EECS 545: Machine Learning
Lecture 1. Introduction

Honglak Lee

1/10/2020

Outline

- Administrative
- What is machine learning?

Teaching staff

- Instructor: Honglak Lee
  - Email: honglak@eecs.umich.edu
  - Office: CSE 3773
- Graduate Student Instructor:
  - Sungryull Sohn (srsohn@umich.edu)
  - Yijie Guo (guoyijie@umich.edu)
  - GS Oh (gsoh@umich.edu)
- For office hours, please see the future announcement and calendar.
- For all questions, please use Piazza:
  - https://piazza.com/class/jqn72gu5i3v4l4

About this course

- Graduate-level introduction of machine learning
- Foundations of machine learning
  - Mathematical derivation, Implementation of the algorithms, applications
- Topics
  - supervised learning
  - unsupervised learning
  - reinforcement learning
- Other topics
  - deep learning, learning theory, probabilistic models, sparsity and feature selection, Bayesian techniques, ensemble methods

About this course

- Cover practical applications of machine learning
  - computer vision, data mining, speech recognition, text processing, robot perception and control, etc.
- Our goal is to help you to
  - Understand fundamentals of machine learning
  - Learn technical details of ML algorithms
  - Learn how to implement some important algorithms
  - Use machine learning algorithms for your research and applications.

Text books

There will be no official textbook for the course. However, the following materials will be helpful:

- Chris Bishop, “Pattern Recognition and Machine Learning”. Springer, 2007.
- Kevin Murphy, “Machine Learning: A Probabilistic Perspective”, 2012.
- David Barber, “Bayesian Reasoning and Machine Learning”, 2017. (available online) http://web4.cs.ucl.ac.uk/staff/D.Barber/pmwiki/pmwiki.php?n=Brml.Online)
- Ian Goodfellow and Yoshua Bengio and Aaron Courville, “Deep Learning”, (available online) http://www.deeplearningbook.org/
- Hastie, Tibshrani, Fiedman, “Elements of Statistical Learning”. Springer, 2010. (available online) http://www-stat.stanford.edu/~tibs/ElemStatLearn/
- Sutton and Barto, "Reinforcement Learning: An Introduction," MIT Press, 2018 (available online) http://incompleteideas.net/book/the-book-2nd.html
- (optional) Boyd and Vandenberghe, "Convex Optimization," Cambridge University Press, 2004. (available online) http://www.stanford.edu/~boyd/cvxbook/
- (optional) Mackay, “Information Theory, Inference, and Learning Algorithms”. Cambridge University Press. 2003. (available online) http://www.inference.phy.cam.ac.uk/itprnn/book.pdf

Prerequisites

- Undergrad linear algebra (e.g., MATH 217, MATH 417)
- Multivariate calculus
- Undergrad probability and statistics (e.g., EECS 401)
- Programming skills (equivalent to EECS 280, EECS 281, and experience in python)
  - Nontrivial level of programming is required.
- EECS 492: Introduction to AI
  - (advisory: you can still take this course without completing 492)

- NOTE: If you **have not** taken **at least two** of linear algebra, multivariate calculus, and probability courses, it is **strongly recommended** that you finish them first before taking this course.

Grading policy

- Homework: 30%
- Midterm: 30% (tentative date: late March/early April)
  - Honor code
- Project: 40%
  - progress report (10%)
  - final report (30%)

- Note: There will be no final exam.

- Extra credits: Up to 2% may be awarded for participation (in class and piazza).

Language of Choice: **Python**

- **Python** is a great language overall for machine learning, with modern libraries and excellent online tutorials for various tasks
- We will utilize **Python** throughout the course, and we encourage you to do the same.
  - We will be using popular libraries, such as **numpy** and **pytorch**.
- There will be a tutorial session (date: TBD)

Homework

- There will be 5 or 6 problem sets.
- Goal: strengthen the understanding of the fundamental concepts, mathematical formulations, algorithms, detailed implementations, and the applications.
- The problem sets will also include programming assignments to implement algorithms covered in the class.
- Homework #1 will be out next Tue (1/14) – due 1/28, 5pm via Gradescope (linked to Canvas).

Late days

- 3 maximum late days per assignment
  - No homework will be accepted 3 days after the due date.
- Total 8 late days allowed.
- After using up all late days, your assignment will be penalized by 20% from your scores.

Study group

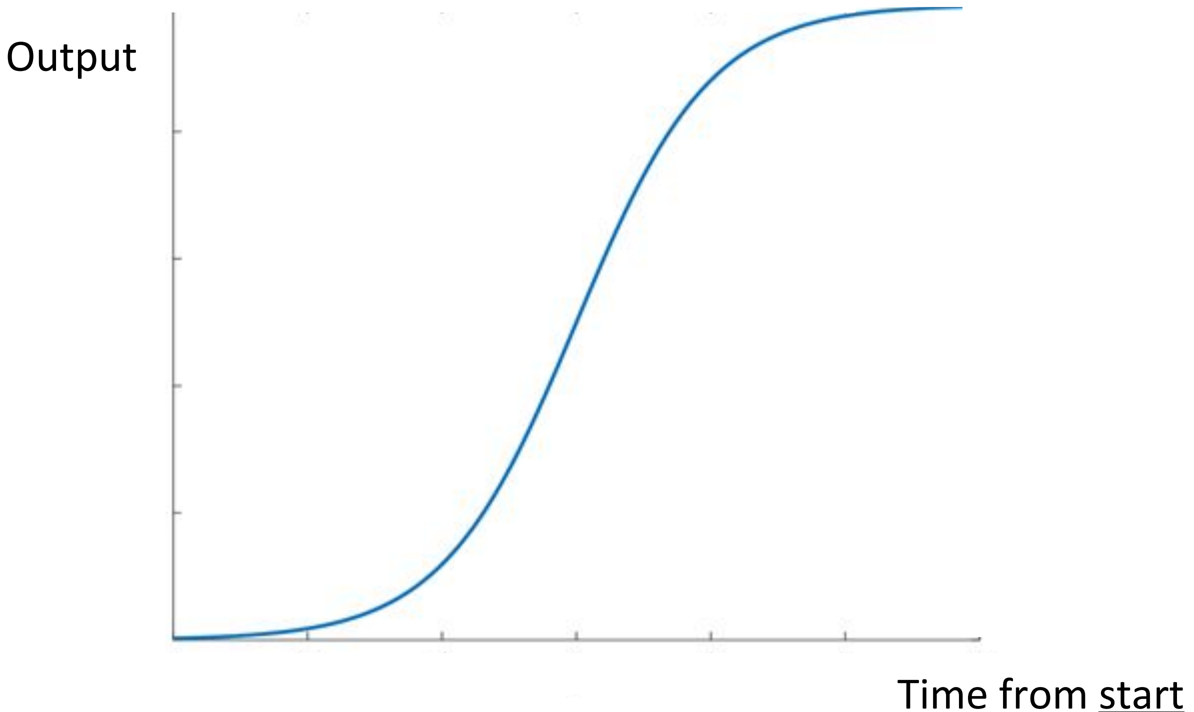
- Form your study group early on!
  - Up to <u>four</u> people are allowed.
- For homework, you may discuss between the study group members, but you should write your own solution <u>independently</u>.
- In the homework submissions, you must put:
  - the names of other people you collaborated.
  - submission time.
- Please start on homework early. (Warning: cramming does not work!)

Course Project

- Scope
  - develop new algorithms and theory in machine learning
  - apply existing algorithms to new problems
  - apply to your own problems of interest
- Milestones (tentative)
  - project proposals (tentative due date: 2/4)
  - progress reports (tentative due date: 3/10)
  - poster presentations (tentative date: 4/24)
  - Final project report (tentative due date: 4/27)
- Requirement
  - Write a 8-page paper
  - Submit the final code
  - Give a poster presentation
- Evaluation is based on:
  - novelty, technical quality, significance, and presentation quality of the project.

Course Project

- 4 or 5 people can form a project group.
- Talk to instructor if you want to get suggestions about project topics.
- Start early!!! (form your group and start working)



Other Information

- Review sessions
  - Will hold review sessions on background materials (linear algebra, probability, Python.)

- Beginning-of-course survey: https://bit.ly/39XZ5ZY

Any questions?

Outline

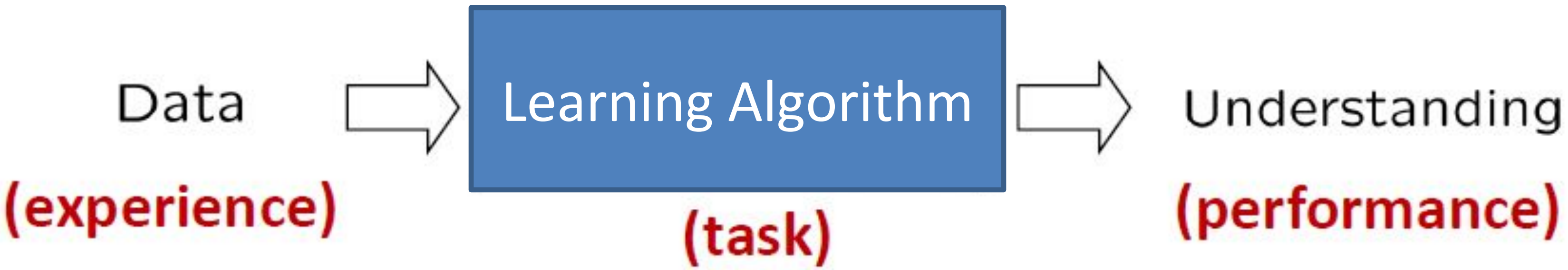
- Administrative
- What is machine learning?

Definition of Machine Learning

- Formal definition (Mitchell 1997): A computer program **A** 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**.

Informal definition

- Algorithms that improve their prediction performance at some task with experience (or data).



Example: Spam email filtering

"A computer program is said to *learn* from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E."

- Suppose your email program watches which emails you do or do not mark as spam, and based on that learns how to better filter spam.

- Task:
  – Classifying emails as spam or not spam.

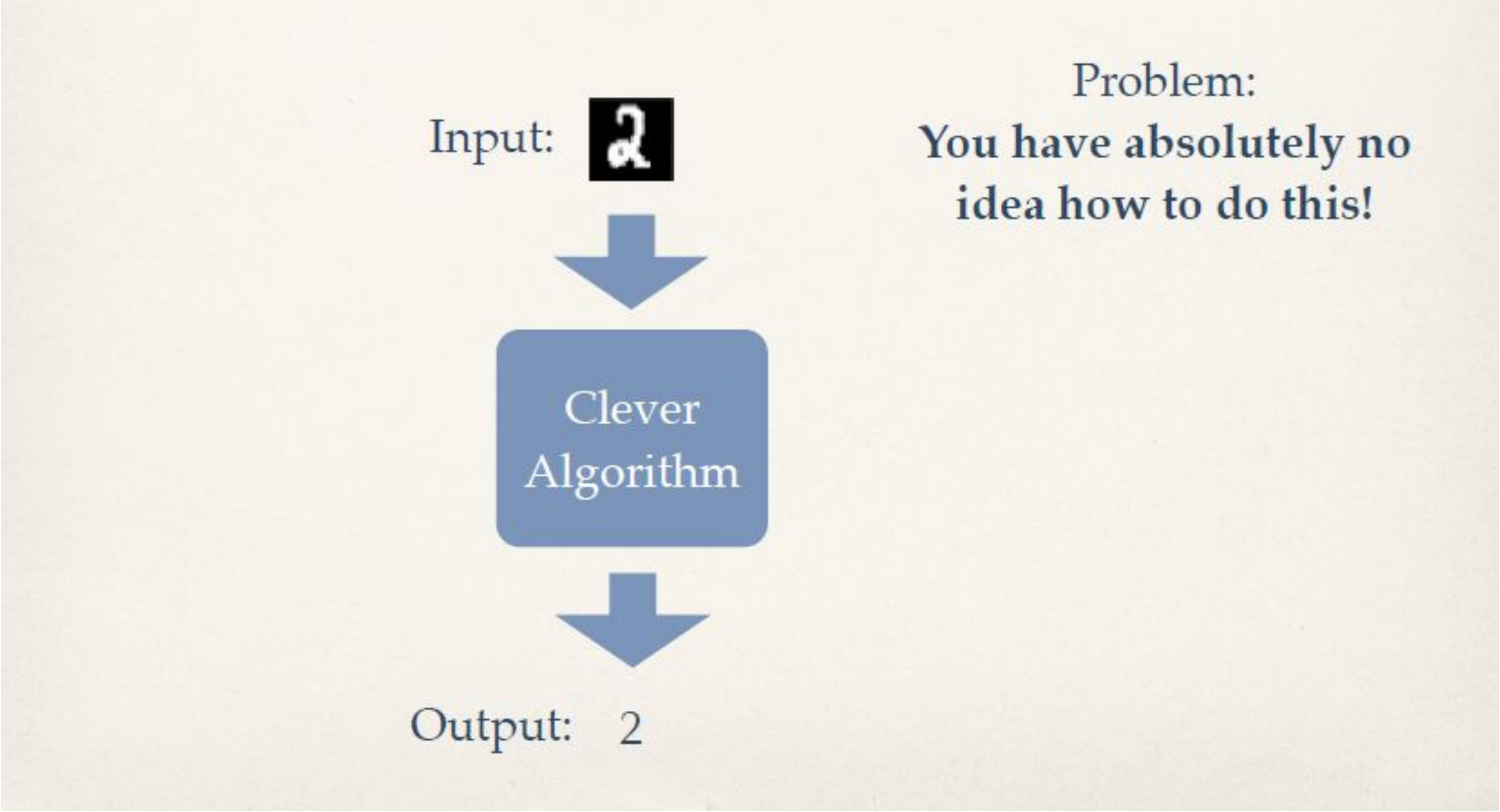
- Experience:
  – Watching you label emails as spam or not spam.

