Topics:

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

CS 4644 / 7643-A

ZSOLT KIRA

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

Python + Numpy Tutorial

Machine Learning Overview

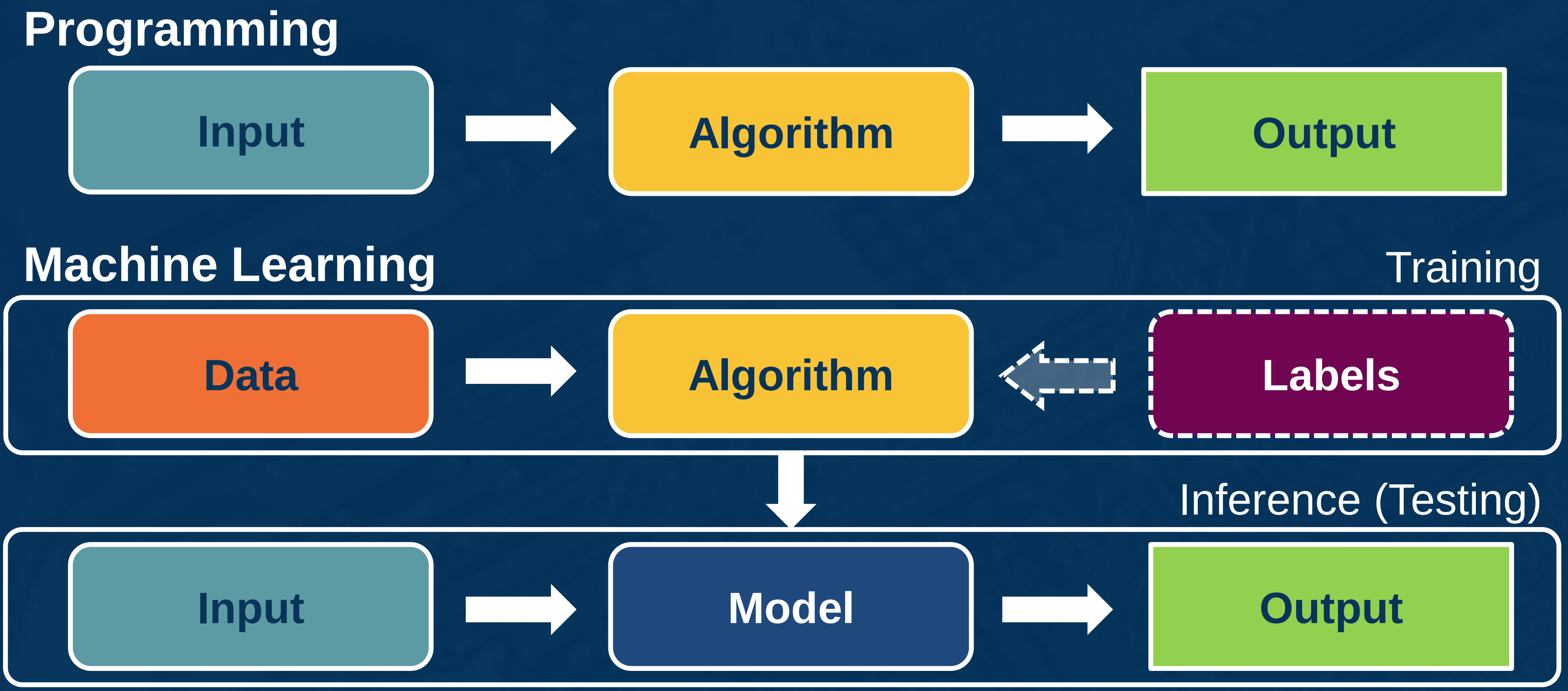
What is Machine Learning (ML)?

"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)

Georgia Tech

How is it Different than Programming?

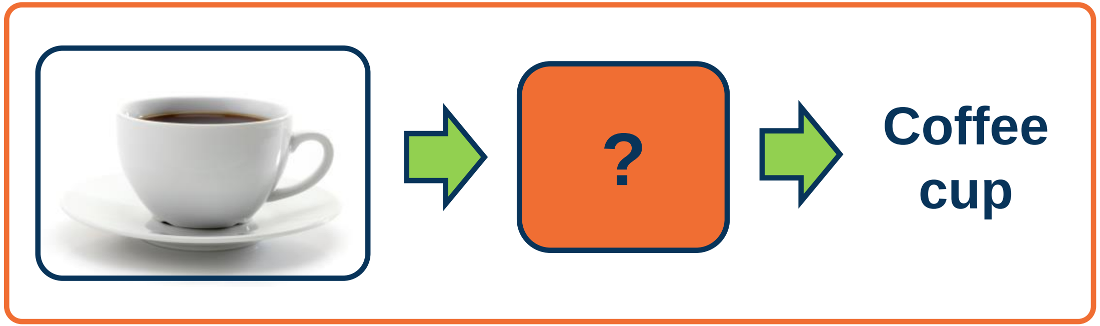


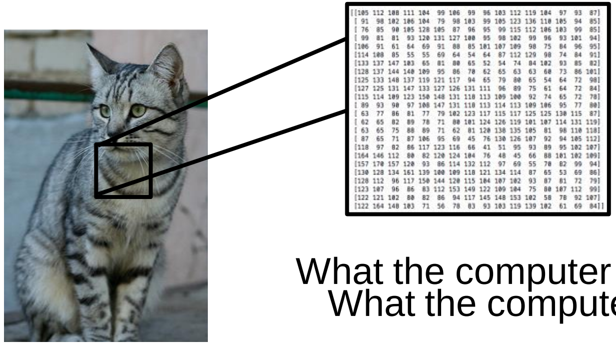
Machine learning thrives when it is **difficult to design an algorithm** to perform the task

Applications:

```
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
```





What the computer sees
What the computer sees

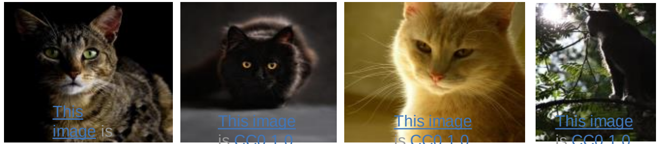
An image is just a big grid of numbers between [0, 255]:

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

Viewpoint Changes

All pixels change when the camera moves!

Illumination



Deformation

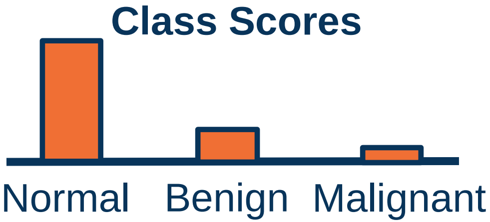
Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Why Image Classification is Hard

