Data Mining: Artificial Neural Network

Lecture Notes for Chapter 3

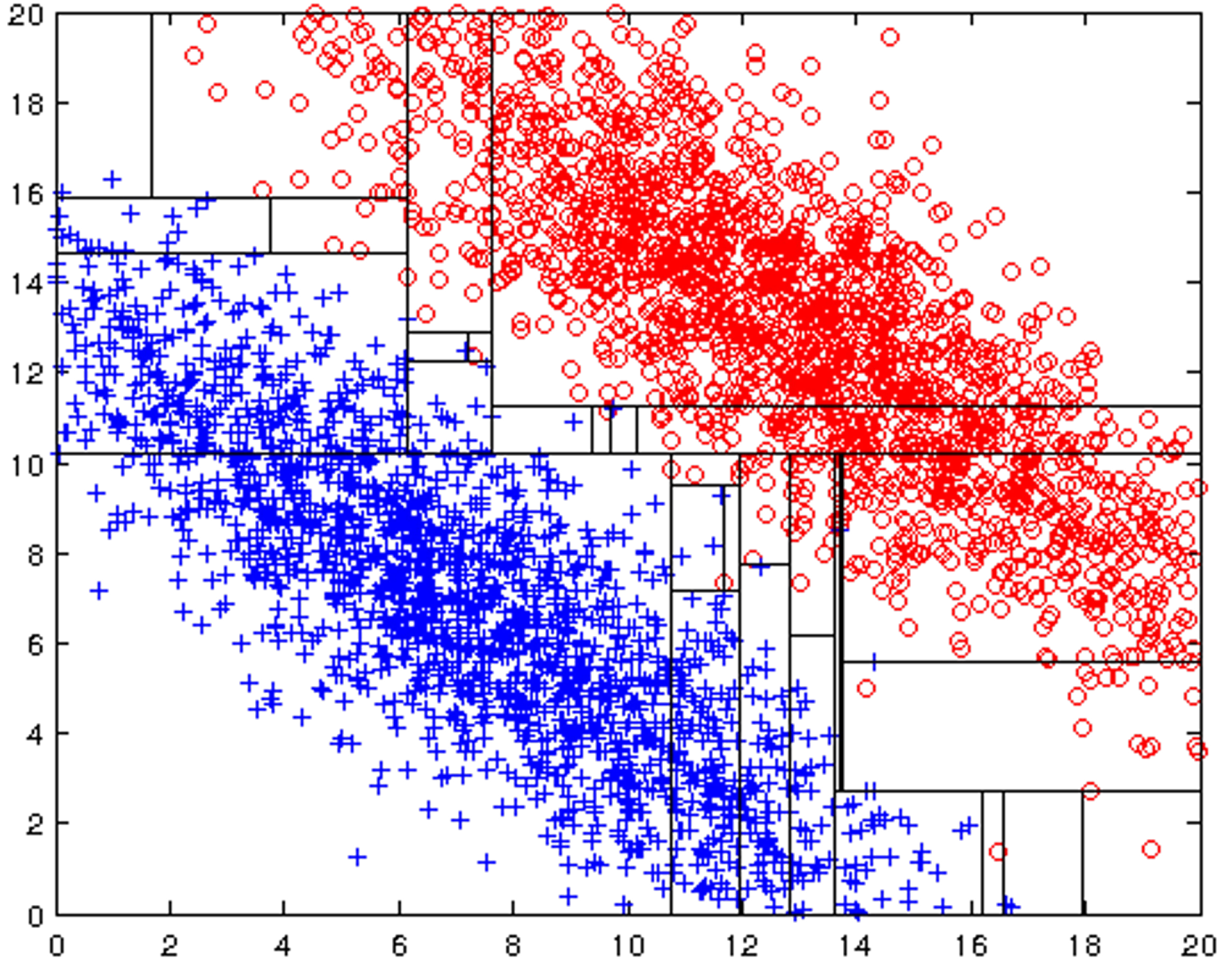
Data Mining

https://ml-graph.github.io/winter-2025/

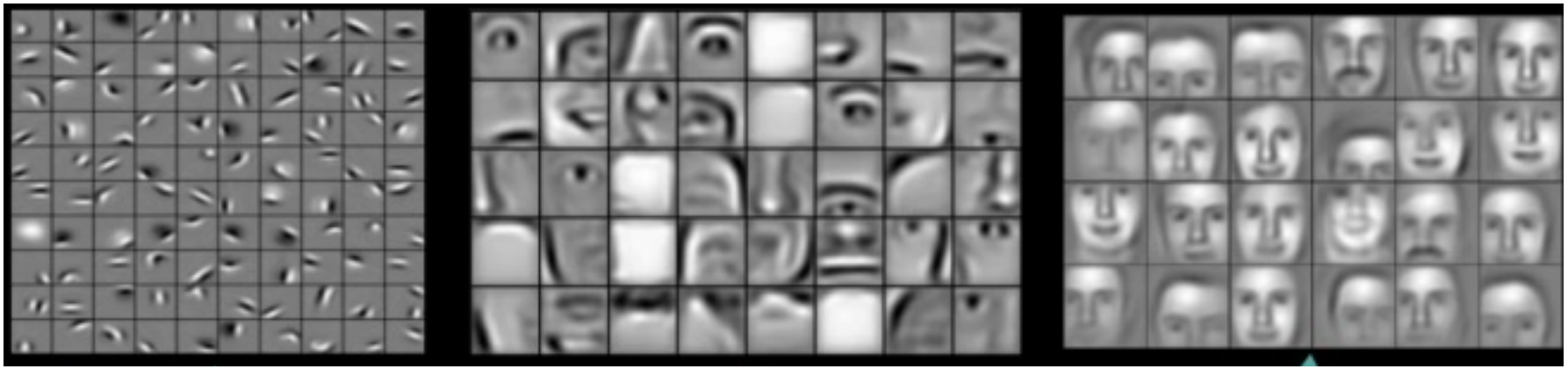
Yu Wang, Ph.D.
yuwang@uoregon.edu
Assistant Professor
Computer Science
University of Oregon
CS 453/553 – Winter 2025

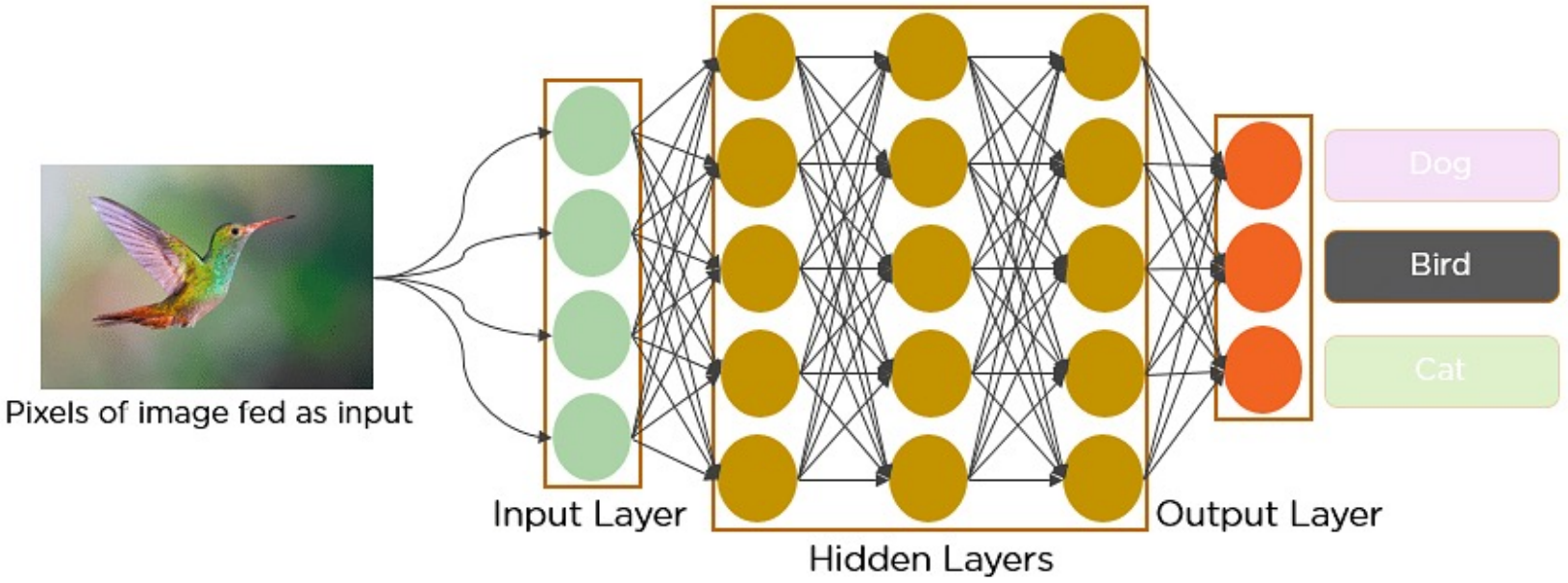
Course Lecture is very heavily based on
"Introduction to Data Mining"
by Tan, Steinbach, Karpatne, Kumar

Limitations of Decision-Tree and many other model

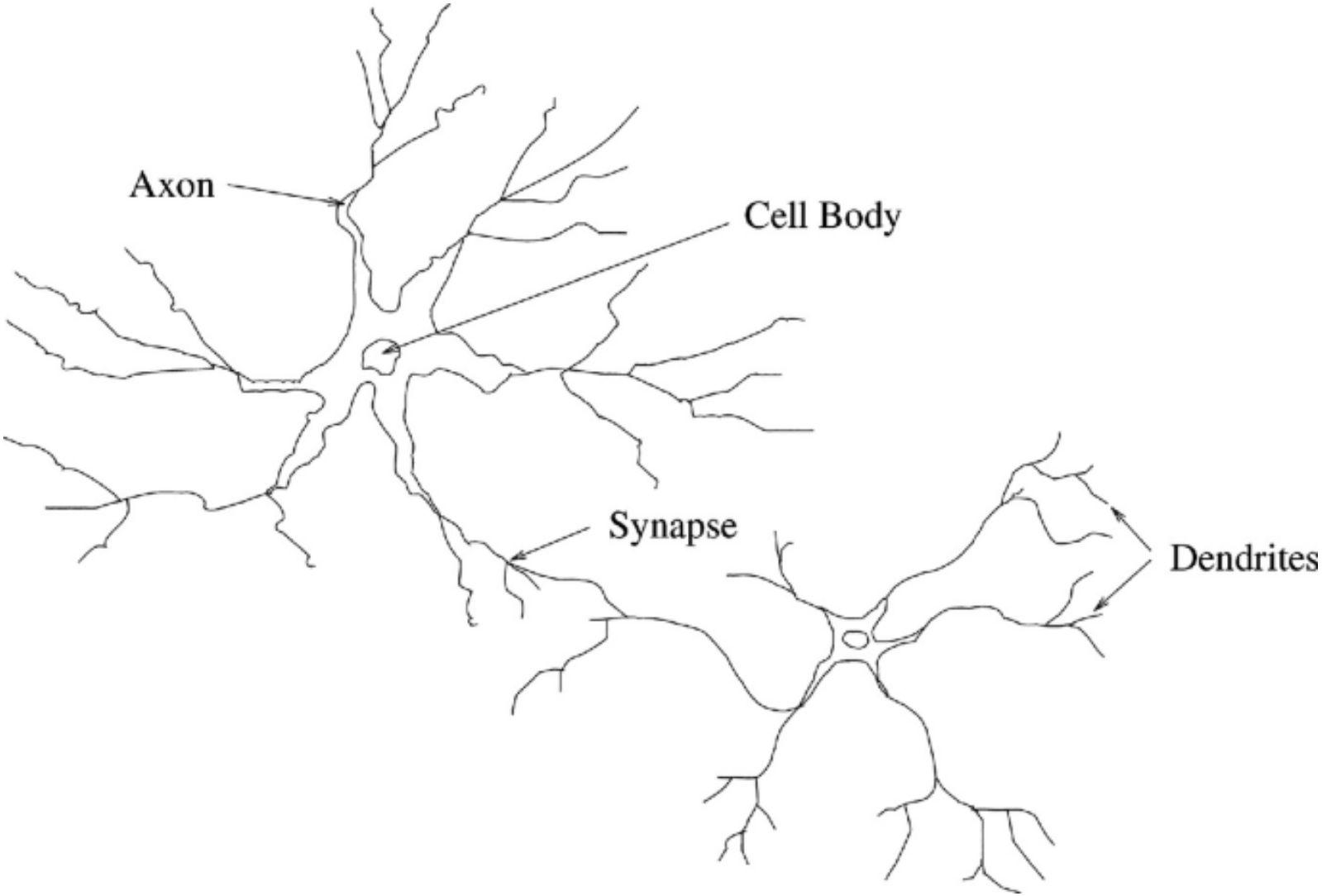


Real-world Intuition





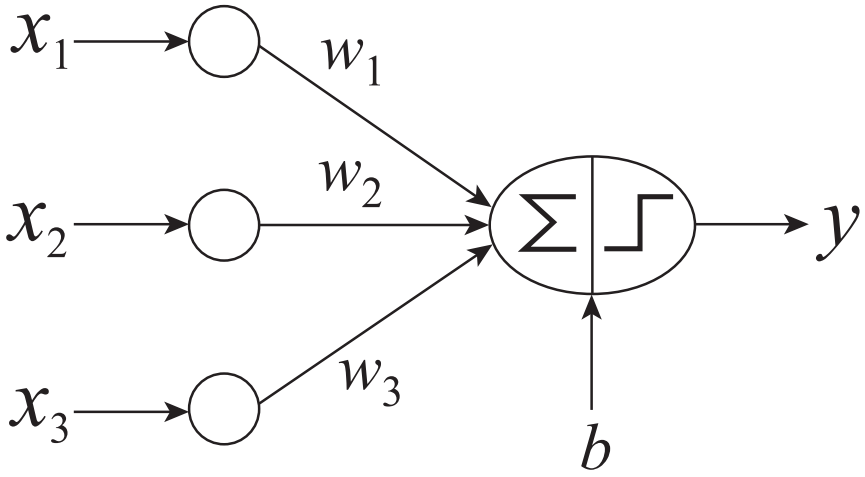
Basic Architecture of Perceptron



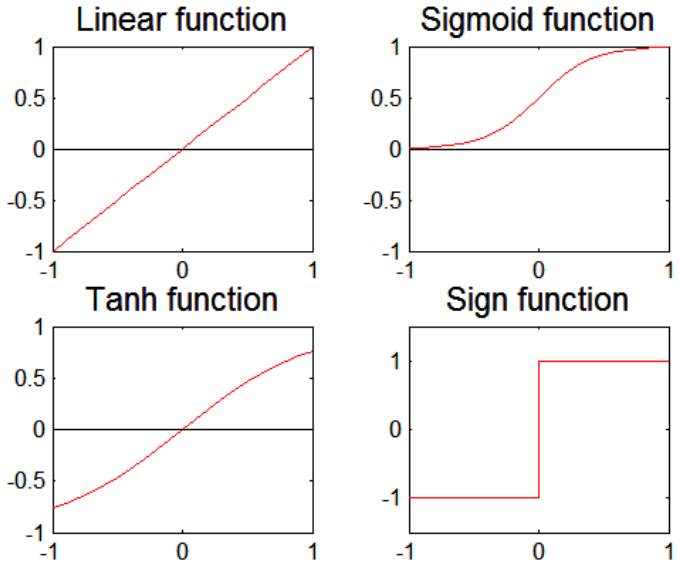
Artificial Neural Networks (ANN)

- Basic Idea: A complex non-linear function can be learned as a composition of simple processing units
- ANN is a collection of simple processing units (nodes) that are connected by directed links (edges)
  - Every node receives signals from incoming edges, performs computations, and transmits signals to outgoing edges
  - Analogous to *human brain* where nodes are neurons and signals are electrical impulses
  - Weight of an edge determines the strength of connection between the nodes
- Simplest ANN: Perceptron (single neuron)

Basic Architecture of Perceptron



$$y = \sigma(w^{\mathrm{T}}x + b)$$



What happens if there is no nonlinear activation?

Linear Regression

- Data - $\{(x^{(i)}, y^{(i)})\}_{i=1}^{N}$

f(x)= $xw + b$

- Regression – Find f that minimizes our uncertainty about y given x

$$y = f(x) + n$$

- Minimizing Mean Squared Error = Minimizing Negative Log-Likelihood

$$\underset{f}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} \left(y^{(i)} - f(x^{(i)})\right)^2$$

Linear Regression

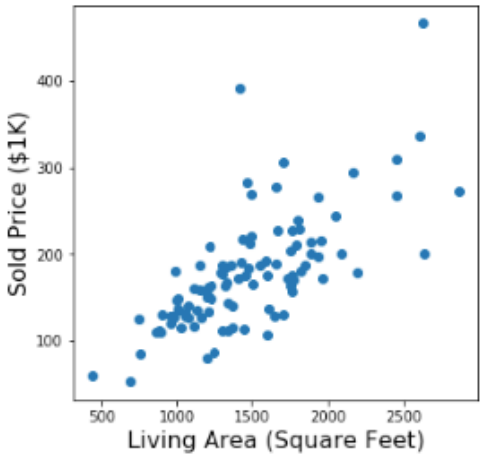
$$\underset{w}{\operatorname{argmin}} \frac{1}{N}\sum_{i=1}^{N}\left(y^{(i)} - w^T\hat{x}^{(i)}\right)^2 = \underset{w}{\operatorname{argmin}} \frac{1}{N}\|y - Xw\|^2$$

Loss/Cost Function

Where

- $y = \left[y^{(1)}, \ldots, y^{(N)}\right]^T \in \mathbb{R}^{N\times 1}$ and
- $X = \left[\hat{x}^{(1)}, \ldots, \hat{x}^{(N)}\right]^T \in \mathbb{R}^{N\times(d+1)}$ (here $d = 1$)
- $w = [w_0, w_1, \ldots, w_d]^{\mathrm{T}} \in \mathbb{R}^{d+1}$

$$\underset{f}{\operatorname{argmin}} \frac{1}{N}\sum_{i=1}^{N}\left(y^{(i)} - f\left(x^{(i)}\right)\right)^2$$



Linear Regression

$$J(w) = \frac{1}{N} \sum_{i=1}^{N} \left(y^{(i)} - w^T \hat{x}^{(i)} \right)^2$$

Compute the minimum value? How to do it in Math?

Find points where gradient = 0

Linear Regression

$$J(w) = \frac{1}{N}\sum_{i=1}^{N}\left(y^{(i)} - w^T\hat{x}^{(i)}\right)^2$$

$$J(W) = \frac{1}{N}(Y - XW)^T(Y - XW)$$

https://www.math.uwaterloo.ca/~hwolkowi/matrixcookbook.pdf

$$J(W) = \frac{1}{N}\left(Y^TY - 2Y^TXW + W^TX^TXW\right)$$

$$\nabla J(W) = \frac{\partial}{\partial W}\left[\frac{1}{N}(Y^TY - 2Y^TXW + W^TX^TXW)\right]$$

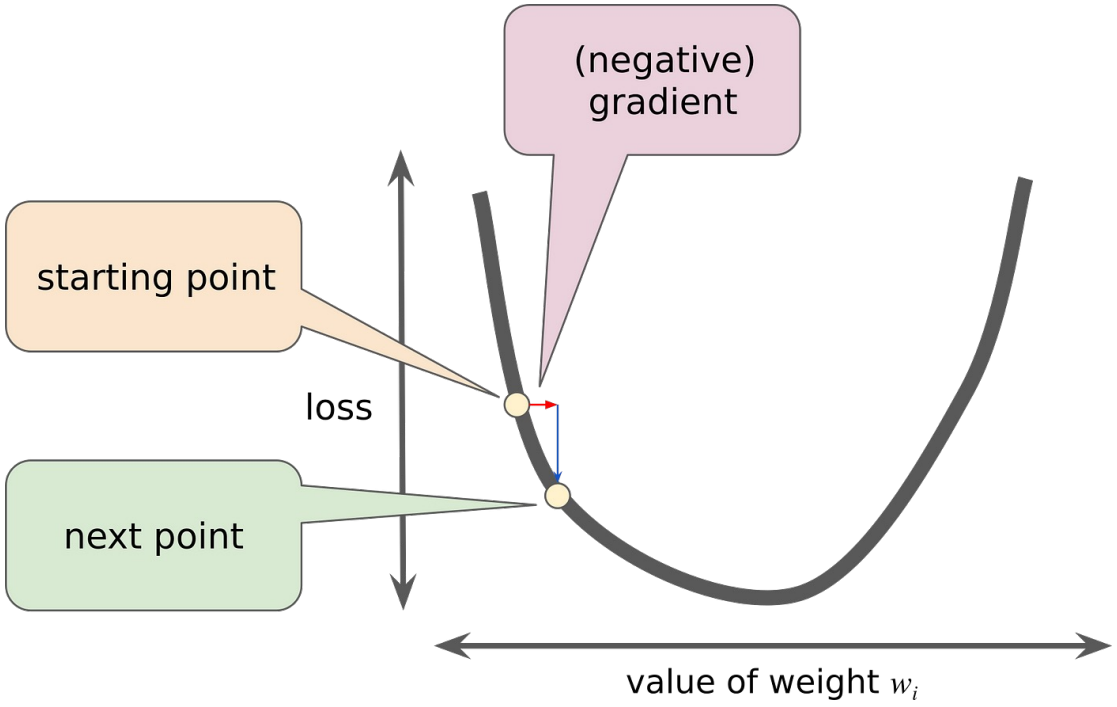
$$X^TXW = X^TY$$

$$W^* = (X^TX)^{-1}X^TY$$

$$\nabla J(W) = -\frac{2}{N}X^TY + \frac{2}{N}X^TXW$$

Linear Regression

$$J(w) = \frac{1}{N} \sum_{i=1}^{N} \left(y^{(i)} - w^T \hat{x}^{(i)} \right)^2$$



If the gradient of a function is non-zero at a point, the direction of the gradient is the direction in which the function increases most quickly

Linear Regression

$$J(w) = \frac{1}{N}\sum_{i=1}^{N}\left(y^{(i)} - w^T\hat{x}^{(i)}\right)^2$$

$$\nabla J(w) = \frac{\partial J(w)}{\partial w}$$

$$\nabla J(w) = -\frac{2}{N}\sum_{i=1}^{N}(y^{(i)} - w^T\hat{x}^{(i)})\hat{x}^{(i)}$$

$$J(w) = \frac{1}{N}\sum_{i=1}^{N}(y^{(i)} - w^T\hat{x}^{(i)})^2$$

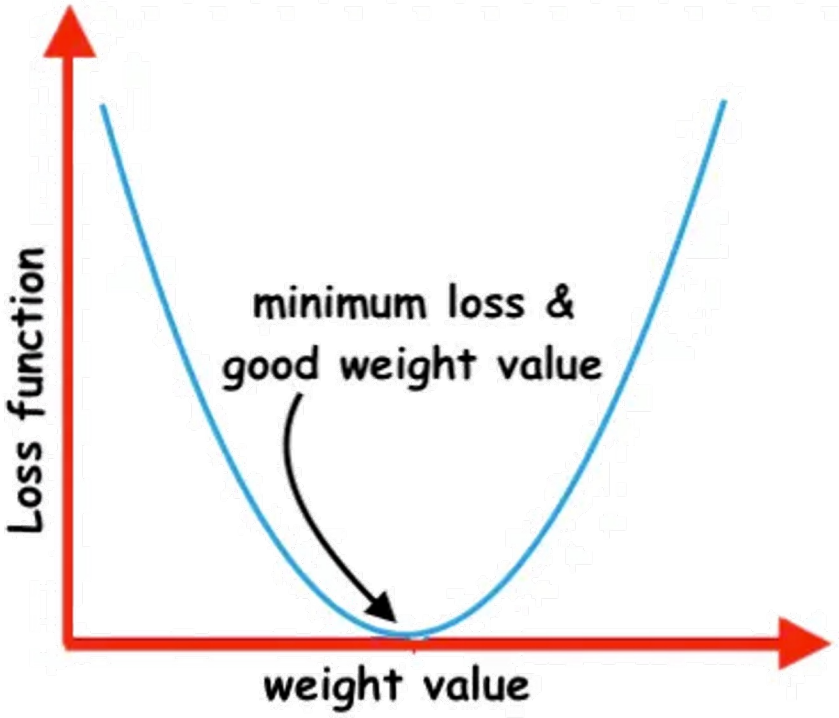
$$w_0 := w_0 + \frac{2\alpha}{N}\sum_{i=1}^{N}(y^{(i)} - (w_0 + w_1 x_1^{(i)}))$$