- Performance measure
  – The number (or fraction) of emails (disjoint from the training emails) correctly classified as spam/not spam.

Slide credit: Andrew Ng

Example

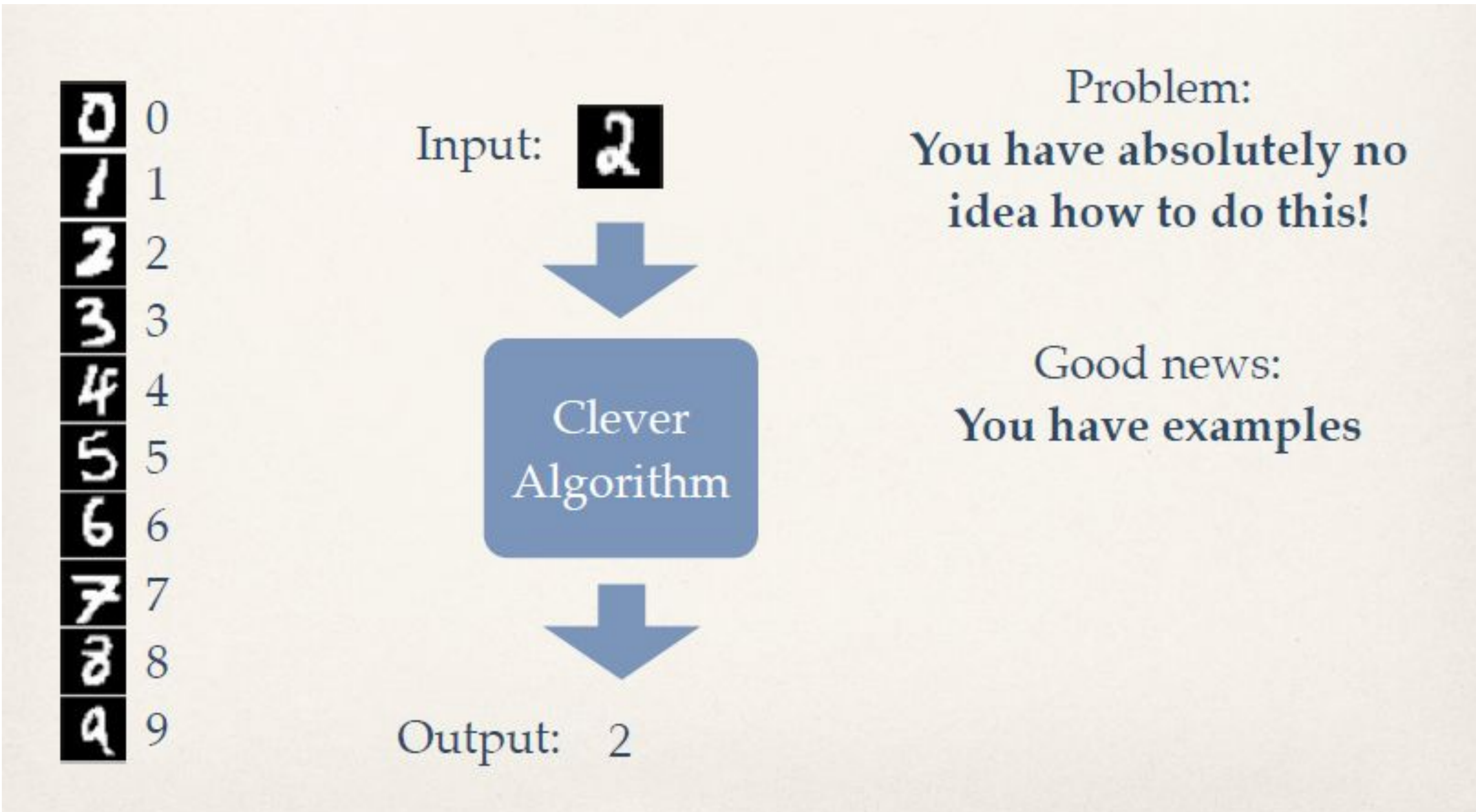
- Problem: Given an image of a handwritten digit, what digit is it?



Slide credit: Kilian Weinberger

Example

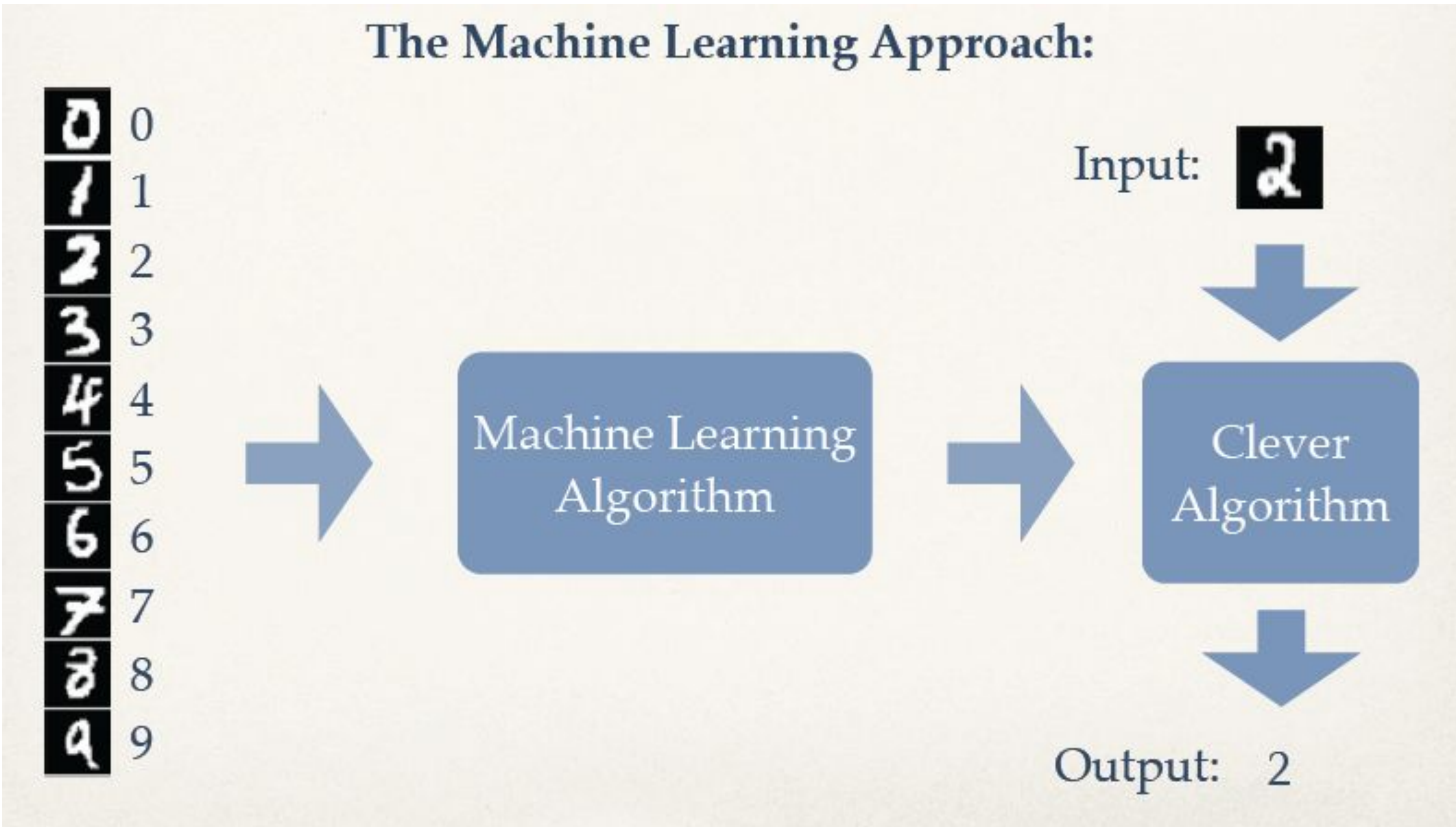
- Problem: Given an image of a handwritten digit, what digit is it?



Slide credit: Kilian Weinberger

Example

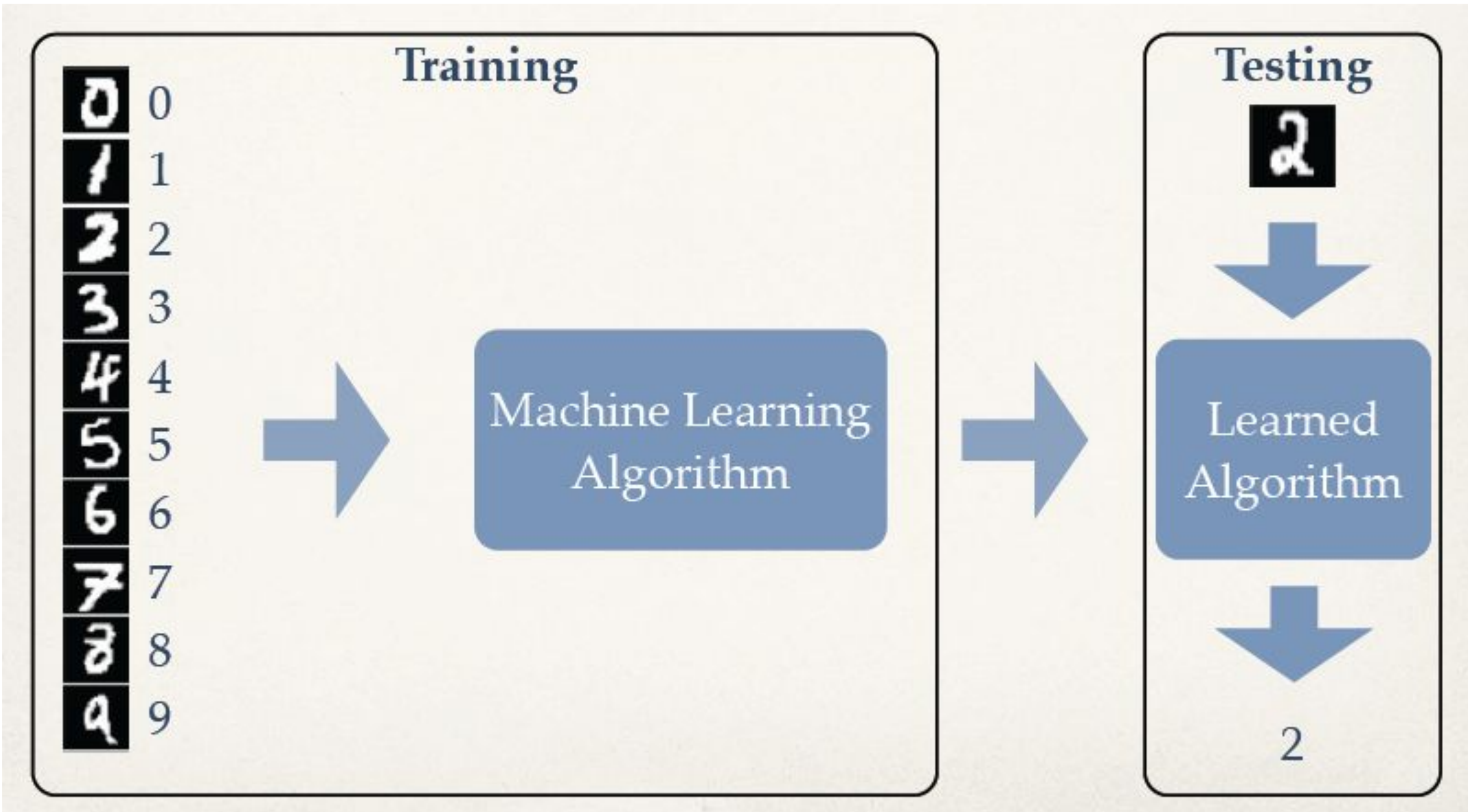
- Problem: Given an image of a handwritten digit, what digit is it?