Application: Computer Vision

Model

Model



Model

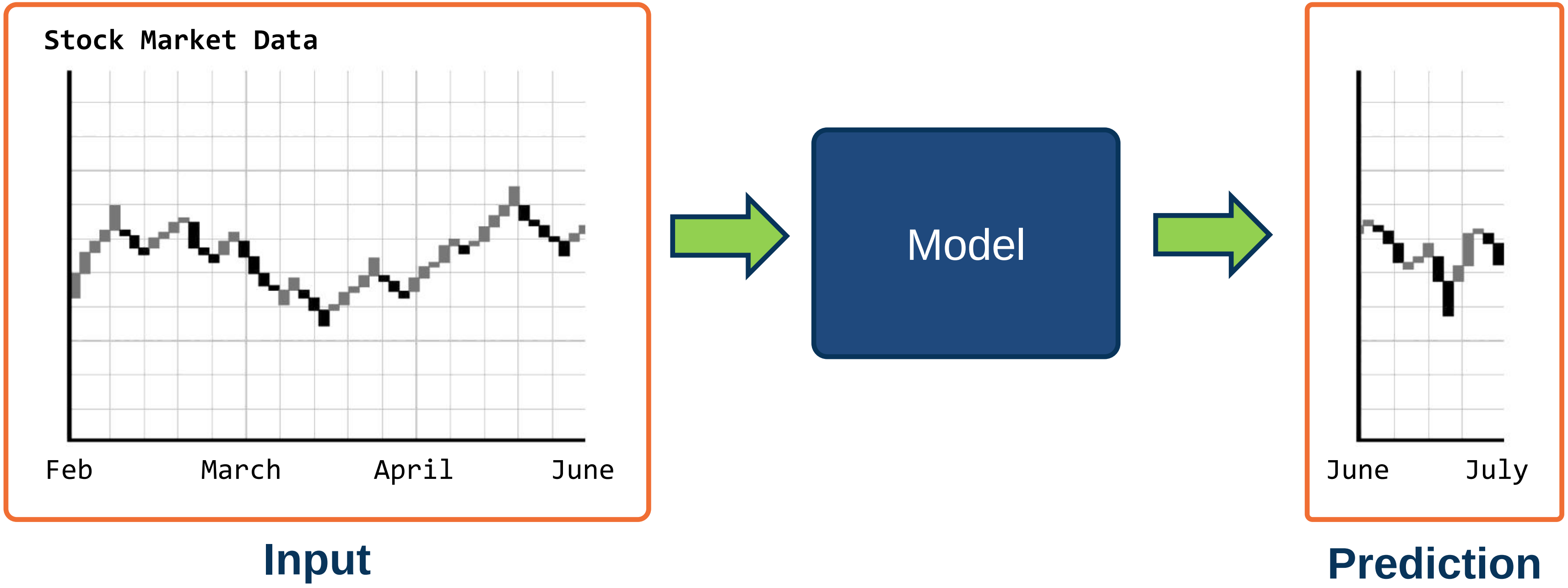
3D Reconstructions

Kanazawa *et al.*, 2018

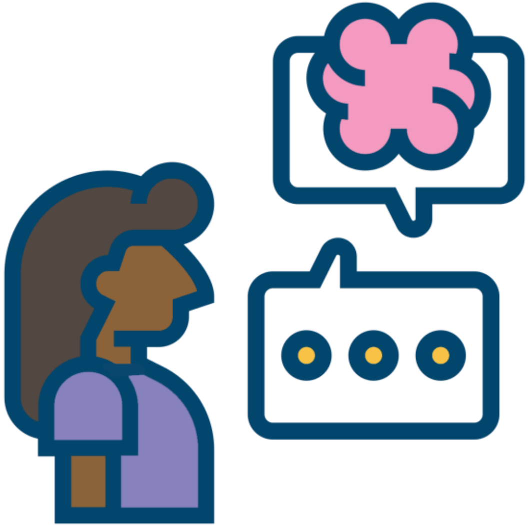
Example: Image Classification

Application: Time-Series Forecasting

Given a series of measurements, **output prediction for next time period**



Application: Natural Language Process (NLP)



Very large number of NLP sub-tasks:

- Syntax Parsing
- Translation
- Named entity recognition
- Summarization

Sequence modeling: Variable length sequential inputs and/or outputs

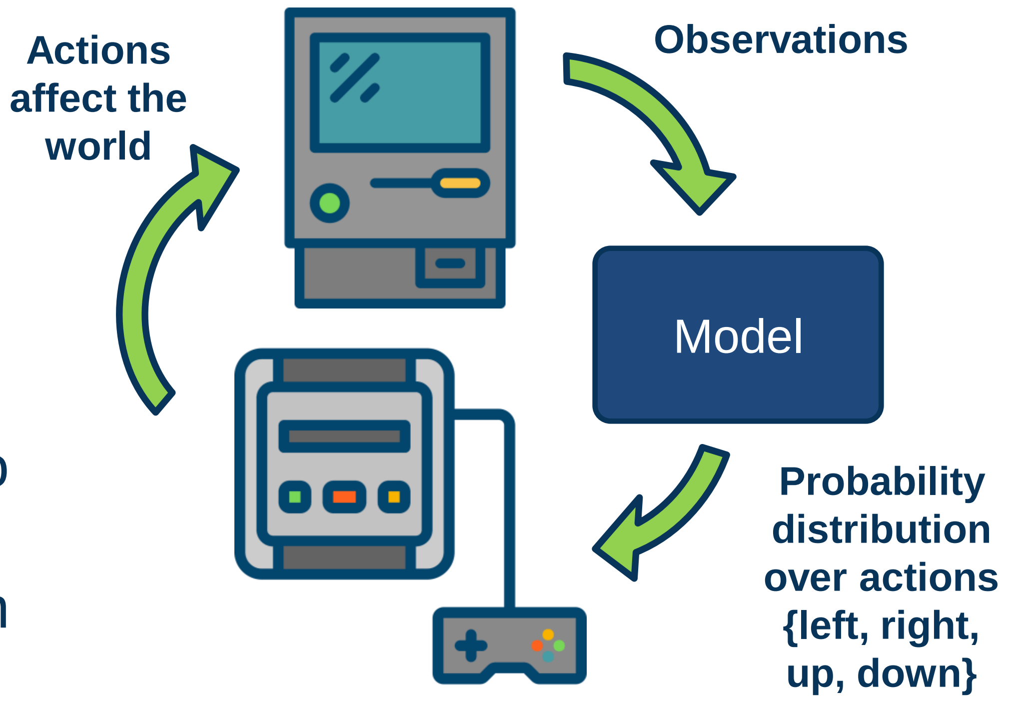
Recent progress: Large-scale language models

Decision-making tasks

- Sequence of inputs/outputs
- Actions affect the environment

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

Application:



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

- **Sense:** Perception
- **Plan:** Planning
- **Act:** Controls/Decision-Making

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

- Evolving landscape

Application:

Example: Robotics

Supervised Learning and Parametric Models

Georgia Tech

Supervised Learning

Unsupervised Learning

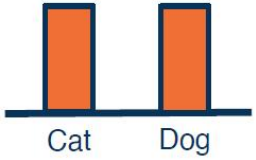
Reinforcement Learning

Types of Machine Learning

Supervised Learning

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

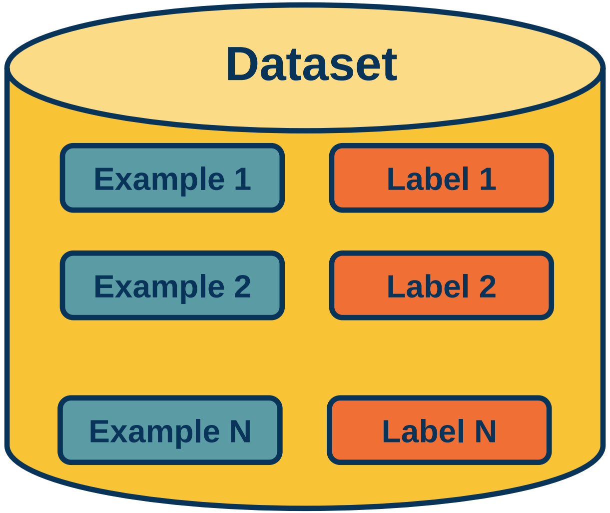
https://en.wikipedia.org/wiki/CatDog



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. $P(y|x)$

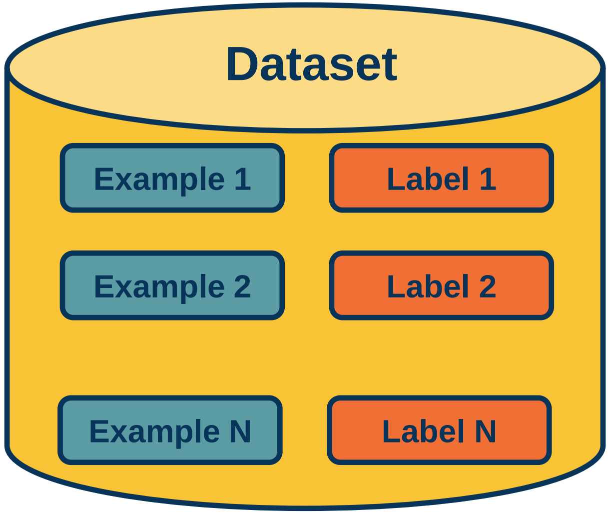
Terminology:

- Model / Hypothesis Class
  - $H: \{h: X \rightarrow Y\}$
  - Learning is search in hypothesis space
- Note **inputs x_i and y_i** are each represented as **vectors**

