Privacy & Security in Machine Learning / Optimization

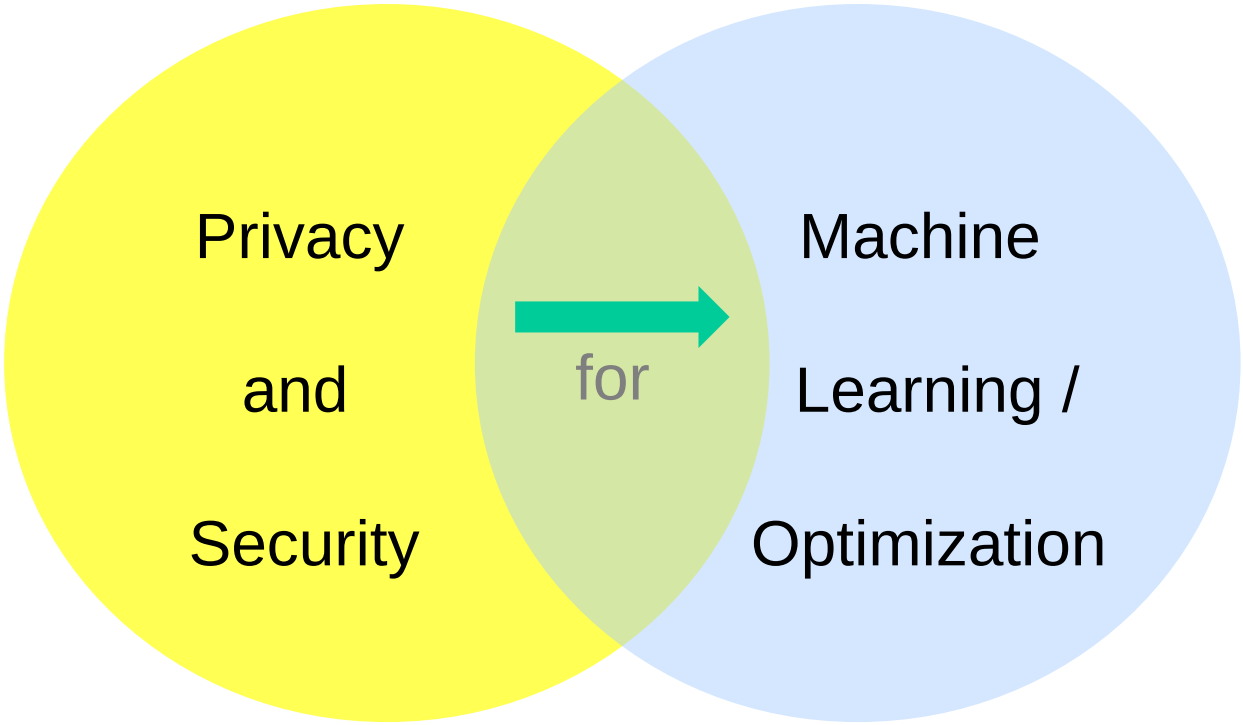
Nitin Vaidya
University of Illinois at Urbana-Champaign

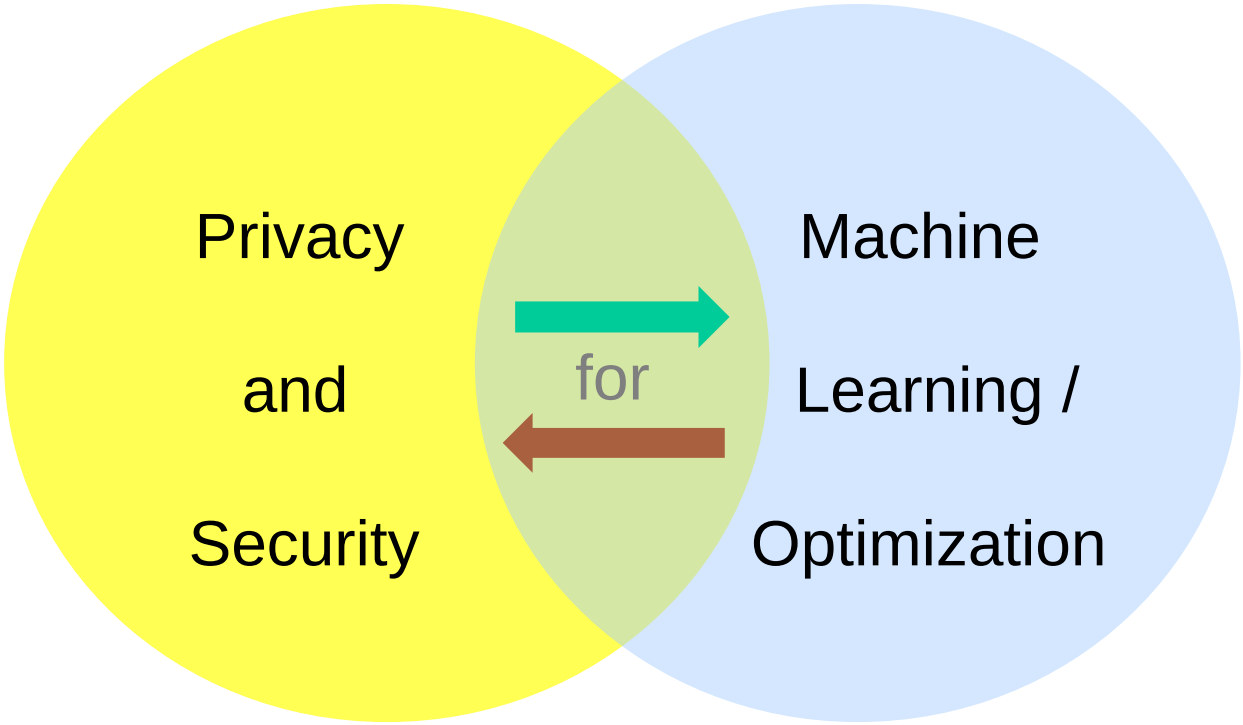
disc.ece.illinois.edu

Research Interests

Distributed algorithms

- Distributed shared memory systems
- Distributed computations over wireless networks
- Distributed optimization

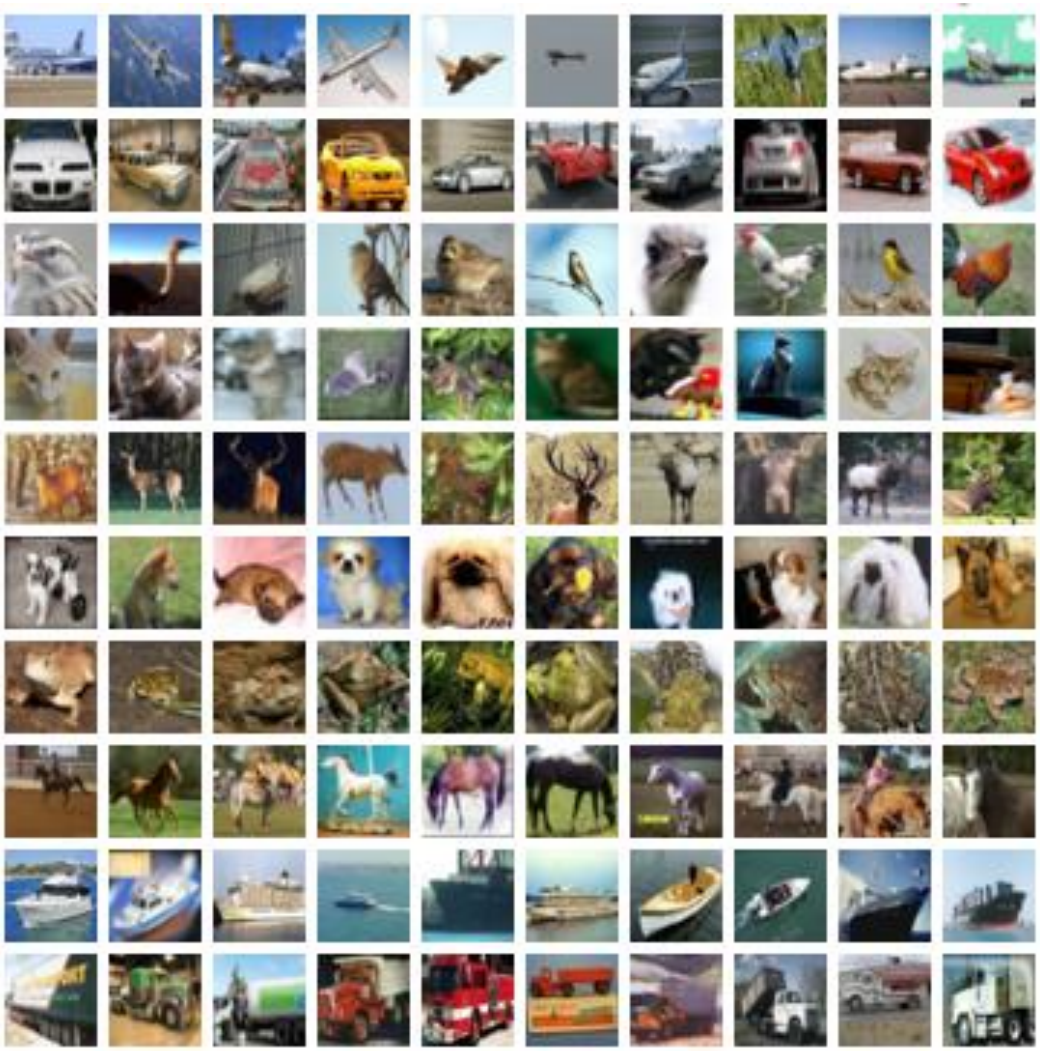

Privacy
and
Security
for
Machine
Learning /
Optimization


Privacy
and
Security
for
Machine
Learning /
Optimization

Outline

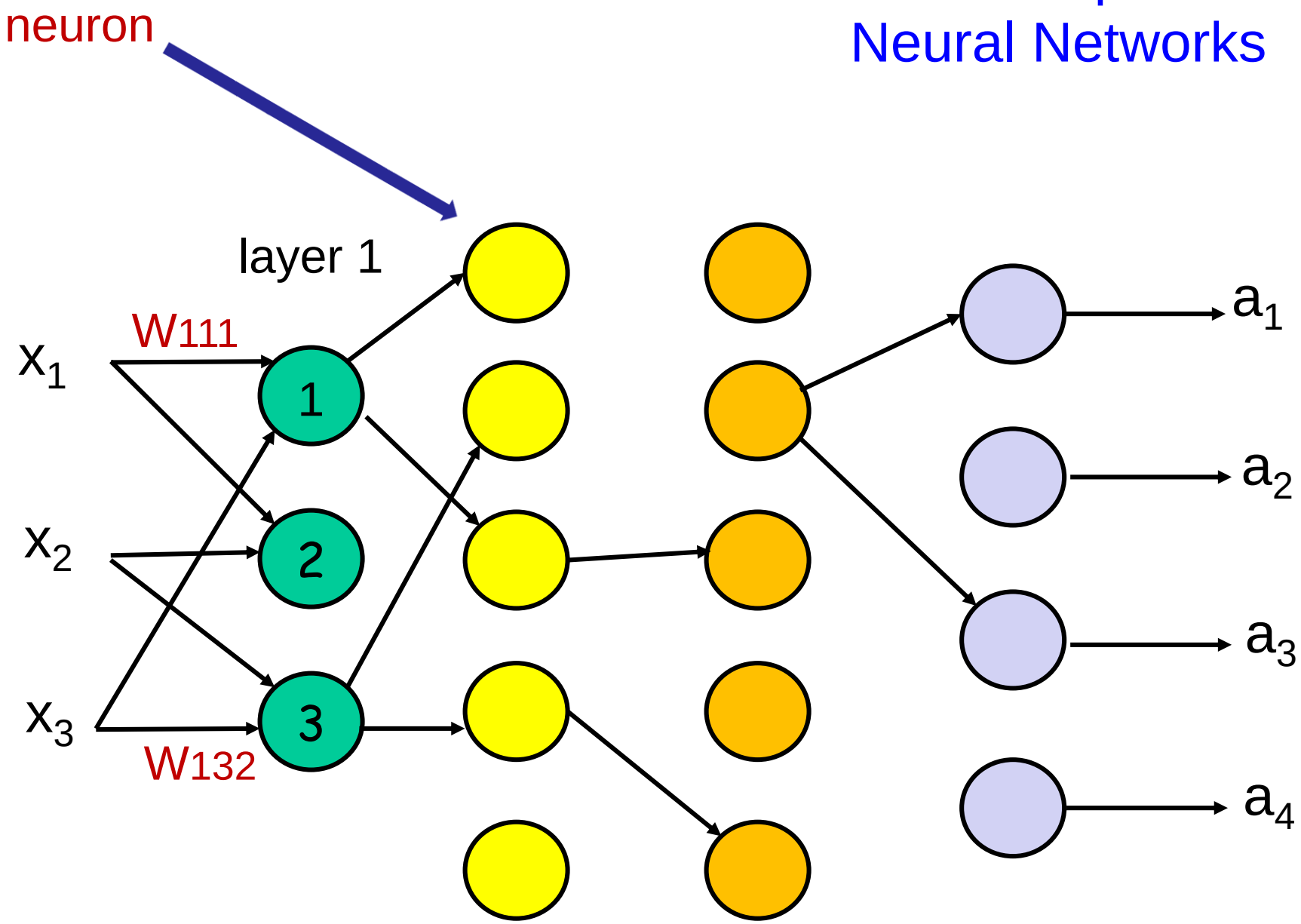
- Motivation – distributed machine learning
- Research problems
  - Privacy-preserving distributed optimization
  - Adversarial learning
  - Robustness to adversarial samples

