CSCI 599: Deep Learning and its Applications

Lecture 3

Spring 2019
Joseph J. Lim

Disclaimer

- This course is taught for the 2nd time @ USC. This course is 599, and thus an **experimental** course.
- The syllabus, course policy, and grading details **may change** over the semester (**check website!**)
- If you prefer a well-structured course, this is **NOT** a course for you, and I encourage you to take the course next year.
- But, it will be **fun** and **challenging**!

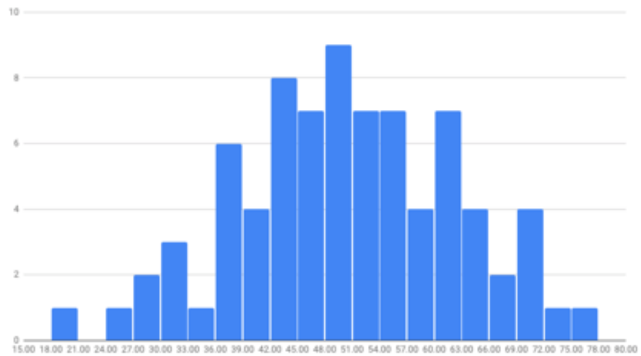
Today's agenda

- CSCI 599 overview
- Learning 101
- Loss function & Optimization
- Neural Networks

Today's agenda

- CSCI 599 overview
- Learning 101
- Loss function & Optimization
- Neural Networks

Entrance Exam



In-class Exam (81 students)

Any student who shouldn't be here

- Did you come to the 1st lecture and put down your info with TA?
- Or, did you take the exam and received an e-mail indicating that you passed the exam?

Welcome to CSCI 599!

Office Hours

- Instructor OH @ RTH 402
  - Tuesday 2-3:30pm
  - This is NOT for homework related questions.
- TA OH @ SAL 125
  - Monday 5-6pm & Tuesday 12:30-1:30pm
  - Extra OH for assignment due & midterm weeks: Monday 4-5pm & Tuesday 11:30-12:30pm

CSCI 599/699

Communication

- Please use **Piazza** for any general communication including questions https://piazza.com/usc/spring2019/csci599/home
- E-mails will be ignored.
- **Register TODAY. Look for your project team mates!**

Communication

- Please do NOT
  - e-mail us individually (**we will not reply**)
  - come to our office without appointment

Syllabus

	Topics/Daily Activities	**Readings and Homework**	**Deliverable/ Due Dates**
Week 1 1/8	Course Introduction / Applications of Deep Learning		
Week 2 1/15	Entrance Exam		
	Module 1: CNN		
Week 3 1/22	Machine Learning 101 + Course registration + Loss functions and Optimization + Neural Networks		
Week 4 1/29	Convolutional Neural Networks + Training Neural Networks	**Assignment 1 OUT**	
Week 5 2/5	CNN Architectures + Deep Learning Software + Cloud service (by TA)		**Course Project Team**

Syllabus

Module 2			
Week 6 2/12	Recurrent Neural Networks		
Week 7 2/19	Deep Generative Models		**Assignment 1**
Week 8 2/26	Deep Reinforcement Learning		**Course Project Proposal**
Week 9 3/5	**In-class Midterm**	**Assignment 2 OUT**	
Week 10 3/12	Spring Break		

Subject to change!

	Module 3: Advanced Topics		
Week 11 3/19	Advanced topics 1 (NLP) + Research Highlight		
Week 12 3/26	Advanced topics 2 (computer vision, robotics) + Research Highlight		**Assignment 2**
Week 13 4/2	Advanced topics 3 (attention, memory, relation networks) + Research Highlight		**Course Project Mid-report**
Week 14 4/9	Advanced topics 4 (AlphaGO, imitation learning, transfer learning) + Research Highlight		
Week 15 4/16	Advanced topics 5 (meta learning) + Research Highlight		**Final report (4/21)**
Week 16 4/23	**Term Project Presentation (4 hours) Spotlight + Poster**		
FINAL	**No Final**		

Important Dates

Subject to change!

- Entrance exam: 1/15
- Assignment 1: 2/19
- Midterm: 3/5
- Project meeting with Instructor #1: 3/6 — 3/8
- Assignment 2: 3/26
- Project meeting with Instructor #2: 4/1 — 4/3
- Project meeting with TA: 2 times (arranged later)
- Final presentation: 4/23 5-9:00pm **4 hours**

Course Project

Subject to change!

- Computational resource (**be conserv**
  $150 Google Cloud credit per student
  $125 Amazon AWS credit per student
- Tentative Schedule for Project
  - Week 5 (2/5): Course Project Team
  - Week 8 (2/26): Course Project Proposal
  - Week 13 (4/2): Mid-report
  - Week 16 (4/23): **Project Presentation** (5-9pm) + Report

Course Project

Subject to change!

- Computational resource (**be conservative!**):
  $150 Google Cloud credit per student
  $125 Amazon AWS credit per student

Course Project

- Team-based project (4-5 students per team)
- Each team will have at least 1 dedicated TA
  - 2 Mandatory meetings with TA
  - 2 meetings with me
- Create your own problems (extra points)
  - **Talk and discuss** with your TAs and me!
  - In the worst case, we will give a project idea
    - Less fun, Less points!

Project Evaluation

- Creativity and difficulty of the problem setup
- Novelty of the approach
- Thoroughness of the experiments
- Quality of student's presentation, report, and meetings with TA/instructor

Extra credit for creating your own project (OK to discuss and get help from TAs)

Team Formation

- Start forming your team NOW!

Today's agenda

- CSCI 599 overview
- Learning 101
- Loss function & Optimization
- Neural Networks

Deep Learning is impacting everywhere

- Machine Learning
- Computer Vision
- Natural Language Processing
- Robotics
- Medical Application
- Graphics
- Finance
- and many more

It's matter of one function

$X \longrightarrow \boxed{f(X)} \longrightarrow Y$

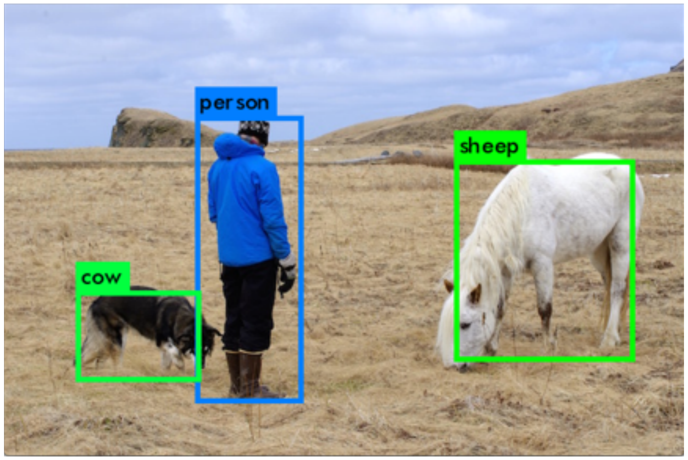
It's matter of one function

$X \longrightarrow \boxed{f(X)} \longrightarrow Y$

Really…?

Let's take a look

Object Detection



Object Detection in Video

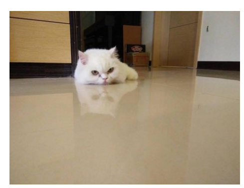
J. Redmon and A. Farhadi. YOLO9000: Better, Faster, Stronger. CVPR 2017.

Object Detection

X: Image → f(X)

Object Detection

X: Image → f(X) → Y: Object label & Bounding box



Semantic Segmentation

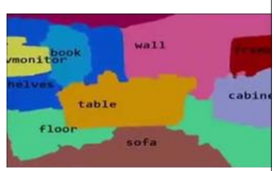
K. He, etl. al. Mask R-CNN. arXiv 2017.

Semantic Segmentation

X: Image → f(X)

Semantic Segmentation

X: Image → f(X) → Y: Pixel-level Label



3D Pose for Furniture

J. Lim, et. al. Parsing IKEA Objects: Fine Pose Estimation. ICCV 2013.

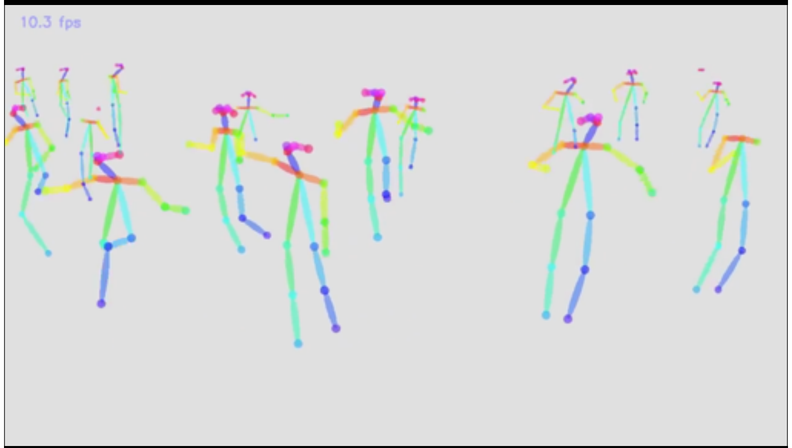
3D Pose for Furniture

X: Image → f(X)

3D Pose for Furniture

X: Image → f(X) → Y: Pose (Scale, rotation, and translation)

Human Pose



Z. Cao, et. al. Realtime Multi-person 2D Pose Estimation using Part Affinity Fields. CVPR 2017.

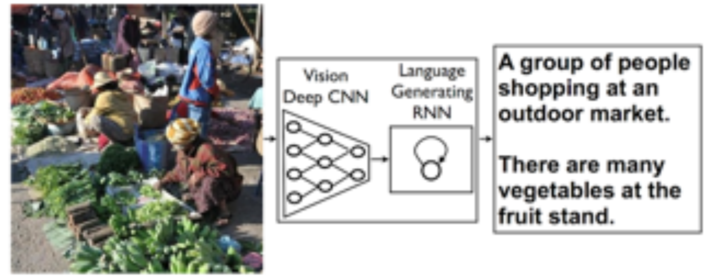
Human Pose

X: Image → f(X)

Human Pose

X: Image → f(X) → Y: Pose of body joint

Image to Caption



From https://research.googleblog.com/2014/11/a-picture-is-worth-thousand-coherent.html

It's matter of one function

X: Image → f(X)

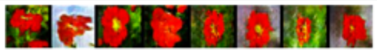
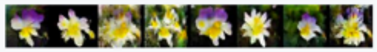
It's matter of one function

X: Image → f(X) → Y: Description

A group of people shopping at an outdoor market.

There are many vegetables at the fruit stand.

Caption to image

Caption	Generated Images
the flower shown has yellow anther red pistil and bright red petals	
this flower has petals that are yellow, white and purple and has dark lines	
the petals on this flower are white with a yellow center	

Reed, Scott, et. al. Generative Adversarial Text to Image Synthesis. ICML 2016.

It's matter of one function

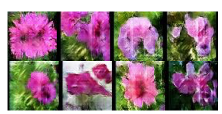
X: Description → f(X)

This flower has a lot of small round pink petals.

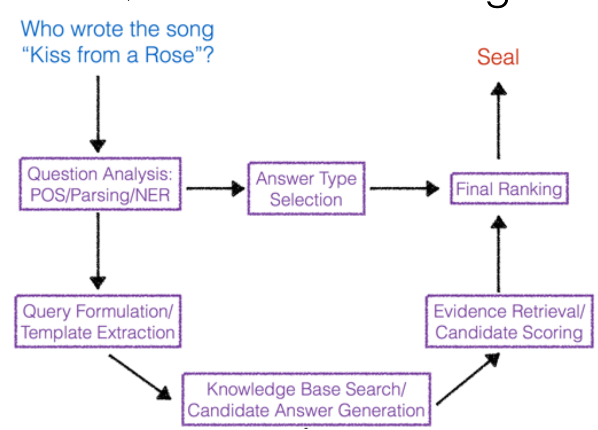
It's matter of one function

X: Description → f(X) → Y: Image

This flower has a lot of small round pink petals.



Question Answering



Kumar, Ankit, et. al. "Ask me anything: Dynamic memory networks for natural language processing." ICML 2016.

Question Answering

X: Query → f(X) → Y: Answer

Who is the most handsome person in the world?

