CS 4644 / 7643-A
DANFEI XU
(SLIDE CREDIT: PROF. ZSOLT KIRA)

Topics:

- Machine learning intro, applications (CV, NLP, etc.)
- Parametric models and their components

- **PS0: This should take less than 2hrs!**
- Please do it now, and give others a chance at waitlist if your background is not sufficient (beef it up and take it next time)
  - Do it even if you're on the waitlist!

- **Piazza**: not all enrolled!
  - Enroll now! http://piazza.com/gatech/spring2022/cs46447643a/
  - Make it active!

- **Office hours** start next week

Collaboration Policy

- **Collaboration**
  - Only on HWs and project (not allowed in HW0/PS0).
  - You may discuss the questions
  - Each student writes their own answers
  - Write on your homework anyone with whom you collaborate
  - Each student must write their own code for the programming part
  - Do NOT search for code implementing what we ask; search for concepts

- **Zero tolerance on plagiarism**
  - Neither ethical nor in your best interest
  - Always credit your sources
  - Don’t cheat. We will find out.

- **Grace period**
  - 2 days grace period for each assignment (**EXCEPT PS0**)
    - Intended for checking submission NOT to replace due date
    - No need to ask for grace, no penalty for turning it in within grace period
    - Can NOT use for PS0

- **After grace period, you get a 0 (no excuses except medical)**
  - Send all medical requests to dean of students (https://studentlife.gatech.edu/)
  - Form: https://gatech-advocate.symplicity.com/care_report/index.php/pid224342

- **DO NOT SEND US ANY MEDICAL INFORMATION!** We do not need any details, just a confirmation from dean of students

Grace Period

Georgia Tech

Learn Numpy!

CS231n Convolutional Neural Networks for Visual Recognition

Python Numpy Tutorial

This tutorial was contributed by Justin Johnson.

We will use the Python programming language for all assignments in this course. Python is a great general-purpose programming language on its own, but with the help of a few popular libraries (numpy, scipy, matplotlib) it becomes a powerful environment for scientific computing.

We expect that many of you will have some experience with Python and numpy; for the rest of you, this section will serve as a quick crash course both on the Python programming language and on the use of Python for scientific computing.

http://cs231n.github.io/python-numpy-tutorial/

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Machine Learning Overview

What is Machine Learning (ML)?

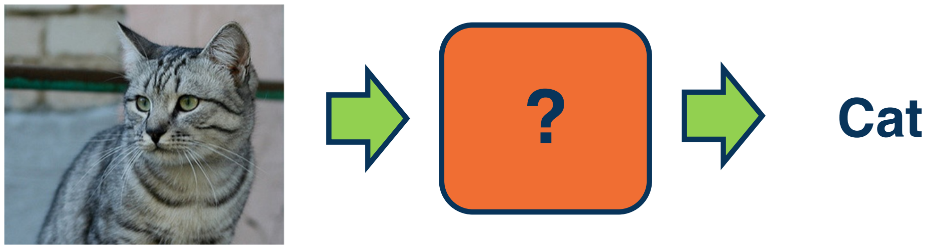
“A computer program is said to ***learn from experience*** *E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.”*

Tom Mitchell (Machine Learning, 1997)

When is Machine Learning useful?

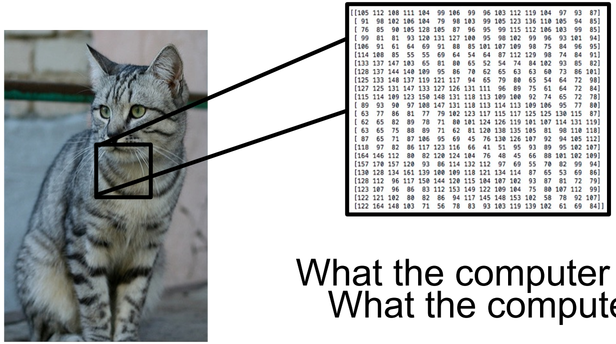
```
algorithm quicksort(A, lo, hi) is
    if lo < hi then
        p := partition(A, lo, hi)
        quicksort(A, lo, p - 1)
        quicksort(A, p + 1, hi)

algorithm partition(A, lo, hi) is
    pivot := A[hi]
    i := lo
    for j := lo to hi do
        if A[j] < pivot then
            swap A[i] with A[j]
            i := i + 1
    swap A[i] with A[hi]
    return i
```



When it’s difficult / infeasible to write a program

Example: Object Recognition



This image by Nikita is licensed under CC-BY 2.0

What the computer sees
What the computer sees

An image is just a big grid of numbers between [0, 255]:

e.g. 800 x 600 x 3
(3 channels RGB)

Viewpoint Changes

All pixels change when the camera moves!

Illumination

This image is CC0 1.0 public domain

This image is CC0 1.0 public domain

This image is CC0 1.0 public domain

This image is CC0 1.0 public domain

Deformation

This image by Umberto Salvagnin is licensed under CC-BY 2.0

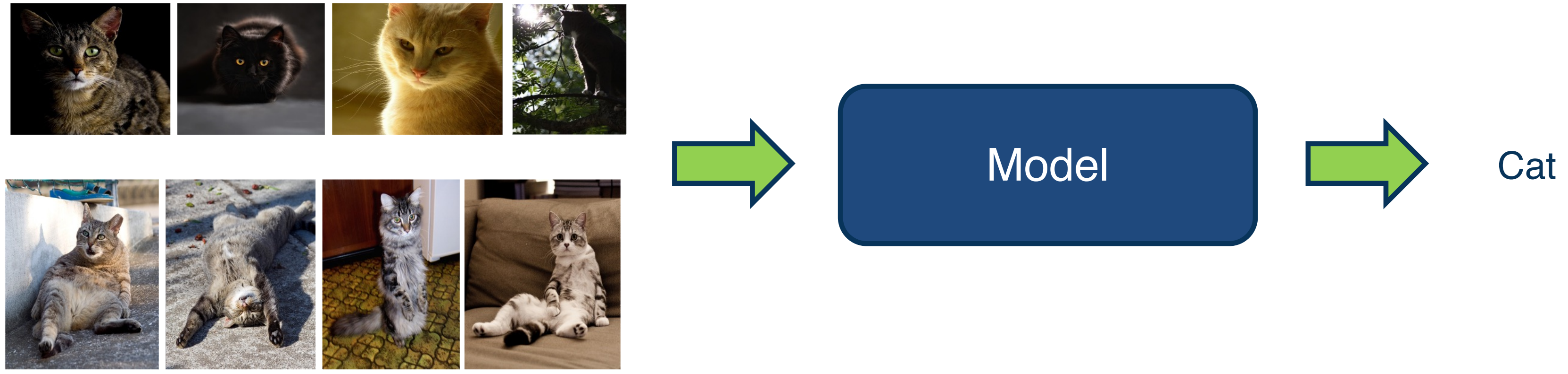
This image by Umberto Salvagnin is licensed under CC-BY 2.0

This image by sare bear is licensed under CC-BY 2.0

This image by Tom Thai is licensed under CC-BY 2.0

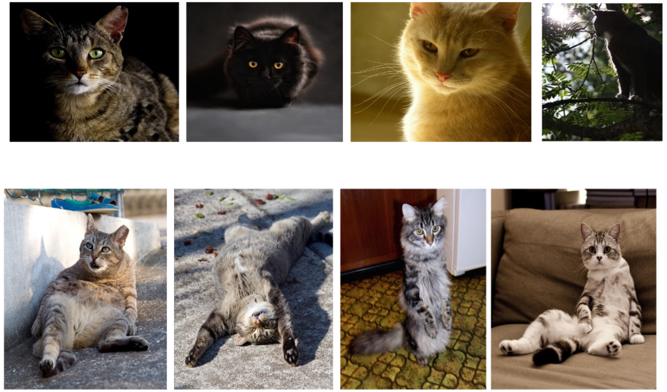
Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