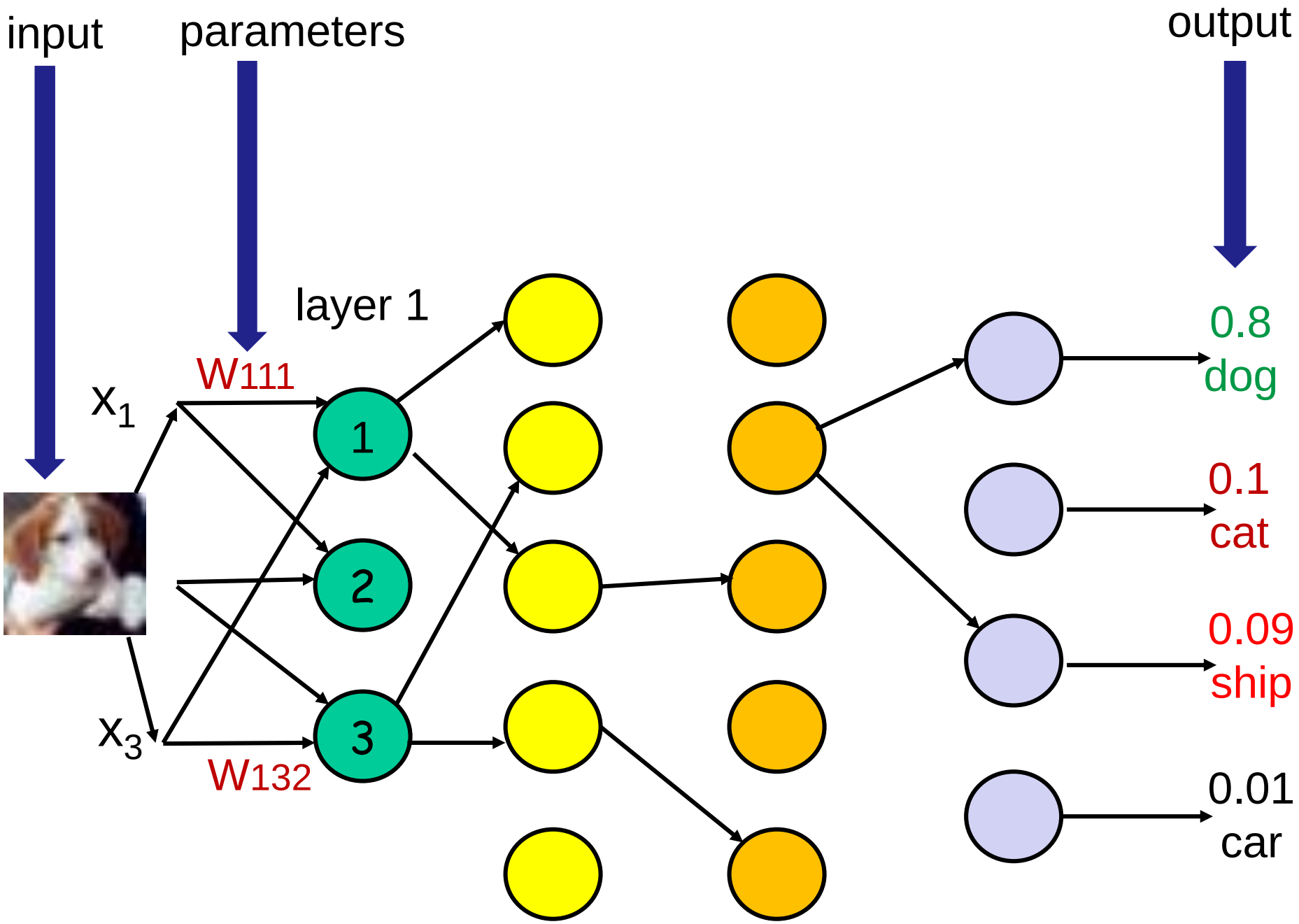
Example – Image Classification



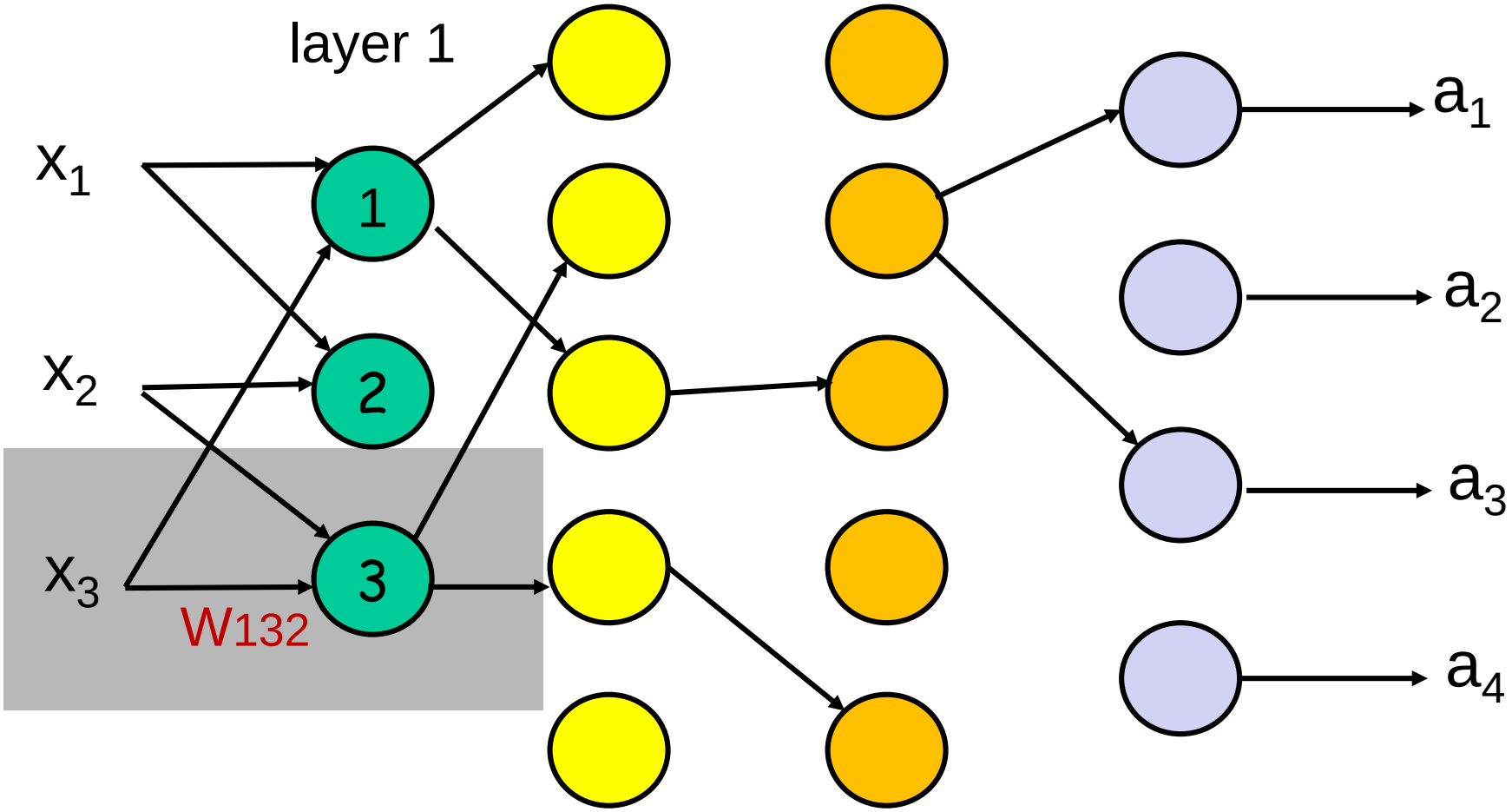
CIFAR-10 dataset

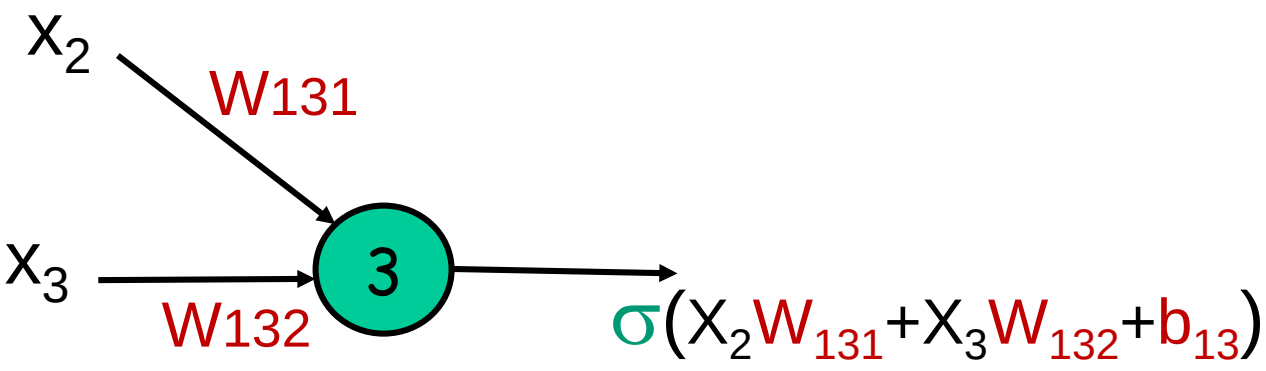
Deep Neural Networks

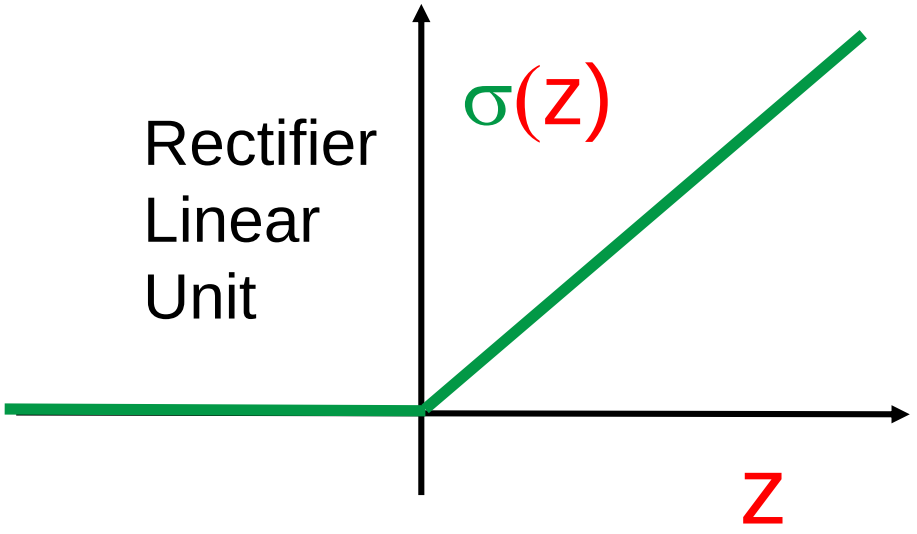


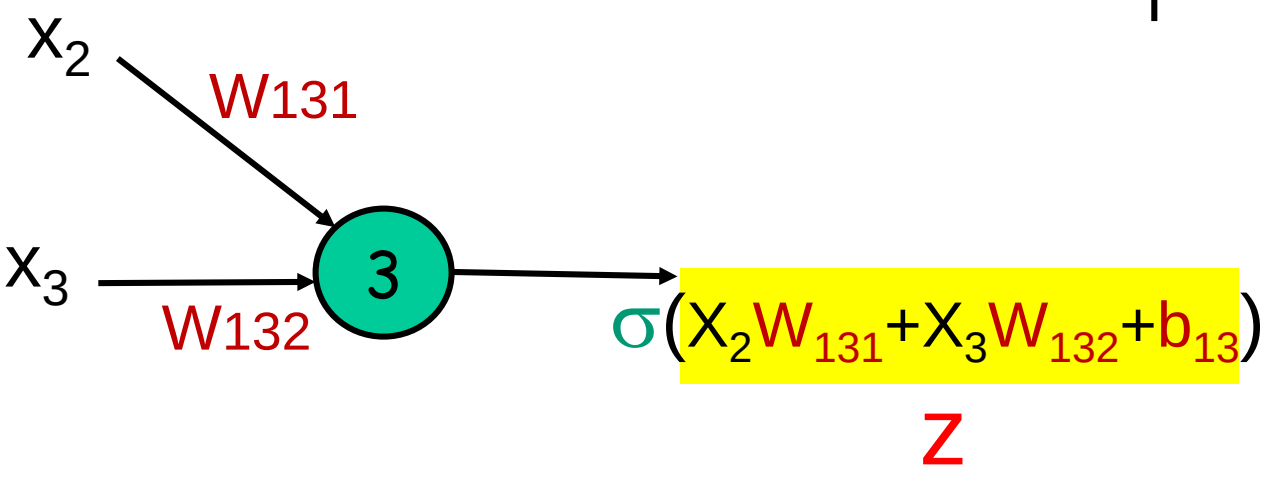

input
parameters
output
layer 1
W111
x_1
x_3
1
2
3
W132
0.8
dog
0.1
cat
0.09
ship
0.01
car

Deep Neural Networks




x_2
W_{131}
x_3
3
W_{132}
$\sigma(X_2W_{131}+X_3W_{132}+b_{13})$


Rectifier
Linear
Unit
σ(z)
z


x2
W131
x3
3
W132
σ(X2W131+X3W132+b13)
z

How to train your ~~dragon~~ network

- Given a machine structure
- Parameters are the only free variables

➔ Choose parameters that maximize accuracy

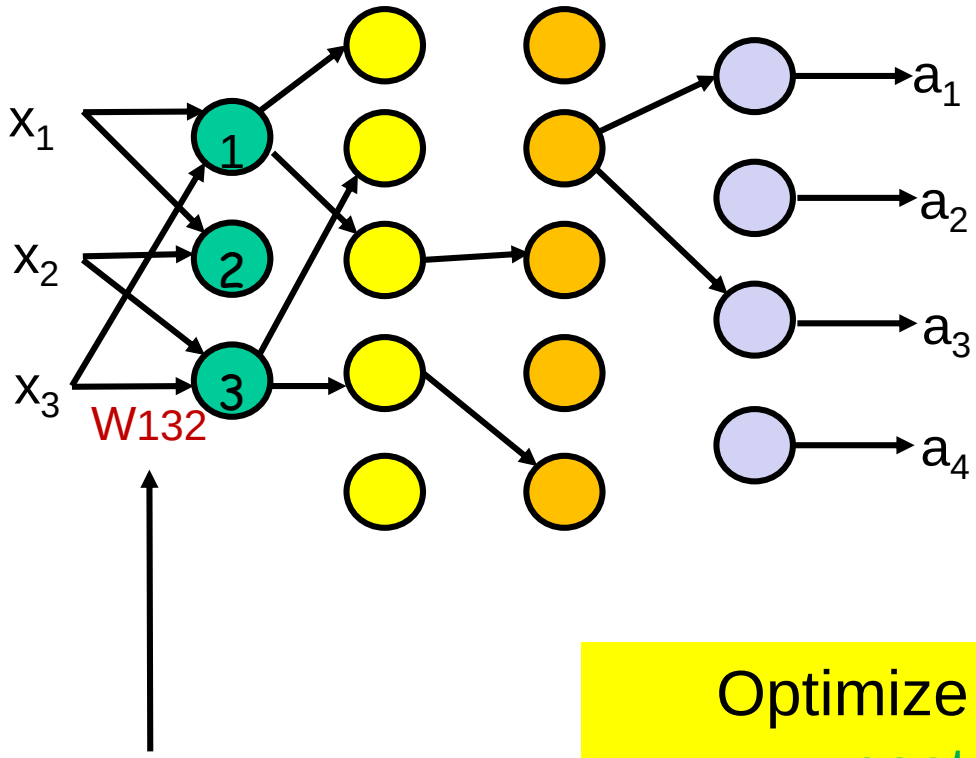
How to train your network

- Given a machine structure
- Parameters are the only free variables

➔ Choose parameters to maximize accuracy

Optimize a suitably defined
cost function h(w)
to find the right parameter vector w