Slide credit: Kilian Weinberger

Example

- Problem: Given an image of a handwritten digit, what digit is it?



Slide credit: Kilian Weinberger

Machine Learning Tasks

- Supervised Learning
  - Classification
  - Regression
- Unsupervised Learning
  - Clustering
  - Density estimation
  - Embedding / Dimensionality reduction
- Reinforcement Learning
  - Learning to act (e.g., robot control, decision making, etc.)

Supervised Learning

Given a dataset $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, where

- $x_i \in \mathcal{X}$: input (feature)
- $y_i \in \mathcal{Y}$: output (label)

a black box ML algorithm produces a prediction function $h : \mathcal{X} \to \mathcal{Y}$, such that $h(x)$ can predict the y values for all x (including training data $x_i \in D$ and unseen test data x^*).

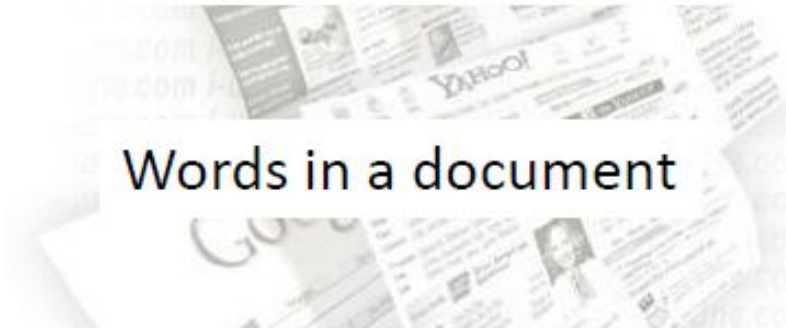
Supervised Learning

- Labels could be discrete or continuous
  - Discrete labels: **classification**
  - Continuous labels: **regression**

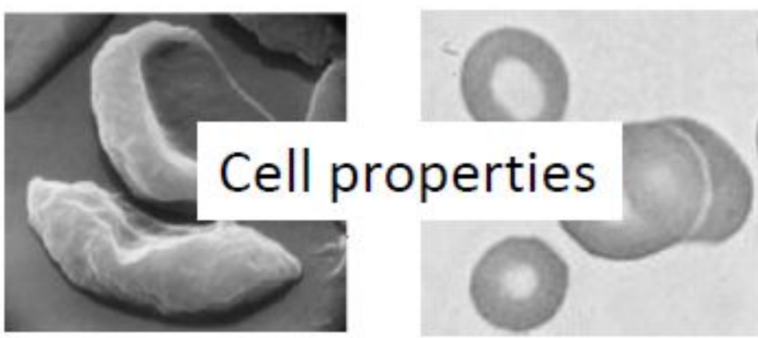
Supervised Learning - Classification

Feature Space $\mathcal{X}$

Label Space $\mathcal{Y}$



"Sports"
"News"
"Science"
...

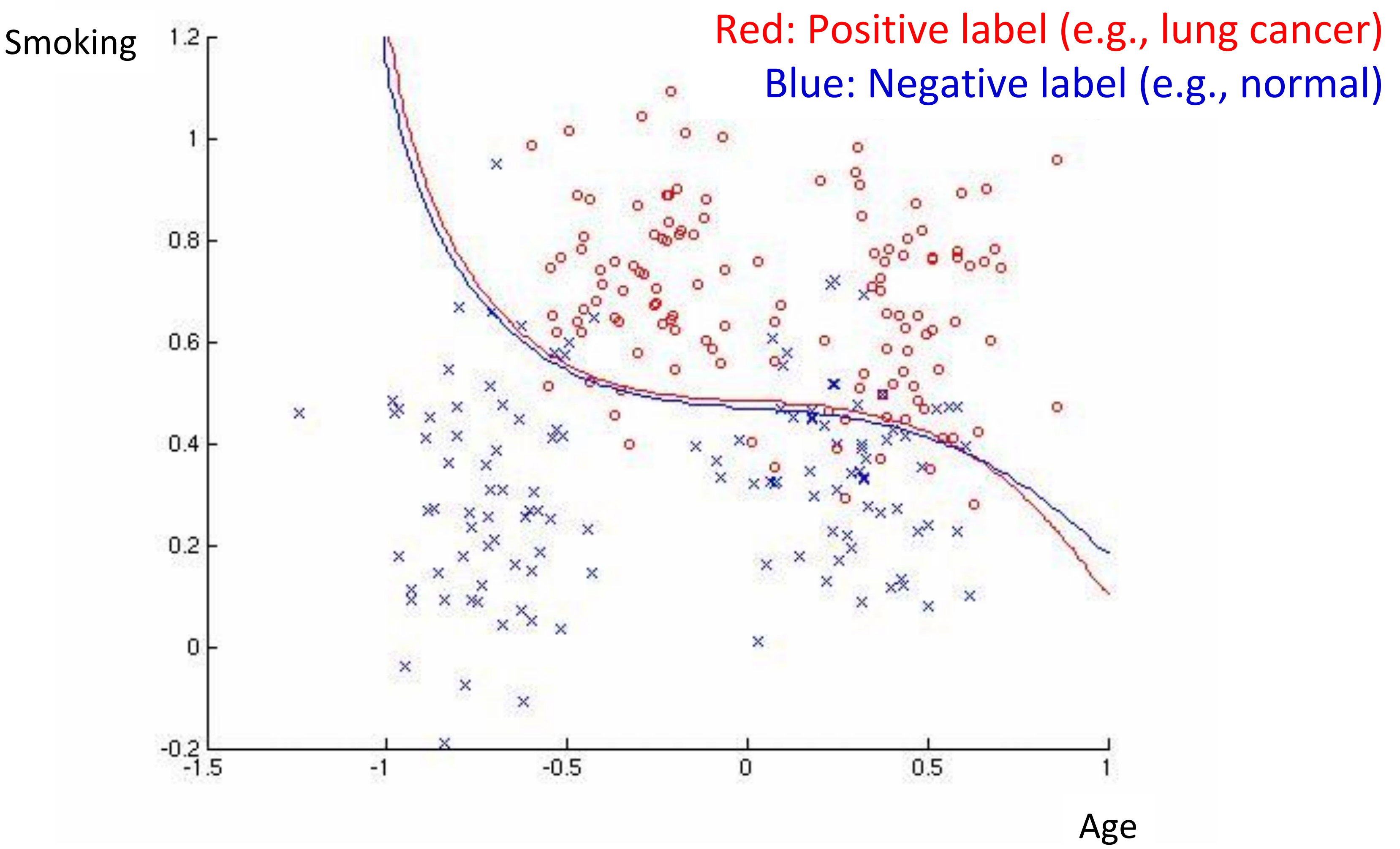


"Anemic cell"
"Healthy cell"

Discrete Labels

Slide credit: Aarti Singh

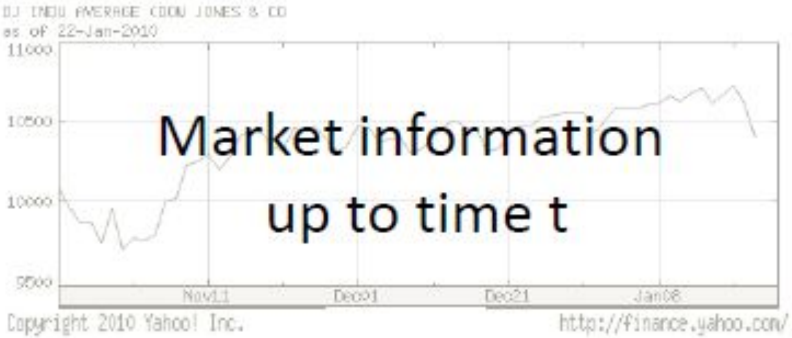
Supervised Learning - Classification



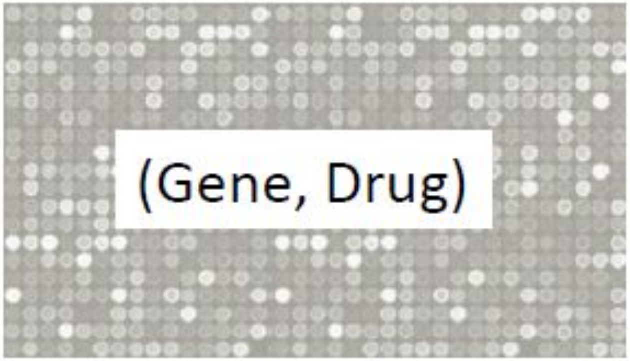
“Learning decision boundaries”

Supervised Learning - Regression

Feature Space $\mathcal{X}$ | **Label** Space $\mathcal{Y}$



Share Price
"$ 24.50"

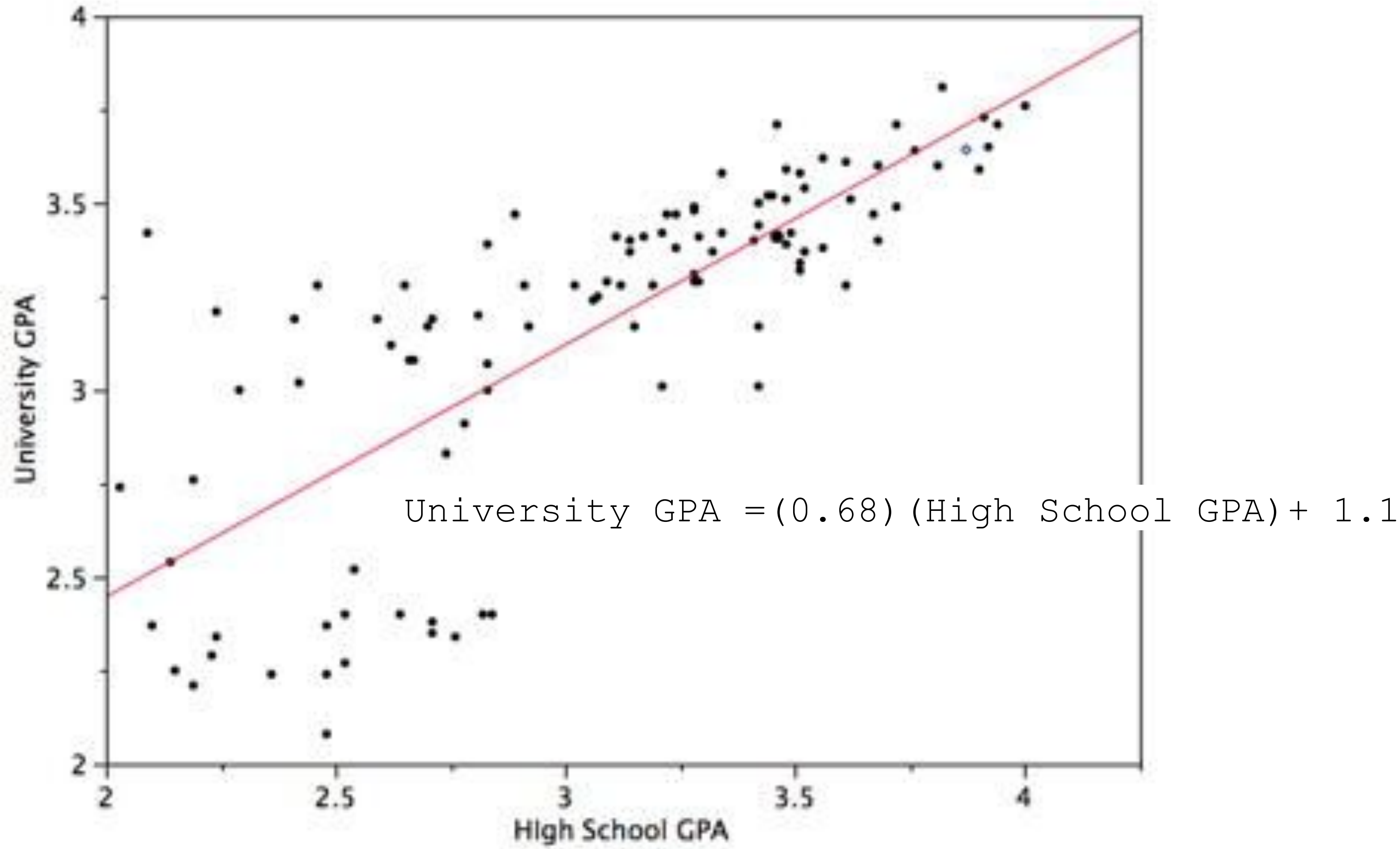


Expression level
"0.01"