The Power of Machine Learning



The Power of Machine Learning

The Power of Machine Learning



Deep Neural Networks

Cat

The Power of (Deep) Machine Learning

TECHNOLOGY

A Massive Google Network Learns To Identify — Cats

June 26, 2012 · 3:00 PM ET

Heard on All Things Considered

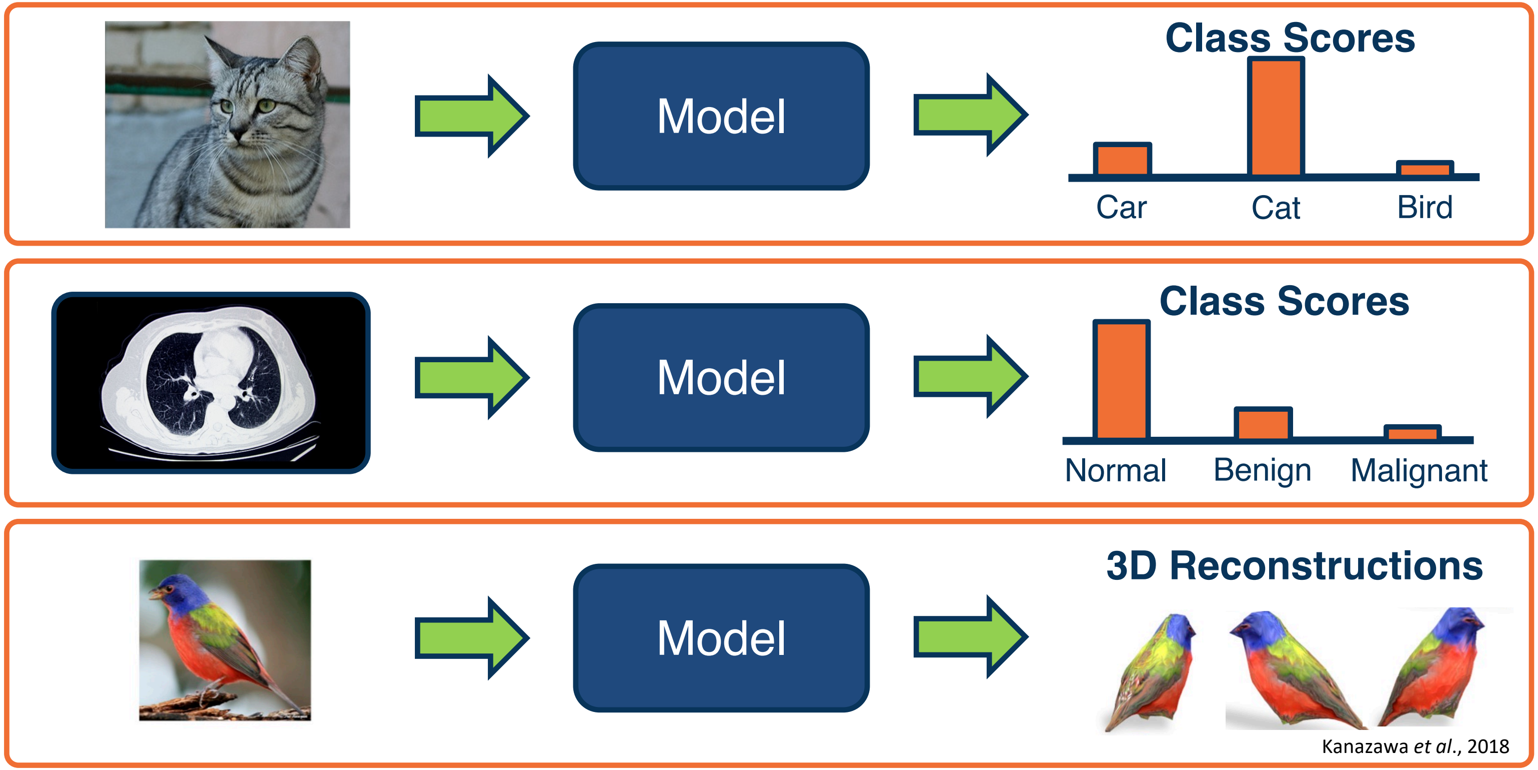
4-Minute Listen

+ PLAYLIST

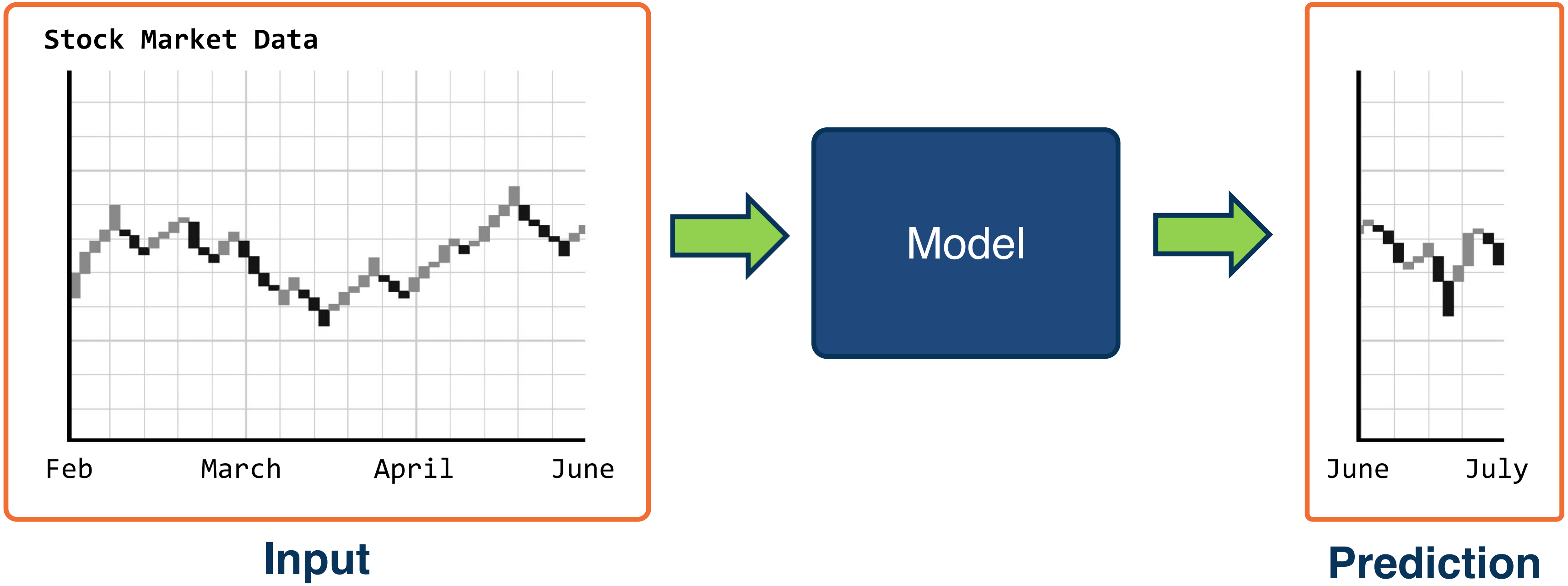
All Things Considered host Audie Cornish talks with Andrew Ng, Associate Professor of Computer Science at Stanford University. He led a Google research team in creating a neural network out of 16,000 computer processors to try and mimic the functions of the human brain. Given three days on YouTube, the network taught itself how to identify — cats.

Source: https://www.npr.org/2012/06/26/155792609/a-massive-google-network-learns-to-identify

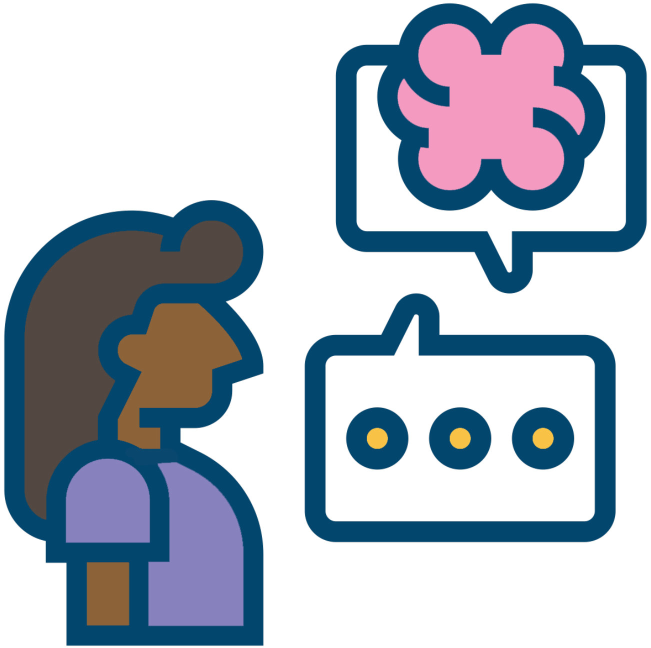
Application: Computer Vision



Application: **Time Series Forecasting**



Application: Natural Language Processing (NLP)



Very large number of NLP sub-tasks:

- Syntax Parsing
- Translation
- Named entity recognition
- Summarization
- Similarity / paraphrasing