How to train your network



parameters
w

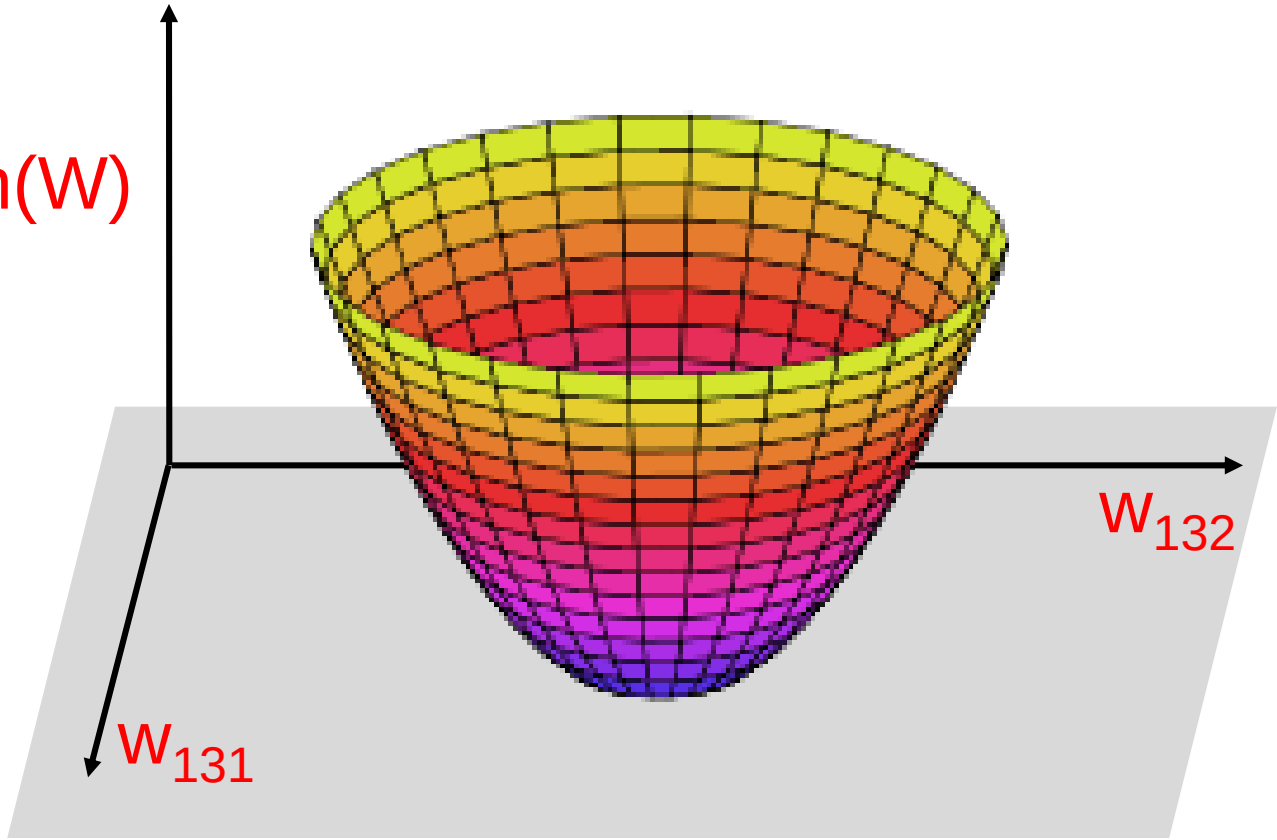
Optimize a suitably defined
cost function h(w)
to find the right parameter vector w

Cost Function h(w)

- Consider input x
- True classification y(x)
- Machine classification a(x,w) using parameters w
- Cost for input x $= \| y(x)-a(x,w) \|^2$
- Total cost h(w) $= \sum_{x} \| y(x)-a(x,w) \|^2$

Convex Optimization

$W = (w_{131}, w_{132}, \ldots)$

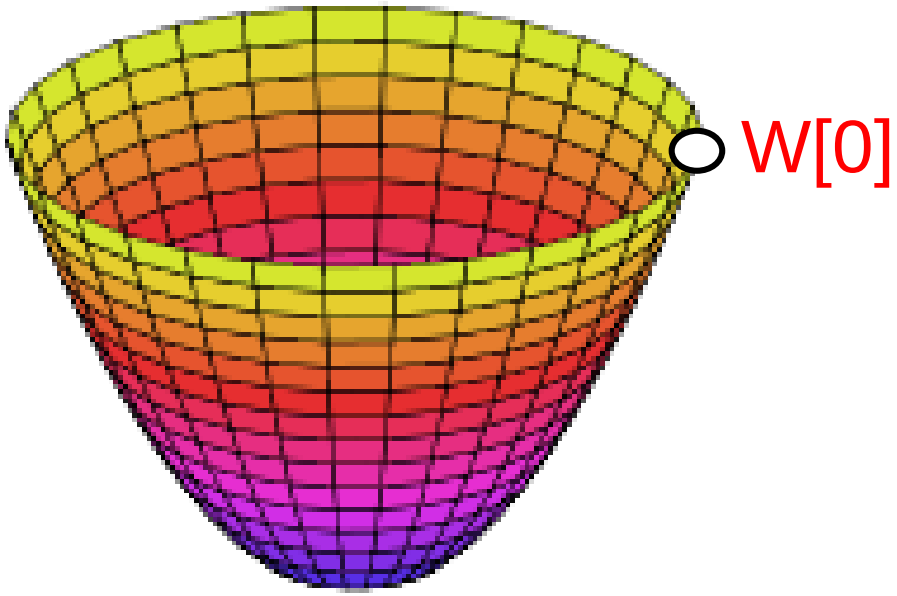


Wikipedia

Convex Optimization

$W = (w_{131}, w_{132}, \ldots)$

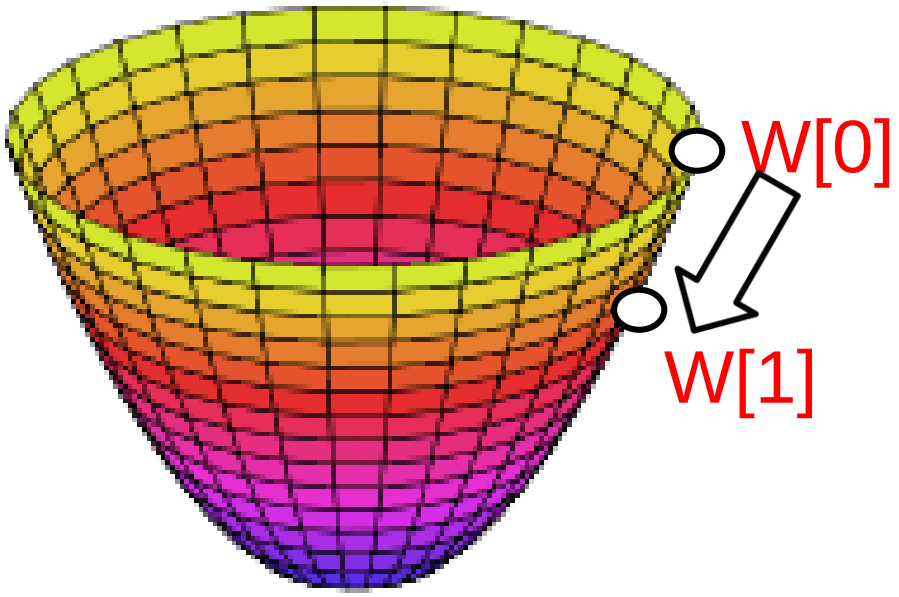
W[1] = (4,3,...)



Convex Optimization

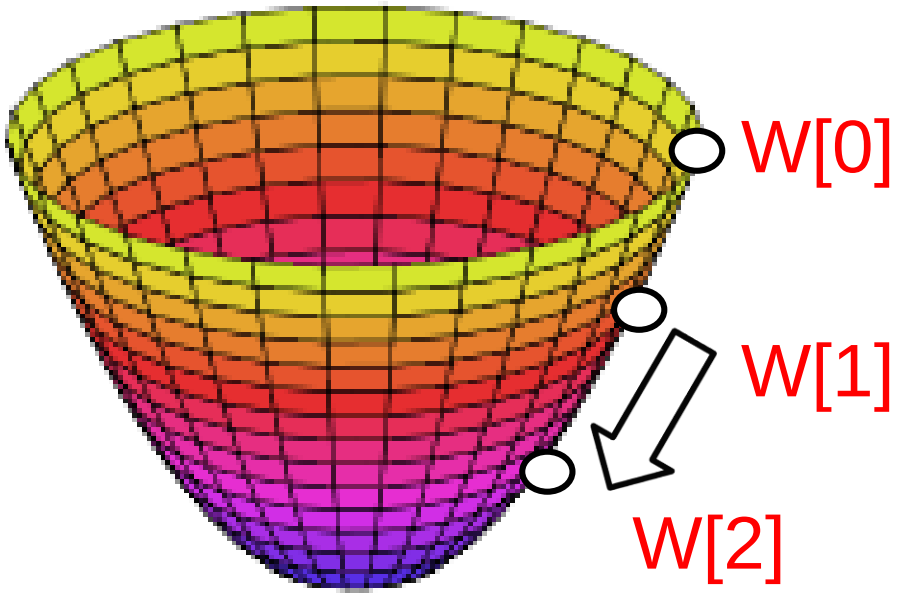
$W = (w_{131}, w_{132}, \ldots)$

W[1] = (4,3,...)
W[2] = (3,2,…)



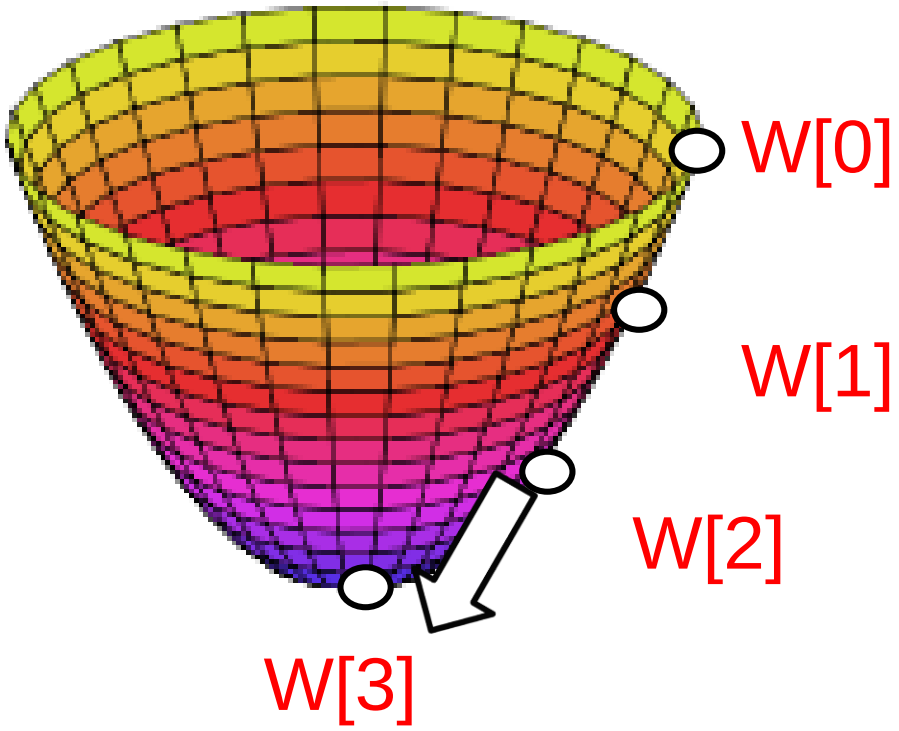
Convex Optimization

$W = (w_{131}, w_{132}, \ldots)$



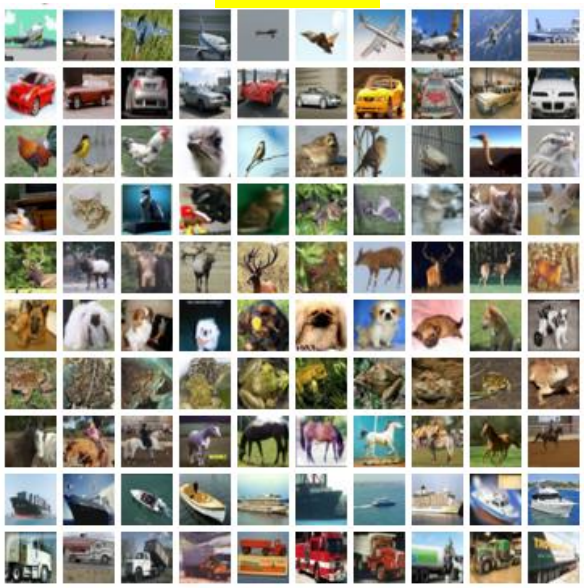
Convex Optimization

$W = (w_{131}, w_{132}, \ldots)$



So far …

h(w)



Training
➔
Optimize
cost function
h(w)