Continuous Labels

Slide credit: Aarti Singh

Supervised Learning - Regression



"Learning regression function f(X)"

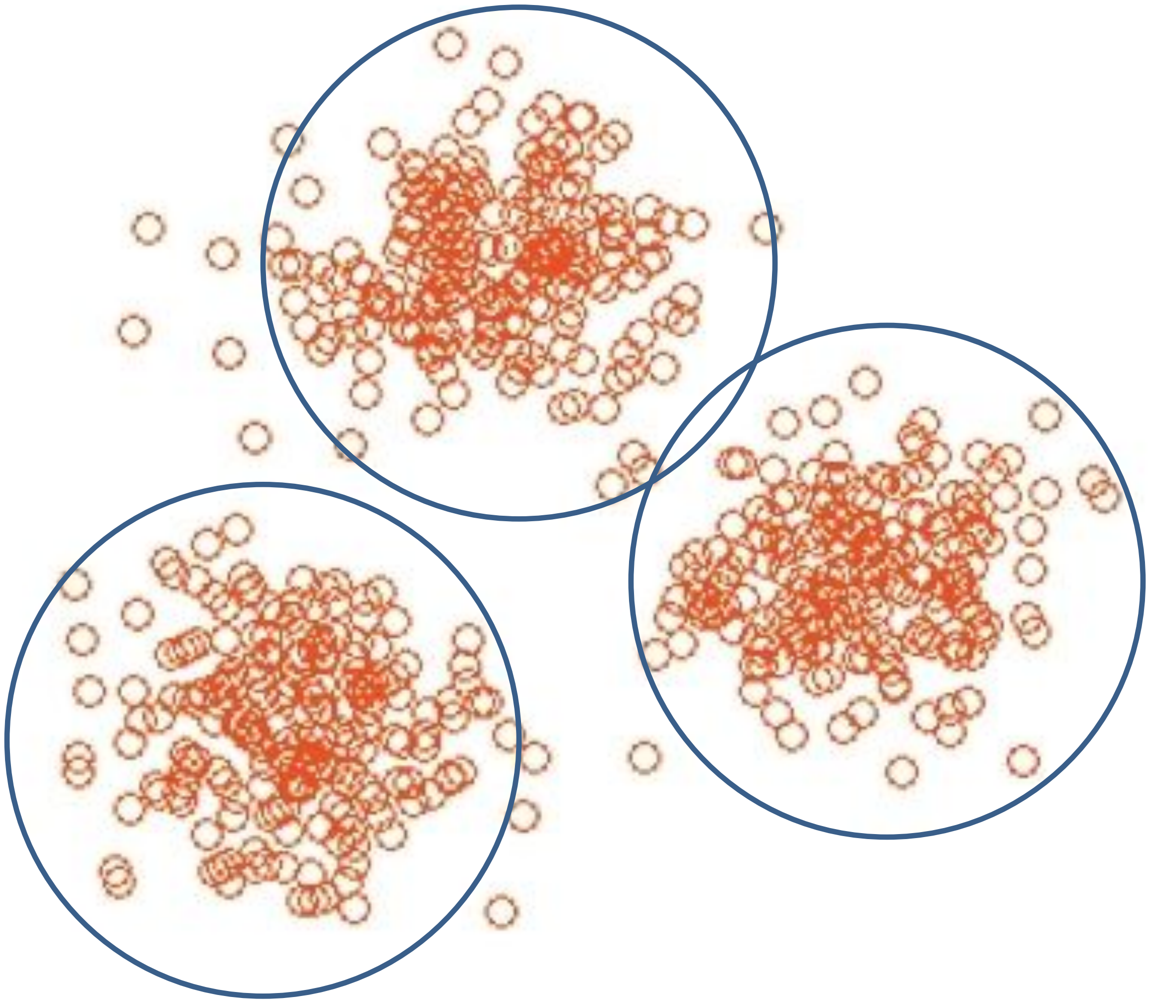
Slide credit: David M. Lane

Unsupervised Learning

- Goal:
  - Given data X <u>without any labels</u>
  - Learn the **structures** or **distribution** of the data
    - Clustering
    - Probability distribution (density)
    - Generating data
    - Embedding & neighborhood relations
- “Learning without teacher (supervision)”

Unsupervised Learning – Clustering

- "Grouping into similar examples"



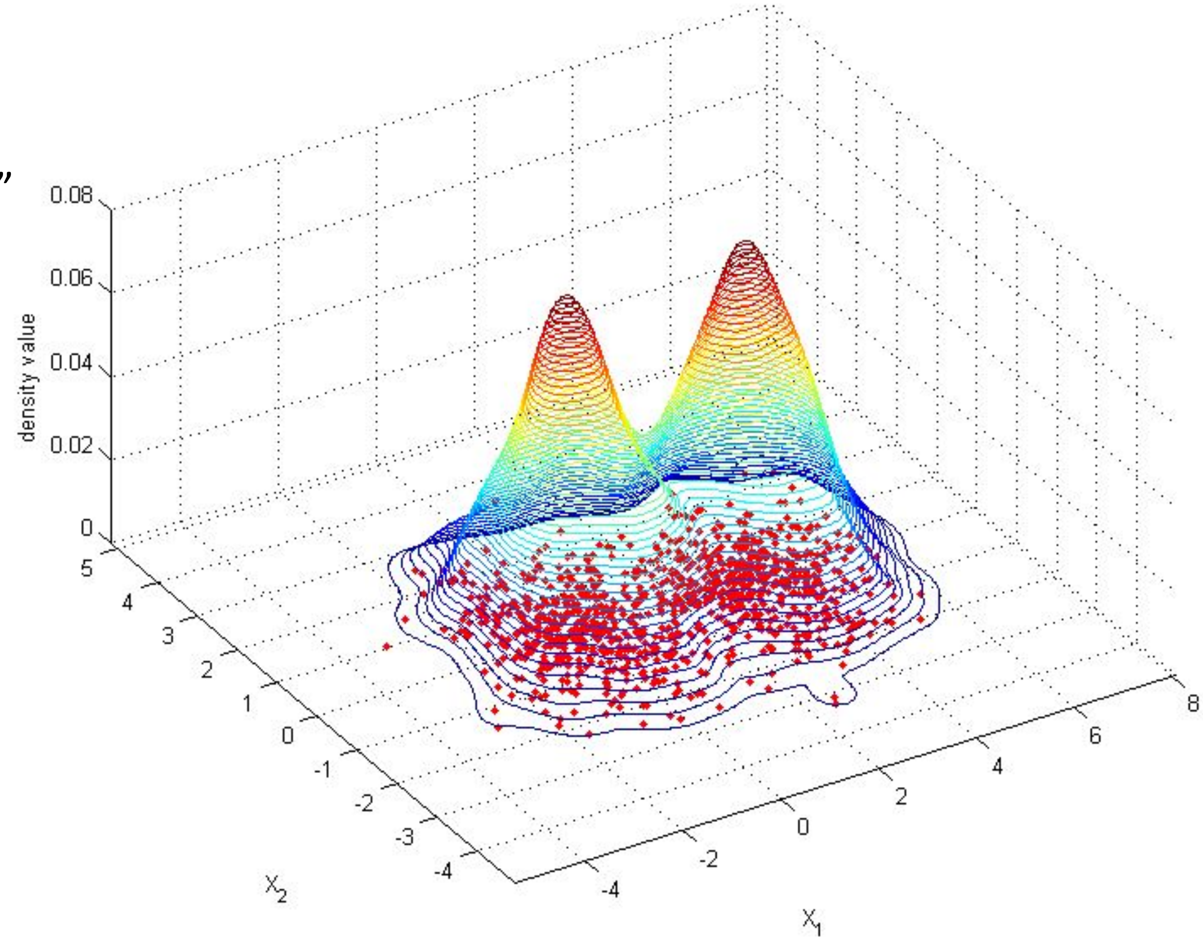
Unsupervised Learning – Clustering

Group similar things e.g. images [Goldberger et al.]

Slide credit: Aarti Singh

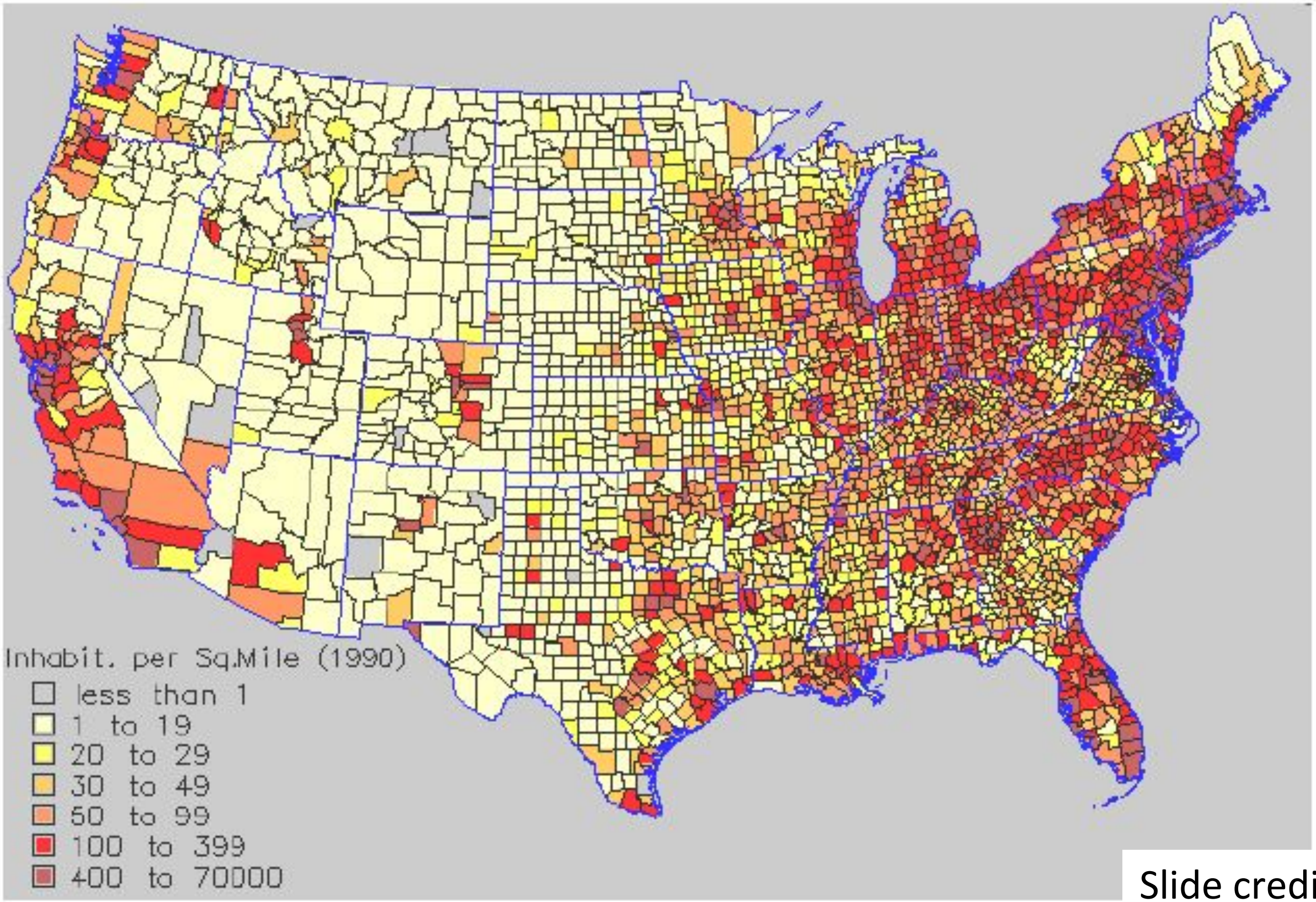
Unsupervised Learning – Density estimation