Dataset

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

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

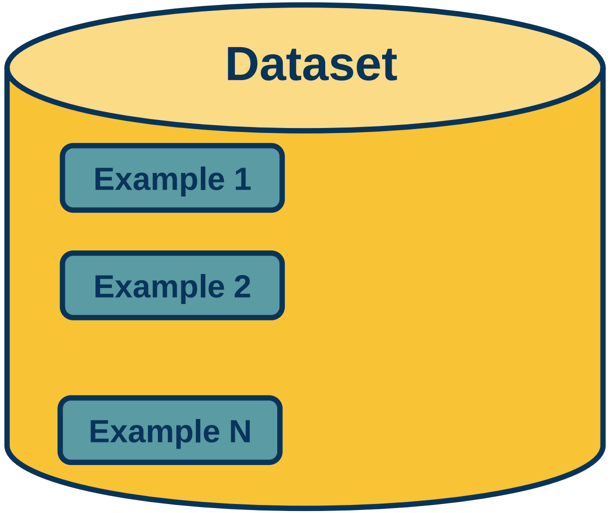


Dataset

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

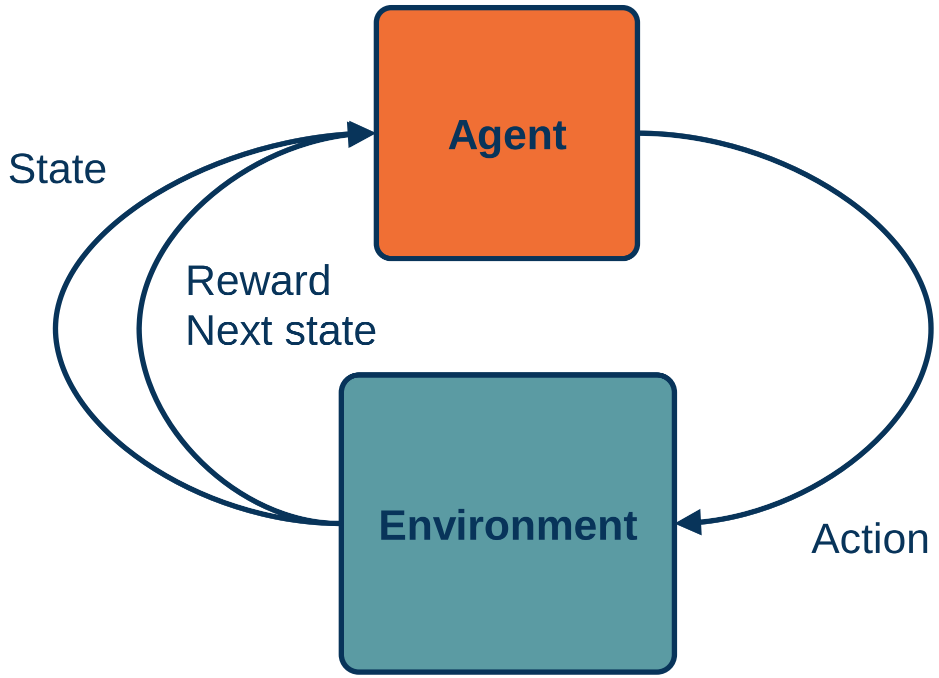
Unsupervised Learning

- 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, etc.



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!

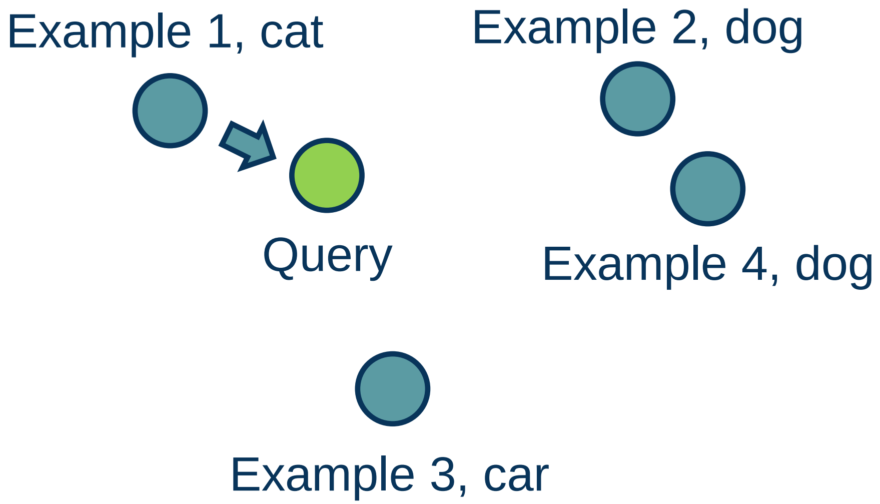
Non-Parametric Model

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

- **Nearest neighbor classifier**
- Decision tree

Capacity (size of hypothesis class) grow with size of training data!

Non-Parametric – Nearest Neighbor



Procedure: Take label of nearest example

- **Curse of Dimensionality**
  - Distances become meaningless in high 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

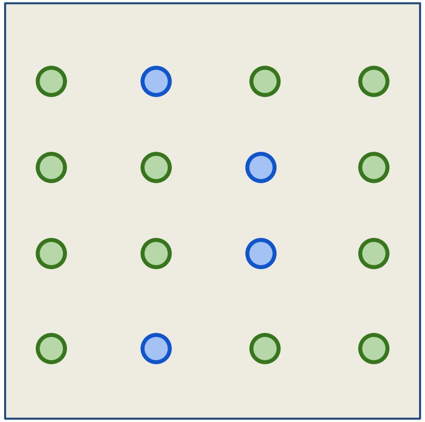
Problems with Instance-Based Learning

k-Nearest Neighbor on images almost **never used.**

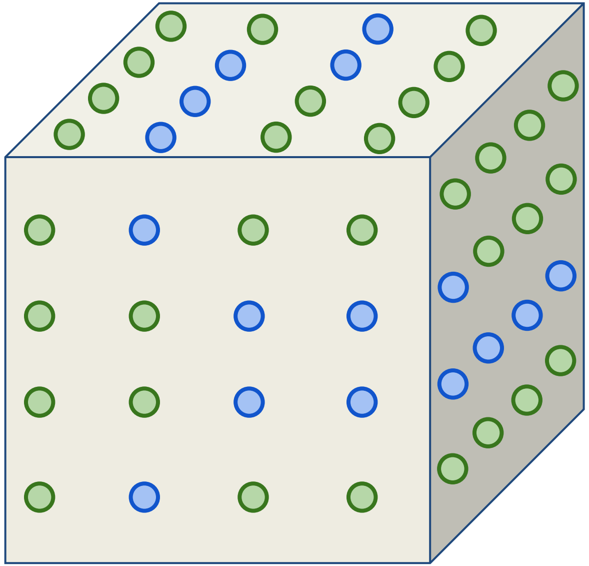
- Curse of dimensionality
  - Lots of weird behavior in high-dimensional spaces, e.g. orthogonality of random vectors, percentage of points around shell, etc.

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

Parametric Model

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

- Logistic regression/classification
- Neural networks

Capacity (size of hypothesis class) **does not** grow with size of training data!