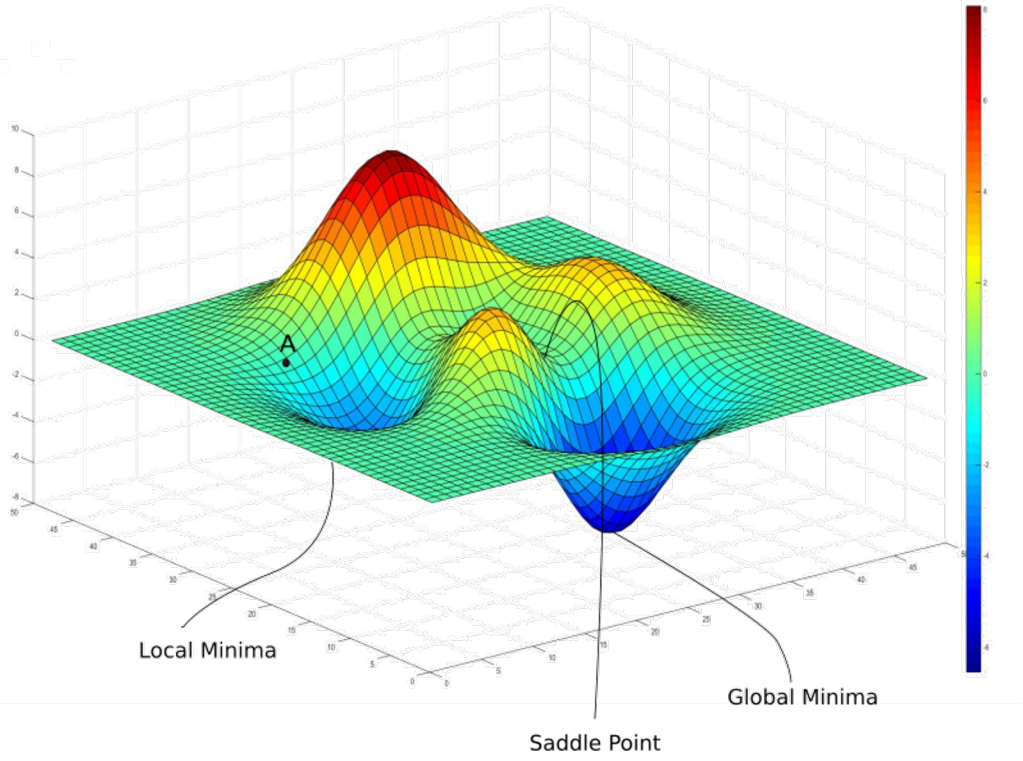
$$\frac{\partial J(w)}{\partial w} = \frac{1}{N}\sum_{i=1}^{N}2(y^{(i)} - w^T\hat{x}^{(i)})(-\hat{x}^{(i)})$$

$$w_1 := w_1 + \frac{2\alpha}{N}\sum_{i=1}^{N}(y^{(i)} - (w_0 + w_1 x_1^{(i)}))x_1^{(i)}$$

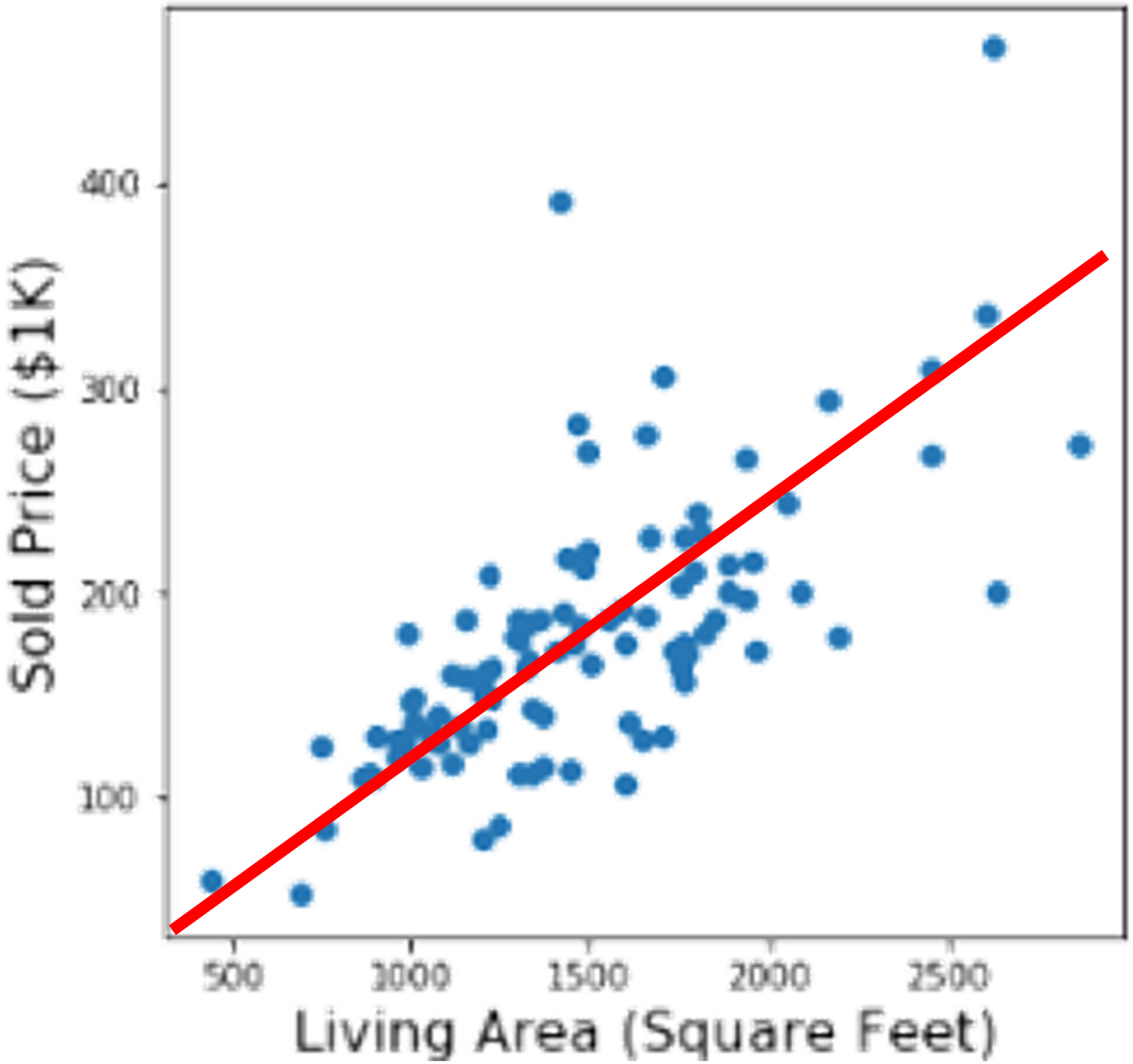
$$= -\frac{2}{N}\sum_{i=1}^{N}(y^{(i)} - w^T\hat{x}^{(i)})\hat{x}^{(i)}$$

Local Minimum vs Global Minimum





Linear Regression

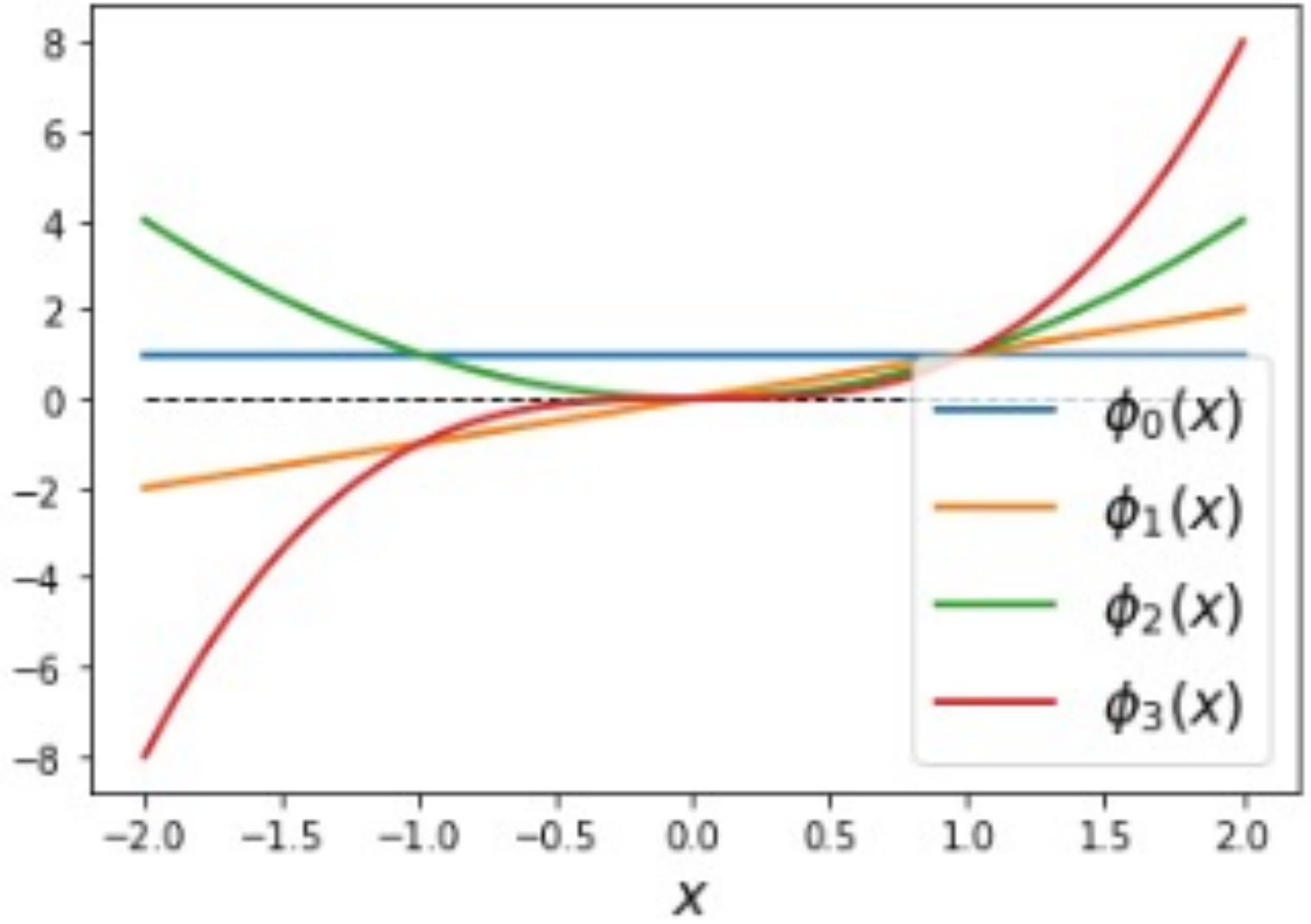


$$f(x) = w_0 x + b$$

Slope

Intercept

Problem of Linear Regression



NonLinear Regression

- So far, we have been using a linear function for regression:

$$f(x) = w^T x + w_0 = \sum_{i=0}^{d} w_i x_i \quad \text{(Assuming } x_0 = 1\text{)}$$

- Lets generalize this model:

$$f(x) = \sum_{i=0}^{M} w_i \phi_i(x) = w^T \phi(x)$$

  where ϕ_i are fixed "basis" functions.

- For linear regression $M = d, \phi_i(x) = x_i$.

NonLinear Regression

$$f(x) = \sum_{i=0}^{M} w_i \phi_i(x) = w^T \phi(x)$$

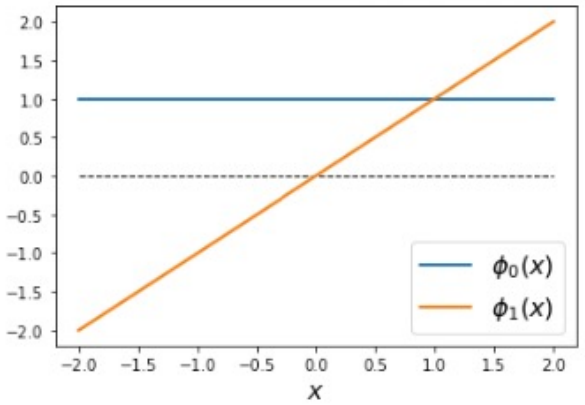
E.g., Polynomial Regression:

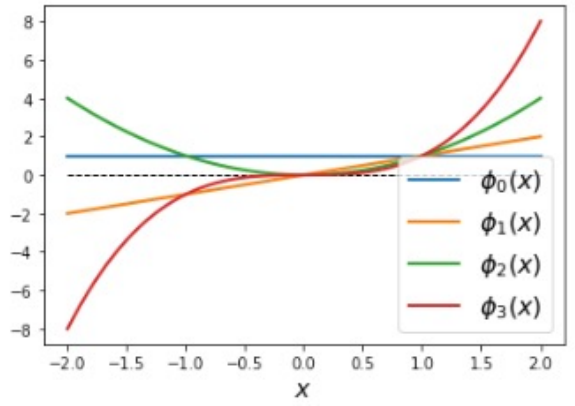
- 1D Polynomial Regression, $\phi(x) = [1, x, x^2, x^3]$:

$$\underset{w}{\operatorname{argmin}} \frac{1}{N} \sum_{n=1}^{N} \left(w^T \phi\left(x^{(i)}\right) - y^{(i)} \right)^2$$

To avoid confusion, note that: $\phi\left(x^{(i)}\right) = \left[1, x^{(i)}, (x^{(i)})^2, (x^{(i)})^3\right]$

$$\mathrm{f}\left(x^{(i)}\right) = w_0 + w_1 x^{(i)} + w_2 \left(x^{(i)}\right)^2 + w_3 \left(x^{(i)}\right)^3$$





NonLinear Regression

Loss: $$\underset{w}{\operatorname{argmin}} \frac{1}{N} \sum_{n=1}^{N} \left(w^T \phi\left(x^{(i)}\right) - y^{(i)}\right)^2 = \underset{w}{\operatorname{argmin}} \|\Phi w - y\|^2$$

Where $\Phi = \left[\phi\left(x^{(1)}\right), \ldots, \phi\left(x^{(N)}\right)\right]^T \in \mathbb{R}^{N \times M}$ and $w \in \mathbb{R}^M$.

Optimization:

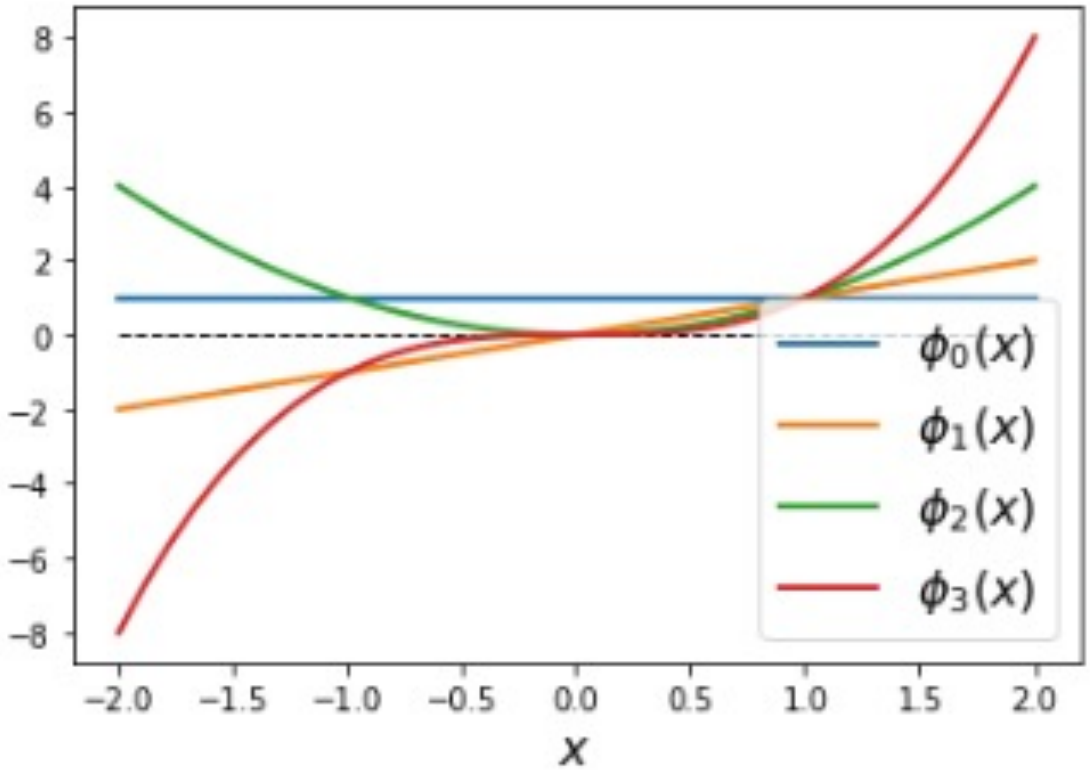
1. Closed form solution: $w^* = (\Phi^T \Phi)^{-1} \Phi^T y$
2. Gradient descent: $w^{(t)} = w^{(t-1)} - \epsilon \nabla_w \, Loss(w^{(t-1)})$

$$J(w) = \frac{1}{N} \sum_{i=1}^{N} \left(y^{(i)} - w^T \hat{x}^{(i)}\right)^2$$

$$W^* = (X^T X)^{-1} X^T Y$$

What is the problem of Nonlinear Regression?

NonLinear Regression



The basis function is all fixed!

Can we learn the basis function?

NonLinear Regression

- Lets first look at what the learning problem might look like:

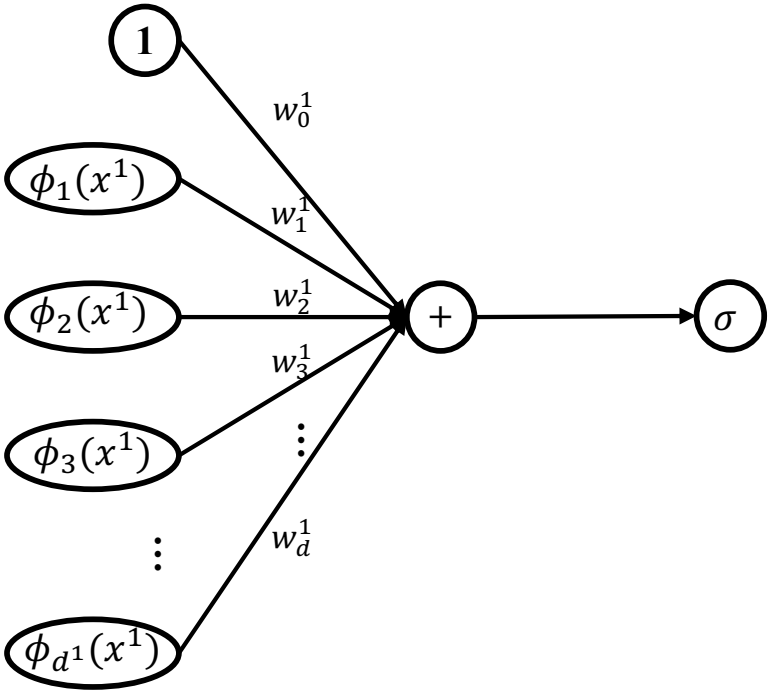
$$\underset{w}{\operatorname{argmin}} \sum_i \left(\left(\sum_j w_j \phi_j\left(x^{(i)}\right) \right) - y^{(i)} \right)^2$$

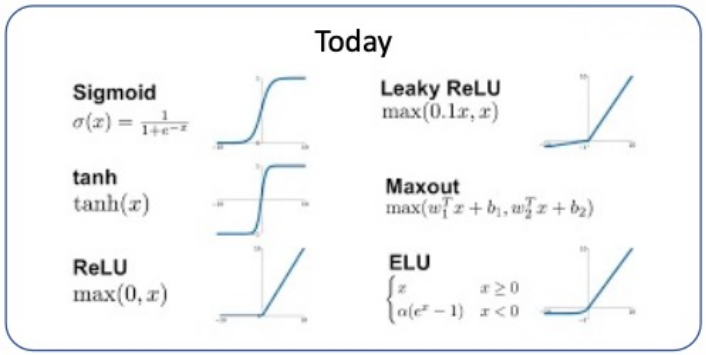
Neural Networks do this for us!

What things are learned here?
What things are fixed here?

NonLinear Regression

1-layer Multi-layer Perceptron





$$\sum_{j=0}^{d^1} w_j^1 \phi_j(x^1)$$

NonLinear Regression

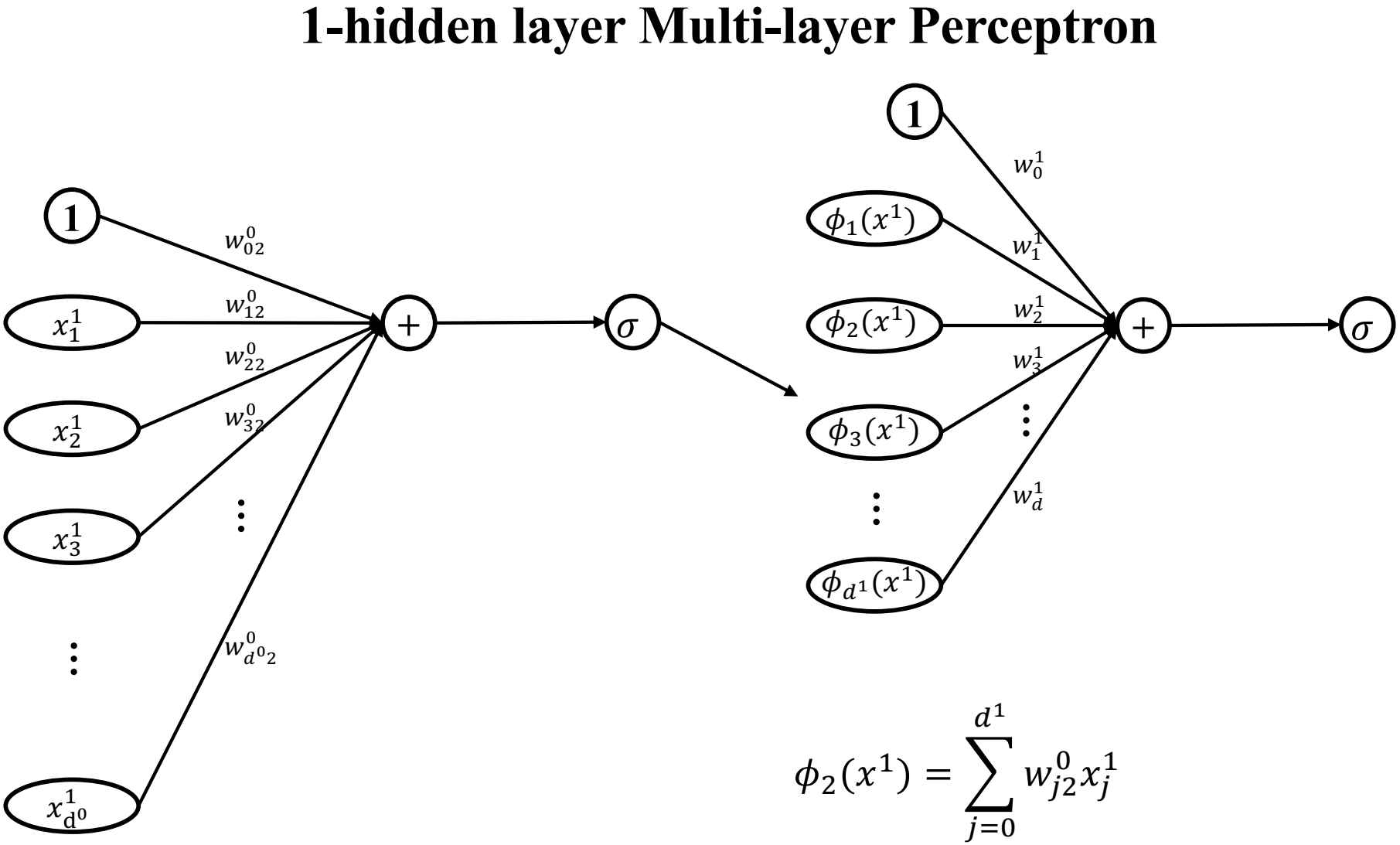
- Lets first look at what the learning problem might look like:

$$\underset{w,\{\phi_j\}_{j=1}^{M}}{\operatorname{argmin}} \sum_i \left(\left(\sum_j w_j \phi_j\left(x^{(i)}\right) \right) - y^{(i)} \right)^2$$

Neural Networks do this for us!

