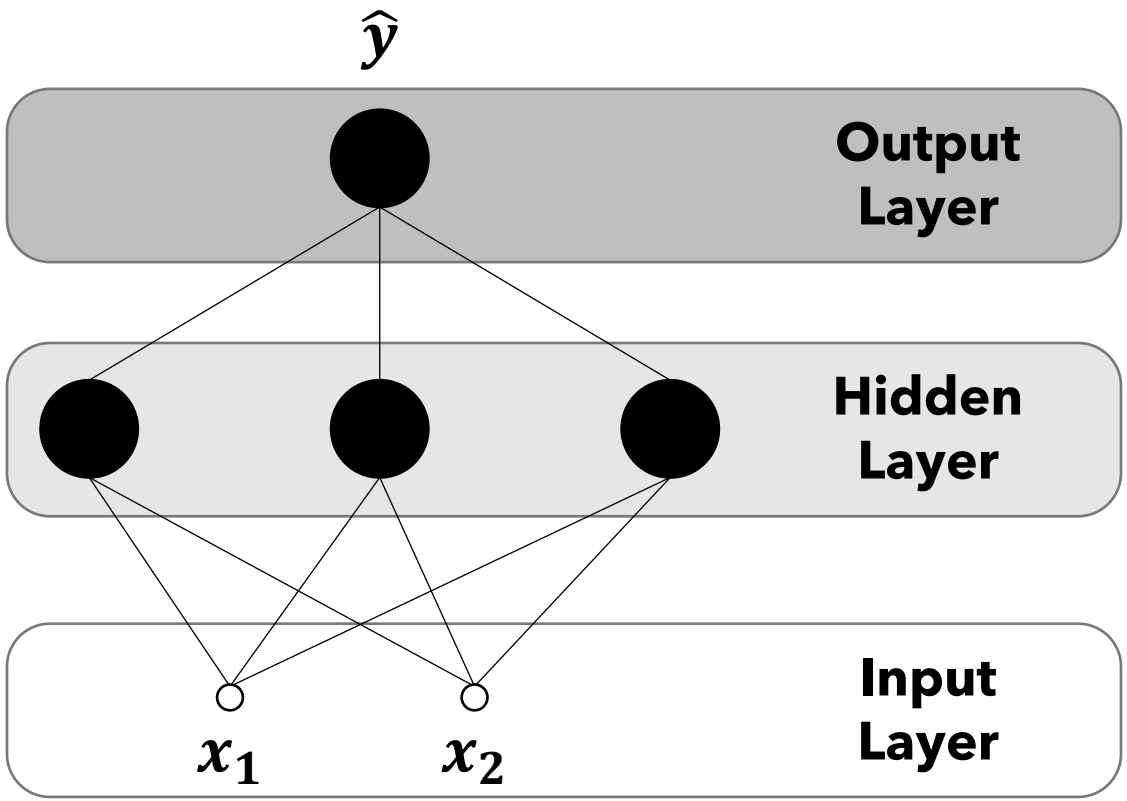


Leaky ReLU $\sigma(z) = \begin{cases} \alpha z & \text{if } z < 0 \\ z & \text{if } z \geq 0 \end{cases}$



Basic Multilayer Neural Network

A Neural Network is a set of connected neurons. A very typical arrangement is a feed-forward multilayer neural network like the one shown below.



Each layer receives its input from the previous layer and forwards its output to the next – thus the **feed-forward** description.

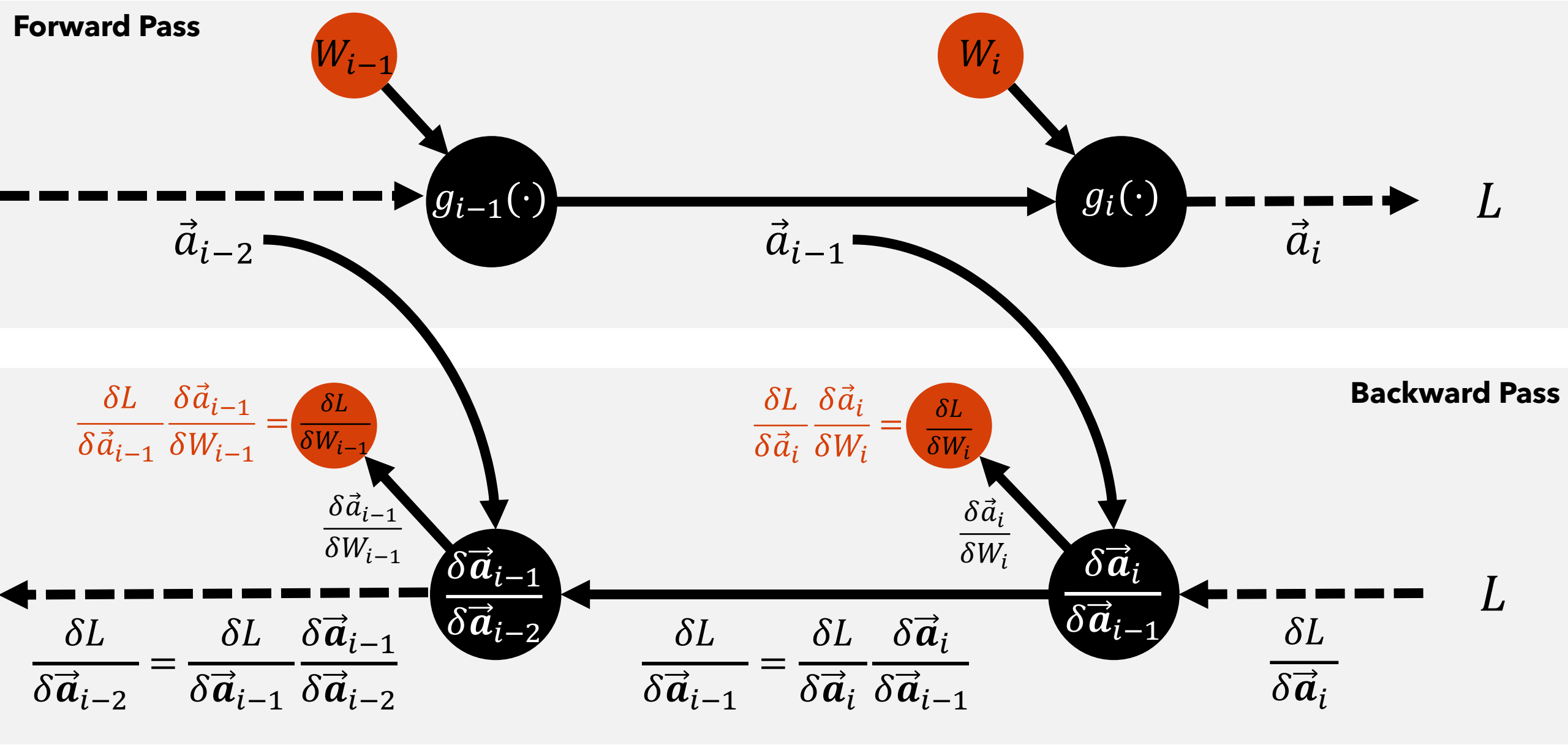
The layers of neurons between the input and output are referred to as **hidden layers**.

- This network for instance is **a 2-layer neural** network (1 hidden and 1 output)
- Number of neurons in a layer is referred to as its width.

Activation functions in different layers can be heterogenous.

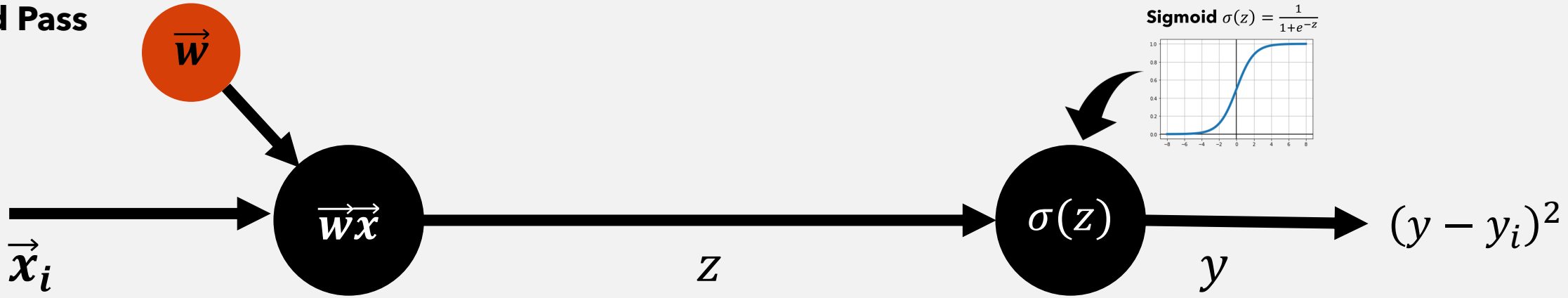
- Output layer's activation is task dependent
  - Linear for regression
  - Sigmoid or softmax for classification

Backpropagation (aka Reverse Mode Auto-differentiation)



Backpropagation (aka Reverse Mode Auto-differentiation)

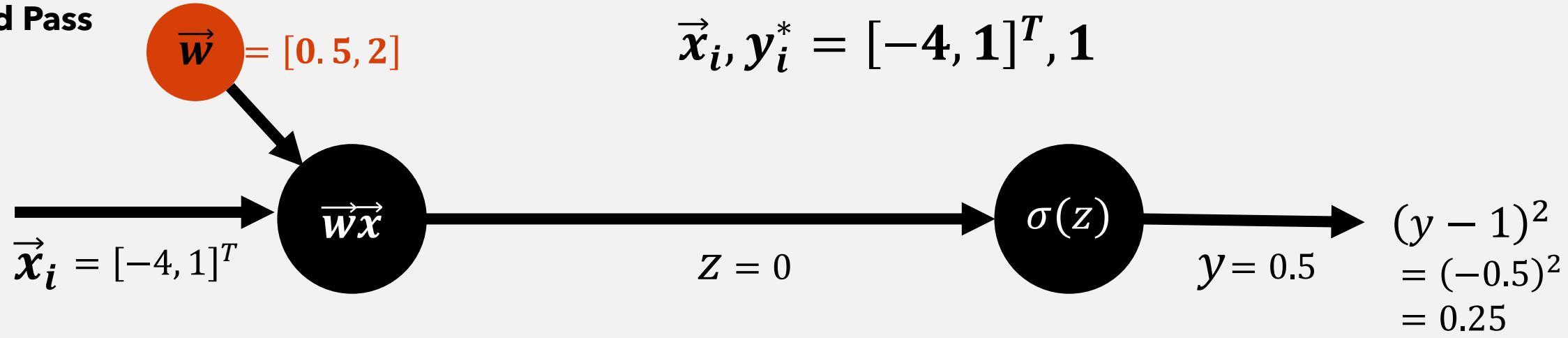
Forward Pass



Backward Pass

Backpropagation (aka Reverse Mode Auto-differentiation)

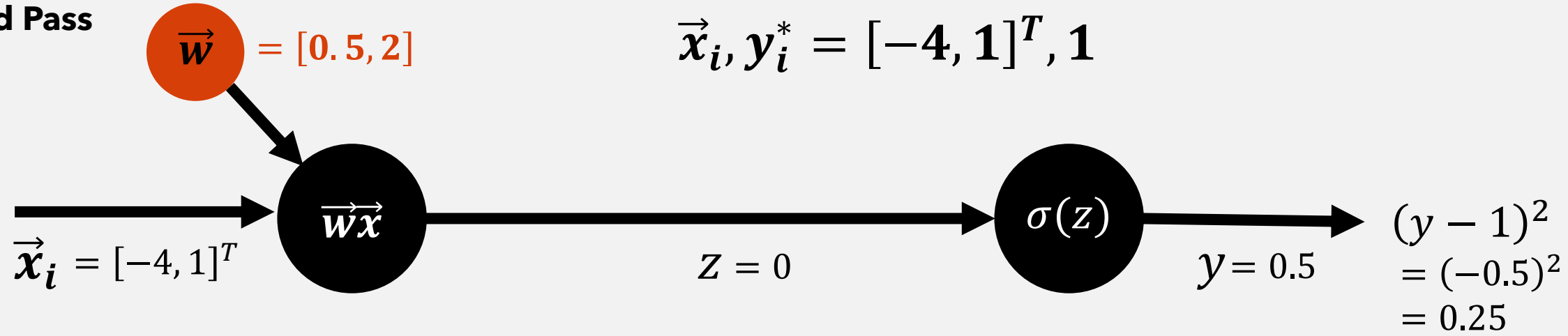
Forward Pass



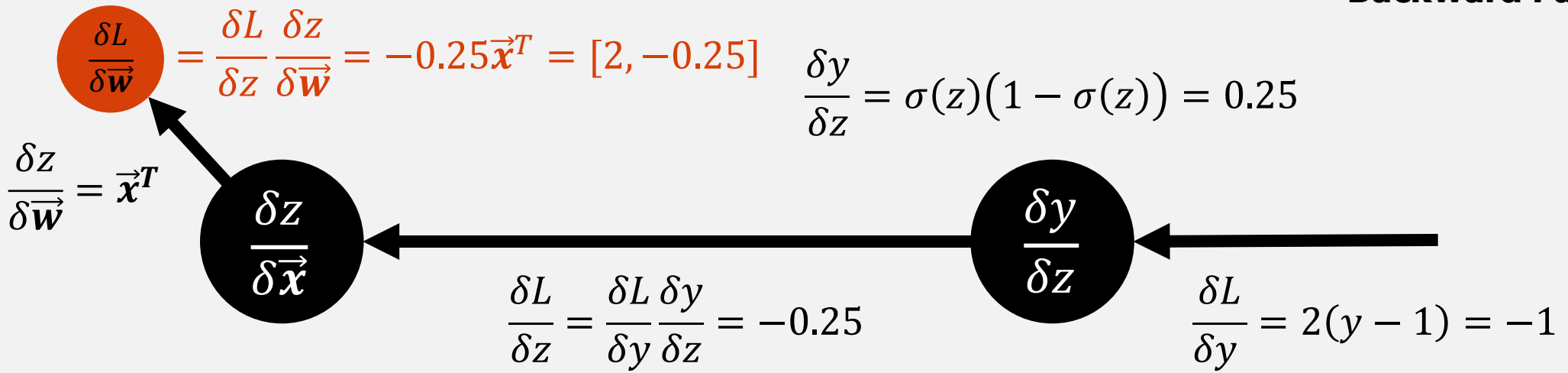
Backward Pass

Backpropagation (aka Reverse Mode Auto-differentiation)

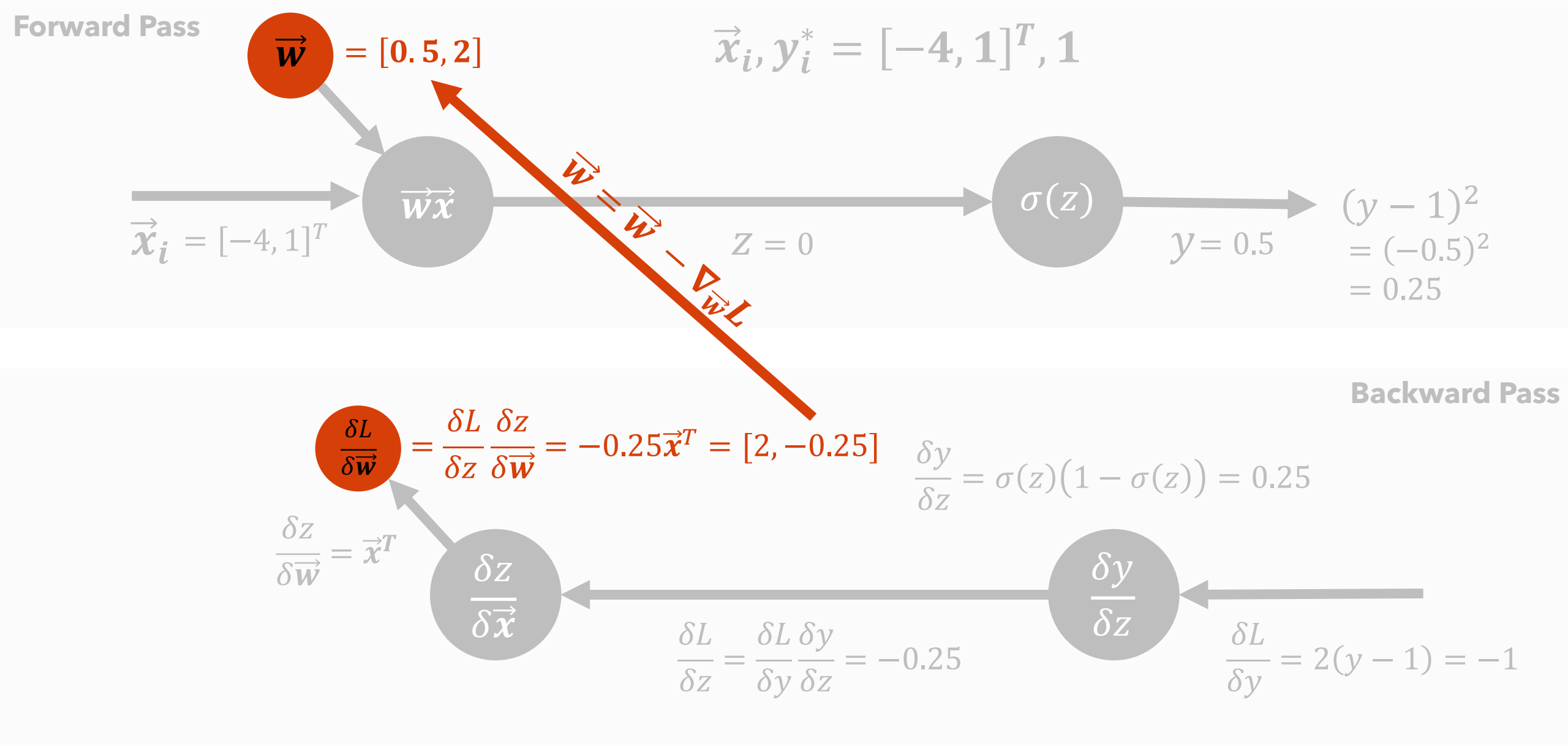
Forward Pass



Backward Pass

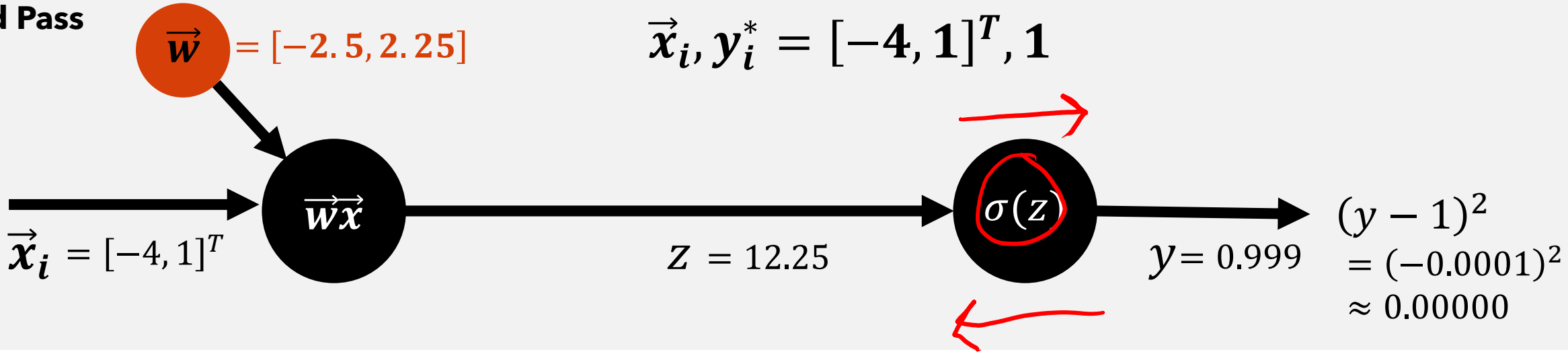


Backpropagation (aka Reverse Mode Auto-differentiation)



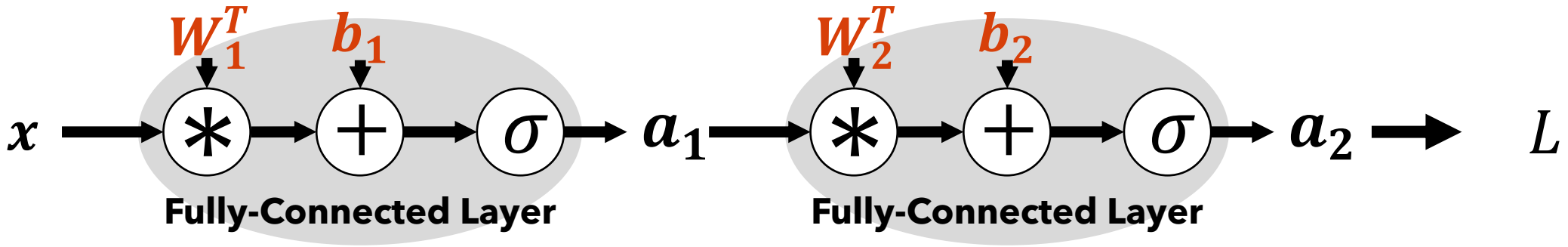
Backpropagation (aka Reverse Mode Auto-differentiation)

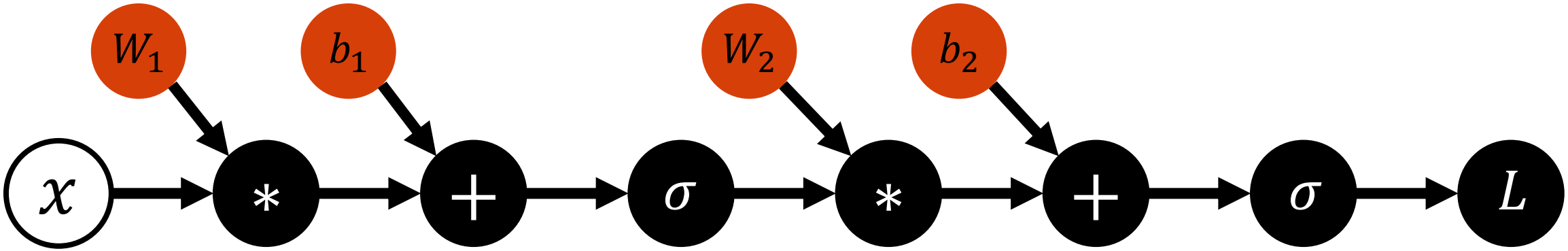
Forward Pass



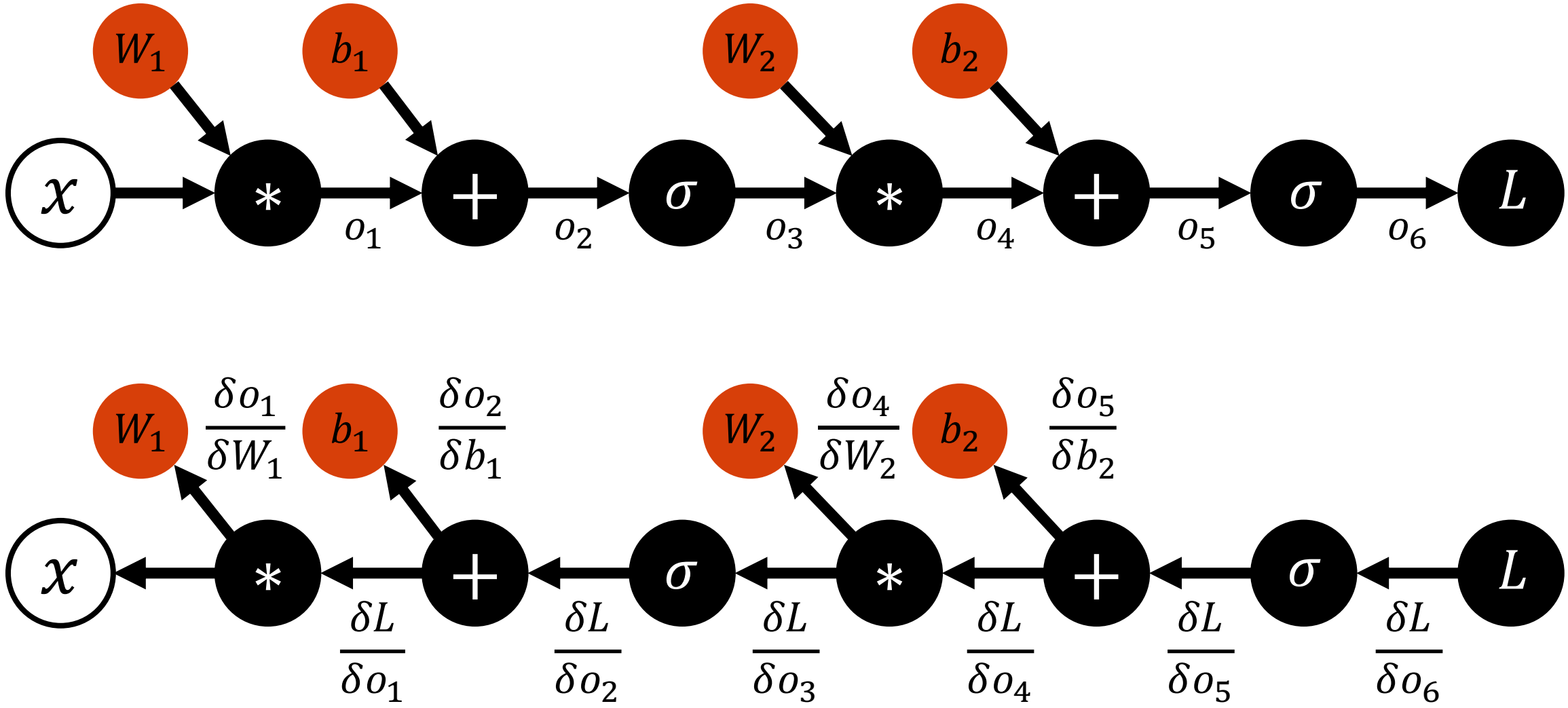
Backward Pass

Neural Networks as Computational Graphs





Neural Networks as Computational Graphs



Today's Learning Objectives

Be able to answer:

- What are the intuitions for some advanced structures in neural networks?

- What are decision trees?
- How are they learned?
  - What is entropy, conditional entropy, and information gain?
- How do they deal with continuous features?

- What do to about overfitting?