Learning is **search**

Parametric – Linear Classifier

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

Training Stage:

Training Data $\{ (x_i,y_i) \} \rightarrow h$ (Learning)

Testing Stage

Test Data $x \rightarrow h(x)$ (Apply function, Evaluate error)

Probabilities to rescue:

X and Y are *random variables*

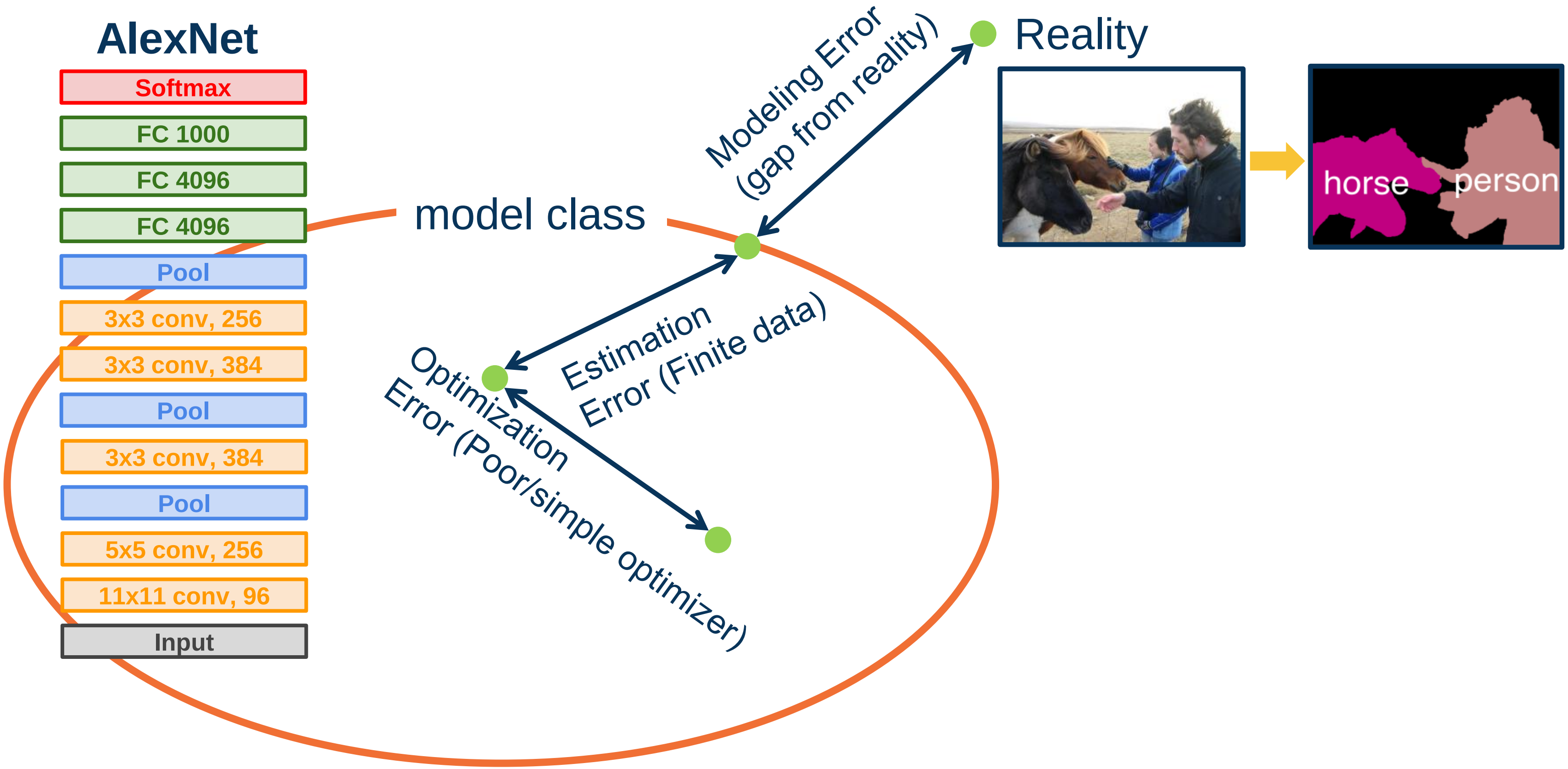
$D = (x_1,y_1), (x_2,y_2), \ldots, (x_N,y_N) \quad \sim P(X,Y)$

IID: Independent Identically Distributed

Both training & testing data sampled IID from P(X,Y)

Learn on training set

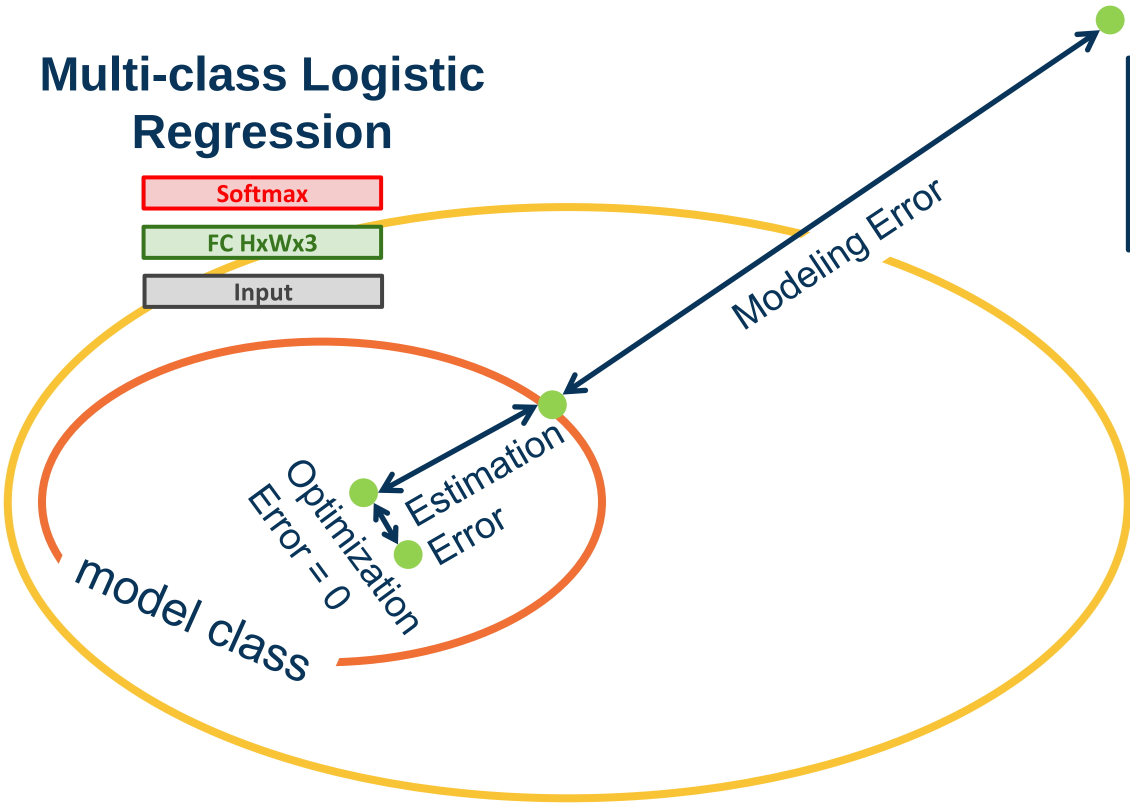
Have some hope of *generalizing* to test set



