Topics:

- Supervised Learning, Linear Classification, Loss functions
- Gradient Descent

CS 4803-DL / 7643-A

ZSOLT KIRA

Machine Learning Applications

Georgia Tech

- **PS0 due tonight!**
  - Please do it, 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**: 147/175 enrolled
  - Enroll now! https://piazza.com/class/kjsselshfiz18c (Code: DL2021)
  - Make it active!

- **Office hours** start this week

Administrivia

- **Collaboration**
  - Only on HWs and project (not allowed in 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

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

Collaboration Policy

Georgia Tech

- **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

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

Georgia Tech

Machine Learning Overview

Georgia Tech

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."

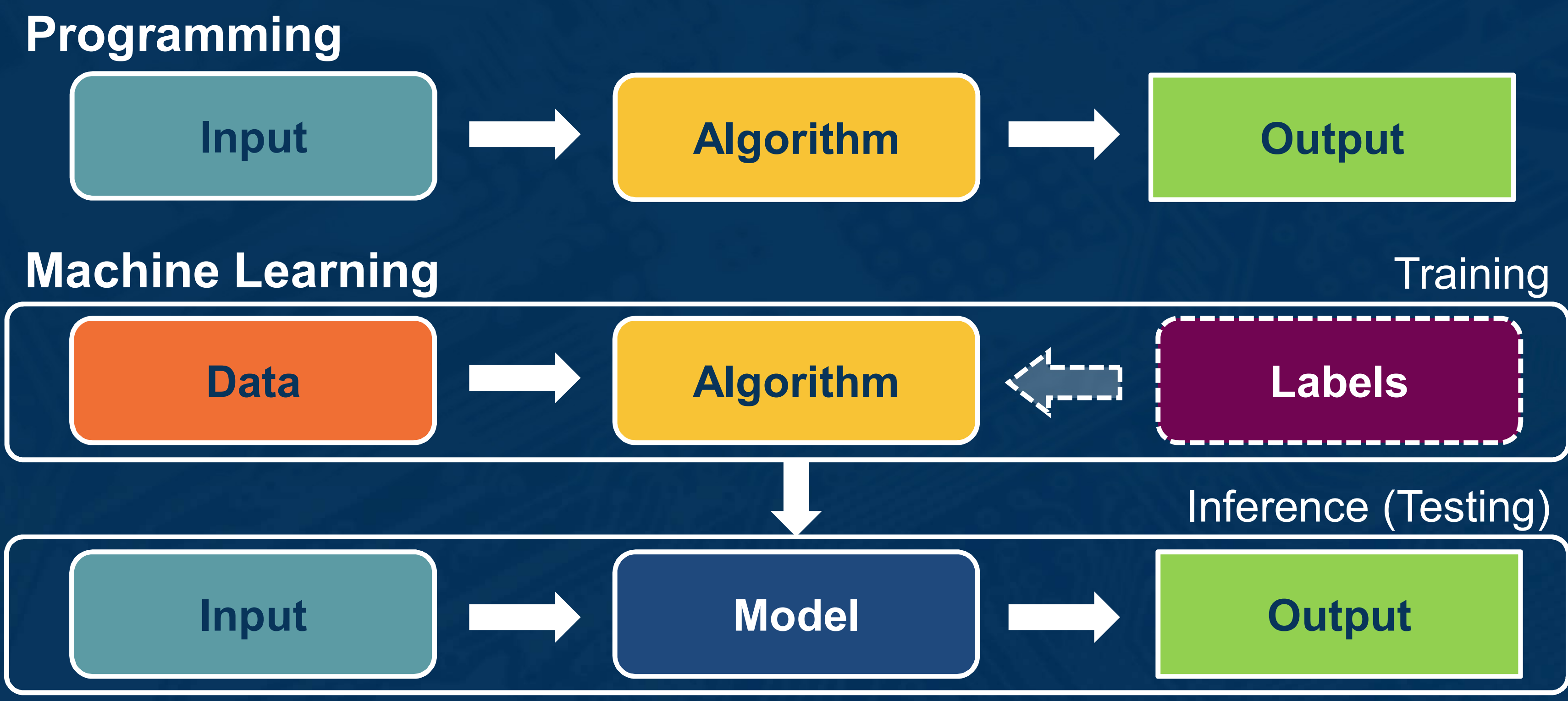
Tom Mitchell (Machine Learning, 1997)

Georgia Tech

Machine Learning is the study of algorithms that:

- **Improve** their **performance**
- on some **task(s)**
- Based on **experience** (typically data)

How is it Different than Programming?

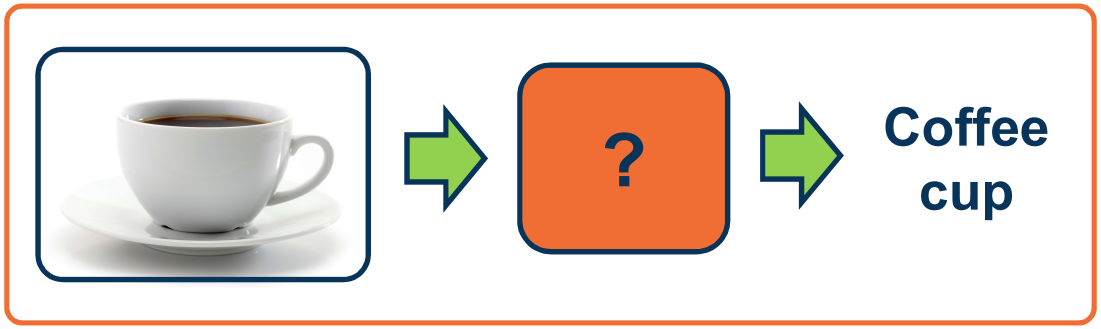


Machine learning thrives when it is **difficult to design an algorithm** to perform the task

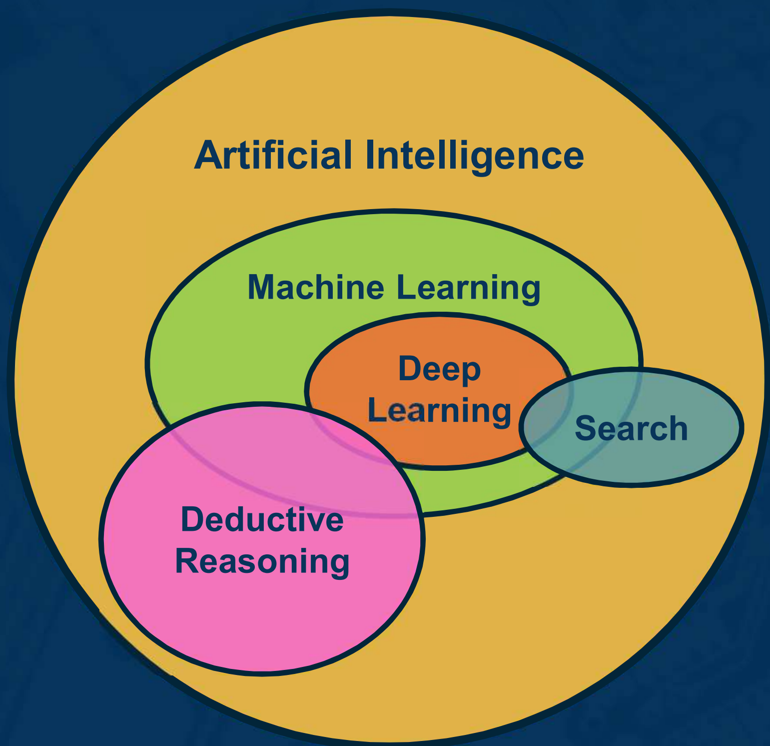
Applications:

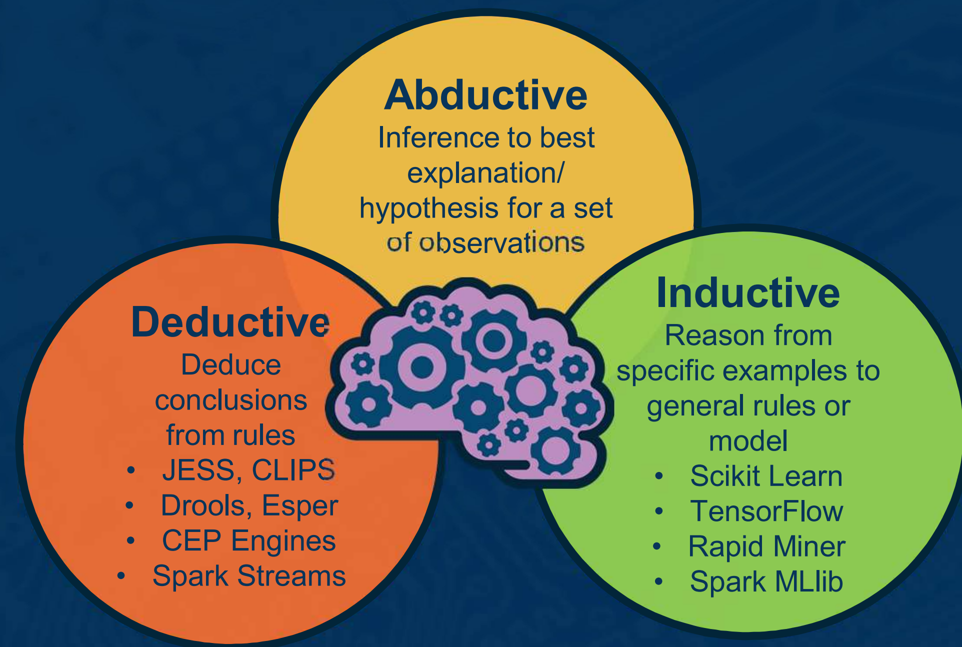
```
algorithm quicksort(A, lo, hi) is
    if lo < hi then
        p := partition(A, lo, hi)
        quicksort(A, lo, p − 1)
        quicksort(A, p + 1, hi)

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
```