Intuition 1: Translational Invariance / Locality Assumption

A Toy Problem

Goal: Classify two 1's followed by two -1's as + and everything else as -

$$\boldsymbol{x} \in \{-\mathbf{1}, \mathbf{0}, \mathbf{1}\}^{\mathbf{4}}$$

$$\begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \end{bmatrix} \checkmark \qquad \begin{bmatrix} 0 \\ -1 \\ 0 \\ 1 \end{bmatrix} \quad \begin{bmatrix} 1 \\ 0 \\ -1 \\ -1 \end{bmatrix} \quad \begin{bmatrix} 0 \\ 1 \\ 1 \\ -1 \end{bmatrix} \quad \begin{bmatrix} 0 \\ 1 \\ 0 \\ -1 \end{bmatrix} \quad \begin{bmatrix} 0 \\ -1 \\ 0 \\ 1 \end{bmatrix}$$

$$\boldsymbol{a} = sign(\boldsymbol{W}^T \boldsymbol{x} + \boldsymbol{b})$$

$$\boldsymbol{W} = \begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \end{bmatrix} \qquad b = \begin{bmatrix} -3.5 \end{bmatrix}$$

A Toy Problem

Goal: Classify two 1's followed by two -1's as + and everything else as -

$$\begin{bmatrix}1\\1\\-1\\-1\end{bmatrix}^T\begin{bmatrix}1\\1\\-1\\-1\end{bmatrix}-3.5=0.5$$

$$\begin{bmatrix}1\\1\\-1\\-1\end{bmatrix}^T\begin{bmatrix}0\\1\\0\\-1\end{bmatrix}-3.5=-1.5$$

$$\begin{bmatrix}1\\1\\-1\\-1\end{bmatrix}^T\begin{bmatrix}0\\-1\\0\\1\end{bmatrix}-3.5=-6.5$$

$$\begin{bmatrix}1\\1\\-1\\-1\end{bmatrix}^T\begin{bmatrix}0\\1\\1\\-1\end{bmatrix}-3.5=-2.5$$

$$\begin{bmatrix}1\\1\\-1\\-1\end{bmatrix}^T\begin{bmatrix}0\\-1\\0\\1\end{bmatrix}-3.5=-5.5$$

$$\begin{bmatrix}1\\1\\-1\\-1\end{bmatrix}^T\begin{bmatrix}1\\0\\-1\\-1\end{bmatrix}-3.5=-0.5$$

A Slightly Bigger Toy Problem

Goal: Classify any signal with two 1's followed by two -1's as positive.

$$\boldsymbol{x} \in \{-\mathbf{1}, \mathbf{0}, \mathbf{1}\}^{\mathbf{8}}$$

$$\begin{pmatrix} 0 \\ 1 \\ 1 \\ -1 \\ -1 \\ 1 \\ 1 \\ 0 \end{pmatrix} \quad \begin{pmatrix} 1 \\ 1 \\ -1 \\ -1 \\ -1 \\ 1 \\ 1 \\ 0 \end{pmatrix} \quad \begin{pmatrix} 1 \\ 0 \\ -1 \\ -1 \\ 1 \\ 1 \\ -1 \\ -1 \end{pmatrix} \quad \begin{pmatrix} 0 \\ 1 \\ 0 \\ -1 \\ 1 \\ 1 \\ 1 \\ 0 \end{pmatrix} \quad \begin{pmatrix} 0 \\ -1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} \quad \begin{pmatrix} 0 \\ -1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

$$\boldsymbol{a} = \max(sign(\boldsymbol{W}^T \boldsymbol{x} + \boldsymbol{b}))$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ -1 & 1 & 1 & 0 & 0 \\ -1 & -1 & 1 & 1 & 0 \\ 0 & -1 & -1 & 1 & 1 \\ 0 & 0 & -1 & -1 & 1 \\ 0 & 0 & 0 & -1 & -1 \\ 0 & 0 & 0 & 0 & -1 \end{pmatrix} \quad \begin{pmatrix} -3.5 \\ -3.5 \\ -3.5 \\ -3.5 \\ -3.5 \end{pmatrix}$$

A Slightly Bigger Toy Problem

Goal: Classify any signal with two 1's followed by two -1's as positive.

$$\boldsymbol{x} \in \{-\mathbf{1}, \mathbf{0}, \mathbf{1}\}^{\mathbf{8}}$$

$$\boldsymbol{a} = \max(sign(\boldsymbol{W}^T \boldsymbol{x} + \boldsymbol{b}))$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ -1 & 1 & 1 & 0 & 0 \\ -1 & -1 & 1 & 1 & 0 \\ 0 & -1 & -1 & 1 & 1 \\ 0 & 0 & -1 & -1 & 1 \\ 0 & 0 & 0 & -1 & -1 \\ 0 & 0 & 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 1 \\ -1 \\ -1 \\ 1 \\ 1 \\ 0 \end{pmatrix} + \begin{pmatrix} -3.5 \\ -3.5 \\ -3.5 \\ -3.5 \\ -3.5 \end{pmatrix} = \begin{pmatrix} 1-3.5 \\ 4-3.5 \\ 0-3.5 \\ -4-3.5 \\ -1-3.5 \end{pmatrix} \xrightarrow{\max(sign(\cdot))} 1$$

A Slightly Bigger Toy Problem

Goal: Classify any signal with two 1's followed by two -1's as positive.

$$\boldsymbol{x} \in \{-\mathbf{1}, \mathbf{0}, \mathbf{1}\}^{\mathbf{8}}$$

$$\boldsymbol{a} = \max(sign(\boldsymbol{W}^T \boldsymbol{x} + \boldsymbol{b}))$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ -1 & 1 & 1 & 0 & 0 \\ -1 & -1 & 1 & 1 & 0 \\ 0 & -1 & -1 & 1 & 1 \\ 0 & 0 & -1 & -1 & 1 \\ 0 & 0 & 0 & -1 & -1 \\ 0 & 0 & 0 & 0 & -1 \end{pmatrix}^T \begin{pmatrix} 0 \\ 1 \\ 0 \\ -1 \\ 1 \\ 1 \\ 1 \\ 0 \end{pmatrix} + \begin{pmatrix} -3.5 \\ -3.5 \\ -3.5 \\ -3.5 \\ -3.5 \end{pmatrix} = \begin{pmatrix} 2 - 3.5 \\ 1 - 3.5 \\ -3 - 3.5 \\ -2 - 3.5 \\ 1 - 3.5 \end{pmatrix} \xrightarrow[\max(sign(\cdot))]{} 0$$

Questions Break!

Consider finding the presence of a pattern $[1, 1, -1, -1]^T$ in a d dimensional vector $\boldsymbol{x} \in \{-1,0,1\}^d$. How many columns would our weight matrix need to have if we follow the same schema as before?

$\boldsymbol{x} \in \{-1,0,1\}^4$

$$\begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \end{bmatrix}$$

$\mathbf{4 \times 1}$

$\boldsymbol{x} \in \{-1,0,1\}^8$

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ -1 & 1 & 1 & 0 & 0 \\ -1 & -1 & 1 & 1 & 0 \\ 0 & -1 & -1 & 1 & 1 \\ 0 & 0 & -1 & -1 & 1 \\ 0 & 0 & 0 & -1 & -1 \\ 0 & 0 & 0 & 0 & -1 \end{bmatrix}$$

$\mathbf{8 \times 5}$

$\boldsymbol{x} \in \{-1,0,1\}^d$

?

$\boldsymbol{d} \times ?$

A Slightly Bigger Toy Problem

Goal: Classify any signal with two 1's followed by two -1's as positive.

$$\begin{bmatrix} 0 \\ 1 \\ 1 \\ -1 \\ -1 \\ 1 \\ 1 \\ 0 \end{bmatrix} \qquad \begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \end{bmatrix} \qquad \begin{bmatrix} \\ \\ \\ \\ \\ \\ \\ \\ \end{bmatrix}$$

A Slightly Bigger Toy Problem

Goal: Classify any signal with two 1's followed by two -1's as positive.

$$\begin{bmatrix} 0 \\ 1 \\ 1 \\ -1 \\ -1 \\ 1 \\ 1 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 0 \\ 1 \\ 1 \\ -1 \end{bmatrix}^T \begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \end{bmatrix} \quad \begin{bmatrix} 1 \\ \\ \\ \\ \\ \\ \\ \end{bmatrix}$$

A Slightly Bigger Toy Problem

Goal: Classify any signal with two 1's followed by two -1's as positive.

$$\begin{pmatrix} 0 \\ 1 \\ 1 \\ -1 \\ -1 \\ 1 \\ 1 \\ 0 \end{pmatrix} \quad \begin{pmatrix} 1 \\ 1 \\ -1 \\ -1 \end{pmatrix}^T \begin{pmatrix} 1 \\ 1 \\ -1 \\ -1 \end{pmatrix} \quad \begin{pmatrix} 1 \\ 4 \end{pmatrix}$$

A Slightly Bigger Toy Problem

Goal: Classify any signal with two 1's followed by two -1's as positive.

$$\begin{bmatrix} 0 \\ 1 \\ 1 \\ -1 \\ -1 \\ 1 \\ 1 \\ 0 \end{bmatrix} \qquad \begin{bmatrix} 1 \\ -1 \\ -1 \\ 1 \end{bmatrix}^T \begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \end{bmatrix} \qquad \begin{bmatrix} 1 \\ 4 \\ 0 \end{bmatrix}$$

A Slightly Bigger Toy Problem

Goal: Classify any signal with two 1's followed by two -1's as positive.

$$\begin{pmatrix} 0 \\ 1 \\ 1 \\ -1 \\ -1 \\ 1 \\ 1 \\ 0 \end{pmatrix} \quad \begin{pmatrix} -1 \\ -1 \\ 1 \\ 1 \end{pmatrix}^T \begin{pmatrix} 1 \\ 1 \\ -1 \\ -1 \end{pmatrix} \quad \begin{pmatrix} 1 \\ 4 \\ 0 \\ -4 \end{pmatrix}$$

A Slightly Bigger Toy Problem

Goal: Classify any signal with two 1's followed by two -1's as positive.

$$\begin{pmatrix} 0 \\ 1 \\ 1 \\ -1 \\ -1 \\ 1 \\ 1 \\ 0 \end{pmatrix} \quad \begin{pmatrix} -1 \\ 1 \\ 1 \\ 0 \end{pmatrix}^T \begin{pmatrix} 1 \\ 1 \\ -1 \\ -1 \end{pmatrix} \quad \begin{pmatrix} 1 \\ 4 \\ 0 \\ -4 \\ -1 \end{pmatrix}$$

Convolutions in 1D

$$(f \circledast g)(t) = \int_{-\infty}^{\infty} f(\tau) g(t - \tau) d\tau$$

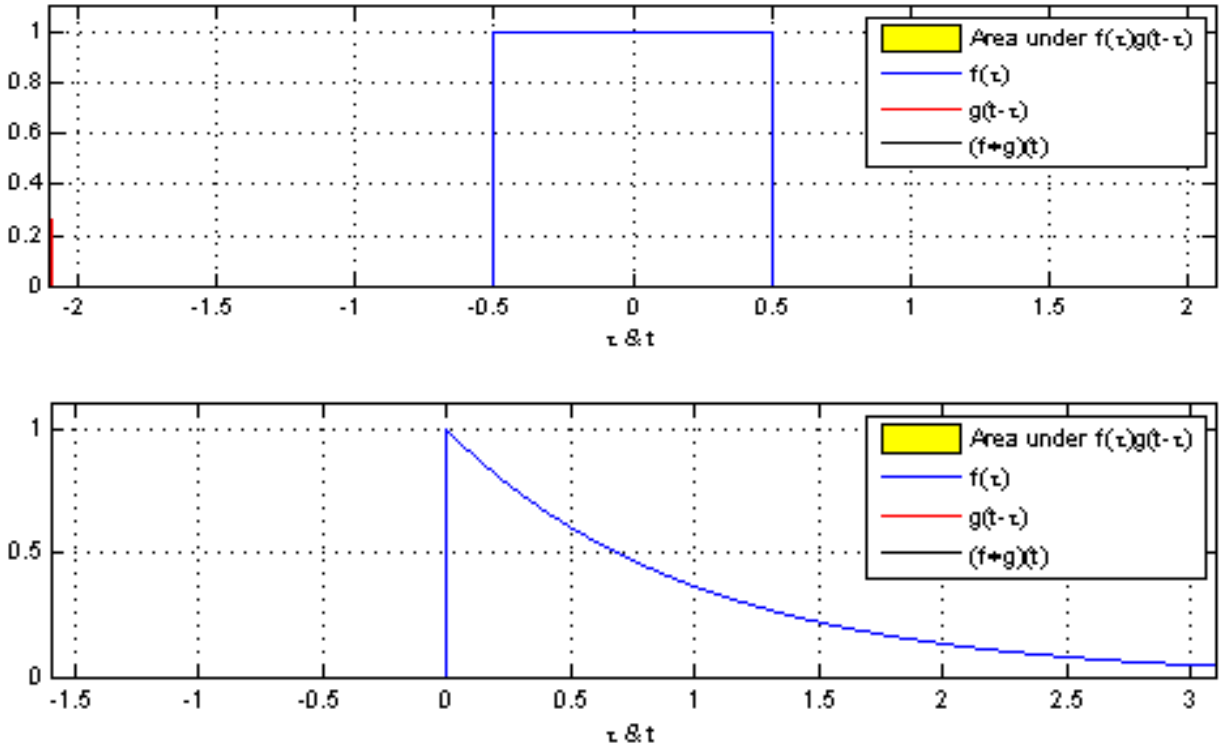
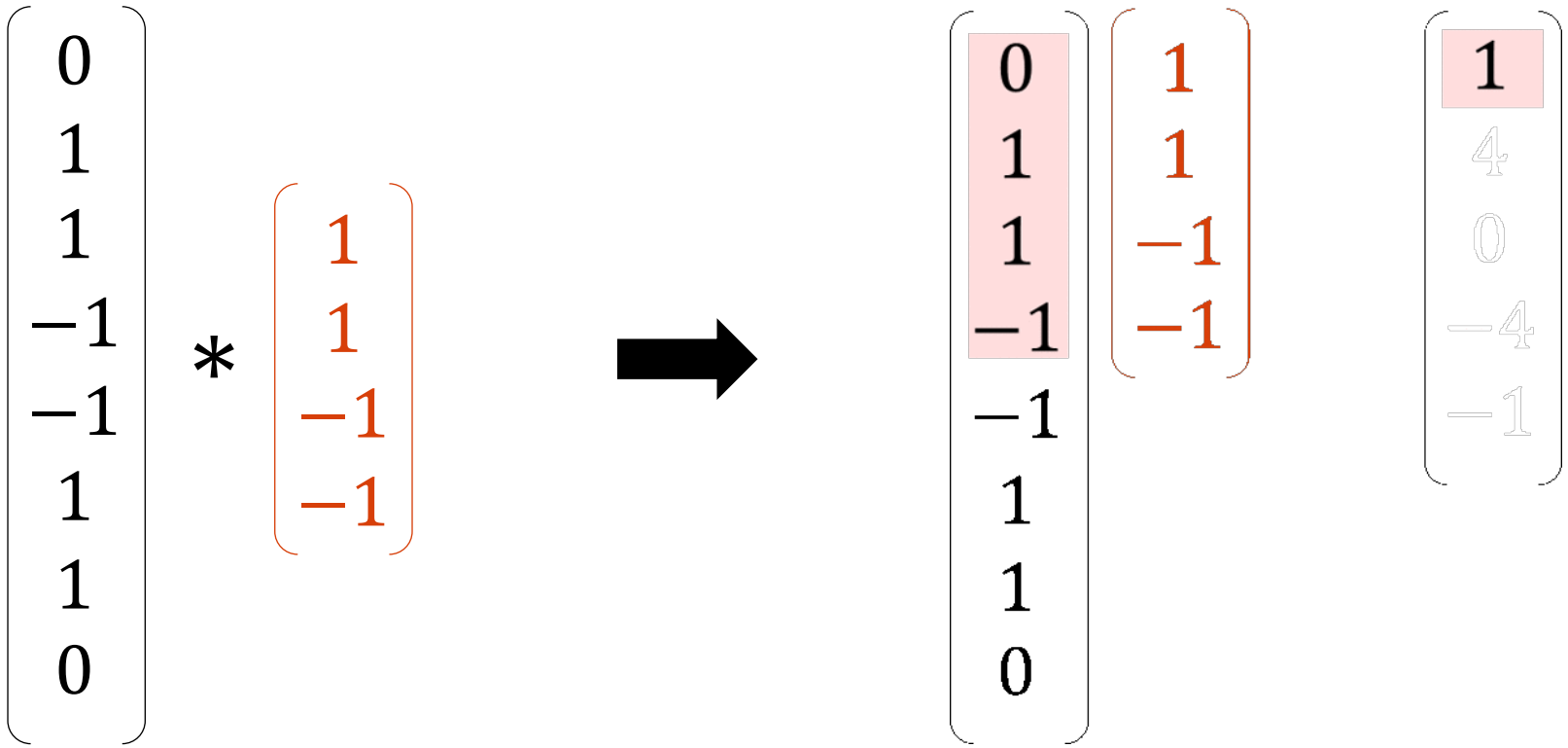


Image credit: https://en.wikipedia.org/wiki/Convolution

Discrete Convolutions in 1D

$$(f \circledast g)(t) = \sum_{\tau=0}^{N} f[t+\tau]g[\tau], \qquad g \in \mathbb{R}^N$$



Convolution's Inductive Bias

Goal: Classify any signal with two 1's followed by two -1's as positive.

$$\boldsymbol{a} = \max(sign(\boldsymbol{W}^T\boldsymbol{x} + \boldsymbol{b})) \longrightarrow \boldsymbol{a} = \max(sign(\boldsymbol{w} \circledast \boldsymbol{x} + \boldsymbol{b}))$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ -1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & -1 & -1 \\ 0 & 0 & 0 & 0 & -1 \end{pmatrix} \quad \begin{pmatrix} -3.5 \\ \\ \\ \\ \\ -3.5 \end{pmatrix} \quad \boldsymbol{W} \quad \begin{pmatrix} \\ \\ \\ \\ \\ -1 \end{pmatrix}$$

SAVED PARAMETERS BY MAKING A LOCALITY ASSUMPTION!
ALSO KNOWN AS "TRANSLATIONAL INVARIANCE"

Example where the Locality Bias Fits

Goal: Have a feature respond to the word “Alexa”

f_{Alexa}

Example where the Locality Bias Fits

Goal: Have a feature respond to the word “Alexa”

f_{Alexa}

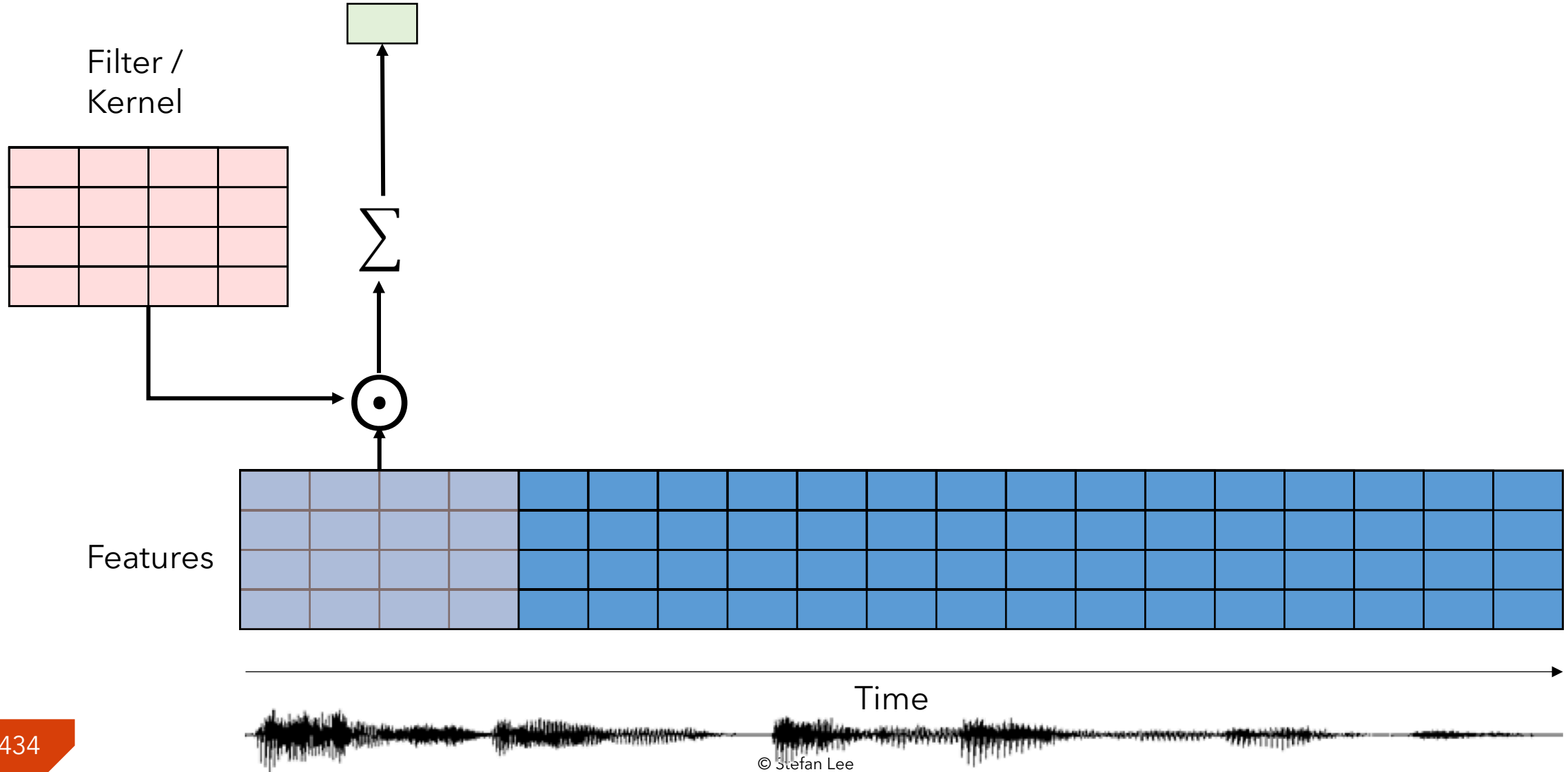
Example where the Locality Bias Fits

Goal: Have a feature respond to the word “Alexa”

f_{Alexa}

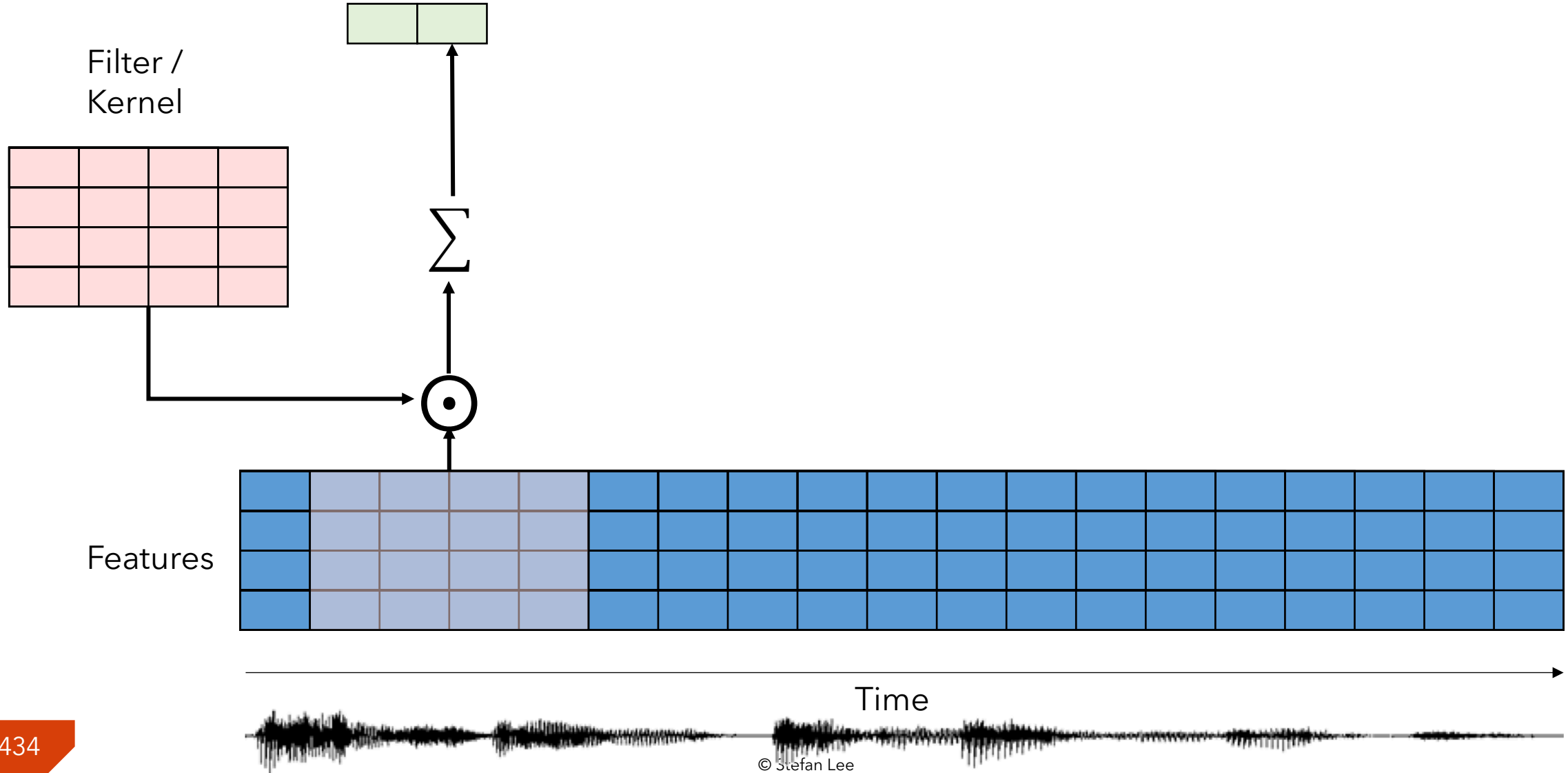
Example where the Locality Bias Fits (1D)

Goal: Have a feature respond to the word “Alexa”



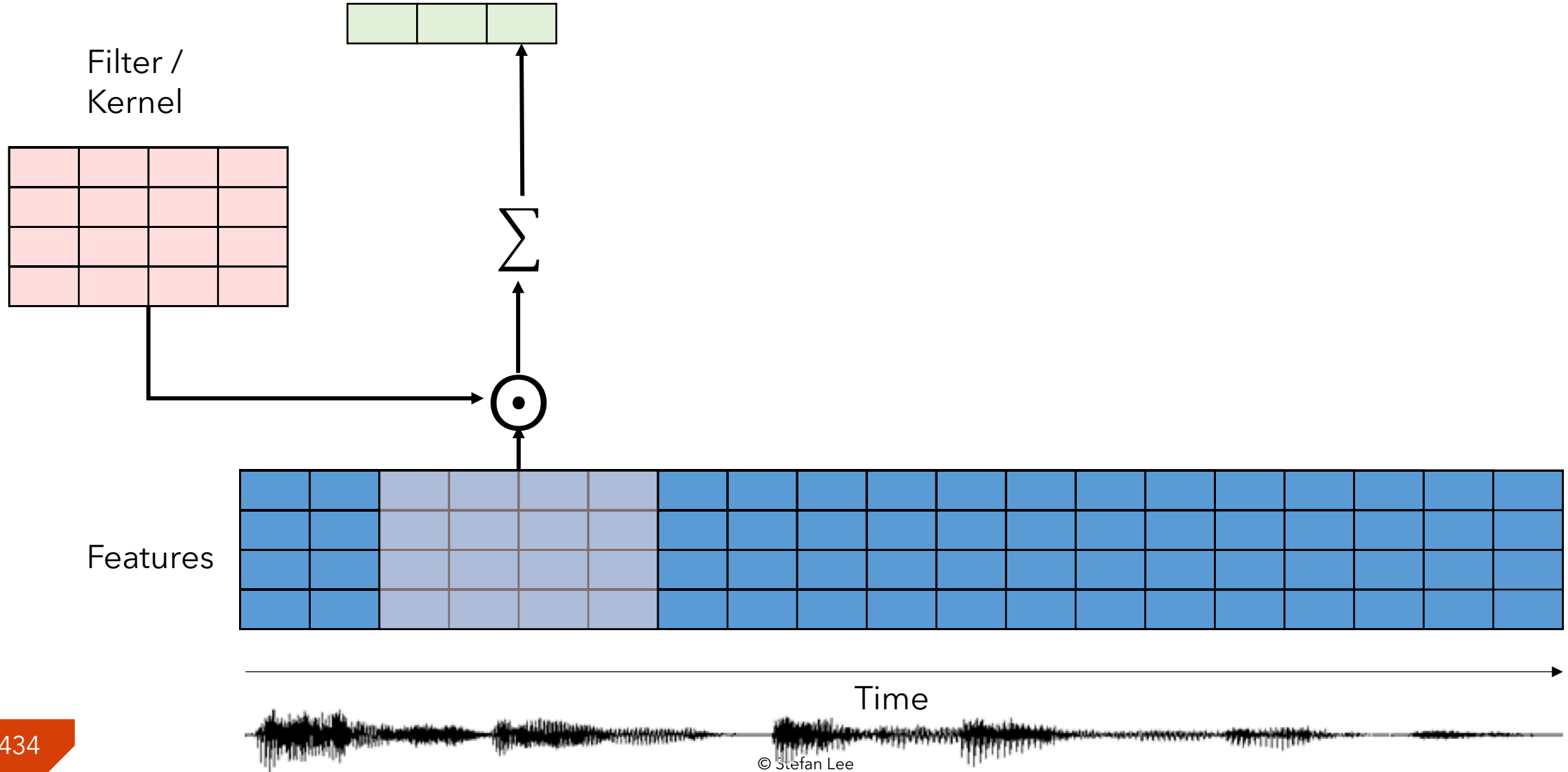
Example where the Locality Bias Fits (1D)