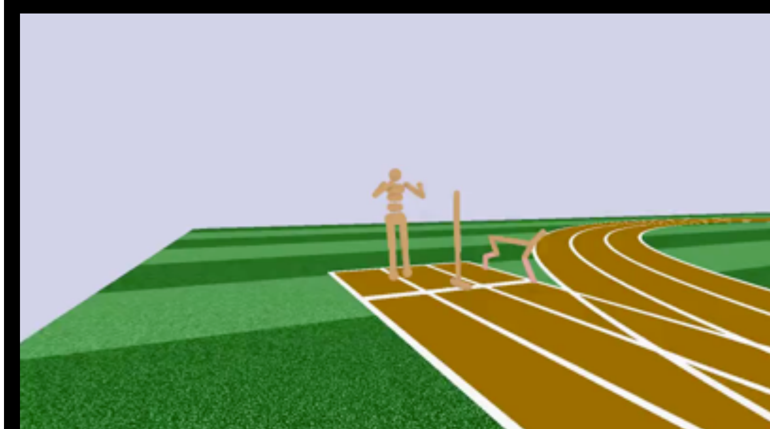
Question Answering

X: Query → f(X) → Y: Answer

Who is the most handsome person in the world?

Me

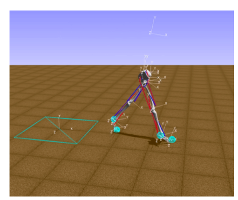
Learning to Walk



From https://blog.openai.com/roboschool/

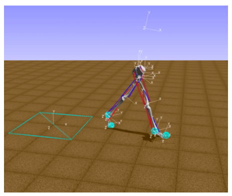
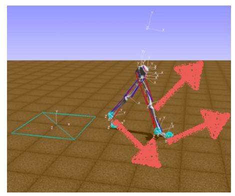
Learning to Walk

X: States → f(X)

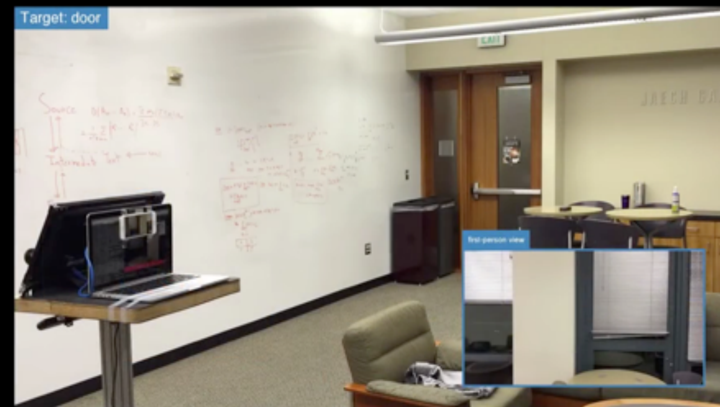


Learning to Walk

X: States → f(X) → Y: Action

Navigation Robot



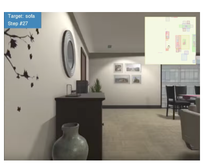
Y. Zhu, et. al. Target-driven Visual Navigation in Indoor Scenes using Deep Reinforcement Learning. ICRA 2017.

Navigation Robot

X: Goal & Observation → f(X)

Goal

Observation

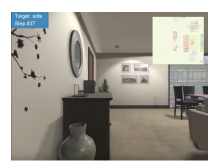


Navigation Robot

X: Goal & Observation → f(X) → Y: Action

Goal

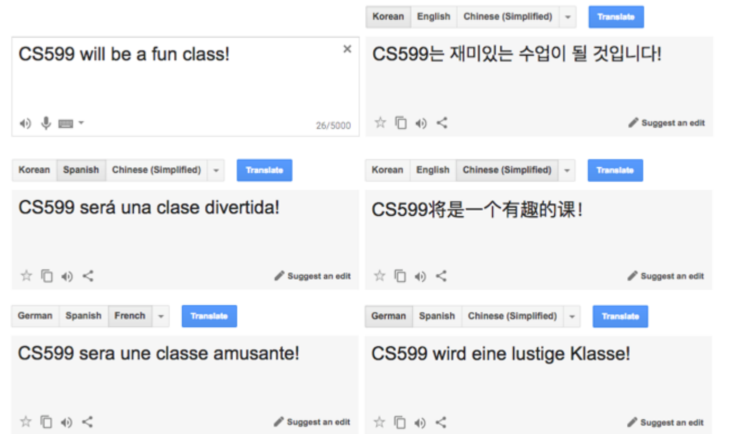
Observation



turn left

Practice

Machine Translation



Google Translate

Machine Translation

X:
Text in
Language A → f(X)

Machine Translation

X:
Text in
Language A → f(X) → Y:
Text in
Language B

Visual Question Answering (VQA)

From http://www.visualqa.org/

Visual Question Answering (VQA)

X: Image and Query → f(X)

Who is holding the kite?

Visual Question Answering (VQA)

X: Image and Query → f(X) → Y: Answer

Who is holding the kite?

Answer: Man

Object Picking Robot

S. Levine, et. al. Learning Hand-Eye Coordination for Robotic Grasping with Deep Learning and Large-Scale Data Collection.. IJRR 2017.

Object Picking Robot

X: Observation → f(X)

Object Picking Robot

X: Observation → f(X) → Y: Action

Autonomous Driving

From AutoX

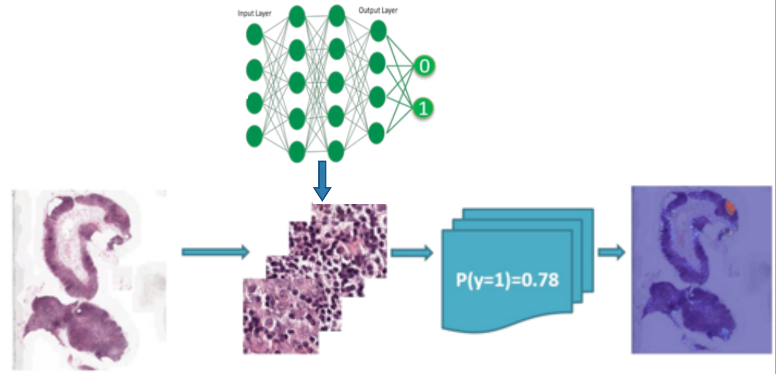
Autonomous Driving

X: Camera Input, Speed, etc → f(X)

Autonomous Driving

X:
Camera Input,
Speed, etc → f(X) → Y:
Action

Cancer Metastases Detection



From PathAI's submission to CAMELYON16.

Cancer Metastases Detection

X:
Image → f(X)

Cancer Metastases Detection

X:
Image → f(X) → Y:
Prevalence

It's matter of one function

X ⟶ f(X) ⟶ Y

Ok. It is true…

It's matter of one function

Question

- How is this related to intelligence?
- What does it mean to have one function for all intelligence?

X ⟶ f(X) ⟶ Y

Ok. It is true…

What's the challenge then?

X ⟶ f(X) ⟶ Y

What's the challenge then?

X ⟶ f(X) ⟶ Y

How do we learn this function?

Types of Learning

X → f(X) → Y

- Supervised Learning
- Unsupervised Learning
- Weakly / Semi-supervised Learning
- Reinforcement Learning

Definition from Dhruv Batra's deep learning course (ECE 5604)

Types of Learning

X → f(X) → Y

- Supervised Learning — desired output (**Y**) in training data
- Unsupervised Learning
- Weakly / Semi-supervised Learning
- Reinforcement Learning

Definition from Dhruv Batra's deep learning course (ECE 5604)

Types of Learning

X → f(X) → Y

- Supervised Learning — desired output (**Y**) in training data
- Unsupervised Learning — **Y** not in training data
- Weakly / Semi-supervised Learning
- Reinforcement Learning

Definition from Dhruv Batra's deep learning course (ECE 5604)

Types of Learning

X → f(X) → Y

- Supervised Learning — desired output (**Y**) in training data
- Unsupervised Learning — **Y** not in training data
- Weakly / Semi-supervised Learning — some of **Y** in training data
- Reinforcement Learning

Definition from Dhruv Batra's deep learning course (ECE 5604)

Types of Learning

X → f(X) → Y

- Supervised Learning — desired output (**Y**) in training data
- Unsupervised Learning — **Y** not in training data
- Weakly / Semi-supervised Learning — some of **Y** in training data
- Reinforcement Learning — rewards based on a set of actions

Definition from Dhruv Batra's deep learning course (ECE 5604)

Our goal is "to approximate"

X → f(X) → Y

There may exist an exact function (f*) mapping from X to Y.

Our goal is not to find this exact function.

Rather, we are happy as long as f(X) can **approximate** f*(x). f does NOT have to be exactly f*.

This course will be about

X → f(X) → Y

(1) How do we learn this function (using deep learning)?
(2) How to formulate a problem into this

Our goal is "to approximate"

X → f(X) → Y

For f(X) to approximate any f*(X), f is better to be highly capable.

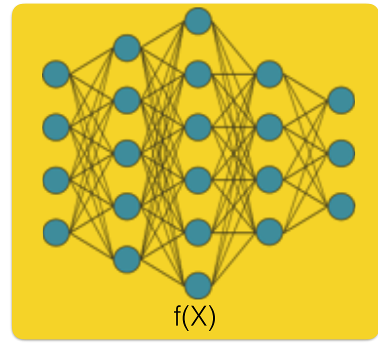
Our goal is "to approximate"

X ⟶ f(X) ⟶ Y

For f(X) to approximate any f*(X), f is better to be highly capable.

Deep learning is an effective method for this
- Non-linear (high capacity)
- Hierarchical
- End-to-End learning

Deep Learning is

X ⟶  ⟶ Y

- Non-linear (high capacity)
- Hierarchical
- End-to-End learning

Linear Classification

X ⟶ f(X) ⟶ Y

Let's first talk about learning a simple function.

Recap: Image classification

X ⟶ f(X) ⟶ Y

50x50x3 numbers
(7500 numbers)

One-hot class vector
(e.g. <0,0,0,1,0>)

Modified from CS 231N @ Stanford

Parametric Approach

$f(x; W) = Wx$

X → $f(X; W)$ → Y

50x50x3 numbers
(7500 numbers)

Weights
(that we want to learn)

One-hot class vector
(e.g. <0,0,0,1,0>)

Modified from CS 231N @ Stanford

Linear Classification

$f(x; W) = Wx$

X → $f(X; W)$ → Y

50x50x3 numbers
(7500 numbers)

Weights
(that we want to learn)

One-hot class vector
(e.g. <0,0,0,1,0>)

Modified from CS 231N @ Stanford

Linear Classification

$$\underbrace{f(x; W)}_{5\text{x}1} = \underbrace{W}_{5\text{x}7500}\underbrace{x}_{7500\text{x}1}$$

X → $f(X; W)$ → Y

50x50x3 numbers
(7500 numbers)

Weights
(that we want to learn)

One-hot class vector
(e.g. <0,0,0,1,0>)

Modified from CS 231N @ Stanford

Linear Classification

$$\underbrace{f(x; W)}_{5\text{x}1} = \underbrace{W}_{5\text{x}7500}\underbrace{x}_{7500\text{x}1} + \underbrace{b}_{5\text{x}1}$$

X → $f(X; W)$ → Y

50x50x3 numbers
(7500 numbers)

Weights
(that we want to learn)

One-hot class vector
(e.g. <0,0,0,1,0>)

Modified from CS 231N @ Stanford

Linear Classification

$$\overset{5\times1}{\boxed{f(x; W)}} = \overset{5\times7500}{\boxed{W}}\overset{7500\times1}{\boxed{x}} + \overset{5\times1}{\boxed{b}}$$

X → f(X; W) → Y

50x50x3 numbers
(7500 numbers)

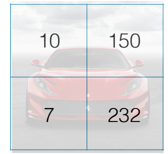
Weights
(that we want to learn)

One-hot class vector
(e.g. <0,0,0,1,0>)

Y: output
X: input
W, b: **learned weight**

Modified from CS 231N @ Stanford

Linear Classification

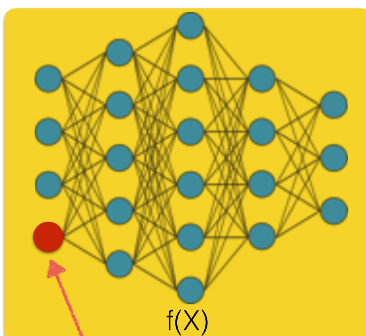


$$\underset{W}{\begin{bmatrix}0.2&-0.3&0&0.6\\-1&0.4&0.2&0.7\\-0.9&0.8&0.2&0.3\end{bmatrix}}\underset{x}{\begin{bmatrix}10\\150\\7\\232\end{bmatrix}}+\underset{b}{\begin{bmatrix}0.2\\-1\\-0.9\end{bmatrix}}=\underset{f(x;\,W,b)}{\begin{bmatrix}96.4\\212.8\\181.1\end{bmatrix}}$$

Modified from CS 231N @ Stanford

Linear Classification

X → f(X) → Y



Apparently, this is an unit used in deep neural networks too.