Sequence modeling: Variable length sequential inputs and/or outputs

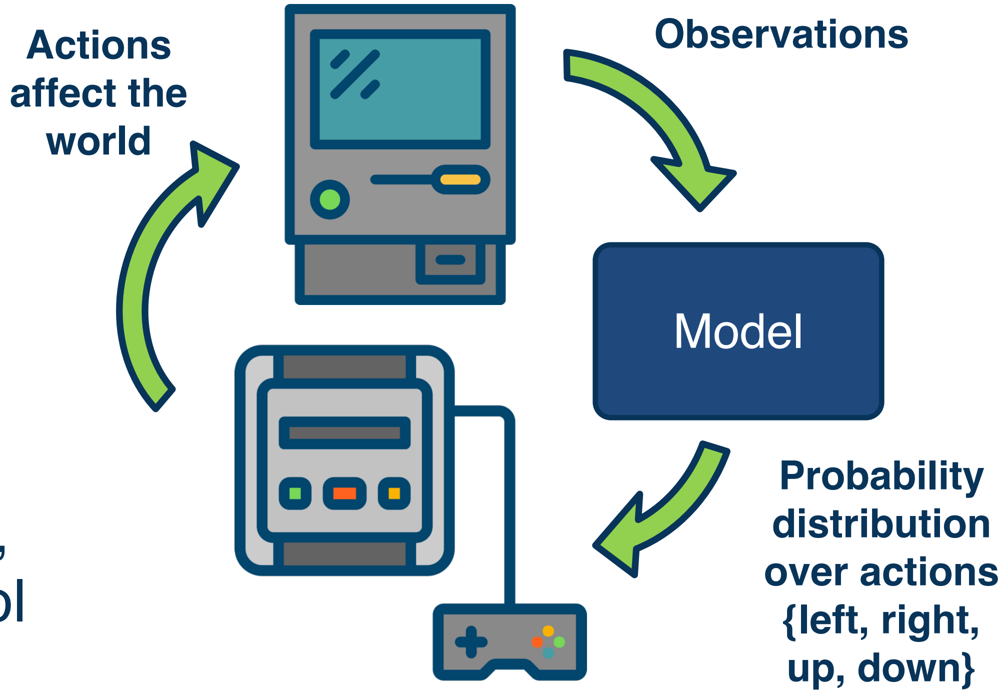
Recent progress: Large-scale Language Models

Application: Decision Making

- Sequence of inputs/outputs
- Actions affect the environment

Examples: Chess / Go, Video Games, Recommendation Systems, Network Congestion Control …

Example: Video Game



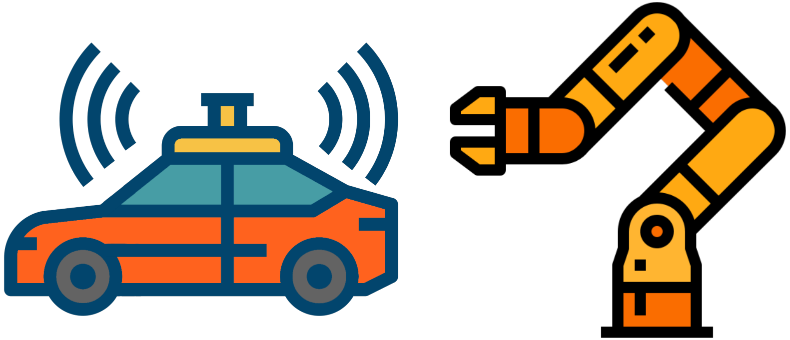
Robotics involves a **combination of AI/ML techniques**:

- **Sense:** Perception
- **Plan:** Planning
- **Act:** Controls

Some things are **learned (perception)**, while others **programmed**

An evolving landscape

Application:



Example: Robotics

Rest of the lecture (also next lecture):

- **Types of Machine Learning Problems**
- **Parametric Models**
- **Linear Classifiers**
- **Gradient Descent**

Supervised Learning

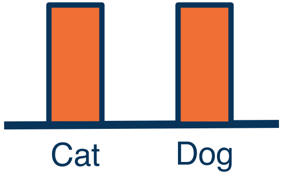
Unsupervised Learning

Reinforcement Learning

Supervised Learning

- Train Input: $\{X, Y\}$
- Learning output: $f : X \rightarrow Y$
- Usually f is a **distribution**, e.g. $P(y|x)$

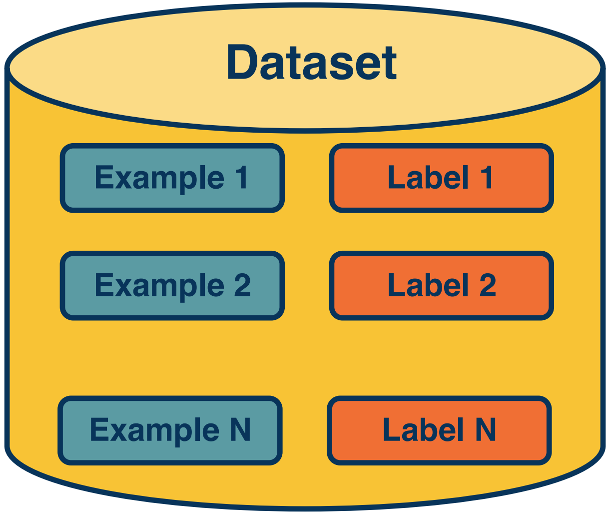
https://en.wikipedia.org/wiki/CatDog



Dataset

$X = \{x_1, x_2, \ldots, x_N\}$ *where* $x \in \mathbb{R}^d$ **Examples**

$Y = \{y_1, y_2, \ldots, y_N\}$ *where* $y \in \mathbb{R}^c$ **Labels**



Supervised Learning

- Train Input: $\{X, Y\}$
- Learning output: $f : X \rightarrow Y$, e.g. $p(y|x)$

Terminology:

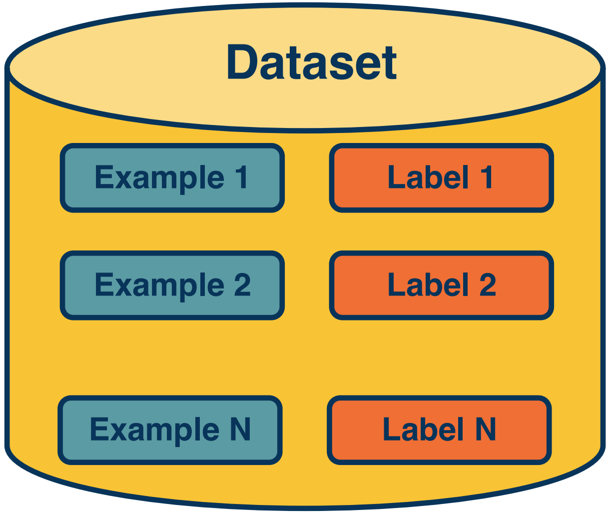
- Model / Hypothesis Class
  - $H: \{f: X \rightarrow Y\}$
  - Learning is search in hypothesis space

E.g., $H = \{f(x) = w^T x \mid w \in \mathbb{R}^d\}$

Dataset

$X = \{x_1, x_2, \ldots, x_N\}$ $where\ x \in \mathbb{R}^d$ **Examples**

$Y = \{y_1, y_2, \ldots, y_N\}$ $where\ y \in \mathbb{R}^c$ **Labels**



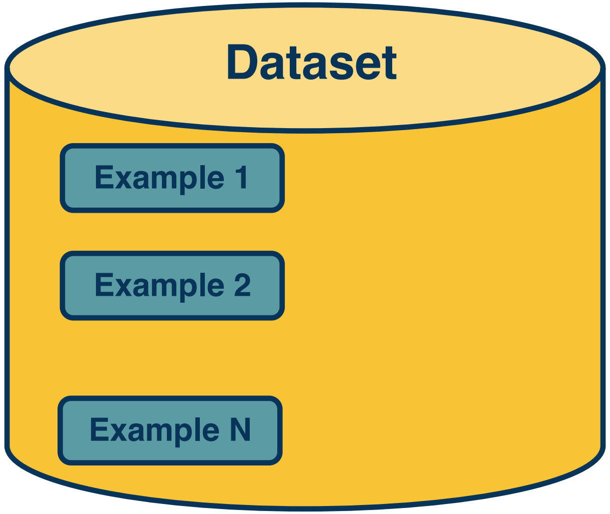
Unsupervised Learning

- Input: $\{X\}$
- Learning output: $p_{data}(x)$
- How likely is x under p_{data}?
- Can we sample from p_{data}?
- Example: Clustering, density estimation, generative modeling, …