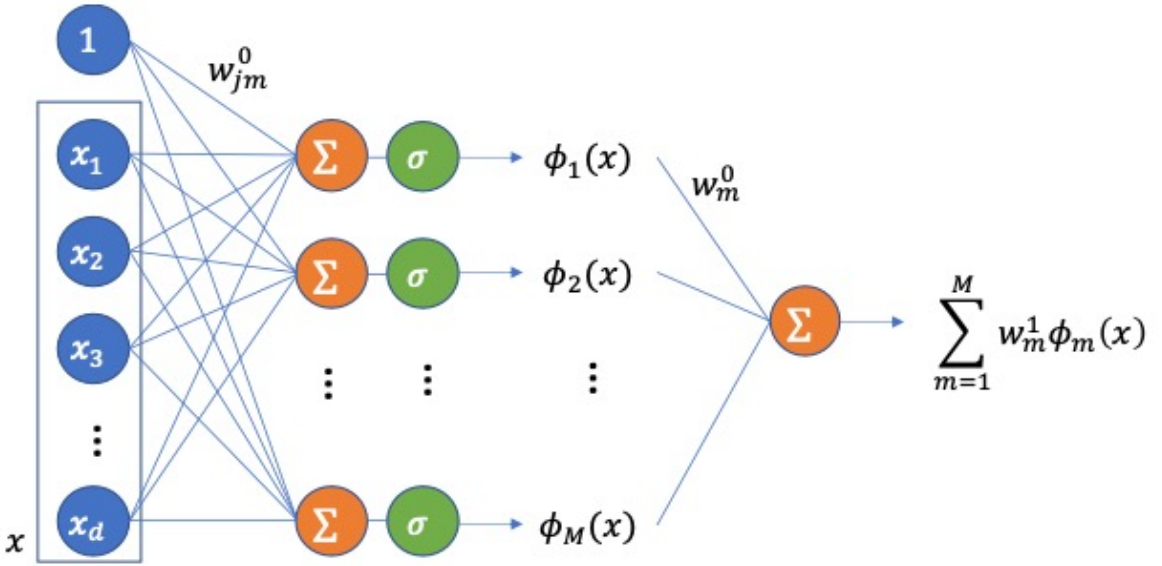
What things are learned here?
What things are fixed here?

NonLinear Regression



NonLinear Regression

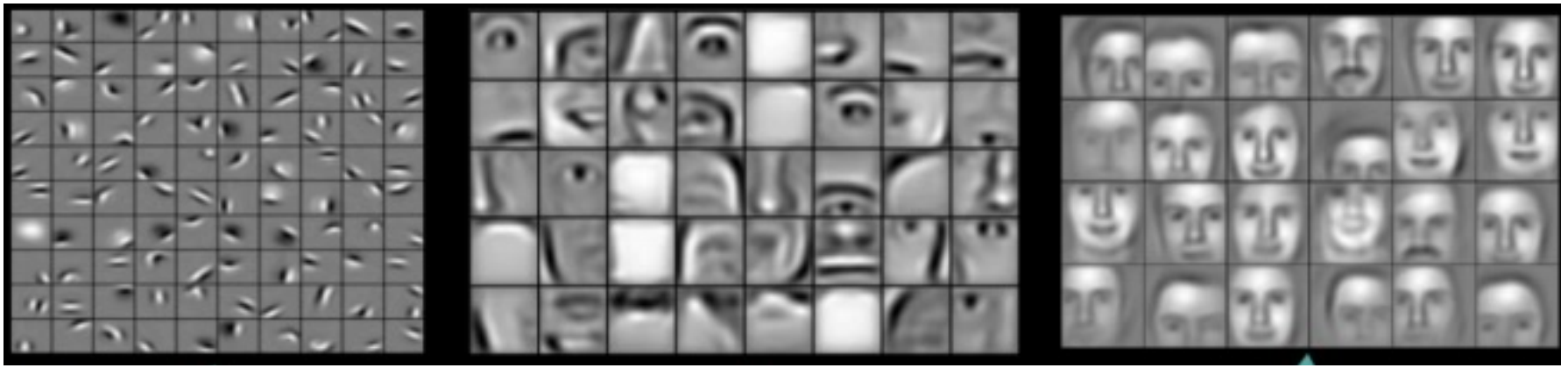
1-hidden layer Multi-layer Perceptron



$$f(x) = \sum_{m=1}^{M} w_m^1 \sigma\left(\sum_{j=1}^{d} W_{jm}^0 x_j\right)$$

Example

- Activations at hidden layers can be viewed as features extracted as functions of inputs
- Every hidden layer represents a level of abstraction
  - *Complex features are compositions of simpler features*

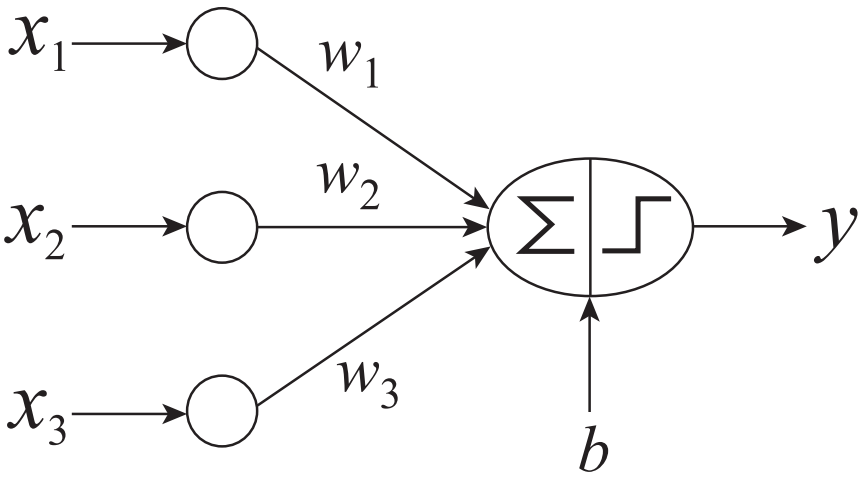


- Number of layers is known as depth of ANN
  - *Deeper networks express complex hierarchy of features*

Question?

1. "Judge a man by his questions rather than by his answers."
   – Voltaire
2. "If I had an hour to solve a problem, I'd spend 55 minutes thinking about the problem and 5 minutes thinking about solutions."
   – Albert Einstein
3. "The art and science of asking questions is the source of all knowledge."
   – Thomas Berger
4. "Asking the right questions takes as much skill as giving the right answers."
   – Robert Half
5. "The wise man doesn't give the right answers, he poses the right questions."
   – Claude Lévi-Strauss
6. "Great questions make great companies."
   – Peter Drucker

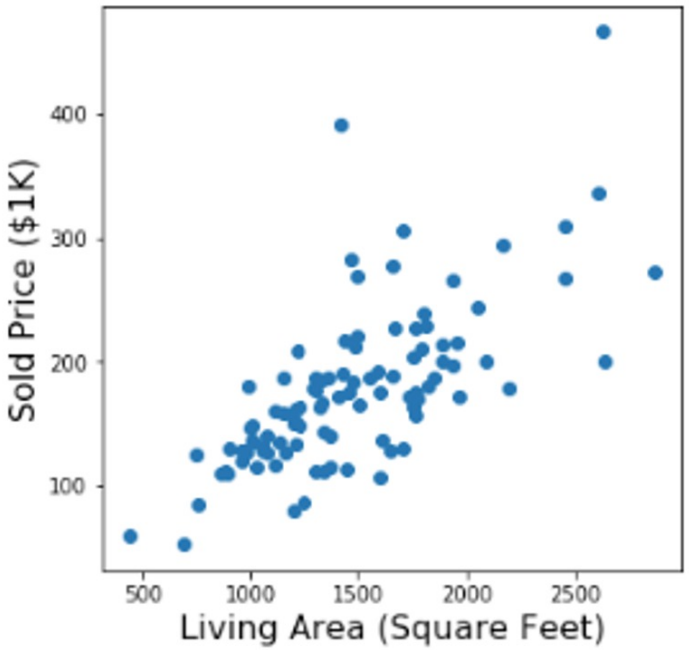
Linear Regression



$$\sigma(w^{\mathrm{T}}x + b)$$

$$\underset{w}{\operatorname{argmin}} \boxed{\frac{1}{N}\sum_{i=1}^{N}\left(y^{(i)} - w^T\hat{x}^{(i)}\right)^2} = \underset{w}{\operatorname{argmin}} \boxed{\frac{1}{N}\|y - Xw\|^2}$$

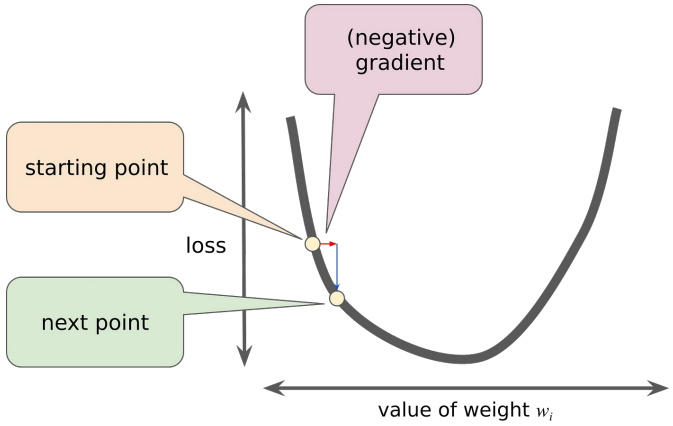
Loss/Cost Function



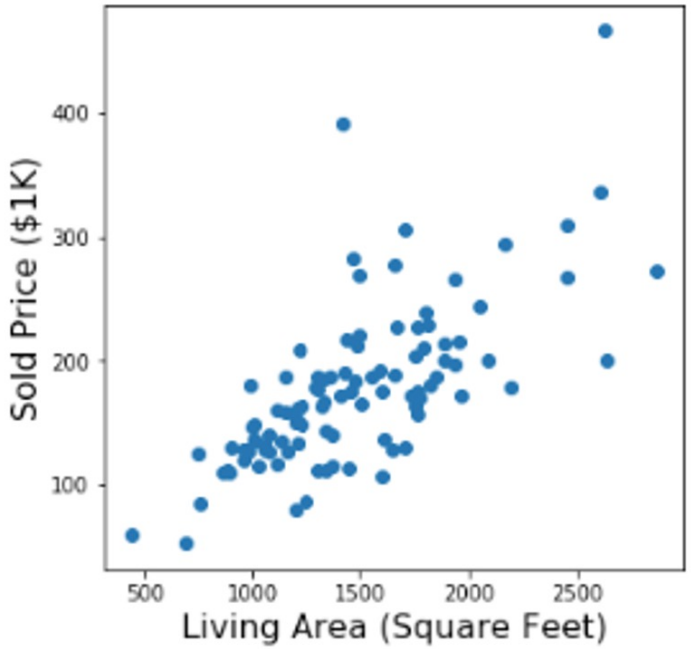
Analytical Solution

Gradient Descent

$$\nabla J(W) = -\frac{2}{N}X^TY + \frac{2}{N}X^TXW \qquad W^* = (X^TX)^{-1}X^TY$$



Linear Regression



$$\mathrm{y}= \sigma(w^{\mathrm{T}}x + b)$$

$$y = \sigma(w^{\mathrm{T}}\phi(x) + b)$$

$$\nabla J(W) = -\frac{2}{N}X^T Y + \frac{2}{N}X^T XW$$

$$\nabla J(W) = -\frac{2}{N}\phi(X)^{\mathrm{T}} Y + \frac{2}{N}\phi(X)^T \phi(X)W$$

$$W^* = (X^T X)^{-1} X^T Y$$

$$W^* = (\phi(X)^T \phi(X))^{-1} \phi(X)^T Y$$

Linear Regression

$$J(w) = \frac{1}{N}\sum_{i=1}^{N}\left(y^{(i)} - w^T\hat{x}^{(i)}\right)^2$$

$$\nabla J(w) = \frac{\partial J(w)}{\partial w}$$

$$\nabla J(w) = -\frac{2}{N}\sum_{i=1}^{N}(y^{(i)} - w^T\hat{x}^{(i)})\hat{x}^{(i)}$$

$$J(w) = \frac{1}{N}\sum_{i=1}^{N}(y^{(i)} - w^T\hat{x}^{(i)})^2$$

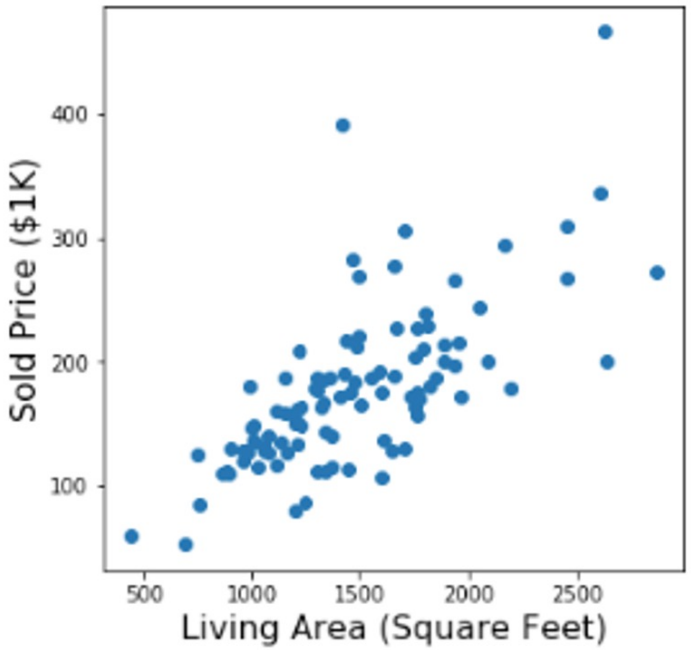
$$w_0 := w_0 + \frac{2\alpha}{N}\sum_{i=1}^{N}(y^{(i)} - (w_0 + w_1 x_1^{(i)}))$$

$$\frac{\partial J(w)}{\partial w} = \frac{1}{N}\sum_{i=1}^{N}2(y^{(i)} - w^T\hat{x}^{(i)})(-\hat{x}^{(i)})$$

$$w_1 := w_1 + \frac{2\alpha}{N}\sum_{i=1}^{N}(y^{(i)} - (w_0 + w_1 x_1^{(i)}))x_1^{(i)}$$

$$= -\frac{2}{N}\sum_{i=1}^{N}(y^{(i)} - w^T\hat{x}^{(i)})\hat{x}^{(i)}$$

Linear Regression

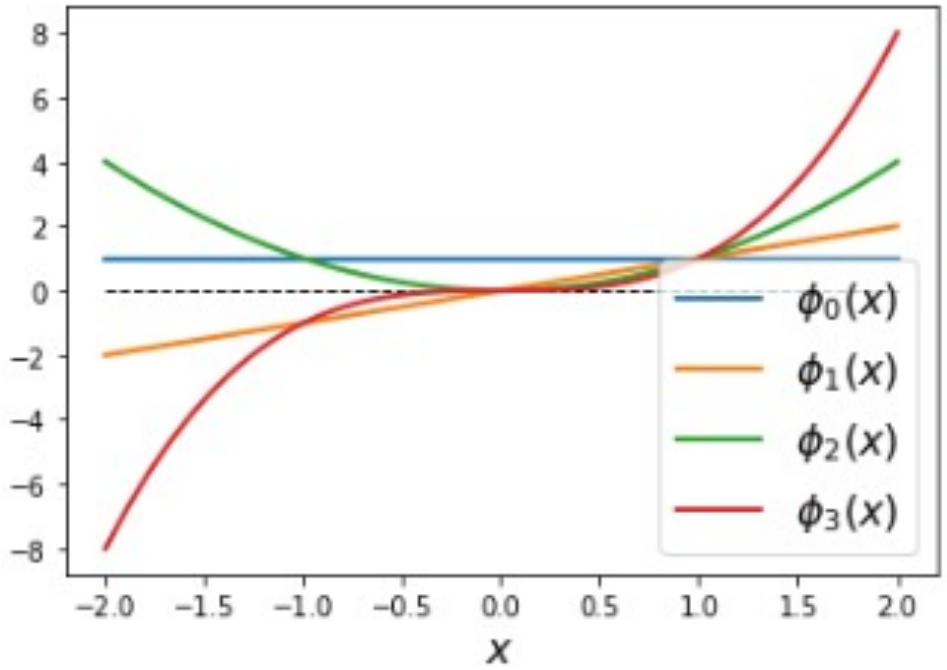


$$y = w_1 x + w_0$$

$$w_0 := w_0 + \frac{2\alpha}{N} \sum_{i=1}^{N} (y^{(i)} - (w_0 + w_1 x_1^{(i)}))$$

$$w_1 := w_1 + \frac{2\alpha}{N} \sum_{i=1}^{N} (y^{(i)} - (w_0 + w_1 x_1^{(i)})) x_1^{(i)}$$

Non-Linear Regression



$$\mathrm{y}= \sigma(w^{\mathrm{T}}x + b)$$

$$y = \sigma(w^{\mathrm{T}}\phi(x) + b)$$

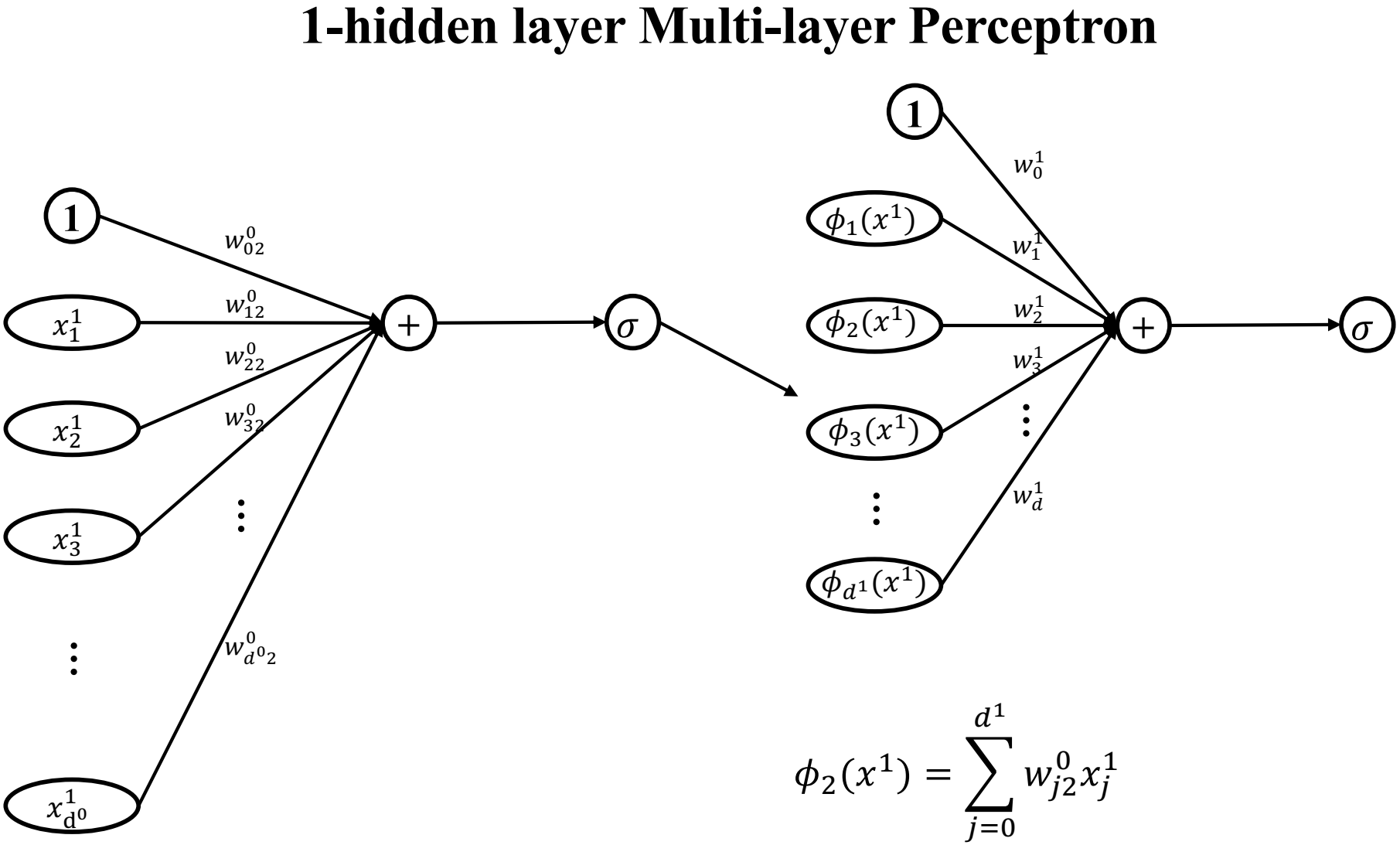
$$\nabla J(W) = -\frac{2}{N}X^T Y + \frac{2}{N}X^T XW$$

$$\nabla J(W) = -\frac{2}{N}\phi(X)^{\mathrm{T}} Y + \frac{2}{N}\phi(X)^T \phi(X)W$$

$$W^* = (X^T X)^{-1}X^T Y$$

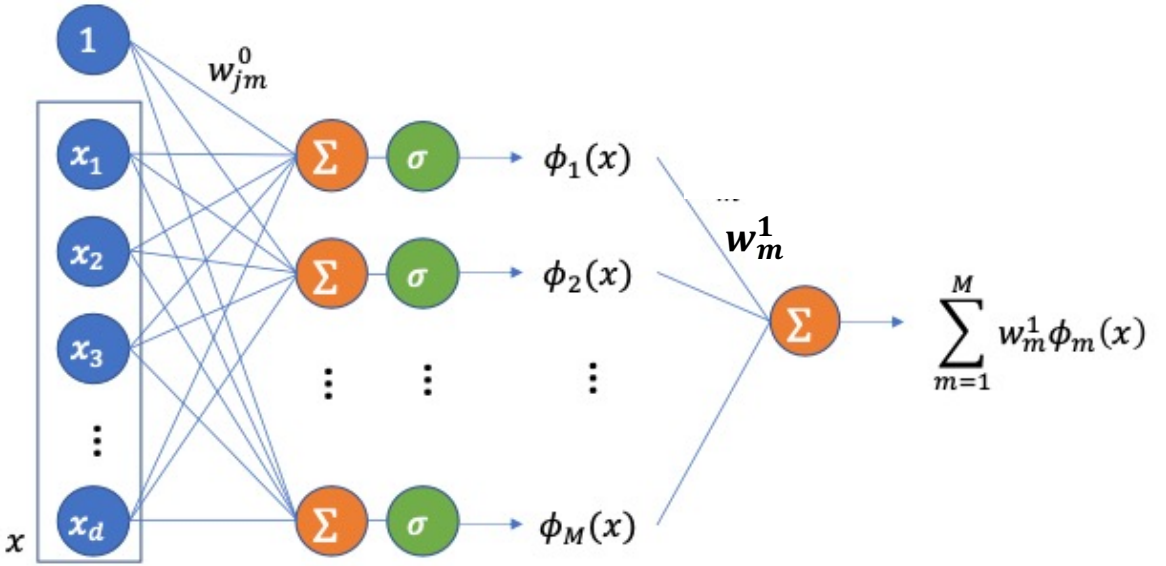
$$W^* = (\phi(X)^T \phi(X))^{-1}\phi(X)^T Y$$