Machine Learning and Artificial Intelligence





Adapted from: https://www.datanami.com/2018/03/20/u-s-pursues-abductive-reasoning-to-divine-intent/

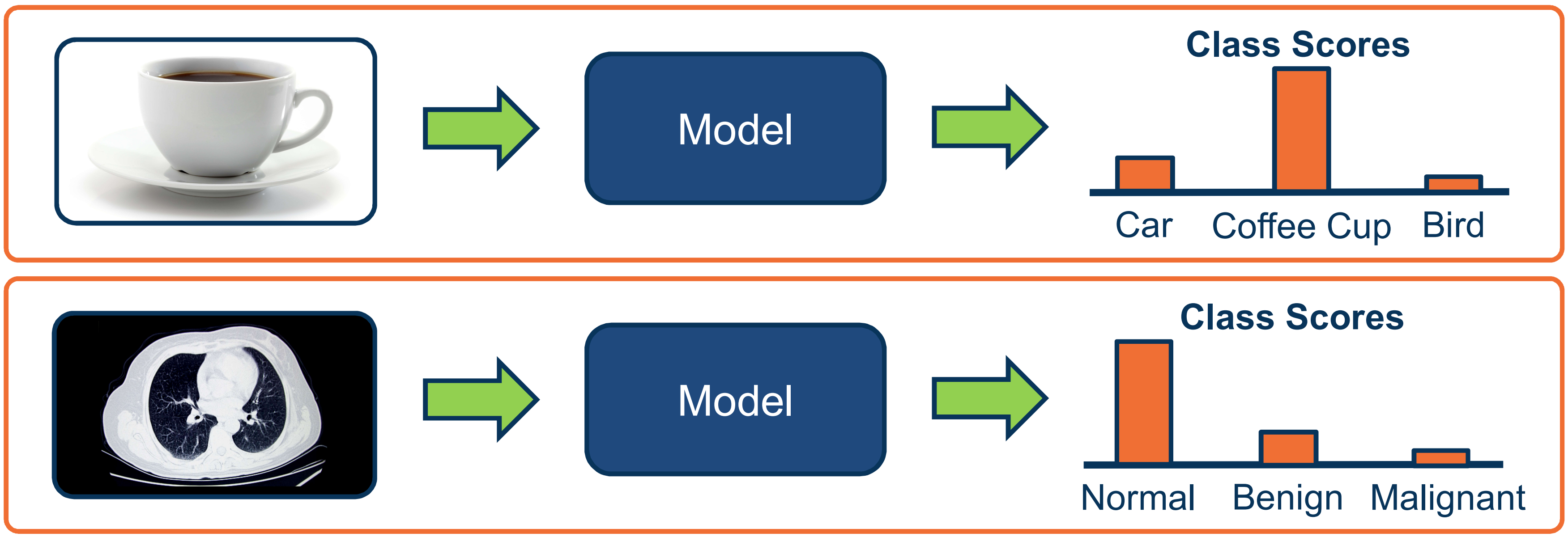
Georgia Tech

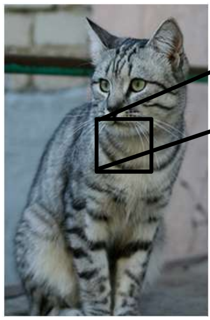
Example: Image Classification

Given an image, output class label

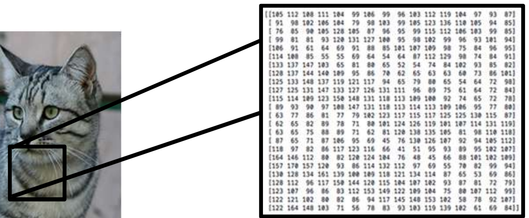
- Often output **probability distribution** over labels

Applications:





This image by Nikita is licensed under CC-BY 2.0

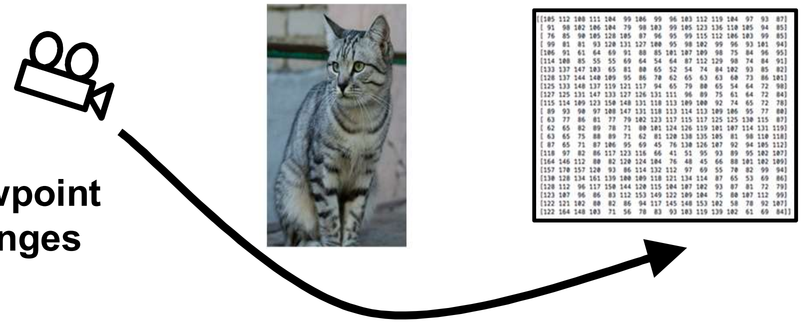


What the computer sees
What the computer sees

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

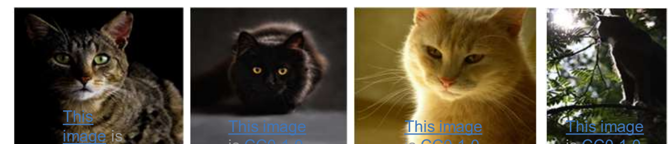
e.g. 800 x 600 x 3
(3 channels RGB)

Viewpoint Changes



All pixels change when the camera moves!

Illumination



Deformation

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

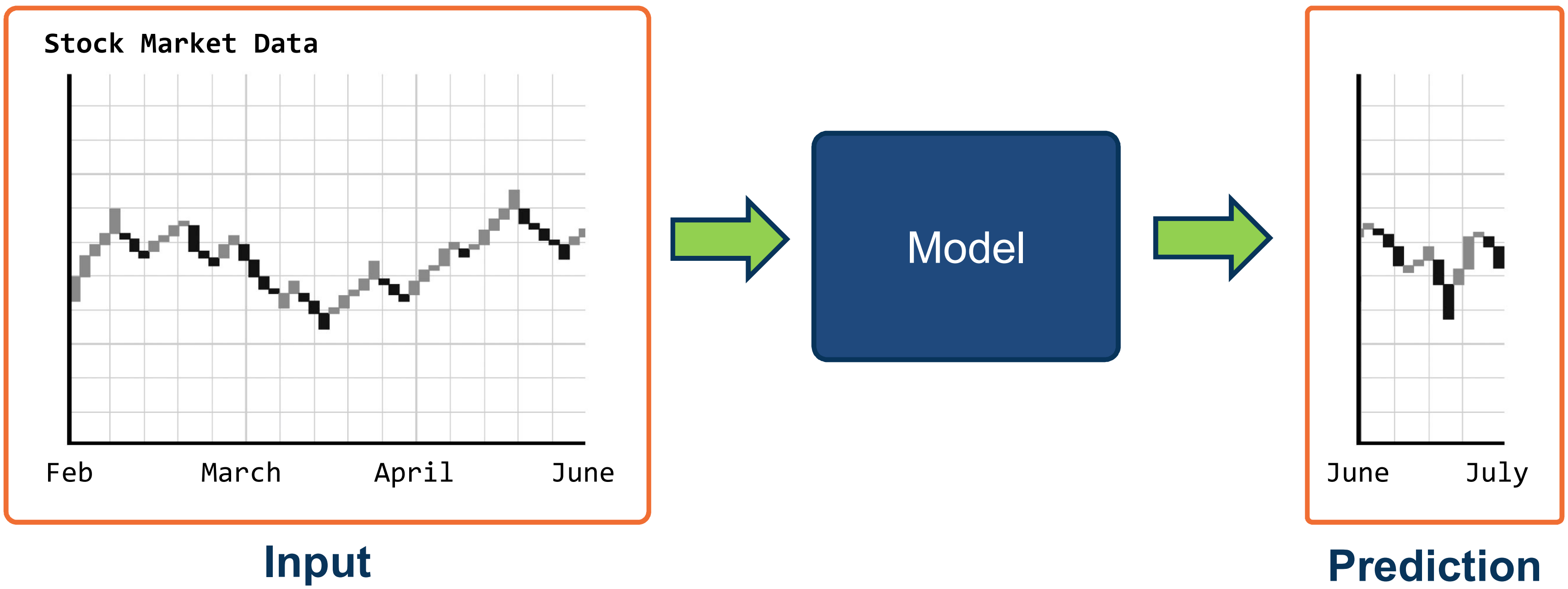
Why Image Classification is Hard

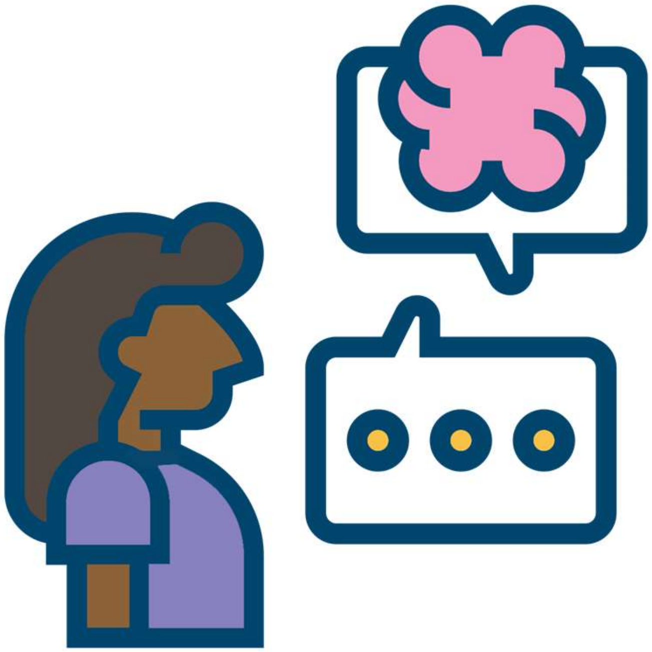
Georgia Tech

Example: Time Series Prediction

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

Application:





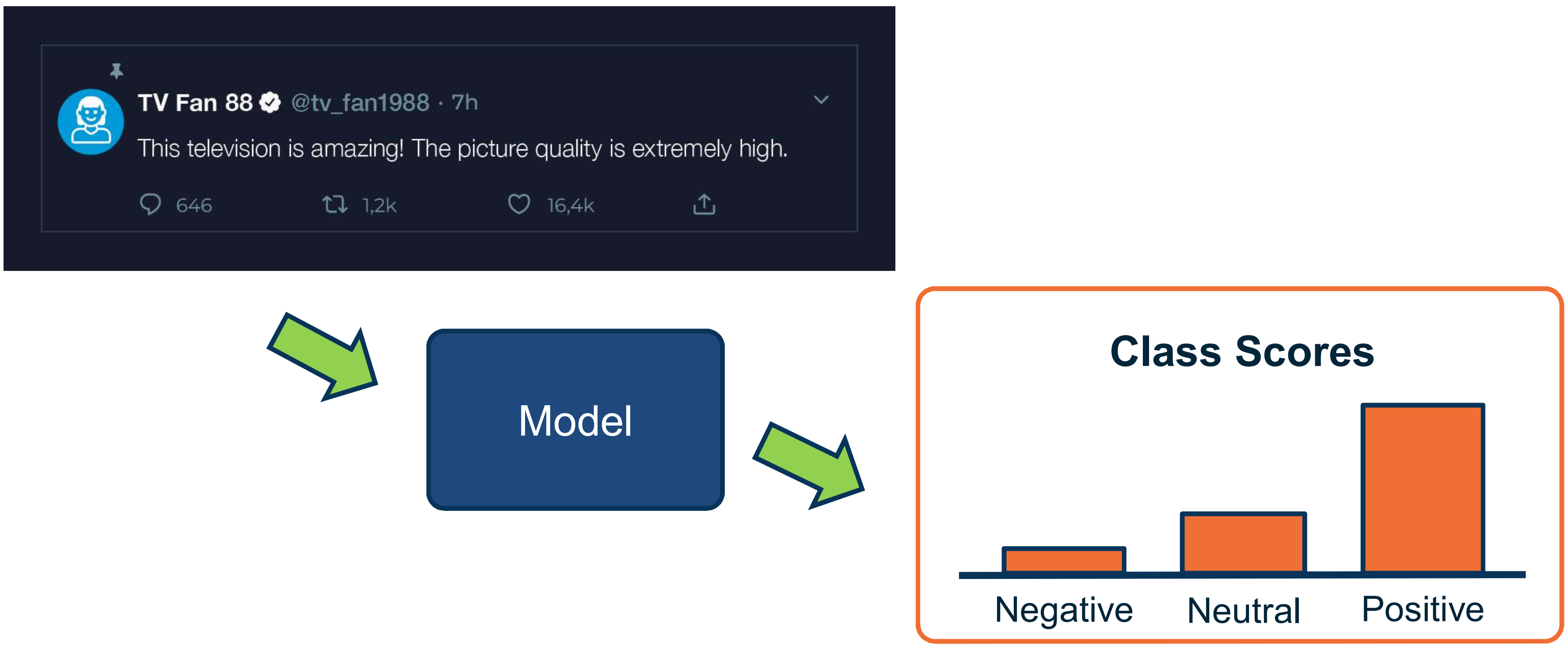
Very large number of NLP sub-tasks:

- Syntax Parsing
- Parts of speech
- Named entity recognition
- Summarization
- Similarity / paraphrasing

Different from classification: Variable length sequential inputs and/or outputs

Example: Natural Language Processing (NLP)

Sentiment Analysis:



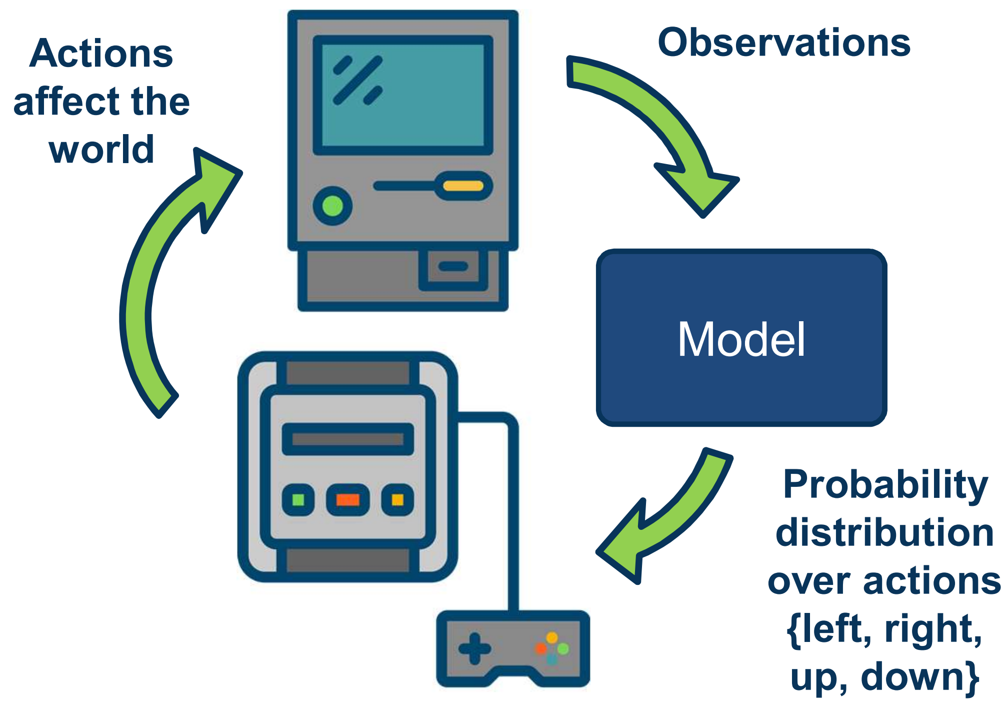
Example: Natural Language Processing (NLP)

Decision-making tasks

- Sequence of inputs/outputs
- Actions affect the environment

Combination of perception and decision-making/controls

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

Georgia Tech

Supervised Learning and Parametric Models

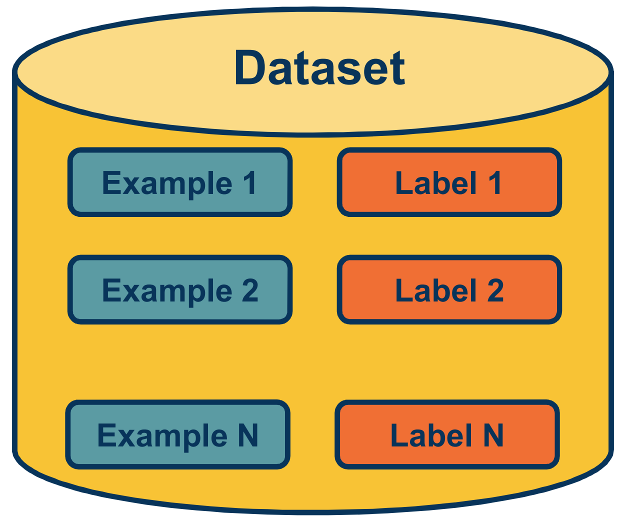
Supervised Learning

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

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**



Supervised Learning

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

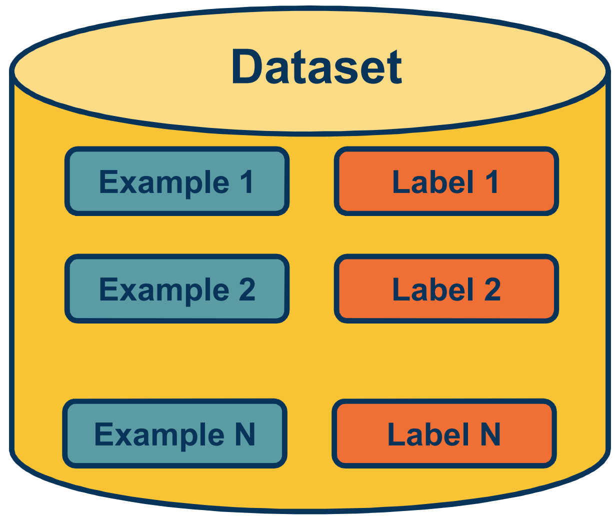
Terminology:

- 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

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

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

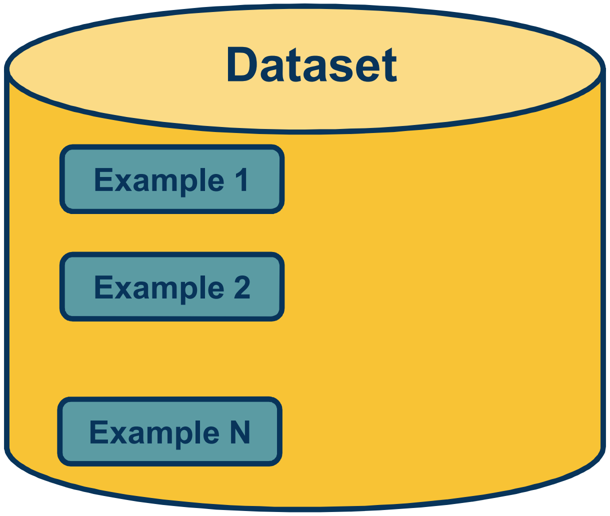


Unsupervised Learning

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

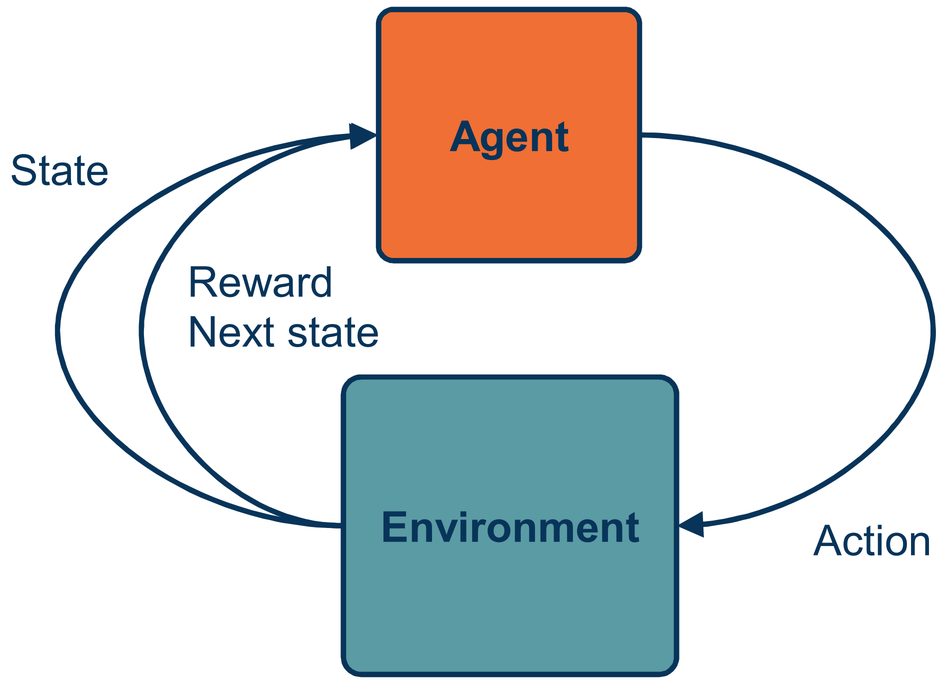
Dataset

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

Types of Machine Learning

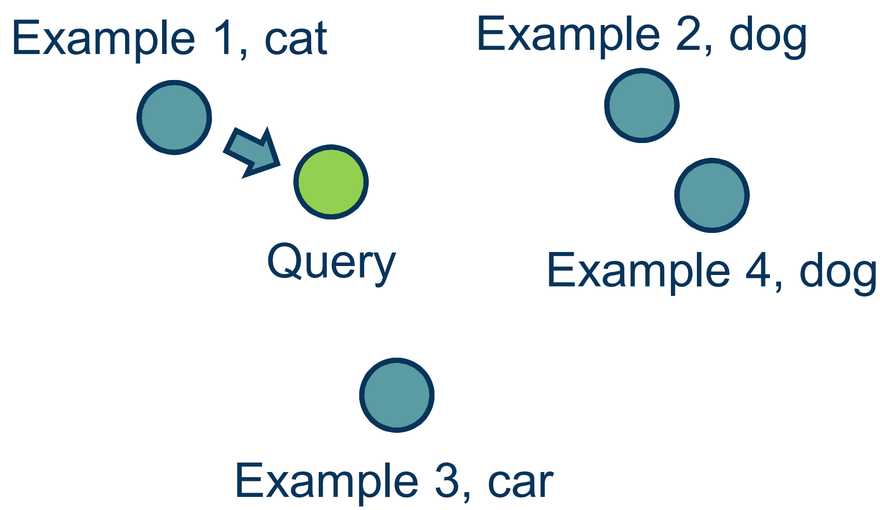
Non-Parametric Model

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

- **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 neighbour search
- **Doesn’t work well when large number of irrelevant features**
  - Distances overwhelmed by noisy features
- **Curse of Dimensionality**
  - Distances become meaningless in high dimensions

Problems with Instance-Based Learning

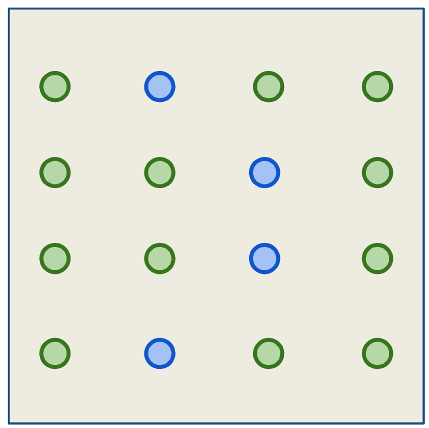
Georgia Tech

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

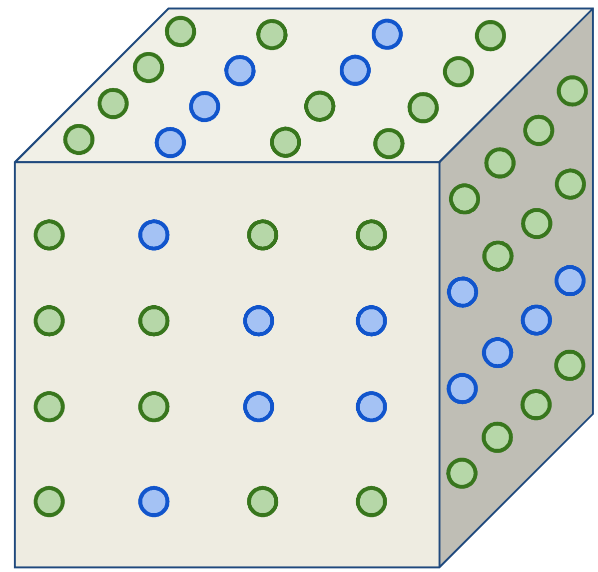
- Curse of dimensionality
  - Lots of weird behavior in high-dimensional spaces, e.g. orthogonality of random vectors, percentage of points around shell, etc.

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

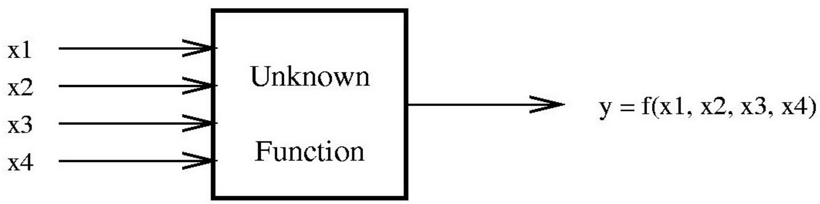
$$f(x, W) = Wx + b$$

Procedure:

Calculate score per class for example

Return label of maximum score (argmax)

A Learning Problem



Example	x_1	x_2	x_3	x_4	y
1	0	0	1	0	0
2	0	1	0	0	0
3	0	0	1	1	1
4	1	0	0	1	1
5	0	1	1	0	0
6	1	1	0	0	0
7	0	1	0	1	0

No Assumptions means no learning

Learning from a Broader Perspective

Georgia Tech

Training Stage:

Training Data { (x_i, y_i) } → h (Learning)

Testing Stage

Test Data x → h(x) (Apply function, Evaluate error)

Procedural View of ML

Georgia Tech

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

Statistical View of ML

20 years of research in Learning Theory oversimplified:

If you have:

Enough training data D

and H is not too complex

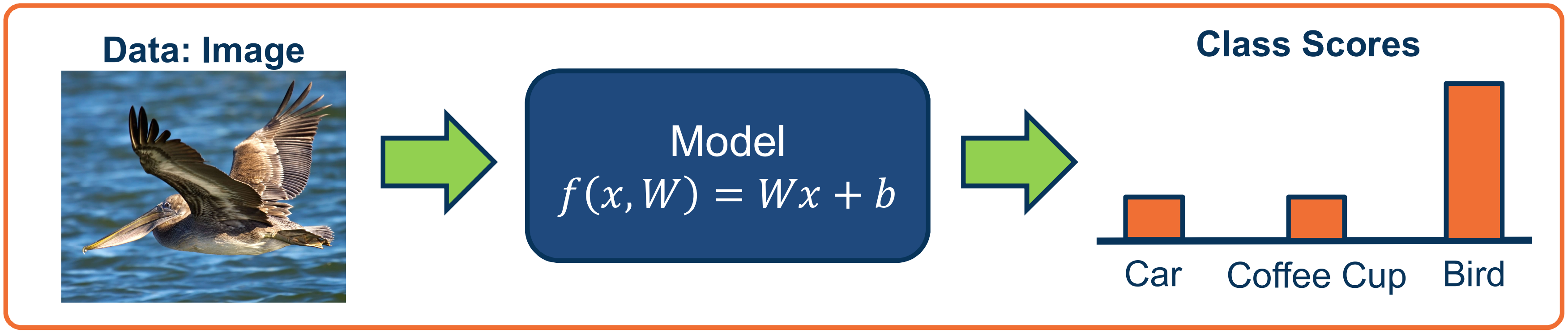
then *probably* we can generalize to unseen test data

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

Guarantees

Georgia Tech

Example: Image Classification



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

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

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

Input Representation: Feature Engineering

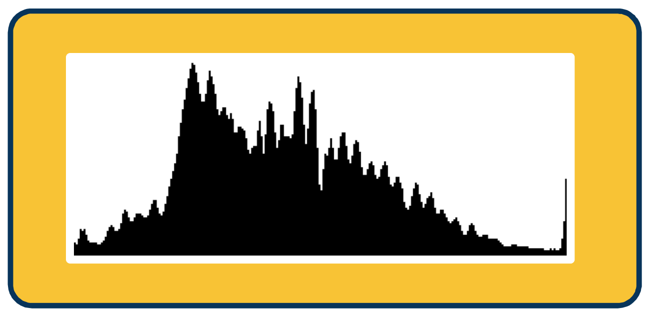
Georgia Tech

Example: Color histogram

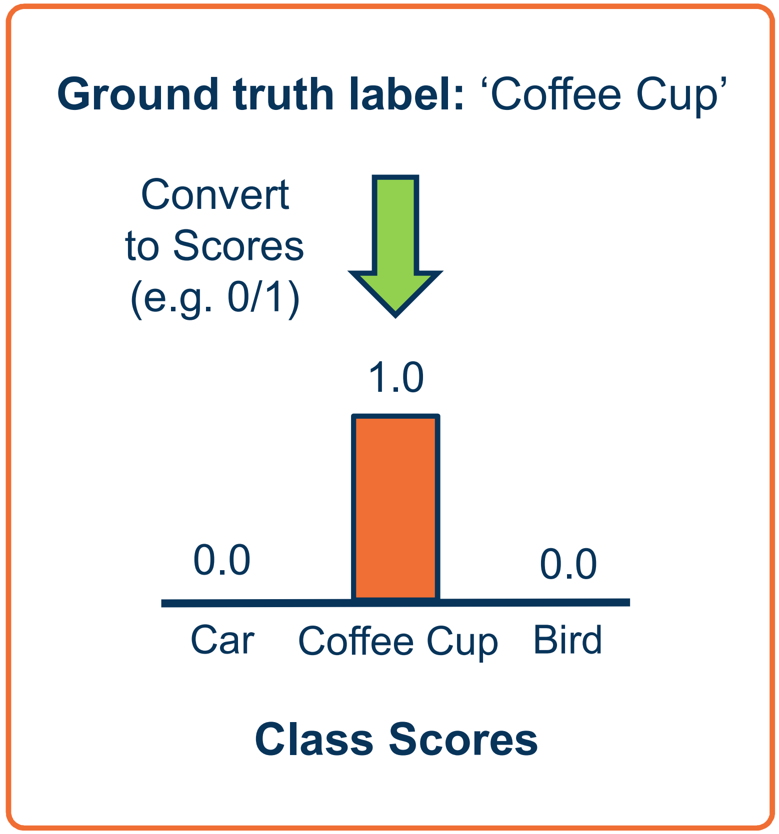
- Vector of numbers representing number of pixels fitting within each bin
- We will later see that learning the feature representation itself is much more effective

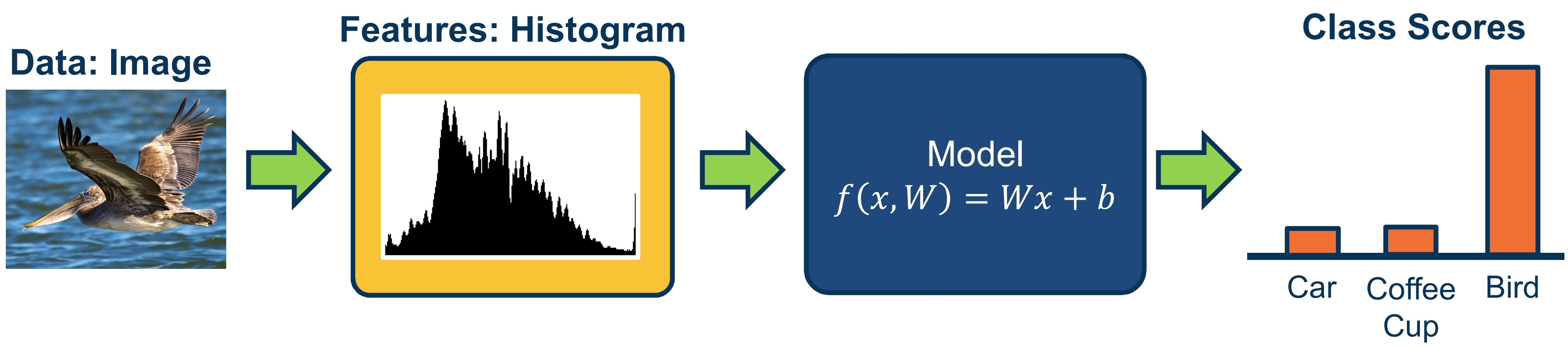
Data: Image

Features: Histogram



- **Labels are categories, but we need a numerical representation**
  - Assigning number to each category is arbitrary
- Instead, represent **probability distribution** over categories
- Ground truth label then becomes a probability distribution where the correct category probability is 1, and all others are 0
- Note for **regression** this is not an issue as the ground truth label (e.g. housing prices) is a number already





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

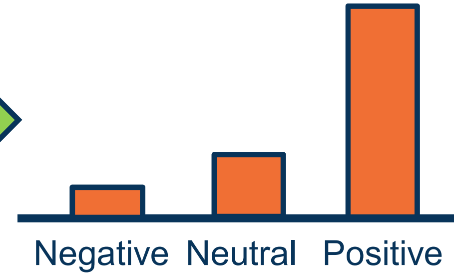
- X is an **image histogram**
- Y is a **ground truth label represented a probability distribution**
- $f(x, W) = Wx + b$ is our model, chosen to be a linear function in this case
- W and b are the **weights** of our model that must be learned

Example: Image Classification

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

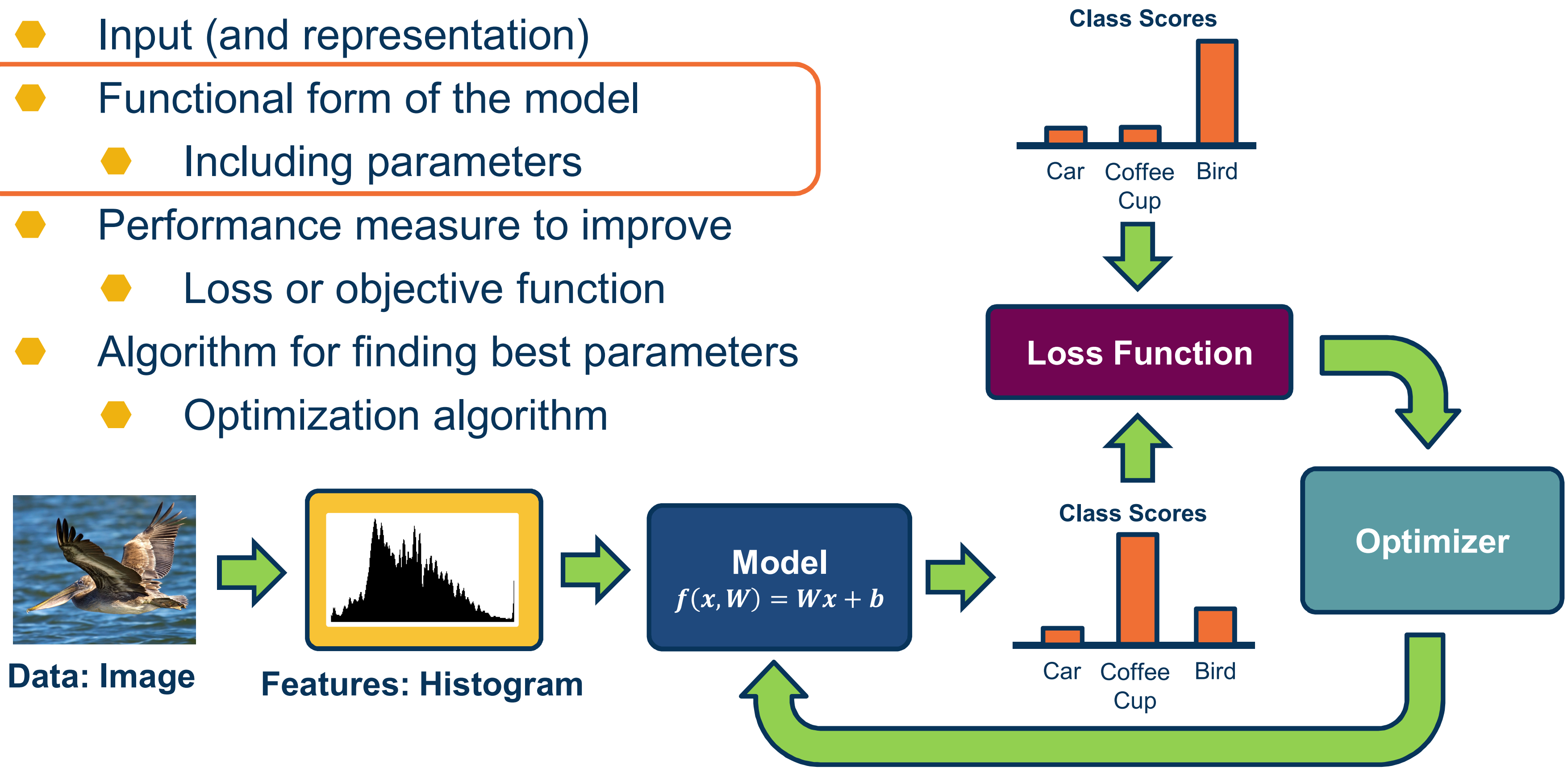
Word	Count
this	1
that	0
is	2
...	
extremely	1
hello	0
onomatopoeia	0
…	