P(X1, X2)
“Probability”



Unsupervised Learning – Density estimation

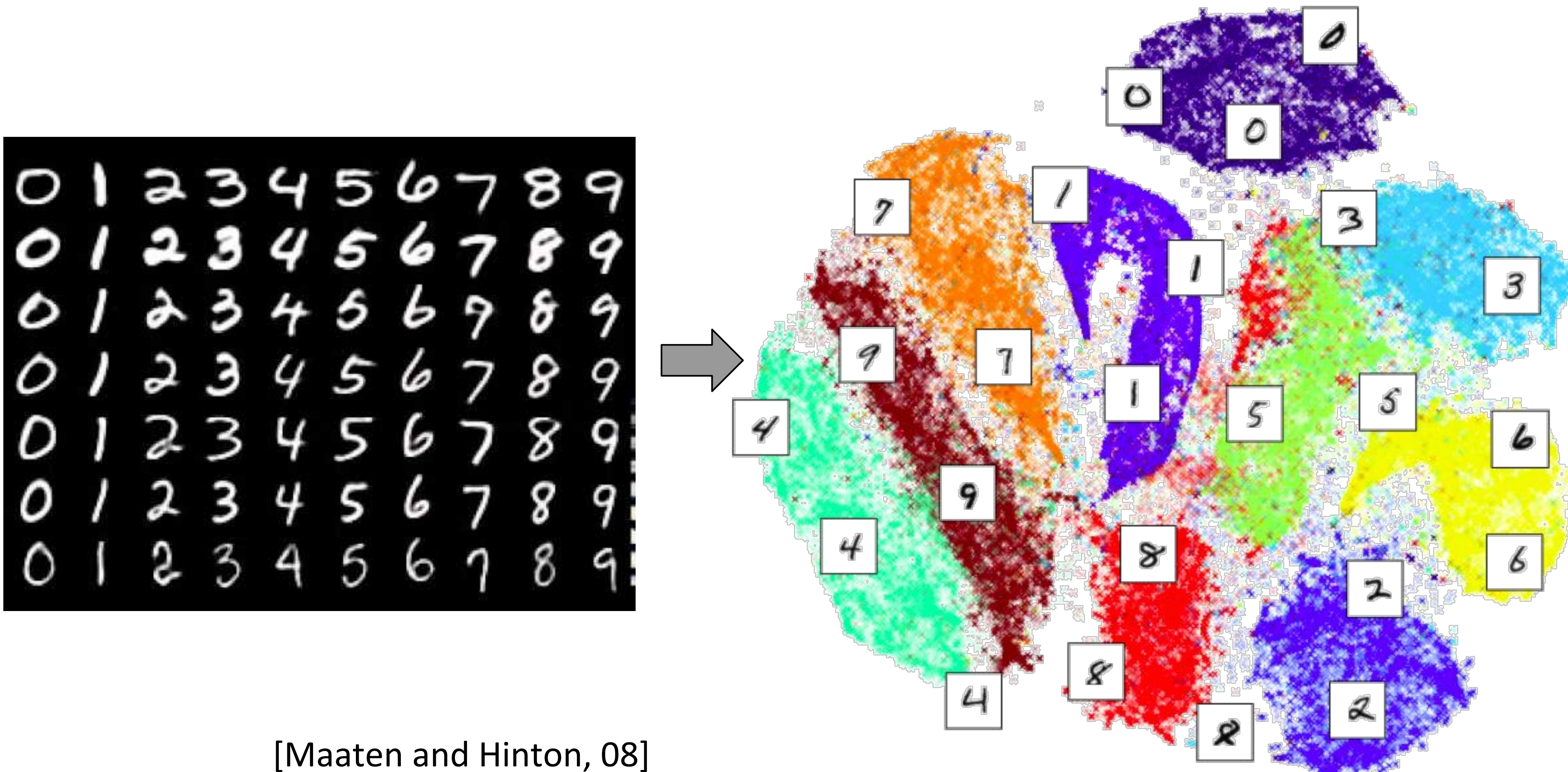
Population density



Slide credit: Aarti Singh

Unsupervised Learning-Embedding and Dimensionality reduction

- E.g., Reducing handwritten digits (784 dim) into low dimensional coordinates



[Maaten and Hinton, 08]

Reinforcement Learning

- Setting
  - Given sequence of states X and "rewards" (e.g., delayed labels)
  - Agent has to take actions A for each time step
- Goal:
  - How to "learn to act" or "make decisions" to maximize the sum of future rewards
- Example: Robot navigation task
  - Input: Dynamical environment + sensor input
  - Action: control signals
  - Rewards: time to reach goal without colliding with obstacles

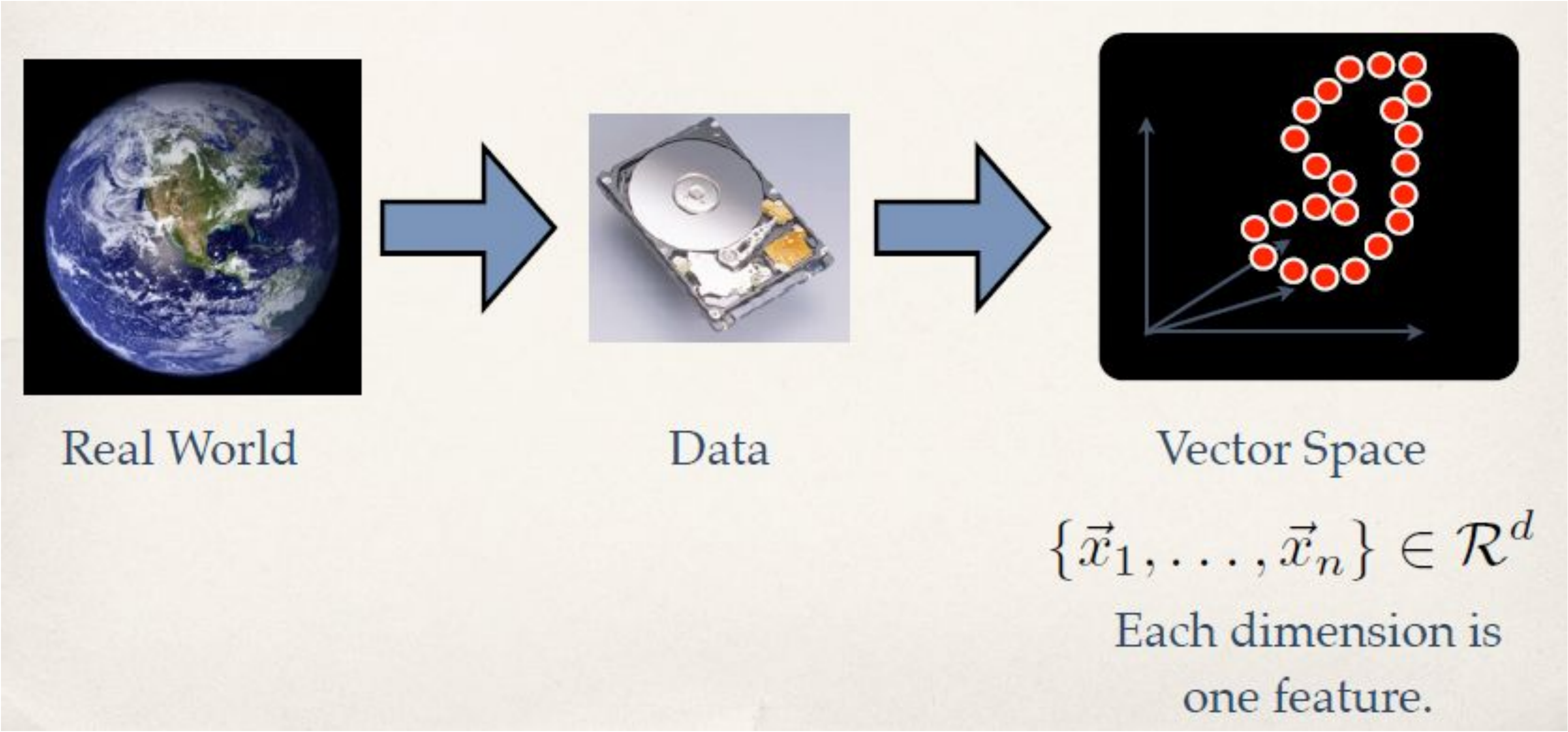
Reinforcement Learning – learning to control

- Example: Robot walking
  - States: sensor inputs, joint angles
  - Action: servo commands for joints
  - Rewards:
    - 1 for reaching the goal
    - -1 for falling down
    - 0 otherwise
- Goal: How can we provide control inputs to maximize the expected future rewards?

Feature representations

Feature Extraction

- Represent data in terms of vectors.
  - Features are **statistics** or **attributes** that describe the data.



Slide credit: Kilian Weinberger

Examples of features: Housing data

- **Given statistics about houses in a local area, predict median value of homes.**
  - #ROOM: average number of rooms per dwelling
  - AREA: average area of house in square foot
  - AGE: proportion of owner-occupied units built prior to 1940
  - CRIME: per capita crime rate by town
  - RESZONE: proportion of residential land zoned for lots over 25,000 sq.ft.
  - INDUS: proportion of non-retail business acres per town
  - NOX: nitric oxides concentration (parts per 10 million)
  - …..
- **Label: Median value of owner-occupied homes**

Examples of features: Recognizing handwritten-digits

- Input: 28x28 pixel digit images
- Output: class labels $\in \{0, \ldots, 9\}$
- The following basic features can be used:
  - Pixel values (784 dimensional vectors)
  - Aspect ratio of the tight bounding boxes
  - Existence of long vertical strokes
  - Existence of long horizontal strokes
  - ....