Multi-NonLinear Regression



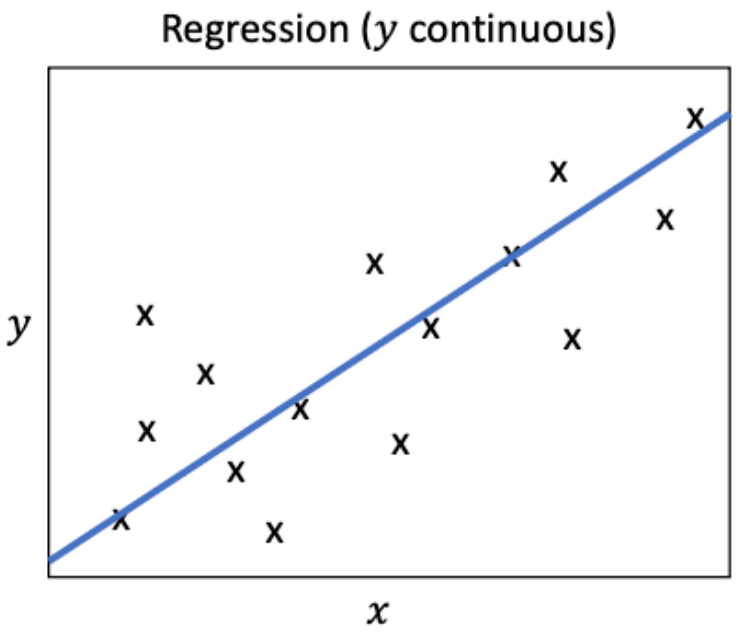
Multi-NonLinear Regression

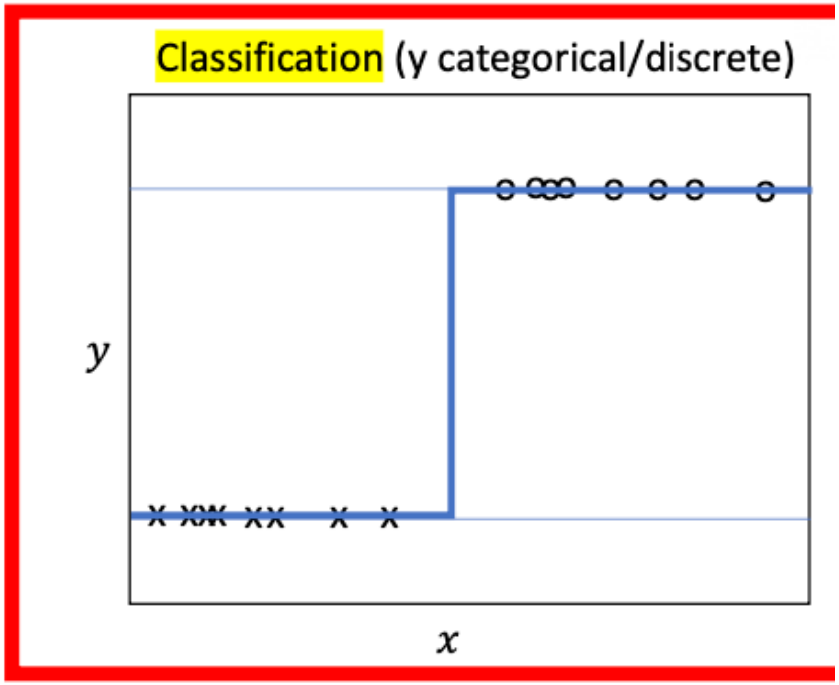
1-hidden layer Multi-layer Perceptron



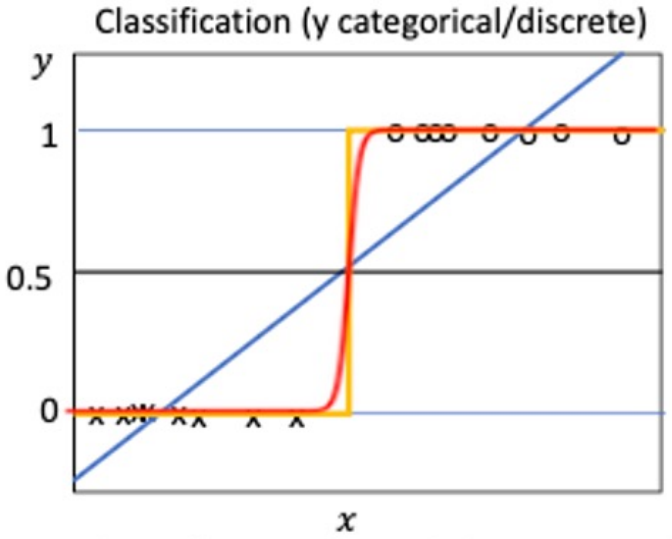
$$f(x) = \sum_{m=1}^{M} w_m^1 \sigma\left(\sum_{j=1}^{d} W_{jm}^0 x_j\right)$$

Classification





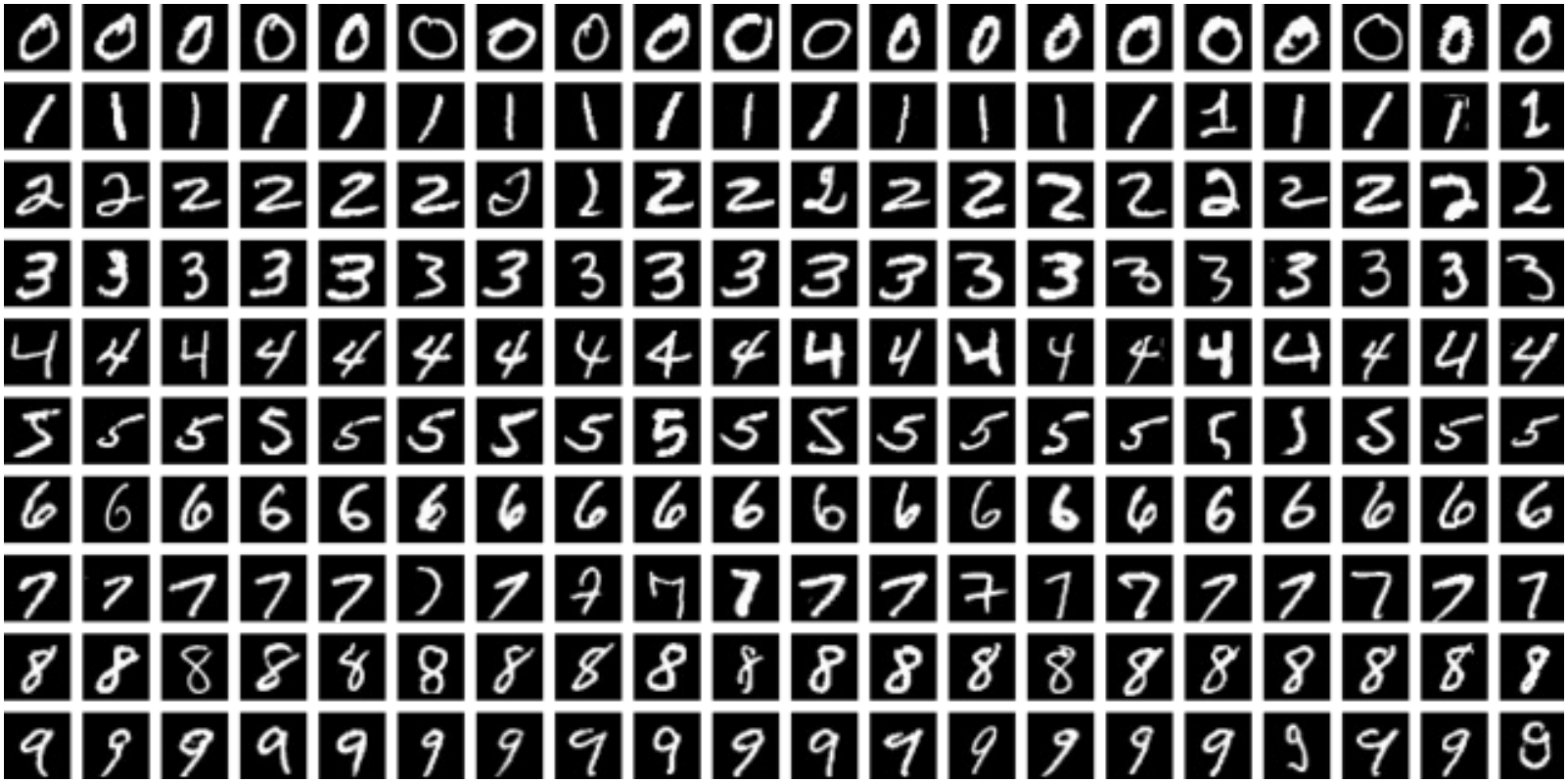
Classification



What loss should we use?

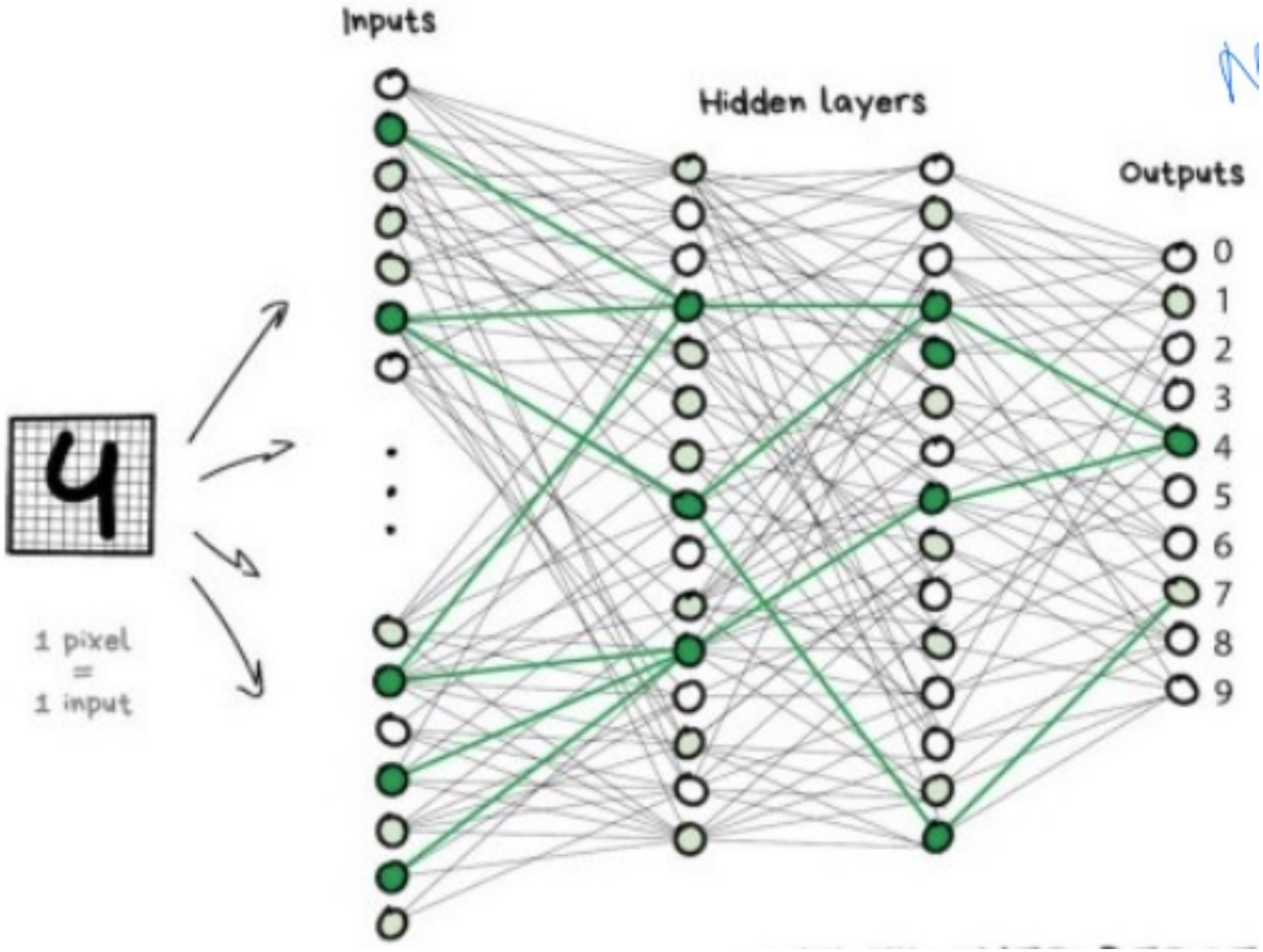
$$\mathcal{L}(w) = -\sum_{i=1}^{N} y^{(i)} \log\left(f\left(x^{(i)}\right)\right) + \left(1 - y^{(i)}\right) \log(1 - f(x_i))$$

MINIST Dataset



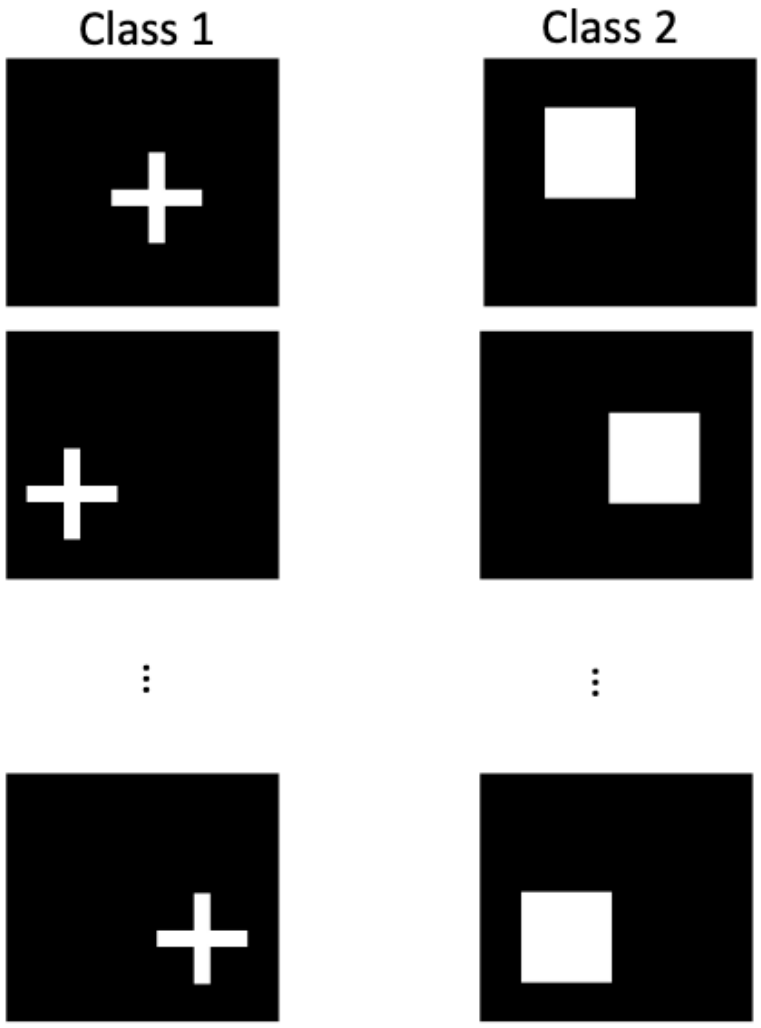
A 256x256 (RGB) image $\Longrightarrow$ ~200K dimensional input x

MLP for MINIST



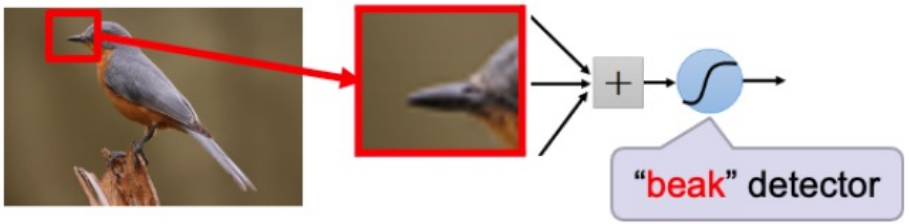
A fully connected network would need a very large number of parameters, very likely to overfit the data

CNN

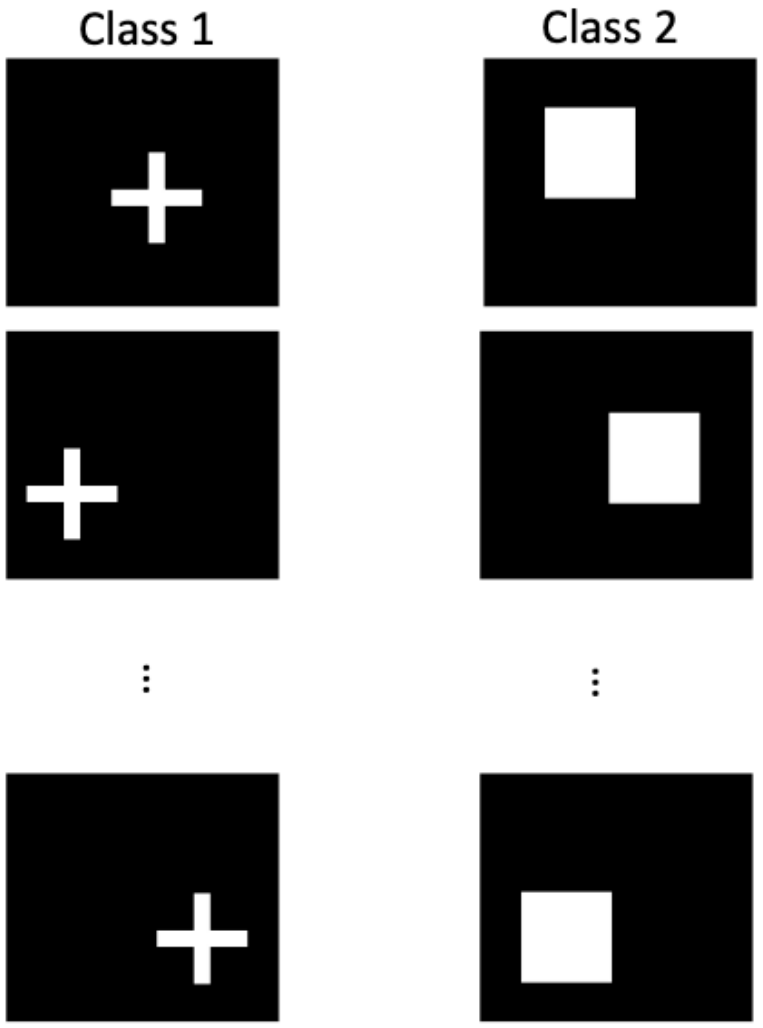


- The features of these classes are spatially local.
- Translation does not change the identity of the classes, i.e., we require a translation equivariant model.
- MLP is not a good match to this problem

Can represent a small region with fewer parameters

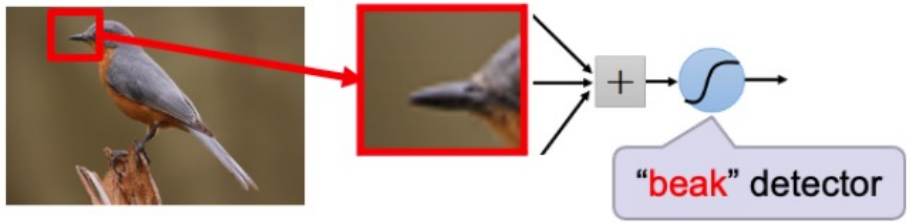


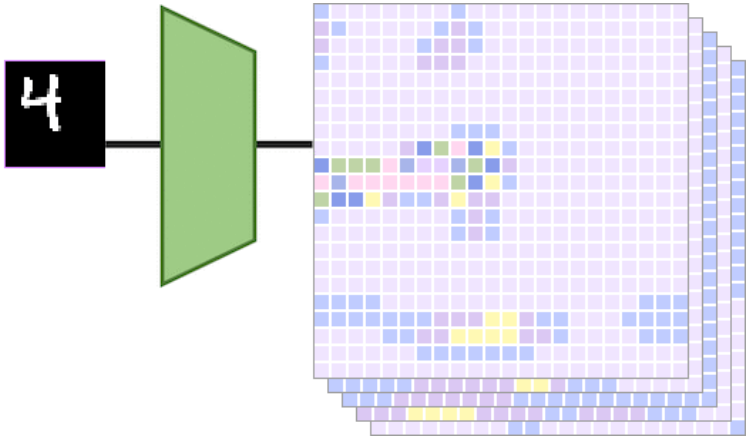
CNN



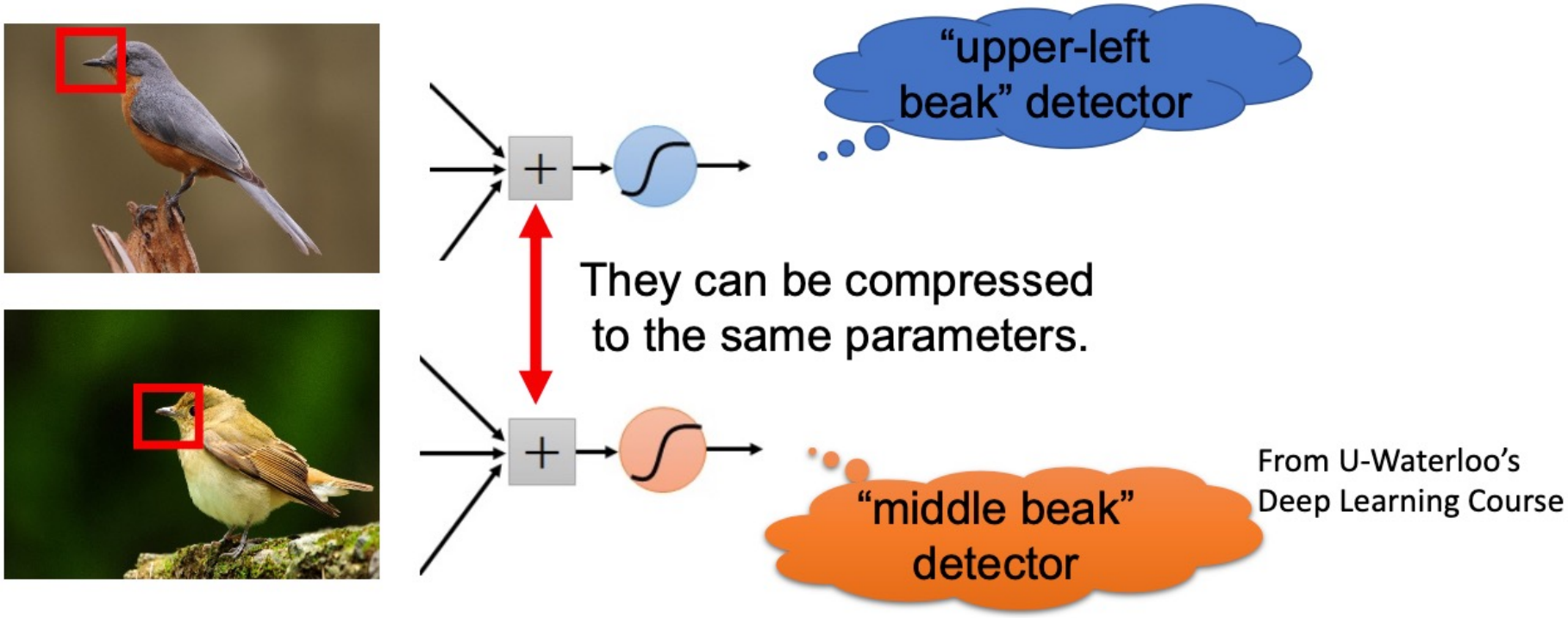
- The features of these classes are spatially local.
- Translation does not change the identity of the classes, i.e., we require a translation equivariant model.
- MLP is not a good match to this problem

Can represent a small region with fewer parameters





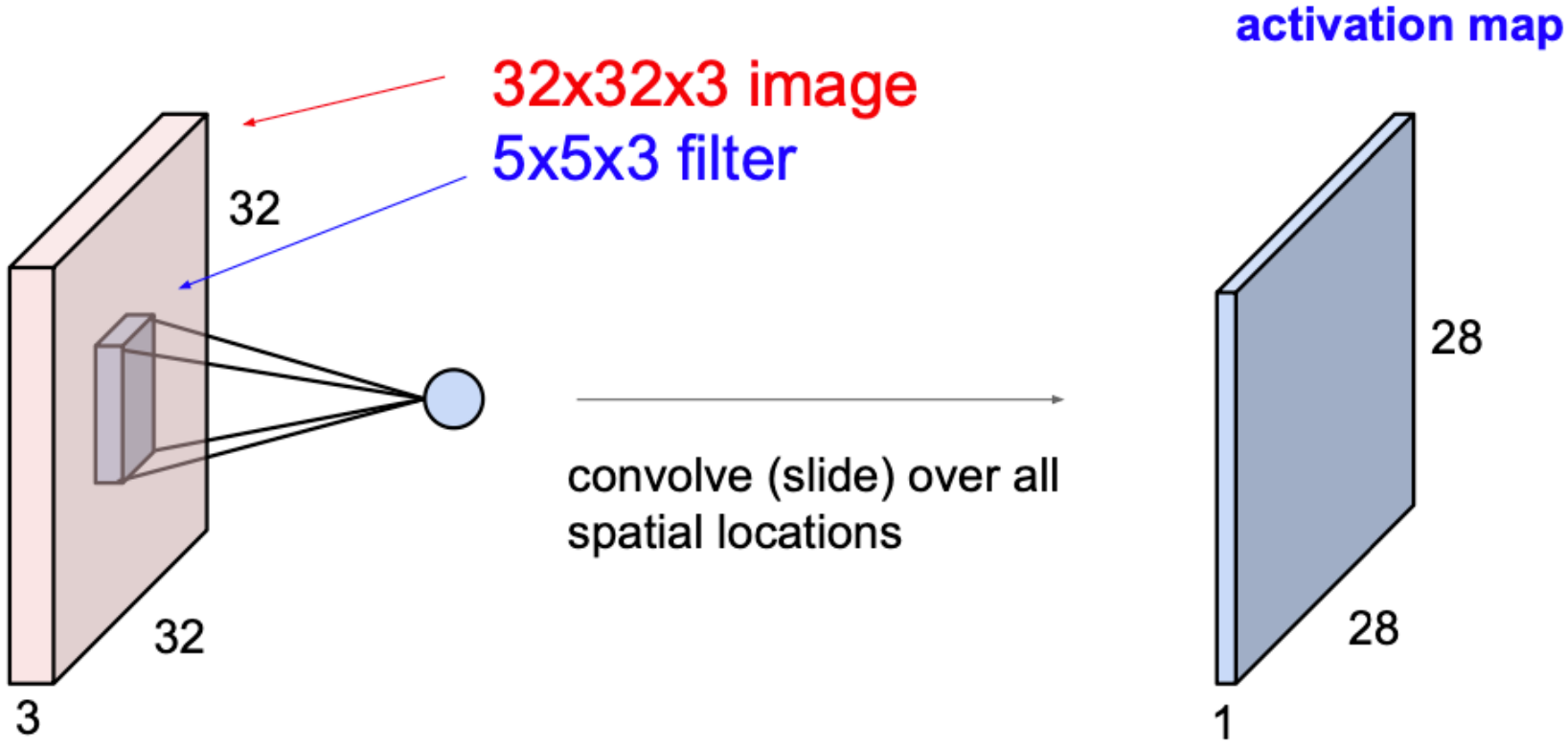
CNN



What about training a lot of such "small" detectors and each detector must "move around".