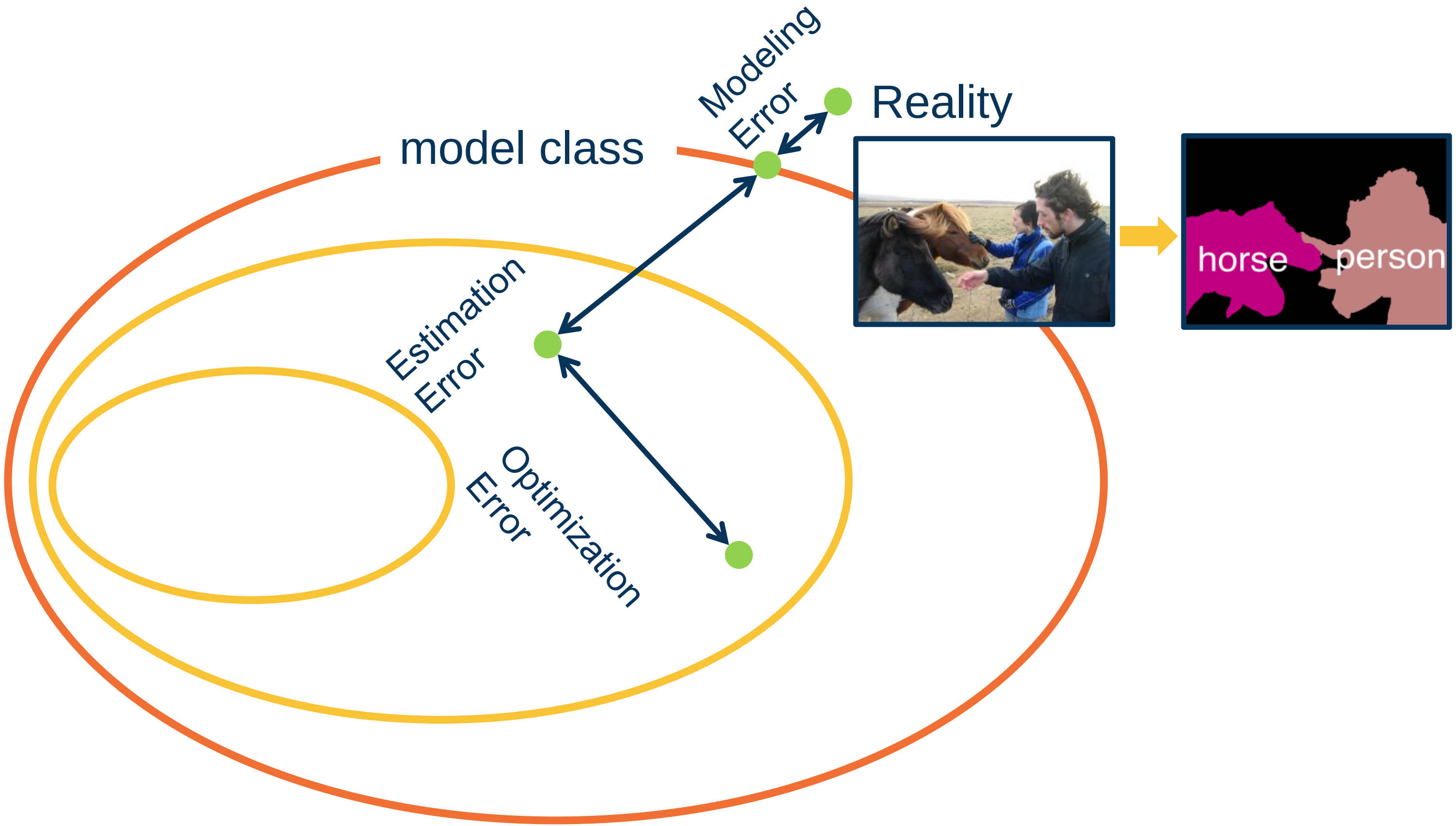
From: slides by Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

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

Generalization

20 years of research in Learning Theory oversimplified:

If you have:

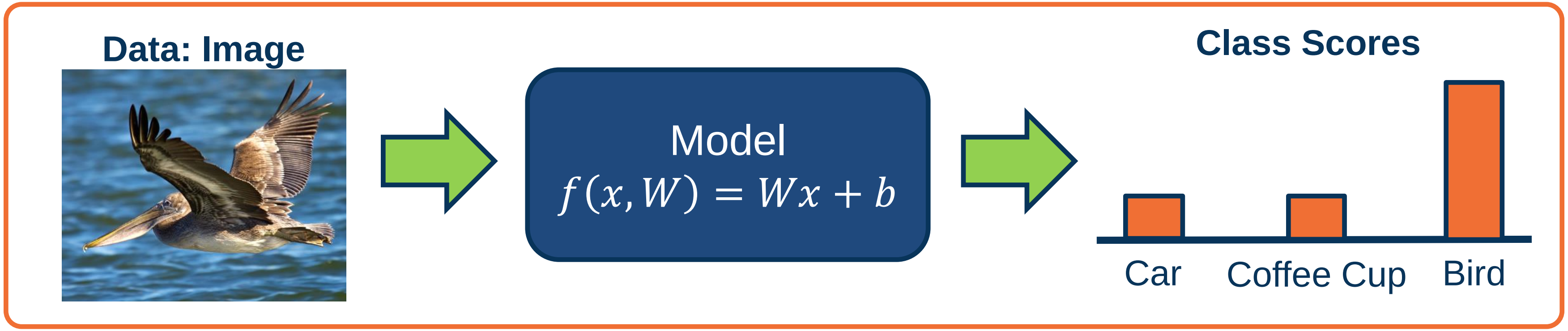
Enough training data D
and H is not too complex
then *probably* we can generalize to unseen test data

Caveats: A number of recent empirical results question our intuitions built from this clean separation.

Zhang et al., Understanding deep learning requires rethinking generalization

Guarantees

Georgia Tech



Input $\{X, Y\}$ where:

- X is an image
- Y is a **ground truth label** annotated by an expert (human)
- $f(x, W) = Wx + b$ is our model, chosen to be a linear function in this case
- W and b are the parameters (**weights**) of our model that must be learned

Example: Image Classification

Input image is **high-dimensional**

- For example n=512 so 512x512 image = **262,144** pixels
- Learning a classifier with high-dimensional inputs is hard

Before deep learning, it was typical to perform **feature engineering**

- Hand-design algorithms for converting raw input into a lower-dimensional set of features

Input Image

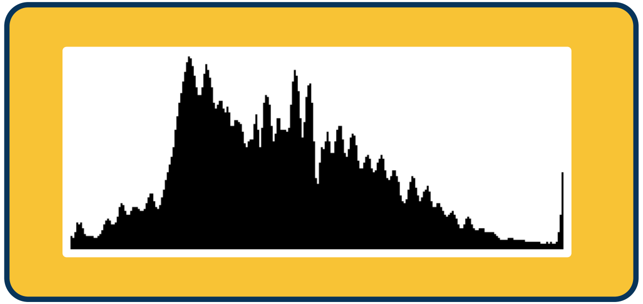
$$x = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nn} \end{bmatrix}$$

Example: Color histogram

- Vector of numbers representing number of pixels fitting within each bin
- We will later see that learning the feature representation itself is much more effective

Data: Image

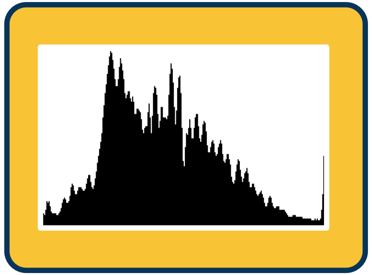
Features: Histogram



Example: Image Classification

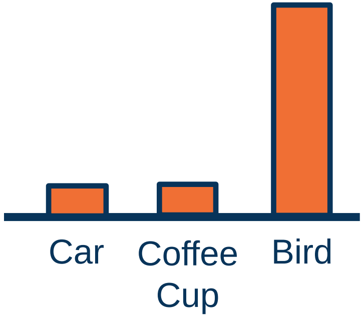
Data: Image

Features: Histogram



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

Class Scores



Input $\{X, Y\}$ where:

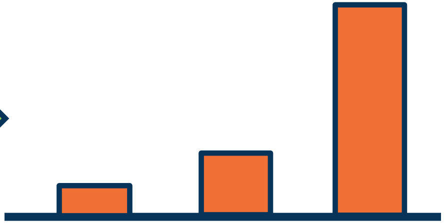
- X is an **image histogram**
- Y is a **ground truth label represented a probability distribution**
- $f(x, W) = Wx + b$ is our model, chosen to be a linear function in this case
- W and b are the **weights** of our model that must be learned

Data: Text

Model

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

Class Scores



Input $\{X, Y\}$ where:

- X is a sentence
- Y is a **ground truth label** annotated by an expert (human)
- $f(x, W) = Wx + b$ is our model, chosen to be a linear function in this case
- W and b are the **weights** of our model that must be learned