Goal: Have a feature respond to the word “Alexa”



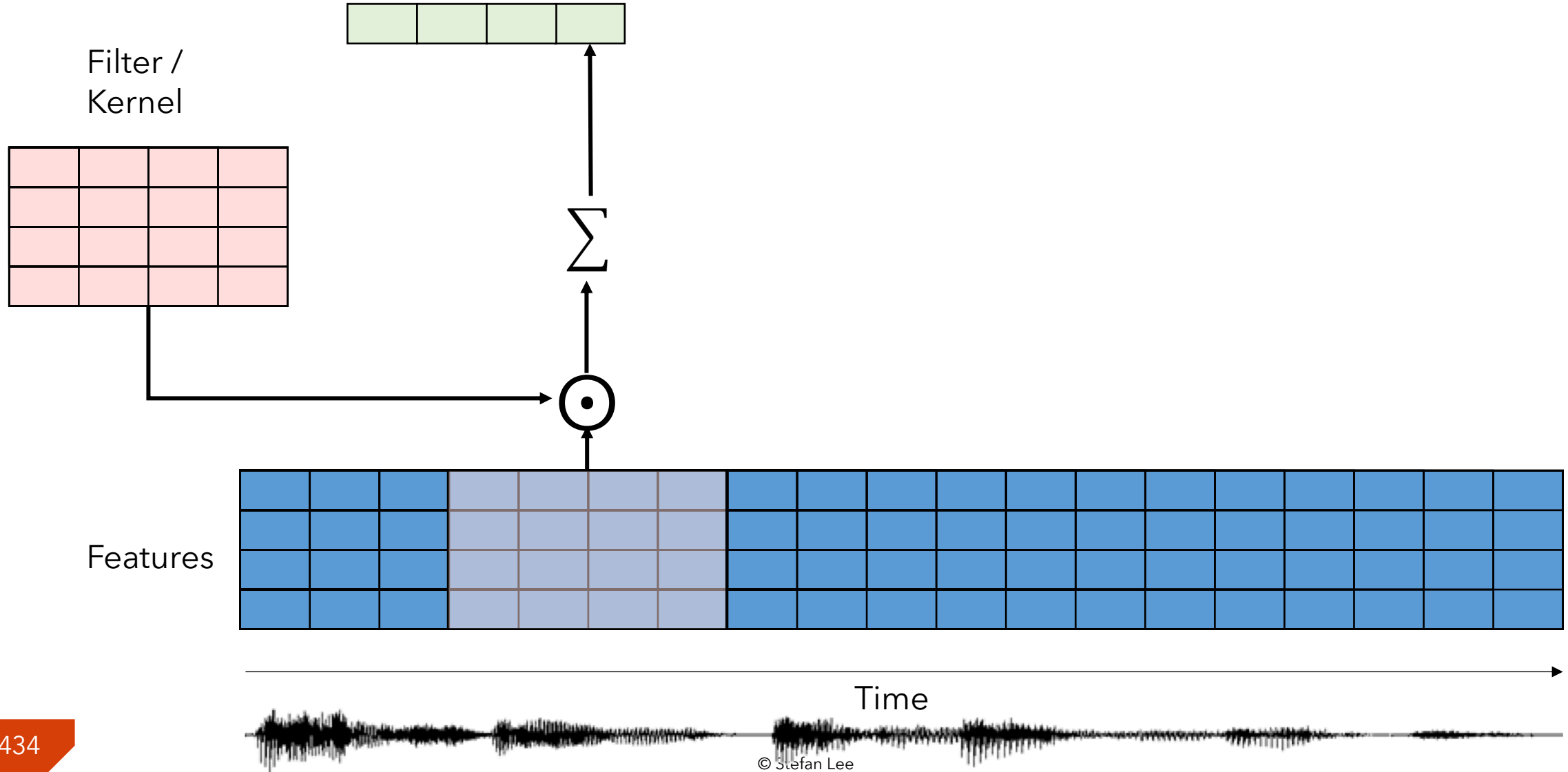
Example where the Locality Bias Fits (1D)

Goal: Have a feature respond to the word “Alexa”



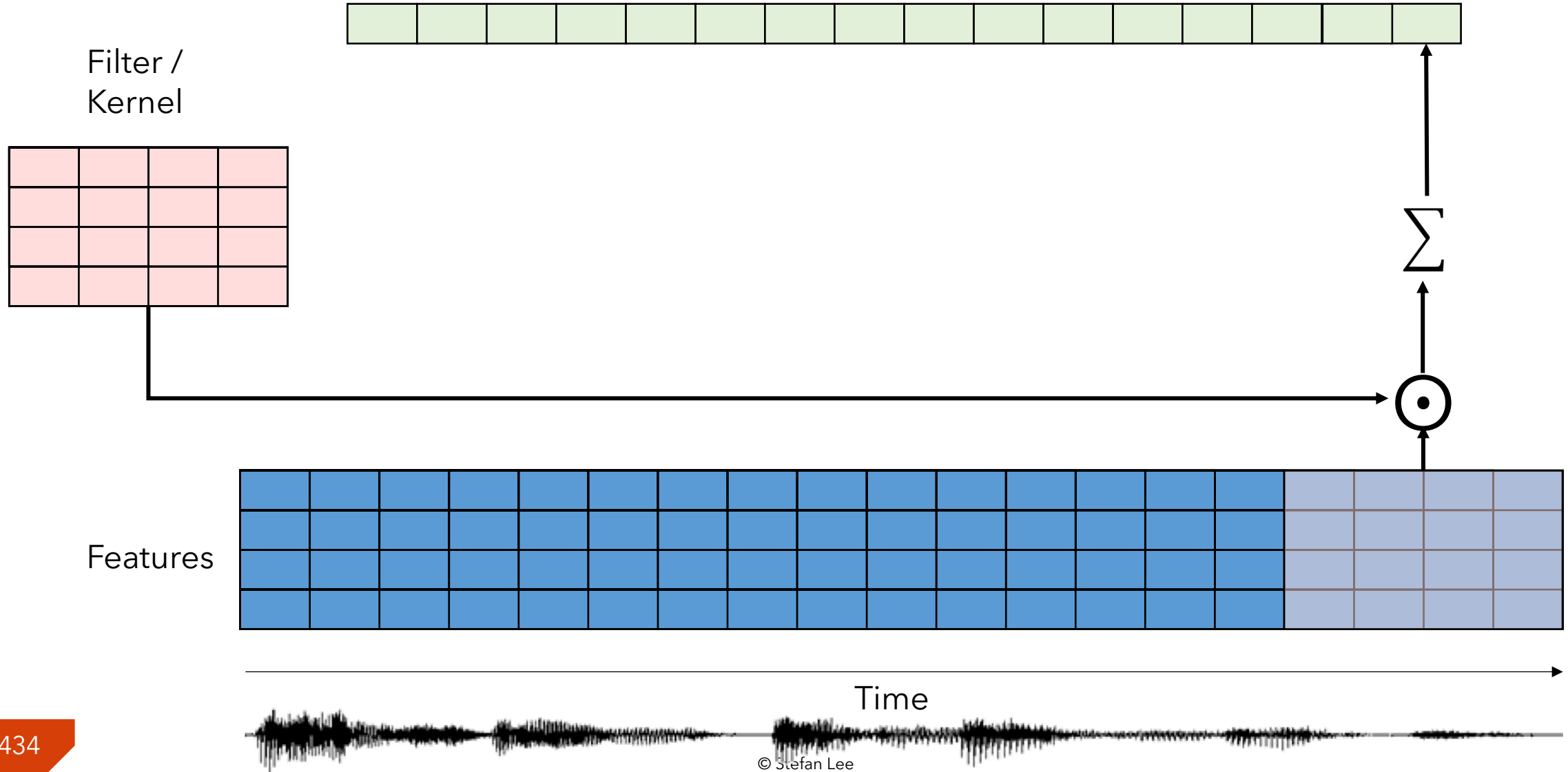
Example where the Locality Bias Fits (1D)

Goal: Have a feature respond to the word “Alexa”



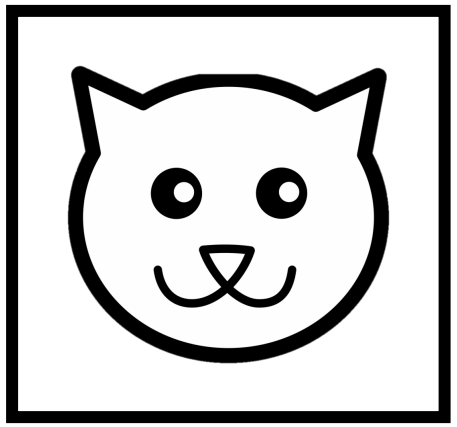
Example where the Locality Bias Fits (1D)

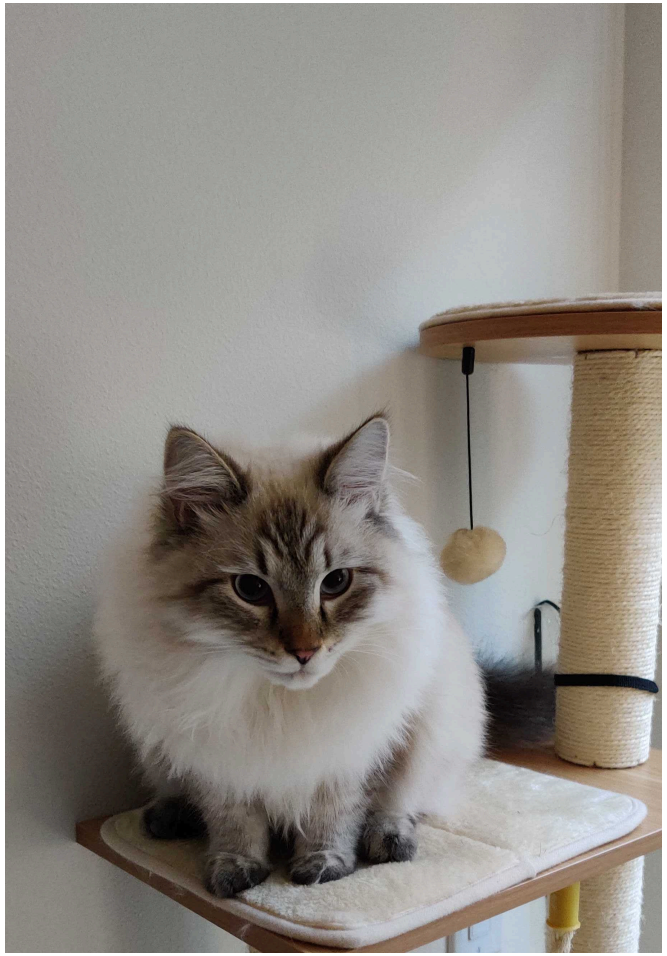
Goal: Have a feature respond to the word “Alexa”



Example where the Locality Bias Fits

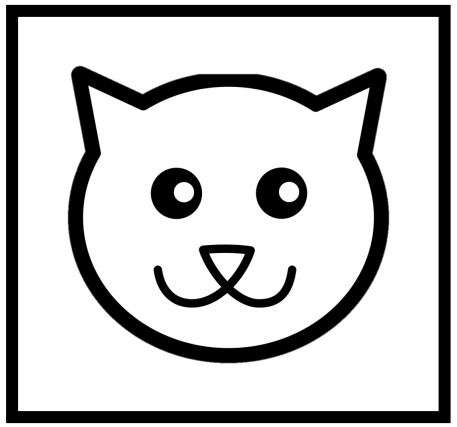
Goal: Have a feature respond to cat faces

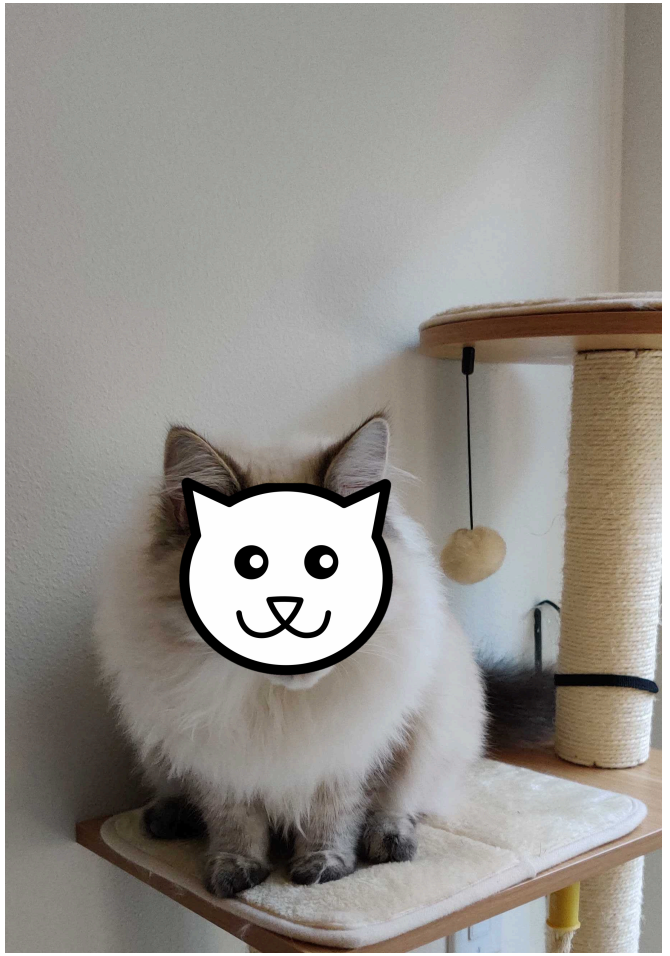




Example where the Locality Bias Fits

Goal: Have a feature respond to cat faces





Discrete Convolution in 2D

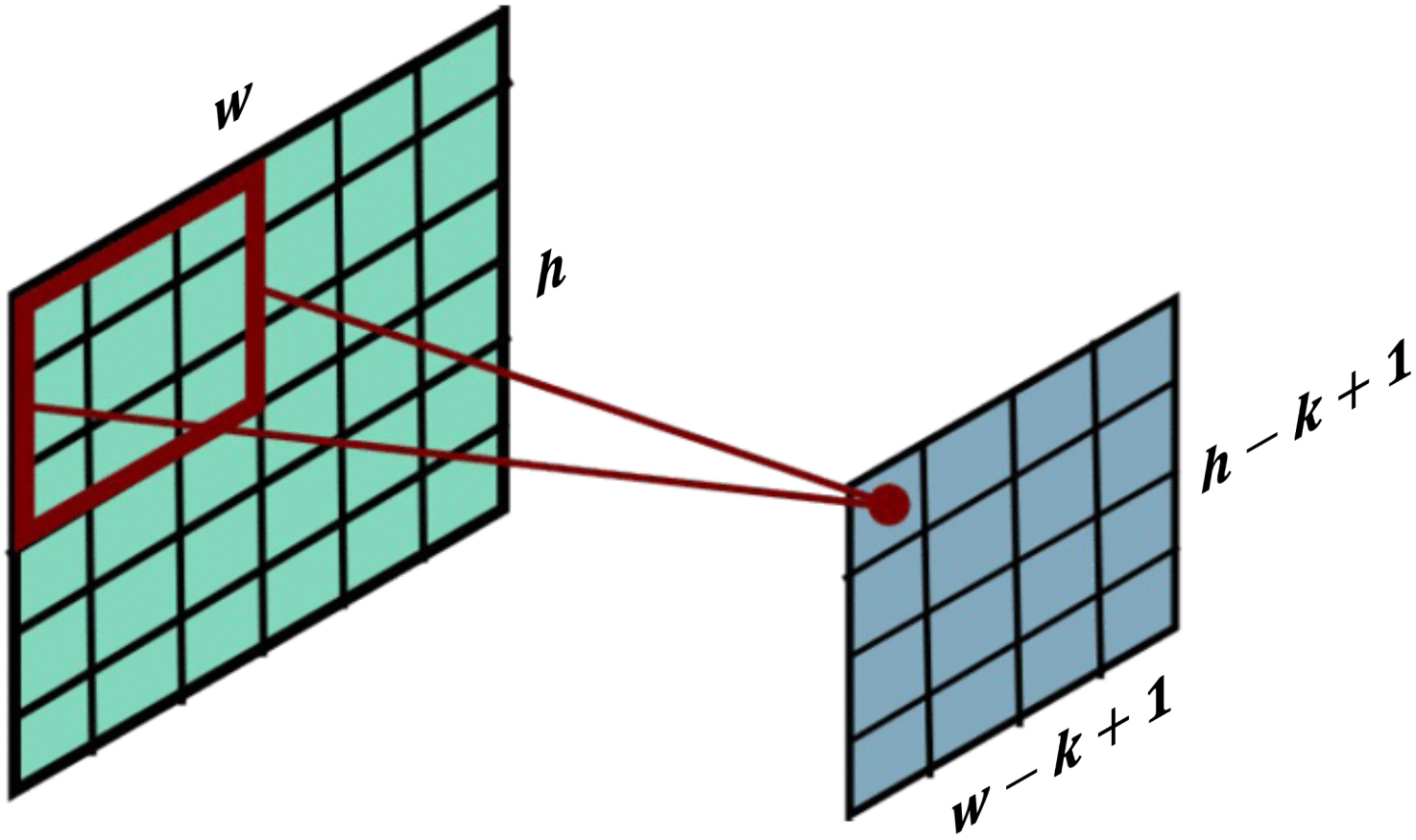
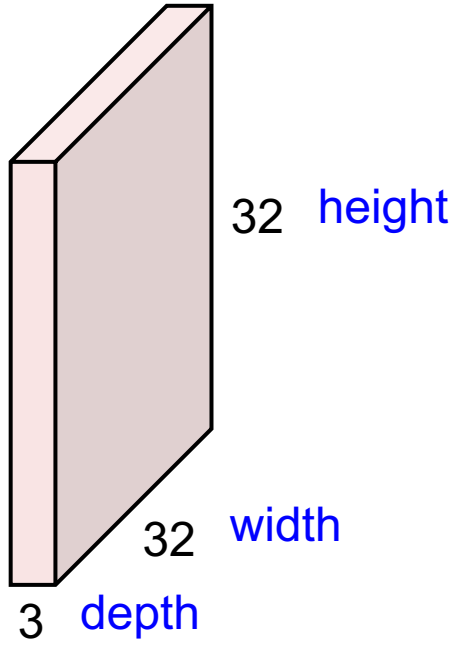


Image credit: Marc'Aurelio Ranzato

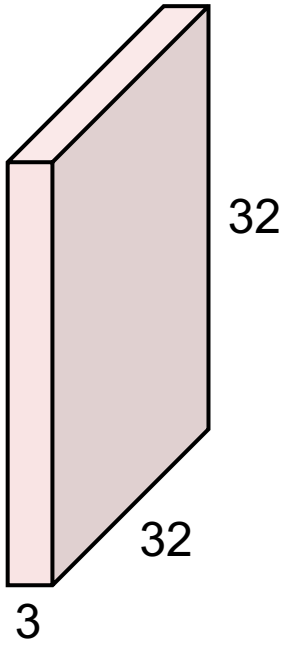
Discrete Convolution in 2D

32x32x3 image -> preserve spatial structure

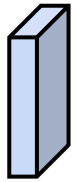


Discrete Convolution in 2D

32x32x3 image

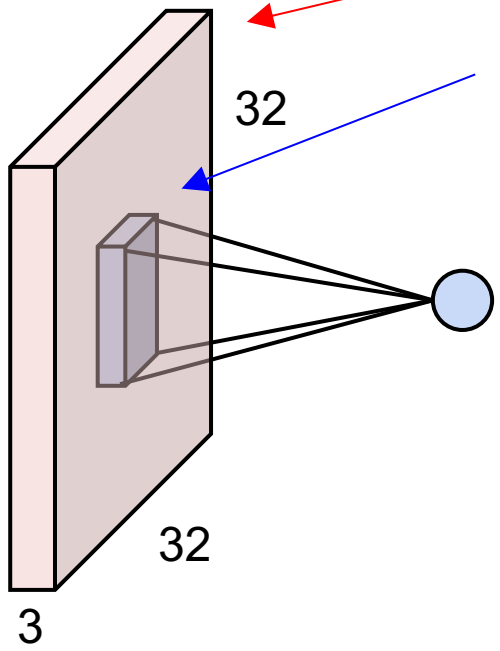


5x5x3 filter



Convolve the filter with the image
i.e. “slide over the image spatially, computing dot products”

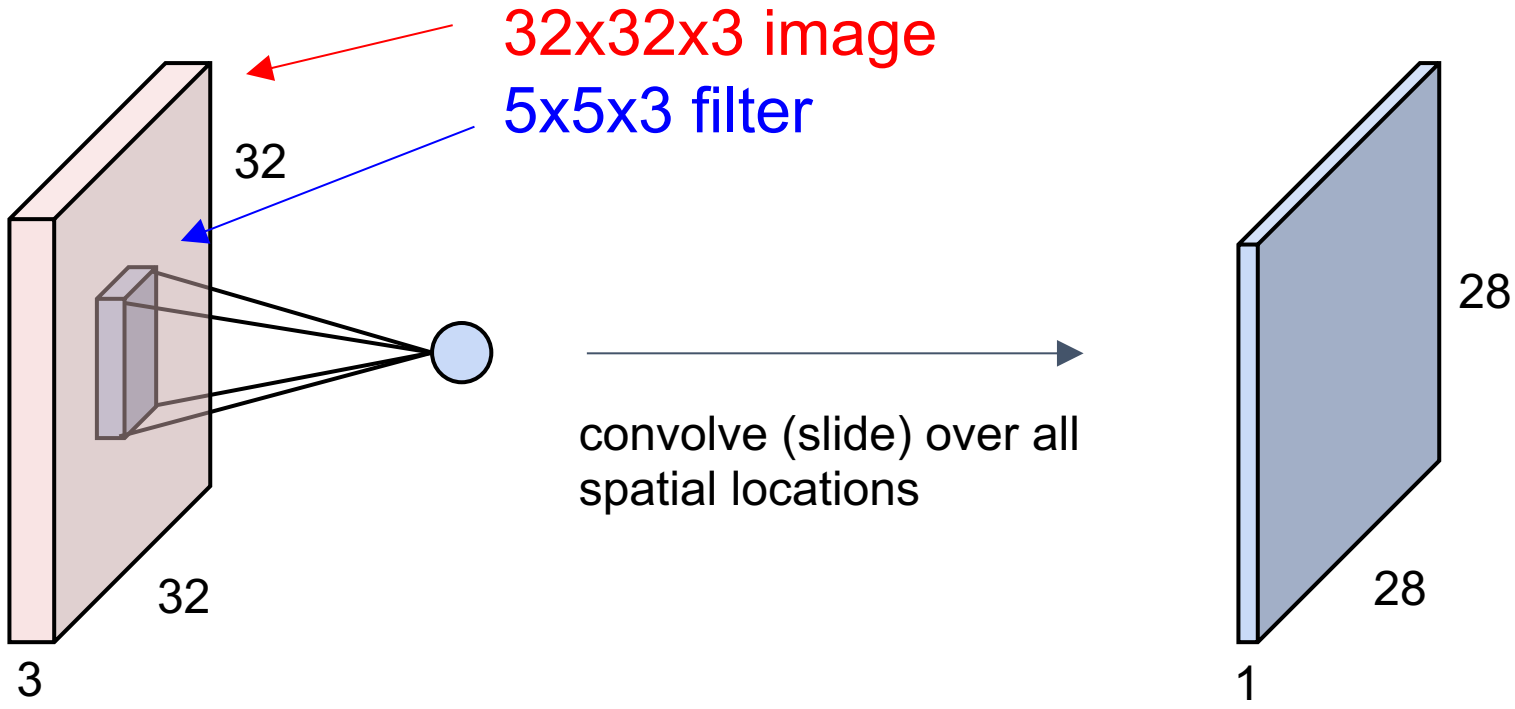
Discrete Convolution in 2D



1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)

$$w^T x + b$$

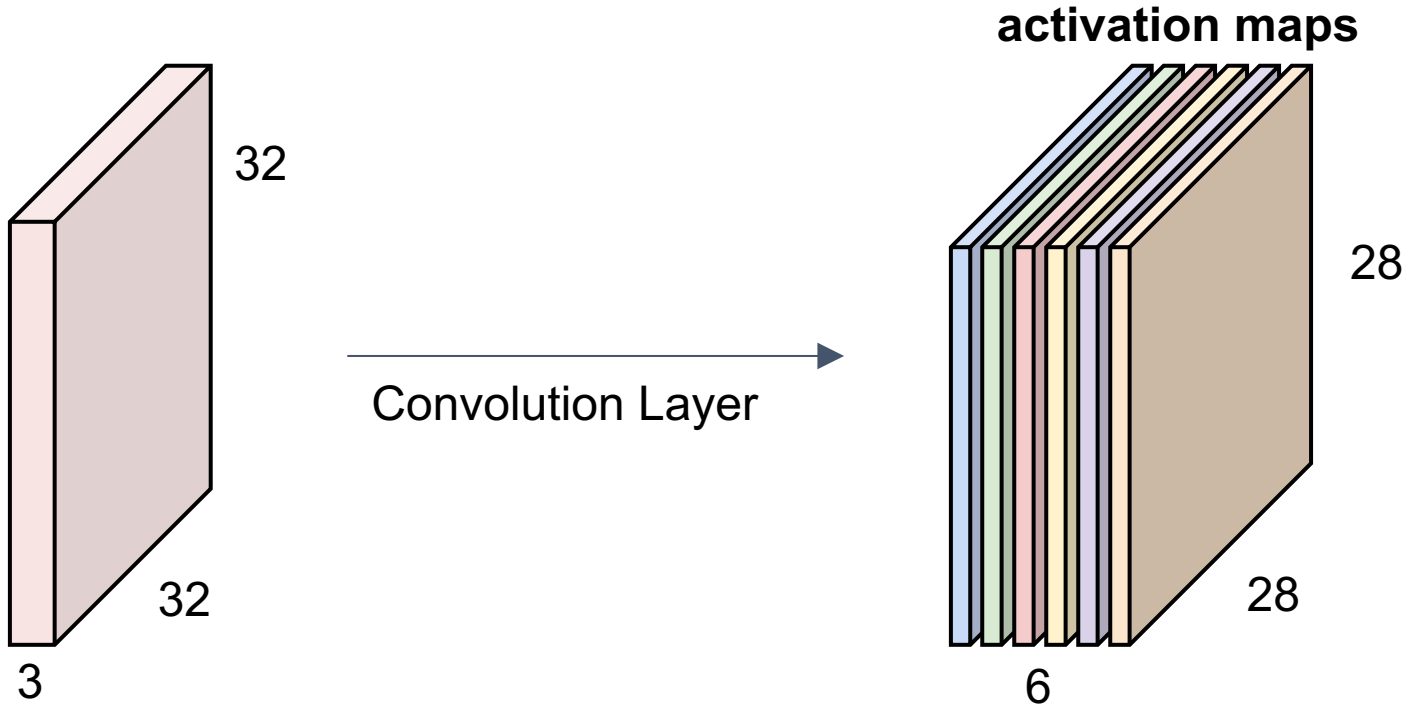
Discrete Convolution in 2D



consider a second, green filter

Discrete Convolution in 2D

For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



We stack these up to get a "new image" of size 28x28x6!