Example: Image Classification

Georgia Tech

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

$$f(x, w) = y$$

Classifier

Input (vector)

Weights

Output (scalar or vector)

- **Input:** Continuous number or vector
- **Output:** A continuous number
  - For classification typically a **score**
  - For regression what we want to regress to (house prices, crime rate, etc.)
- w **is a vector and weights** to optimize to fit target function

Model: Discriminative Parameterized Function

Neural Network

Linear classifiers

[This image](#) is [CC0 1.0](#) public domain

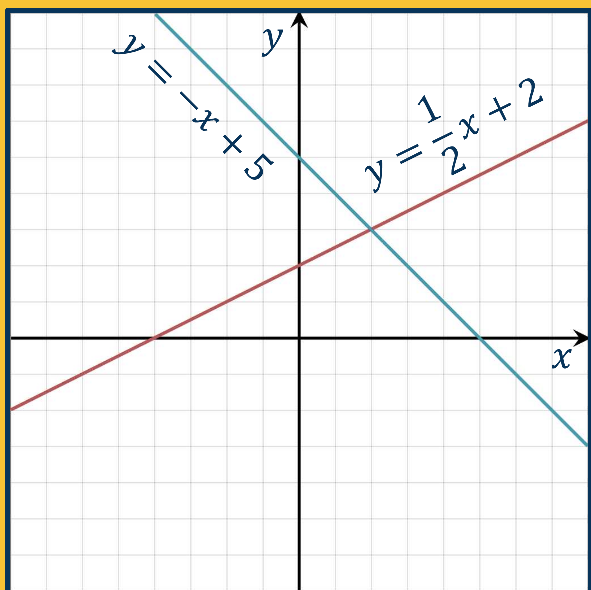
Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Deep Learning as Legos

Georgia Tech

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

$$y = mx + b$$



Our model is:

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

Classifier Result → $f(x, w)$

Weights → w

Input → x

Bias (scalar) → b

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

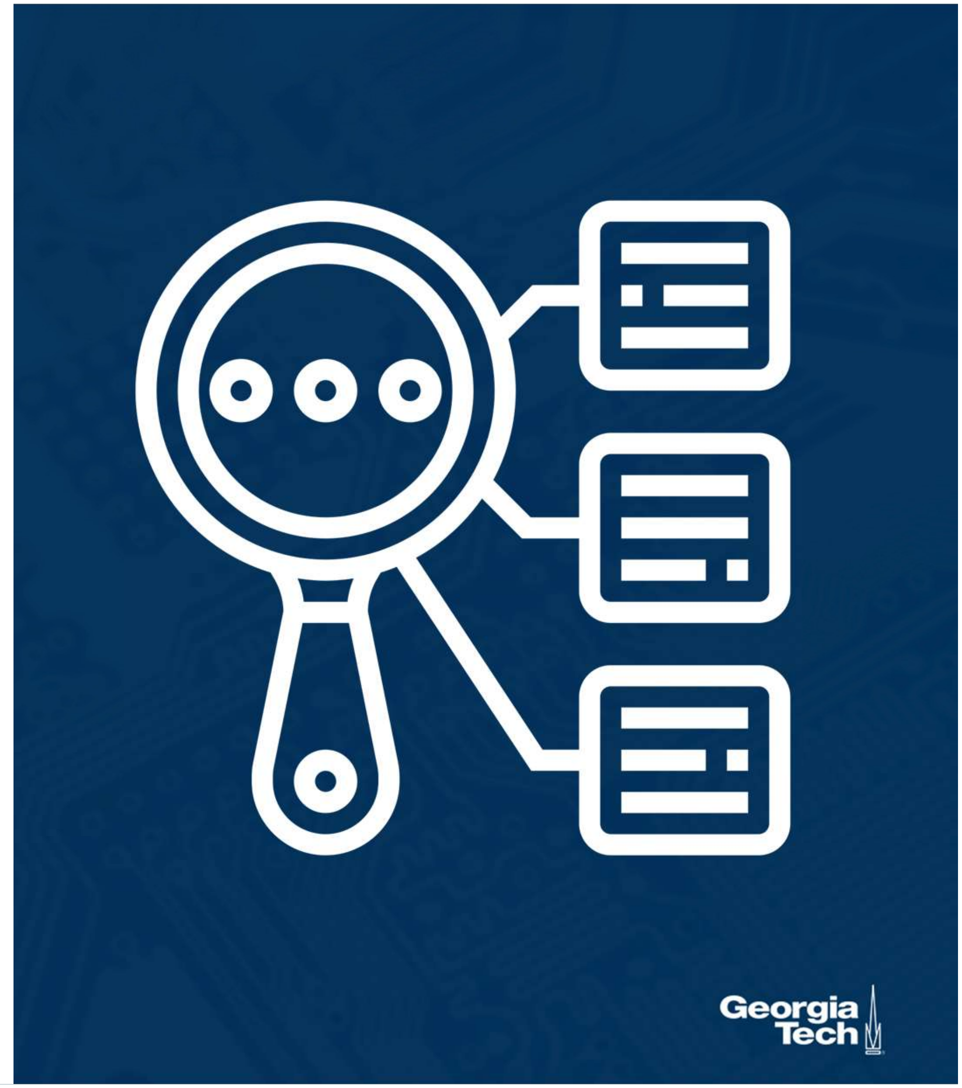
Image adapted from: https://en.wikipedia.org/wiki/Linear_equation#/media/File:Linear_Function_Graph.svg

Simple Function

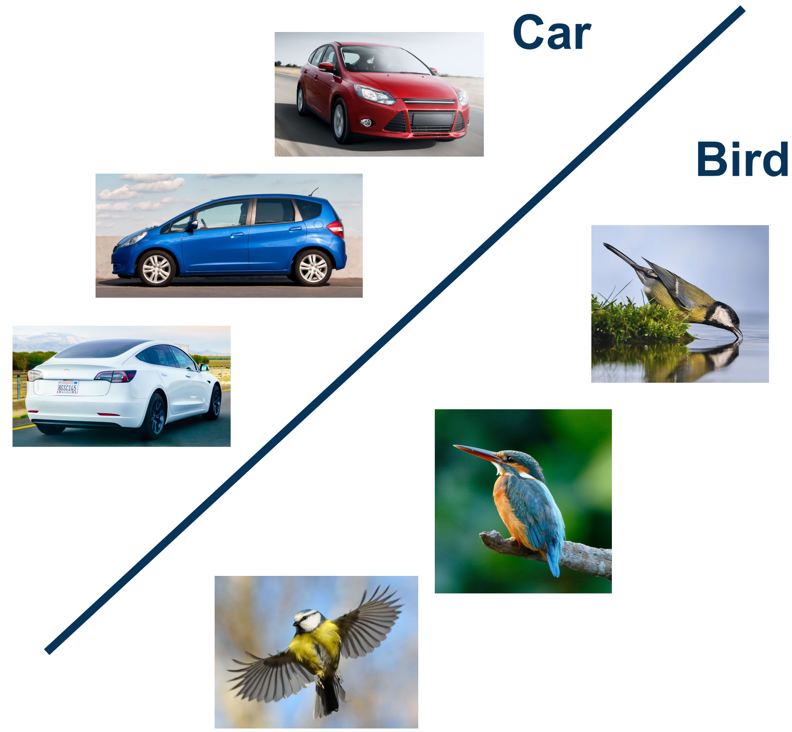
Linear Classification and Regression

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



- **Idea:** Separate classes via high-dimensional linear separators (hyper-planes)
- One of the simplest parametric models, **but surprisingly effective**
  - Very commonly used!
- Let’s look more closely at each element



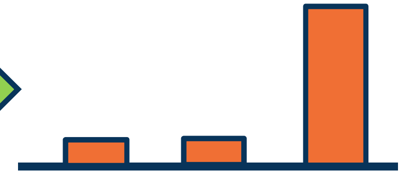
Input Dimensionality

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 $m = n \times n$

Georgia Tech

Model
$$\boldsymbol{f(x, W) = Wx + b}$$

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.)