Today's agenda

- CSCI 599 overview
- Learning 101
- Loss function & Optimization
- Neural Networks

Recap:Our course

x ⟶ f(x) ⟶ y

(1) How do we learn this function (using deep learning)?
(2) How to formulate a problem into this

Loss function & Optimization

x ⟶ f(x) ⟶ y

Loss function & Optimization

x ⟶ f(x; W) ⟶ y

Showing W (to learn) explicitly
Parametric function

Loss function & Optimization

x ⟶ f(x; W) ⟶ y

Showing W (to learn) explicitly
Parametric function

- **Loss function (L)** measures how well learned W can map X to Y (compared to f*).

Loss function & Optimization

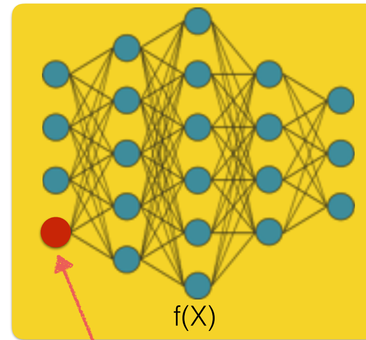
x → f(x; W) → y

Showing W (to learn) explicitly
Parametric function

- **Loss function (L)** measures how well learned W can map X to Y (compared to f*).
- **Optimization** finds the best W given a loss function L (i.e. finding W that minimizes L).

Recap: Linear Classification

X → f(X) → Y



Apparently, this is an unit used in deep neural networks too.

Recap: Linear Classification

$$\underbrace{f(x; W)}_{5\times1} = \underbrace{W}_{5\times7500}\underbrace{x}_{7500\times1} + \underbrace{b}_{5\times1}$$

X → f(X; W) → Y

50x50x3 numbers
(7500 numbers)

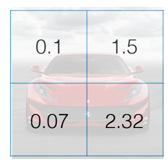
Weights
(that we want to learn)

One-hot class vector
(e.g. <0,0,0,1,0>)

Y: output
X: input
W, b: **learned weight**

Modified from CS 231N @ Stanford

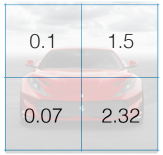
Recap: Linear Classification



$$\underbrace{\begin{bmatrix} 0.1 & -0.15 & 0 & 0.3 \\ -0.5 & 0.2 & 0.1 & 0.35 \\ -0.45 & 0.4 & 0.1 & 0.15 \end{bmatrix}}_{W} \underbrace{\begin{bmatrix} 0.1 \\ 1.5 \\ 0.07 \\ 2.32 \end{bmatrix}}_{x} + \underbrace{\begin{bmatrix} 0.15 \\ -0.1 \\ -0.2 \end{bmatrix}}_{b} = \underbrace{\begin{bmatrix} 0.631 \\ 0.969 \\ 0.71 \end{bmatrix}}_{f(x;\, W,b)} \begin{array}{l} \text{Cat} \\ \text{Car} \\ \text{Airplane} \end{array}$$

Modified from CS 231N @ Stanford

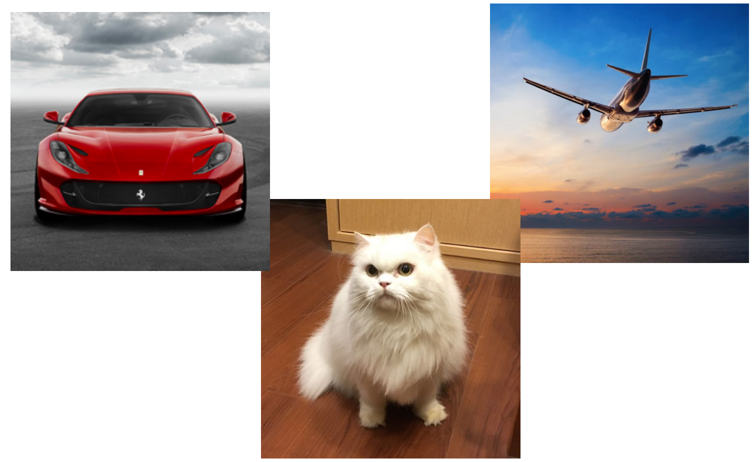
Recap: Linear Classification



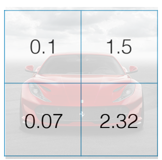
$$\underbrace{\begin{bmatrix} 0.1 & -0.15 & 0 & 0.3 \\ -0.5 & 0.2 & 0.1 & 0.35 \\ -0.45 & 0.4 & 0.1 & 0.15 \end{bmatrix}}_{W} \underbrace{\begin{bmatrix} 0.1 \\ 1.5 \\ 0.07 \\ 2.32 \end{bmatrix}}_{x} + \underbrace{\begin{bmatrix} 0.15 \\ -0.1 \\ -0.2 \end{bmatrix}}_{b} = \underbrace{\begin{bmatrix} 0.631 \\ \boxed{0.969} \\ 0.71 \end{bmatrix}}_{f(x;\, W,b)} \begin{array}{l} \text{Cat} \\ \text{Car} \\ \text{Airplane} \end{array}$$

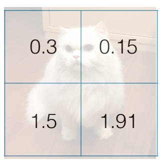
Modified from CS 231N @ Stanford

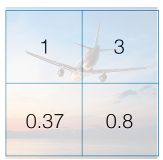
Linear Classification



Linear Classification



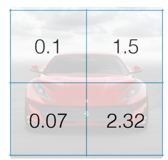




$$W = \begin{bmatrix} 0.1 & -0.15 & 0 & 0.3 \\ -0.5 & 0.2 & 0.1 & 0.35 \\ -0.45 & 0.4 & 0.1 & 0.15 \end{bmatrix} \qquad b = \begin{bmatrix} 0.15 \\ -0.1 \\ -0.2 \end{bmatrix}$$

f(x; W,b)

Linear Classification



$$\underbrace{\begin{bmatrix} 0.1 & -0.15 & 0 & 0.3 \\ -0.5 & 0.2 & 0.1 & 0.35 \\ -0.45 & 0.4 & 0.1 & 0.15 \end{bmatrix}}_{W} \underbrace{\begin{bmatrix} 0.1 \\ 1.5 \\ 0.07 \\ 2.32 \end{bmatrix}}_{x} + \underbrace{\begin{bmatrix} 0.15 \\ -0.1 \\ -0.2 \end{bmatrix}}_{b} = \underbrace{\begin{bmatrix} 0.631 \\ \boxed{0.969} \\ 0.71 \end{bmatrix}}_{f(x;\, W,b)} \begin{array}{l} \text{Cat} \\ \text{Car} \\ \text{Airplane} \end{array}$$

Linear Classification

0.3	0.15
1.5	1.91

$$\underbrace{\begin{bmatrix}0.1 & -0.15 & 0 & 0.3\\ -0.5 & 0.2 & 0.1 & 0.35\\ -0.45 & 0.4 & 0.1 & 0.15\end{bmatrix}}_{W}\underbrace{\begin{bmatrix}0.3\\ 0.15\\ 1.5\\ 1.91\end{bmatrix}}_{x}+\underbrace{\begin{bmatrix}0.15\\ -0.1\\ -0.2\end{bmatrix}}_{b}=\underbrace{\begin{bmatrix}\boxed{0.73}\\ 0.598\\ 0.161\end{bmatrix}}_{f(x;\, W,b)}\begin{array}{l}\text{Cat}\\ \text{Car}\\ \text{Airplane}\end{array}$$

Linear Classification

1	3
0.37	0.8

$$\underbrace{\begin{bmatrix}0.1 & -0.15 & 0 & 0.3\\ -0.5 & 0.2 & 0.1 & 0.35\\ -0.45 & 0.4 & 0.1 & 0.15\end{bmatrix}}_{W}\underbrace{\begin{bmatrix}1\\ 3\\ 0.37\\ 0.8\end{bmatrix}}_{x}+\underbrace{\begin{bmatrix}0.15\\ -0.1\\ -0.2\end{bmatrix}}_{b}=\underbrace{\begin{bmatrix}0.04\\ 0.317\\ \boxed{0.707}\end{bmatrix}}_{f(x;\, W,b)}\begin{array}{l}\text{Cat}\\ \text{Car}\\ \text{Airplane}\end{array}$$

Linear Classification

0.1	1.5
0.07	2.32

0.631
0.969
0.71

0.3	0.15
1.5	1.91

0.73
0.598
0.161

1	3
0.37	0.8

0.04
0.317
0.707

$$W = \begin{bmatrix}0.1 & -0.15 & 0 & 0.3\\ -0.5 & 0.2 & 0.1 & 0.35\\ -0.45 & 0.4 & 0.1 & 0.15\end{bmatrix}\qquad b = \begin{bmatrix}0.15\\ -0.1\\ -0.2\end{bmatrix}$$

Cat Car Airplane

Linear Classification

0.1	1.5
0.07	2.32

0.631
0.969
0.71

0.3	0.15
1.5	1.91

0.73
0.598
0.161

1	3
0.37	0.8

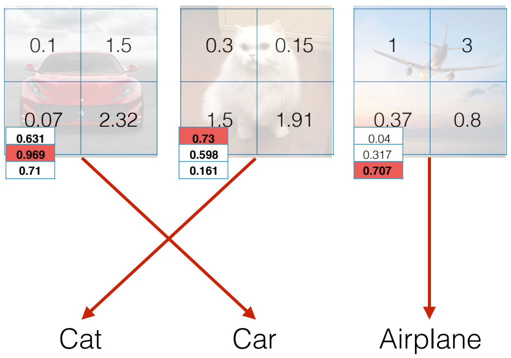
0.04
0.317
0.707

$$W = \begin{bmatrix}0.1 & -0.15 & 0 & 0.3\\ -0.5 & 0.2 & 0.1 & 0.35\\ -0.45 & 0.4 & 0.1 & 0.15\end{bmatrix}\qquad b = \begin{bmatrix}0.15\\ -0.1\\ -0.2\end{bmatrix}$$

Cat Car Airplane

Looks perfect! Doesn't it?

Linear Classification



0.1	-0.15	0	0.3
-0.5	0.2	0.1	0.35
-0.45	0.4	0.1	0.15

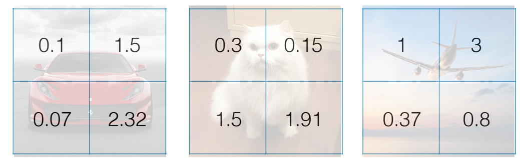
W

0.15
-0.1
-0.2

b

But… What if we have different W or b?