Convolutional Neural Networks

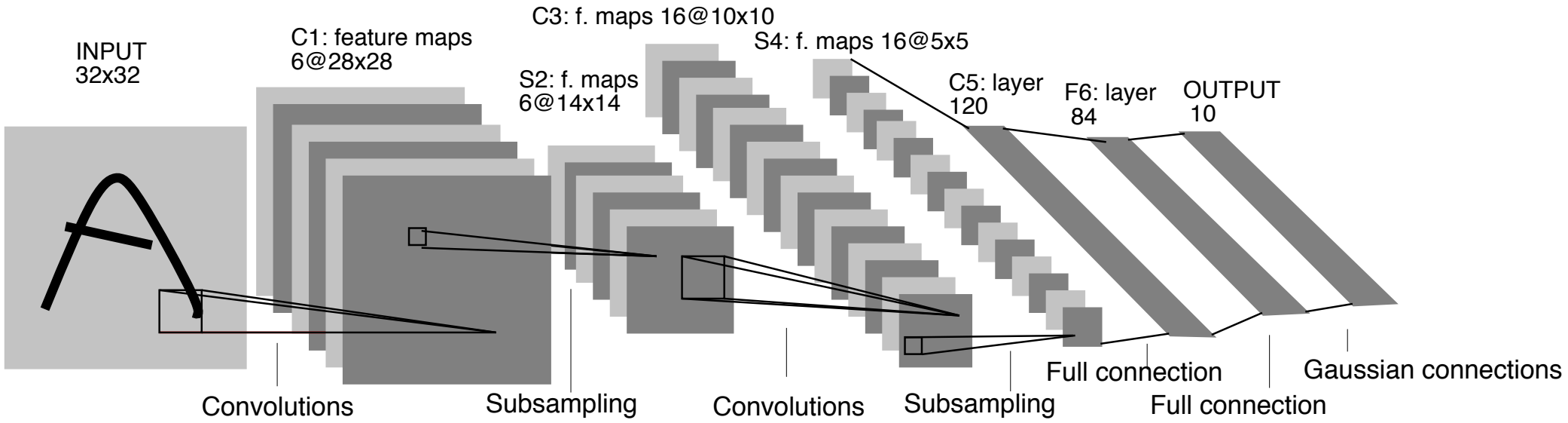


Image credit: Yann LeCun , Kevin Murphy

What do convolutional layers learn from images?

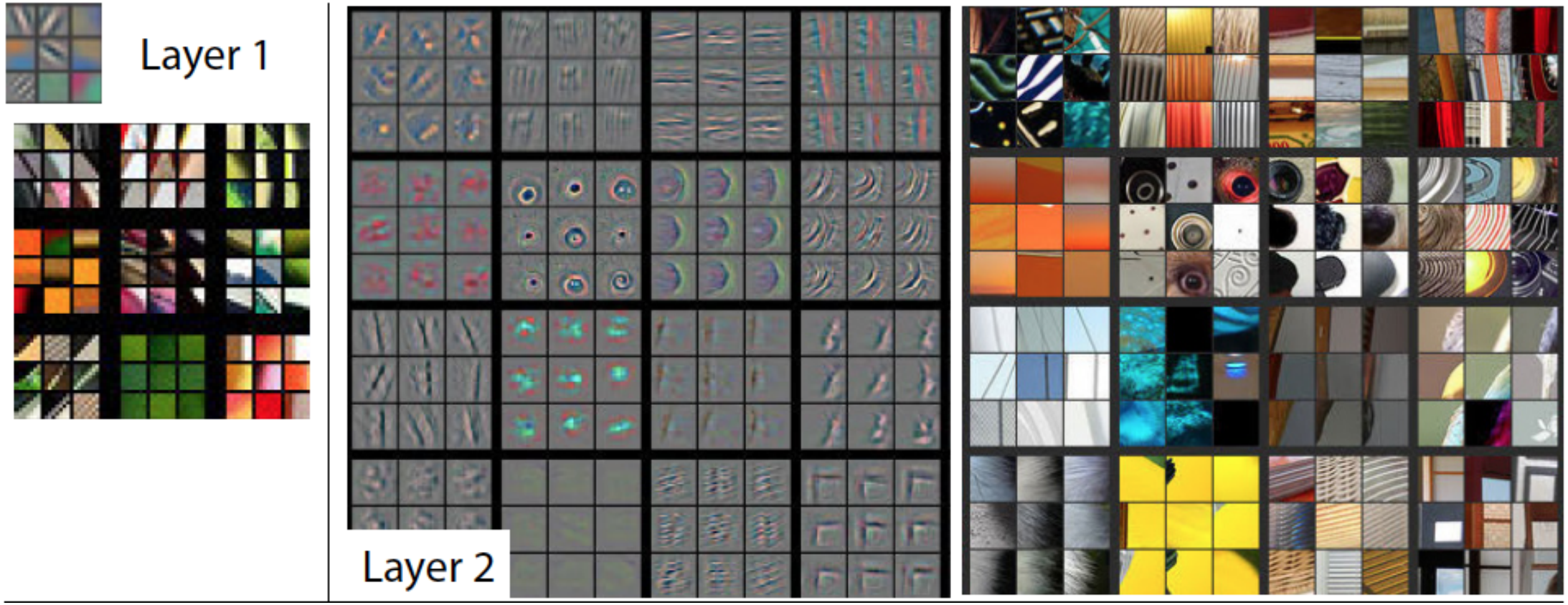


Image credit: [Zeiler & Fergus ECCV2014]

What do convolutional layers learn from images?

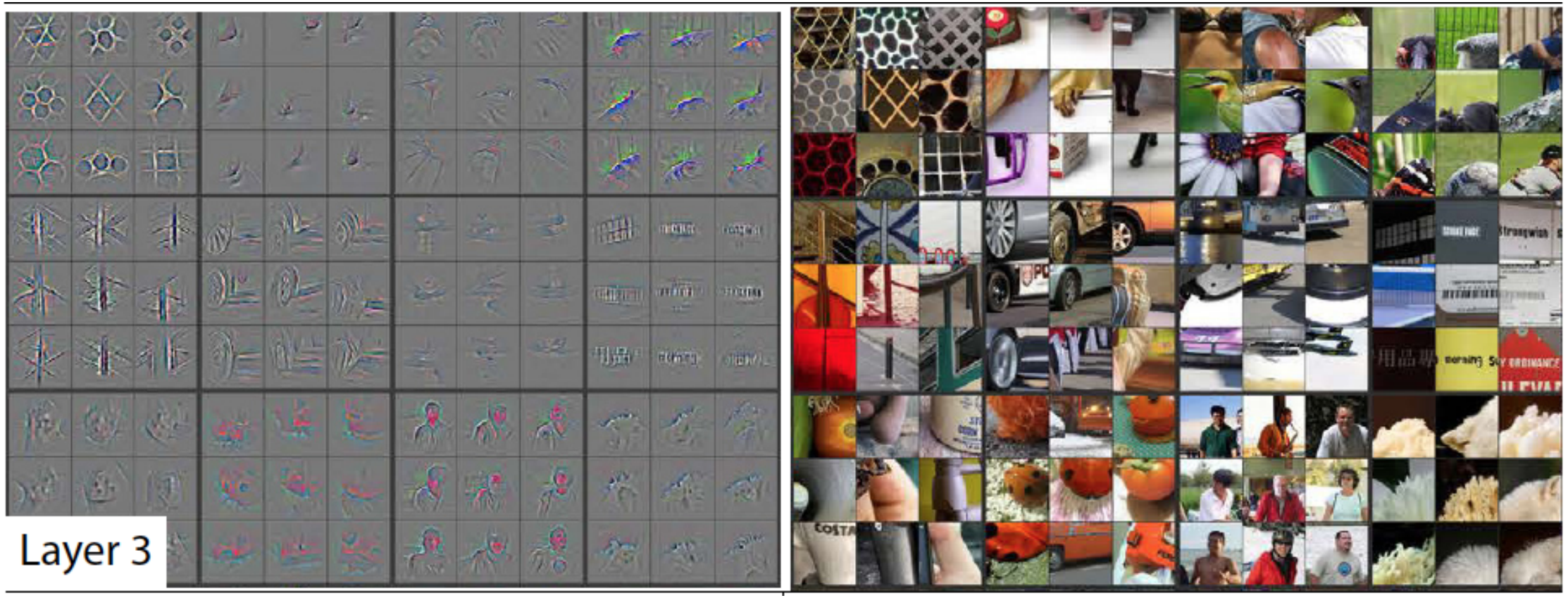


Image credit: [Zeiler & Fergus ECCV2014]

What do convolutional layers learn from images?

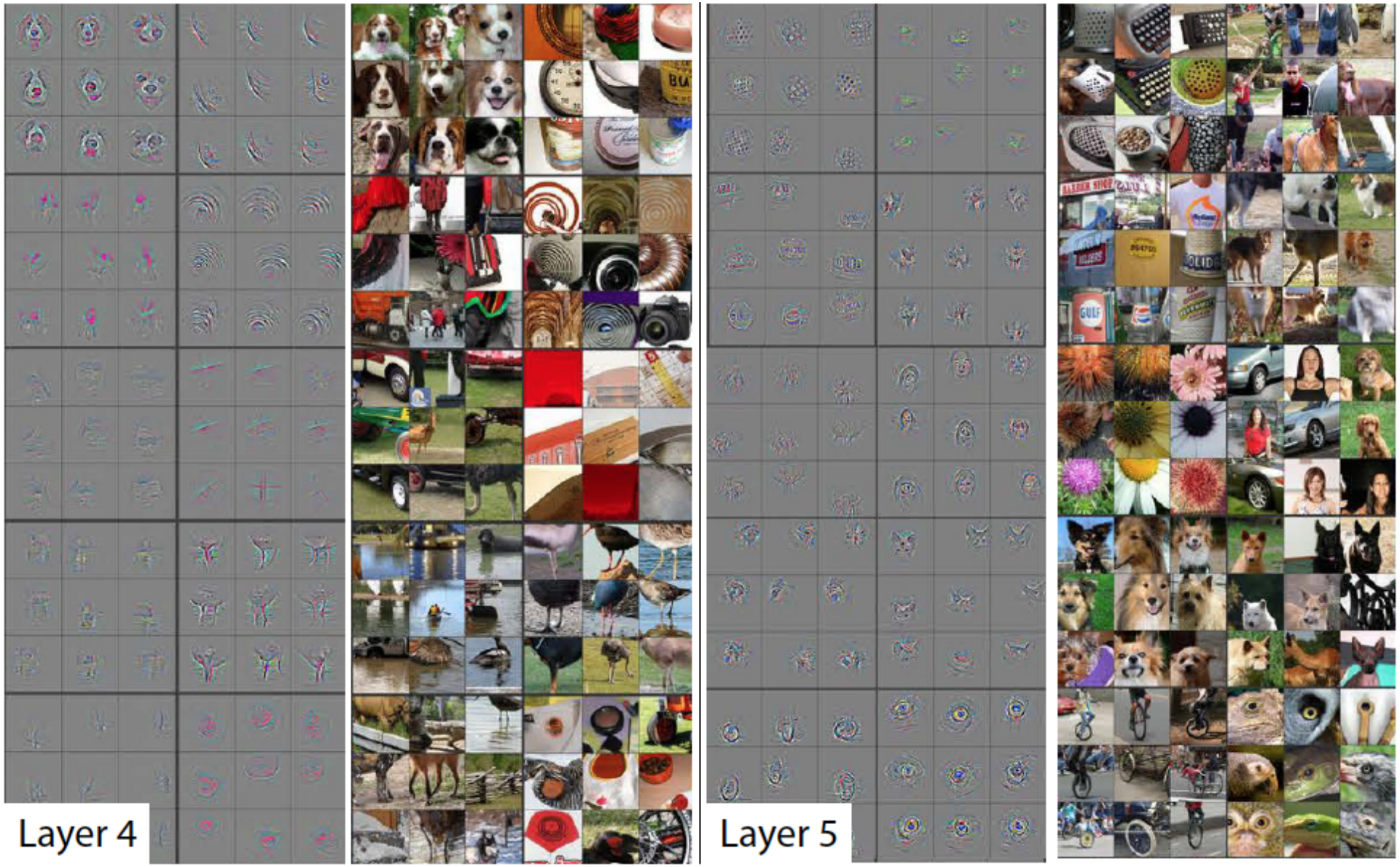


Image credit: [Zeiler & Fergus ECCV2014]

Intuition 2: Recurrence

A Toy Problem

Goal: Classify positively is an even number of 1's, negative for odd. $x \in \{0,1\}^?$

$$\begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}$$

CS 101 Online Parity Algorithm:

```
parity =0
for bit in x:
    parity = (parity + bit) % 2

return parity
```

Core actions repeated over the sequence:

- Store a memory of the state
- Update that memory based on local information

A Toy Problem

Goal: Classify positively is an even number of 1's, negative for odd. $\boldsymbol{x} \in \{\mathbf{0}, \mathbf{1}\}^?$

$$\begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}$$

CS 102 Online Parity Algorithm:

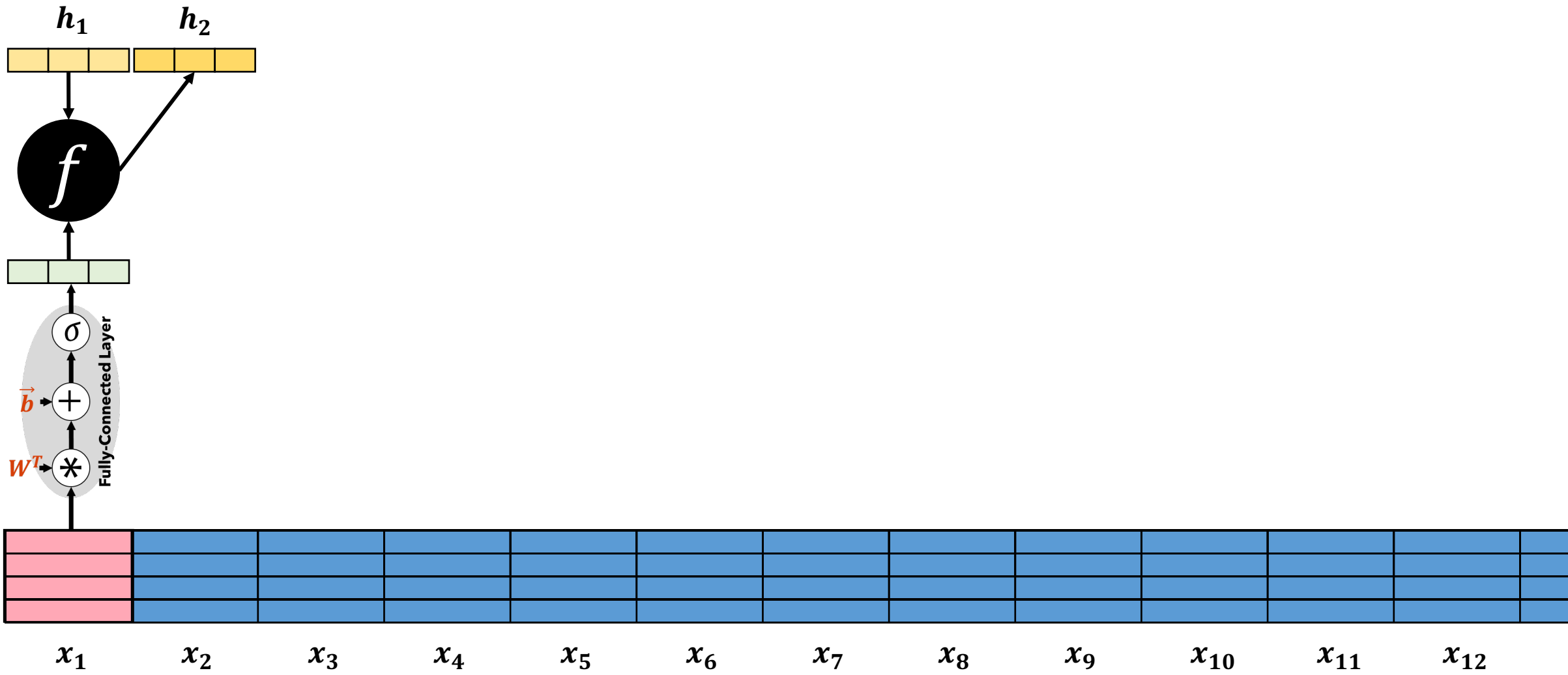
```
def parity(x):
    if x == None: return 0
    return (x[-1] + parity(x[:-1]))%2
```

Core actions repeated over the sequence:

- Store a memory of the state
- Update that memory based on local information

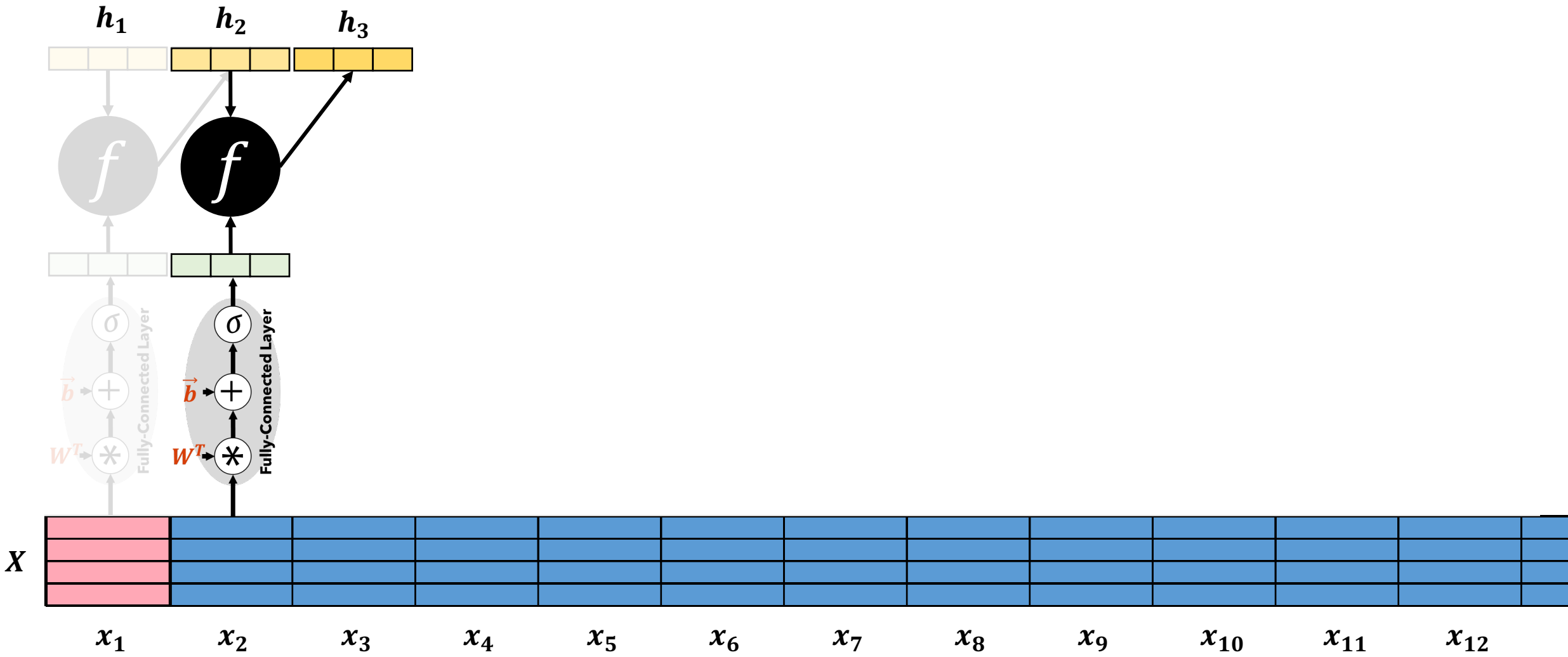
Variable Input → Fixed Output

Idea- Keep a memory around and update it with the same function over time.



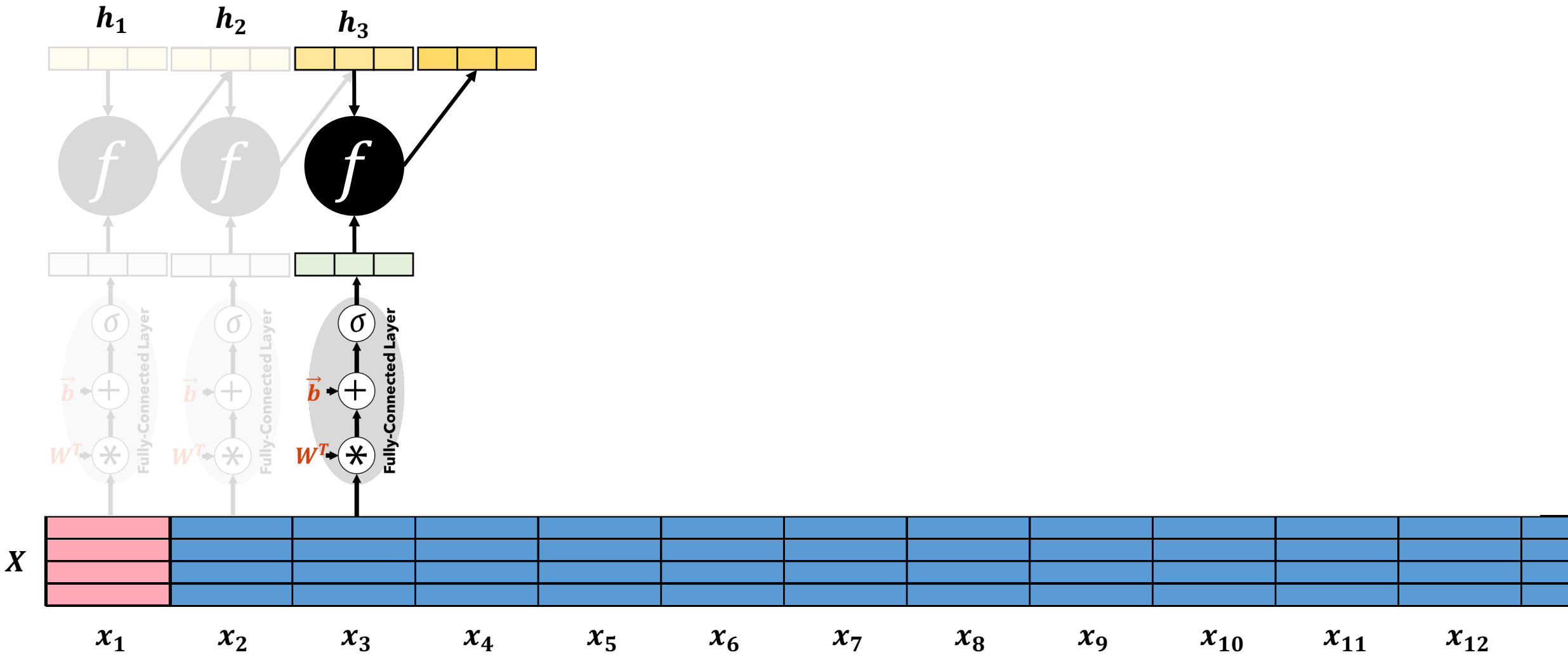
Variable Input → Fixed Output

Idea- Keep a memory around and update it with the same function over time.



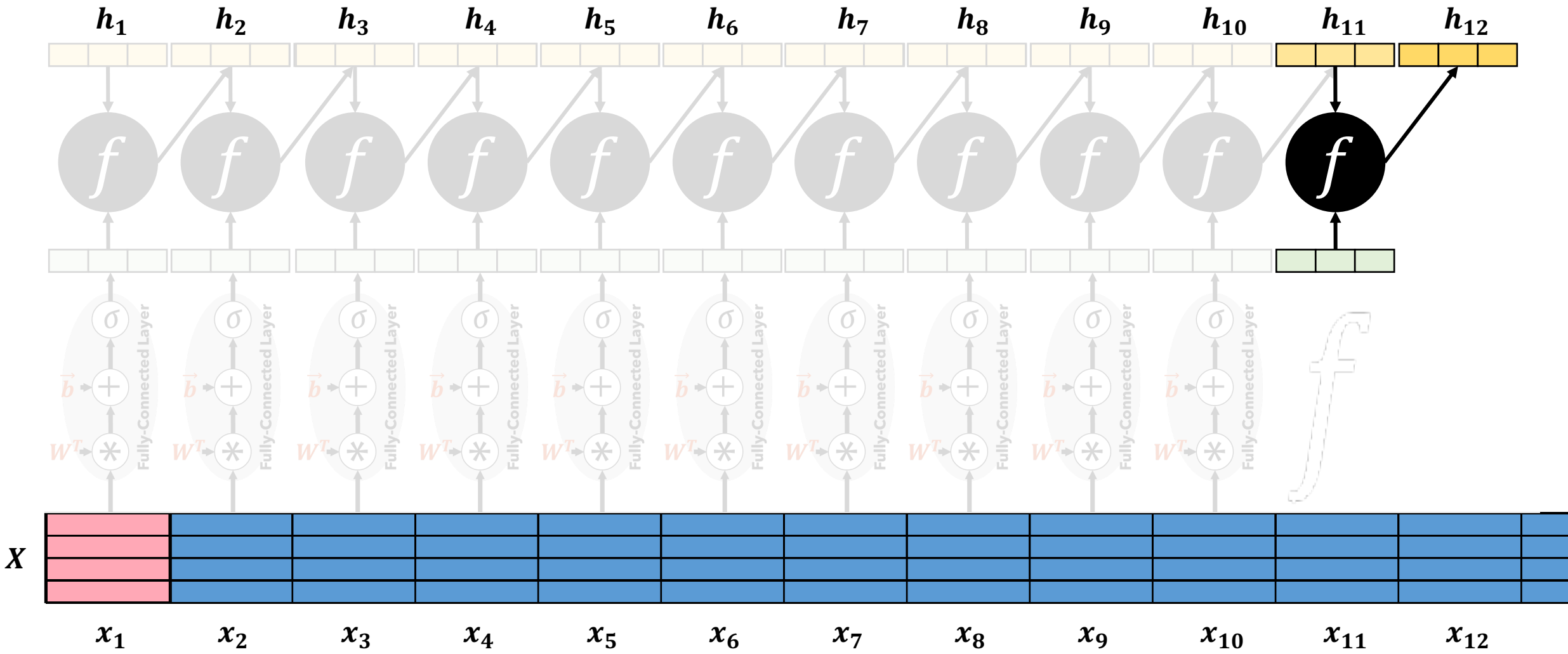
Variable Input → Fixed Output

Idea- Keep a memory around and update it with the same function over time.