Learning pipeline

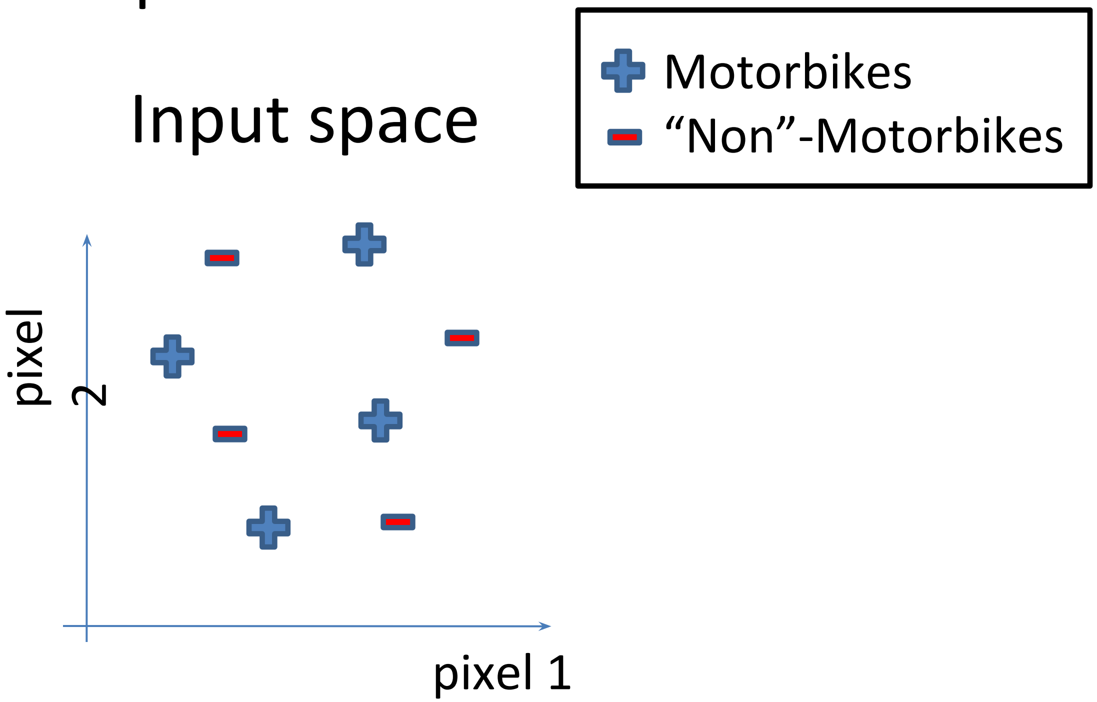
pixel 1

pixel 2

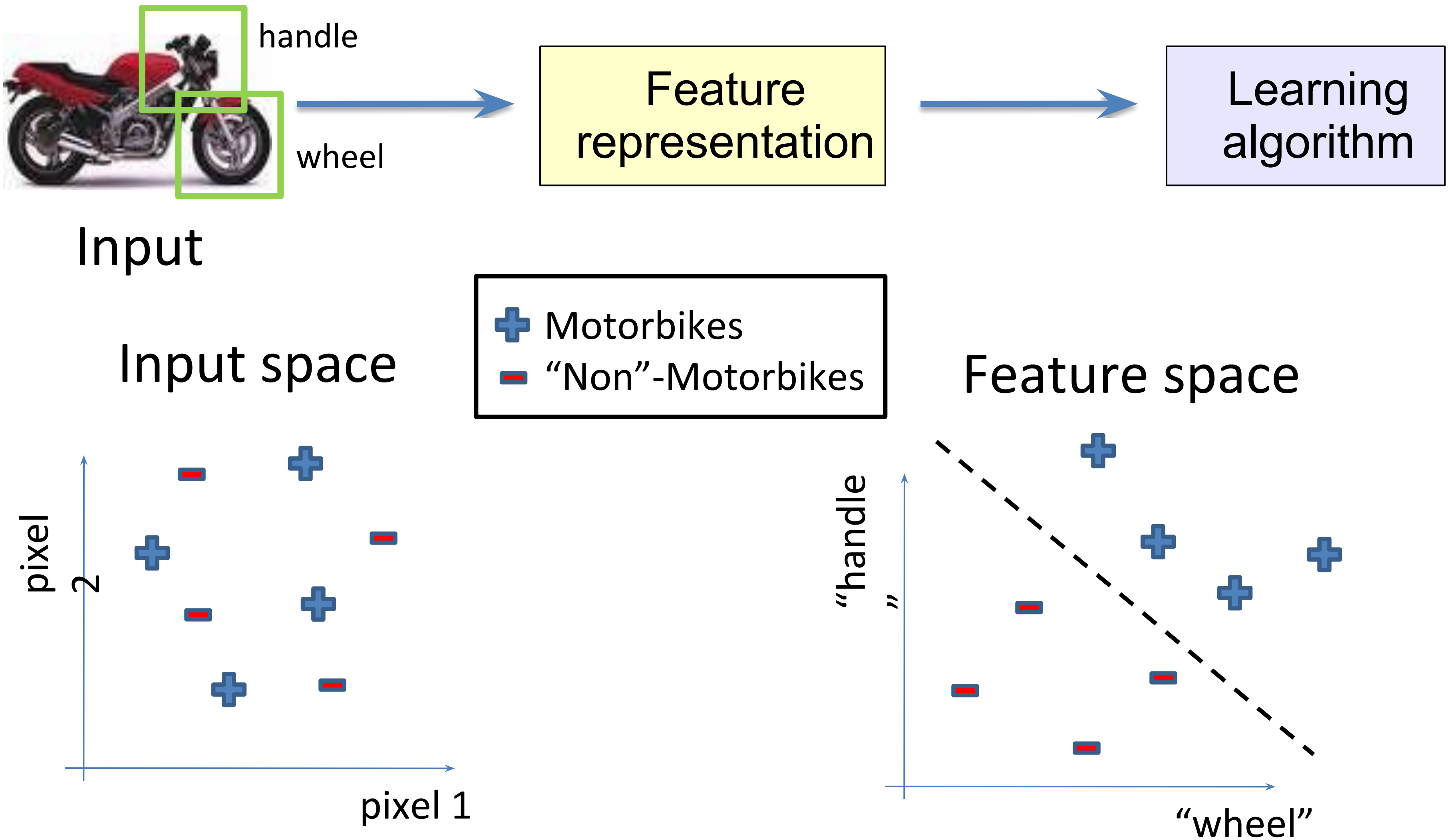
Learning algorithm

Input

Input space



Learning pipeline



(Traditional) Computer Perception Pipeline

Input data → Feature representation → Learning algorithm

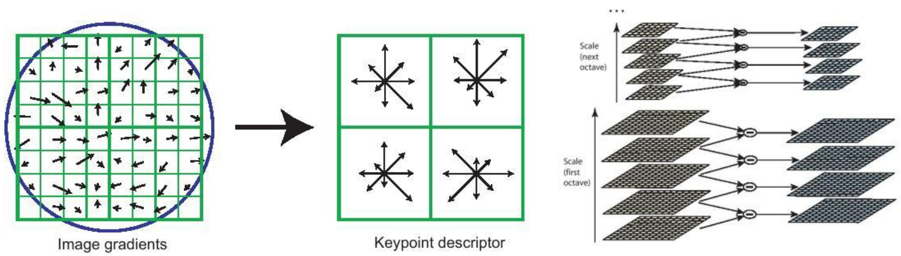
Object detection

Imag
e

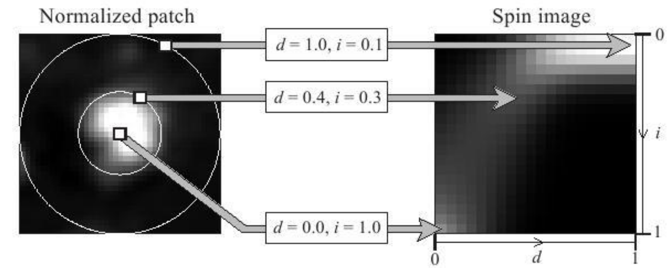
Low-level
vision
features

Recognitio
n

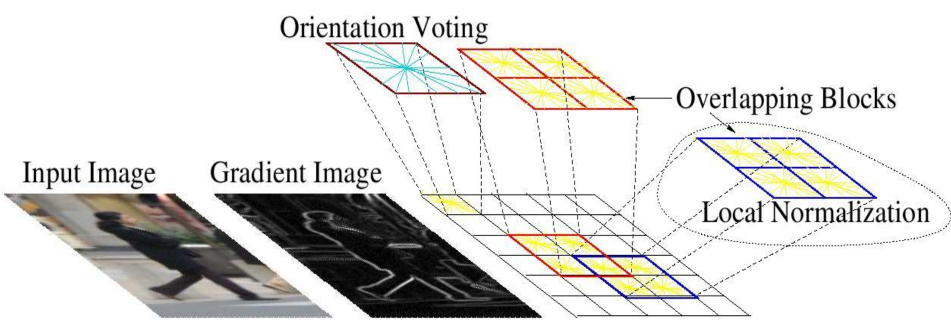
(Traditional) Computer vision features



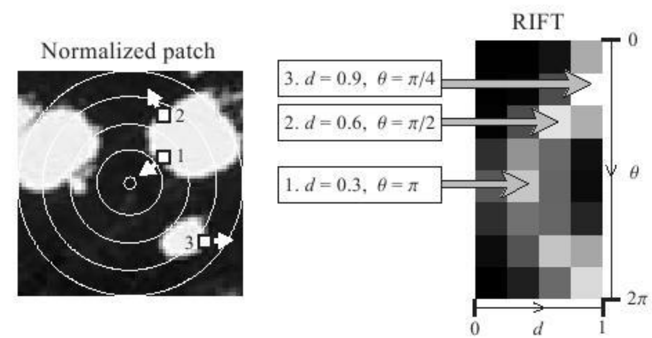
SIFT



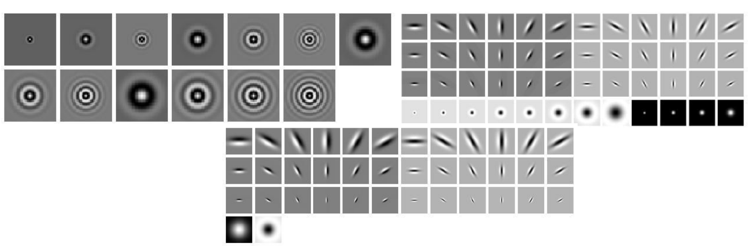
Spin image



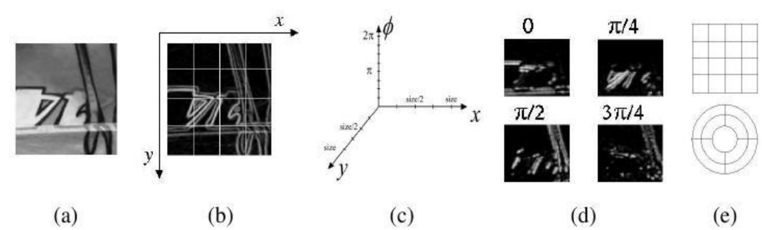
HoG



RIFT



Texton



GLOH

Learning feature hierarchies with Deep Learning

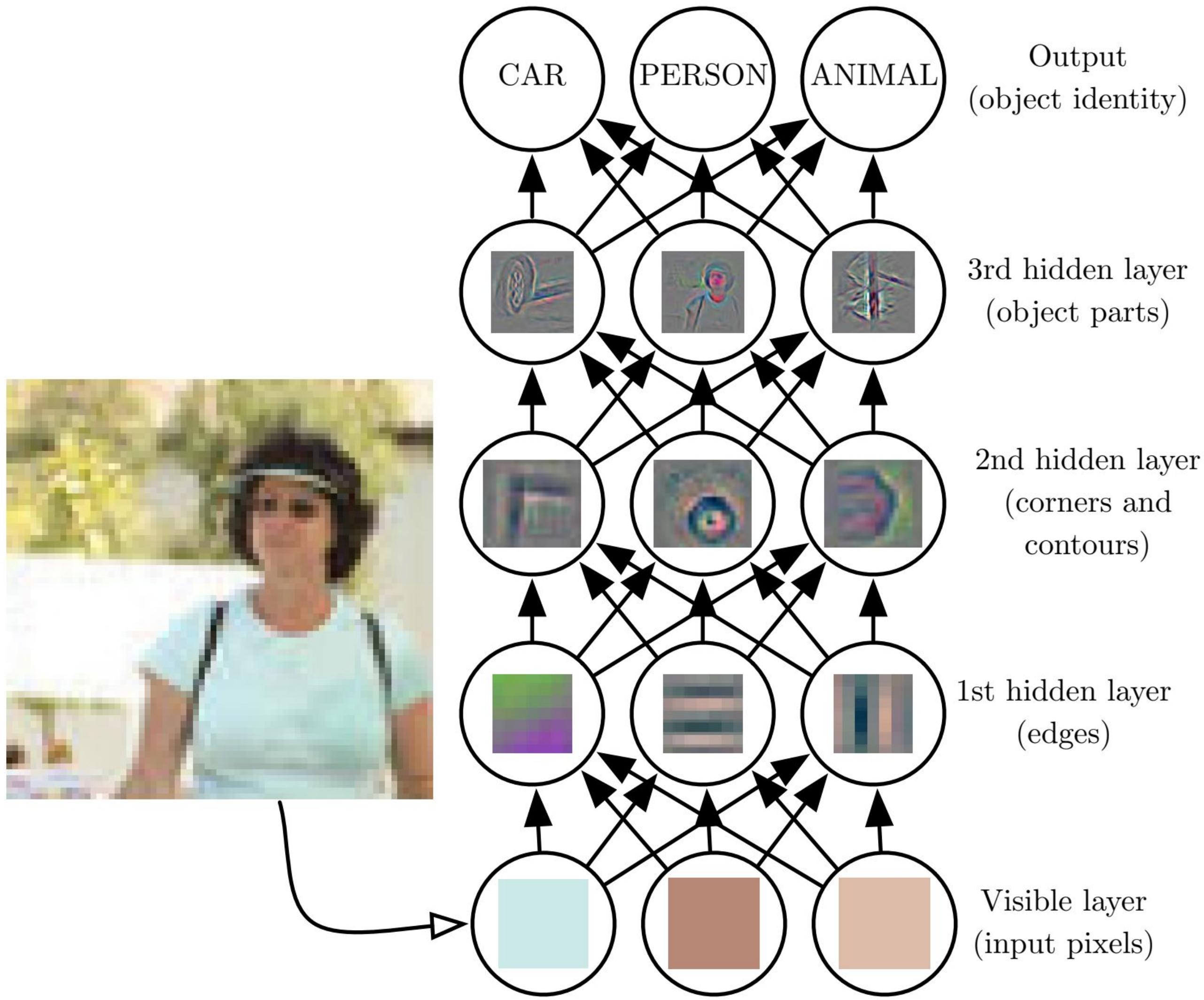
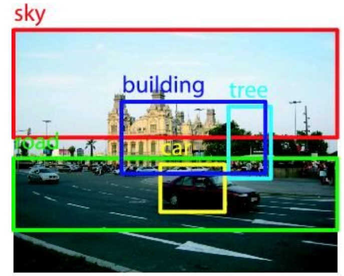


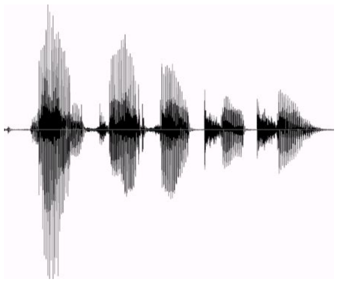
Figure source: Ian Goodfellow, Yoshua Bengio, Aaron Courville. Deep Learning, 2016

ML applications

Examples of ML applications

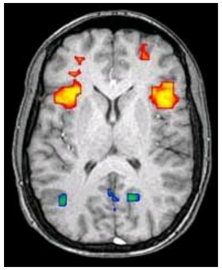
- Computer vision
- Speech recognition
- Robotics
- Text classification
- Medical image recognition
- Time series prediction/classification
- ….