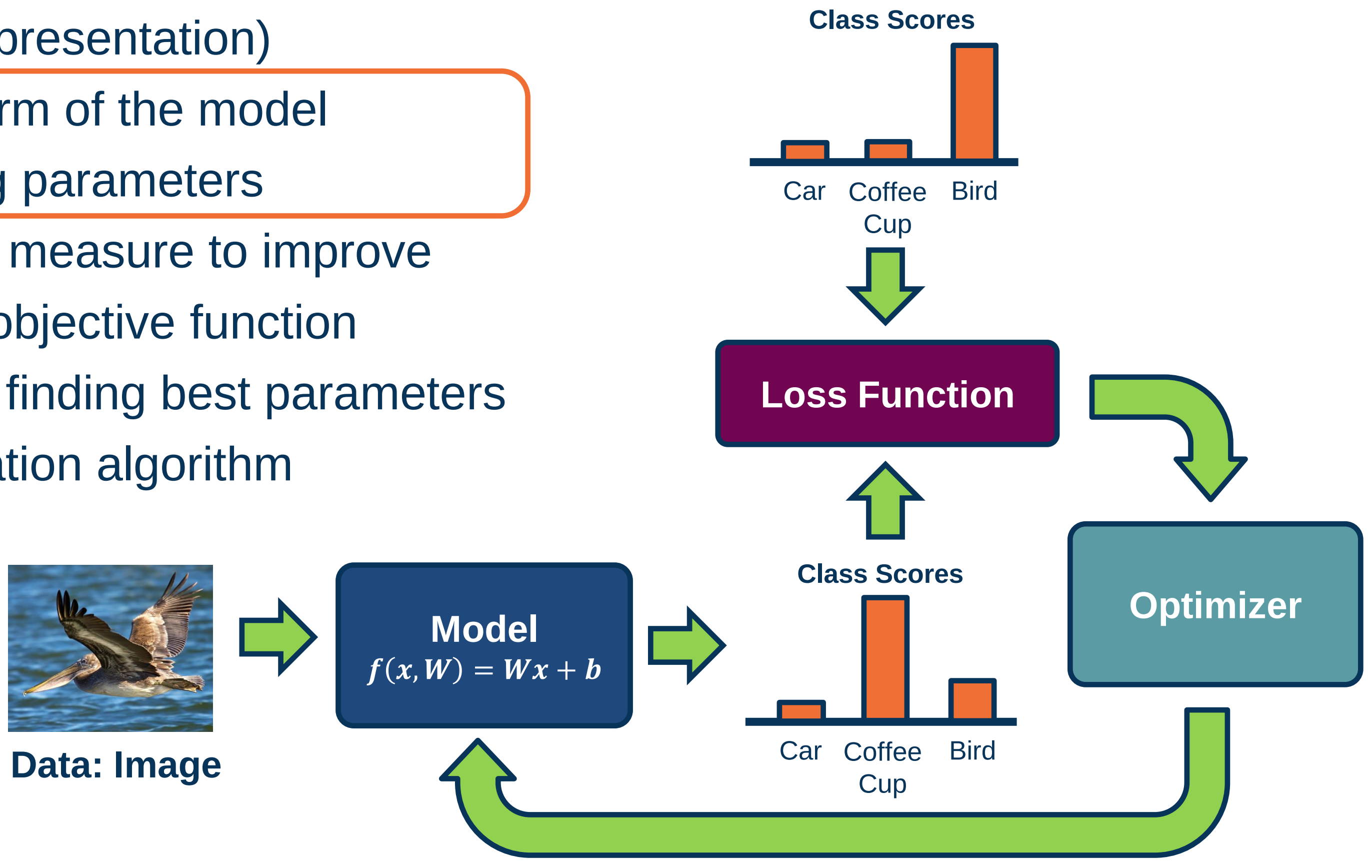
Word Histogram

Word	Count
this	1
that	0
is	2
...	
extremely	1
hello	0
onomatopoeia	0
…	

Example: Image Classification

Components of a Parametric Learning Algorithm

- 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

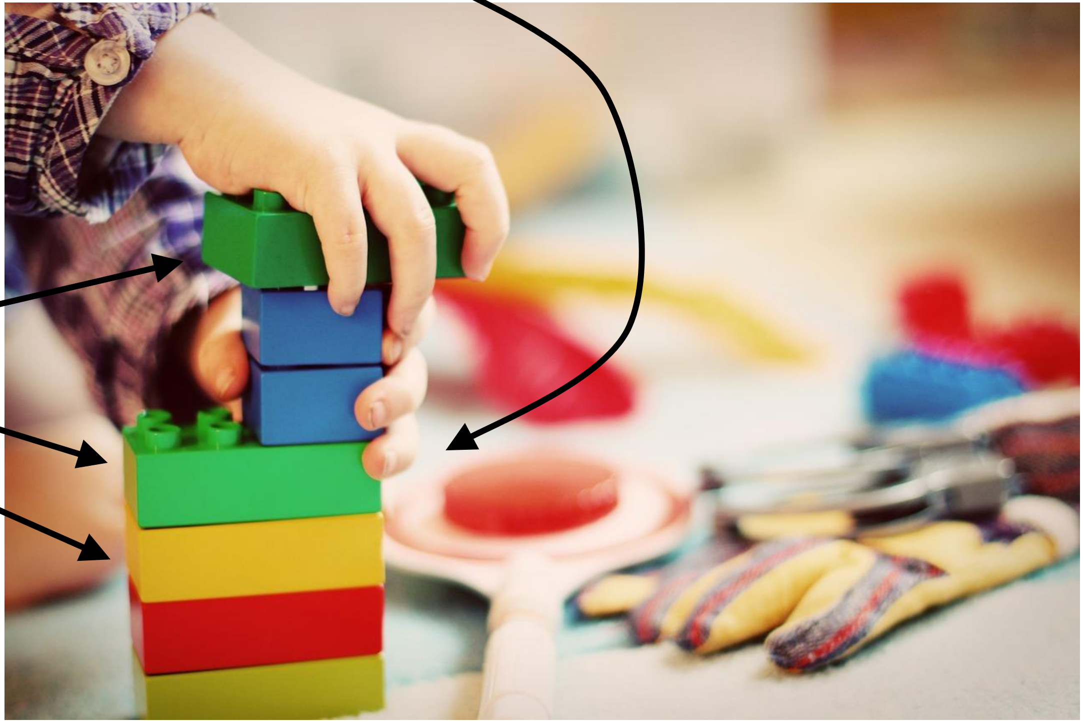


Components of a Parametric Model

Georgia Tech

Neural Network

Linear classifiers

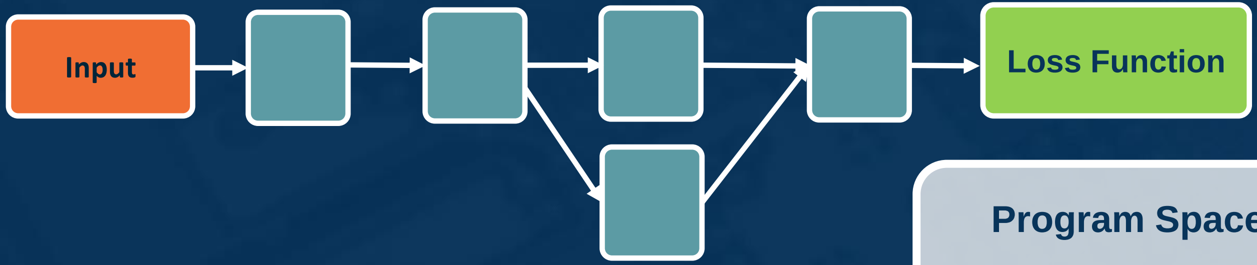


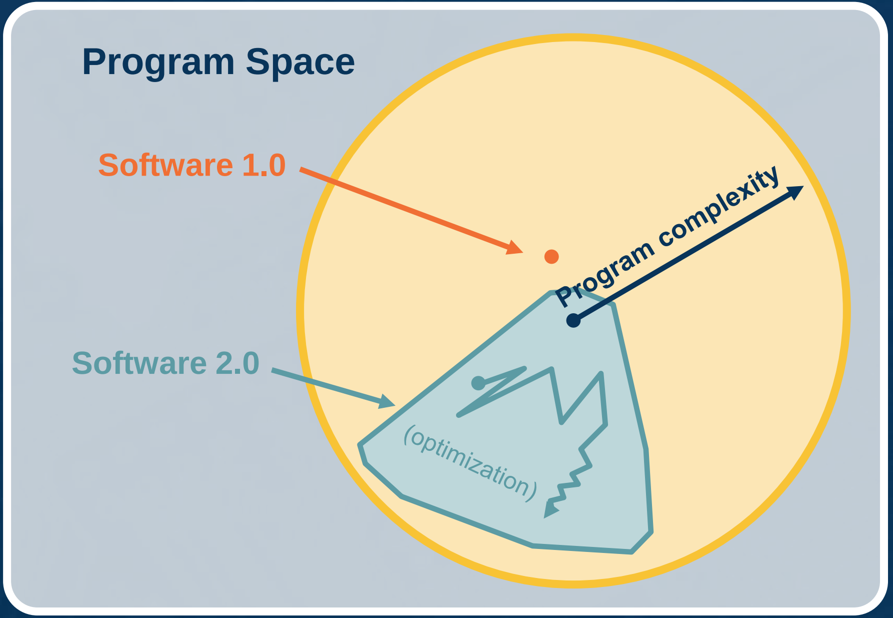
This image is CC0 1.0 public domain

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

Deep Learning as Legos

The Power of Deep Learning





Adapted from figure by Andrej Karpathy

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 →

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

$$\boldsymbol{W} \qquad \boldsymbol{x} \qquad \boldsymbol{b}$$

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

Weights

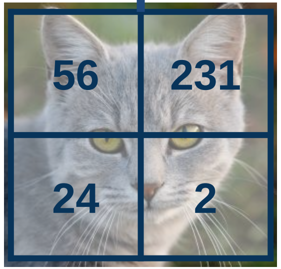
- We can move the bias term into the weight matrix, and a "1" at the end of the input
- Results in **one matrix-vector multiplication**!