Dataset

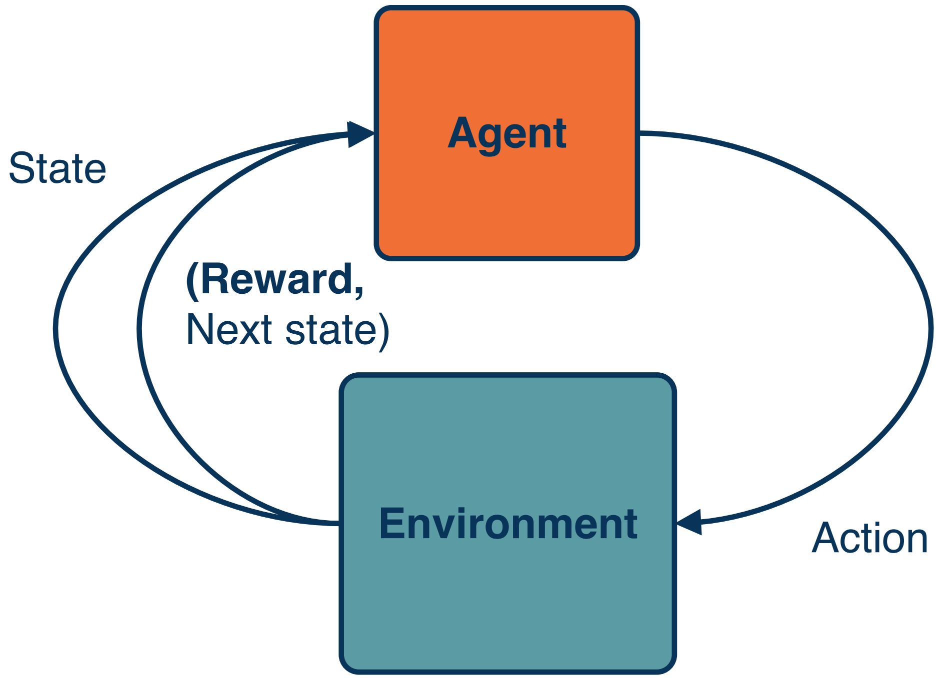
$$X = \{x_1, x_2, \dots, x_N\} \; where \; x \in \mathbb{R}^d$$

Examples



Reinforcement Learning

- Supervision in form of **reward**
- No supervision on what action to take



Adapted from: http://cs231n.stanford.edu/slides/2020/lecture_17.pdf

Supervised Learning

- Train Input: $\{X, Y\}$
- Learning output: $f : X \rightarrow Y$, e.g. $P(y|x)$

Unsupervised Learning

- Input: $\{X\}$
- Learning output: $P(x)$
- Example: Clustering, density estimation, etc.

Reinforcement Learning

- Supervision in form of **reward**
- No supervision on what action to take

Very often combined, sometimes within the same model!

Rest of the lecture (also next lecture):

- Types of Machine Learning Problems
- **Parametric Models**
- Linear Classifiers
- Gradient Descent

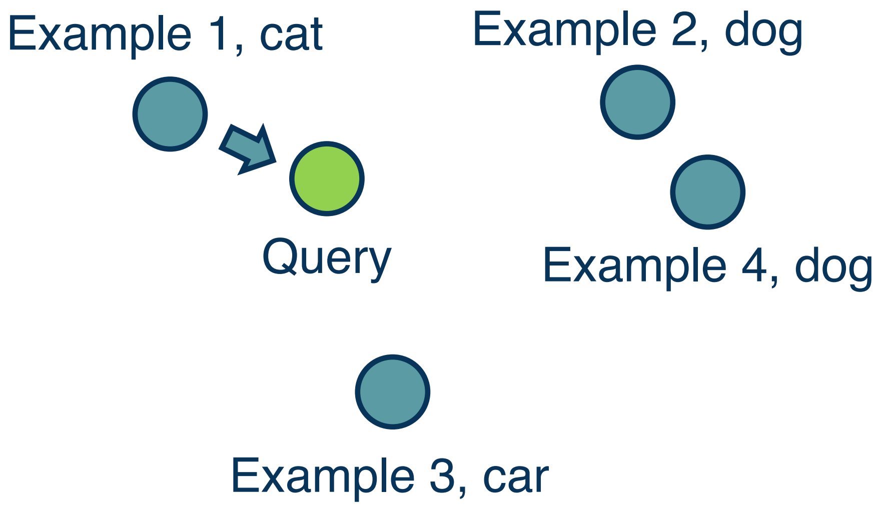
Non-Parametric Model

No explicit model for the function, **examples**:

- **Nearest neighbor classifier**
- Decision tree

Hypothesis class changes with the number of data points

Non-Parametric – Nearest Neighbor



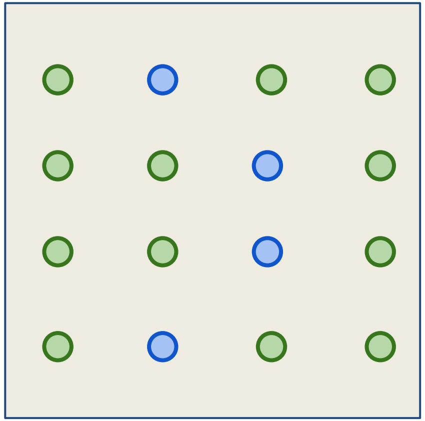
Procedure: Take label of nearest example

k-Nearest Neighbor on high-dimensional data (e.g., images) is *almost never* used.

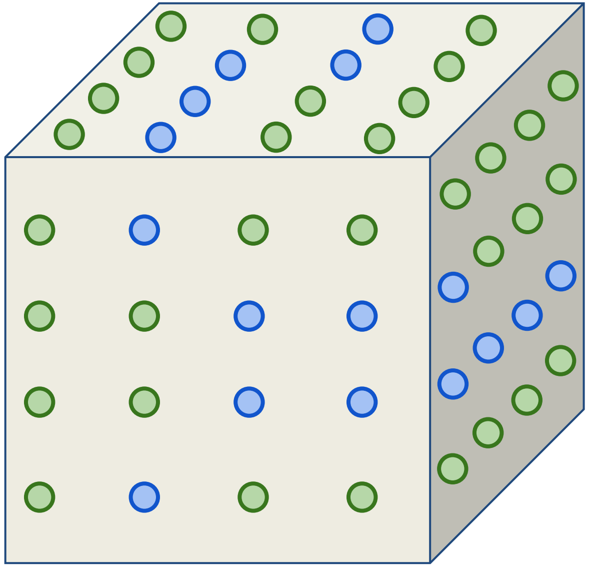
Curse of dimensionality

Dimensions = 1
Points = 4

Dimensions = 2
Points = 4^2



Dimensions = 3
Points = 4^3



Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

- **Curse of Dimensionality**
  - Data required increases exponentially with the number of dimensions
- **Doesn't work well when large number of irrelevant features**
  - Distances overwhelmed by noisy features
- **Expensive**
  - No Learning: most real work done during testing
  - For every test sample, must search through all dataset – very slow!
  - Must use tricks like approximate nearest neighbor search