CNN

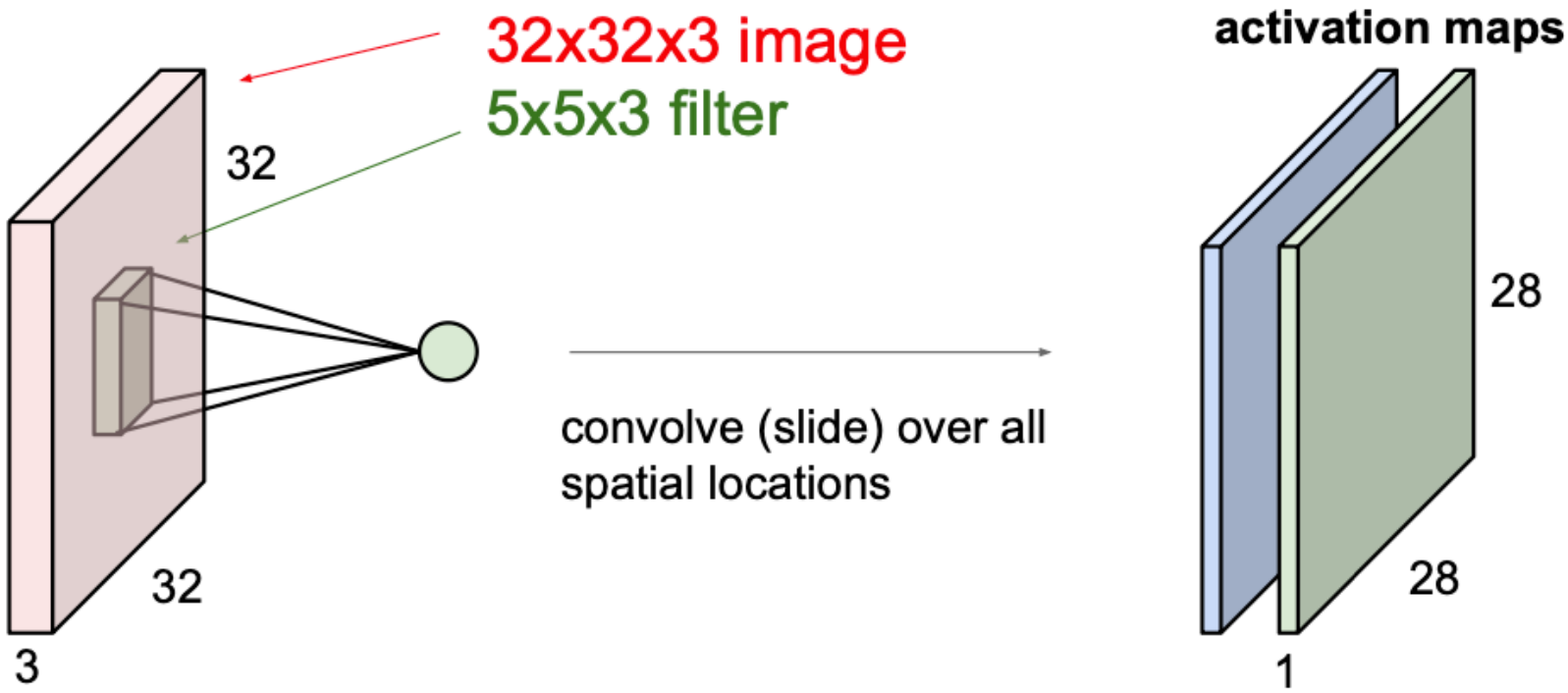
Convolution Layer



CNN

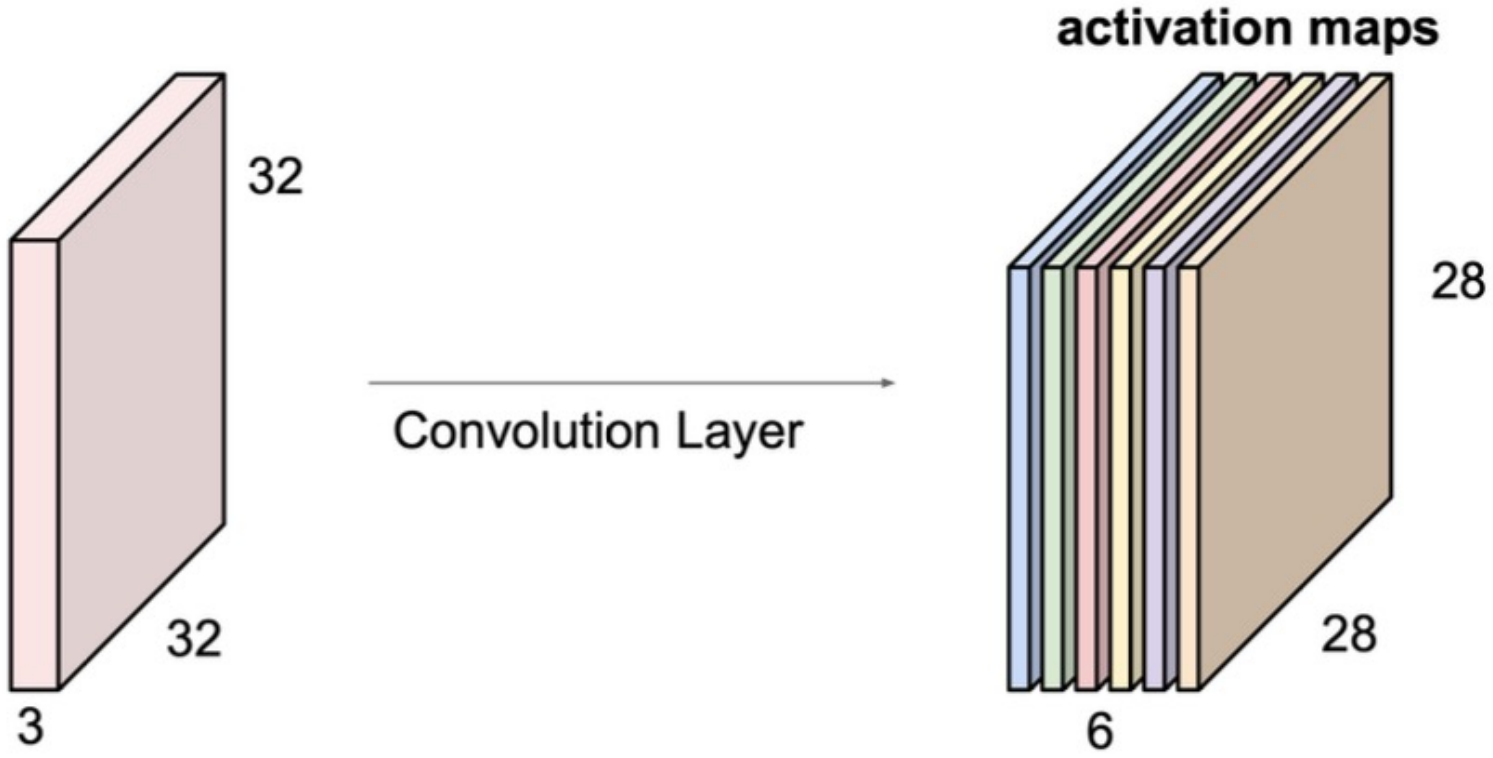
Convolution Layer

consider a second, green filter



CNN

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!

CNN

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

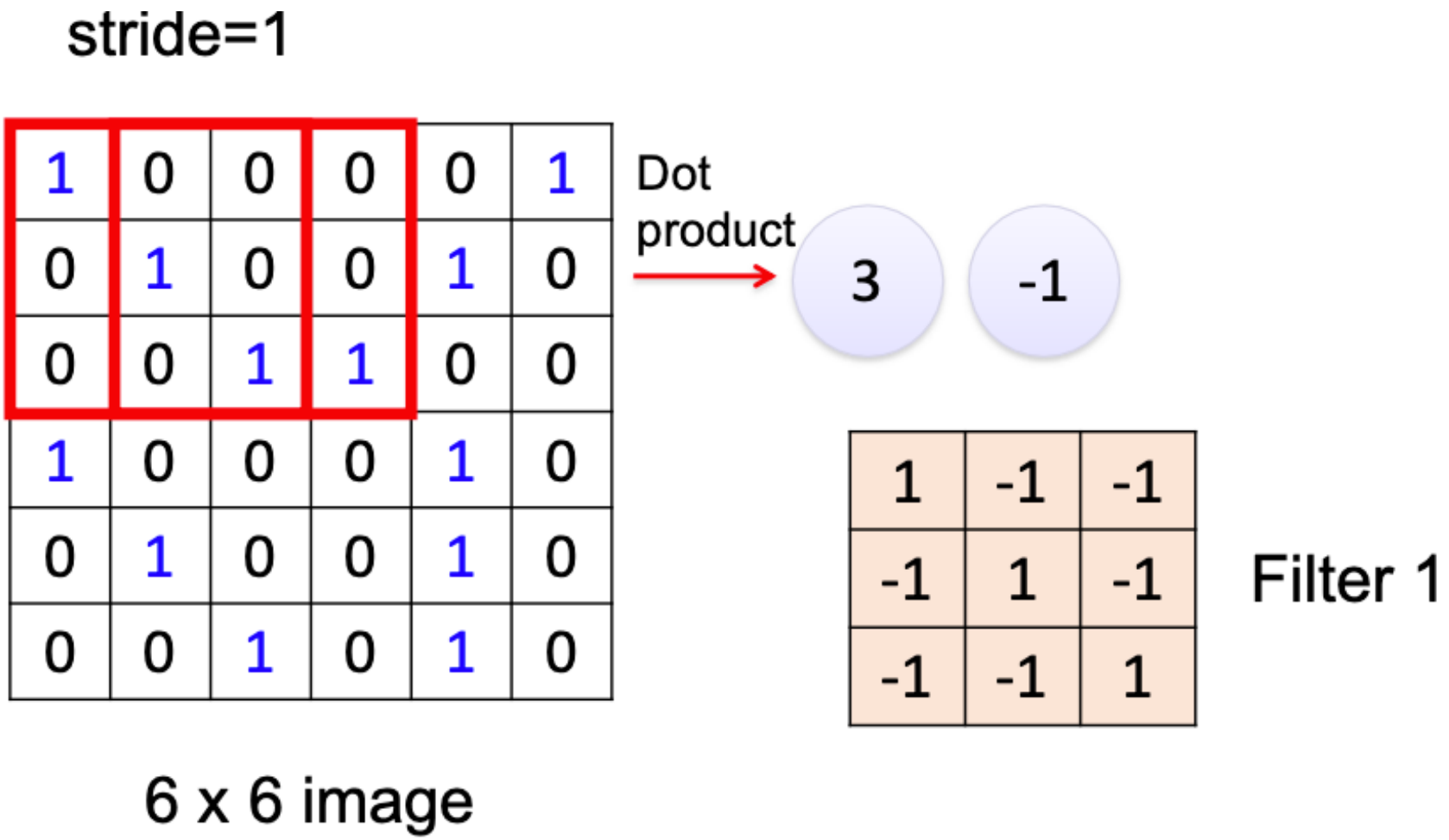
-1	1	-1
-1	1	-1
-1	1	-1

Filter 2

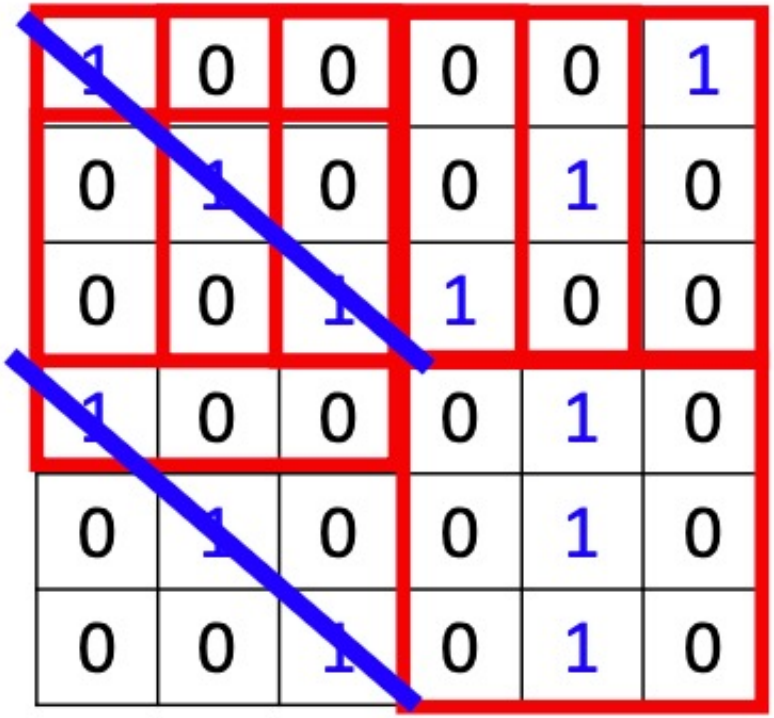
⋮ ⋮

Each filter detects a small pattern (3 x 3).

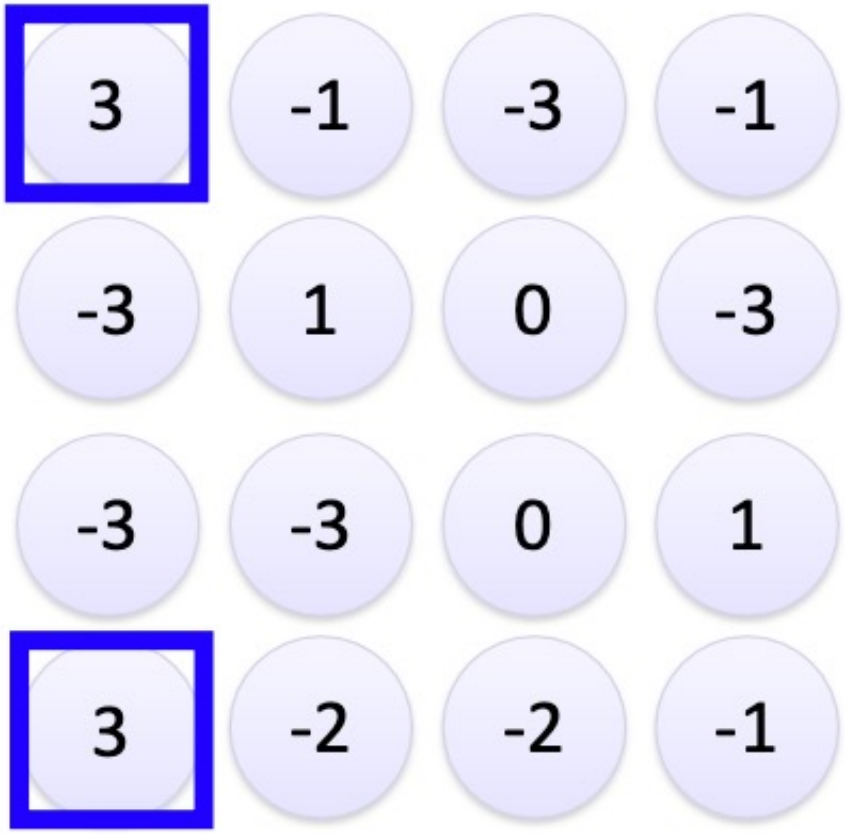
CNN


stride=1
1 0 0 0 0 1
0 1 0 0 1 0
0 0 1 1 0 0
1 0 0 0 1 0
0 1 0 0 1 0
0 0 1 0 1 0
6 x 6 image
Dot product
3
-1
1 -1 -1
-1 1 -1
-1 -1 1
Filter 1

CNN



6 x 6 image



CNN

Convolutions (typically with *prespecified* filters) are a common operation in many computer vision applications

Original image z

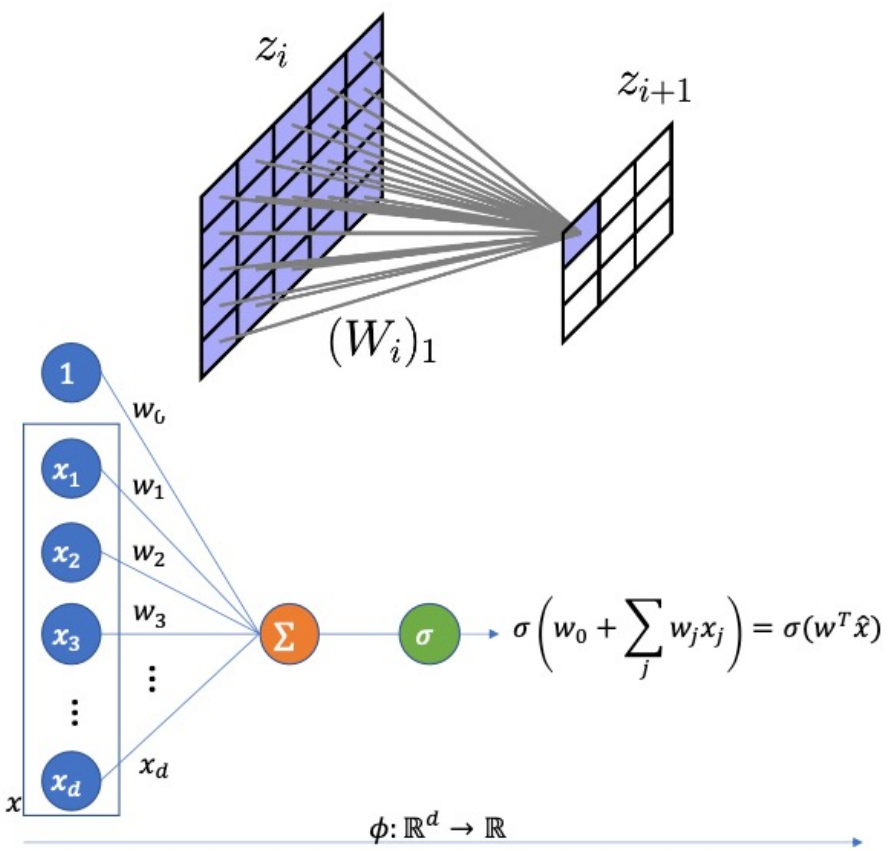
Gaussian blur

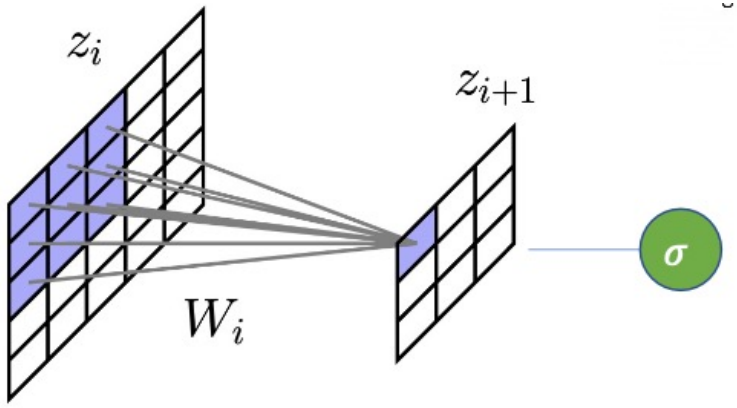
Image gradient

$$z * \begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 4 & 4 & 1 \end{bmatrix} /273$$

$$\left(\left(z * \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \right)^2 + \left(z * \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \right)^2 \right)^{\frac{1}{2}}$$

CNN





Convolution is a linear operator

We need to follow convolution with a nonlinearity (e.g., ReLU) to get nonlinear functions.