WIKIPEDIA

REUTERS



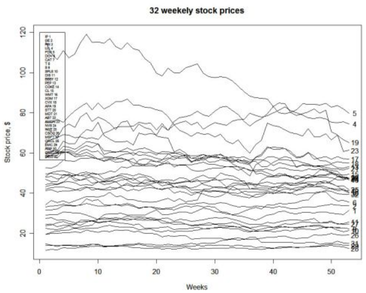


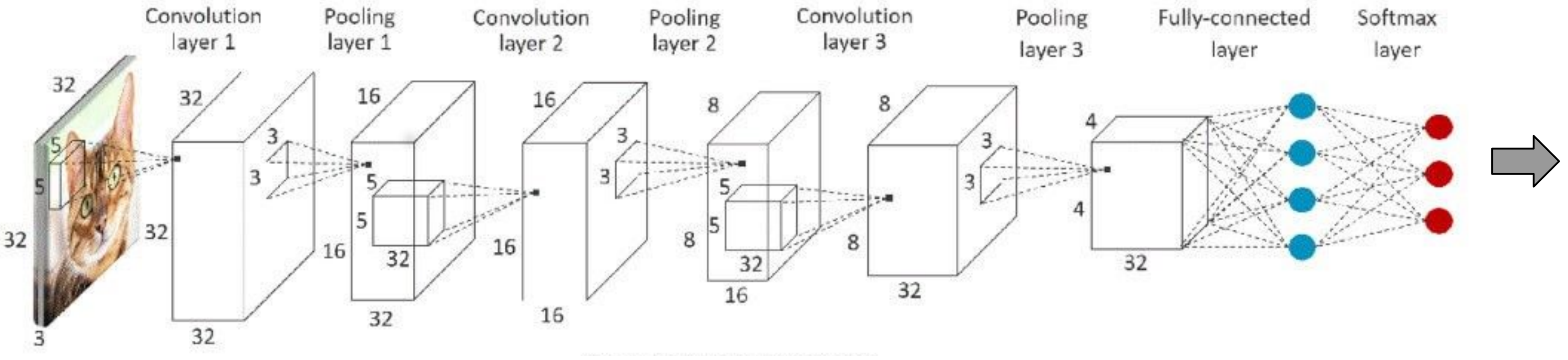
Image classification

- Goal: predict the class of input image

Training dataset
(image, label)

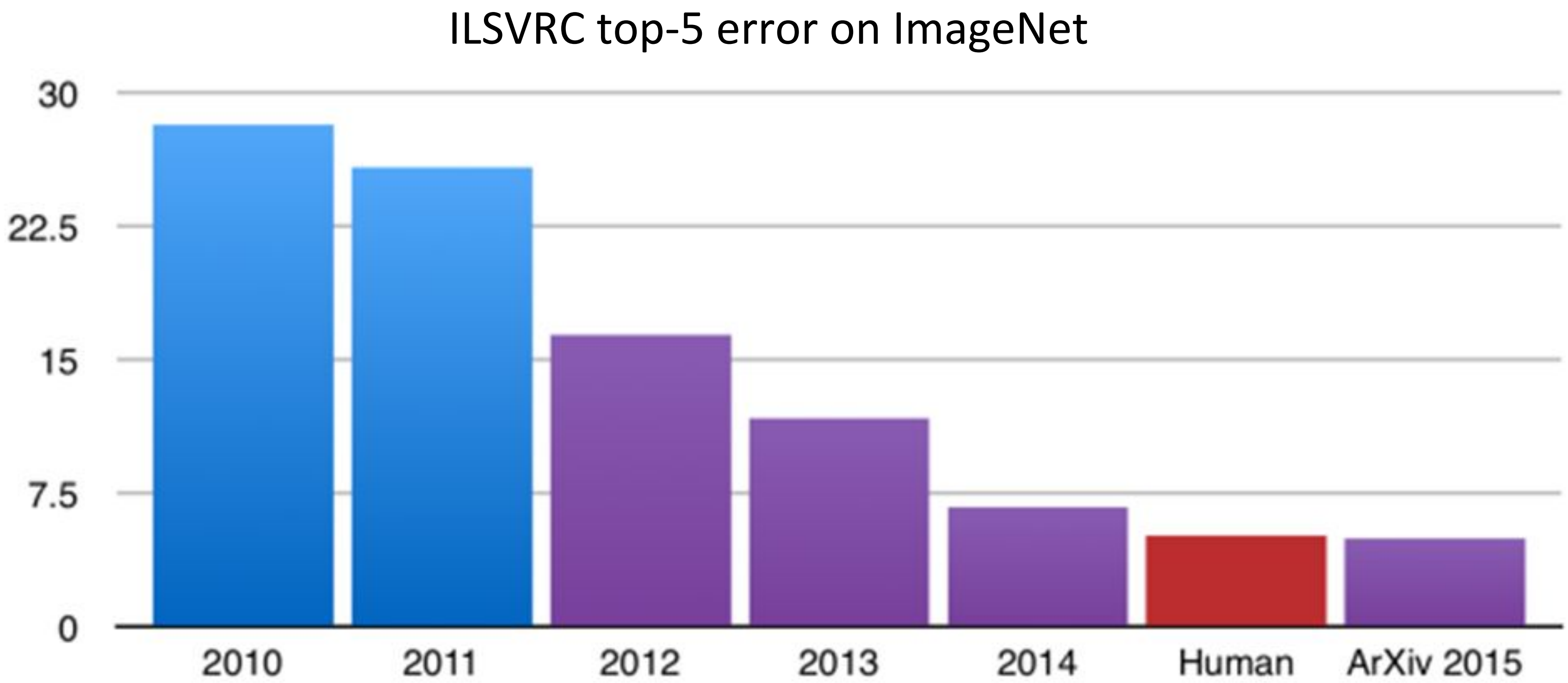
Input

Prediction



cat

Image classification

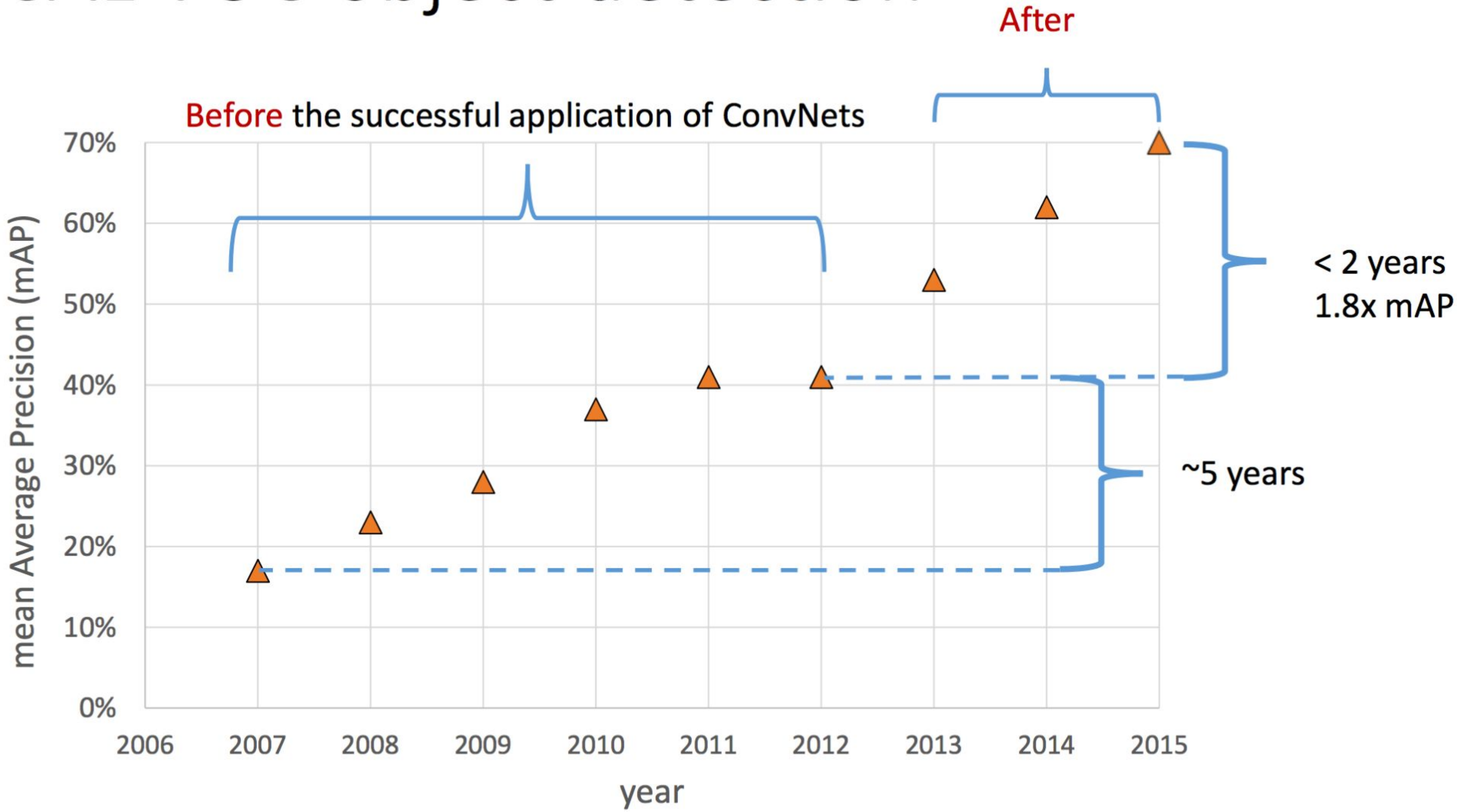


Slide credit: Rob Fergus

Object Detection

Huang et al., Speed/accuracy trade-offs for modern convolutional object detectors. CVPR 2017
figure/code at: https://github.com/tensorflow/models/tree/master/research/object_detection