Problems with Instance-Based Learning

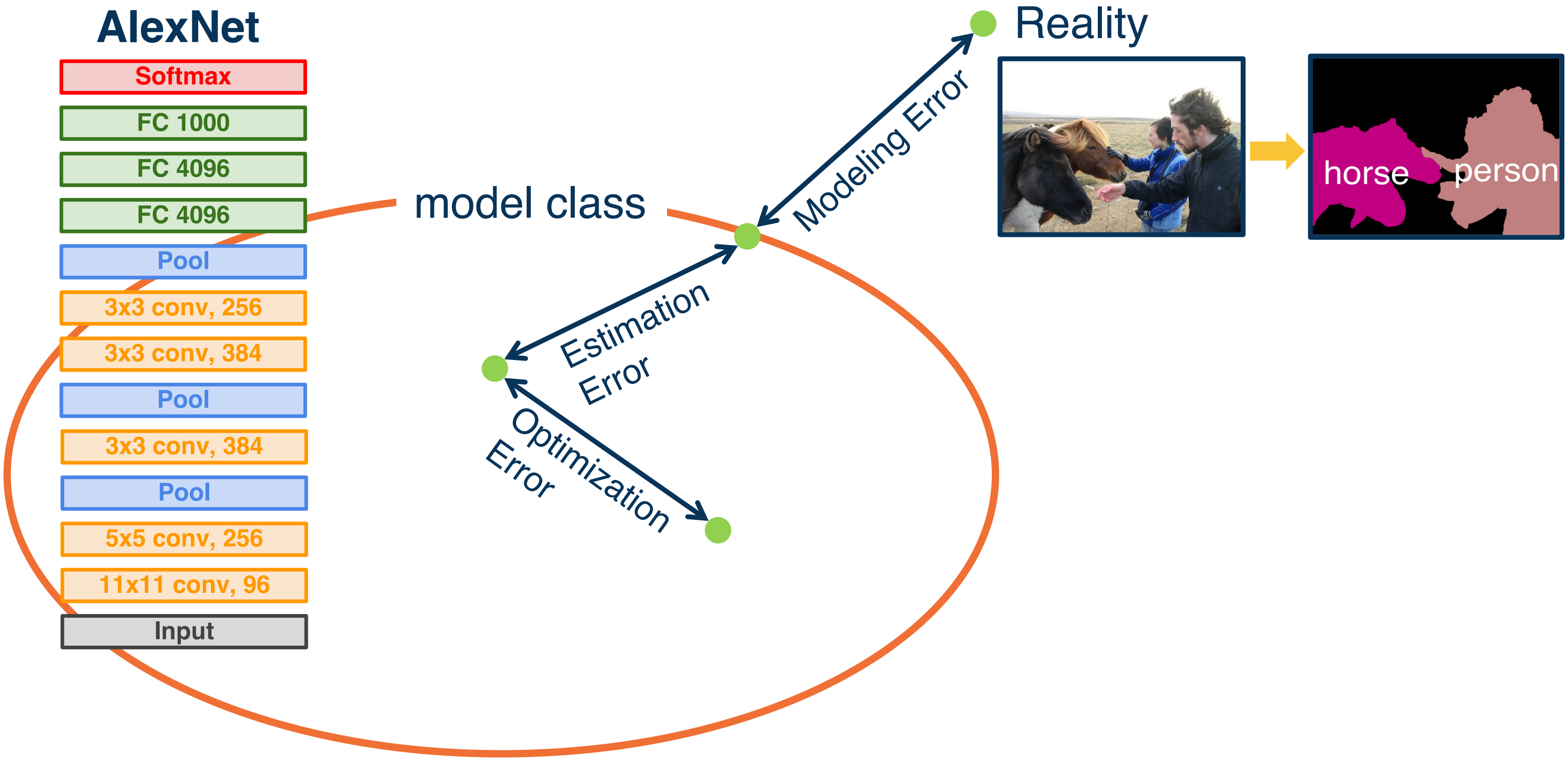
Parametric Model

Explicitly model the function $f : X \to Y$ in the form of a parametrized function $f(x, W) = y$, **examples**:

- Logistic regression/classification
  - Number of parameters grows **linearly** with the number of dimensions!
- Neural networks
- Hypothesis classes doesn't change

Parametric – Linear Classifier

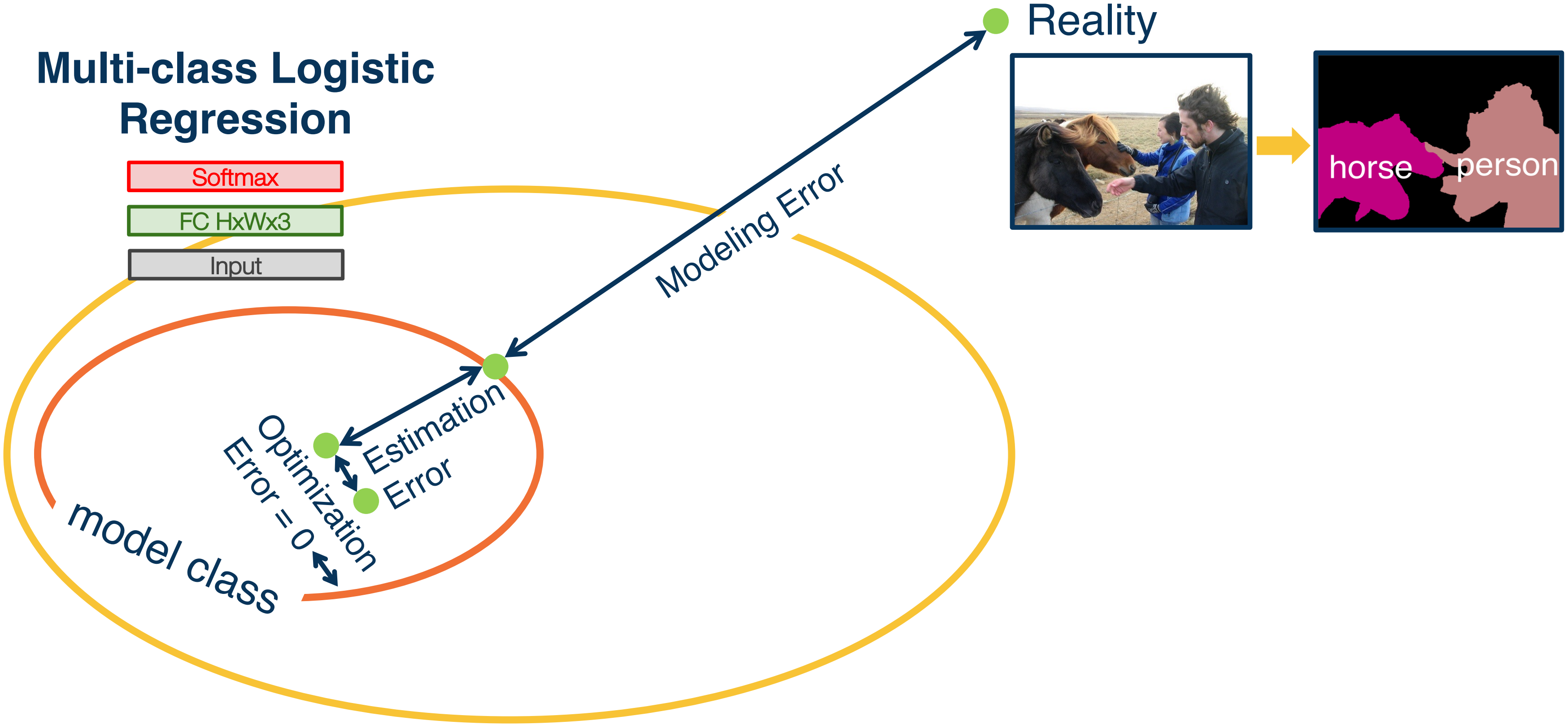
$$f(x, W) = Wx + b$$



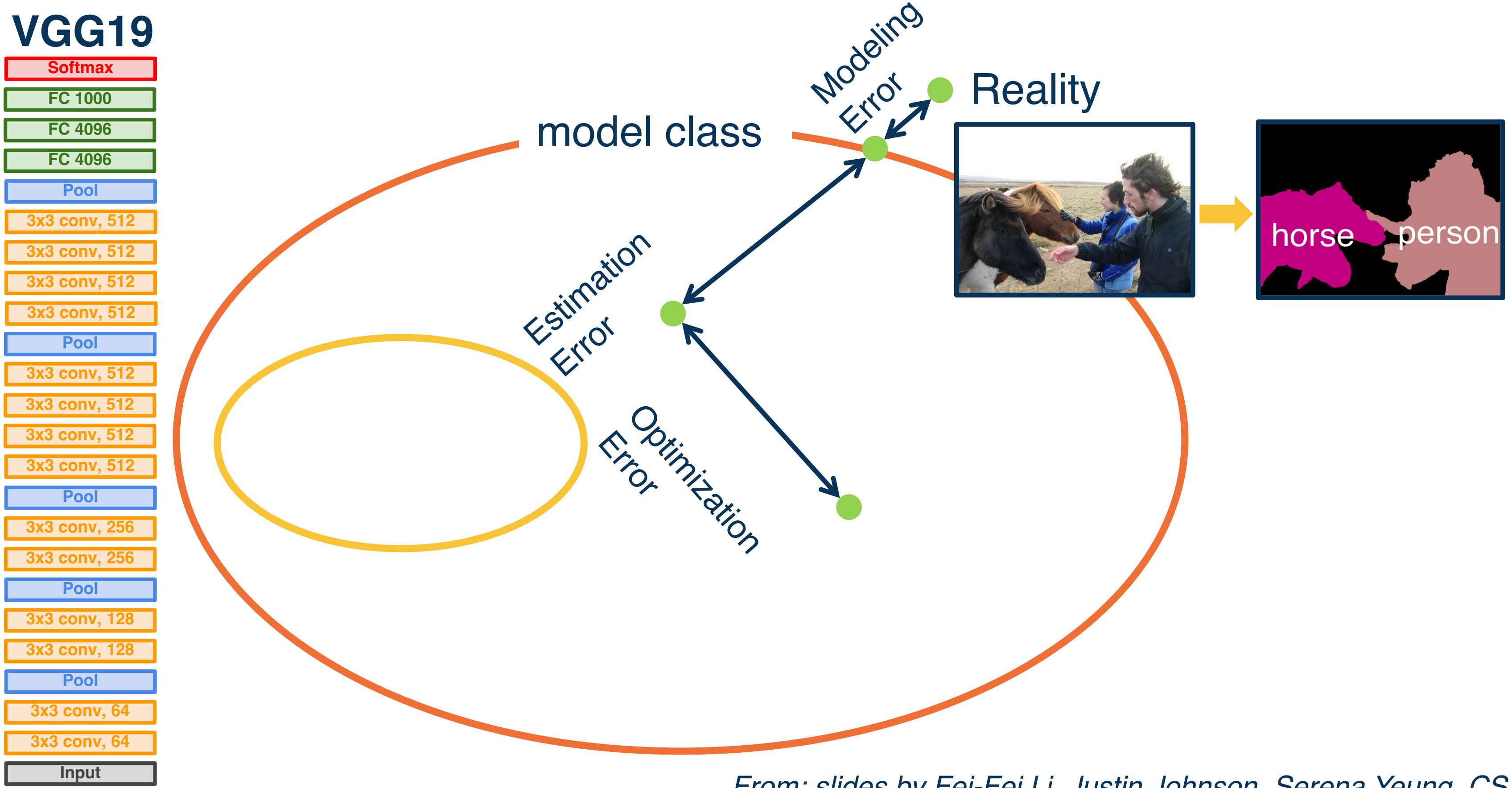
From: slides by Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Generalization