Variable Input → Fixed Output

Idea- Keep a memory around and update it with the same function over time.



Recurrent Neural Networks

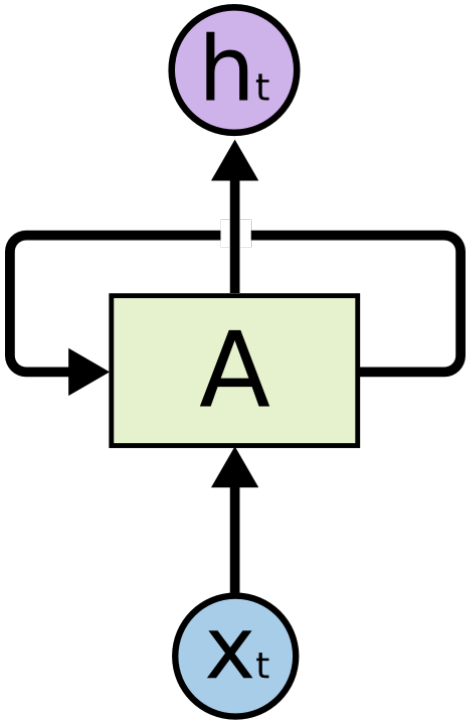


Image Credit: Christopher Olah

Recurrent Neural Networks

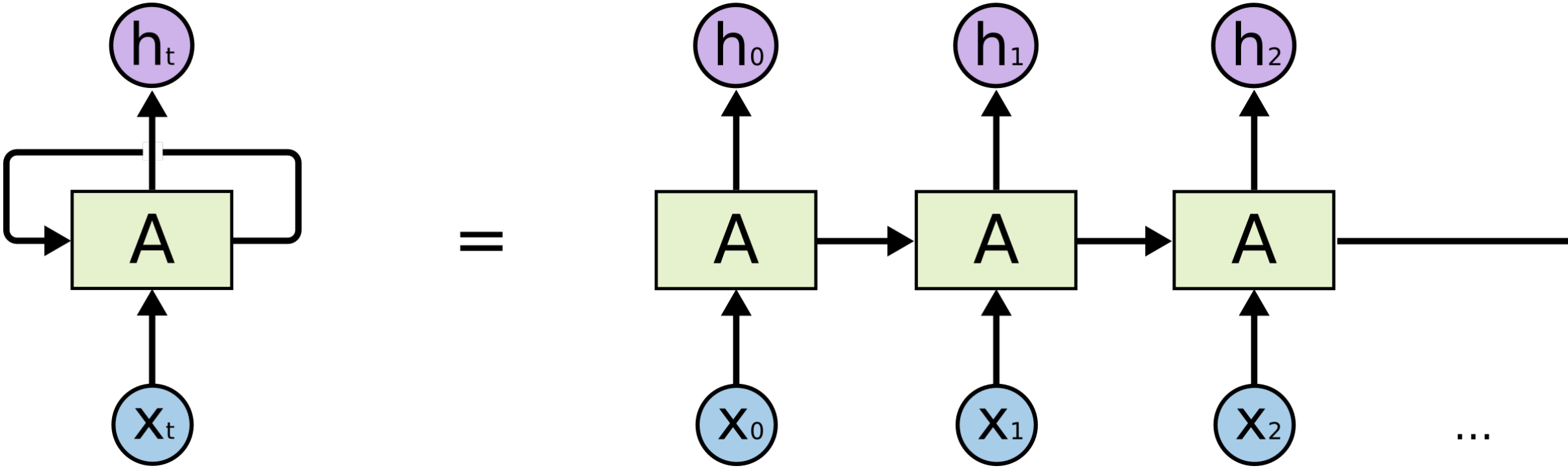


Image Credit: Christopher Olah

Today's Learning Objectives

Be able to answer:

- ~~What are the intuitions for some advanced structures in neural networks~~?

- What are decision trees?
- How are they learned?
  - What is entropy, conditional entropy, and information gain?
- How do they deal with continuous features?

- What do to about overfitting?

New Topic: Decision Trees

Question Break!

Classify this new point according to this decision tree:

No one saw me drop my food.
The dropped food was sticky.
The dropped food is a steak.
I am not a puma.

Should I eat my dropped food?

B Eat it!

D Don't eat it!

New Topic: Decision Trees

Algorithm Name: Decision Trees
Type: Supervised Classifiers / Regressor
Synonyms: Classification and Regression Trees (CART)

Key Idea: Recursively subdivide the input space.
Works for discrete or continuous inputs.

Algorithms for learning decision tress:

- ID3
- C4.5 C5

Related concept: random forests (multiple decision trees)

Starting to Think About Decision Trees (Discrete Inputs)

Consider the dataset below – we want to predict fuel efficiency.

If you could only use one attribute to make the decision, which would you choose?

# Cylinders	Weight	Acceleration	Fuel Efficiency
4	Low	High	Good
6	Medium	Medium	Bad
4	Medium	Medium	Bad
8	High	Low	Bad
4	Low	Low	Good

Starting to Think About Decision Trees (Discrete Inputs)

When we split on a discrete attribute, it shatters our dataset into subsets for each attribute.

Split on # Cylinders

# Cylinders	Weight	Acceleration	Fuel Efficiency
4	Low	High	Good
4	Medium	Medium	Bad
4	Low	Low	Good

# Cylinders	Weight	Acceleration	Fuel Efficiency
6	Medium	Medium	Bad

# Cylinders	Weight	Acceleration	Fuel Efficiency
8	High	Low	Bad

Split on Weight

# Cylinders	Weight	Acceleration	Fuel Efficiency
4	Low	High	Good
4	Low	Low	Good

# Cylinders	Weight	Acceleration	Fuel Efficiency
6	Medium	Medium	Bad
4	Medium	Medium	Bad

# Cylinders	Weight	Acceleration	Fuel Efficiency
8	High	Low	Bad

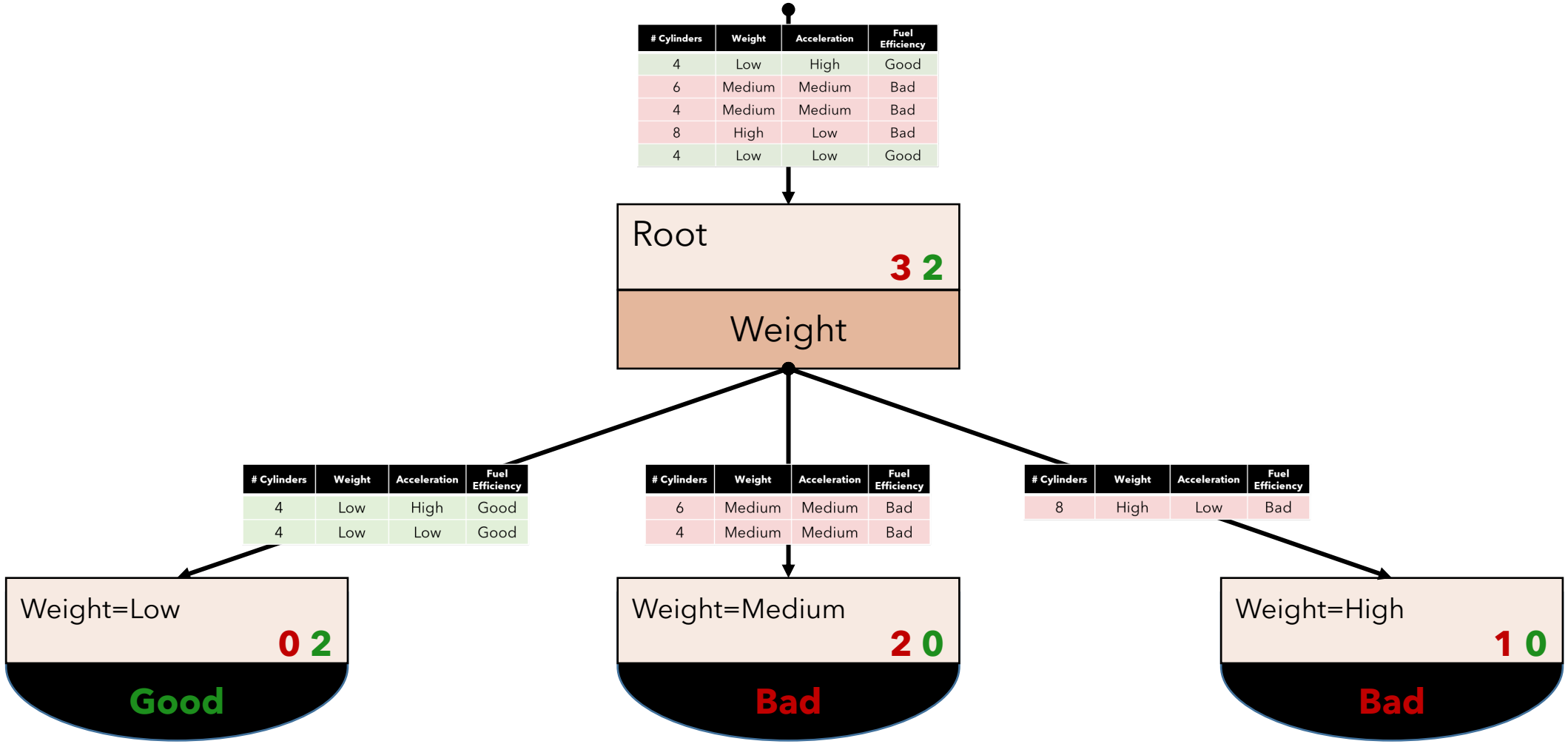
Split on Acceleration

# Cylinders	Weight	Acceleration	Fuel Efficiency
8	High	Low	Bad
4	Low	Low	Good

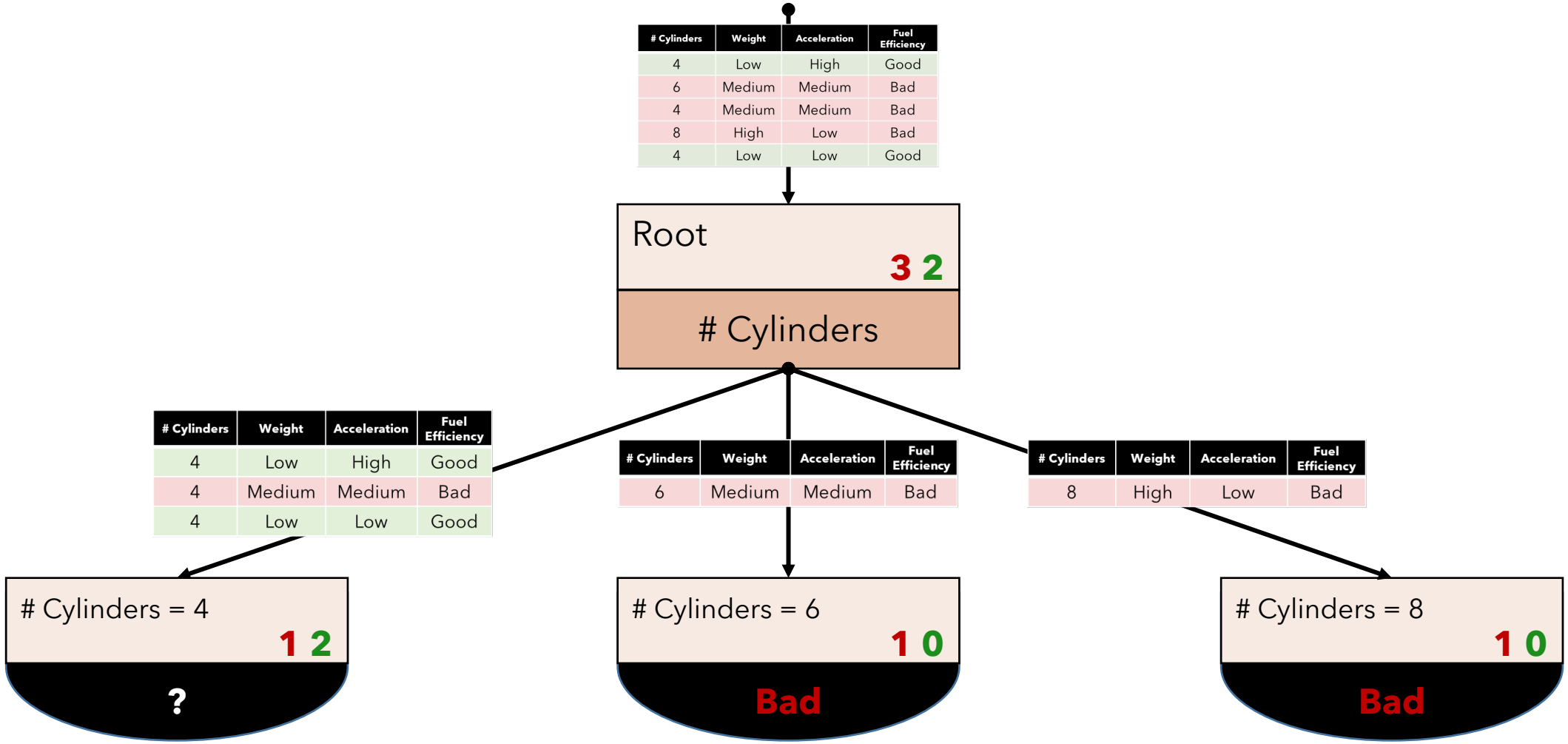
# Cylinders	Weight	Acceleration	Fuel Efficiency
6	Medium	Medium	Bad
4	Medium	Medium	Bad

# Cylinders	Weight	Acceleration	Fuel Efficiency
4	Low	High	Good

Starting to Think About Decision Trees (Discrete Inputs)



Starting to Think About Decision Trees (Discrete Inputs)



# Cylinders	Weight	Acceleration	Fuel Efficiency
4	Low	High	Good
6	Medium	Medium	Bad
4	Medium	Medium	Bad
8	High	Low	Bad
4	Low	Low	Good

# Cylinders	Weight	Acceleration	Fuel Efficiency
4	Low	High	Good
4	Medium	Medium	Bad
4	Low	Low	Good

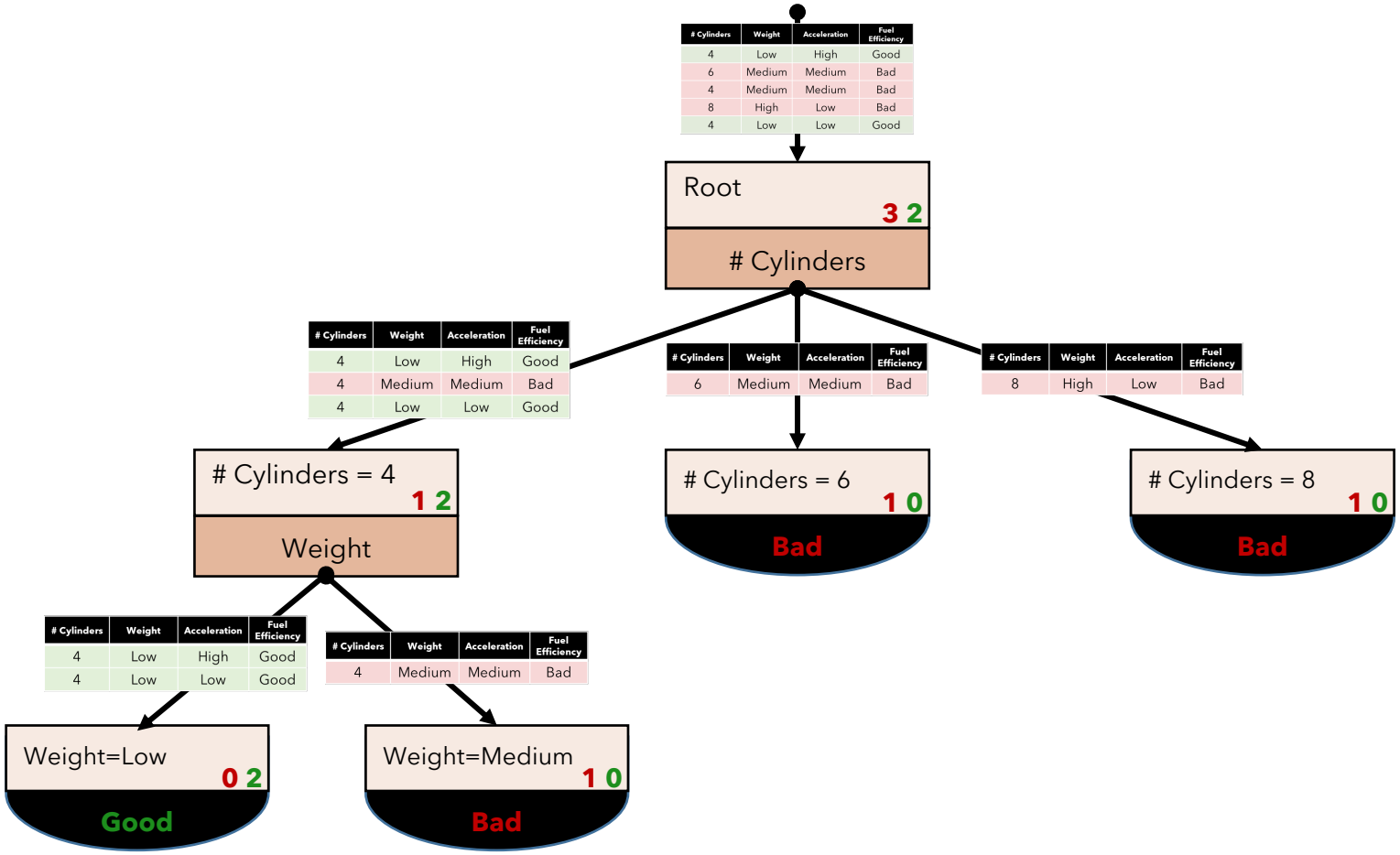
# Cylinders	Weight	Acceleration	Fuel Efficiency
6	Medium	Medium	Bad

# Cylinders	Weight	Acceleration	Fuel Efficiency
8	High	Low	Bad

One of the splits isn't pure, what now?

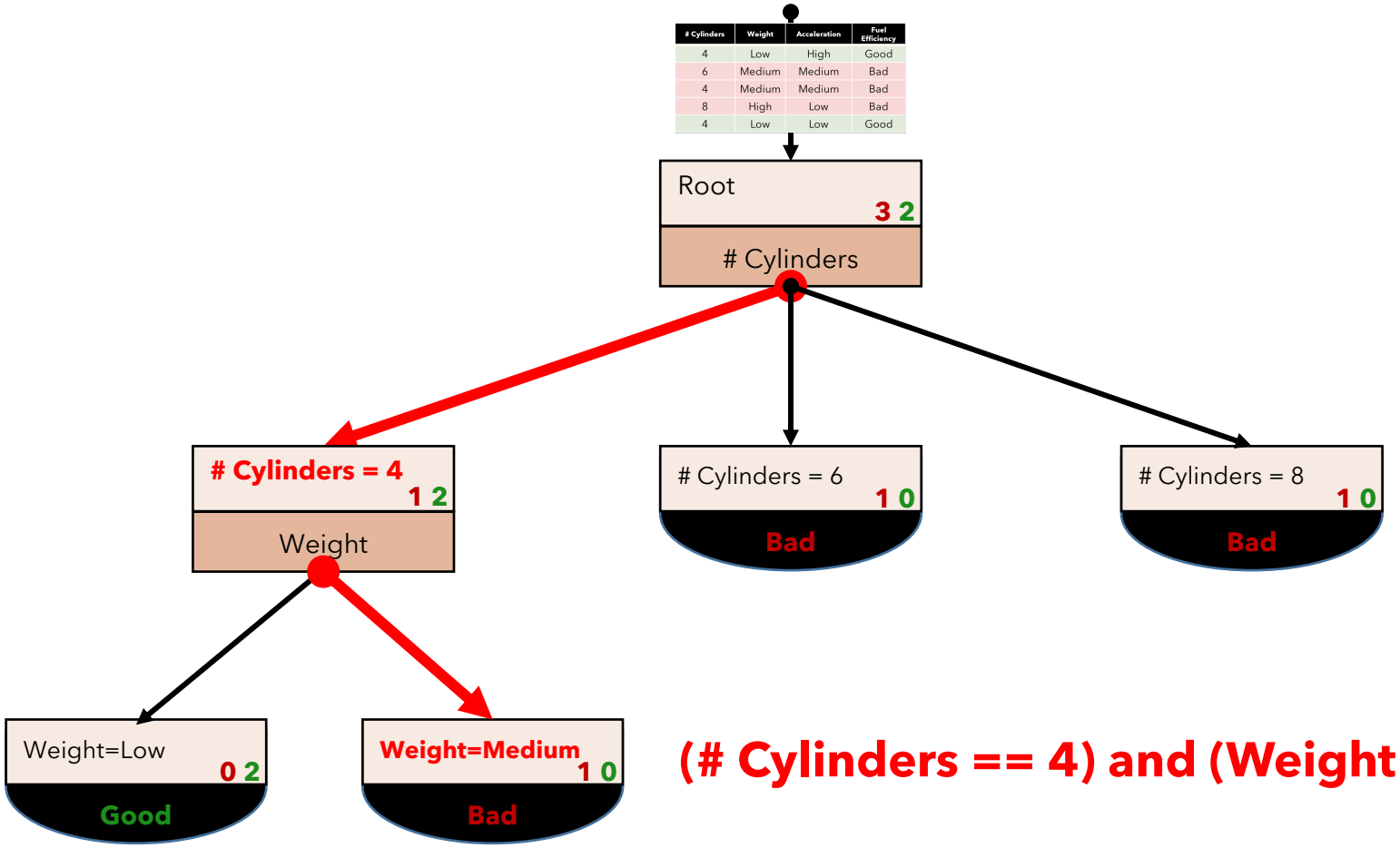
Starting to Think About Decision Trees (Discrete Inputs)

Split on another attribute and repeat this process..



Starting to Think About Decision Trees (Discrete Inputs)

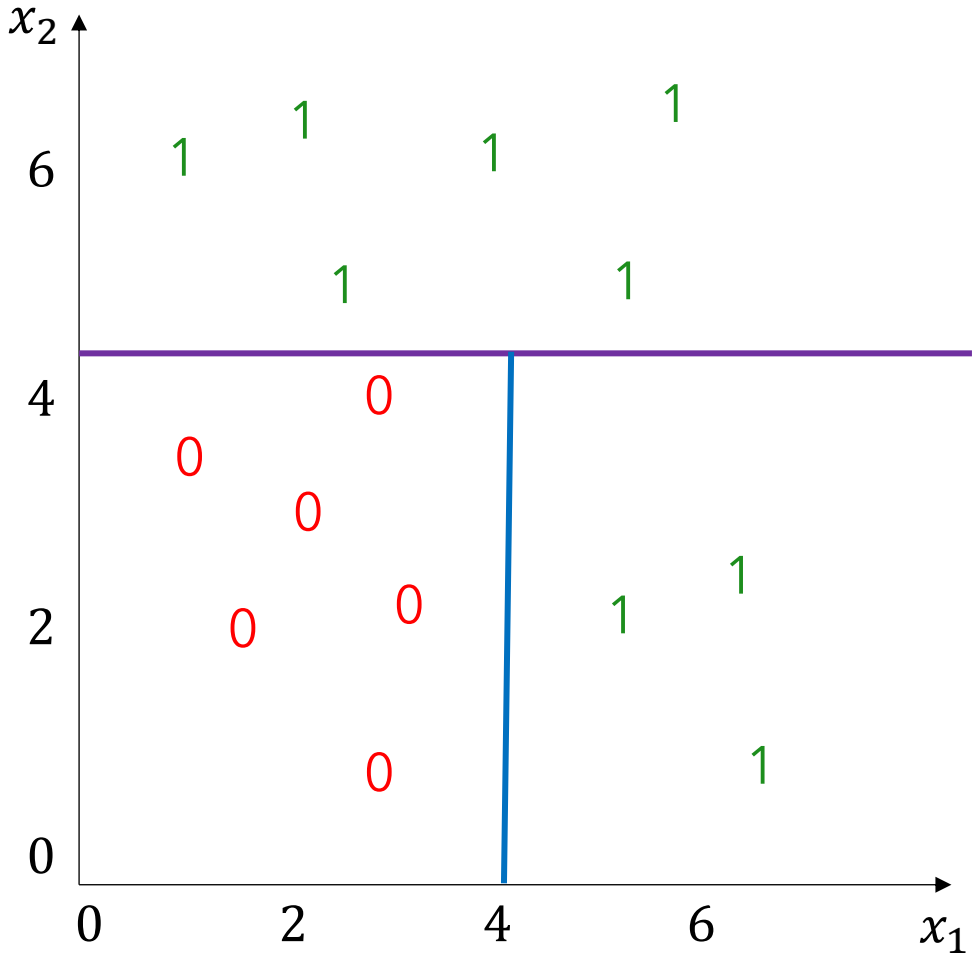
Notice that paths in the tree are logical formulas:

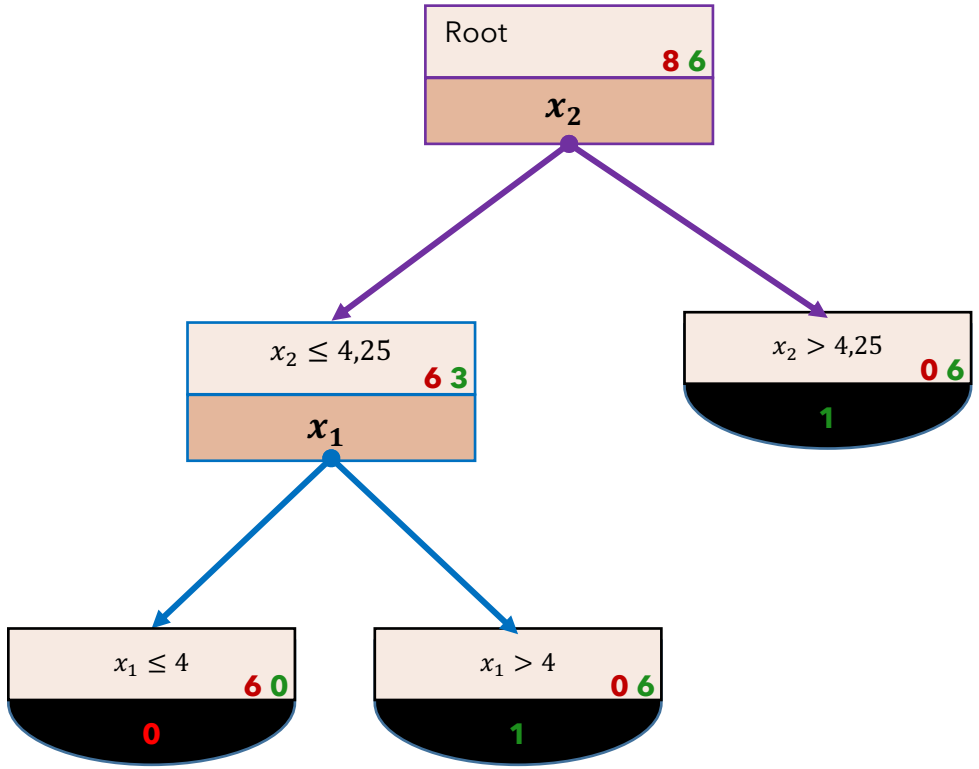


(# Cylinders == 4) and (Weight == Medium)