Linear Classification



-0.25	0.1	0.05	0.175
0.05	-0.075	0	0.15
-0.225	0.2	0.05	0.075

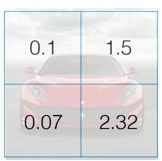
W

f(x; W,b)

0.35
0.1
0

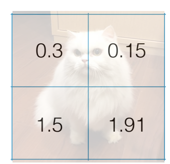
b

Linear Classification



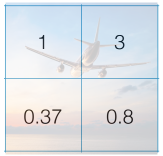
$$\underbrace{\begin{bmatrix} -0.25 & 0.1 & 0.05 & 0.175 \\ 0.05 & -0.075 & 0 & 0.15 \\ -0.225 & 0.2 & 0.05 & 0.075 \end{bmatrix}}_{W} \underbrace{\begin{bmatrix} 0.1 \\ 1.5 \\ 0.07 \\ 2.32 \end{bmatrix}}_{x} + \underbrace{\begin{bmatrix} 0.35 \\ 0.1 \\ 0 \end{bmatrix}}_{b} = \underbrace{\begin{bmatrix} \boxed{0.884} \\ 0.34 \\ 0.455 \end{bmatrix}}_{f(x;\, W,b)} \begin{matrix} \text{Cat} \\ \text{Car} \\ \text{Airplane} \end{matrix}$$

Linear Classification



$$\underbrace{\begin{bmatrix} -0.25 & 0.1 & 0.05 & 0.175 \\ 0.05 & -0.075 & 0 & 0.15 \\ -0.225 & 0.2 & 0.05 & 0.075 \end{bmatrix}}_{W} \underbrace{\begin{bmatrix} 0.3 \\ 0.15 \\ 1.5 \\ 1.91 \end{bmatrix}}_{x} + \underbrace{\begin{bmatrix} 0.35 \\ 0.1 \\ 0 \end{bmatrix}}_{b} = \underbrace{\begin{bmatrix} \boxed{0.699} \\ 0.39 \\ 0.181 \end{bmatrix}}_{f(x;\, W,b)} \begin{matrix} \text{Cat} \\ \text{Car} \\ \text{Airplane} \end{matrix}$$

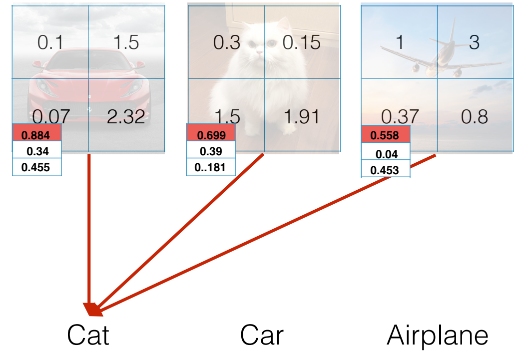
Linear Classification



-0.25	0.1	0.05	0.175
0.05	-0.075	0	0.15
-0.225	0.2	0.05	0.075

$$\underbrace{\begin{bmatrix} -0.25 & 0.1 & 0.05 & 0.175 \\ 0.05 & -0.075 & 0 & 0.15 \\ -0.225 & 0.2 & 0.05 & 0.075 \end{bmatrix}}_{W} \underbrace{\begin{bmatrix} 1 \\ 3 \\ 0.37 \\ 0.8 \end{bmatrix}}_{x} + \underbrace{\begin{bmatrix} 0.35 \\ 0.1 \\ 0 \end{bmatrix}}_{b} = \underbrace{\begin{bmatrix} \boxed{0.558} \\ 0.04 \\ 0.453 \end{bmatrix}}_{f(x;\, W,b)} \begin{matrix} \text{Cat} \\ \text{Car} \\ \text{Airplane} \end{matrix}$$

Linear Classification



-0.25	0.1	0.05	0.175
0.05	-0.075	0	0.15
-0.225	0.2	0.05	0.075

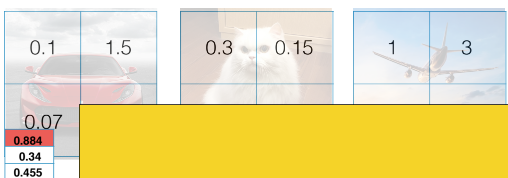
W

0.35
0.1
0

b

Oops…

Linear Classification



-0.25	0.1	0.05	0.175
0.05	-0.075	0	0.15
5	0.075		

W matters!

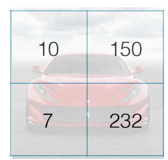
Cat Car Airplane

0

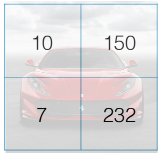
b

Oops…

Loss functions



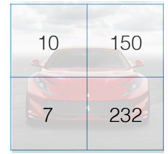
Loss functions



Cat	0
Car	1
Airplane	0

Ground truth

Loss functions



W1

0.1	-0.15	0	0.3
-0.5	0.2	0.1	0.35
-0.45	0.4	0.1	0.15

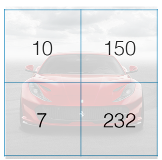
W2

-0.25	0.1	0.05	0.175
0.05	-0.075	0	0.15
-0.225	0.2	0.05	0.075

Cat	0
Car	1
Airplane	0

Ground truth

Loss functions



W1

0.1	-0.15	0	0.3
-0.5	0.2	0.1	0.35
-0.45	0.4	0.1	0.15

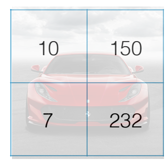
W2

-0.25	0.1	0.05	0.175
0.05	-0.075	0	0.15
-0.225	0.2	0.05	0.075

Cat	0	0.631	0.884
Car	1	0.969	0.34
Airplane	0	0.71	0.455

Ground truth

Loss functions



W1

0.1	-0.15	0	0.3
-0.5	0.2	0.1	0.35
-0.45	0.4	0.1	0.15

W2

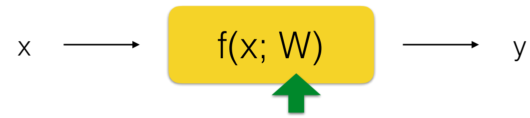
-0.25	0.1	0.05	0.175
0.05	-0.075	0	0.15
-0.225	0.2	0.05	0.075

Cat	0	0.631	0.884
Car	1	0.969	0.34
Airplane	0	0.71	0.455

Ground truth

Which one is better?

Loss functions

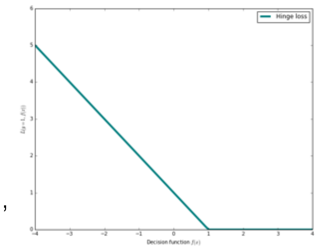


Showing W (to learn) explicitly
Parametric function

L measures how well learned W can map X to Y.

- Hinge Loss
- L_1, L_2 Loss
- Cross-Entropy

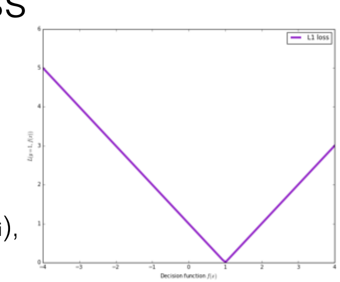
Hinge Loss



- Given f(x; W,b) & examples (x_i, y_i), minimize:

$$Loss = \frac{1}{N}\sum_{i=1}^{N}\sum_{j\neq y_i} max(0, 1 + f(x_i; W, b)_j - f(x_i; W, b)_{y_i})$$

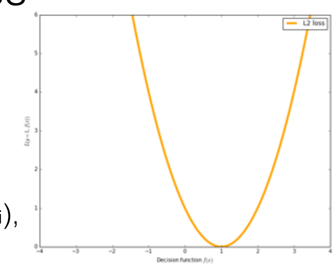
L_1 Loss



- Given f(x; W,b) & examples (x_i, y_i), minimize:

$$Loss = \frac{1}{N}\sum_{i=1}^{N} \| y_i - f(x_i; W, b)\|_1$$

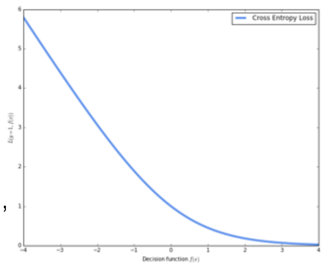
L_2 Loss



- Given f(x; W,b) & examples (x_i, y_i), minimize:

$$Loss = \frac{1}{N}\sum_{i=1}^{N} \| y_i - f(x_i; W, b)\|_2^2$$

Cross-Entropy Loss

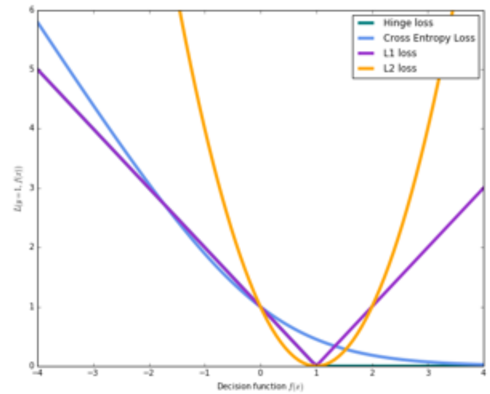


- Given f(x; W,b) & examples (x_i, y_i), minimize:

$$Loss = -\frac{1}{N}\sum_{i=1}^{N} \hat{y_i}\, log(P(y_i = j\,|x_i))$$

$$P(y_i = j\,|x_i) = \frac{e^{f(x_i;W,b)}}{\sum_{j=1}^{C} e^{f(x_j;W,b)}}$$

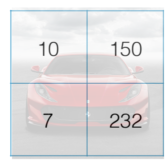
Loss functions



Each loss function penalizes differently.

Does a loss function matter?

Does a loss function matter?



W_1

W_{11}	W_{11}	W_{11}	W_{11}
W_{11}	W_{11}	W_{11}	W_{11}
W_{11}	W_{11}	W_{11}	W_{11}

W_2

W_{21}	W_{21}	W_{21}	W_{21}
W_{21}	W_{21}	W_{21}	W_{21}
W_{21}	W_{21}	W_{21}	W_{21}

	Ground truth	W_1	W_2
Cat	0	0	0.07
Car	1	1.2	0.9
Airplane	0	0	0.05

Which one is better?

Does a loss function matter?

	f(x; W1)	f(x; W2)
L_1-loss	0.2	0.22

	Ground truth		
Cat	0	0	0.07
Car	1	1.2	0.9
Airplane	0	0	0.05

Ground truth

Which one is better?

Does a loss function matter?

	f(x; W1)	f(x; W2)
L_1-loss	0.2	0.22
L_2-loss	0.04	0.0174

	Ground truth		
Cat	0	0	0.07
Car	1	1.2	0.9
Airplane	0	0	0.05

Ground truth

Which one is better?

Does a loss function matter?

	f(x; W1)	f(x; W2)
L_1-loss	0.2	0.22
L_2-loss	0.04	0.0174

Loss function matters!

	Ground truth		
Cat	0	0	0.07
Car	1	1.2	0.9
Airplane	0	0	0.05

Ground truth

Which one is better?

Regularization

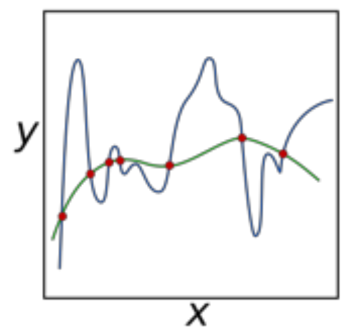
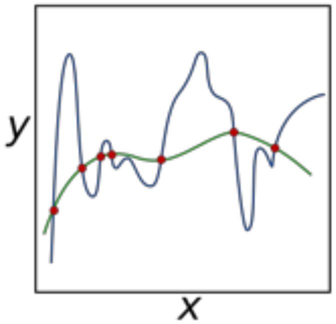


Image from Wikipedia

Regularization



Ockham's razor:
"Among competing hypotheses, the simplest is the best."
William of Ockham, 1285 - 1347

Image from Wikipedia

Regularization

Data loss Regularization

$$L(W) = \frac{1}{N}\sum_{i=1}^{N} \| y_i - f(x_i; W, b) \|_2^2 + \lambda R(W)$$

Ockham's razor:
"Among competing hypotheses, the simplest is the best."
William of Ockham, 1285 - 1347

Regularization

Data loss Regularization

$$L(W) = \underbrace{\frac{1}{N}\sum_{i=1}^{N} \| y_i - f(x_i; W, b) \|_2^2}_{\text{fitting to the data}} + \lambda R(W)$$

Ockham's razor:
"Among competing hypotheses, the simplest is the best."
William of Ockham, 1285 - 1347

Regularization

Data loss Regularization

$$L(W) = \underbrace{\frac{1}{N}\sum_{i=1}^{N} \| y_i - f(x_i; W, b) \|_2^2}_{\text{fitting to the data}} + \underbrace{\lambda R(W)}_{\text{preferring a simple model}}$$

Ockham's razor:
"Among competing hypotheses, the simplest is the best."
William of Ockham, 1285 - 1347

Optimization

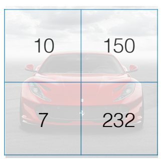
Find W that **minimize a loss function** (L).