Multi-class Logistic Regression



From: slides by Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n



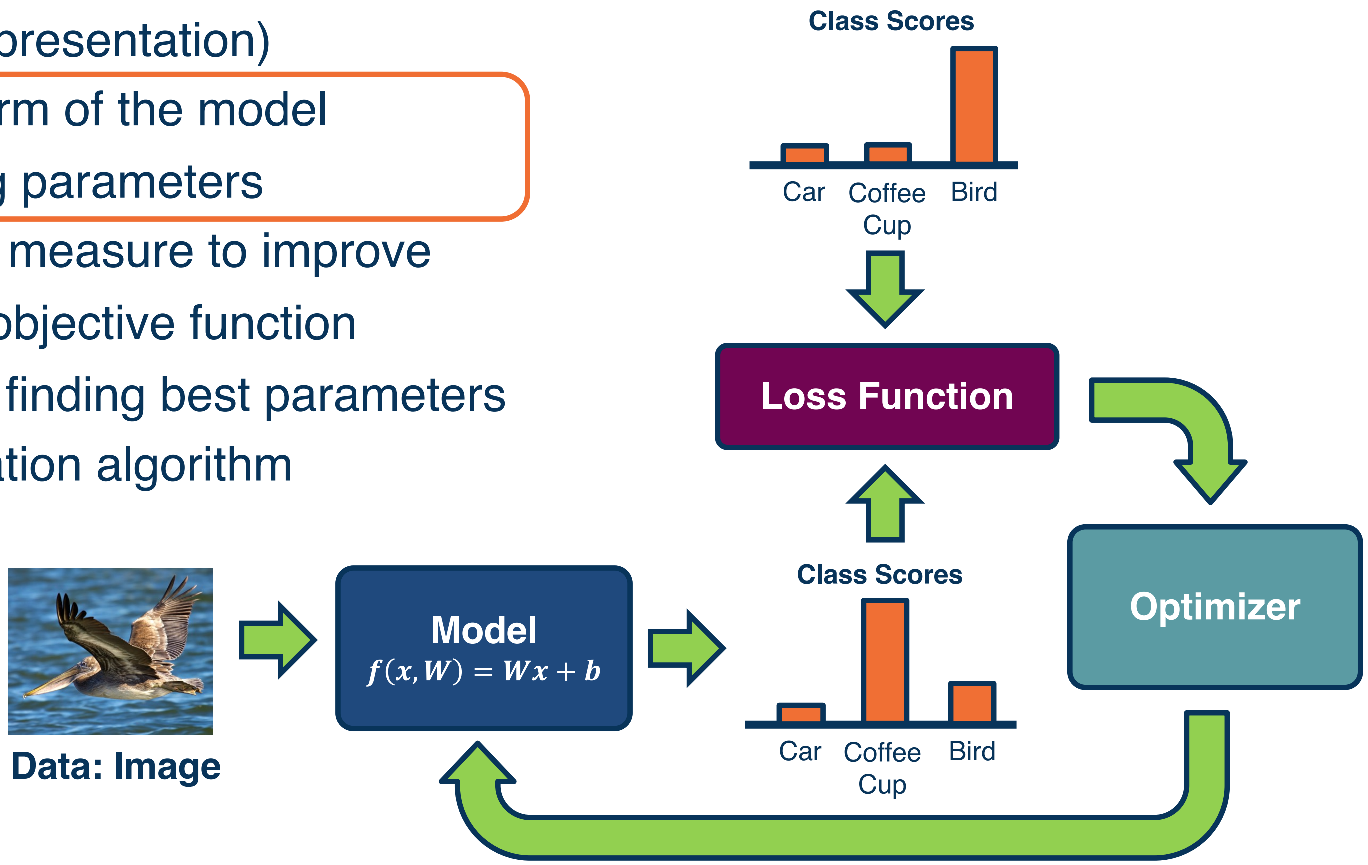
From: slides by Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Types of Errors and Generalization

Rest of the lecture (also next lecture):

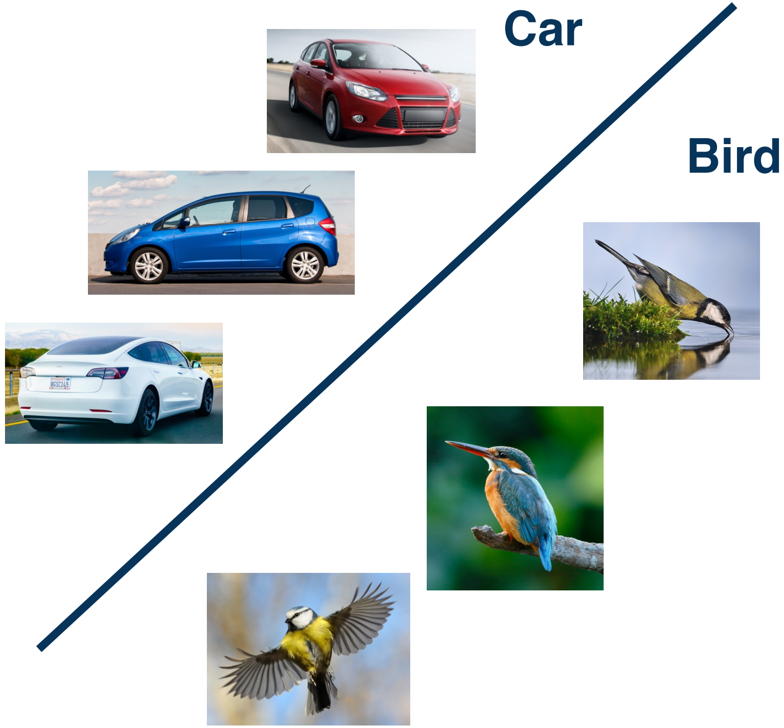
- Types of Machine Learning Problems
- Parametric Models
- **Linear Classifiers**
- Gradient Descent

- Input (and representation)
- Functional form of the model
  - Including parameters
- Performance measure to improve
  - Loss or objective function
- Algorithm for finding best parameters
  - Optimization algorithm



Components of a Parametric Model

What is the **simplest function** you can think of?