Starting to Think About Decision Trees (Continuous Inputs)

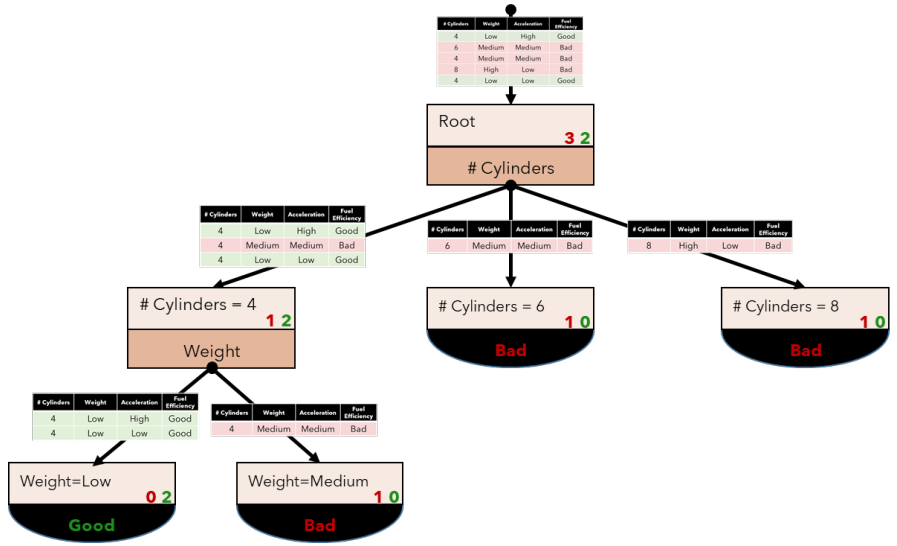
Decision trees divide the feature space into axis-parallel rectangles and label each rectangle with one class / value

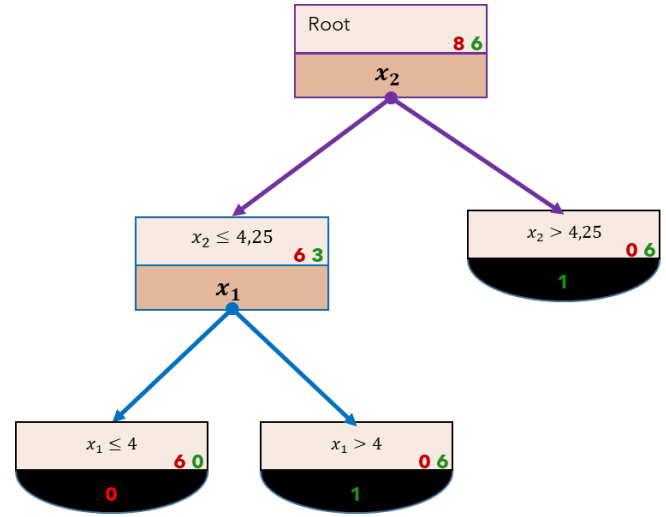




A decision tree is a tree structured model where:

- **Internal nodes** perform tests against an individual input feature value
- Each branch from an internal node represents a potential value or range of values of the tested attribute.
- Each **leaf node** predicts an output
  - Either class or continuous value





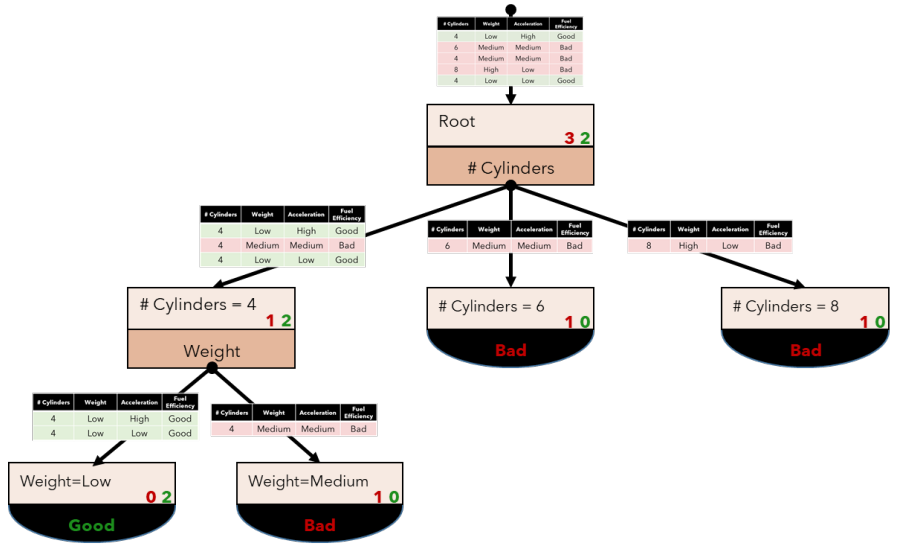
Decision Trees

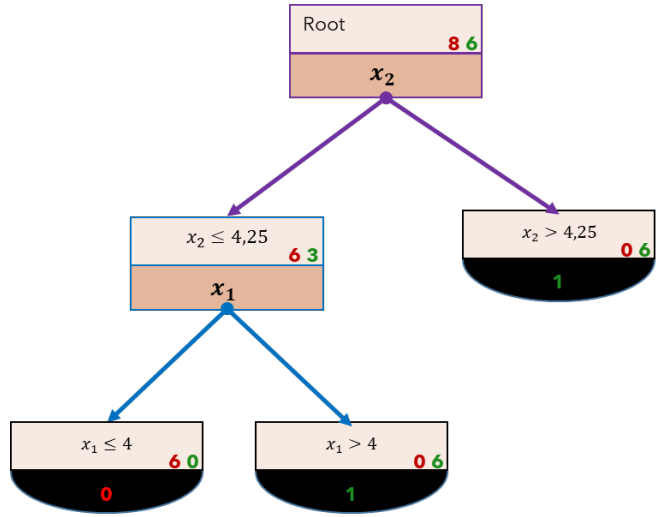
Decision trees have many appealing properties and are work-horses in many practical ML applications.

- Resulting models are human-interpretable (more or less)

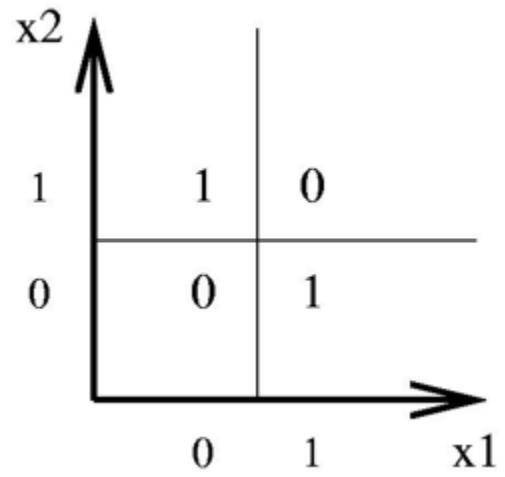
- Deals with mixed discrete / continuous inputs without needing careful input encoding
  - Recall our binary encoding for categorical inputs in kNN for HW1

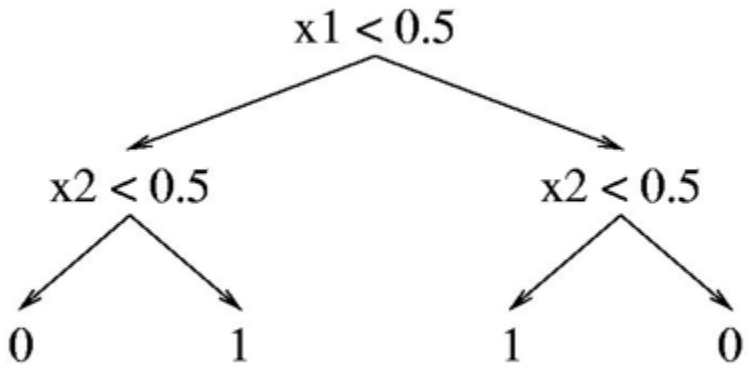
- Can represent arbitrarily complex functions as depth increases





Decision Trees



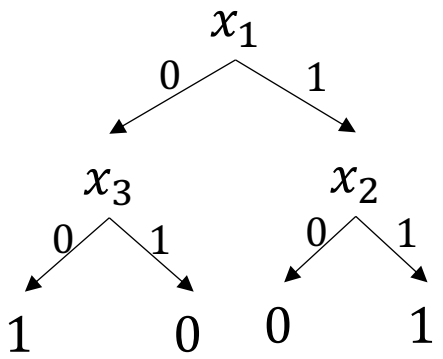


Decision trees can represent any arbitrary Boolean function.

- The tree will need to have exponentially many nodes in the worst case.

As the number of nodes (or depth) of the tree increases, the model can represent more complex functions (aka the hypothesis space grows):

- **Depth 1** ("decision stump") can represent any Boolean function of one feature.
- **Depth 2** Any Boolean function of two features and some Boolean functions of three features (e.g., *(x_1 and x_2) or (not x_1 and not x_3)*)
- Etc..



A Real Example

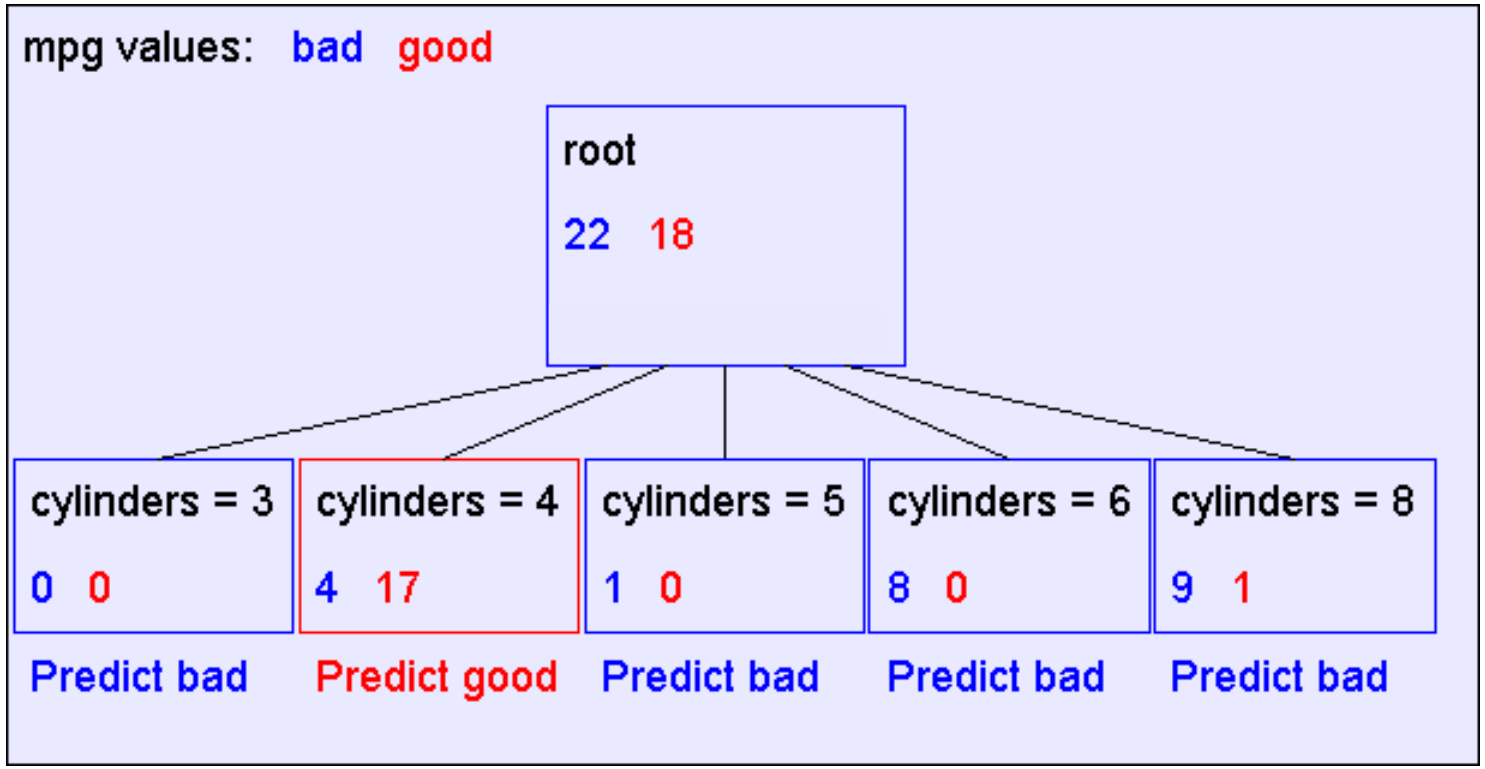
Dataset of car properties and mile per gallon as being good or bad. No continuous variables – some categorical, some ordinal. 40 records total.

mpg	cylinders	displacement	horsepower	weight	acceleration	modelyear	maker
good	4	low	low	low	high	75to78	asia
bad	6	medium	medium	medium	medium	70to74	america
bad	4	medium	medium	medium	low	75to78	europe
bad	8	high	high	high	low	70to74	america
bad	6	medium	medium	medium	medium	70to74	america
bad	4	low	medium	low	medium	70to74	asia
bad	4	low	medium	low	low	70to74	asia
bad	8	high	high	high	low	75to78	america
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
bad	8	high	high	high	low	70to74	america
good	8	high	medium	high	high	79to83	america
bad	8	high	high	high	low	75to78	america
good	4	low	low	low	low	79to83	america
bad	6	medium	medium	medium	high	75to78	america
good	4	medium	low	low	low	79to83	america
good	4	low	low	medium	high	79to83	america
bad	8	high	high	high	low	70to74	america
good	4	low	medium	low	medium	75to78	europe
bad	5	medium	medium	medium	medium	75to78	europe

https://archive.ics.uci.edu/ml/datasets/auto+mpg

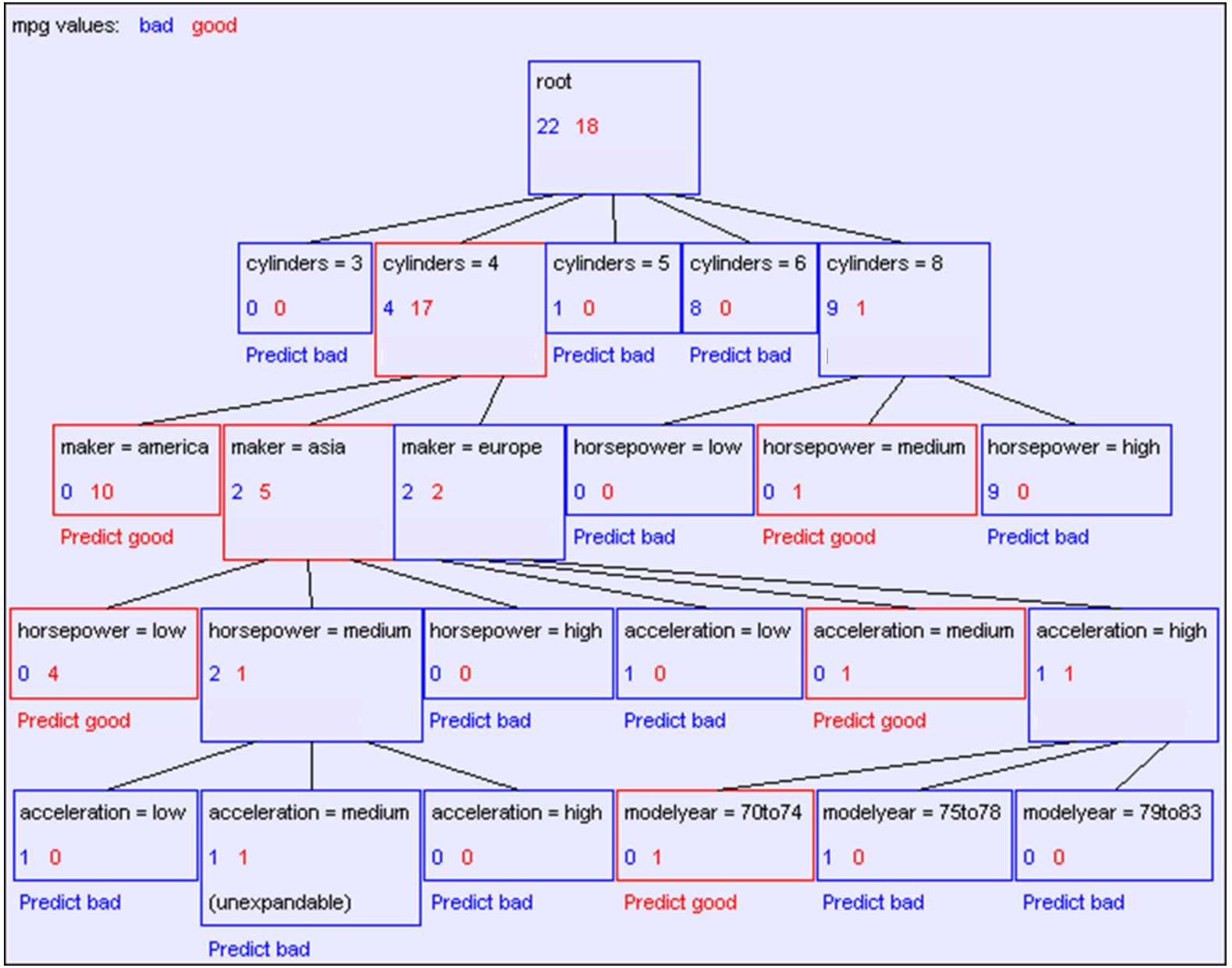
A Real Example

A Decision Stump – Splitting on number of cylinders



A Real Example

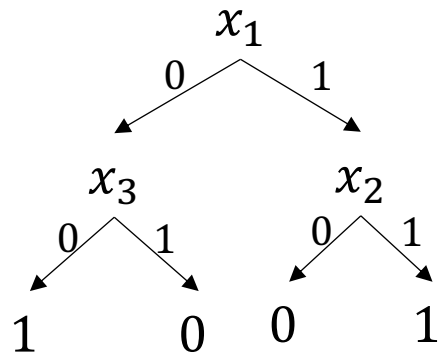
Full Tree

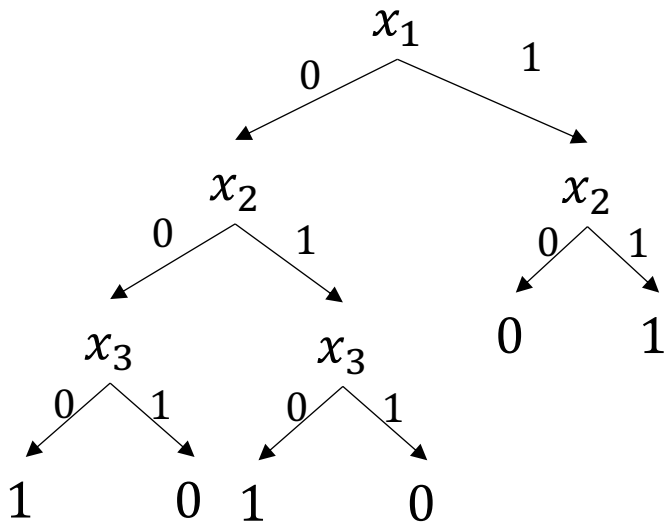


Example from Carlos Geustrin

A Few Notes

- Not all features/attributes need to appear in the tree
- A feature/attribute may appear in multiple branches
- On any given path, repeating discrete features is not useful
- Many trees represent the same concept / logical equation but not all trees will have the same size.





Example from Carlos Geustrin

How do we learn decision trees from data?

Easy Mode: find a decision tree that achieves minimum error on training data. Trivially achievable with large enough tree

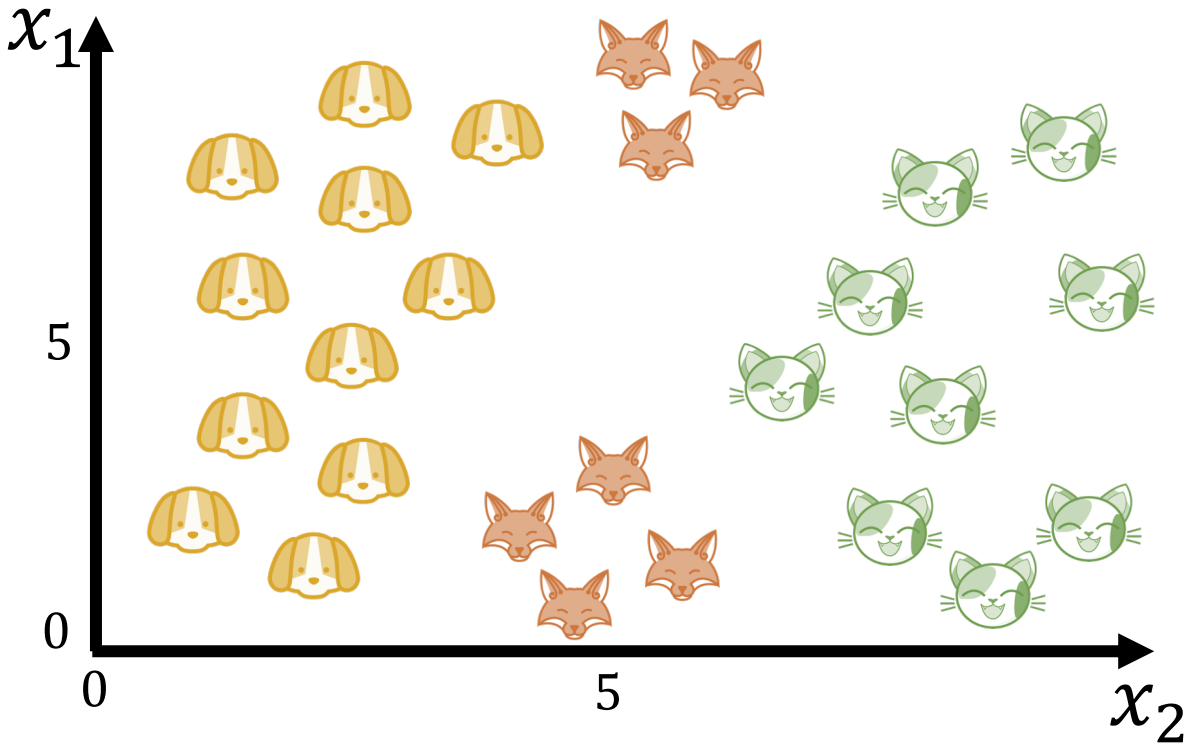
- Imagine a tree deep enough that each leaf contains only one example (or all examples sharing identical feature values)

NP-Hard Mode: find the smallest decision tree that achieves the minimum training error

- Why might we care about "smallest"?
  Small trees represent *simpler* functions → less likely to overfit.

Most popular methods build the tree greedily – can't guarantee finding smallest.

Let's Do It Intuitively/Manually First



Greedy Decision Tree Algorithm

Consider the simple recursive algorithm below:

BasicGreedyAlgorithm(dataset S):

Based on S, choose "best" test t to split the data

Split S into subset $S_1, \ldots, S_k$ for each unique outcome of t applied to S

For i in {1,…,k}:

Create child node c_i = BasicGreedyAlgorithm(S_i)

Greedy Decision Tree Algorithm

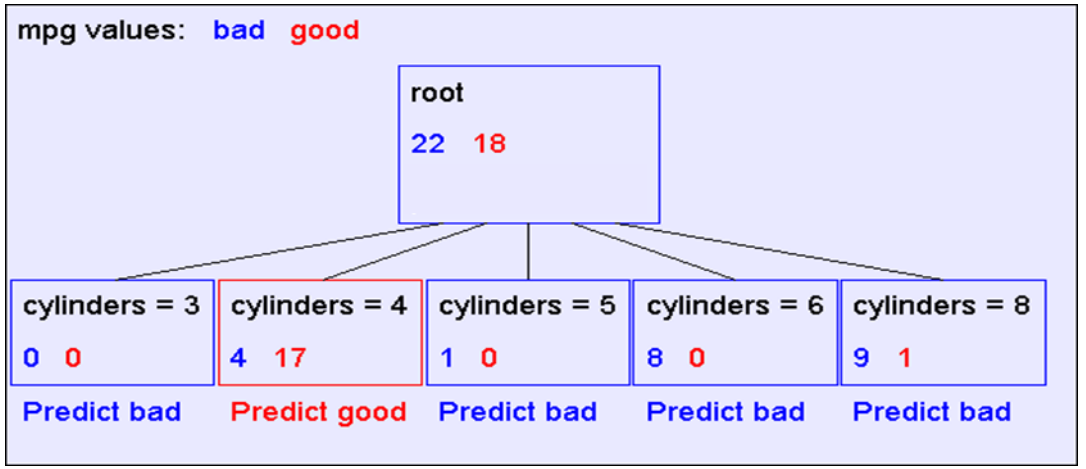
BasicGreedyAlgorithm(dataset S):

Based on S, choose "best" test t to split the data

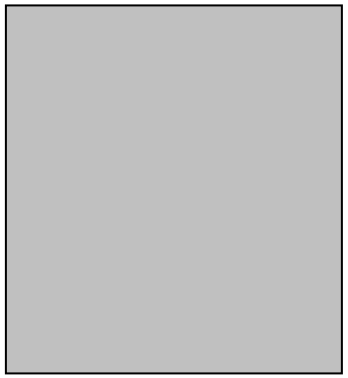
Split S into subset $S_1, \ldots, S_k$ for each unique outcome of t applied to S

For i in {1,…,k}:

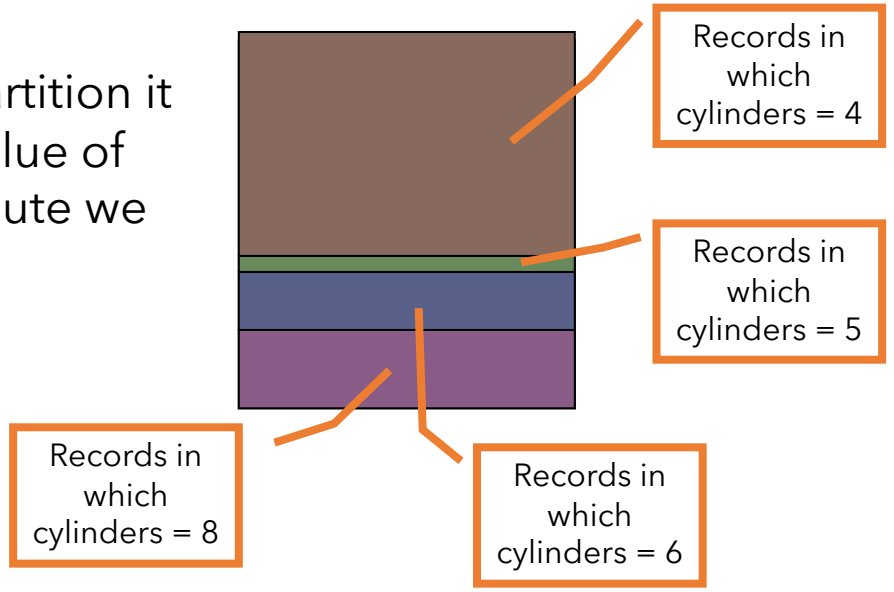
Create child node c_i = BasicGreedyAlgorithm(S_i)



Take the dataset...