Machine
parameters
w

Outline

- Motivation – distributed machine learning
- Research problems
  - Privacy-preserving distributed optimization
  - Adversarial learning
  - Robustness to adversarial samples

Distributed Machine Learning

- Data is distributed across different agents
  - Mobile users
  - Hospitals
  - Competing vendors

Distributed Machine Learning

- Data is distributed across different agents
  - Mobile users
  - Hospitals
  - Competing vendors

Distributed Machine Learning

- Data is distributed across different agents ➔ Collaborate to learn

Distributed Machine Learning

- Data is distributed across different agents

➔ Collaborate to learn

Training
➔
Optimize
cost function
$\Sigma_i h_i(w)$

Machine
parameters
w

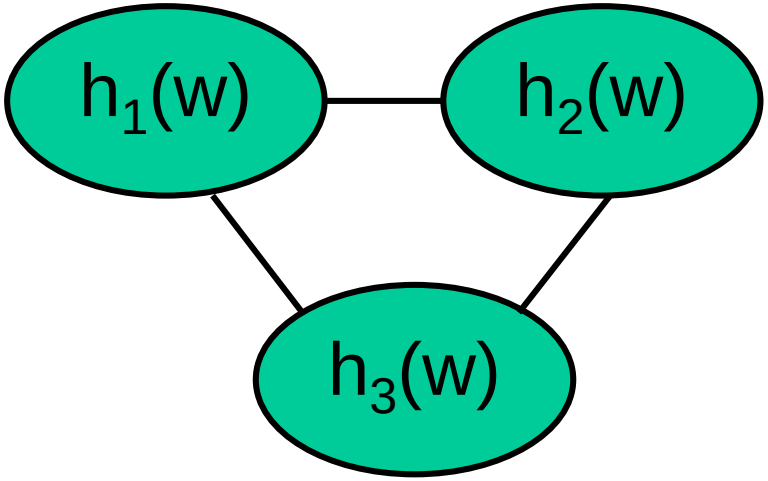
Distributed Optimization

- 30+ years or work
- Recent interest due to machine learning applications

Distributed Optimization

Different architectures

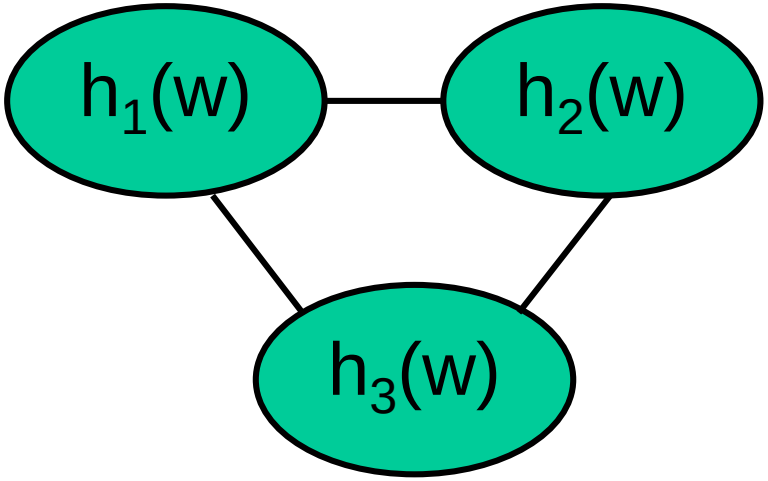
- Peer-to-peer



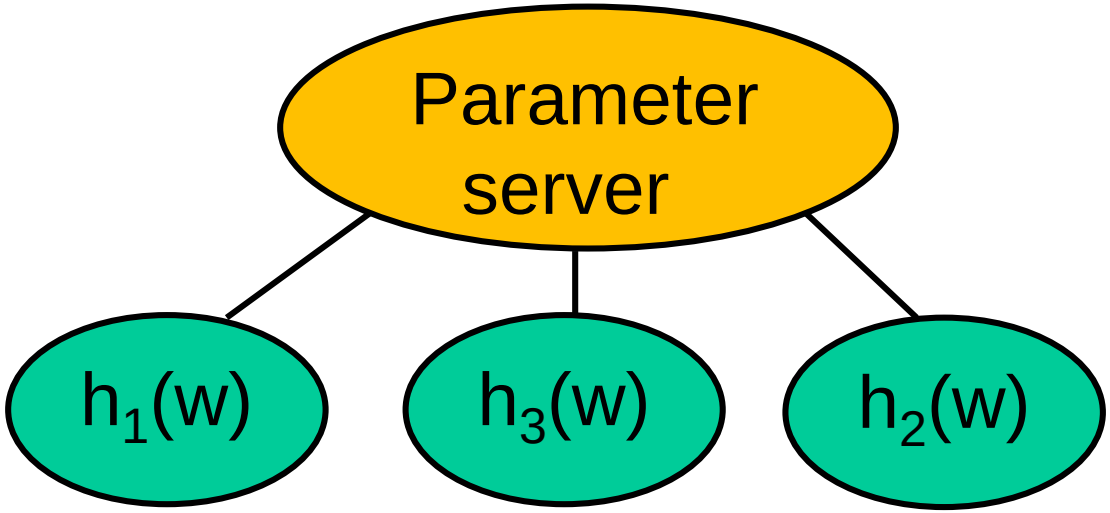
Distributed Optimization

Different architectures

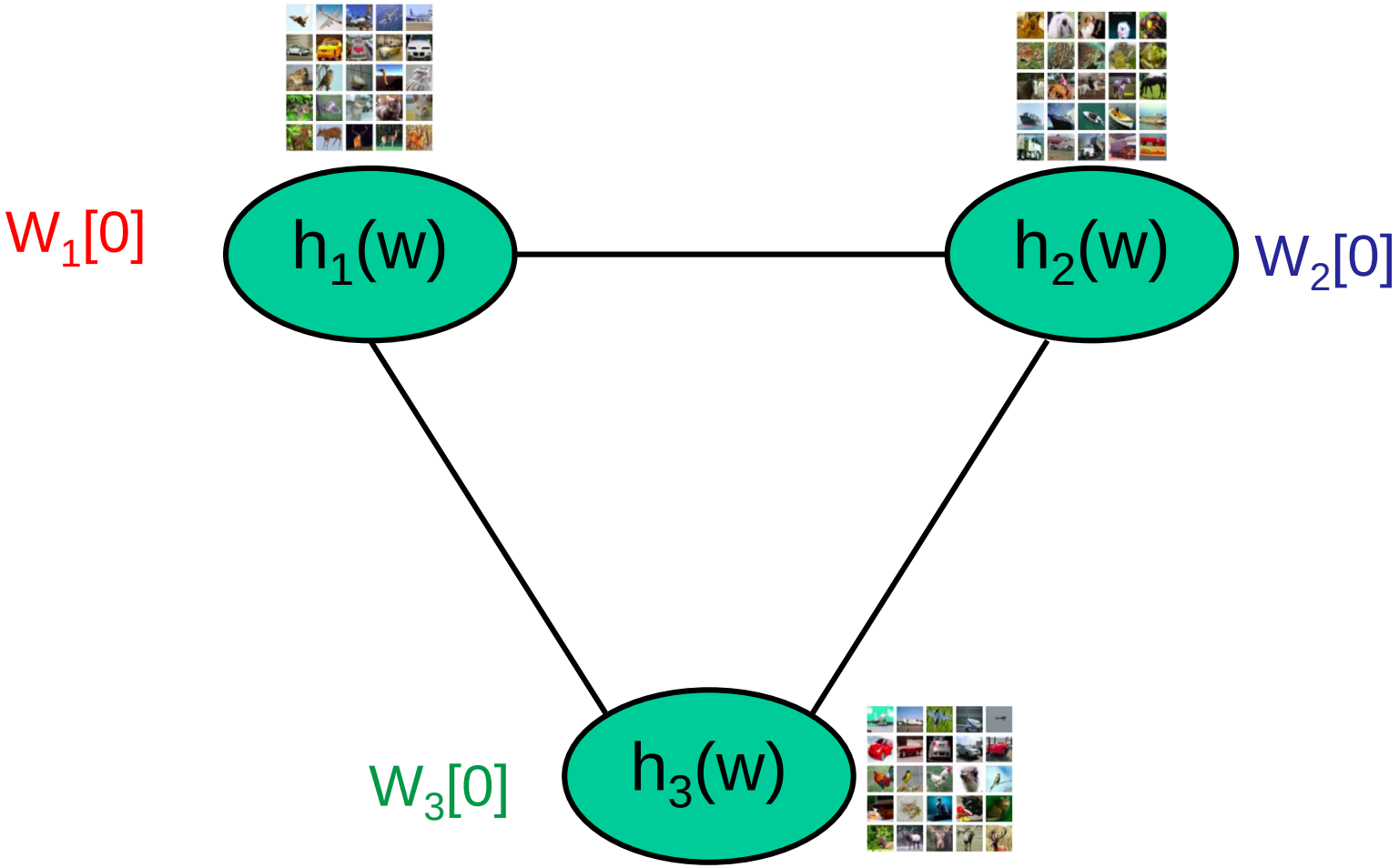
- Peer-to-peer



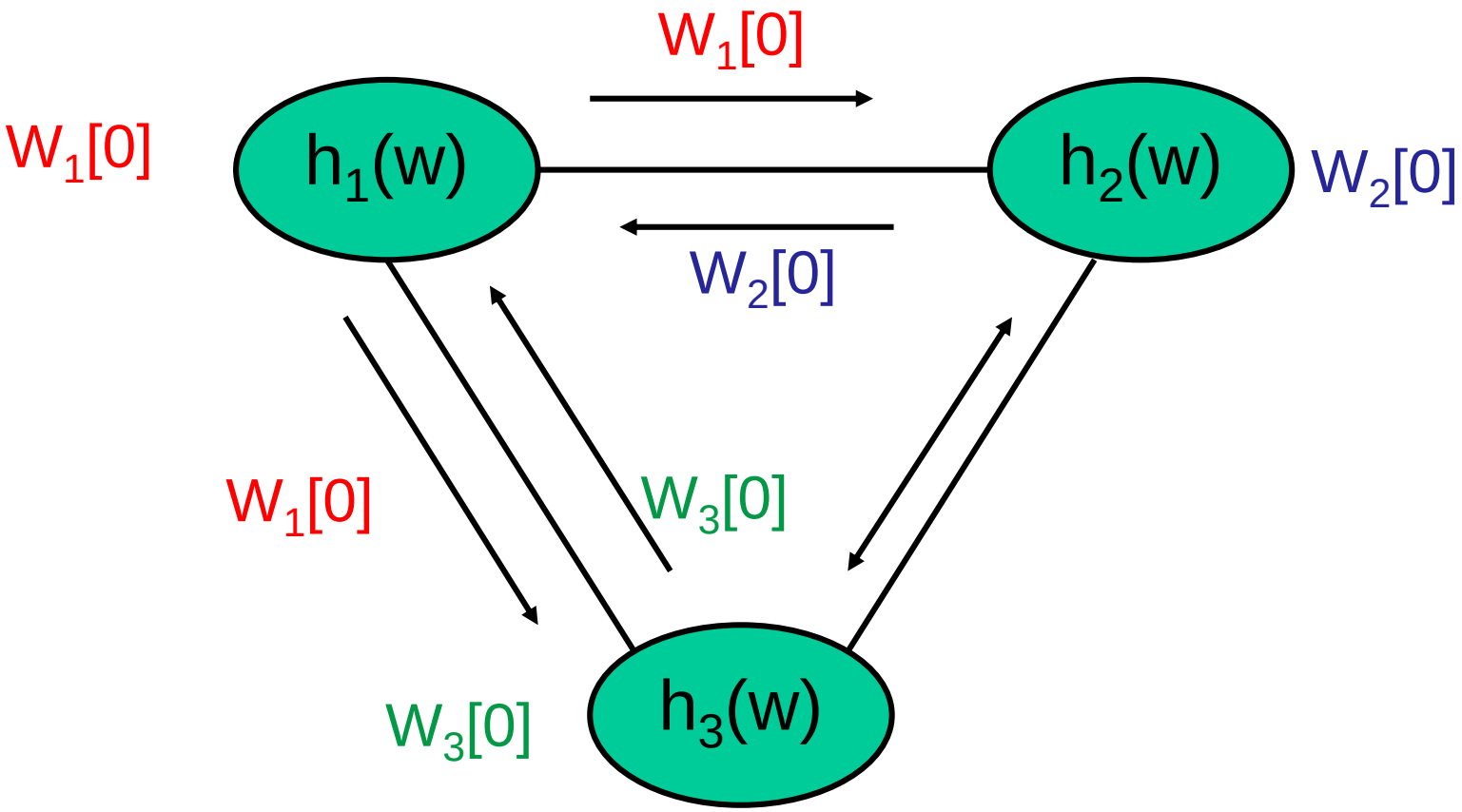
- Parameter server



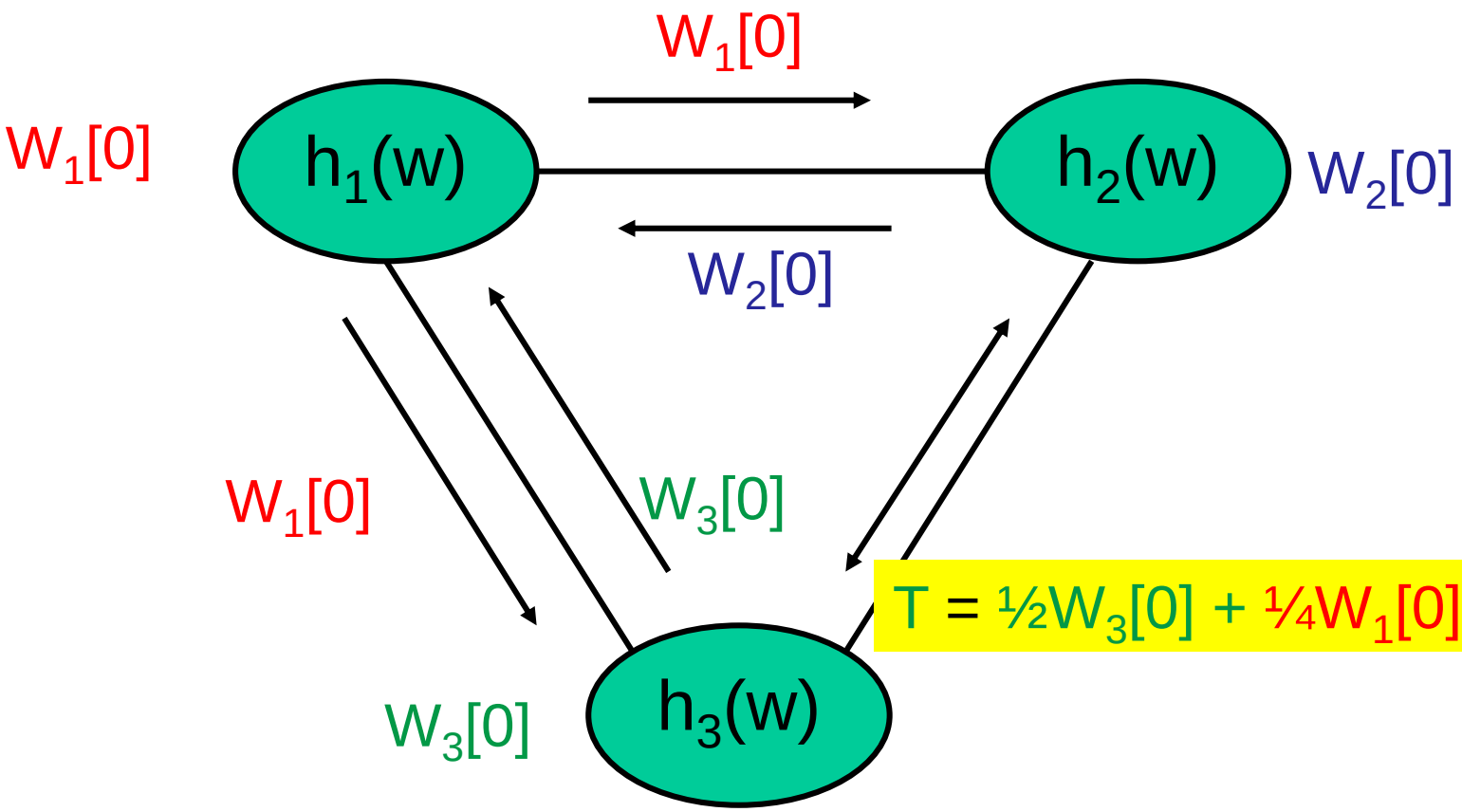
Distributed Gradient Method



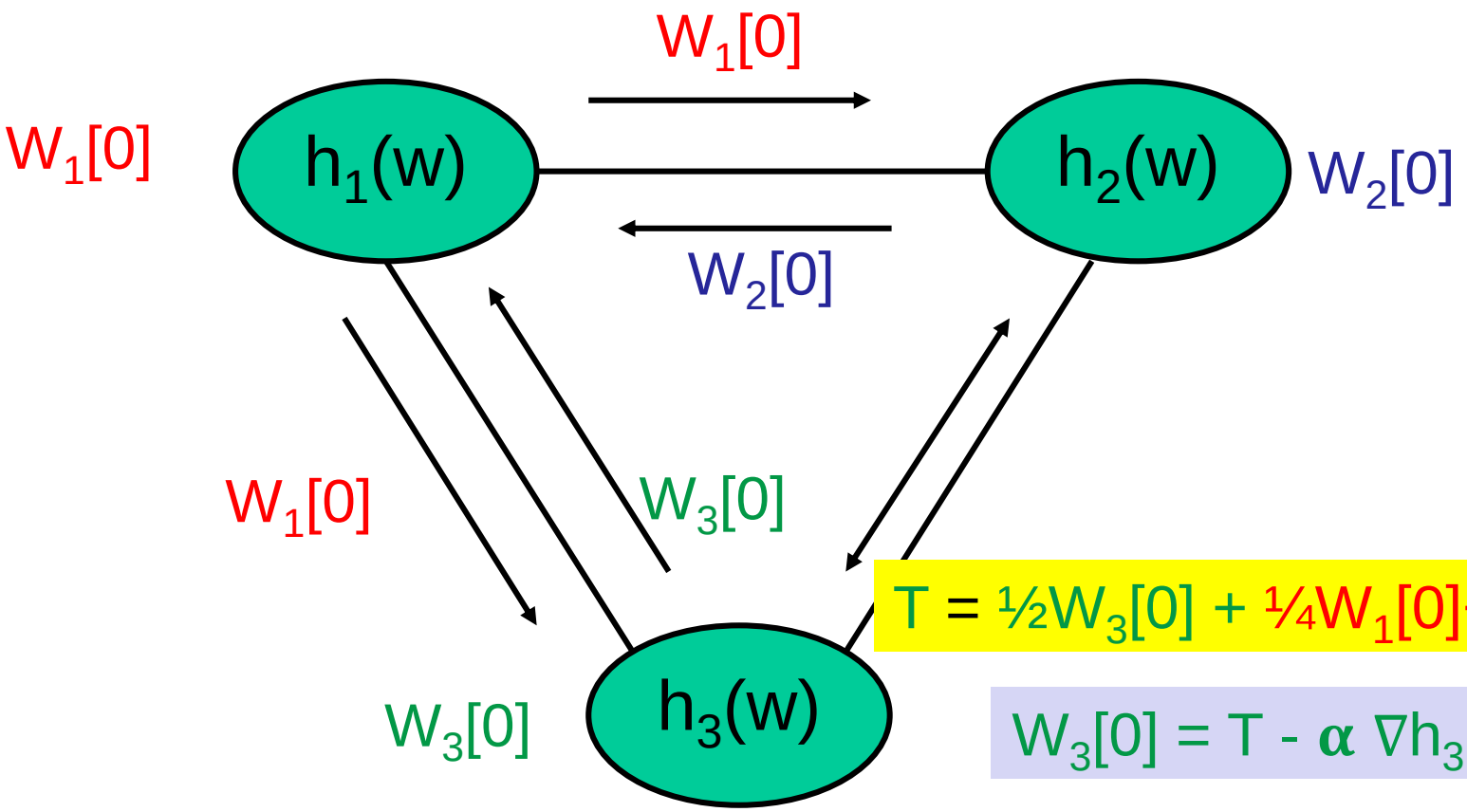
Distributed Gradient Method



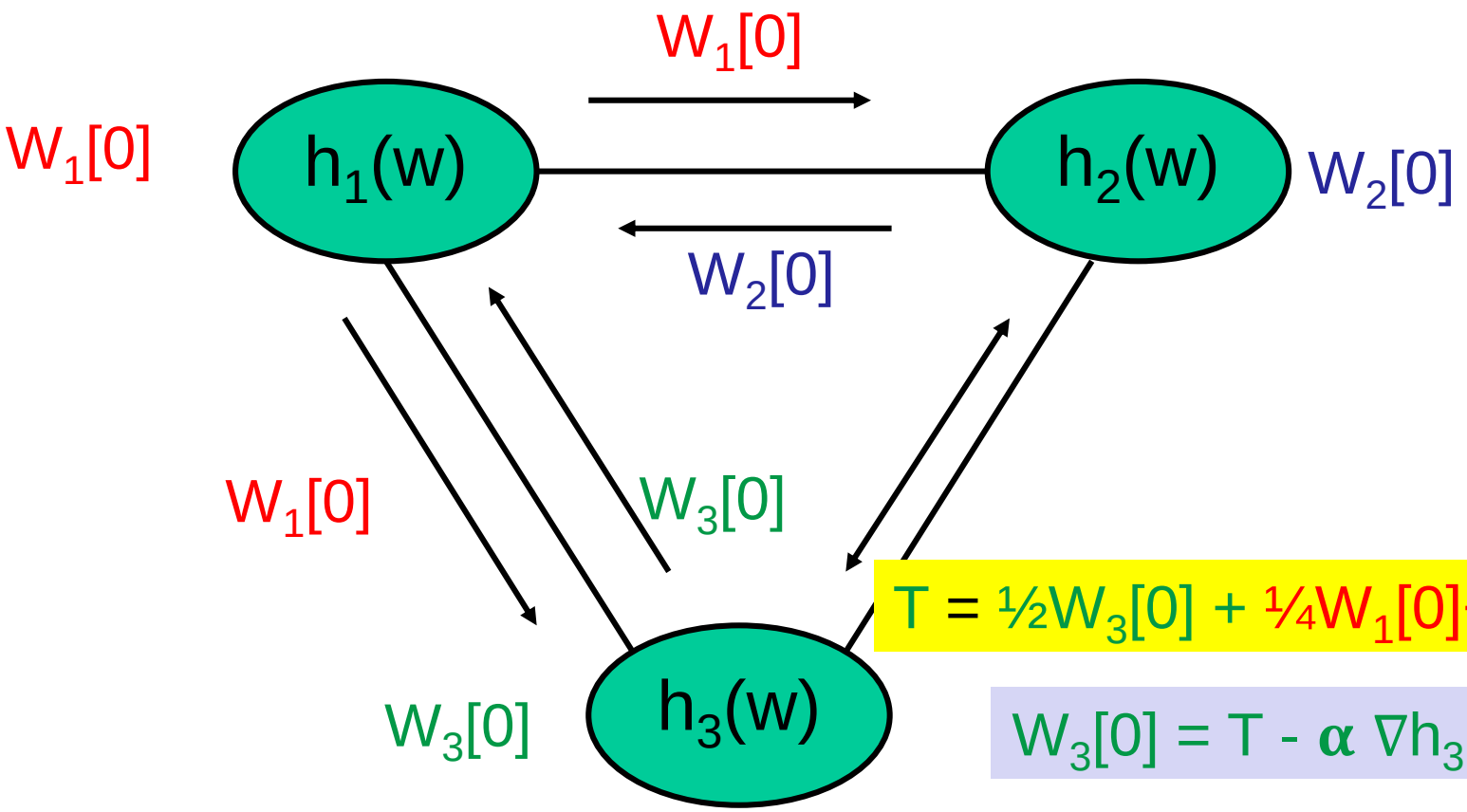
Distributed Gradient Method



Distributed Gradient Method



Works in incomplete networks too !!