Weights

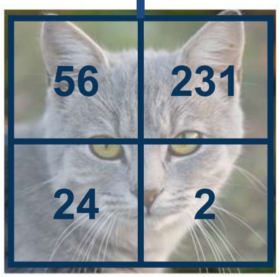
- We can move the bias term into the weight matrix, and a "1" at the end of the input
- Results in **one matrix-vector multiplication**!

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**)

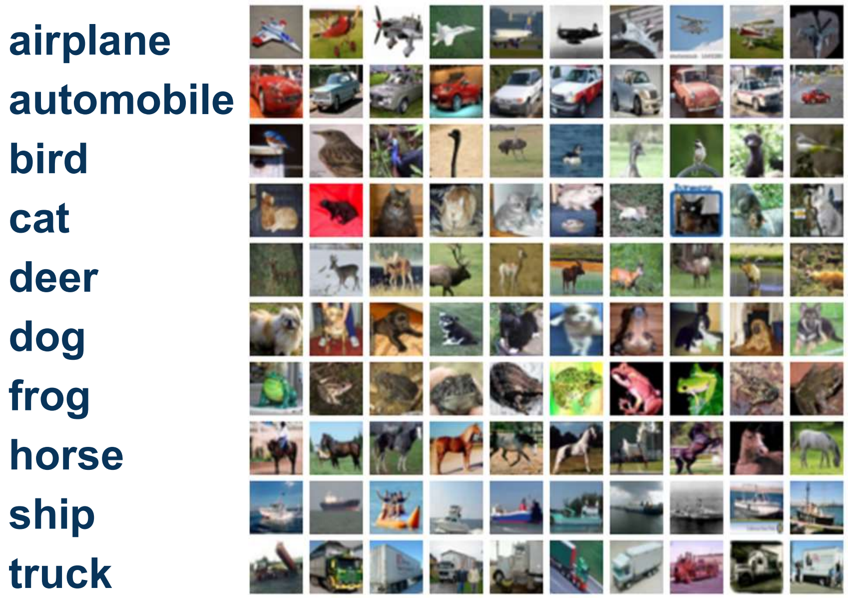
Stretch pixels into column



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

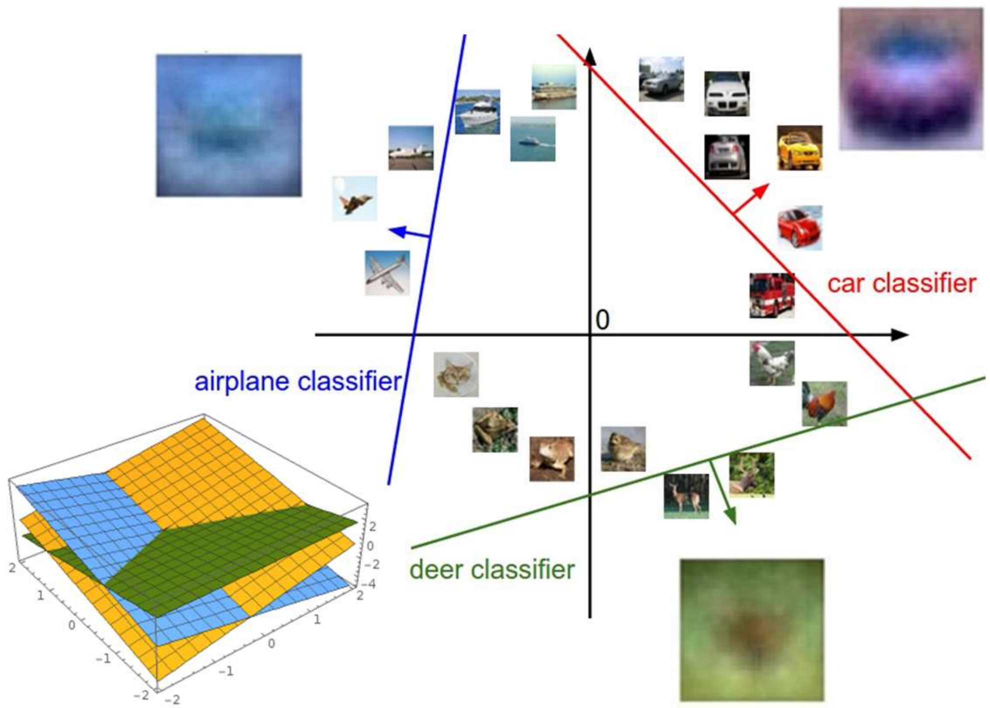


Visual Viewpoint

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

Interpreting a Linear Classifier



Plot created using Wolfram Cloud

Geometric Viewpoint

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

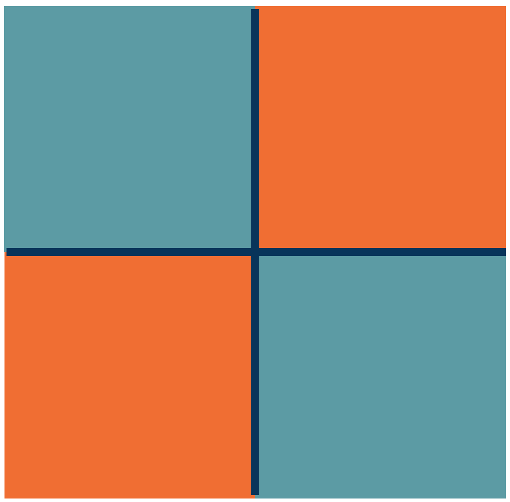
Array of **32x32x3** numbers
(3072 numbers total)