Optimization

Find W that **minimize a loss function** (L).

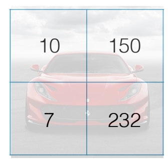
Note that we are finding W based on L, **NOT** f!

Optimization



x

Optimization

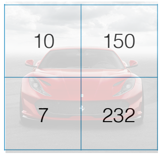


x

0
1
0

y* (ideal output)

Optimization



x

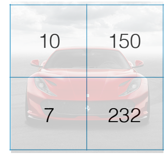
?	?	?	?
?	?	?	?
?	?	?	?

W

0
1
0

y* (ideal output)

Optimization



x

?	?	?	?
?	?	?	?
?	?	?	?

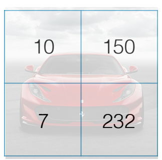
W

0
1
0

y* (ideal output)

f(x; W)

Optimization



x

?	?	?	?
?	?	?	?
?	?	?	?

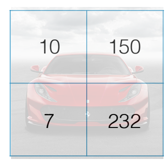
W

0
1
0

y* (ideal output)

f(x; W) $\rightarrow$ y*

Optimization



x

?	?	?	?
?	?	?	?
?	?	?	?

W

0
1
0

y* (ideal output)

f(x; W) $\rightarrow$ y*

L(W; f,x,y*)

Optimization

How do we find W minimizing L?

* Random search
* Analytic solution
* Numerical approach (gradient descent)

ut)

f(x; W) → y*

L(W; f,x,y*)

Optimization

How do we find W minimizing L?

* Random search
* Analytic solution
* Numerical approach (gradient descent)

ut)

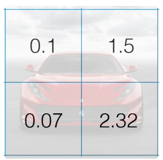
f(x; W) → y*

L(W; f,x,y*)

As an example, let's pick a simple loss function.

$L(W) = \| f(x; W) - y^* \|_2$

1. Random Search



x

?	?	?	?
?	?	?	?
?	?	?	?

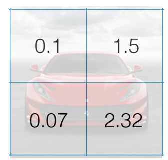
W

0
1
0

y* (ideal output)

$L(W) = \| f(x; W) - y^* \|_2$

1. Random Search



x

0.05	0.15	-0.1	0.2
-0.1	0.2	0.25	0.05
0.4	-0.15	0.25	0.15

W

$L(W) = \| f(x; W) - y^* \|_2$

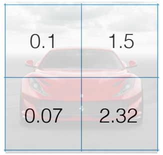
0.987
0.223
0.38

f(x; W)

0
1
0

y* (ideal output)

1. Random Search



x

-0.1	0.2	0.15	0.05
0.25	0.05	0.2	-0.1
0.25	0.15	-0.15	0.4

W

L(W) = || f(x; W) - y* ||$_2$

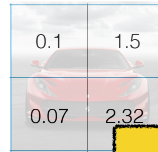
0.416
-0.118
1.167

f(x; W)

0
1
0

y* (ideal output)

1. Random Search



x

?	?	?	?
?	?	?	?

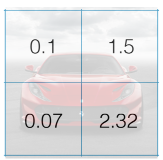
0
1
0

l output)

L(W) = ||

Too slow

2. Analytic Solution



x

?	?	?	?
?	?	?	?
?	?	?	?

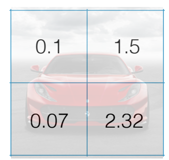
W

0
1
0

y* (ideal output)

L(W) = || f(x; W) - y* ||$_2$

2. Analytic Solution



x

?	?	?	?
?	?	?	?
?	?	?	?

W

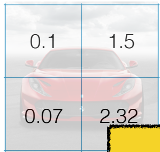
0
1
0

y* (ideal output)

L(W) = || f(x; W) - y* ||$_2$

$$\min_W L(W) = (X^T X)^{-1} X^T y^*$$

2. Analytic Solution



x

?	?	?	?
?	?	?	?

0
1
0

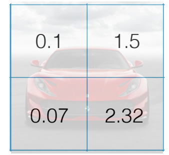
l output)

L(W) = ||

Very hard to find.

W

3. Numerical Solution (gradient descent)



x

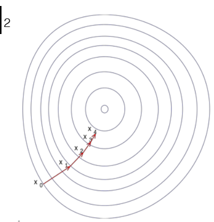
?	?	?	?
?	?	?	?
?	?	?	?

W

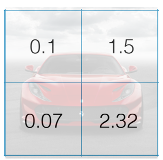
0
1
0

y* (ideal output)

$L(W) = \| f(x; W) - y^* \|_2$



3. Numerical Solution (gradient descent)



x

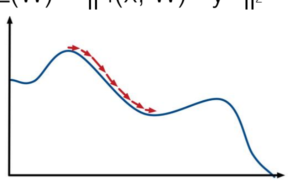
?	?	?	?
?	?	?	?
?	?	?	?

W

0
1
0

y* (ideal output)

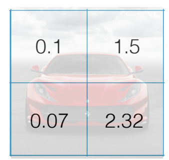
$L(W) = \| f(x; W) - y^* \|_2$



The derivative of a function in one-dimension

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

3. Numerical Solution (gradient descent)



x

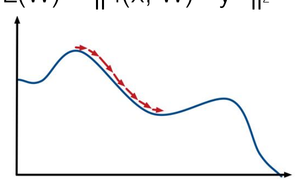
?	?	?	?
?	?	?	?
?	?	?	?

W

0
1
0

y* (ideal output)

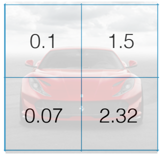
$L(W) = \| f(x; W) - y^* \|_2$



The **derivative** of a function in one-dimension

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

3. Numerical Solution (gradient descent)



x

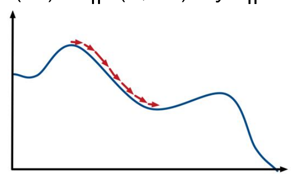
?	?	?	?
?	?	?	?
?	?	?	?

W

0
1
0

y* (ideal output)

L(W) = || f(x; W) - y* ||$_2$



The **derivative** of a function in one-dimension

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

The **gradient** is the vector of partial derivatives in a higher dimension

3. Numerical Solution (gradient descent)

-0.1	0.2	0.15	0.05
0.25	0.05	0.2	-0.1
0.25	0.15	-0.15	0.4

W

?	?	?	?
?	?	?	?
?	?	?	?

gradient (dW)

L(W) = 1.6688

3. Numerical Solution (gradient descent)

-0.1	0.2	0.15	0.05
0.25	0.05	0.2	-0.1
0.25	0.15	-0.15	0.4

W

-0.1+ 0.001	0.2	0.15	0.05
0.25	0.05	0.2	-0.1
0.25	0.15	-0.15	0.4

W+h

?	?	?	?
?	?	?	?
?	?	?	?

gradient (dW)

L(W) = 1.6688

3. Numerical Solution (gradient descent)

-0.1	0.2	0.15	0.05
0.25	0.05	0.2	-0.1
0.25	0.15	-0.15	0.4

W

-0.1+ 0.001	0.2	0.15	0.05
0.25	0.05	0.2	-0.1
0.25	0.15	-0.15	0.4

W+h

?	?	?	?
?	?	?	?
?	?	?	?

gradient (dW)

L(W) = 1.6688

0.416
-0.118
1.167

f(x; W+h)

0
1
0

y* (ideal output)

3. Numerical Solution (gradient descent)

W

-0.1	0.2	0.15	0.05
0.25	0.05	0.2	-0.1
0.25	0.15	-0.15	0.4

W+h

-0.1+ 0.001	0.2	0.15	0.05
0.25	0.05	0.2	-0.1
0.25	0.15	-0.15	0.4

gradient (dW)

?	?	?	?
?	?	?	?
?	?	?	?

L(W) = 1.6688

f(x; W+h)

0.416
-0.118
1.167

L(W+h) = 1.6693

y* (ideal output)

0
1
0

3. Numerical Solution (gradient descent)

W

-0.1	0.2	0.15	0.05
0.25	0.05	0.2	-0.1
0.25	0.15	-0.15	0.4

W+h

-0.1+ 0.001	0.2	0.15	0.05
0.25	0.05	0.2	-0.1
0.25	0.15	-0.15	0.4

gradient (dW)

?	?	?	?
?	?	?	?
?	?	?	?

L(W) = 1.6688 L(W+h) = 1.6693

3. Numerical Solution (gradient descent)

W

-0.1	0.2	0.15	0.05
0.25	0.05	0.2	-0.1
0.25	0.15	-0.15	0.4

W+h

-0.1+ 0.001	0.2	0.15	0.05
0.25	0.05	0.2	-0.1
0.25	0.15	-0.15	0.4

gradient (dW)

?	?	?	?
?	?	?	?
?	?	?	?

L(W) = 1.6688 L(W+h) = 1.6693

Gradient =

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

3. Numerical Solution (gradient descent)

W

-0.1	0.2	0.15	0.05
0.25	0.05	0.2	-0.1
0.25	0.15	-0.15	0.4

W+h

-0.1+ 0.001	0.2	0.15	0.05
0.25	0.05	0.2	-0.1
0.25	0.15	-0.15	0.4

gradient (dW)

?	?	?	?
?	?	?	?
?	?	?	?

L(W) = 1.6688 L(W+h) = 1.6693

Gradient = (1.6693-1.6688)/0.001 = 0.5

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

3. Numerical Solution (gradient descent)

W

-0.1	0.2	0.15	0.05
0.25	0.05	0.2	-0.1
0.25	0.15	-0.15	0.4

W+h

-0.1+ 0.001	0.2	0.15	0.05
0.25	0.05	0.2	-0.1
0.25	0.15	-0.15	0.4

gradient (dW)

0.5	?	?	?
?	?	?	?
?	?	?	?

L(W) = 1.6688   L(W+h) = 1.6693

Gradient = (1.6693-1.6688)/0.001 = 0.5

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

3. Numerical Solution (gradient descent)

W

-0.1	0.2	0.15	0.05
0.25	0.05	0.2	-0.1
0.25	0.15	-0.15	0.4

W+h

-0.1+ 0.001	**0.2+ 0.001**	0.15	0.05
0.25	0.05	0.2	-0.1
0.25	0.15	-0.15	0.4

gradient (dW)

0.5	**?**	?	?
?	?	?	?
?	?	?	?

L(W) = 1.6688

3. Numerical Solution (gradient descent)

W

-0.1	0.2	0.15	0.05
0.25	0.05	0.2	-0.1
0.25	0.15	-0.15	0.4

W+h

-0.1+ 0.001	**0.2+ 0.001**	0.15	0.05
0.25	0.05	0.2	-0.1
0.25	0.15	-0.15	0.4

gradient (dW)

0.5	**?**	?	?
?	?	?	?
?	?	?	?

L(W) = 1.6688

f(x; W+h)	y* (ideal output)
0.418	0
-0.118	1
1.167	0

3. Numerical Solution (gradient descent)

W

-0.1	0.2	0.15	0.05
0.25	0.05	0.2	-0.1
0.25	0.15	-0.15	0.4

W+h

-0.1+ 0.001	**0.2+ 0.001**	0.15	0.05
0.25	0.05	0.2	-0.1
0.25	0.15	-0.15	0.4

gradient (dW)

0.5	**?**	?	?
?	?	?	?
?	?	?	?

L(W) = 1.6688

f(x; W+h)	y* (ideal output)
0.418	0
-0.118	1
1.167	0

L(W+h) = 1.6697

3. Numerical Solution (gradient descent)

W

-0.1	0.2	0.15	0.05
0.25	0.05	0.2	-0.1
0.25	0.15	-0.15	0.4

W+h

-0.1+ 0.001	**0.2+ 0.001**	0.15	0.05
0.25	0.05	0.2	-0.1
0.25	0.15	-0.15	0.4

gradient (dW)

0.5	**?**	?	?
?	?	?	?
?	?	?	?