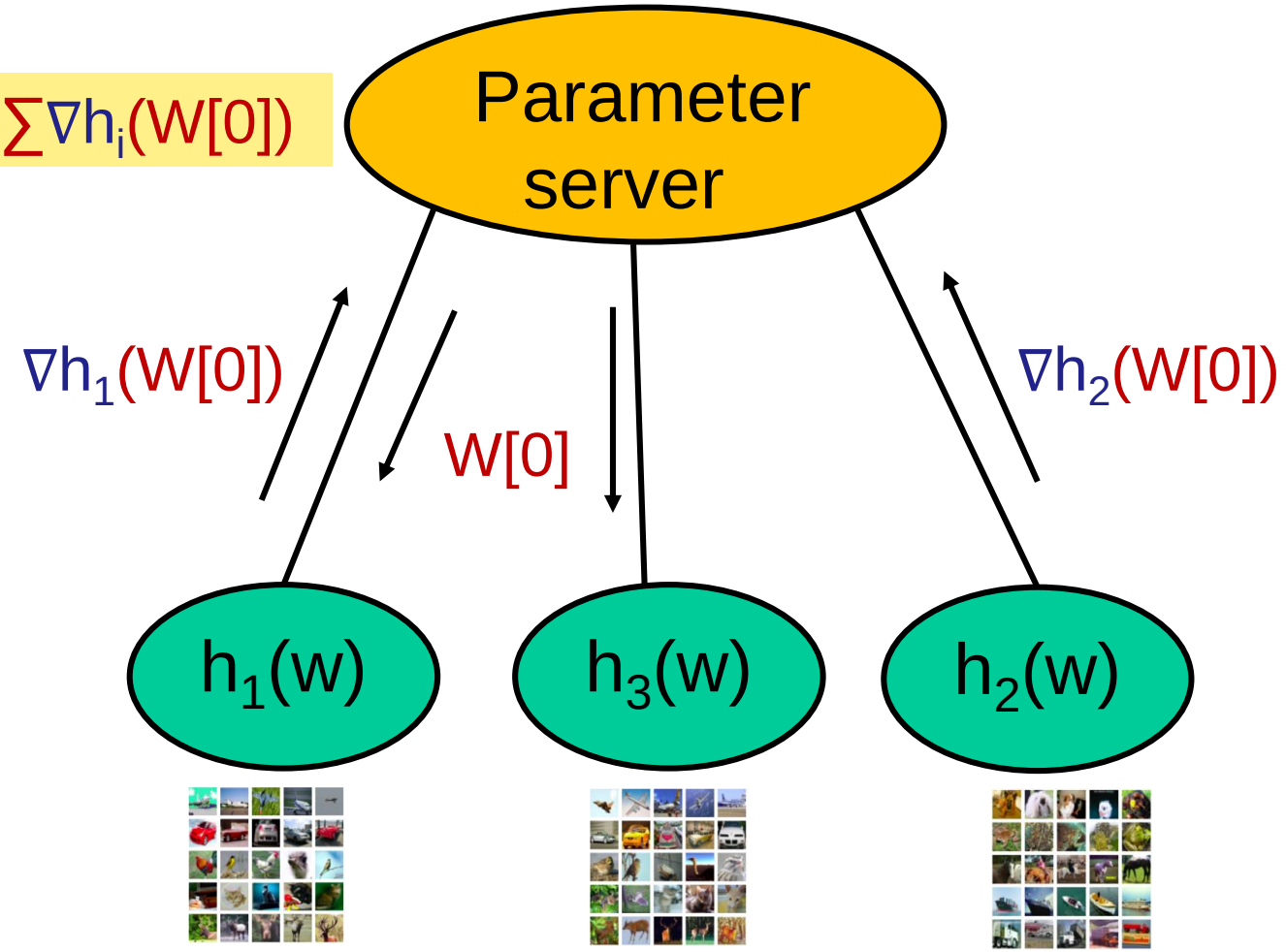


$T = \frac{1}{2}W_3[0] + \frac{1}{4}W_1[0] + \frac{1}{4}W_2[0]$

$W_3[0] = T - \alpha \nabla h_3(T)$

Parameter Server Architecture

$$W[1] = W[0] - \alpha \sum \nabla h_i(W[0])$$

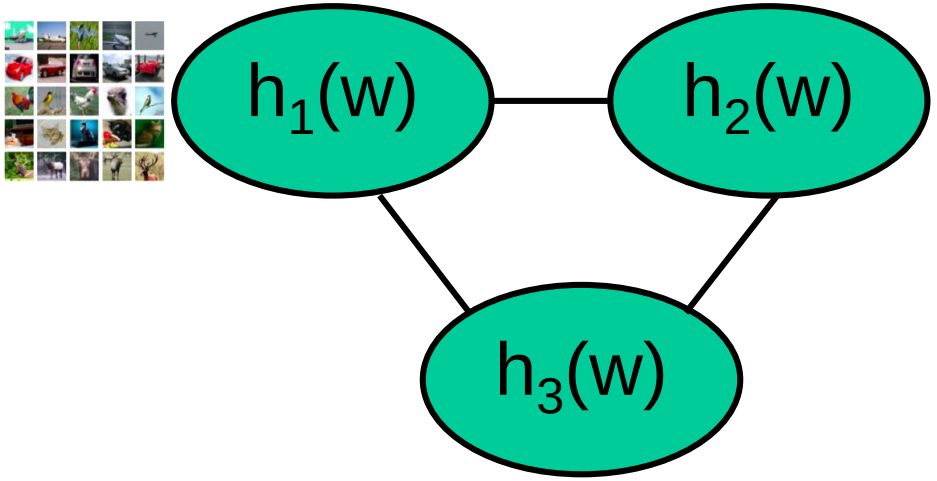


Outline

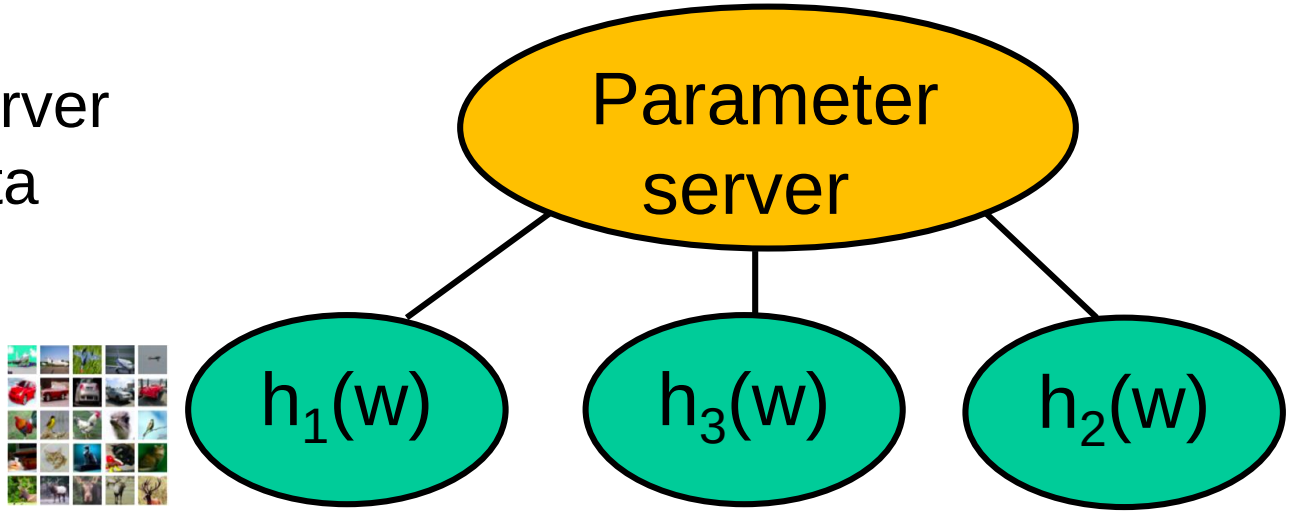
- Motivation – distributed machine learning
- Research problems
  - Privacy-preserving distributed optimization
  - Adversarial learning
  - Robustness to adversarial samples

Privacy Challenge

- Peers may learn each other's data

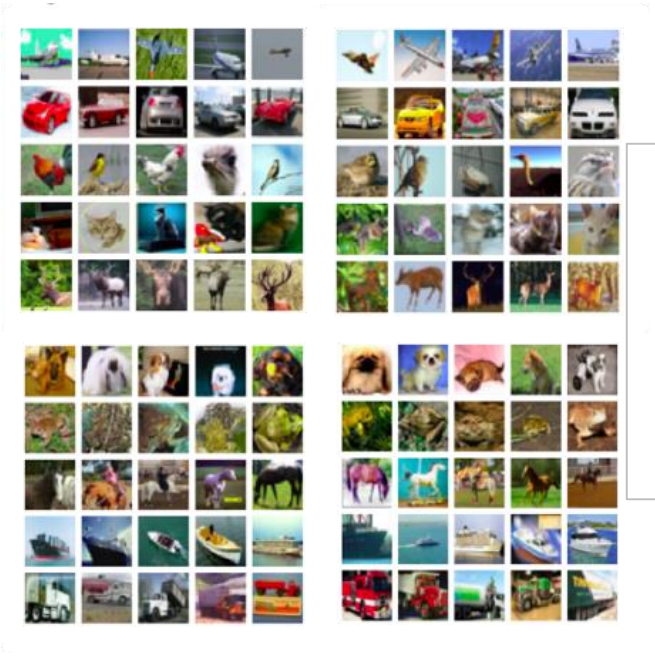


- Parameter server may learn data



Privacy-Preserving Optimization

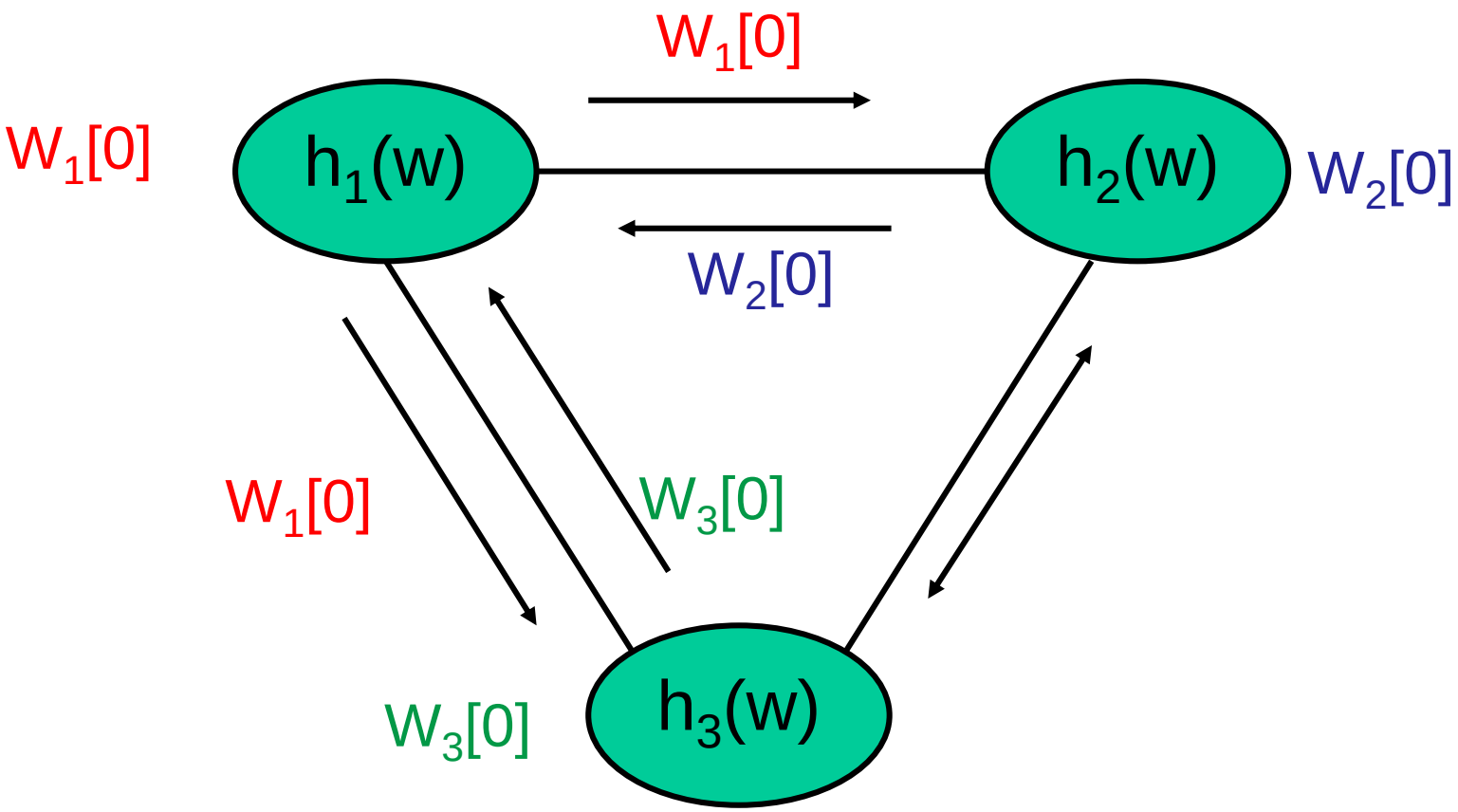
Can agents collaboratively learn, and yet protect own data ?