…and partition it by the value of the attribute we split on



Greedy Decision Tree Algorithm

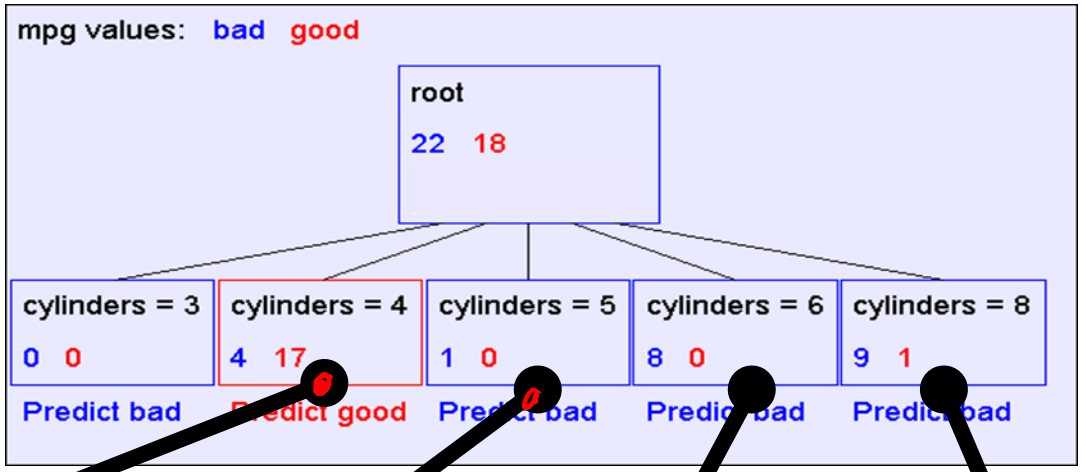
BasicGreedyAlgorithm(dataset S):

Based on S, choose "best" test t to split the data

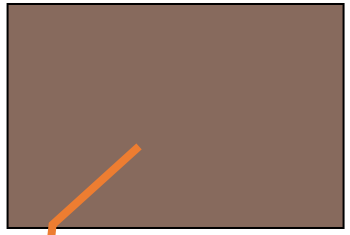
Split S into subset $S_1, \ldots, S_k$ for each unique outcome of t applied to S

For i in {1,…,k}:

Create child node c_i = BasicGreedyAlgorithm(S_i)



Build subtree with



Records in which cylinders = 4

Build subtree with

Records in which cylinders = 5

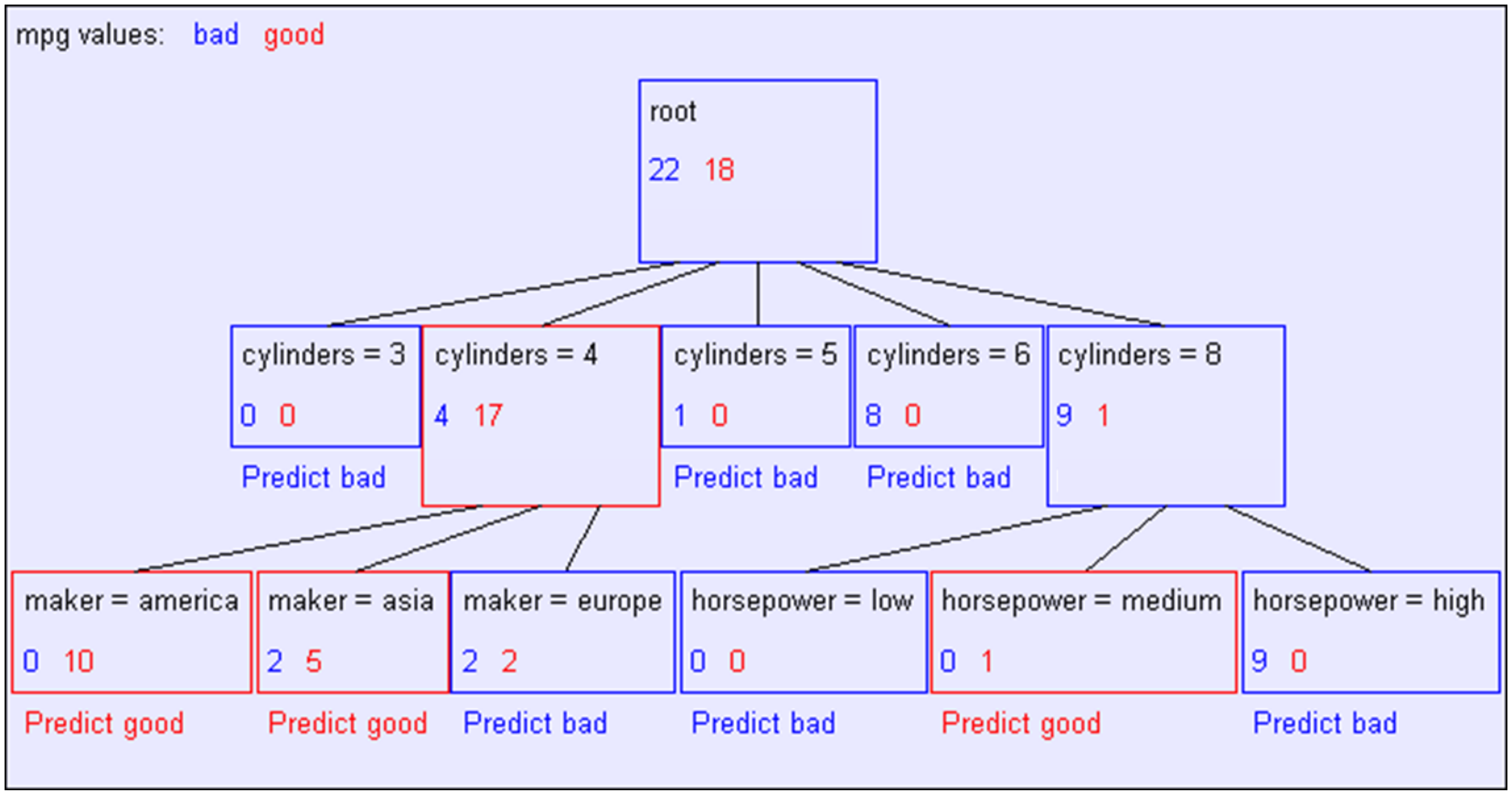
Build subtree with

Records in which cylinders = 6

Build subtree with

Records in which cylinders = 8

Greedy Decision Tree Algorithm



Example from Carlos Geustrin

Back to our recursive learning algorithm:

How do we know which is best?

BasicGreedyAlgorithm(dataset S):

Based on S, choose "best" test t to split the data

Split S into subset $S_1, \ldots, S_k$ for each unique outcome of t applied to S

For i in {1,…,k}:

Create child node c_i = BasicGreedyAlgorithm(S_i)

When do we stop? What are the base cases of this recursion?

How to choose tests / splits / attributes?

One idea: Choose whichever test / attribute that reduces uncertainty about the class labels in the resulting dataset splits.

# Cylinders	Weight	Acceleration	Fuel Efficiency
4	Low	High	Good
6	Medium	Medium	Bad
4	Medium	Medium	Bad
8	High	Low	Bad
4	Low	Low	Good

Split on # Cylinders

# Cylinders	Weight	Acceleration	Fuel Efficiency
4	Low	High	Good
4	Medium	Medium	Bad
4	Low	Low	Good

# Cylinders	Weight	Acceleration	Fuel Efficiency
6	Medium	Medium	Bad

# Cylinders	Weight	Acceleration	Fuel Efficiency
8	High	Low	Bad

Split on Weight

# Cylinders	Weight	Acceleration	Fuel Efficiency
4	Low	High	Good
4	Low	Low	Good

# Cylinders	Weight	Acceleration	Fuel Efficiency
6	Medium	Medium	Bad
4	Medium	Medium	Bad

# Cylinders	Weight	Acceleration	Fuel Efficiency
8	High	Low	Bad

Entropy of a Random Variable Y: More uncertainty, more entropy!

Discrete Distributions:

$$H(Y) = -\sum_{i=1}^{k} P(Y = y_i) \log_2 P(Y = y_i)$$

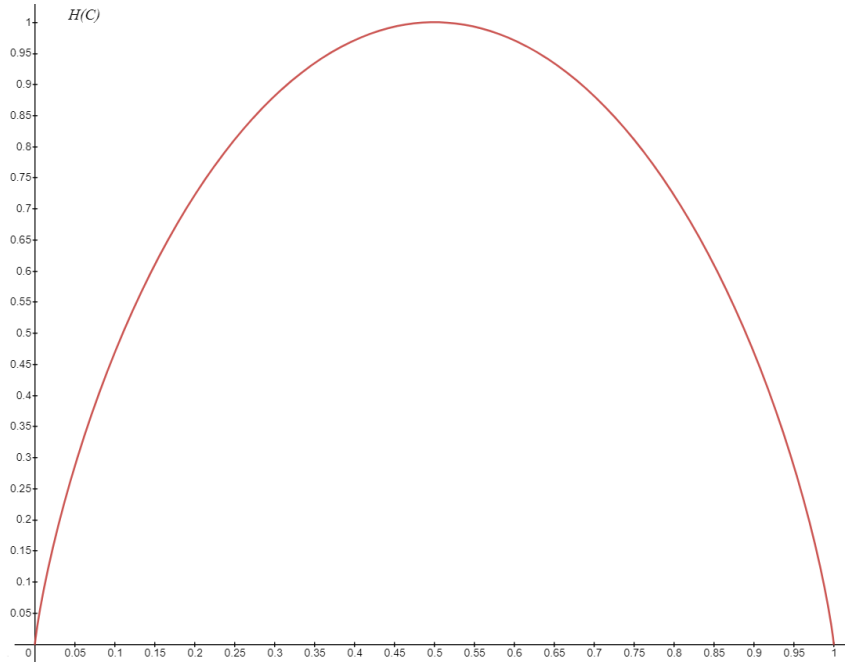
Continuous Distributions:

$$H(Y) = -\int_{y \in Y} P(Y = y) \log_2 P(Y = y)\, dy$$

For example, the entropy of the outcome of a coin flip C with probability of heads θ is:

$$H(C) = -\sum_{i=1}^{k} P(Y = y_i) \log_2 P(Y = y_i)$$

$$= -\theta \log_2 \theta - (1-\theta) \log_2 (1-\theta)$$



The flip is most uncertain at $\theta = 0.5$ and least at either $\theta = 0$ or $\theta = 1$ where the outcome is fixed.

Measuring Uncertainty

$$H(Y) = -\sum_{i=1}^{k} P(Y = y_i) \log_2 P(Y = y_i)$$

What about the entropy of the class labels in a dataset?

# Cylinders	Weight	Acceleration	Fuel Efficiency
4	Low	High	Good
6	Medium	Medium	Bad
4	Medium	Medium	Bad
8	High	Low	Bad
4	Low	Low	Good

$$P(Y = Good) = 2/5$$

$$P(Y = Bad) = 3/5$$

$$H(Y) = -\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5} \approx 0.971$$

How to choose tests / splits / attributes?

One idea: Choose whichever test / attribute that reduces uncertainty about the class labels in the resulting dataset splits.

# Cylinders	Weight	Acceleration	Fuel Efficiency
4	Low	High	Good
6	Medium	Medium	Bad
4	Medium	Medium	Bad
8	High	Low	Bad
4	Low	Low	Good

$H(Y) \approx 0.971$

Split on # Cylinders

# Cylinders	Weight	Acceleration	Fuel Efficiency
4	Low	High	Good
4	Medium	Medium	Bad
4	Low	Low	Good

# Cylinders	Weight	Acceleration	Fuel Efficiency
6	Medium	Medium	Bad

# Cylinders	Weight	Acceleration	Fuel Efficiency
8	High	Low	Bad

$H(Y|Cylinders) \approx ?$

Split on Weight

# Cylinders	Weight	Acceleration	Fuel Efficiency
4	Low	High	Good
4	Low	Low	Good

# Cylinders	Weight	Acceleration	Fuel Efficiency
6	Medium	Medium	Bad
4	Medium	Medium	Bad

# Cylinders	Weight	Acceleration	Fuel Efficiency
8	High	Low	Bad

$H(Y|Weight) \approx ?$

Conditional Entropy of a Random Variable Y Given X:

Entropy of Y when only considering records were $X = x_j$

$$H(Y|X) = -\sum_{j}^{v} P(X = x_j) \sum_{i=1}^{k} P\left(Y = y_i | X = x_j\right) \log_2 P(Y = y_i | X = x_j)$$

Weighted by probability of $X = x_j$

Continuous distribution version replaces sums with integrals as is typical.

What is the conditional entropy of the class labels in the following split?

Split on # Cylinders

# Cylinders	Weight	Acceleration	Fuel Efficiency
4	Low	High	Good
4	Medium	Medium	Bad
4	Low	Low	Good

# Cylinders	Weight	Acceleration	Fuel Efficiency
6	Medium	Medium	Bad

# Cylinders	Weight	Acceleration	Fuel Efficiency
8	High	Low	Bad

$$P(Y = Good \mid \#Cylinders = 4) = 2/3$$
$$P(Y = Bad \mid \#Cylinders = 4) = 1/3$$
$$P(\#Cylinders = 4) = 3/5$$
$$P(Y = Good \mid \#Cylinders = 6) = 0$$
$$P(Y = Bad \mid \#Cylinders = 6) = 1$$
$$P(\#Cylinders = 6) = 1/5$$
$$P(Y = Good \mid \#Cylinders = 8) = 0$$
$$P(Y = Bad \mid \#Cylinders = 8) = 1$$
$$P(\#Cylinders = 8) = 1/5$$

$$H(Y|X) = -\frac{3}{5}\left(\frac{2}{3}\log_2\frac{2}{3} + \frac{1}{3}\log_2\frac{1}{3}\right) - \frac{1}{5}\left(\frac{0}{1}\log_2\frac{0}{1} + 1\log_2 1\right) - \frac{1}{5}\left(\frac{0}{1}\log_2\frac{0}{1} + 1\log_2 1\right)$$

$$\approx 0.551$$

Question Break!

Split on Weight

# Cylinders	Weight	Acceleration	Fuel Efficiency
4	Low	High	Good
4	Low	Low	Good

# Cylinders	Weight	Acceleration	Fuel Efficiency
6	Medium	Medium	Bad
4	Medium	Medium	Bad

# Cylinders	Weight	Acceleration	Fuel Efficiency
8	High	Low	Bad

$$H(Y|Weight) \approx ?$$

What is the conditional entropy of this data split?

A 1

B 0

C 0.579

D 0.286

How to choose tests / splits / attributes?

One idea: Choose whichever test / attribute that reduces uncertainty about the class labels in the resulting dataset splits.

# Cylinders	Weight	Acceleration	Fuel Efficiency
4	Low	High	Good
6	Medium	Medium	Bad
4	Medium	Medium	Bad
8	High	Low	Bad
4	Low	Low	Good

$$H(Y) \approx 0.971$$

Split on # Cylinders

# Cylinders	Weight	Acceleration	Fuel Efficiency
4	Low	High	Good
4	Medium	Medium	Bad
4	Low	Low	Good

# Cylinders	Weight	Acceleration	Fuel Efficiency
6	Medium	Medium	Bad

# Cylinders	Weight	Acceleration	Fuel Efficiency
8	High	Low	Bad

$$H(Y|Cylinders) \approx 0.551$$

Split on Weight

# Cylinders	Weight	Acceleration	Fuel Efficiency
4	Low	High	Good
4	Low	Low	Good

# Cylinders	Weight	Acceleration	Fuel Efficiency
6	Medium	Medium	Bad
4	Medium	Medium	Bad

# Cylinders	Weight	Acceleration	Fuel Efficiency
8	High	Low	Bad

$$H(Y|Weight) \approx 0$$

How to choose tests / splits / attributes?

Information Gain: $\mathrm{IG}(X) = H(Y) - H(Y|X)$

Entropy of class label before splitting on an attribute minus the conditional entropy after the split.

# Cylinders	Weight	Acceleration	Fuel Efficiency
4	Low	High	Good
6	Medium	Medium	Bad
4	Medium	Medium	Bad
8	High	Low	Bad
4	Low	Low	Good

$H(Y) \approx 0.971$

Split on # Cylinders

# Cylinders	Weight	Acceleration	Fuel Efficiency
4	Low	High	Good
4	Medium	Medium	Bad
4	Low	Low	Good

# Cylinders	Weight	Acceleration	Fuel Efficiency
6	Medium	Medium	Bad

# Cylinders	Weight	Acceleration	Fuel Efficiency
8	High	Low	Bad

$H(Y|Cylinders) \approx 0.551$

Split on Weight

# Cylinders	Weight	Acceleration	Fuel Efficiency
4	Low	High	Good
4	Low	Low	Good

# Cylinders	Weight	Acceleration	Fuel Efficiency
6	Medium	Medium	Bad
4	Medium	Medium	Bad

# Cylinders	Weight	Acceleration	Fuel Efficiency
8	High	Low	Bad

$H(Y|Weight) \approx 0$

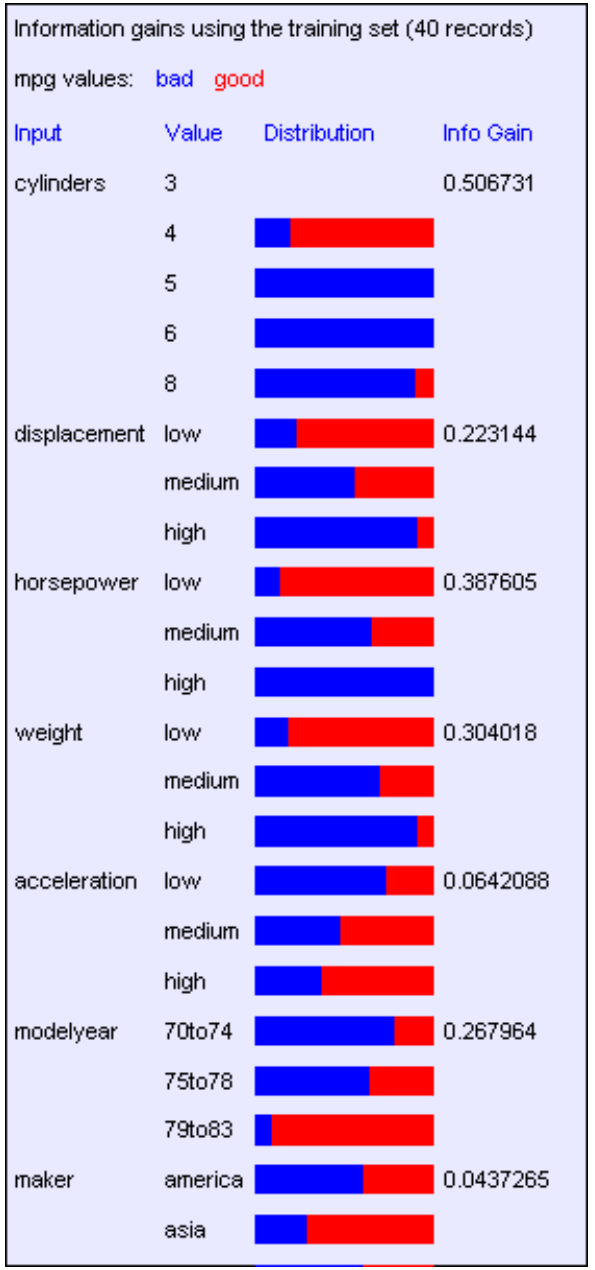
$$\mathrm{IG}(\#\mathrm{Cylinders}) = H(Y) - H(Y|\#\mathrm{Cylinders}) = 0.971 - 0.551 = 0.42$$
$$\mathrm{IG}(\mathrm{Weight}) = H(Y) - H(Y|\mathrm{Weight}) = 0.971 - 0 = 0.971$$

We reduce uncertainty *more* by choosing to split on the Weight attribute.

Information Gain for the every first split

mpg	cylinders	displacement	horsepower	weight	acceleration	modelyear	maker
good	4	low	low	low	high	75to78	asia
bad	6	medium	medium	medium	medium	70to74	america
bad	4	medium	medium	medium	low	75to78	europe
bad	8	high	high	high	low	70to74	america
bad	6	medium	medium	medium	medium	70to74	america
bad	4	low	medium	low	medium	70to74	asia
bad	4	low	medium	low	low	70to74	asia
bad	8	high	high	high	low	75to78	america
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
bad	8	high	high	high	low	70to74	america
good	8	high	medium	high	high	79to83	america
bad	8	high	high	high	low	75to78	america
good	4	low	low	low	low	79to83	america
bad	6	medium	medium	medium	high	75to78	america
good	4	medium	low	low	low	79to83	america
good	4	low	low	medium	high	79to83	america
bad	8	high	high	high	low	70to74	america
good	4	low	medium	low	medium	75to78	europe
bad	5	medium	medium	medium	medium	75to78	europe

https://archive.ics.uci.edu/ml/datasets/auto+mpg



Greedy Decision Tree Algorithm

Back to our recursive learning algorithm:

BasicGreedyAlgorithm(dataset S):

Based on S, choose "best" test t to split the data:

→ Evaluate the information gain of possible splits and pick the largest

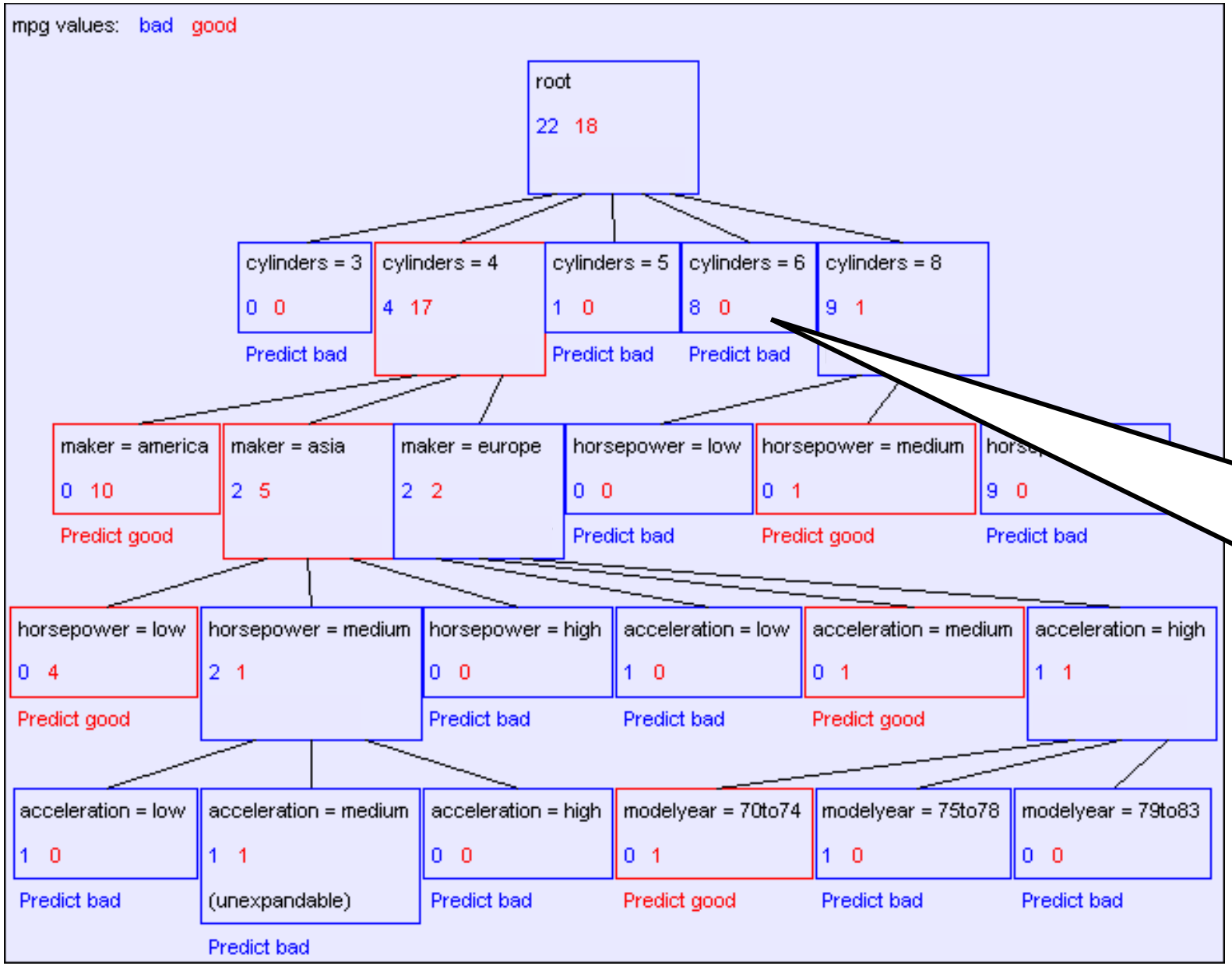
Split S into subset $S_1, \ldots, S_k$ for each unique outcome of t applied to S

For i in {1,…,k}:

Create child node c_i = BasicGreedyAlgorithm(S_i)

When do we stop? What are the base cases of this recursion?