CNN

bird

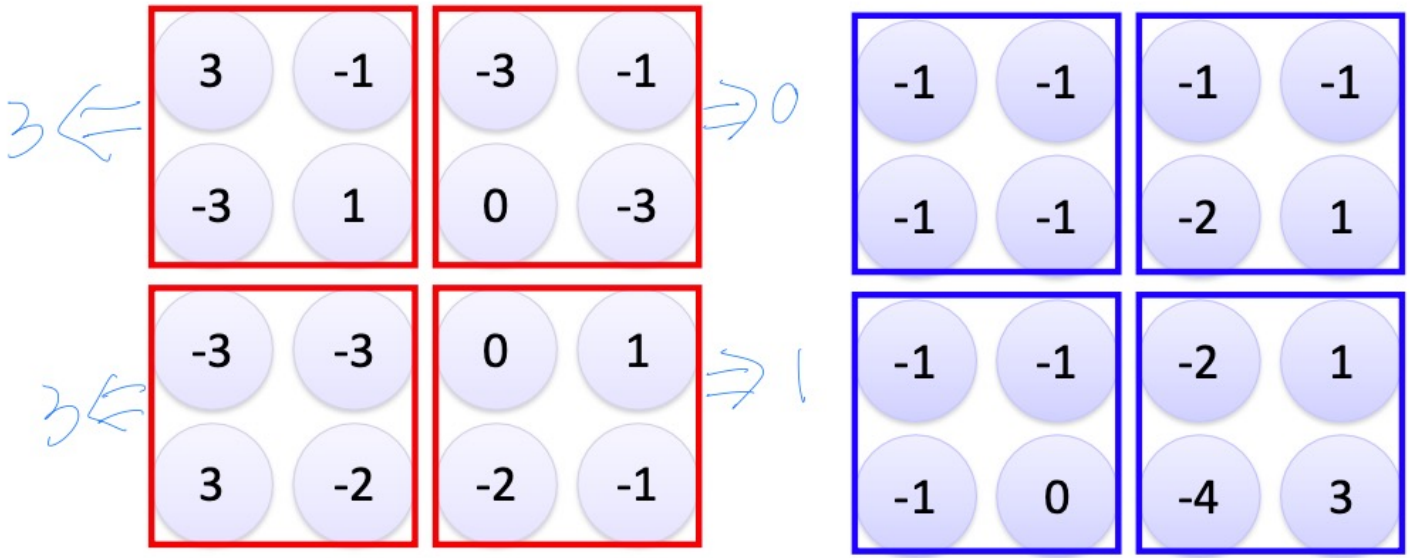
bird

1	-1	-1
-1	1	-1
-1	-1	1

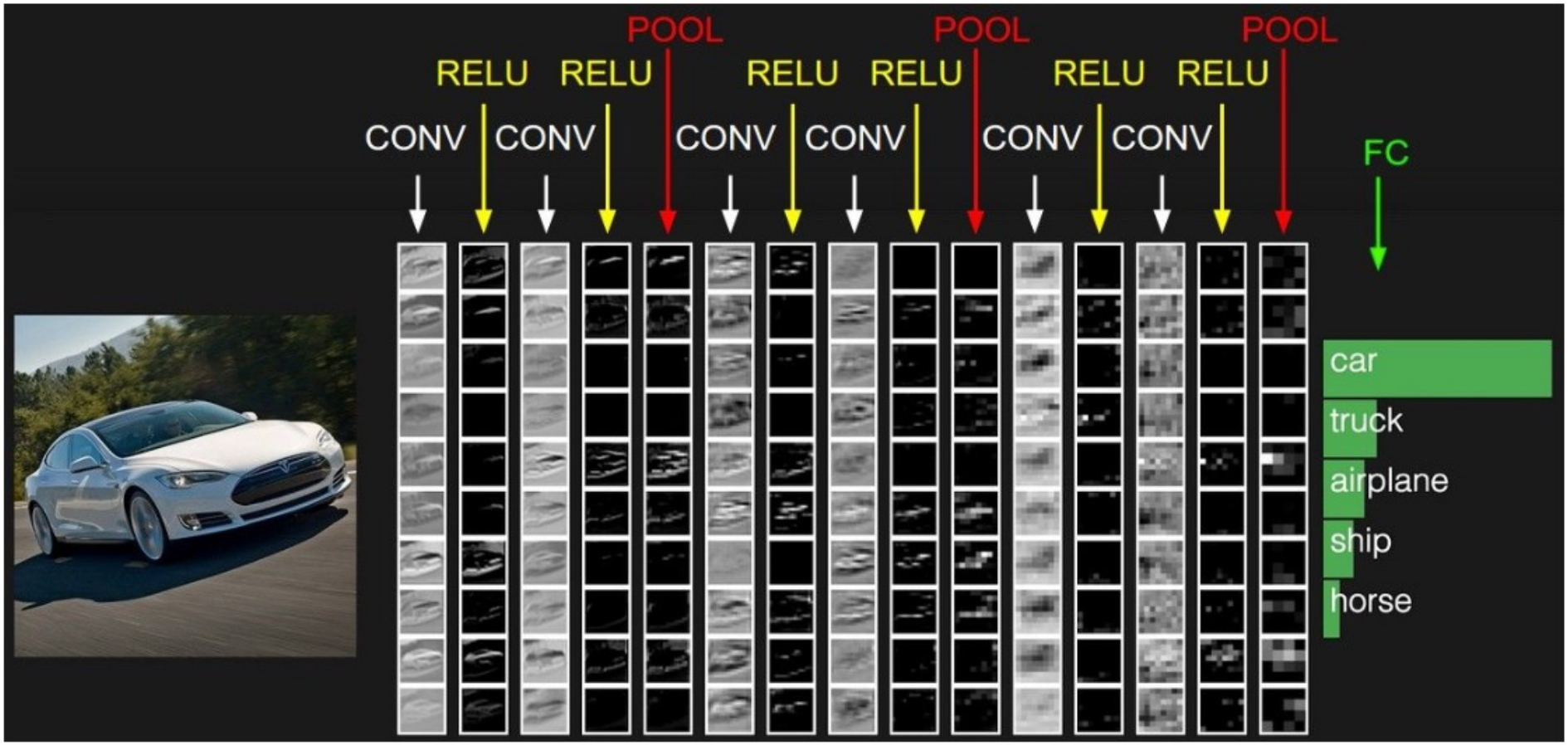
Filter 1

-1	1	-1
-1	1	-1
-1	1	-1

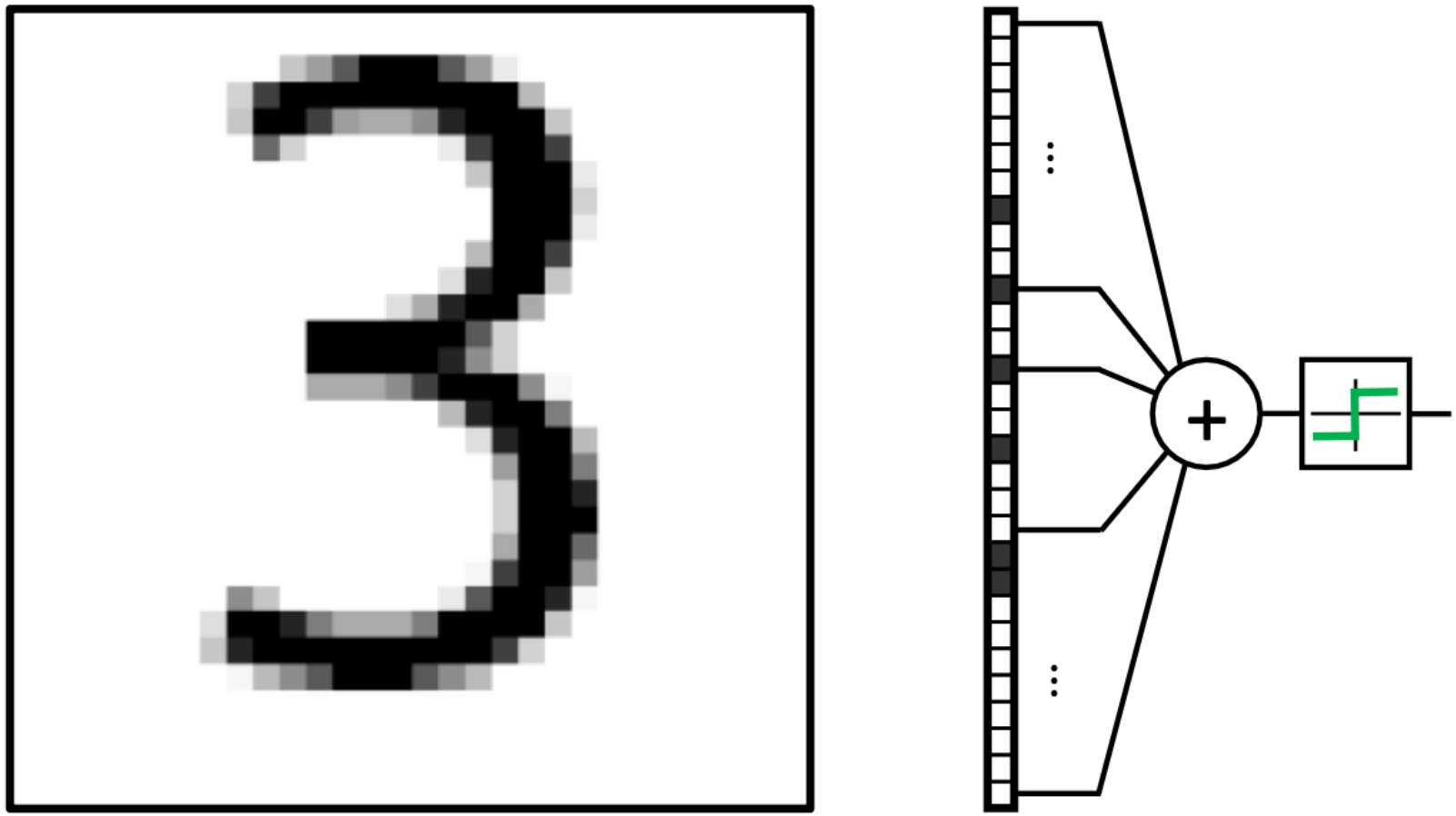
Filter 2



CNN


POOL
POOL
POOL
RELU
RELU
RELU
RELU
RELU
RELU
CONV
CONV
CONV
CONV
CONV
CONV
FC
car
truck
airplane
ship
horse

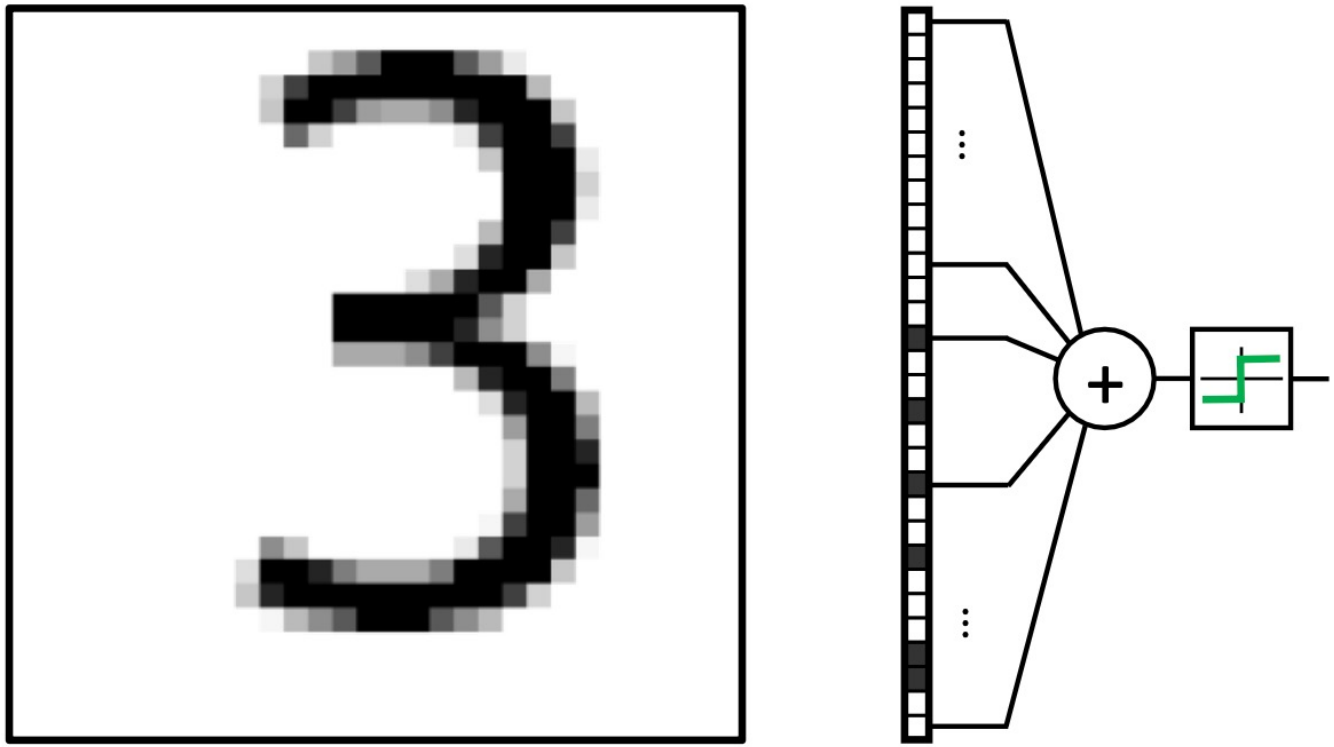
Equivariant and Invariant



input image

input vector

Equivariant and Invariant



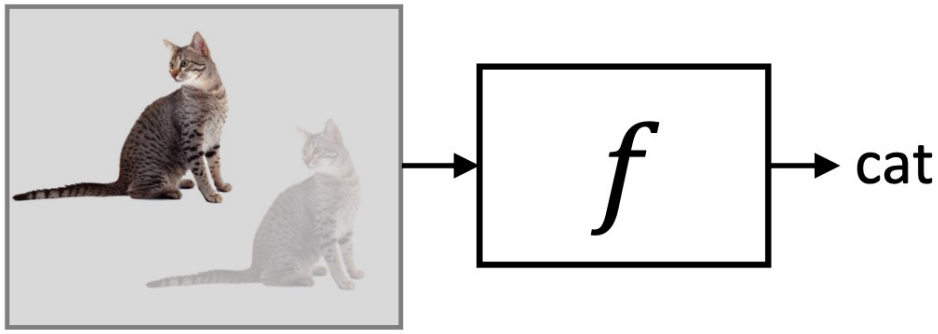
input image input vector

must learn shift invariance from data!

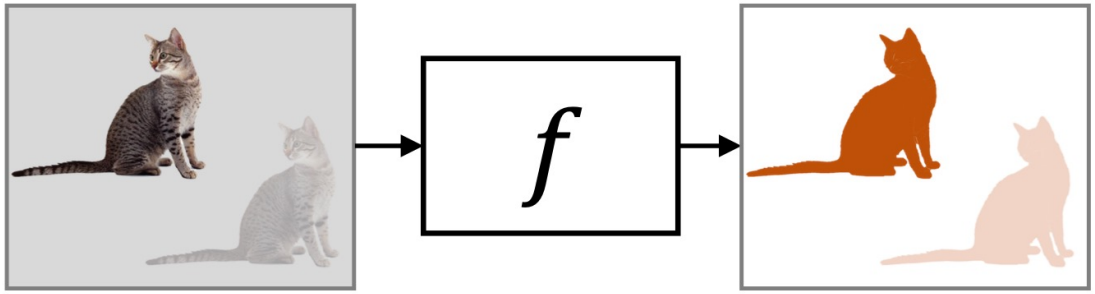
Code Demo

Equivariant and Invariant

$\mathfrak{G}$-invariance $f(\rho(\mathfrak{g})x) = f(x)$



$\mathfrak{G}$-equivariance $f(\rho(\mathfrak{g})x) = \rho(\mathfrak{g})f(x)$



Equivariant and Invariant

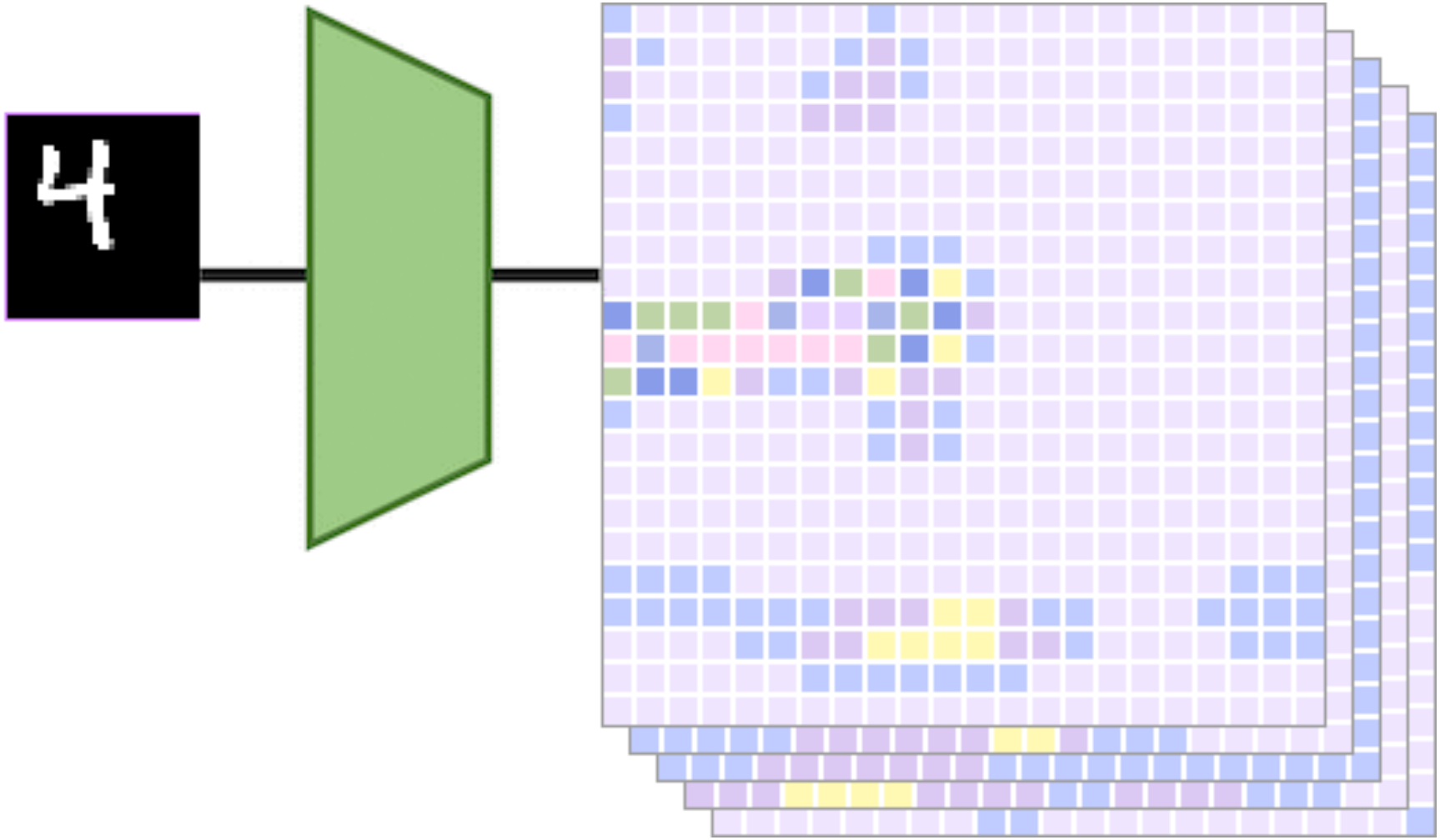
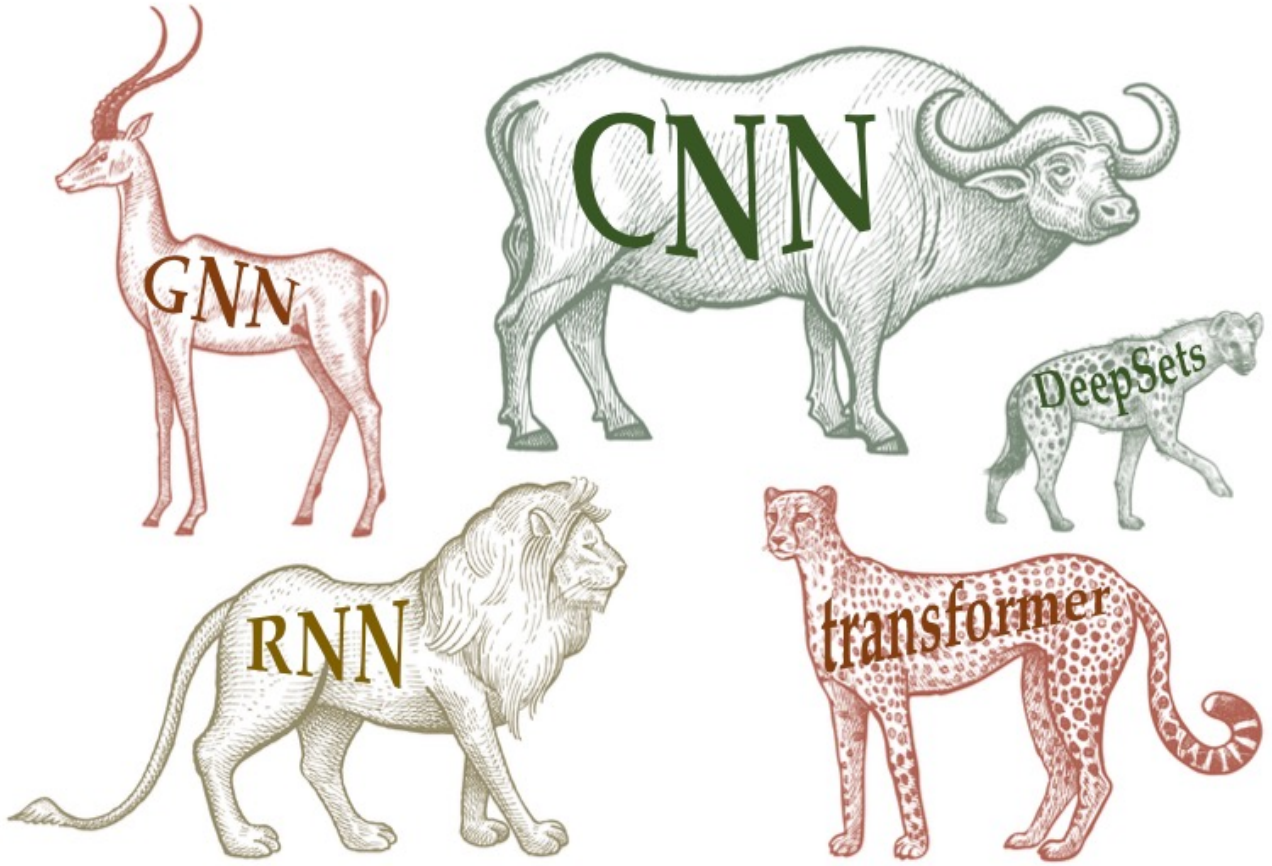


Image Convolution is ?

DNNs

Deep Neural Networks

20th Century Zoo of Neural Network Architectures

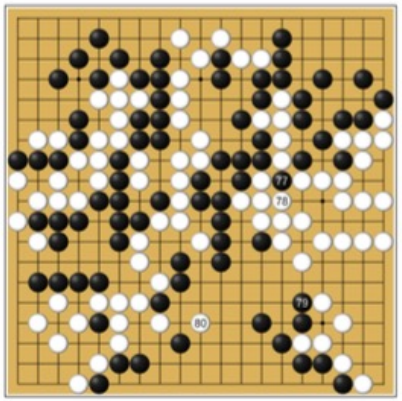


DNNs

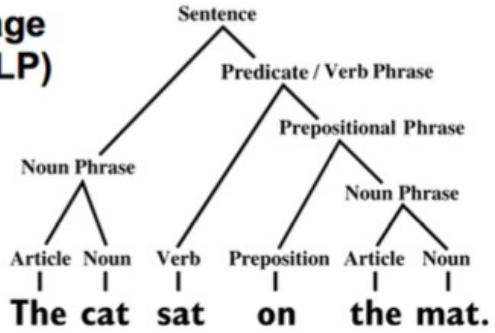
IMAGENET

Speech data

Grid games



Natural language processing (NLP)



...

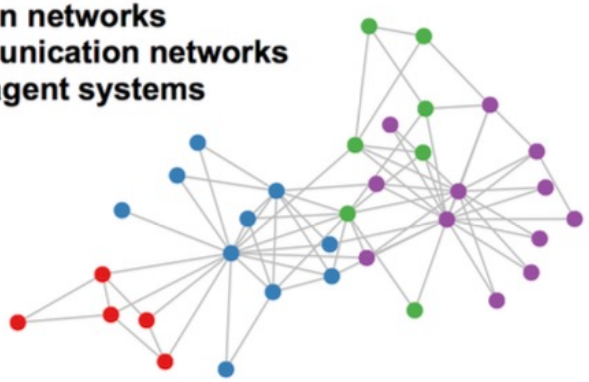
Deep neural nets that exploit:

- translation equivariance (weight sharing)
- hierarchical compositionality

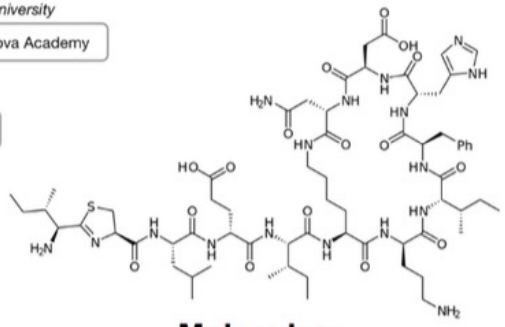
DNNs

A lot of real-world data does not “live” on grids

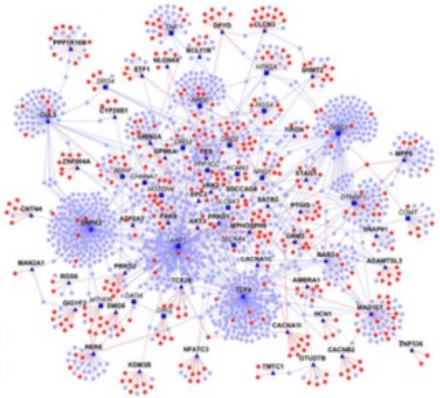
Social networks
Citation networks
Communication networks
Multi-agent systems



Knowledge graphs



Molecules



Protein interaction networks

Road maps

From CNN to GNN

* slide from Thomas Kipf, **University of Amsterdam**

Sequential Data

The teacher told the student that he was brilliant.