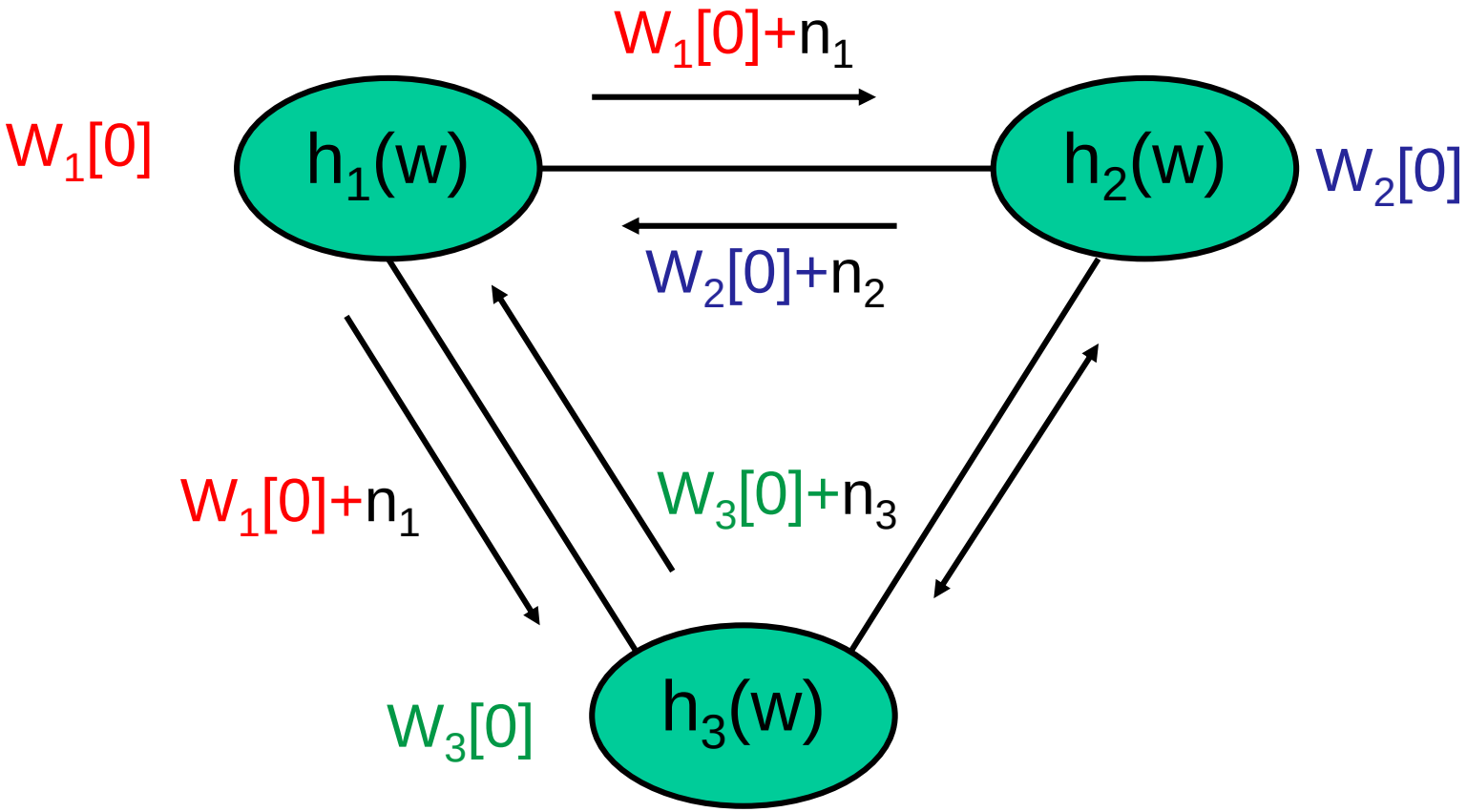
Optimize
cost function

$$\sum_i h_i(w)$$

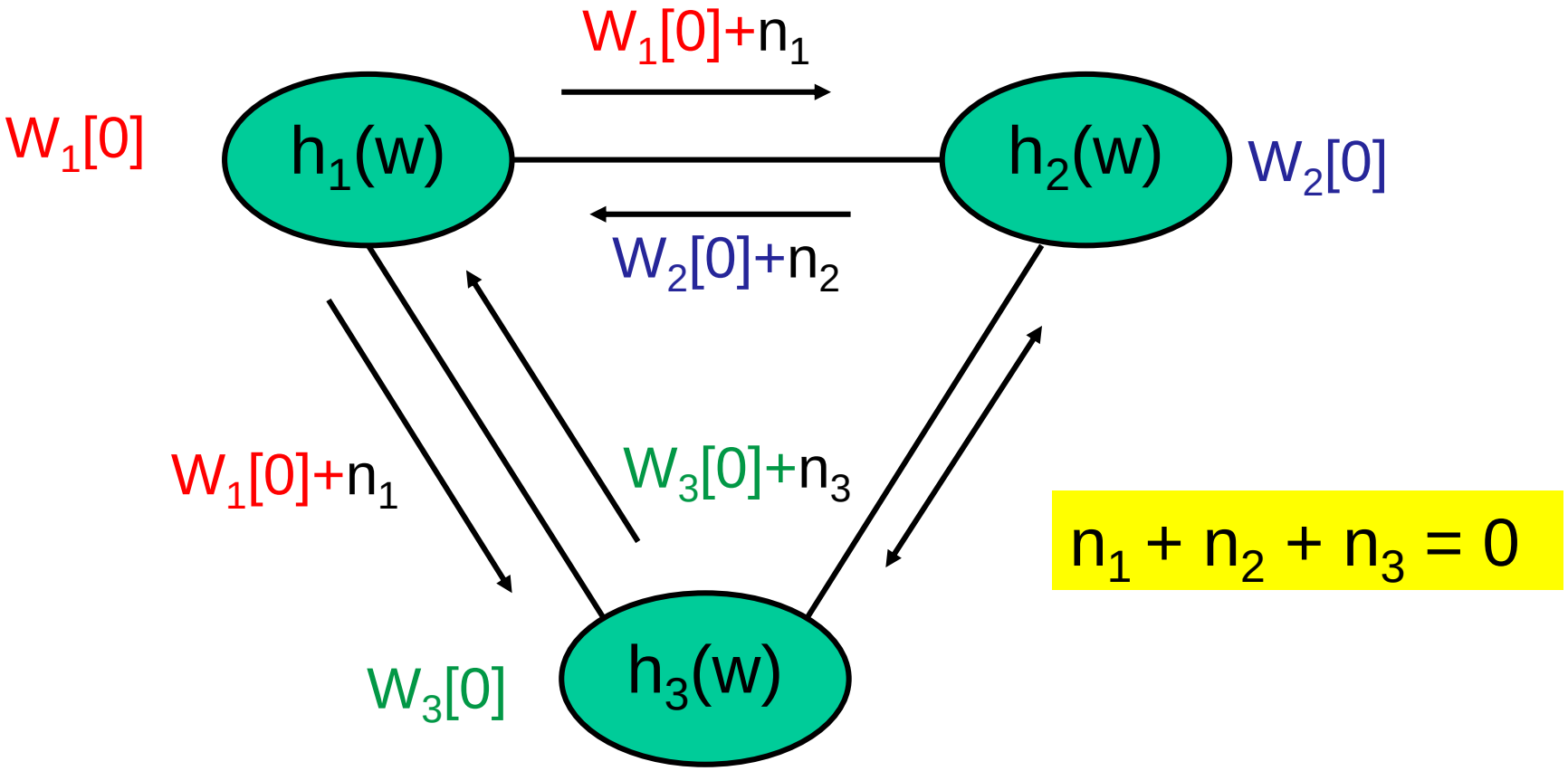
Peer-to-Peer Architecture



Add Inter-Dependent Noise



Add Inter-Dependent Noise



Key Idea

- Add correlated noise in information exchanged between agents
- Noise "cancels" over the network
- But can prevent coalition of bad agents learning information about others

Privacy-Preserving Optimization

Can agents collaboratively learn, and yet protect own data ?

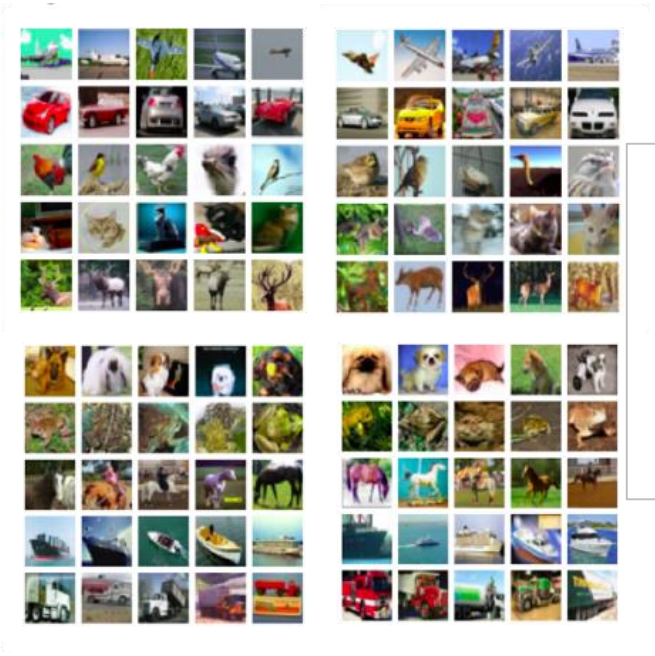
Optimize cost function

$\Sigma_i h_i(w)$

Yes!*

Privacy-Preserving Optimization

Can agents collaboratively learn, and yet protect own data ?



Optimize cost function

$$\sum_i h_i(w)$$

Yes!*

*** conditions apply**

Private Learning on Networks *

Shripad Gade Nitin H. Vaidya

Department of Electrical and Computer Engineering, and
Coordinated Science Laboratory,
University of Illinois at Urbana-Champaign.
Email: {gade3, nhv}@illinois.edu

Technical Report

Abstract

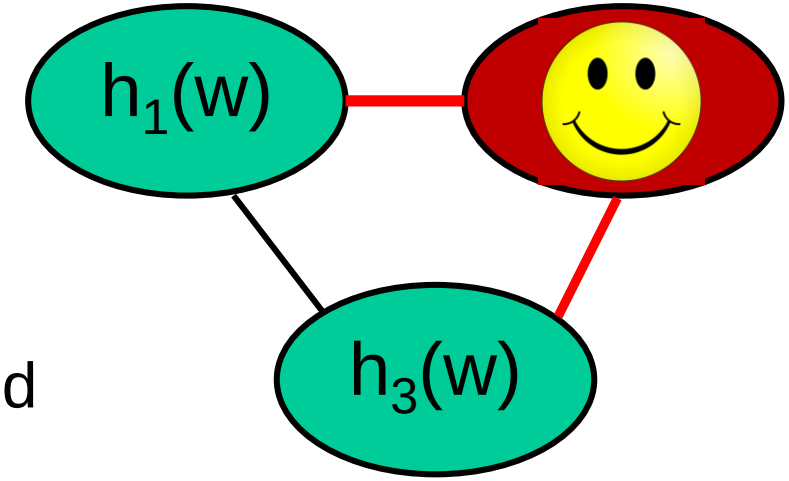
Continual data collection and widespread deployment of machine learning algorithms, particularly the distributed variants, have raised new privacy challenges. In a distributed machine learning scenario, the dataset is stored among several machines and they solve a distributed optimization problem to collectively learn the underlying model. We present a secure multi-party computation inspired privacy preserving distributed algorithm for optimizing a convex function consisting of several possibly non-convex functions. Each individual objective function is privately stored with an agent while the agents communicate model parameters with neighbor machines connected in a network. We show that our algorithm can correctly optimize the overall objective function and learn the underlying model accurately. We further prove that under a vertex connectivity condition on the topology, our algorithm preserves privacy of individual objective functions. We establish limits on the what a coalition of adversaries can learn by observing the messages and states shared over a network.

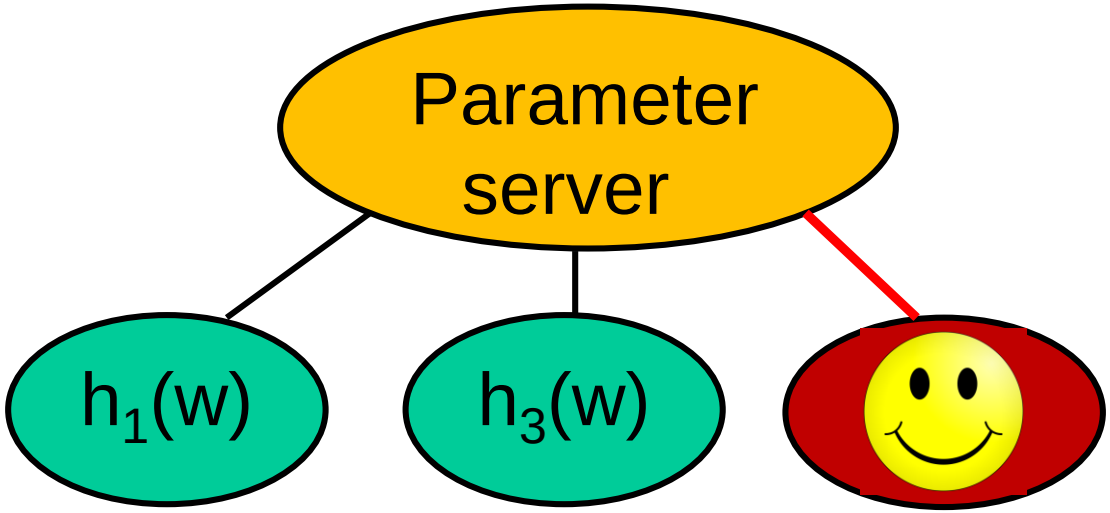
Outline

- Motivation – distributed machine learning
- Research problems
  - Privacy-preserving distributed optimization
  - Adversarial learning
  - Robustness to adversarial samples

Adversarial Agents

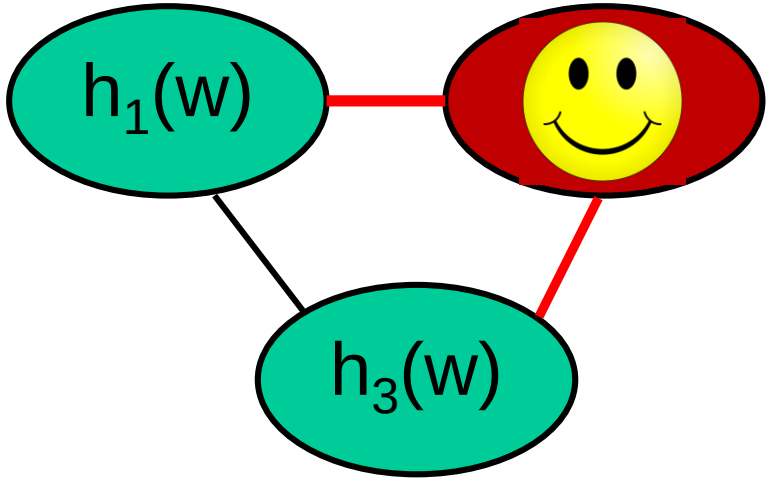
- Adversarial agents may send bogus information
- Learned parameters impacted

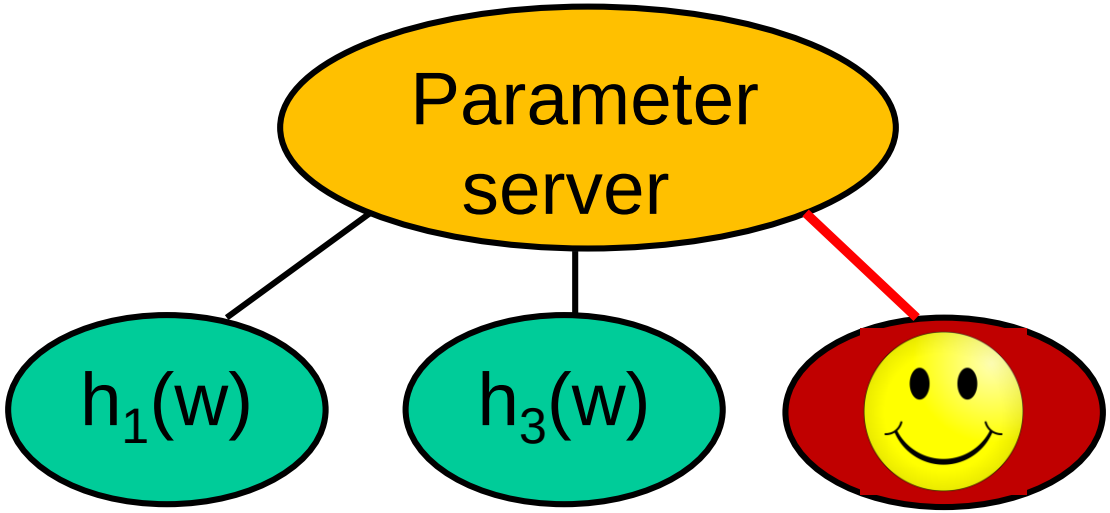




Adversarial Agents

Can good agents learn despite bad agents?

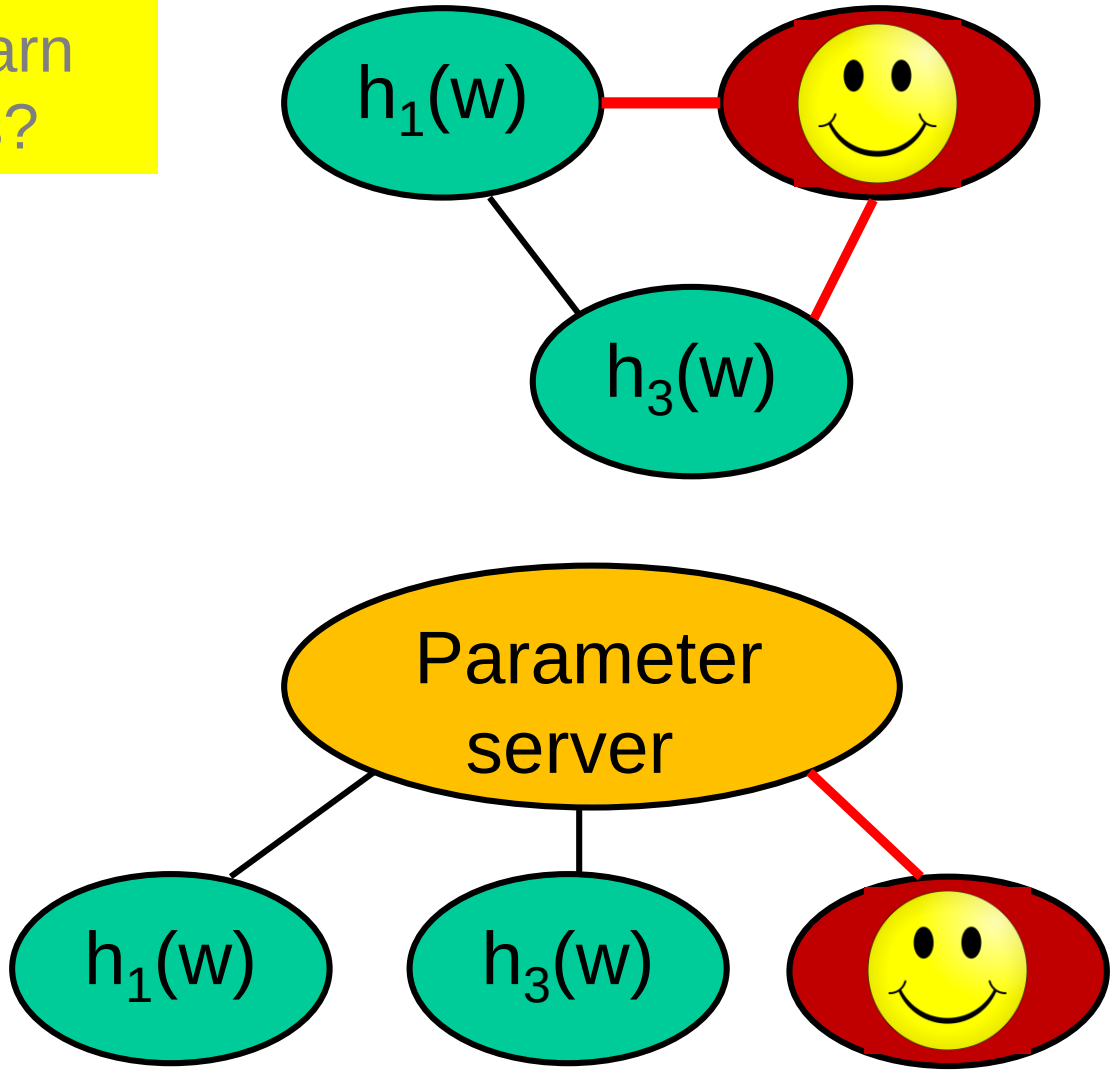




Adversarial Agents

Can good agents learn despite bad agents?