Our model is:

$$f(x, w) = w \cdot x + b$$

Classifier Result

Weights

Input

Bias (scalar)

(Note if $\boldsymbol{w}$ and $\mathbf{x}$ are column vectors we often show this as $\boldsymbol{w}^T\boldsymbol{x}$)

Simple Function

Linear Classification and Regression

Simple linear classifier:

- Calculate score:

  $$f(x, w) = w \cdot x + b$$

- Binary classification rule (w is a vector):

  $$y = \begin{cases} 1 & \text{if } f(x, w) > = 0 \\ 0 & \text{otherwise} \end{cases}$$

- For multi-class classifier take class with highest (max) score

  $$f(x, W) = Wx + b$$

Georgia Tech

Data: Image

Model

$$f(x, W) = Wx + b$$

Class Scores

$$x = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nn} \end{bmatrix}$$

Flatten

$$x = \begin{bmatrix} x_{11} \\ x_{12} \\ \vdots \\ x_{21} \\ x_{22} \\ \vdots \\ x_{n1} \\ \vdots \\ x_{nn} \end{bmatrix}$$

To simplify notation we will refer to inputs as $x_1 \cdots x_m$ where $m = n \times n$

Input Dimensionality

Model

$$f(x, W) = Wx + b$$

Classifier for class 1 →
Classifier for class 2 →
Classifier for class 3 →

$$\underbrace{\begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1m} \\ w_{21} & w_{22} & \cdots & w_{2m} \\ w_{31} & w_{32} & \cdots & w_{3m} \end{bmatrix}}_{\boldsymbol{W}} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}}_{\boldsymbol{x}} + \underbrace{\begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}}_{\boldsymbol{b}}$$

(Note that in practice, implementations can use xW instead, assuming a different shape for W. That is just a different convention and is equivalent.)

- We can move the bias term into the weight matrix, and a “1” at the end of the input
- Results in **one matrix-vector multiplication**!

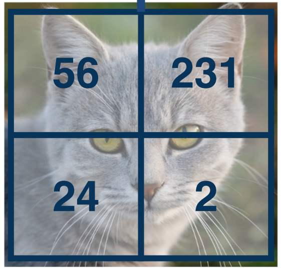
Model

$$f(x, W) = Wx + b$$

$$\underbrace{\begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1m} & b_1 \\ w_{21} & w_{22} & \cdots & w_{2m} & b_2 \\ w_{31} & w_{32} & \cdots & w_{3m} & b_3 \end{bmatrix}}_{W} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \\ 1 \end{bmatrix}}_{x}$$

Example with an image with **4 pixels**, and **3 classes** (cat/dog/ship)

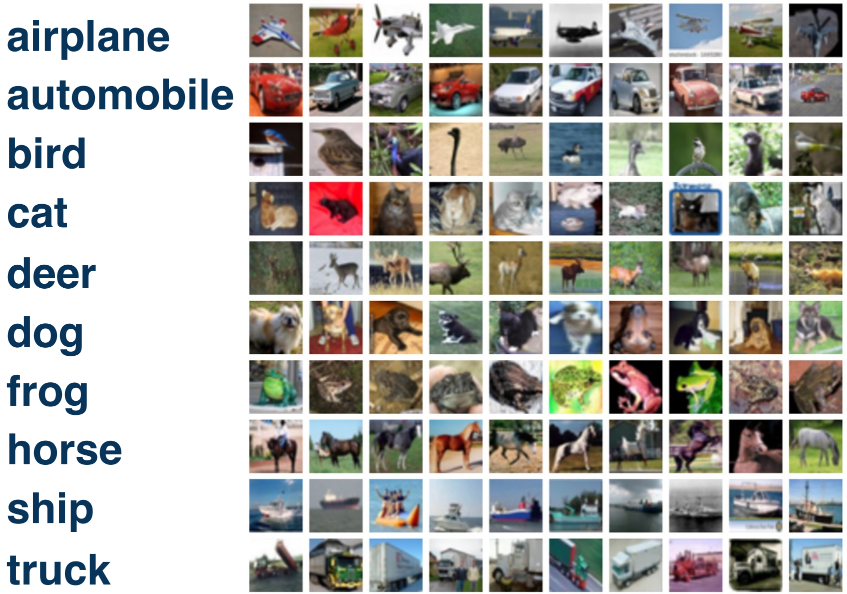
Stretch pixels into column



Input image

$$\underbrace{\begin{bmatrix} 0.2 & -0.5 & 0.1 & 2.0 \\ 1.5 & 1.3 & 2.1 & 0.0 \\ 0 & 0.25 & 0.2 & -0.3 \end{bmatrix}}_{W} \begin{bmatrix} 56 \\ 231 \\ 24 \\ 2 \end{bmatrix} + \underbrace{\begin{bmatrix} 1.1 \\ 3.2 \\ -1.2 \end{bmatrix}}_{b} = \begin{array}{cl} \begin{bmatrix} -96.8 \\ 437.9 \\ 61.95 \end{bmatrix} & \begin{array}{l} \text{Cat score} \\ \text{Dog score} \\ \text{Ship score} \end{array} \end{array}$$

Adapted from slides by Fei-Fei Li, Justin Johnson, Serena Yeung, from CS 231n

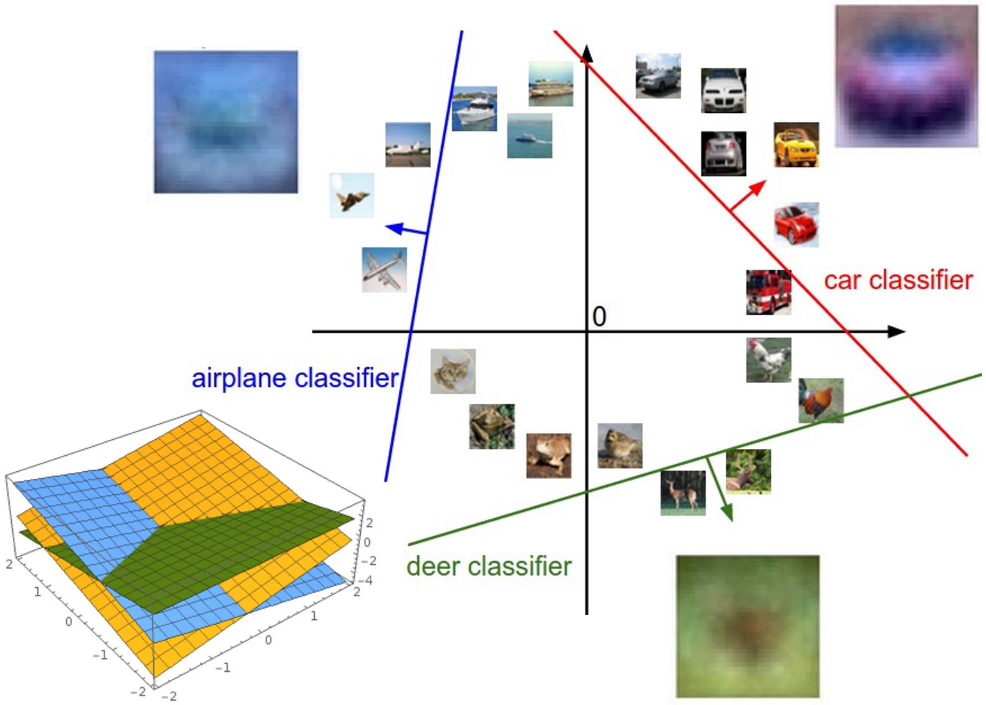


Visual Viewpoint

We can convert the weight vector back into the shape of the image and visualize

Adapted from slides by Fei-Fei Li, Justin Johnson, Serena Yeung, from CS 231n

Geometric Viewpoint



Plot created using Wolfram Cloud

$$f(x, W) = Wx + b$$

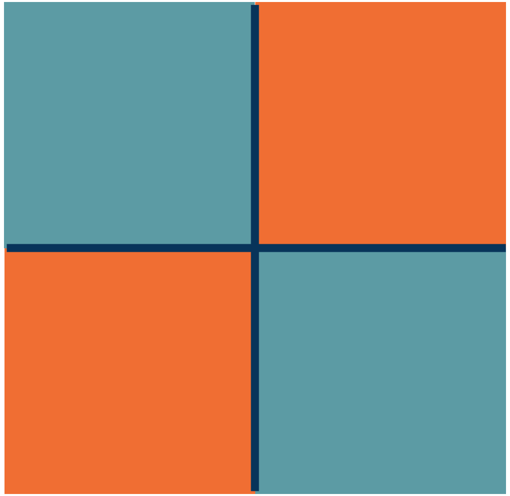
Array of **32x32x3** numbers
(3072 numbers total)