L(W) = 1.6688 L(W+h) = 1.6697

3. Numerical Solution (gradient descent)

W

-0.1	0.2	0.15	0.05
0.25	0.05	0.2	-0.1
0.25	0.15	-0.15	0.4

W+h

-0.1+ 0.001	**0.2+ 0.001**	0.15	0.05
0.25	0.05	0.2	-0.1
0.25	0.15	-0.15	0.4

gradient (dW)

0.5	**?**	?	?
?	?	?	?
?	?	?	?

L(W) = 1.6688 L(W+h) = 1.6697

Gradient = (1.6697-1.6688)/0.001 = 0.9

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

3. Numerical Solution (gradient descent)

W

-0.1	0.2	0.15	0.05
0.25	0.05	0.2	-0.1
0.25	0.15	-0.15	0.4

W+h

-0.1+ 0.001	**0.2+ 0.001**	0.15	0.05
0.25	0.05	0.2	-0.1
0.25	0.15	-0.15	0.4

gradient (dW)

0.5	**0.9**	?	?
?	?	?	?
?	?	?	?

L(W) = 1.6688 L(W+h) = 1.6697

Gradient = (1.6697-1.6688)/0.001 = 0.9

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

3. Numerical Solution (gradient descent)

Various optimization strategy for stability, accuracy, or speed

- Gradient Descent
- Stochastic gradient descent
- Adam optimization
- Batch and Minibatch algorithms

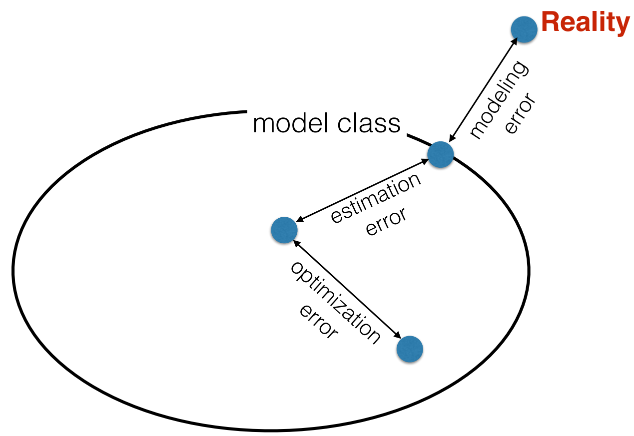
3. Numerical Solution (gradient descent)

- Gr
- Sto
- Ad
- Batch and Minibatch algorithms

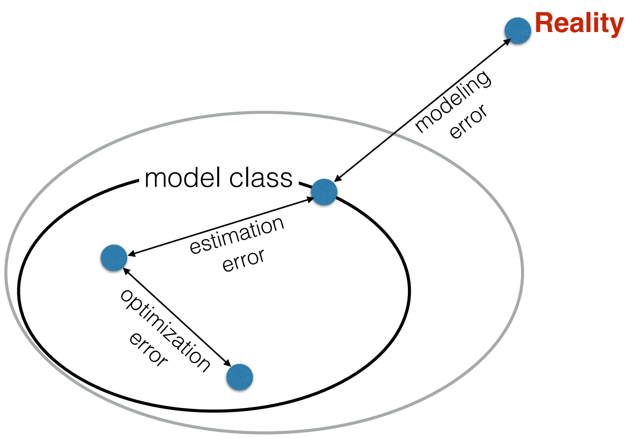
Issues

1. Local minima
2. Exploding gradient
3. Inexact gradient

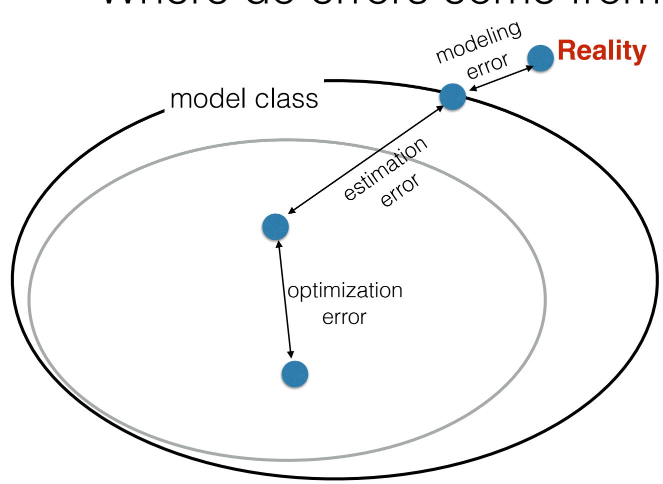
Where do errors come from?



Where do errors come from?



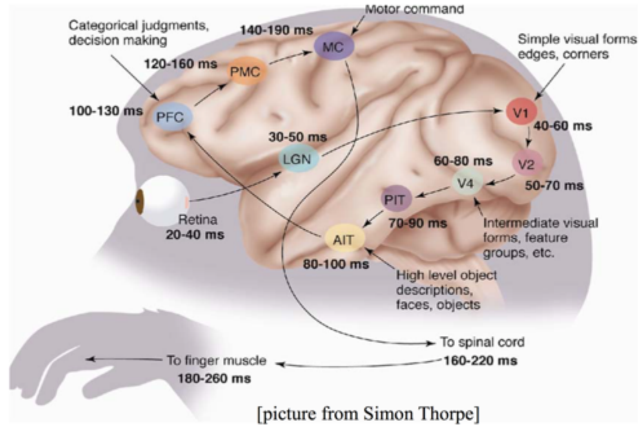
Where do errors come from?



Today's agenda

- CSCI 599 overview
- Learning 101
- Loss function & Optimization
- Neural Networks

Deep Learning is motivated by human brain



[picture from Simon Thorpe]

Slide credit: Marc'Aurelio Ranzato, Yann LeCun

Neuron

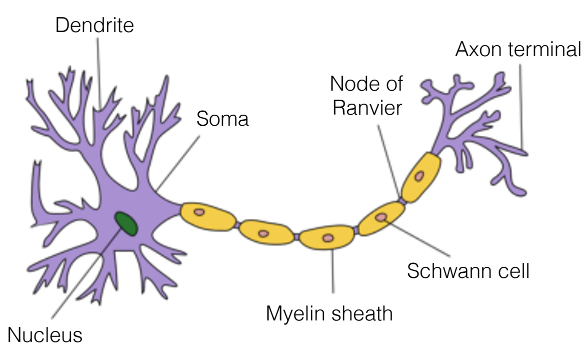


Image from Wikipedia

Neuron

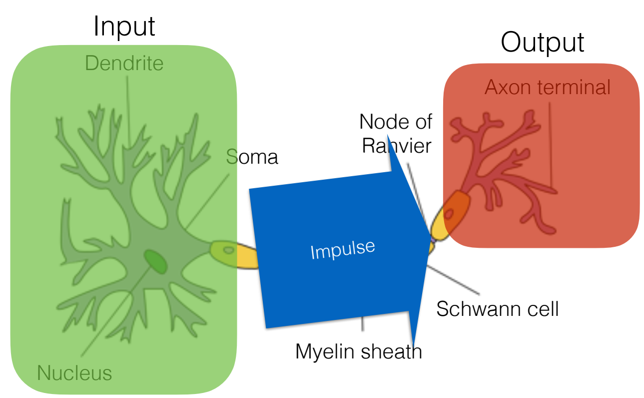
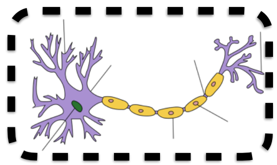
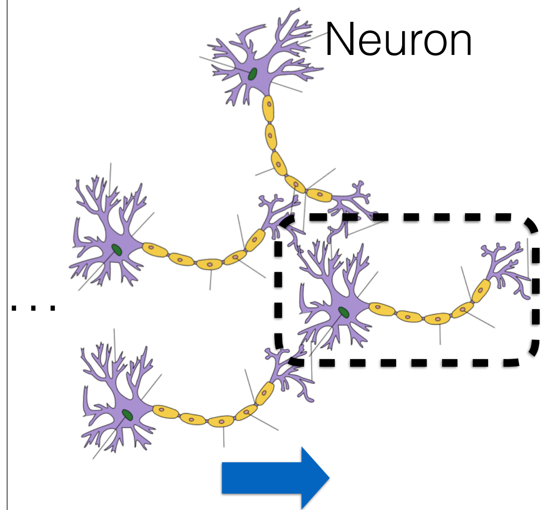


Image from Wikipedia

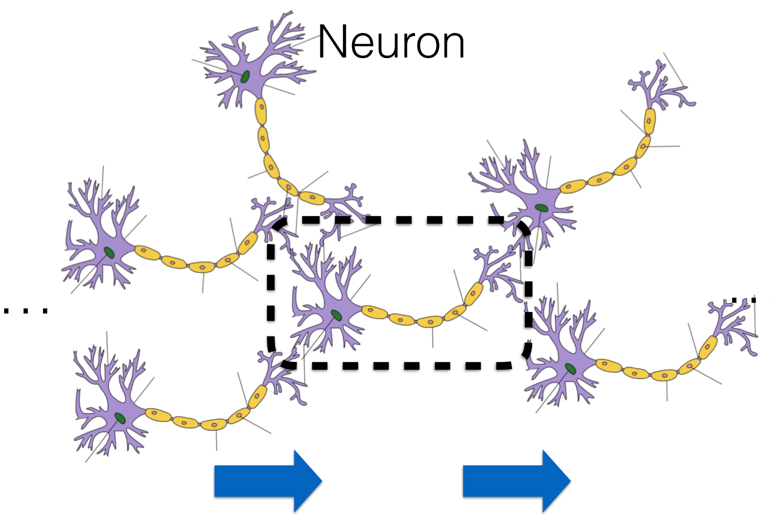
Neuron



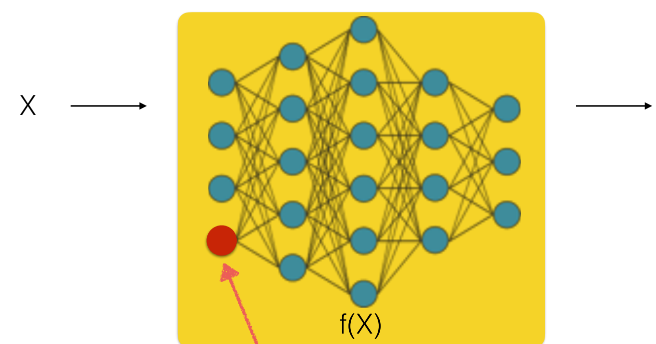
Neuron



Neuron



Neural Networks



Each unit represents “neuron”

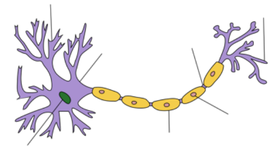
What is Neural Networks?

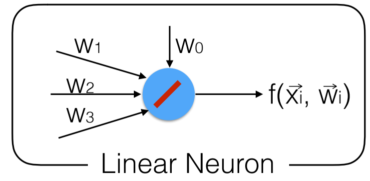
In simple terms,

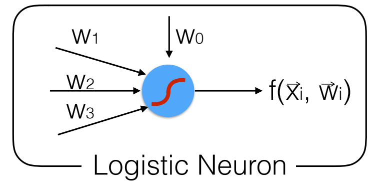
a set of **neurons (atomic functions)**

connected in a **non-linear** way

"Neuron"



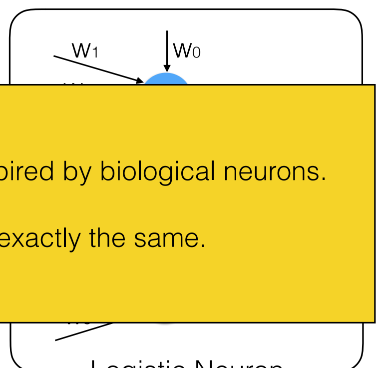




More neurons!

Modified from the HKUST slide

"Neuron"



Artificial Neurons are inspired by biological neurons.

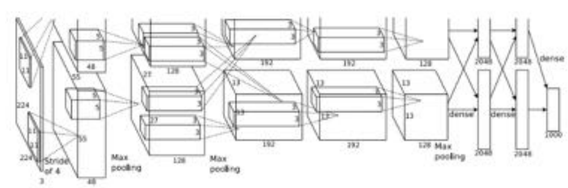
They are **NOT** exactly the same.

More neurons!