Yes!*



Key Idea

- Need to filter bad information
- Define “outliers” appropriately

Non-Bayesian Learning in the Presence of Byzantine Agents

Lili Su(✉) and Nitin H. Vaidya

Department of Electrical and Computer Engineering,
University of Illinois at Urbana-Champaign, Champaign, USA
{lilisu3,nhv}@illinois.edu

Abstract. This paper addresses the problem of non-Bayesian learning over multi-agent networks, where agents repeatedly collect partially informative observations about an *unknown* state of the world, and try to collaboratively learn the true state. We focus on the impact of the Byzantine agents on the performance of consensus-based non-Bayesian learning. Our goal is to design an algorithm for the *non-faulty* agents to collaboratively learn the true state through local communication.

We propose an update rule wherein each agent updates its local beliefs as (up to normalization) the product of (1) the likelihood of the *cumulative* private signals and (2) the weighted geometric average of the beliefs of its incoming neighbors and itself (using Byzantine consensus). Under mild assumptions on the underlying network structure and the global identifiability of the network, we show that all the non-faulty agents asymptotically agree on the true state almost surely.

Keywords: Distributed learning · Byzantine agreement · Fault-tolerance · Adversary attacks · Security

Boston Marriott Copley Place
July 6-8, 2016. Boston, MA, USA

Multi-Agent Optimization in the Presence of Byzantine Adversaries: Fundamental Limits*

Lili Su and Nitin Vaidya

***Abstract*— We study multi-agent optimization problem in the presence of Byzantine adversaries, where each agent i has a local cost function $h_i(x)$, and some unknown subset of agents suffer Byzantine faults. The goal is to optimize a global objective that properly aggregates the local cost functions. Ideally, we would like to optimize $\frac{1}{|\mathcal{N}|}\sum_{i\in\mathcal{N}} h_i(x)$, where $\mathcal{N}$ is the set of non-faulty agents. However, we show that this ideal goal is unachievable. Therefore, we define a relaxed version of the problem, named Byzantine multi-agent optimization, for which the goal is to generate an output that is an optimum of a global cost function formed as a *convex combination* of local cost functions kept by the non-faulty agents. More precisely, there must exist nonnegative weights α_i for $i \in \mathcal{N}$ such that $\sum_{i\in\mathcal{N}} \alpha_i = 1$, and the output is an optimum of $\sum_{i\in\mathcal{N}} \alpha_i h_i(x)$.**

In this paper, we focus on the impact of Byzantine attacks on the maximal achievable number of nonzero weights. To characterize the fundamental limits, we assume that the argument of each local cost function is a (real-valued) scalar, the network is fully-connected, and there is no restriction on the information exchange among agents. We show that the number of nonzero weights (α_i's) that can be guaranteed is at most $|\mathcal{N}| - f$, where f is the maximum number of Byzantine faulty agents. Additionally, we present algorithms that achieve this upper bound. By exploiting Byzantine broadcast for information exchange between agents, our proposed algorithms essentially

$$\boxed{\text{output}\quad x_o \quad\in\quad \operatorname*{argmin}_x \quad \frac{1}{n}\sum_{i=1}^{n} h_i(x)} \tag{1}$$

Each function $h_i(x)$ is assumed to be a *convex* function. (Section III will fully describe the properties of the cost functions.)

Due to its many potential applications, distributed multi-agent optimization has been a topic of significant research activity, as noted above. The applications include distributed machine learning, distributed resource allocation, and distributed robotics. In distributed machine learning problem [1], x represents parameters that need to be learned, using data available to a collection of agents. $f_i(x)$ denotes a loss function for agent i that depends on data available to agent i. In the resource allocation problem, the argument x represents allocation of shared resources to the agents, and the local cost functions depends on the *fairness* of the resource allocation. The global objective is to allow the agents to collaboratively agree on the most fair resource allocation. As a simple

Outline

- Motivation – distributed machine learning
- Research problems
  - Privacy-preserving distributed optimization
  - Adversarial learning
  - Robustness to adversarial samples

Adversarial Samples

- Machine learning seems to work well
- If it seems too good to be true …

Adversarial Samples

- Several researchers have shown that it is easy to fool a machine

Original

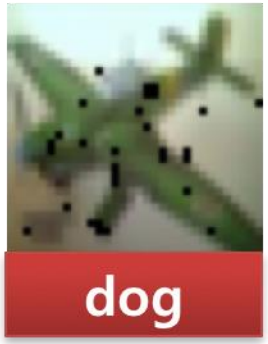
6

Adversarial

8

original sample

adversarial sample



Can we solve the problem?

May be … or not

- Some interesting ideas that seem promising in early evaluations

… but not mature enought to report yet

Summary

- Achieving privacy/security in learning is non-trivial
- Some promising progress
- Plenty to keep us busy for a while …

disc.ece.illinois.edu

Collaborators

- Lili Su (Ph.D. candidate)
- Shripad Gade (Ph.D. candidate)
- Nishad Phadke (BS thesis)
- Brian Wang (BS thesis)
- Professor Jungmin So (on sabbatical)

Collaborators

- Lili Su (Ph.D. candidate)
- Shripad Gade (Ph.D. candidate)
- Nishad Phadke (BS thesis)
- Brian Wang (BS thesis)
- Professor Jungmin So (on sabbatical)

Other related effort -- fault-tolerant control

- Professor Aranya Chakarabortty (on sabbatical)

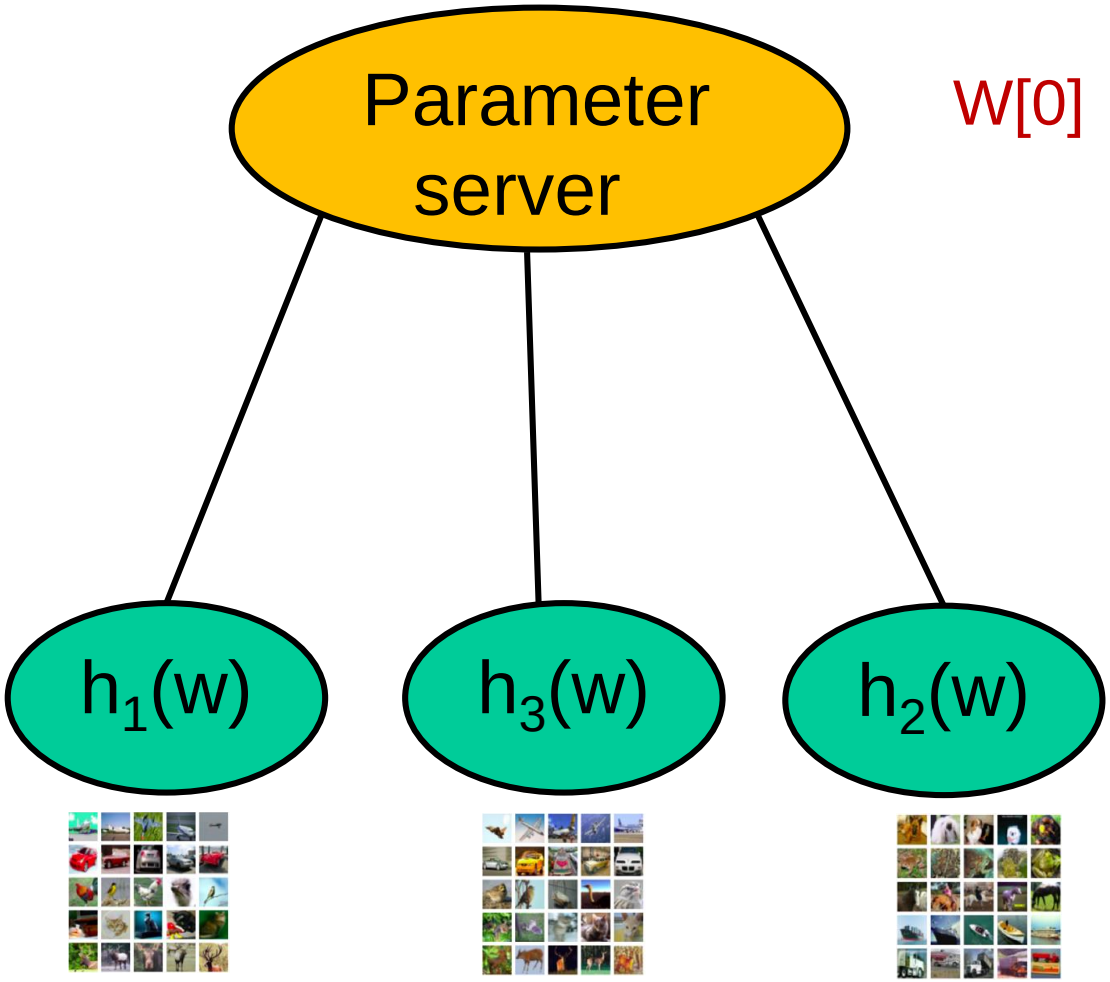
Summary

- Achieving privacy/security in learning is non-trivial
- Some promising progress
- Plenty to keep us busy for a while …

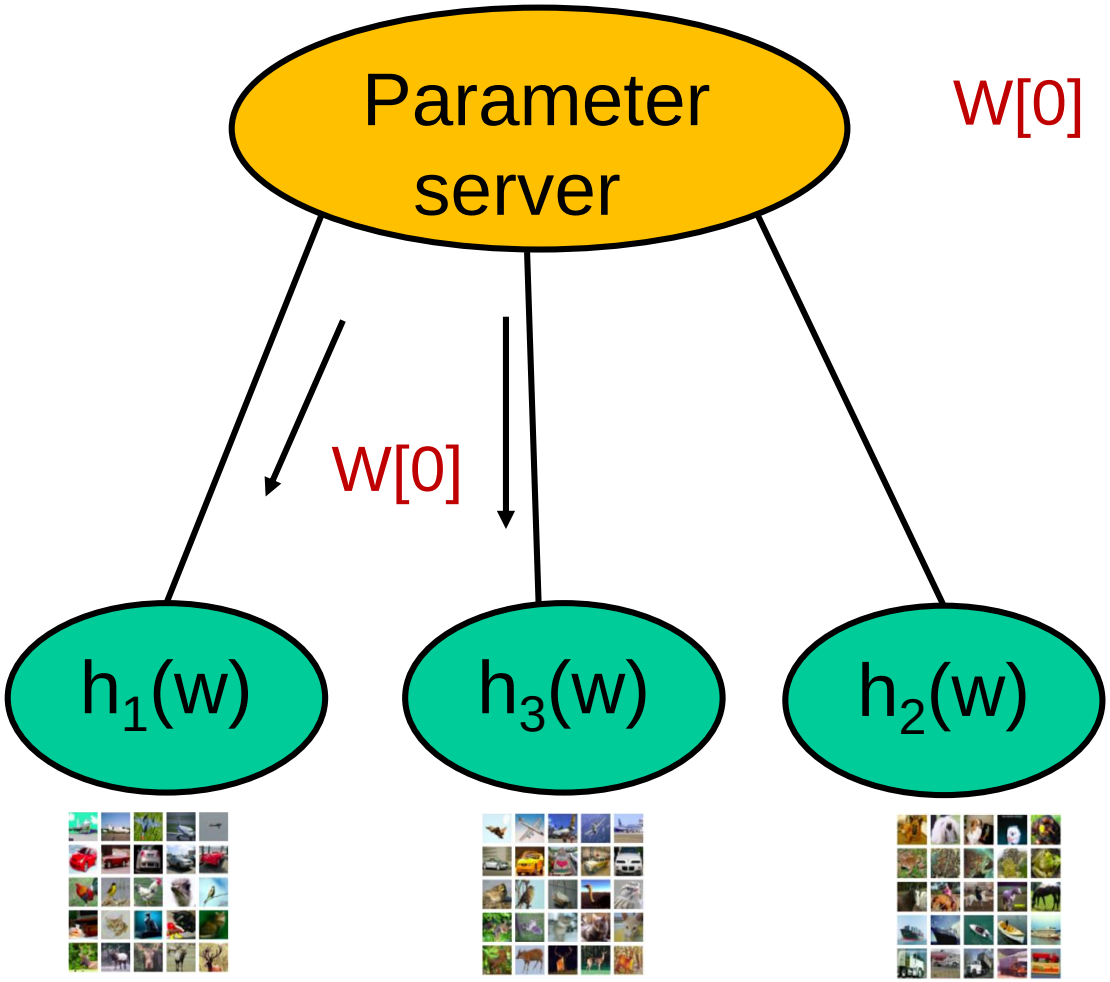
disc.ece.illinois.edu

Parameter Server Architecture

- Distributed gradient method



Distributed Optimization



Distributed Optimization