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

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

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

Stretch pixels into column



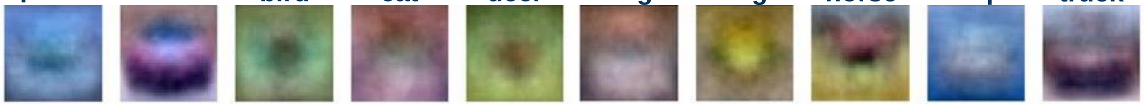
Input image

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

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

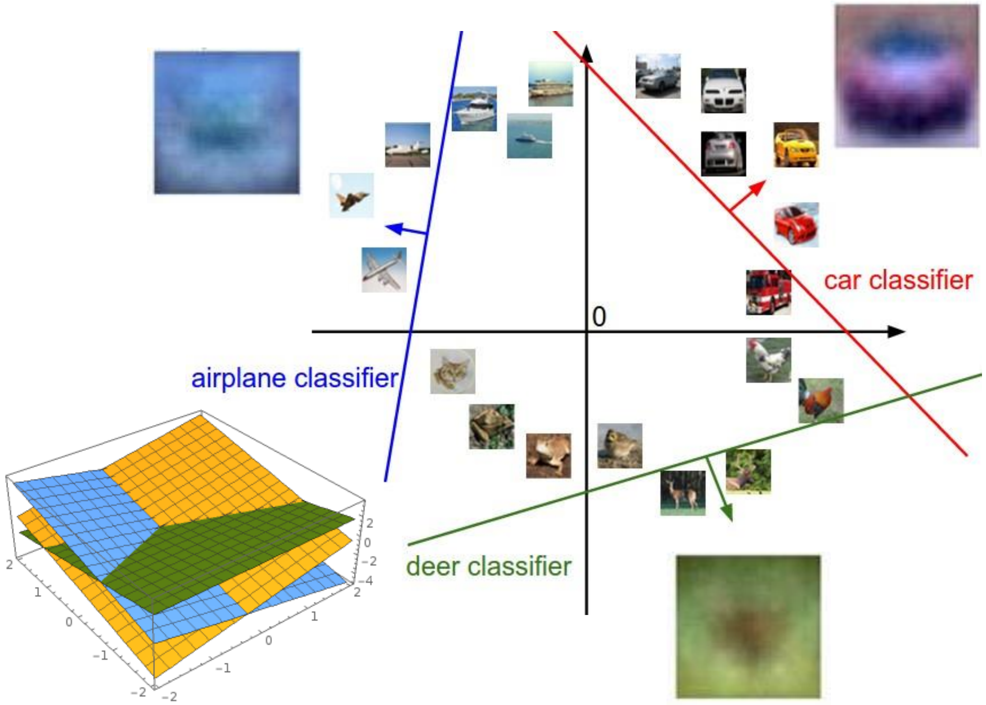
Visual Viewpoint

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

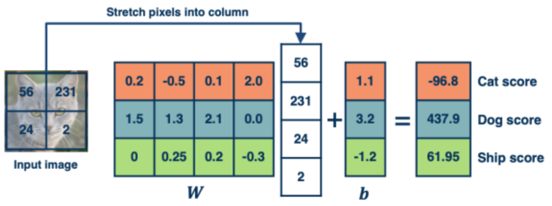
$$f(x, W) = Wx + b$$

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

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

Algebraic Viewpoint

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

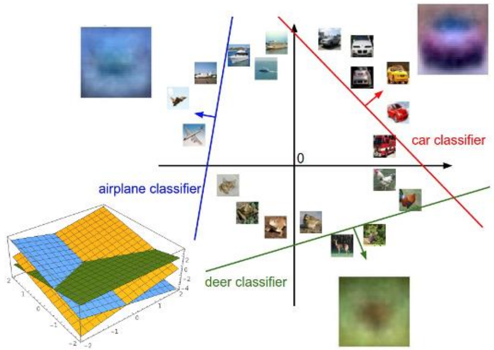


Visual Viewpoint

One template per class

Geometric Viewpoint

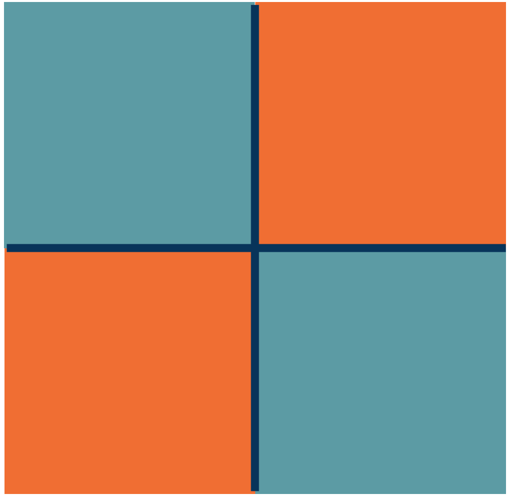
Hyperplanes cutting up space



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

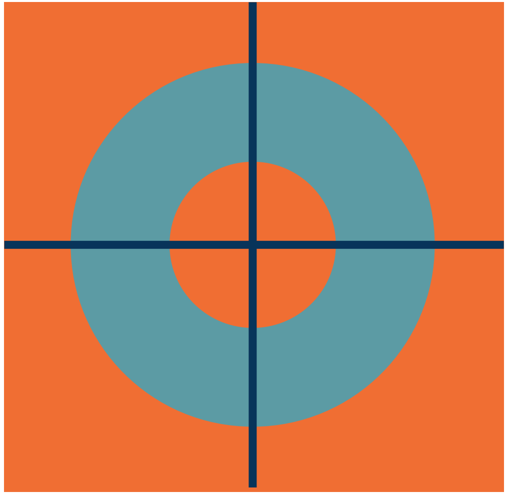
Class 1:
number of pixels > 0 odd

Class 2:
number of pixels > 0 even



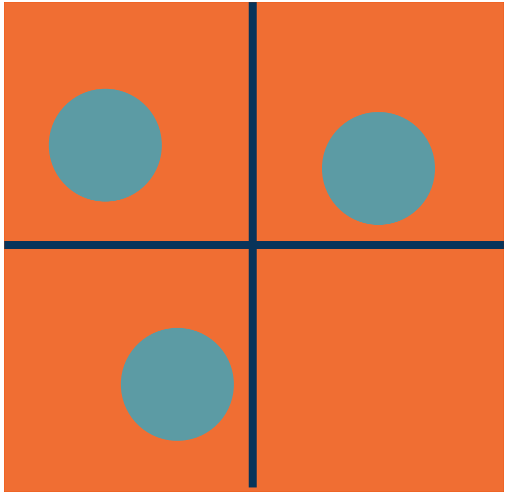
Class 1:
1 < = L2 norm < = 2

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

- We will learn complex, parameterized functions
  - Start w/ simple building blocks such as linear classifiers