Modified from the HKUST slide

(Convolutional) Neural Networks

X →



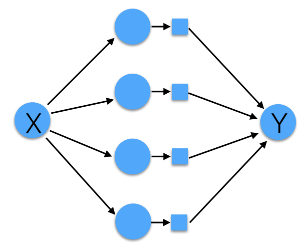
→ Y

A Krizhevsky, et. al. ImageNet Classification with Deep Convolutional Neural Networks. NIPS 2012.

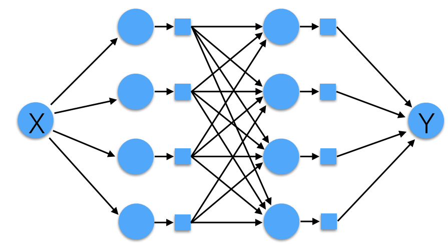
Deeper Neural Networks

Szegedy et. al., Going deeper with convolutions, CVPR 2015

Why “Deep Learning”?



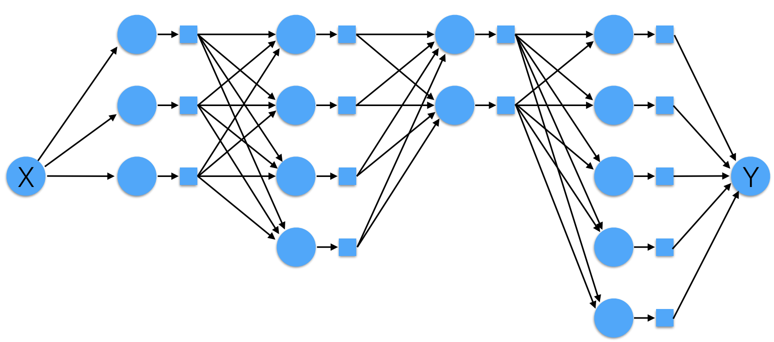
Shallow network



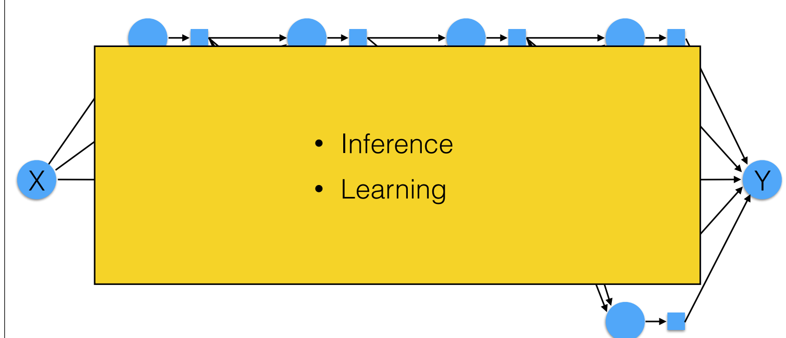
Deep network

The algorithm for training a **deep** network

Neural Networks Example

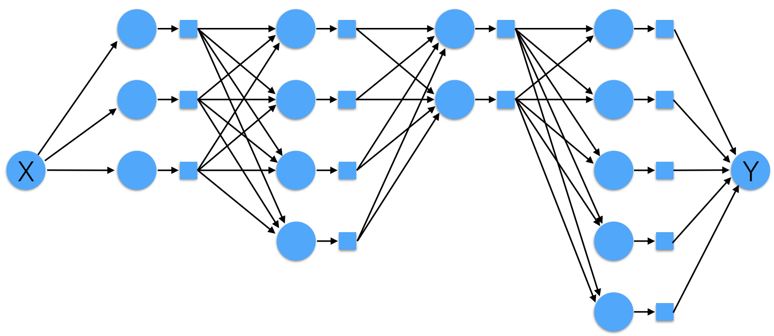


Neural Networks Example

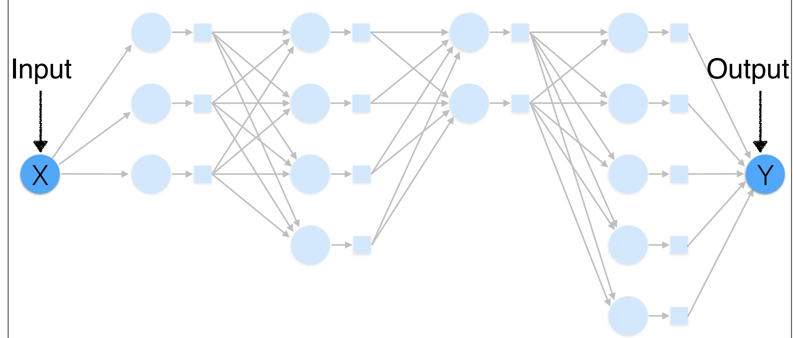


- Inference
- Learning

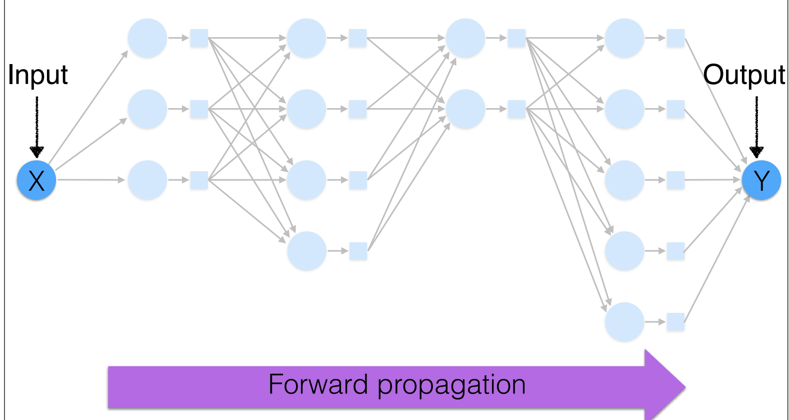
Forward propagation



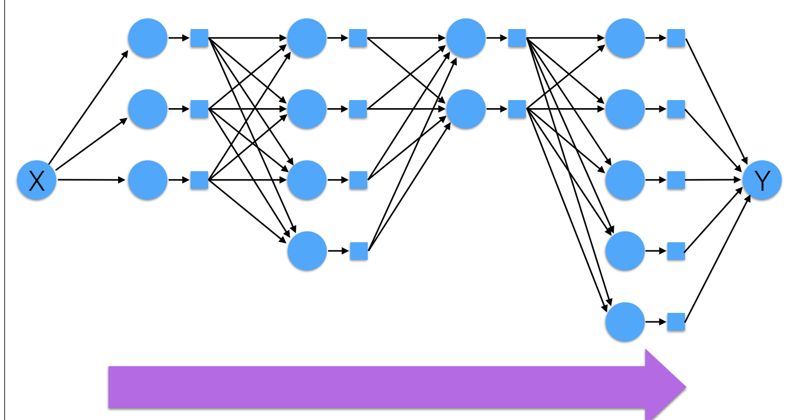
Forward propagation



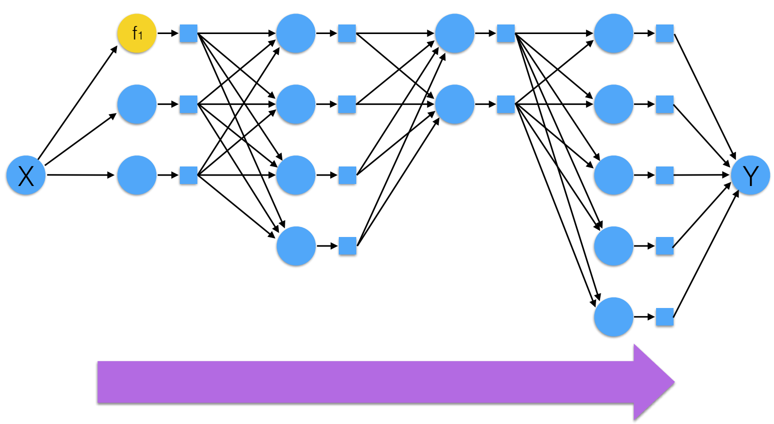
Forward propagation



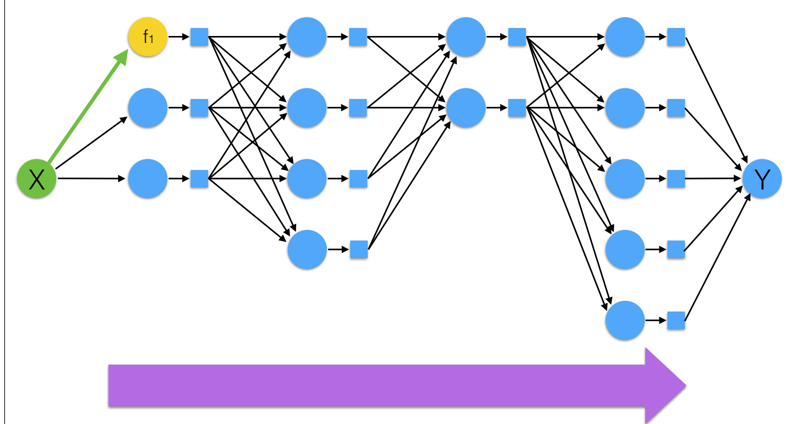