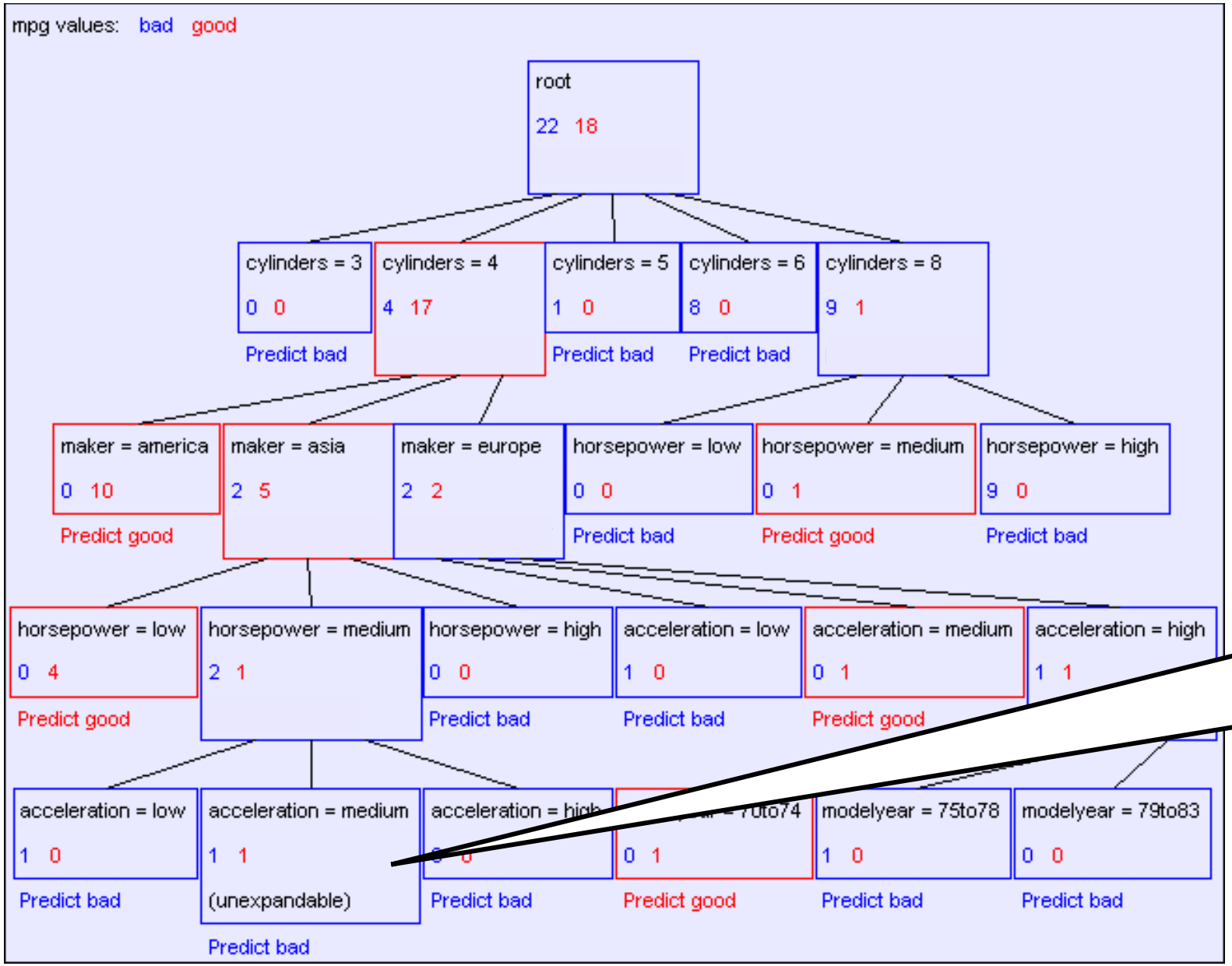
When do we stop? **Base Case 1**



Don't split a node if all matching records have the same output value

Example from Carlos Geustrin

When do we stop? **Base Case 2**

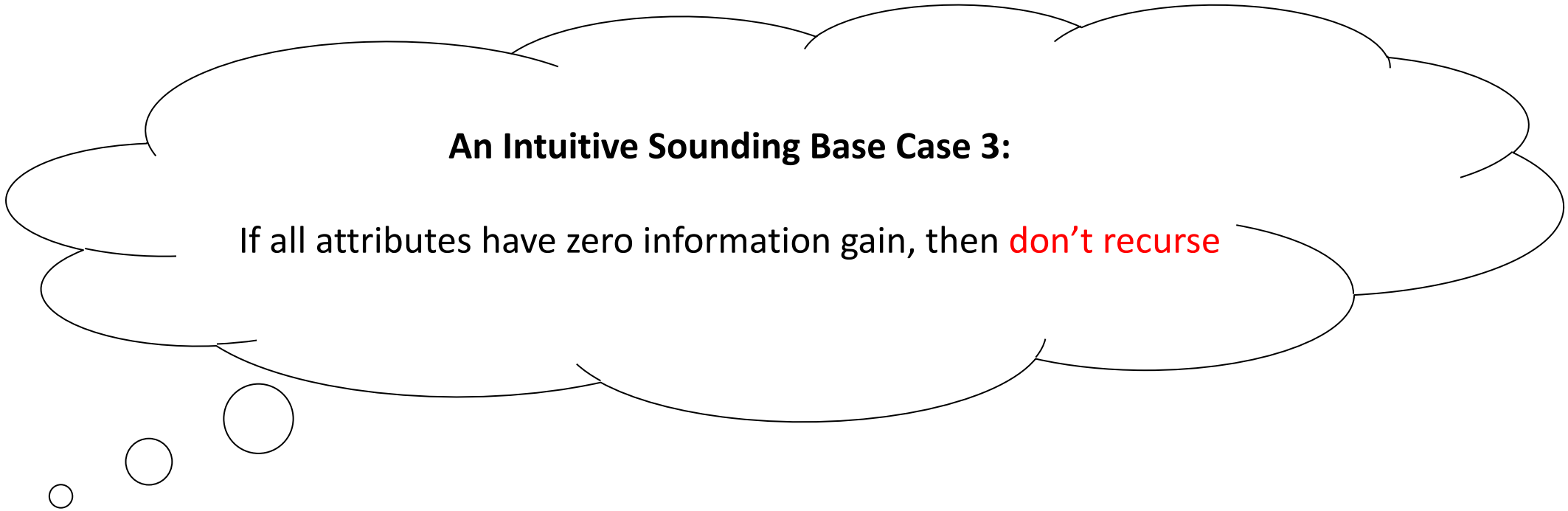


Don't split if none of the attributes can divide the examples

Example from Carlos Geustrin

When do we stop?

- **Base Case One:** If all records in current data subset have the same output/class then don't recurse because no further improvement could be made
- **Base Case Two:** If all records have exactly the same set of input features/attributes then don't recurse because nothing will split them



The Problem With Stopping At 0 Information Gain

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

Suppose that **Y = A XOR B**

What happens if we use the proposed Base Case 3?

Split on A

A	B	Y
0	0	0
0	1	1

A	B	Y
1	0	1
1	1	0

Split on B

A	B	Y
0	0	0
1	0	1

A	B	Y
0	1	1
1	1	0

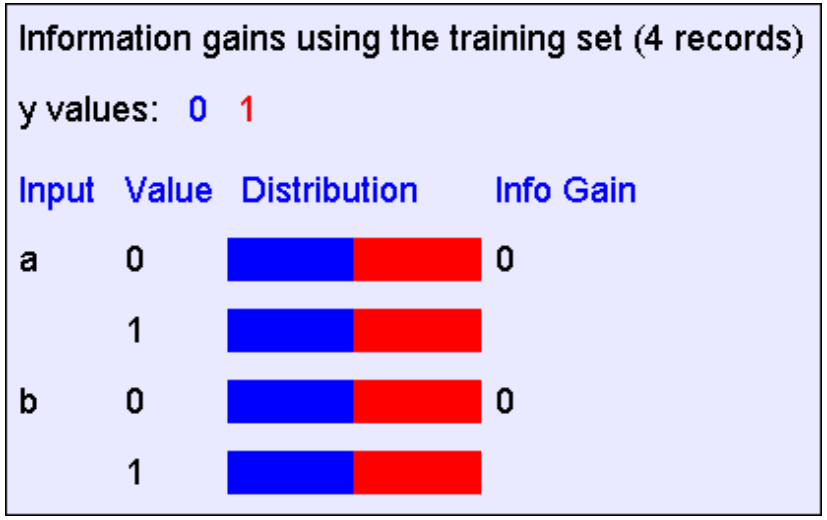
The Problem With Stopping At 0 Information Gain

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

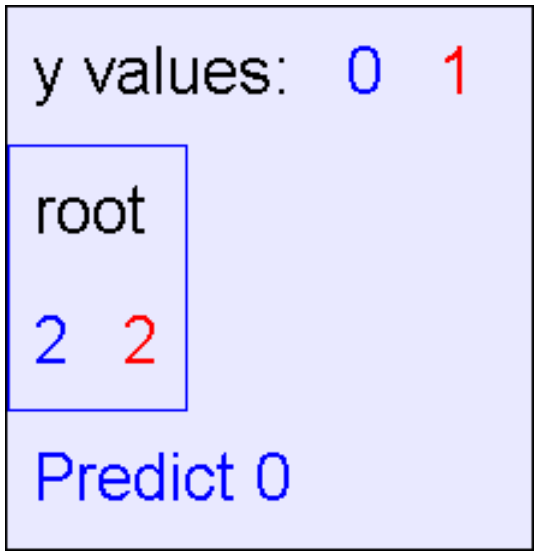
Suppose that **Y = A XOR B**

What happens if we use the proposed Base Case 3?

The information gains:



The resulting decision tree:



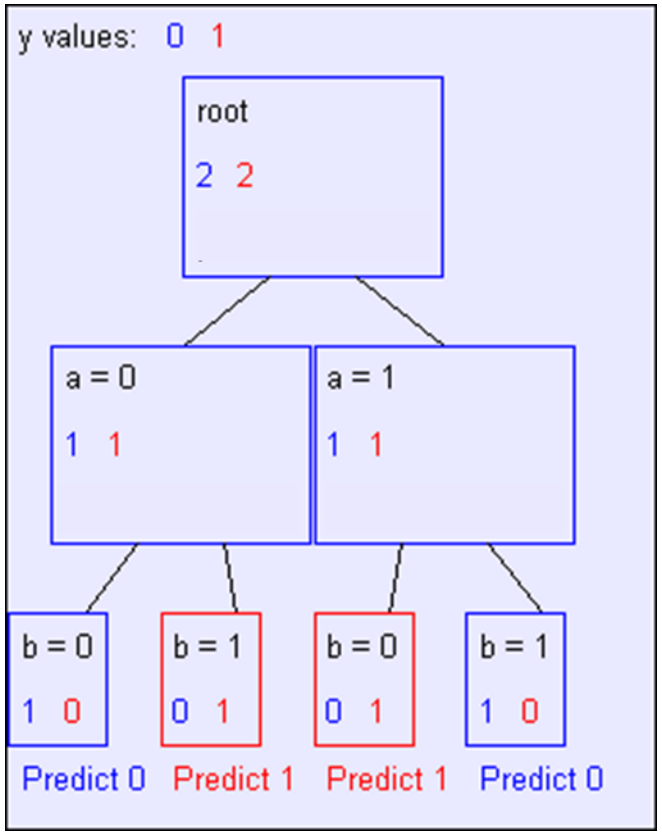
Example from Carlos Geustrin

The Problem With Stopping At 0 Information Gain

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

Suppose that **Y = A XOR B**

If we ignore the proposed (bad idea) base case:



Example from Carlos Geustrin

Back to our recursive learning algorithm:

BasicGreedyAlgorithm(dataset S):

If S all have same label or all same features:

return Leaf(majorityLabel(S))

Based on S, choose "best" test t to split the data:

Evaluate the information gain of possible splits and pick the largest

Split S into subset $S_1, \ldots, S_k$ for each unique outcome of t applied to S

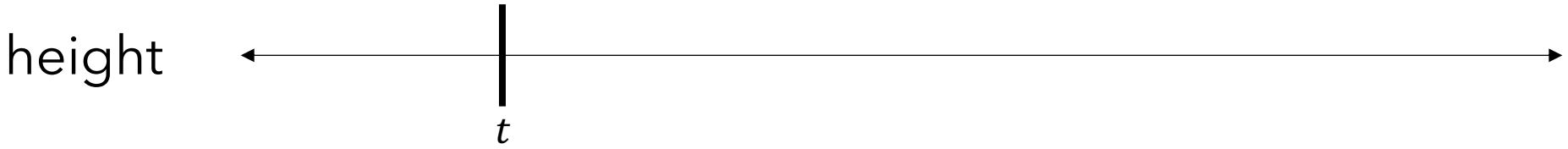
For i in {1,…,k}:

Create child node c_i = BasicGreedyAlgorithm(S_i)

Dealing with Continuous Features

Want to make splits of the form $X_i \geq t$ for some threshold t:

- Now need to pick the best attribute (X_i) and a threshold t
- There are infinitely many possible such splits… can't compute information gain for all of them!

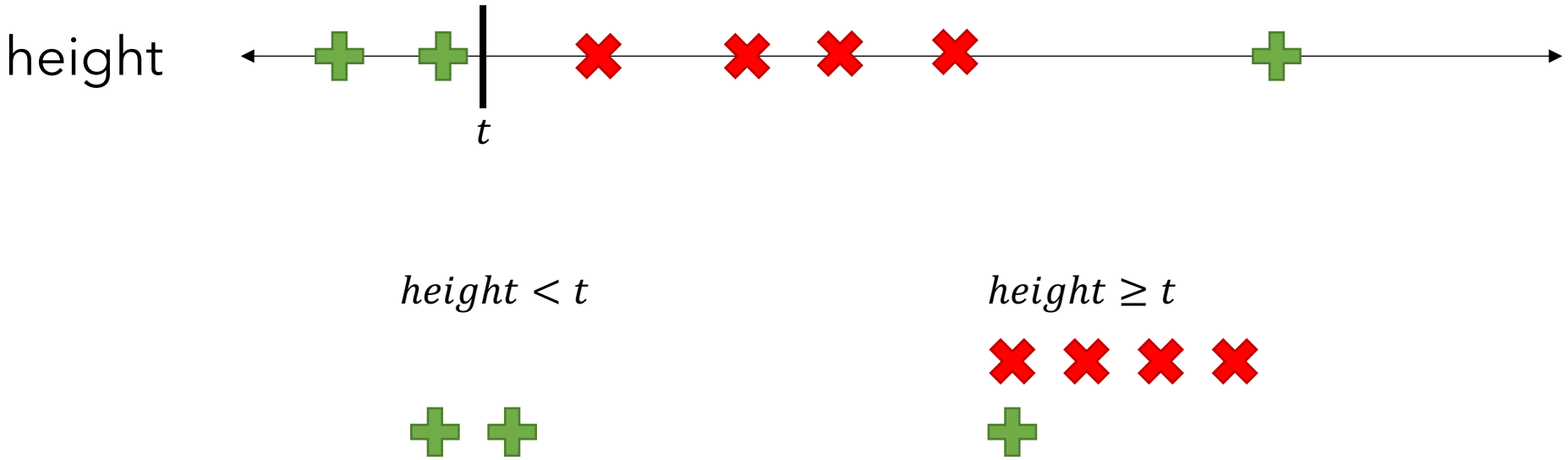


- Any ideas?

Dealing with Continuous Features

Want to make splits of the form $X_i \geq t$ for some threshold t:

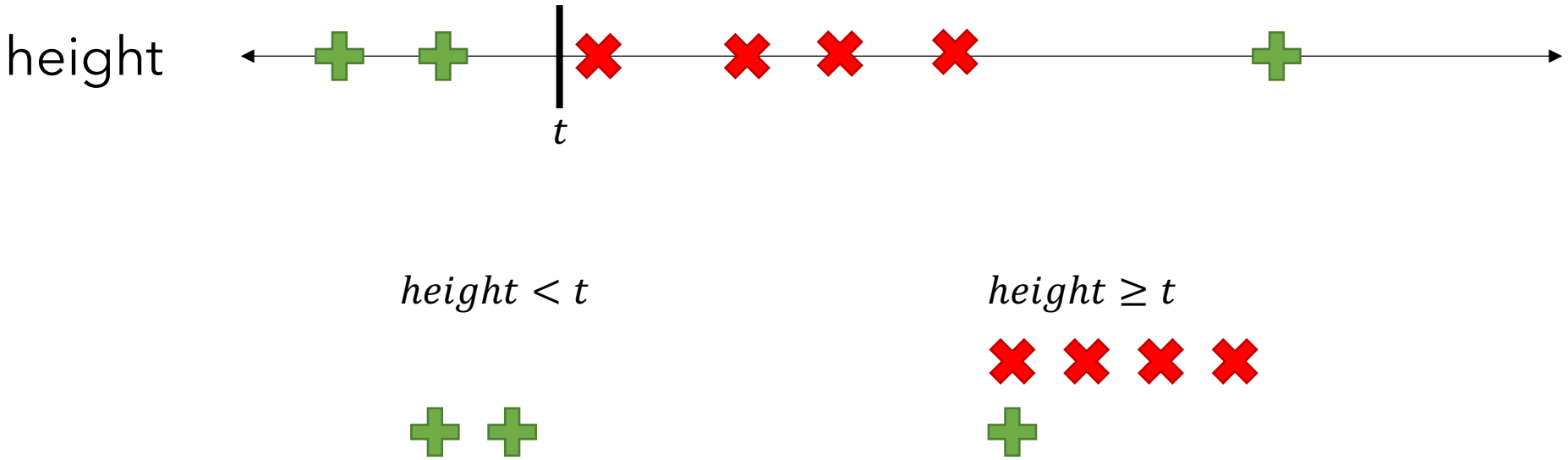
- There are infinitely many possible such splits, but the information gain only changes when a threshold crosses at datapoint



Dealing with Continuous Features

Want to make splits of the form $X_i \geq t$ for some threshold t:

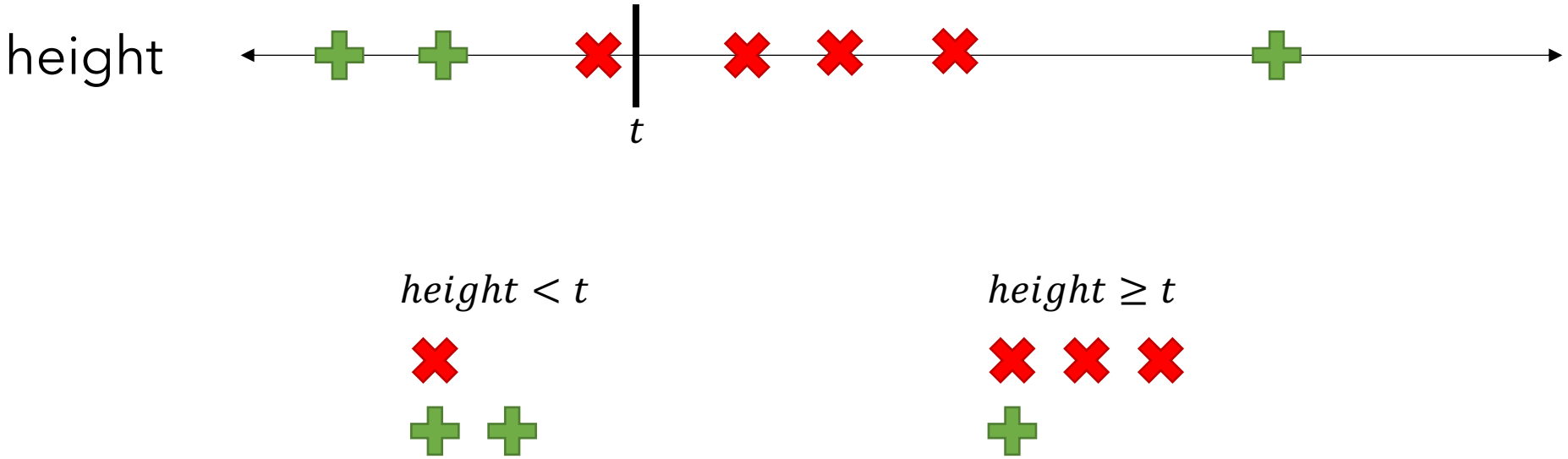
- There are infinitely many possible such splits, but the information gain only changes when a threshold crosses at datapoint



Dealing with Continuous Features

Want to make splits of the form $X_i \geq t$ for some threshold t:

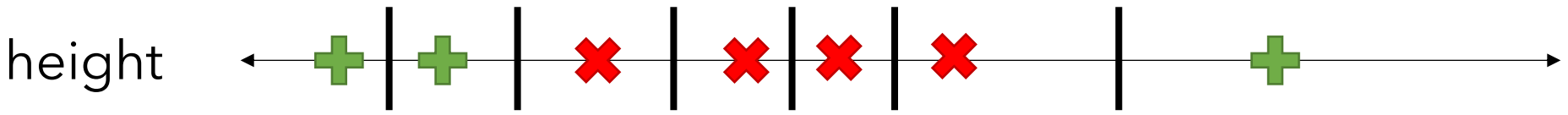
- There are infinitely many possible such splits, but the information gain only changes when a threshold crosses at datapoint



Dealing with Continuous Features

Want to make splits of the form $X_i \geq t$ for some threshold t:

- Sort the values of X_i in the dataset, consider thresholds that occur in between consecutive datapoints.
- Compute information gain for each and choose the max.



Considering both discrete and continuous features

Suppose we have both continuous and discrete features:

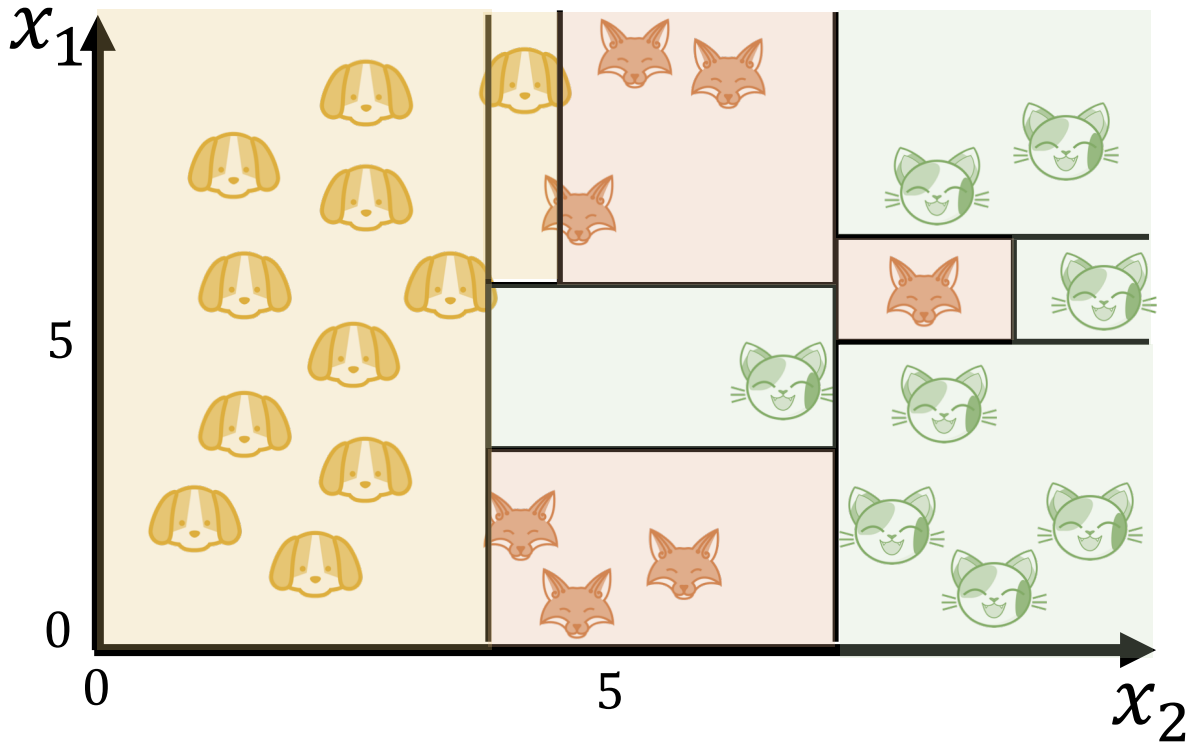
- **Continuous:** Cylinder Volume, Vehicle Weight
- **Discrete:** Manufacturer, Diesel/Regular

Don't really need to do anything special for this case.

- At each step consider all possible splits including:
  - Splits on discrete attributes
  - Threshold splits in-between datapoints for continuous attributes

Problem of Overfitting

As decision trees get bigger (deeper), they can represent very complex functions.



Really large trees can achieve minimal error on training sets, but may not generalize to test data – overfitting.

Problem of Overfitting

Really large trees can achieve minimal error on **training sets**, but may not generalize to **test data** – overfitting.

Avoiding Overfitting

Option 1: Add more hyperpameters to control tree size:

- Limit depth / limit number of nodes / only split if at least K datapoints present

Option 2: Early stopping based on validation performance

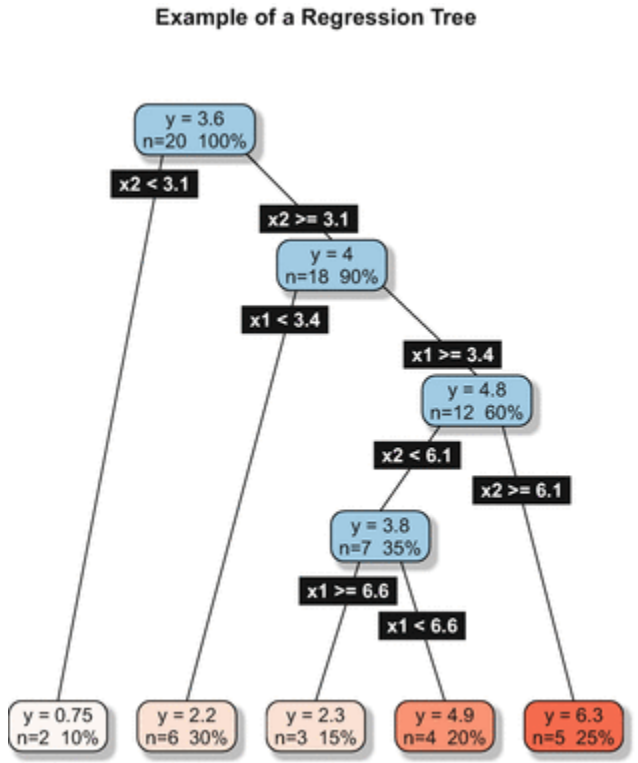
- Monitor the validation accuracy and stop splitting a branch when performance saturates.
- Can be tricky for the same reasons that our proposed Base Case 3 can be.

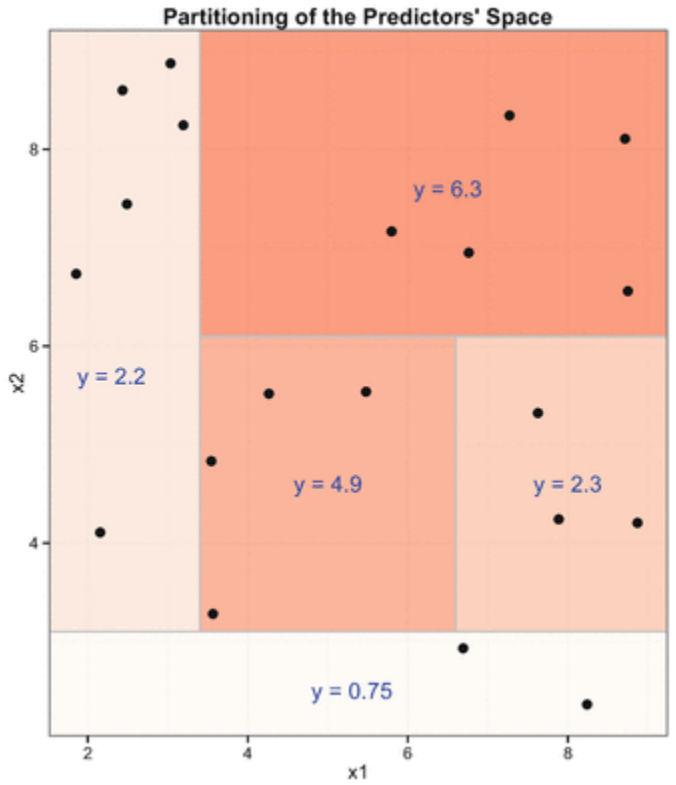
Option 3: Post Pruning

- Grow full tree on the training set, then consider the impact of removing each node on validation performance.
- Greedily prune the node that most improves validation set performance.

We've talked mostly about classification trees. Can also do regression:

- Leaf nodes store the average output value of datapoints in them rather than the majority class.
- Rather the conditional entropy, consider something like change in squared error when choosing splits.





Today's Learning Objectives

Be able to answer:

- ~~What are the intuitions for some advanced structures in neural networks?~~

- ~~What are decision trees?~~
- ~~How are they learned?~~
  - ~~What is entropy, conditional entropy, and information gain?~~
- ~~How do they deal with continuous features?~~

- ~~What do to about overfitting?~~

Next time

Next Time: We'll keep talking a bit about Decision Trees and ensemble methods.