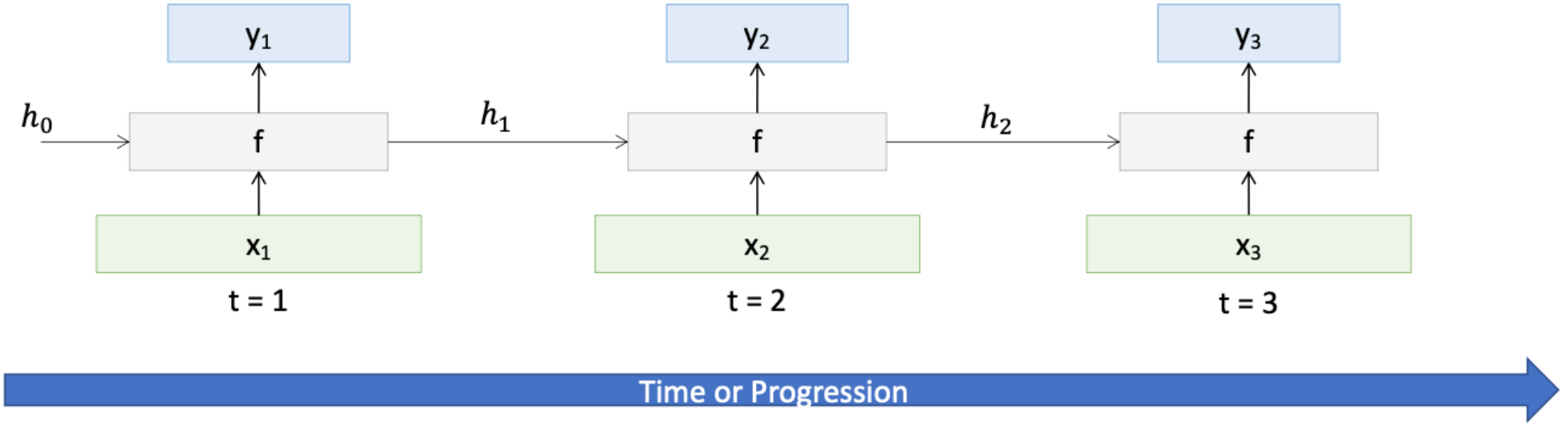
The student told the teacher that he was brilliant.

You read this sentence from left to right and

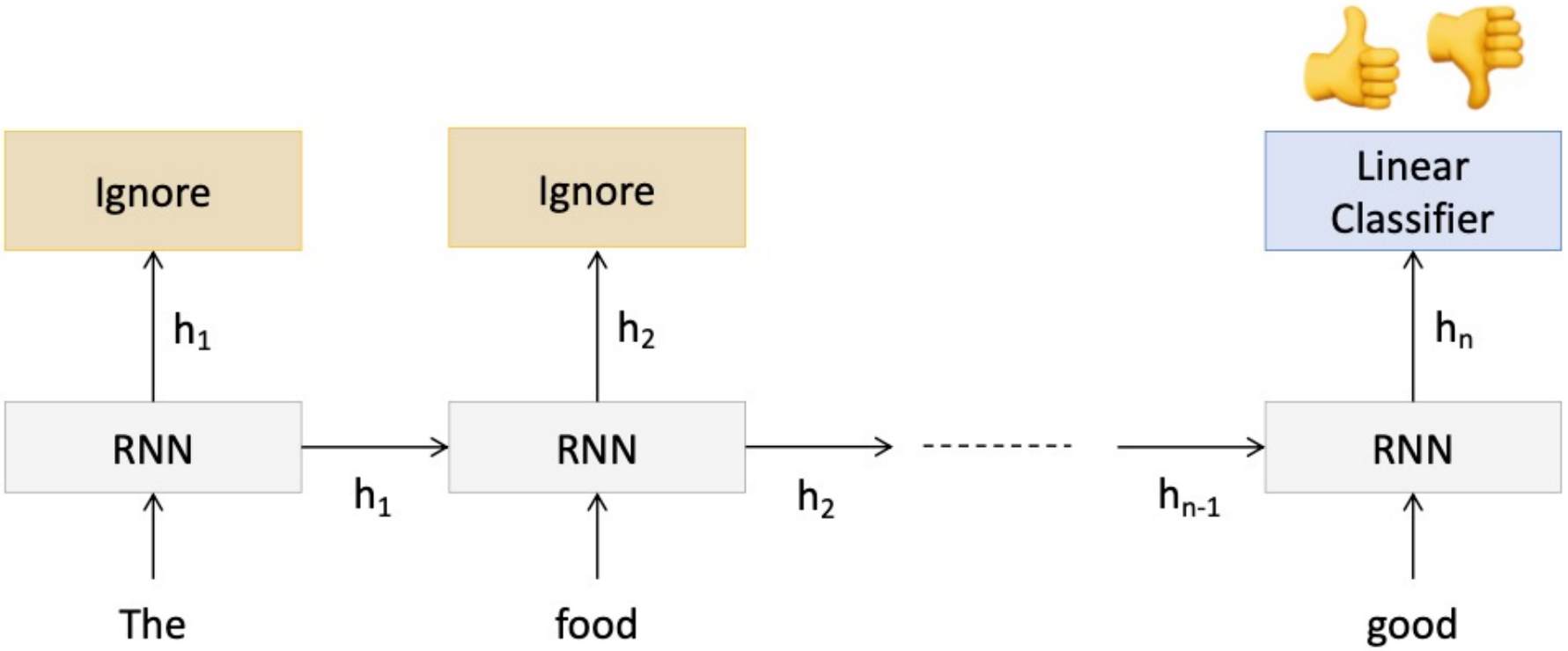
understand the sentence

Sequential Data - RNN

What if we have sequences of variable lengths, like sentences or videos that we would like to analyze?

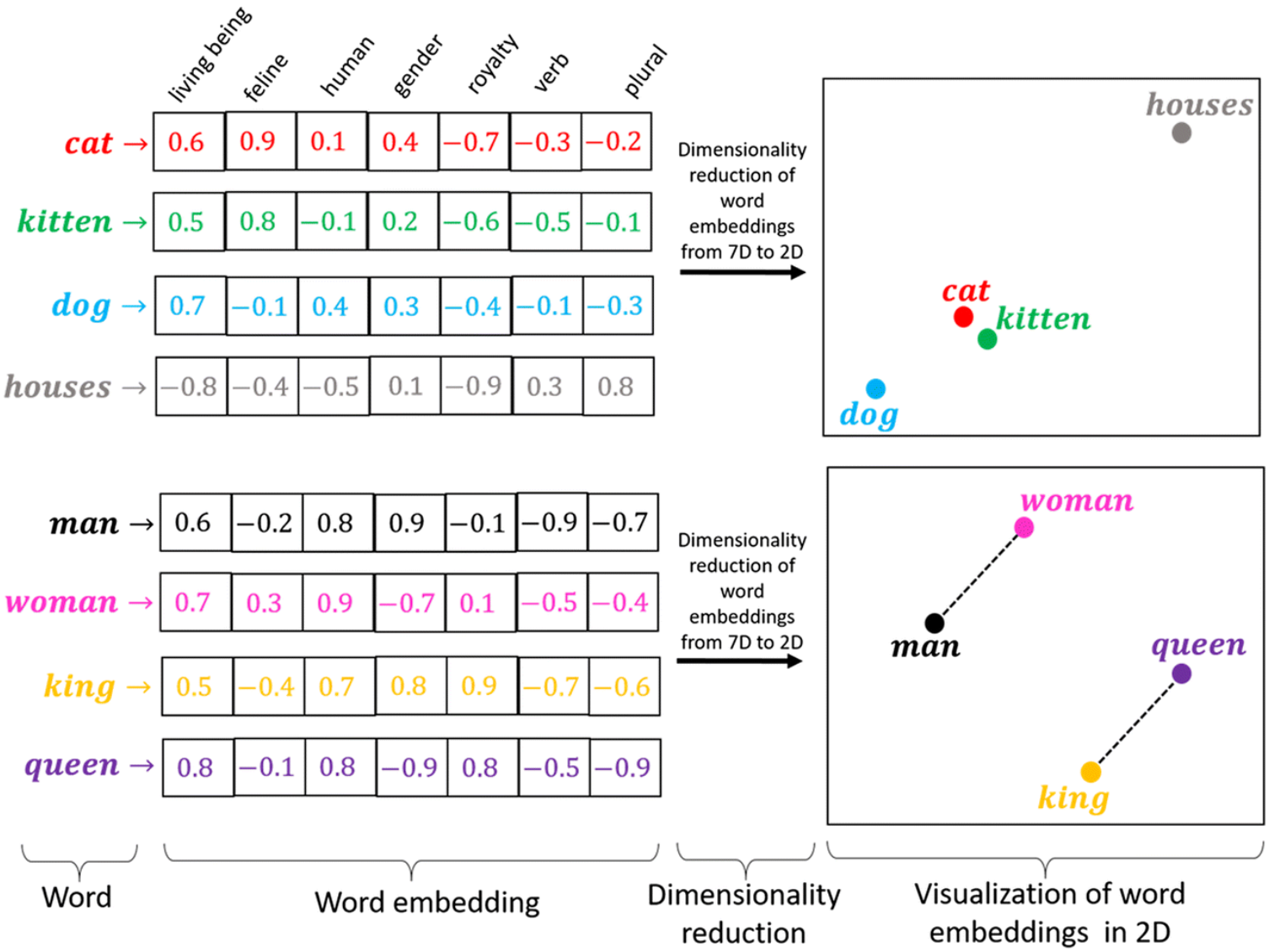


Sequential Data - RNN

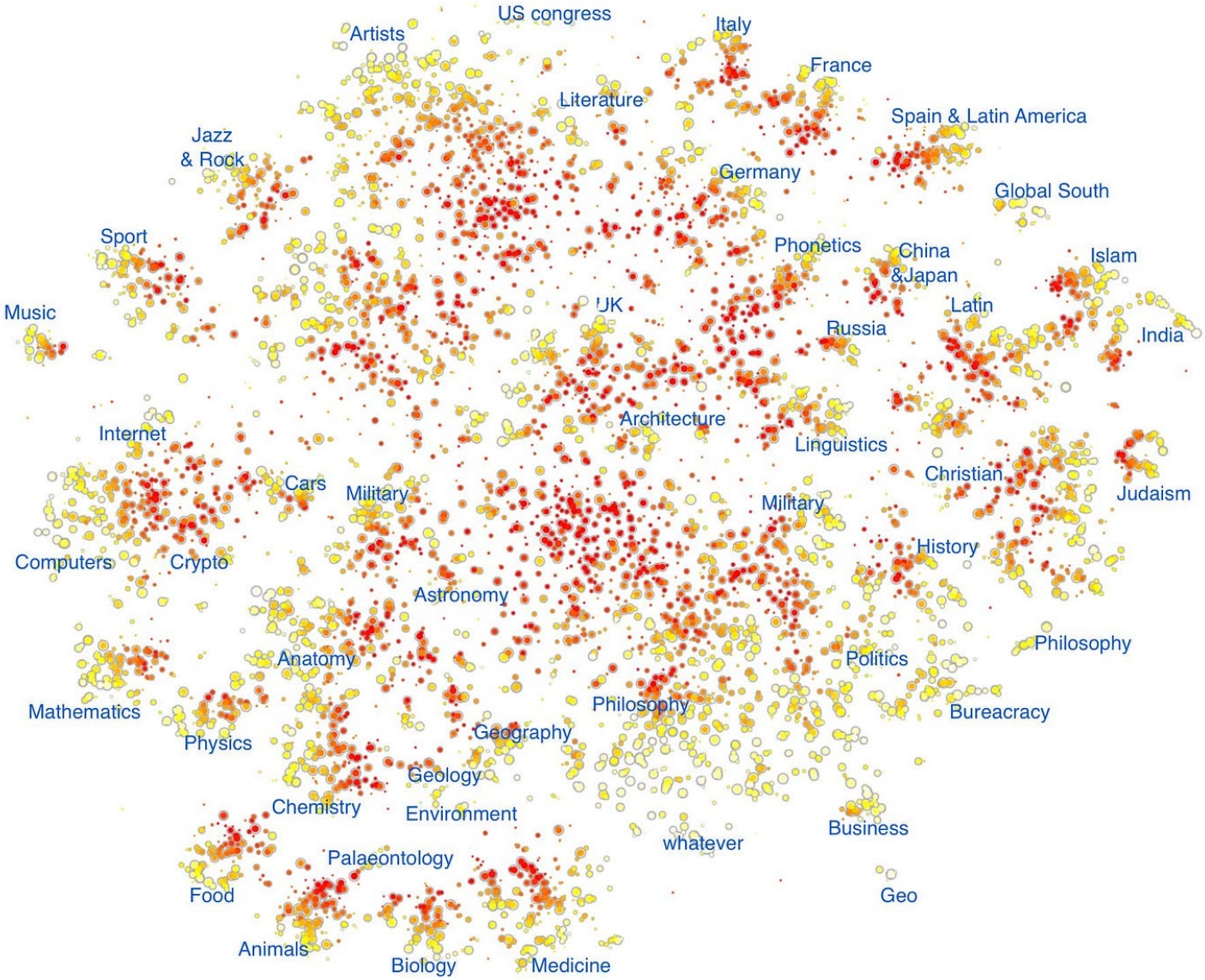


Token embedding –
DNA of the Token

Sequential Data - RNN

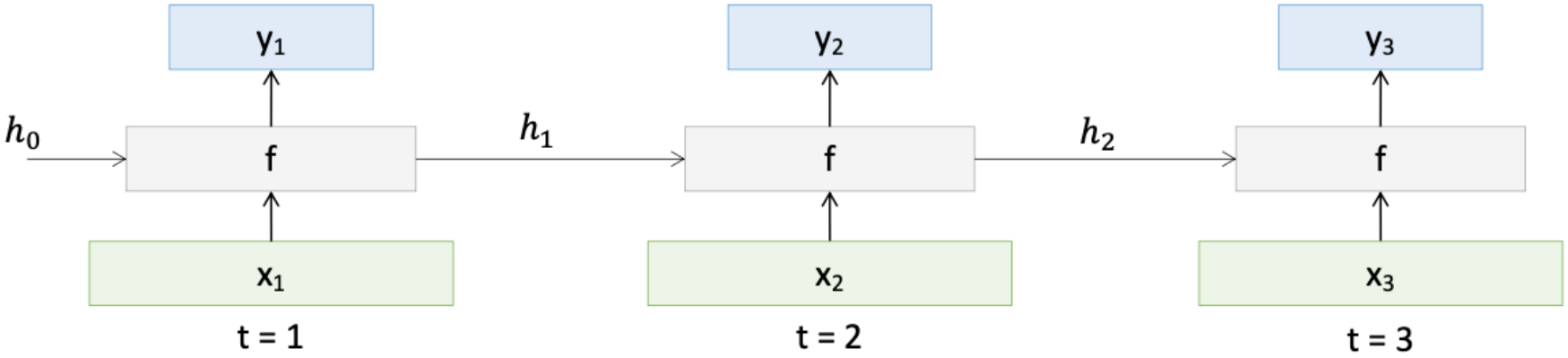


Sequential Data - RNN

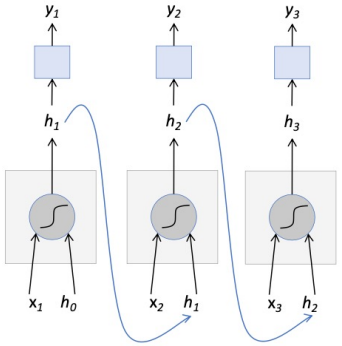


Sequential Data - RNN

What if we have sequences of variable lengths, like sentences or videos that we would like to analyze?

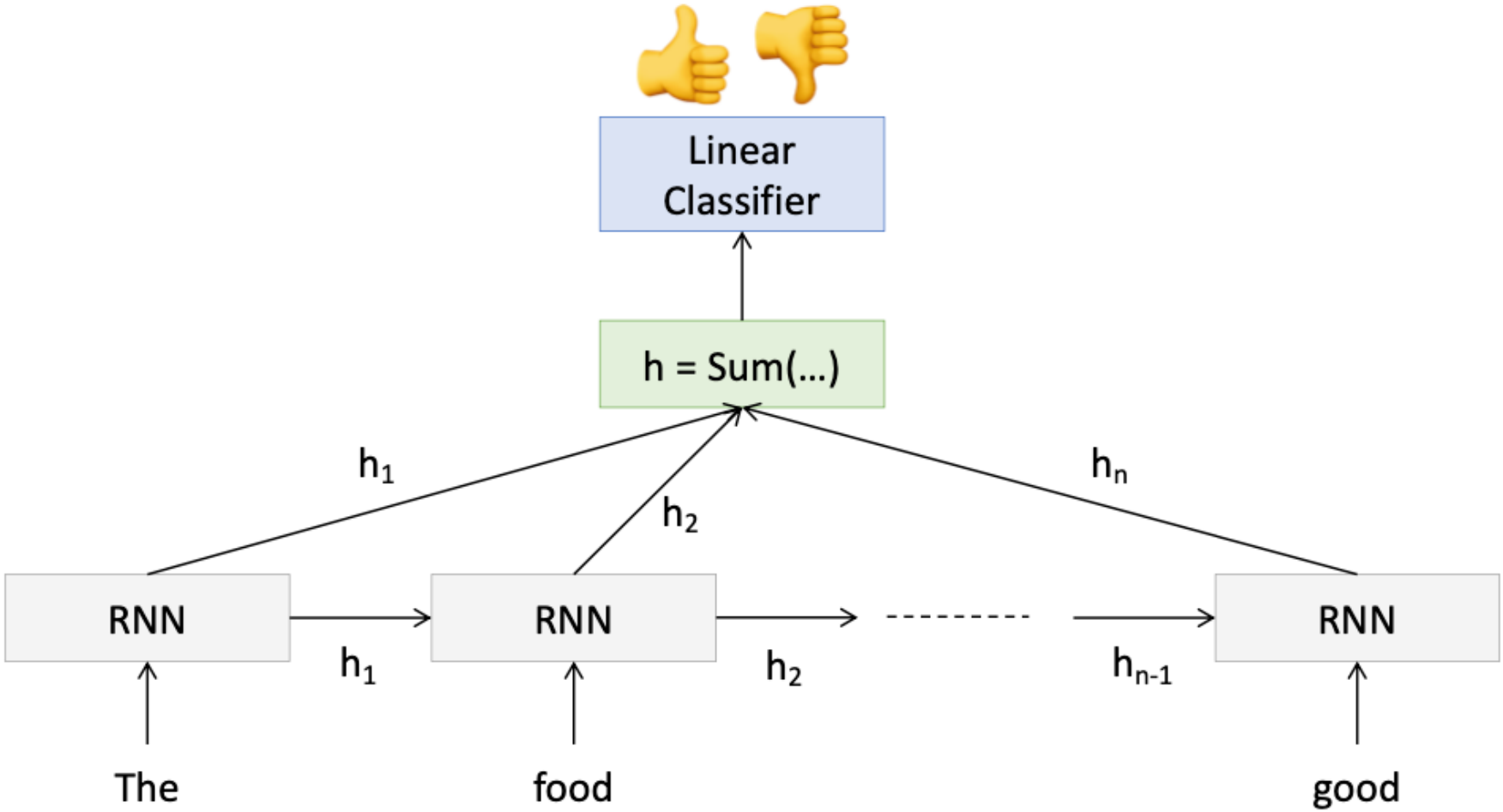


How would you setup the model (token embedding, MLP layer)



- $h_t = \sigma(W^T \begin{bmatrix} x_t \\ h_{t-1} \end{bmatrix})$
- $y_t = F(h_t)$

Sequential Data - RNN

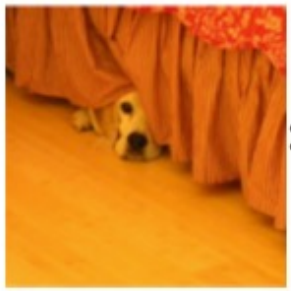


Sequential Data - RNN

Image Captioning

- Given an image, produce a sentence describing its contents

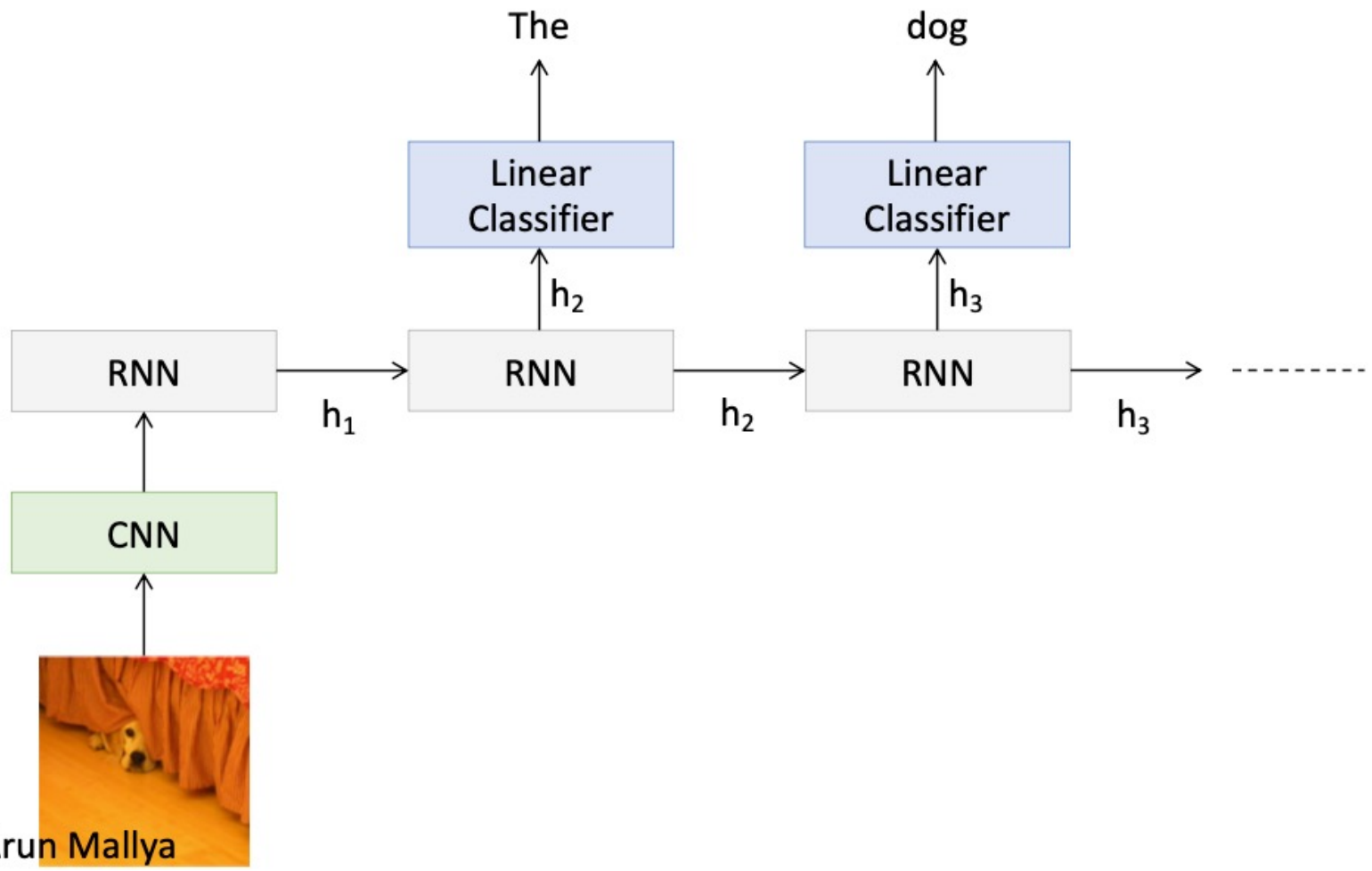
- Inputs: Image feature (from a CNN)
- Outputs: Multiple words (let's consider one sentence)