Forward propagation



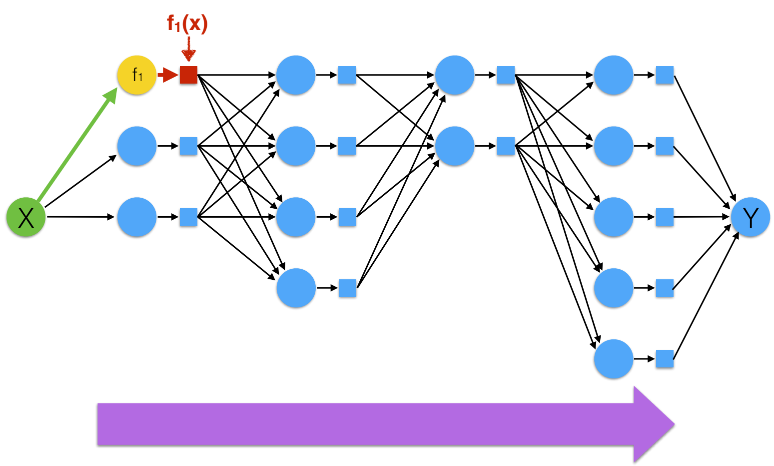
Forward propagation



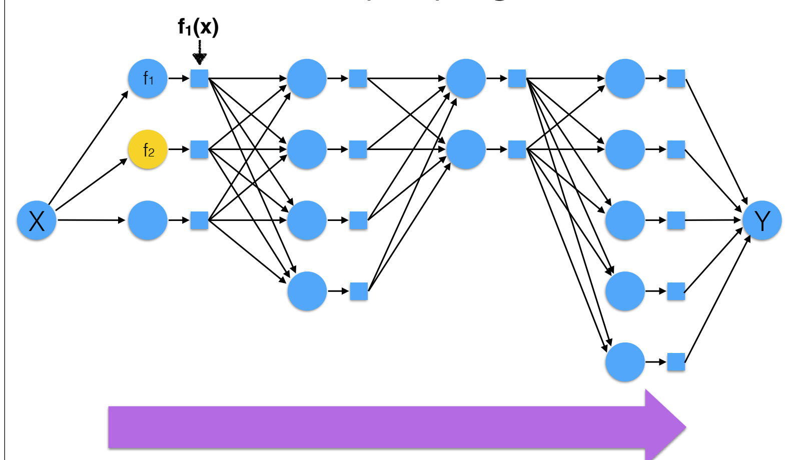
Forward propagation



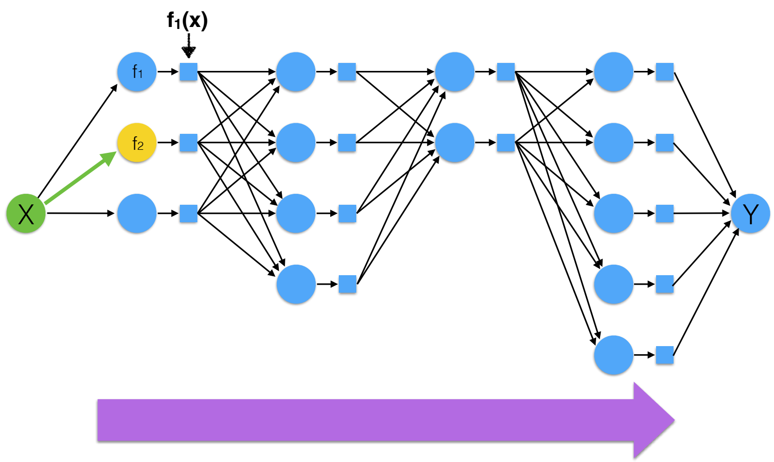
Forward propagation



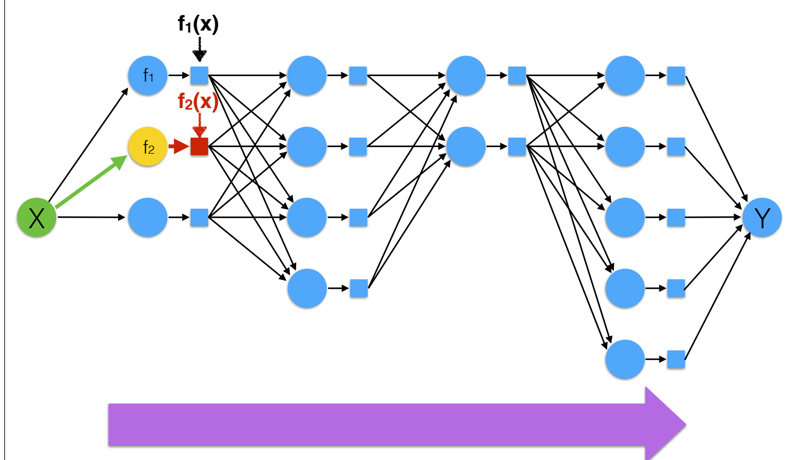
Forward propagation



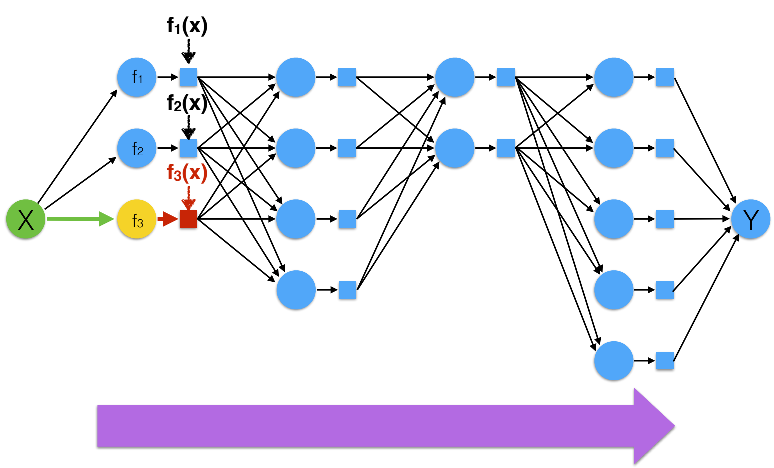
Forward propagation



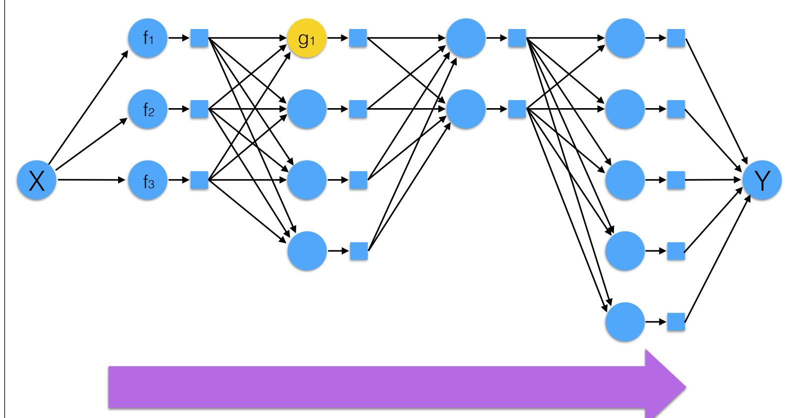
Forward propagation



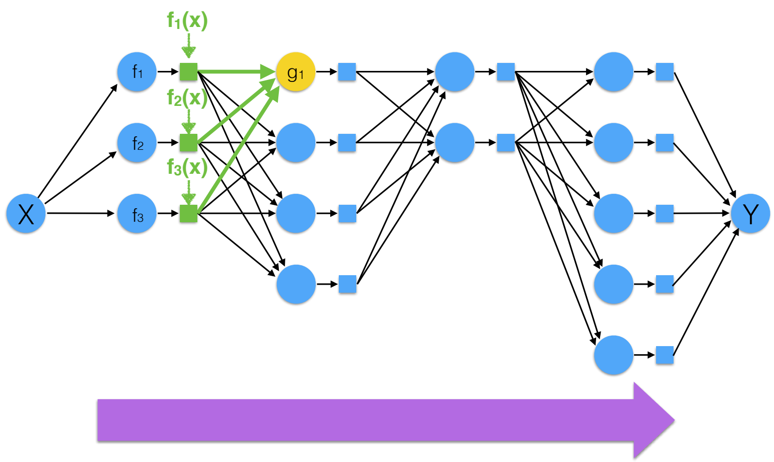
Forward propagation



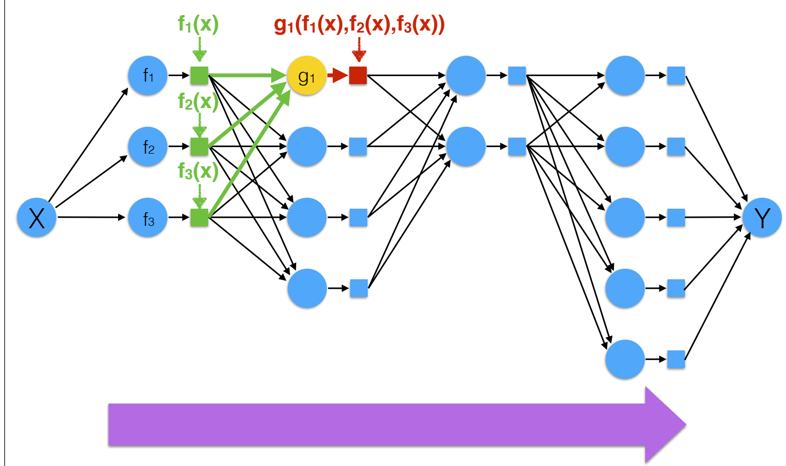
Forward propagation



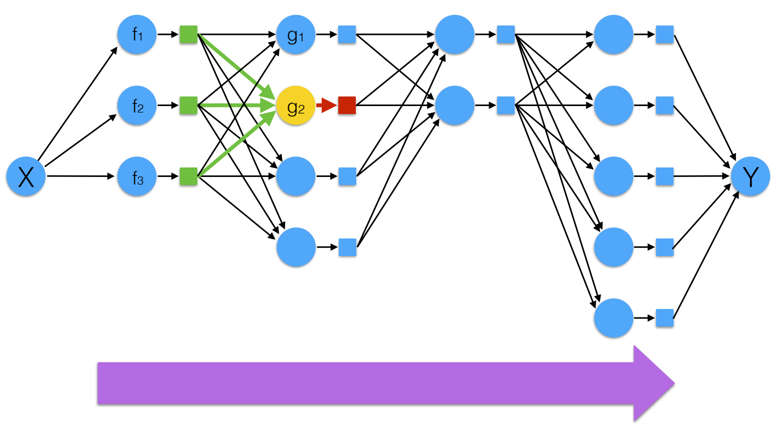
Forward propagation



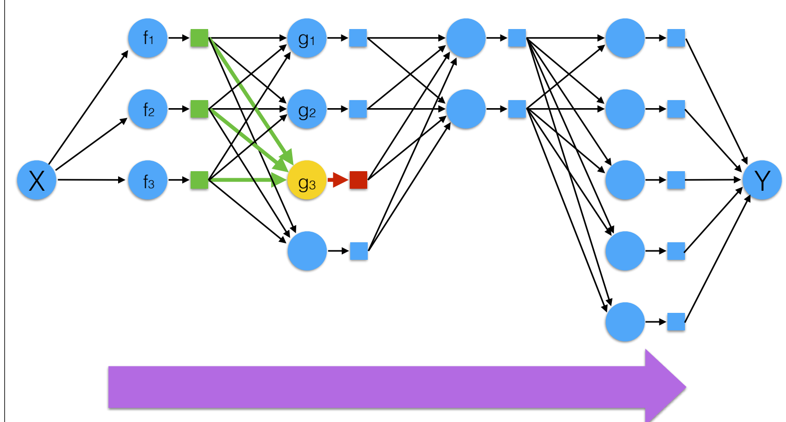
Forward propagation



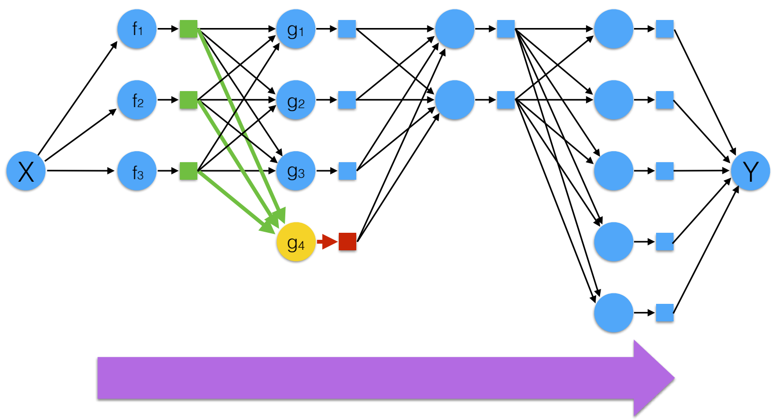
Forward propagation



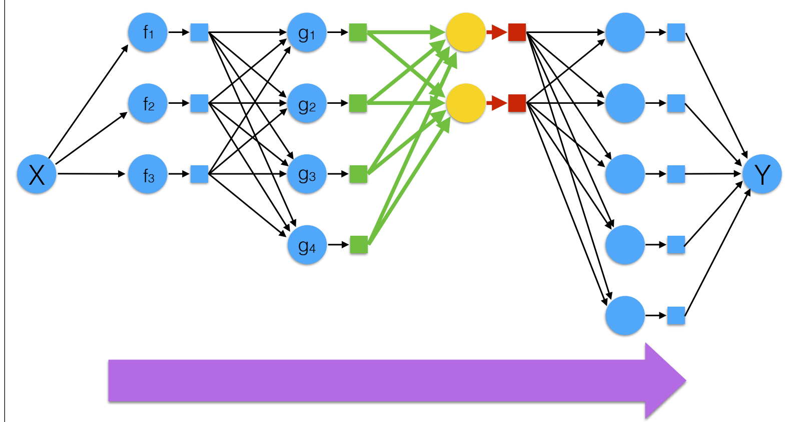
Forward propagation



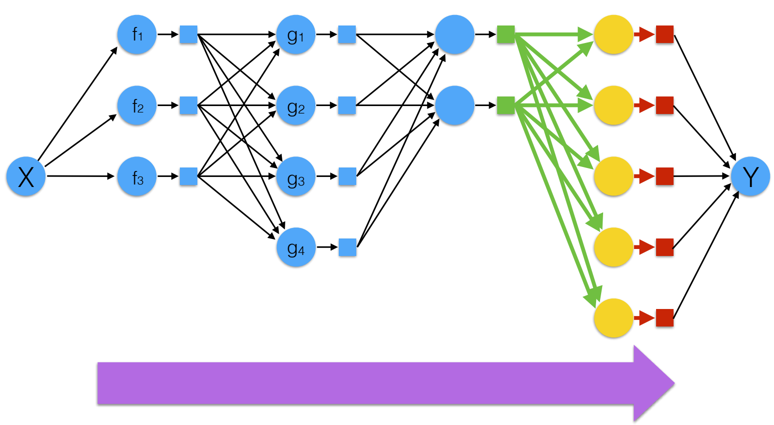
Forward propagation



Forward propagation



Forward propagation



Forward propagation