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

Hard Cases for a Linear Classifier

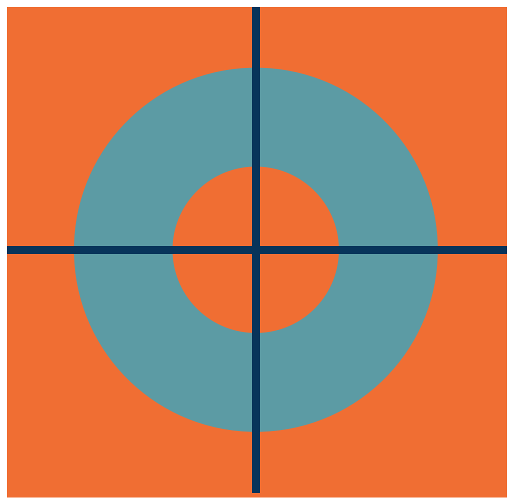
Class 1:
number of pixels > 0 odd

Class 2:
number of pixels > 0 even



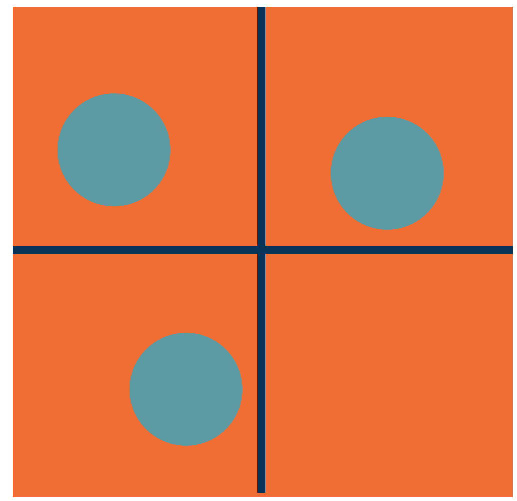
Class 1:
1 < = L2 norm < = 2

Class 2:
Everything else



Class 1:
Three modes

Class 2:
Everything else

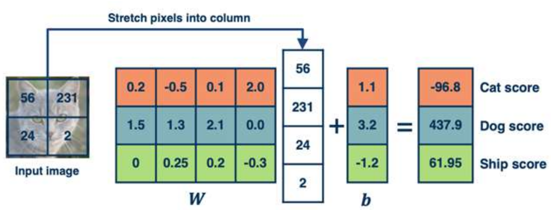


Adapted from slides by Fei-Fei Li, Justin Johnson, Serena Yeung, from CS 231n

Linear Classifier: Three Viewpoints

Algebraic Viewpoint

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

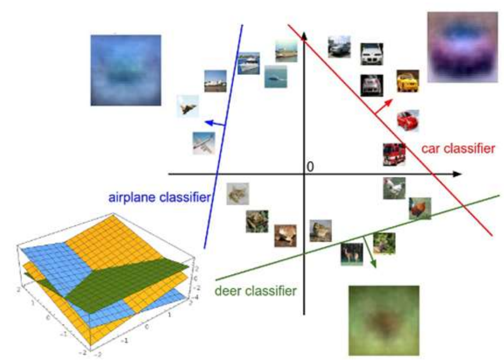


Visual Viewpoint

One template per class

Geometric Viewpoint

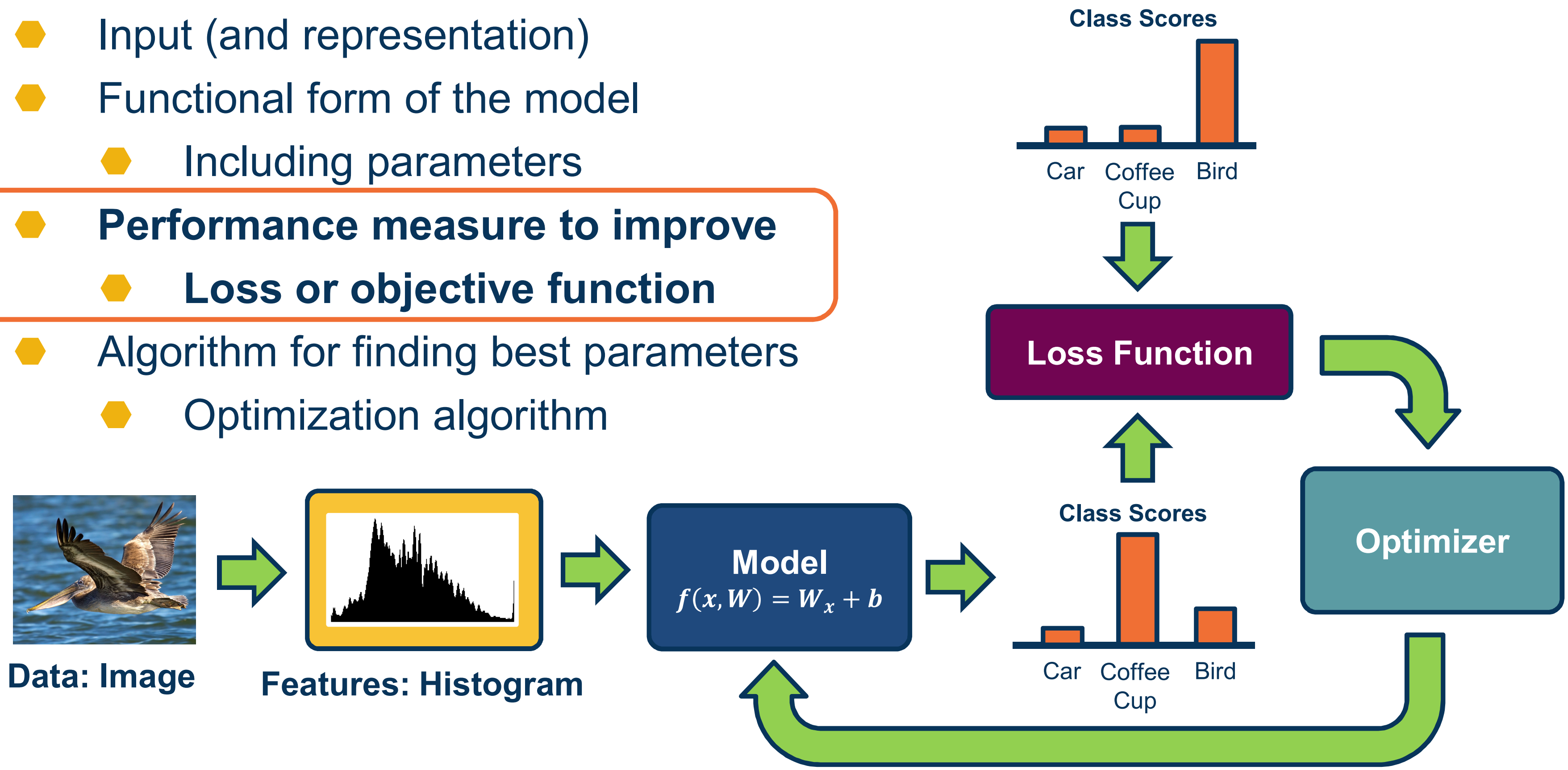
Hyperplanes cutting up space



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

Performance Measure for a Classifier

- 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

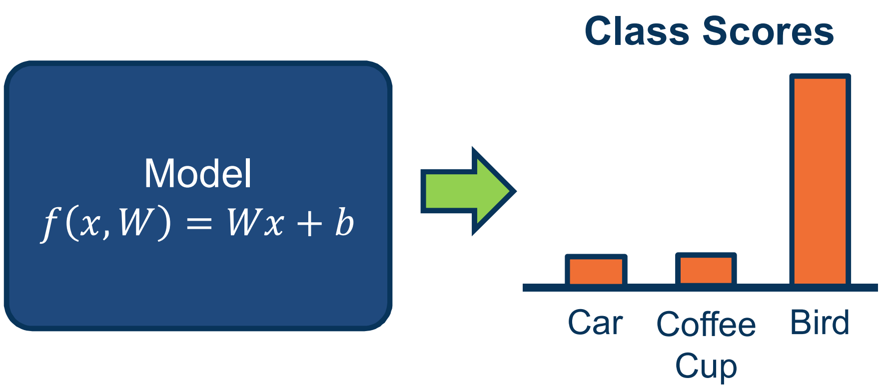


Components of a Parametric Model

- The output of a classifier can be considered a **score**
- For binary classifier, use rule:

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

  - Can be used for many classes by considering one class versus all the rest (one versus all)
- For multi-class classifier can take the maximum



Classification using Scores

Converting Scores to Probabilities

Several issues with scores:

- Not very interpretable (no bounded value)

We often want **probabilities**

- More interpretable
- Can relate to probabilistic view of machine learning

We use the **softmax** function to convert scores to probabilities

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

Scores

$$P(Y = k | X = x) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

Softmax Function

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:

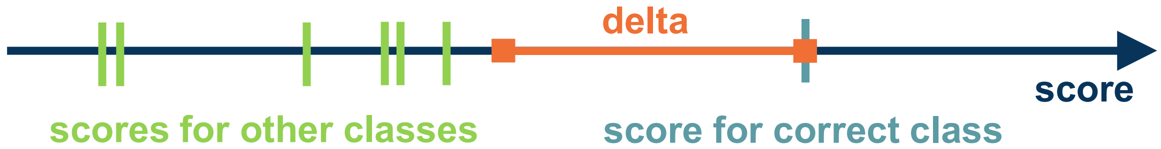
$$L = \frac{1}{N} \sum L_1(f(x_i, W), y_i)$$

Performance Measure

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,



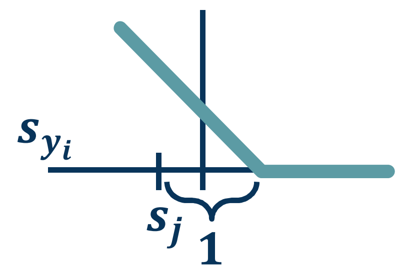
and using the shorthand for the scores vector: $s = f(x_i, W)$

the SVM loss has the form:

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

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

Example: "Hinge Loss"



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

Performance Measure for Scores

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, 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

SVM Loss Example

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

SVM Loss Example

Multiclass SVM loss:

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

Q: What happens to loss if car image scores change a bit?

No change for small values

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**

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

Multiclass SVM loss:

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

Q: What is min/max of loss value?

[0,inf]

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



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

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

SVM Loss Example

SVM Loss Example

Multiclass SVM loss:

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

Q: At initialization W is small so all s ≈ 0.
What is the loss?

C-1

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



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

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

Multiclass SVM loss:

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

Q: What if the sum was over all classes? (including j = y_i)

No difference (add constant 1)

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

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

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

Multiclass SVM loss:

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

Q: What if we used mean instead of sum?

No difference
Scaling by constant

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



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

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

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

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

E.g. Suppose that we found a W such that L = 0.

Q: Is this W unique?

No 2W also has L=0

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

Performance Measure for Probabilities

- If we use the softmax function to convert scores to probabilities, the right loss function to use is **cross-entropy**
- Can be derived by looking at the distance between two probability distributions (output of model and ground truth)
- Can also be derived from a maximum likelihood estimation perspective

$$L_i = -\log P(Y = y_i | X = x_i)$$

Maximize log-prob of correct class =
Maximize the log likelihood
= Minimize the negative log likelihood

- If we use the softmax function to convert scores to probabilities, the right loss function to use is **cross-entropy**

Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**

$$s = f(x_i; W)$$

$$P(Y = k | X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

Softmax Function

Probabilities must be >= 0

Probabilities must sum to 1

$$L_i = -\log P(Y = y_i | X = x_i)$$

	Unnormalized log-probabilities / logits	exp → Unnormalized probabilities	normalize → Probabilities
cat	**3.2**	**24.5**	**0.13**
car	5.1	164.0	0.87
frog	-1.7	0.18	0.00

→ $L_i = -\log(0.13)$

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

Cross-Entropy Loss Example

Georgia Tech

Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**

$$s = f(x_i; W)$$

Probabilities must be >= 0

$$P(Y = k | X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

Softmax Function

Probabilities must sum to 1

$$L_i = -\log P(Y = y_i | X = x_i)$$

$$L_i = -\log(0.13)$$

Q: What is the min/max of possible loss L_i?

Infimum is 0, max is unbounded (inf)

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

Softmax Classifier (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**

$$s = f(x_i; W)$$

Probabilities must be >= 0

$$P(Y = k | X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

Softmax Function

Probabilities must sum to 1

$$L_i = -\log P(Y = y_i | X = x_i)$$

$$L_i = -\log(0.13)$$

Q: At initialization all s will be approximately equal; what is the loss?

Log(C), e.g. log(10) ≈ 2

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