: The dog is hiding

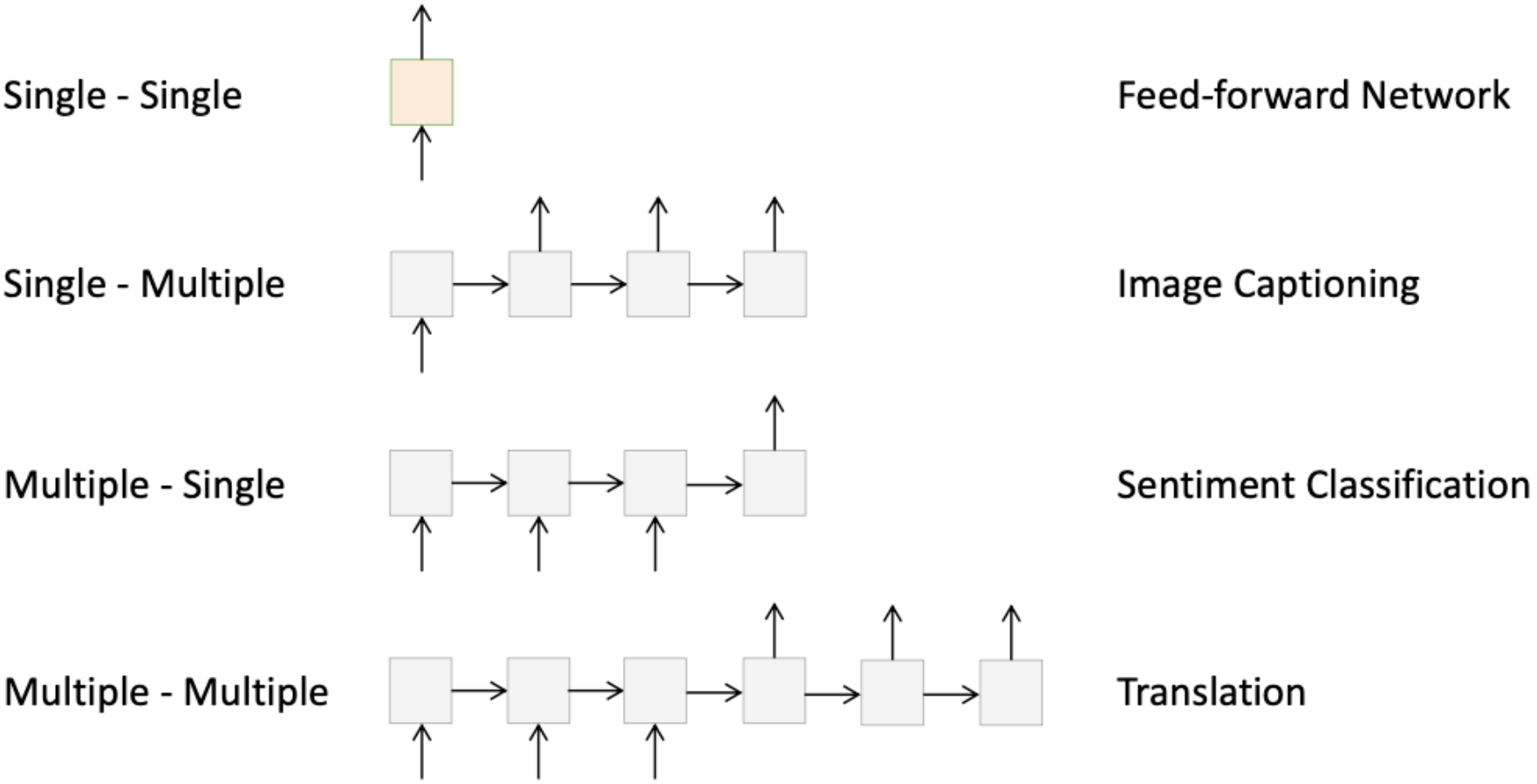
Sequential Data - RNN

Image Captioning

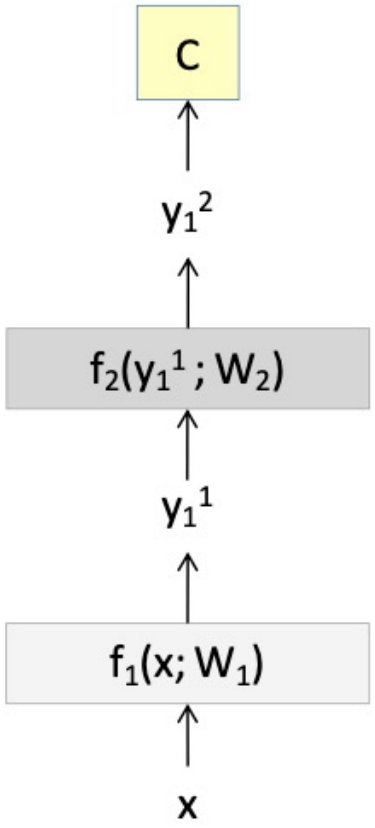


Sequential Data - RNN

Input – Output Scenarios



Sequential Data - RNN



$$y_1 = f_1(x; W_1)$$

$$y_2 = f_2(y_1; W_2)$$

$$C = \text{Loss}(y_2, y_{GT})$$

$$\text{Find } \frac{\partial C}{\partial W_1}, \frac{\partial C}{\partial W_2}$$

$$\frac{\partial C}{\partial W_2} = \left(\frac{\partial C}{\partial y_2}\right)\left(\frac{\partial y_2}{\partial W_2}\right)$$

$$\frac{\partial C}{\partial W_1} = \left(\frac{\partial C}{\partial y_1}\right)\left(\frac{\partial y_1}{\partial W_1}\right)$$

$$= \left(\frac{\partial C}{\partial y_2}\right)\left(\frac{\partial y_2}{\partial y_1}\right)\left(\frac{\partial y_1}{\partial W_1}\right)$$

Sequential Data - RNN

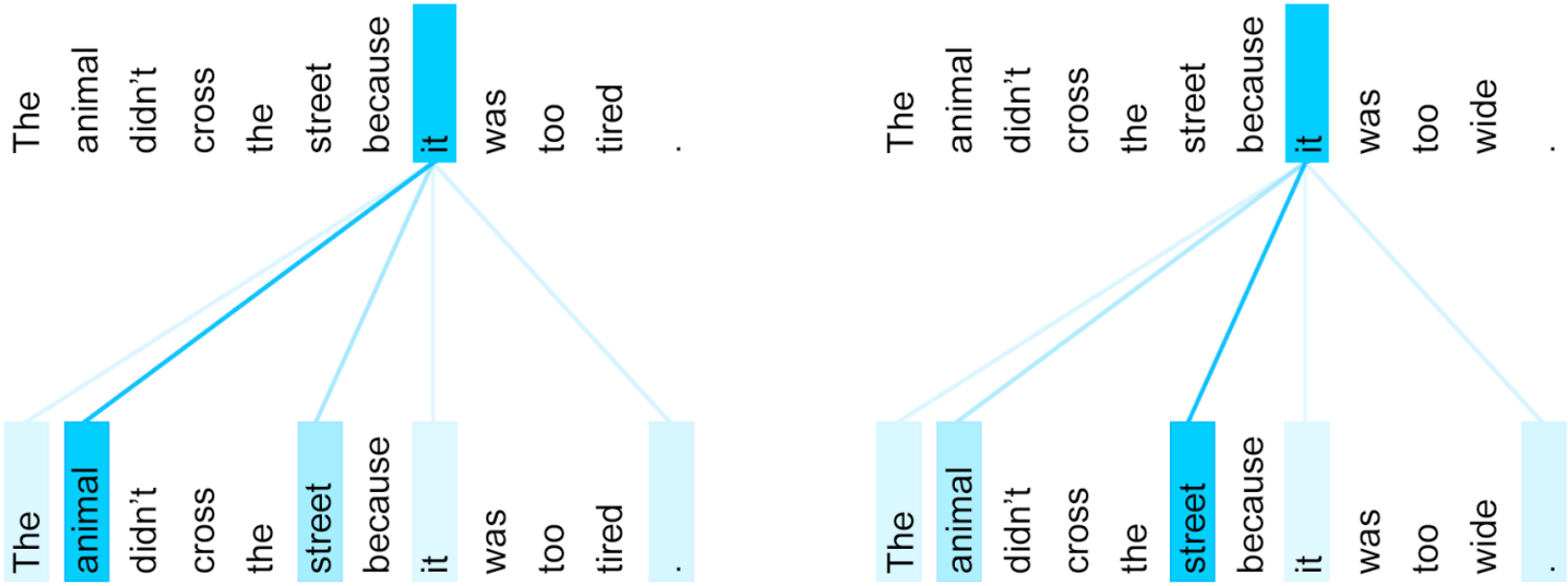
$$h_t = f(W h_{t-1} + b)$$

$$\|\frac{\partial L}{\partial h_0}\| \approx \sigma_{\max}^T \|\frac{\partial L}{\partial h_T}\|$$

- Largest singular value > 1 → **Exploding gradients**
  - Slight error in the late time steps causes drastic updates in the early time steps → Unstable learning
- Largest singular value < 1 → **Vanishing gradients**
  - Gradients passed to the early time steps is close to 0. → Uninformed correction

Any other problem with RNN?

Sequential Data - Transformer



Key **Value** **Query**

Sequential Data - Transformer

Think of YouTube.

First you enter your ***Query*** *in the search bar. Then your* ***Query*** *is compared against a set of* ***Keys*** *(in this case, video titles, tags and descriptions etc. within the YouTube database). After this, YouTube proceeds to retrieve the videos that best match your* ***Query****. These video results are referred to as* ***Values****.*

Key **Value** **Query**

I would like to taste the local food in France.

Sequential Data - Transformer

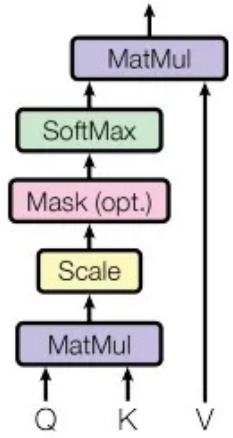
I would like to taste the local food in France.

$$Q = XW_Q \qquad K = XW_K \qquad V = XW_V$$

$$Q \cdot K = QK^T =$$

	I	would	like	to	taste	the	local	food	in	France
I	h									
would		h								
like			h							
to				h						
taste					h			h		
the						h				
local							h			h
food					h			h		
in									h	
France							h			h

$$softmax\left(\frac{Q_i K^T}{\sqrt{d_k}}\right) = \frac{e^{\left(\frac{Q_i K^T}{\sqrt{d_k}}\right)}}{\sum_{j=1}^{d_k} e^{\left(\frac{Q_j K^T}{\sqrt{d_k}}\right)}}$$



Sequential Data - Transformer

I would like to taste the local food in France.

$$Q = XW_Q \qquad K = XW_K \qquad V = XW_V$$

$$softmax\left(\frac{QK^T}{\sqrt{d_k}}\right) =$$

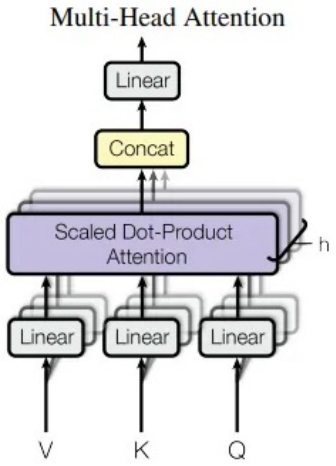
Attention Weight Matrix

	I	would	like	to	taste	the	local	food	in	France	
I								0.02			
would								0.02			
like								0.05			
to								0.01			
taste								0.2			
the								0.01			
local								0.2			
food	0.02	0.02	0.05	0.01	0.2	0.01	0.2	0.1	0.09	0.3	= 1
in								0.09			
France								0.3			
								= 1			

Attention

↓

Multi-Attention

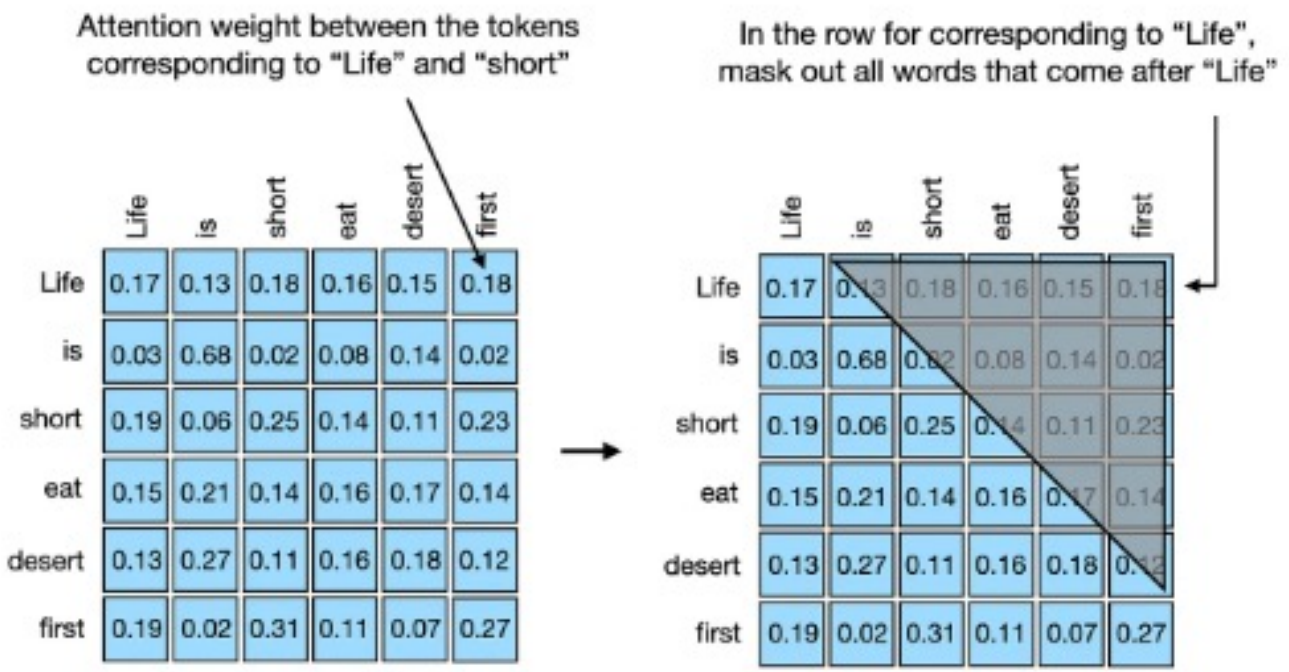


Sequential Data - Transformer

I would like to taste the local food in France.

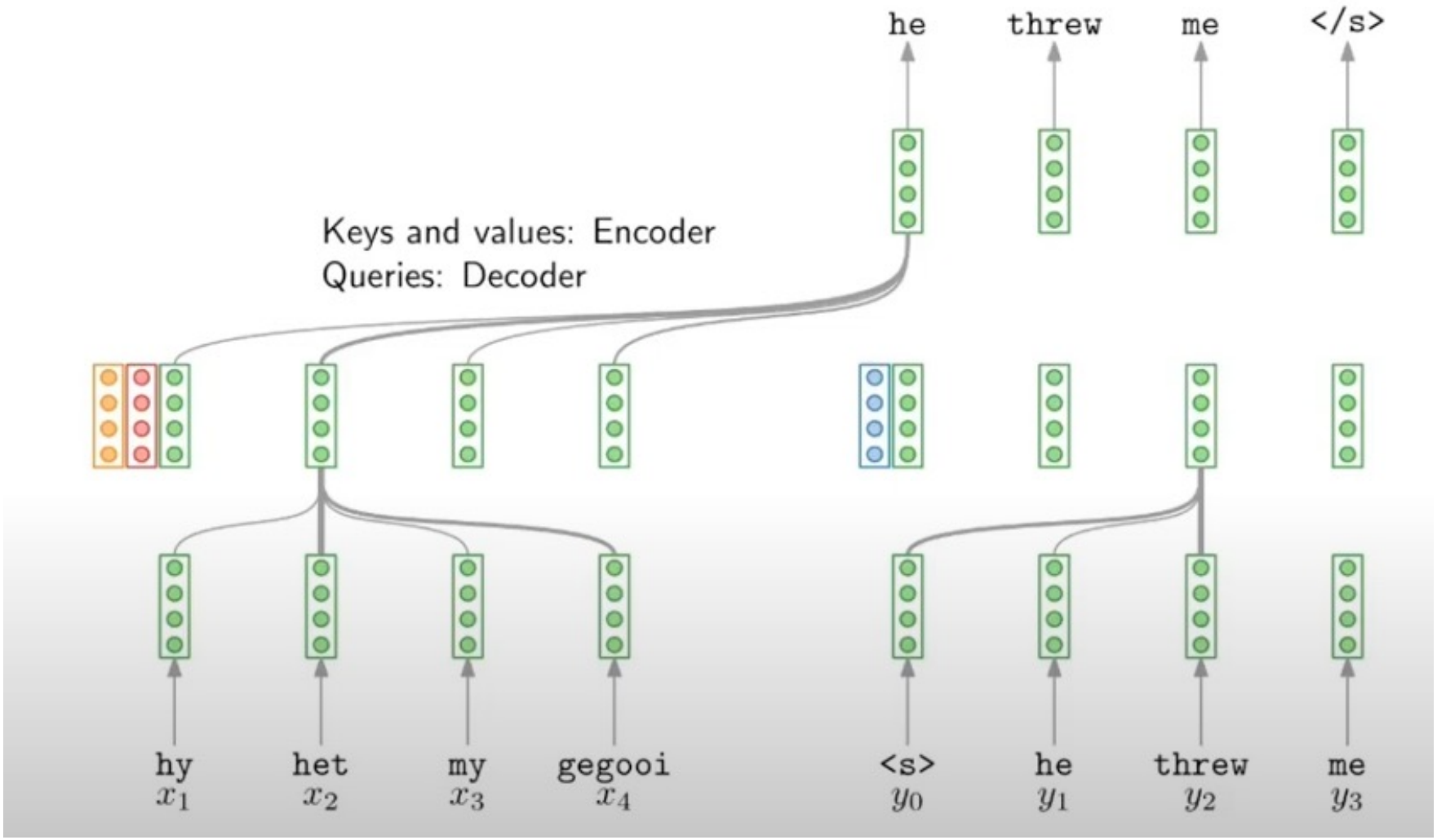
$$Q = XW_Q \qquad K = XW_K \qquad V = XW_V$$

Self-Attention **Causal-Attention**



Sequential Data - Transformer

Cross-attention



Sequential Data - Transformer

Multi-Attention

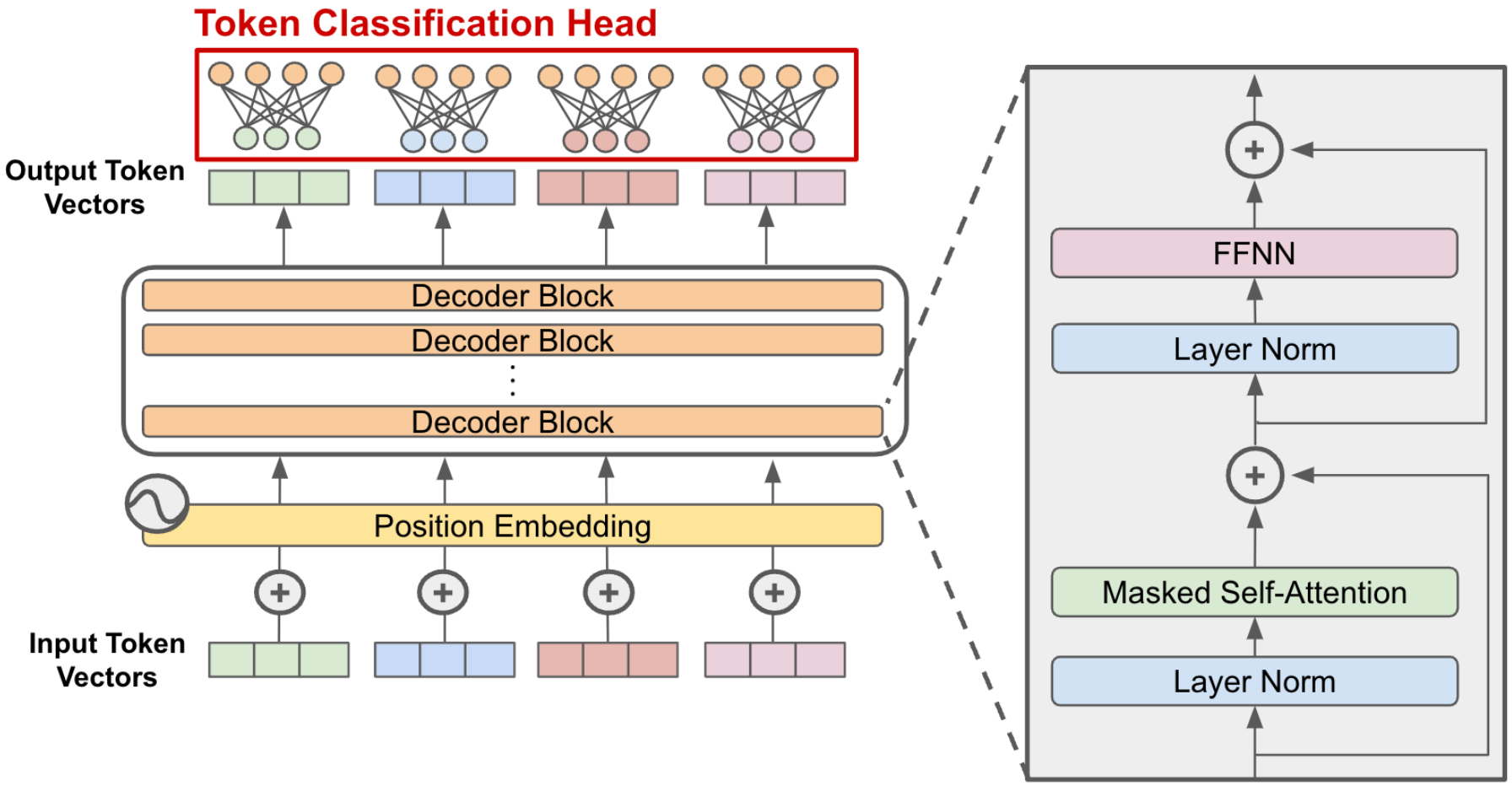
Self-Attention

Causal-Attention

Cross-Attention

Sentiment Analysis	Next-token Prediction	Translation

Sequential Data - Transformer



Sequential Data - Transformer

The teacher told the student that he was brilliant.

The student told the teacher that he was brilliant.

Two sentences using exactly the same tokens but end up with very different meaning

Sequential Data - Transformer

0:	0	0	0	0	8:	1	0	0	0
1:	0	0	0	1	9:	1	0	0	1
2:	0	0	1	0	10:	1	0	1	0
3:	0	0	1	1	11:	1	0	1	1
4:	0	1	0	0	12:	1	1	0	0
5:	0	1	0	1	13:	1	1	0	1
6:	0	1	1	0	14:	1	1	1	0
7:	0	1	1	1	15:	1	1	1	1

Sequential Data - Transformer

$$P(k, 2i) = \sin\left(\frac{k}{n^{2i/d}}\right)$$

$$P(k, 2i+1) = \cos\left(\frac{k}{n^{2i/d}}\right)$$

k: Position of an object in the input sequence, $0 \leq k < L/2$

d: Dimension of the output embedding space

Positional Encoding Matrix with d=4, n=100

Sequence	Index of token, k	i=0	i=0	i=1	i=1
I	0	P_{00}=sin(0) = 0	P_{01}=cos(0) = 1	P_{02}=sin(0) = 0	P_{03}=cos(0) = 1
am	1	P_{10}=sin(1/1) = 0.84	P_{11}=cos(1/1) = 0.54	P_{12}=sin(1/10) = 0.10	P_{13}=cos(1/10) = 1.0
a	2	P_{20}=sin(2/1) = 0.91	P_{21}=cos(2/1) = -0.42	P_{22}=sin(2/10) = 0.20	P_{23}=cos(2/10) = 0.98
Robot	3	P_{30}=sin(3/1) = 0.14	P_{31}=cos(3/1) = -0.99	P_{32}=sin(3/10) = 0.30	P_{33}=cos(3/10) = 0.96