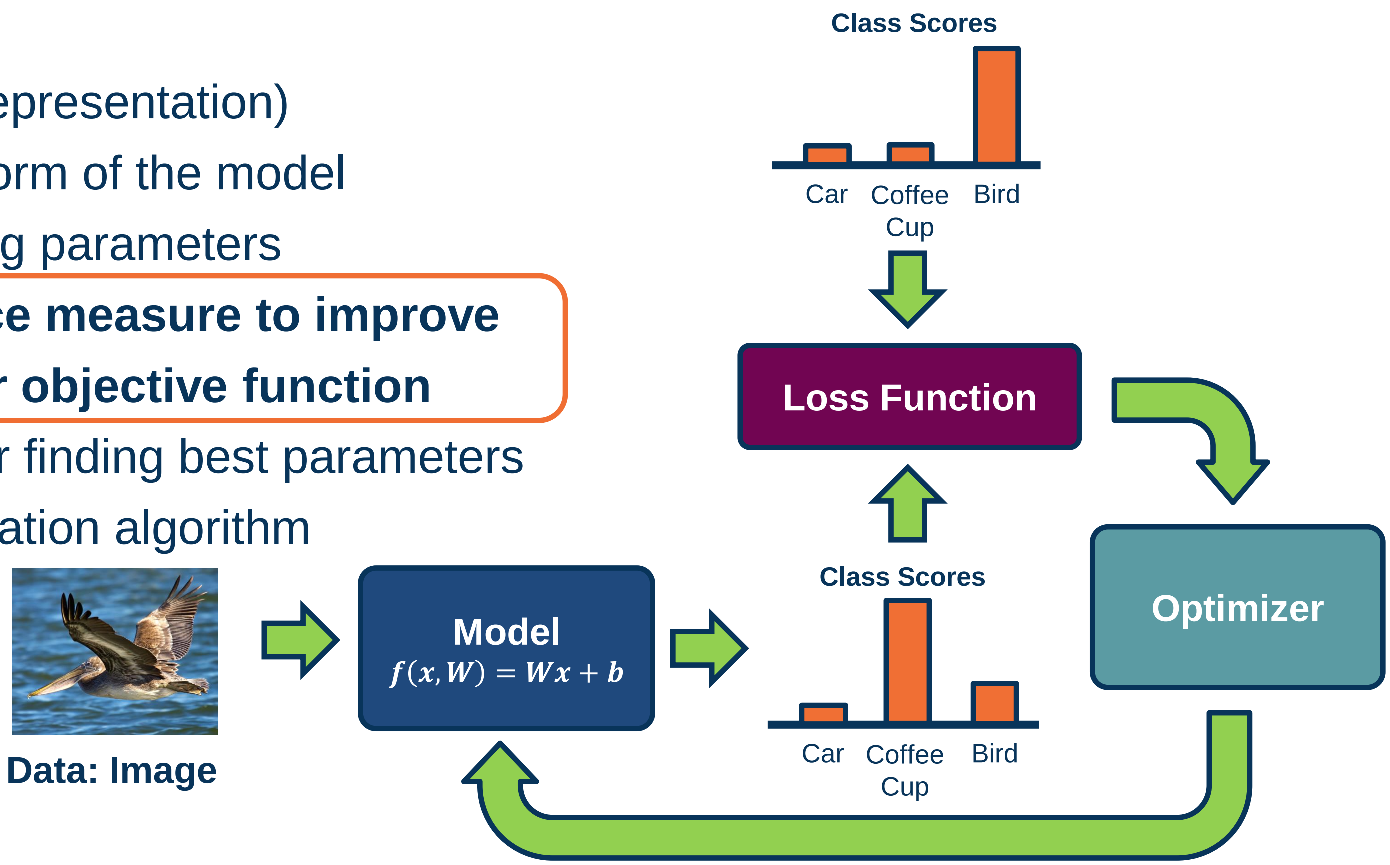
- Key is to learn parameters, but learning is hard
  - Sources of generalization error
  - Add bias/assumptions via architecture, loss, optimizer

- Components of parametric classifiers:
  - Input/Output, Model (function), Loss function, Optimizer
  - Example: Image/Label, Linear Classifier, Hinge Loss, ?

Summary

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



We need a performance measure to **optimize**

- Penalizes model for being wrong
- Allows us to modify the model to reduce this penalty
- Known as an **objective** or **loss** function

In machine learning we use **empirical risk minimization**

- Reduce the loss over the **training** dataset
- We **average** the loss over the training data

Given a dataset of examples:

$$\{(x_i, y_i)\}_{i=1}^{N}$$

Where x_i is image and

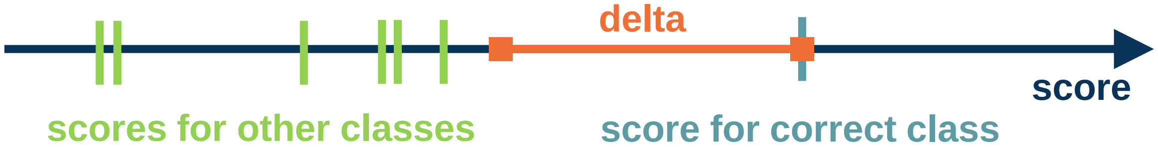
y_i is (integer) label

Loss over the dataset is a sum of loss over examples:

$$L = \frac{1}{N} \sum L(f(x_i, W), y_i)$$

Performance Measure

Multiclass SVM loss:



Given an example (x_i, y_i) where x_i is the image and where y_i is the (integer) label,

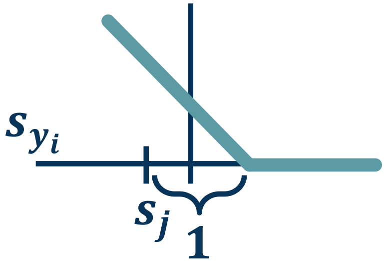
and using the shorthand for the scores vector: $s = f(x_i, W)$

the SVM loss has the form:

$$L_i = \sum_{j \neq y_i} \begin{cases} 0 & \text{if } s_{y_i} \geq s_j + 1 \\ s_j - s_{y_i} + 1 & \text{otherwise} \end{cases}$$

$$= \sum_{j \neq y_i} max(0, s_j - s_{y_i} + 1)$$

Example: "Hinge Loss"



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

Multiclass SVM loss:

Given an example (x_i, y_i) where x_i is the image and where y_i is the (integer) label,

and using the shorthand for the scores vector: $s = f(x_i, W)$

the SVM loss has the form:

$$L_i = \sum_{j \neq y_i} max(0, s_j - s_{y_i} + 1)$$

= max(0, 5.1 - 3.2 + 1)
+max(0, -1.7 - 3.2 + 1)
= max(0, 2.9) + max(0, -3.9)
= 2.9 + 0
= 2.9

Suppose: 3 training examples, 3 classes.
With some W the scores $f(x, W) = Wx$ are:



cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Losses:	**2.9**		

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

SVM Loss Example

Georgia Tech

Multiclass SVM loss:

Given an example (x_i, y_i) where x_i is the image and where y_i is the (integer) label,

and using the shorthand for the scores vector: $s = f(x_i, W)$

the SVM loss has the form:

$$L_i = \sum_{j \neq y_i} max(0, s_j - s_{y_i} + 1)$$

= max(0, 1.3 - 4.9 + 1)
  +max(0, 2.0 - 4.9 + 1)
= max(0, -2.6) + max(0, -1.9)
= 0 + 0
= 0

Suppose: 3 training examples, 3 classes.
With some W the scores $f(x, W) = Wx$ are:



cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Losses:		**0.0**	

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

Multiclass SVM loss:

Given an example (x_i, y_i) where x_i is the image and where y_i is the (integer) label,

and using the shorthand for the scores vector: $s = f(x_i, W)$

the SVM loss has the form:

$$L_i = \sum_{j \neq y_i} max(0, s_j - s_{y_i} + 1)$$

L = (2.9 + 0 + 12.9)/3
= **5.27**

Suppose: 3 training examples, 3 classes.
With some W the scores $f(x, W) = Wx$ are:



cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**
Losses:	2.9	0	12.9

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

- We will learn complex, parameterized functions
  - Start w/ simple building blocks such as linear classifiers

- Key is to learn parameters, but learning is hard
  - Sources of generalization error
  - Add bias/assumptions via architecture, loss, optimizer

- Components of parametric classifiers:
  - Input/Output, Model (function), Loss function, Optimizer
  - Example: Image/Label, Linear Classifier, Hinge Loss, ?

Summary