Adapted from slides by Fei-Fei Li, Justin Johnson, Serena Yeung, from CS 231n

Hard Cases for a Linear Classifier

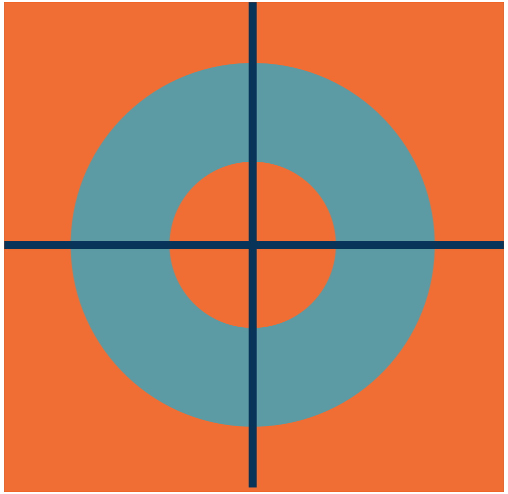
Class 1:
number of pixels > 0 odd

Class 2:
number of pixels > 0 even



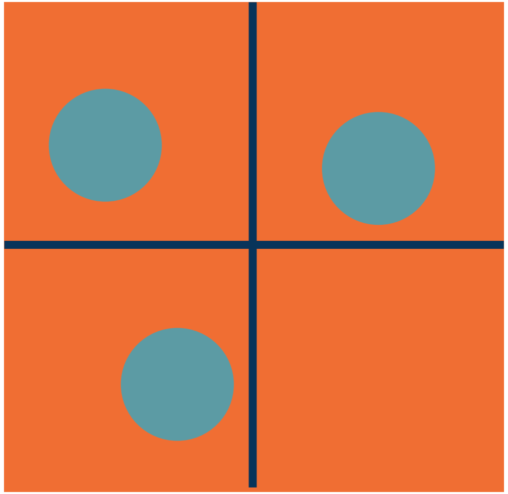
Class 1:
1 < = L2 norm < = 2

Class 2:
Everything else



Class 1:
Three modes

Class 2:
Everything else



Adapted from slides by Fei-Fei Li, Justin Johnson, Serena Yeung, from CS 231n

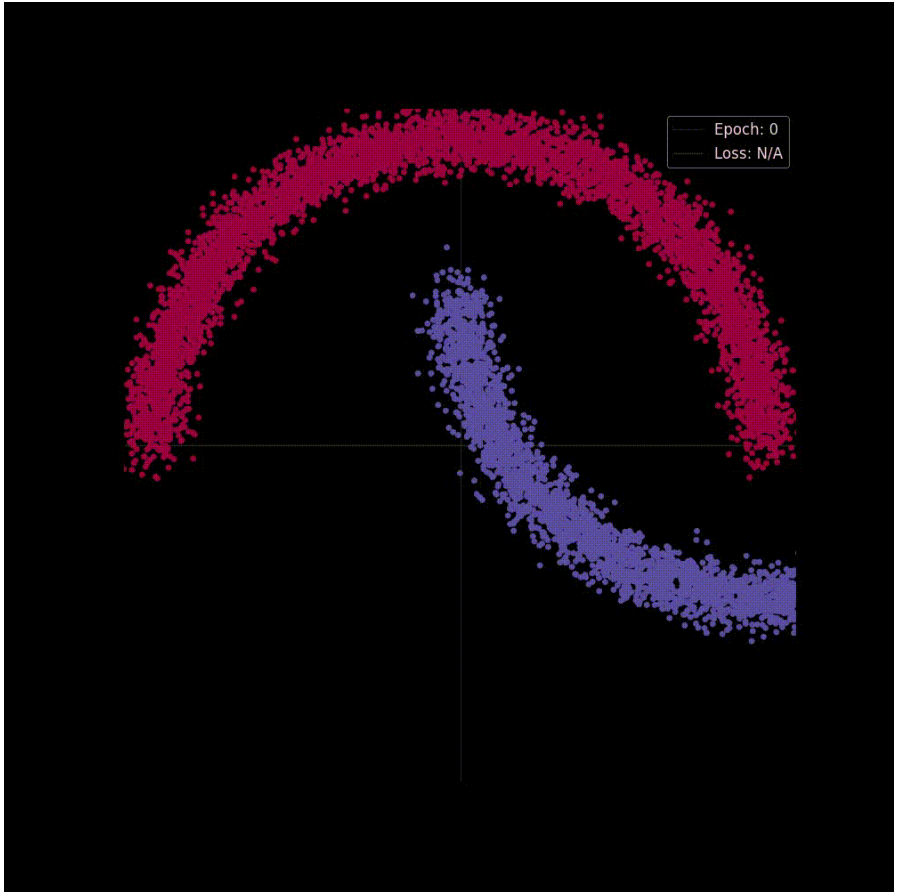
Neural Network

Linear
classifiers

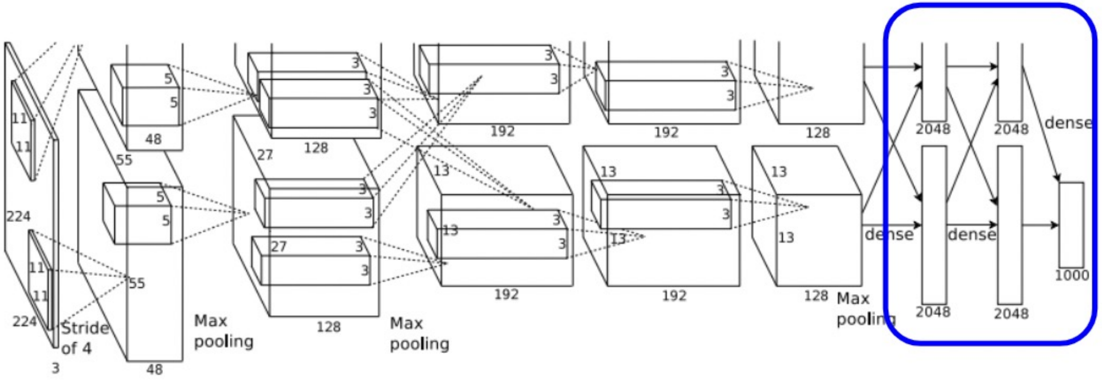
Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

(Deep) Representation Learning for Classification

A function that transforms raw data space into a linearly-separable space

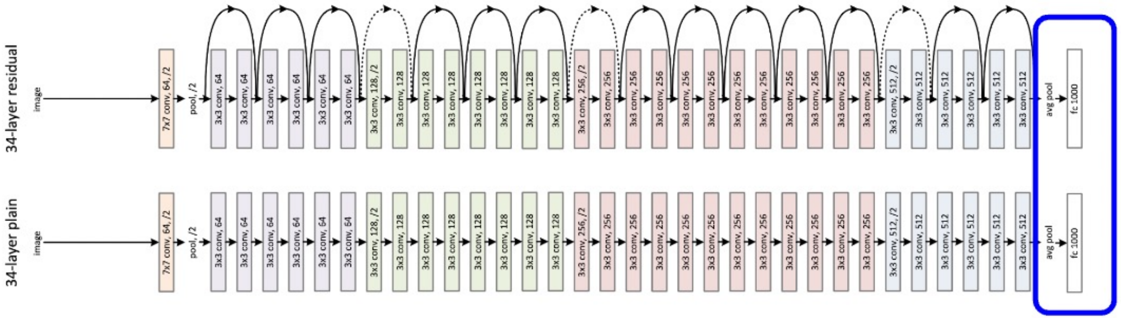


https://khalidsaifullaah.github.io/neural-networks-from-linear-algebraic-perspective



[Krizhevsky et al. 2012]

Linear layers

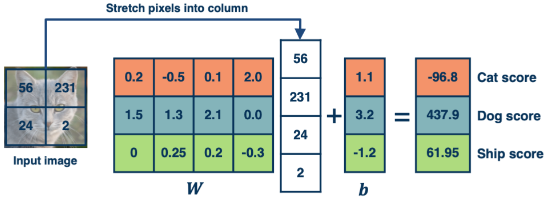


[He et al. 2015]

Slide Credit: Fei-Fei Li, Ranjay Krishna, Danfei Xu, CS 231n

Algebraic Viewpoint

$$f(x, W) = Wx$$

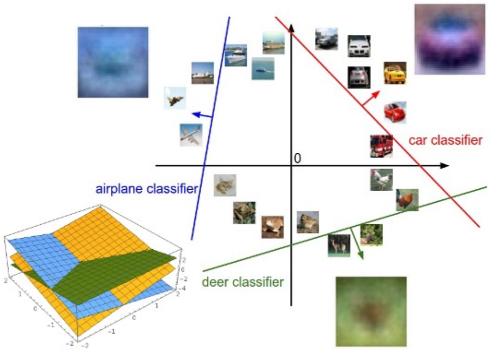


Visual Viewpoint

One template per class

Geometric Viewpoint

Hyperplanes cutting up space



Adapted from slides by Fei-Fei Li, Justin Johnson, Serena Yeung, from CS 231n

Next time:

- Input (and representation)
- Functional form of the model
  - Including parameters
- **Performance measure to improve**
  - **Loss or objective function**
- Algorithm for finding best parameters
  - Optimization algorithm