PASCAL VOC object detection

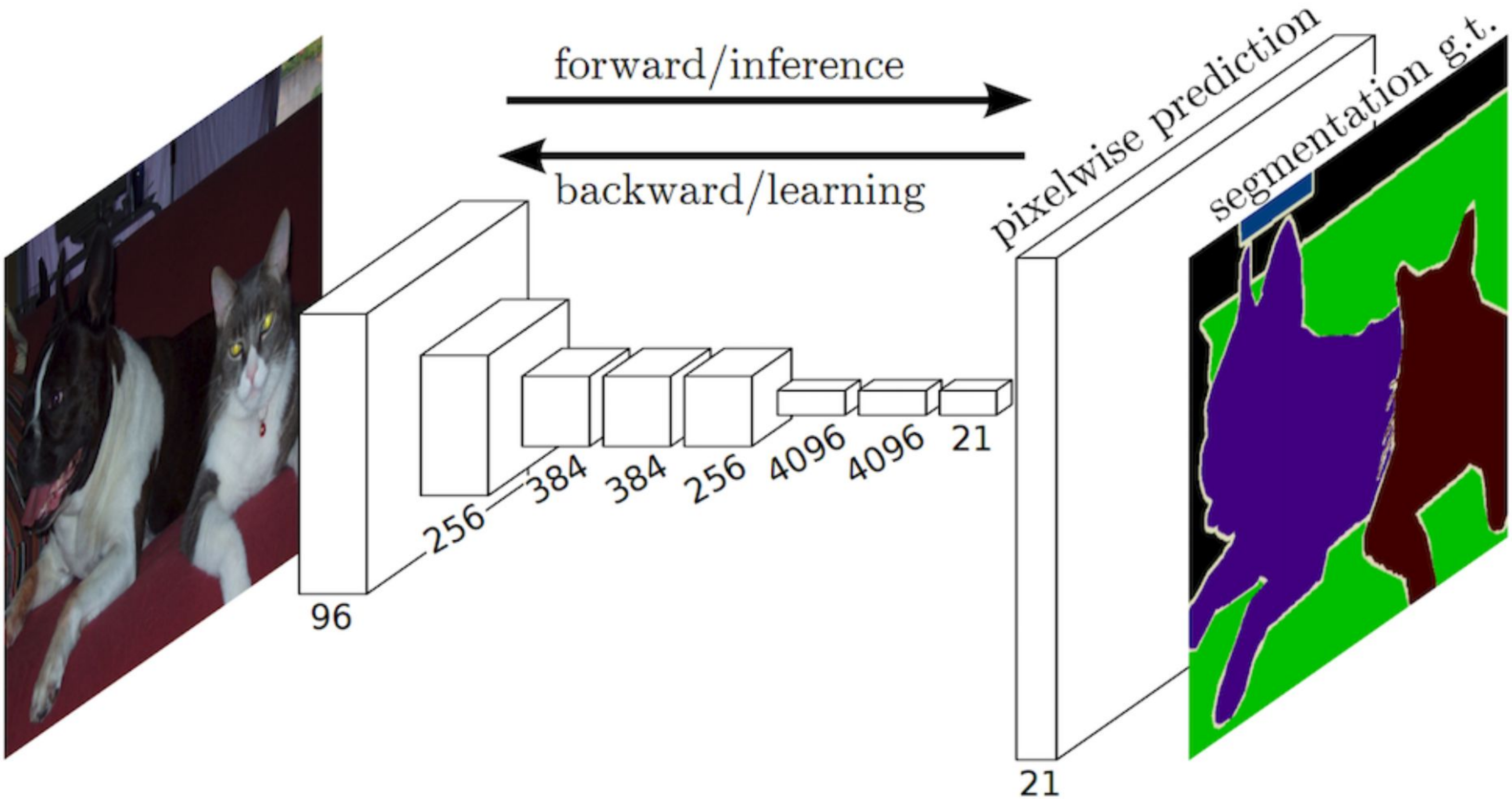


Precision: higher is better

(Figur from Ross Girshick)

Semantic Segmentation

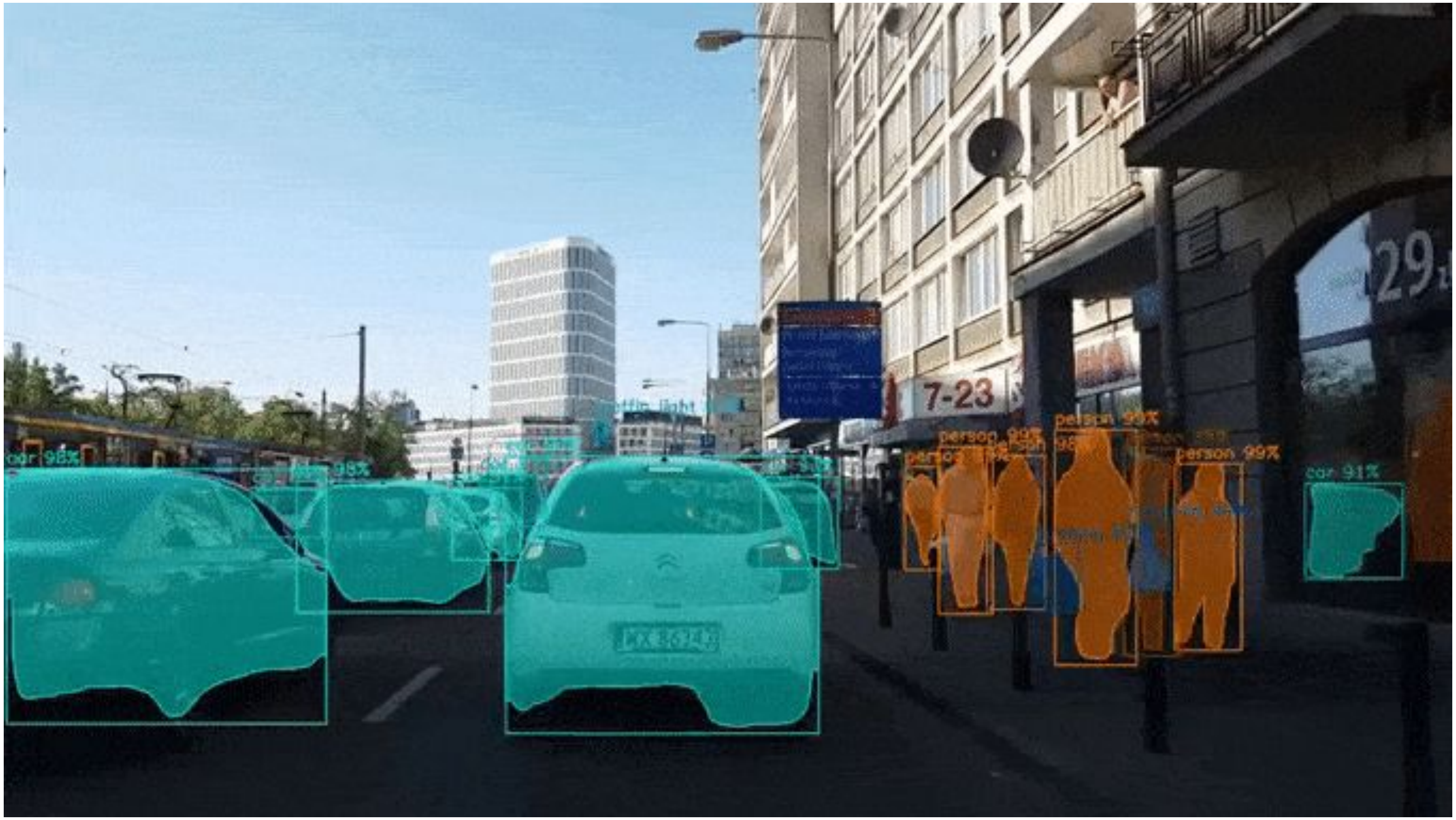
- Goal: Segment object regions and predict class labels for each region
- Can be formulated as pixel-wise classification



(Long et al, "Fully Convolutional Networks for Semantic Segmentation", CVPR, 2015.)

Instance Segmentation

Goal: Perform segmentation (pixel-level masking) for each individual object



Mask R-CNN (He et al., 2017)
Image from: https://github.com/matterport/Mask_RCNN

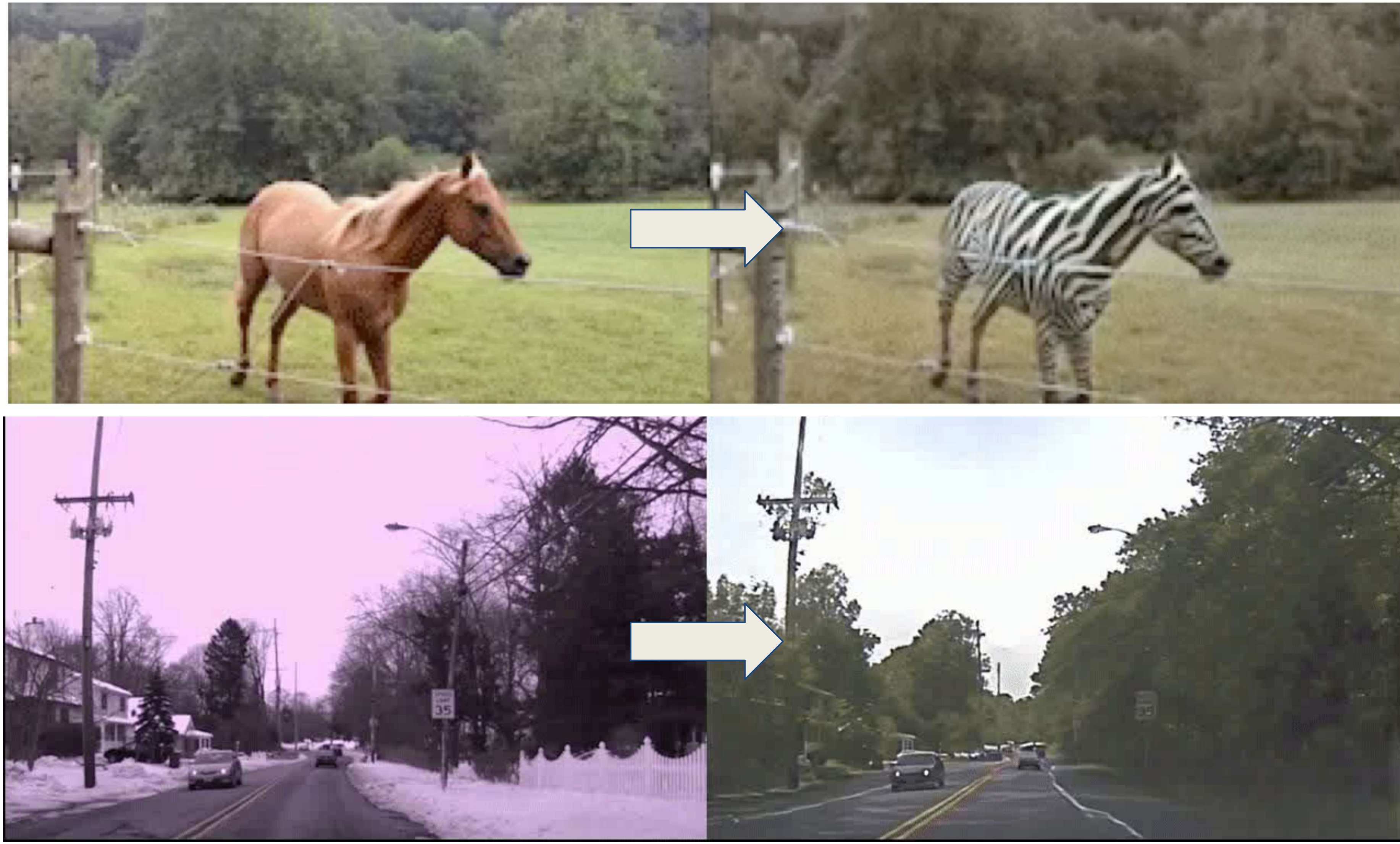
Image Generation: Generative Adversarial Networks

(Radford et al, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks", ICLR, 2016.)

Image Generation: Generative Adversarial Networks

[Karras et al. 2017] Progressive Growing of GANs for Improved Quality, Stability, and Variation

Unsupervised Image-to-Image generation



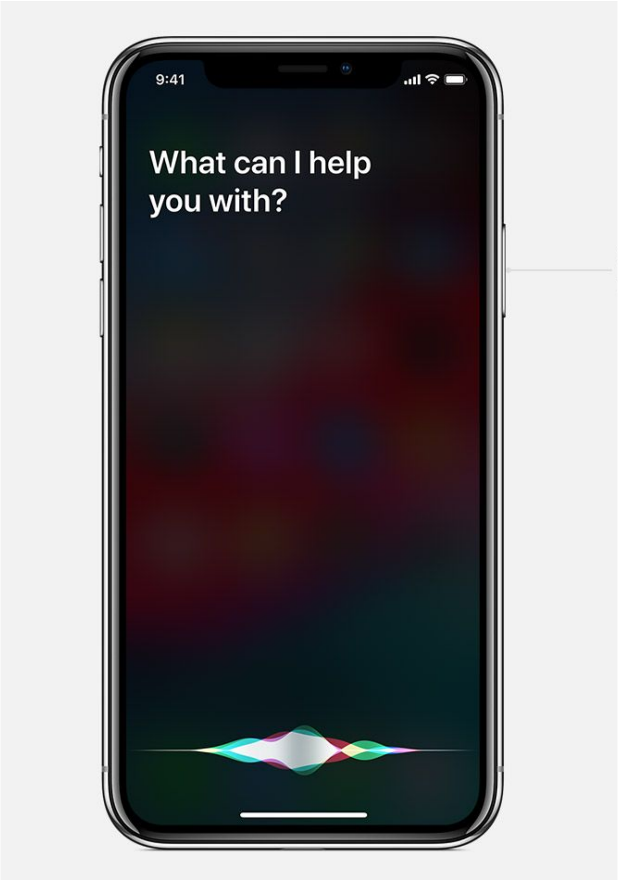
[CycleGAN: Zhu, Park, Isola & Efros, 2017]

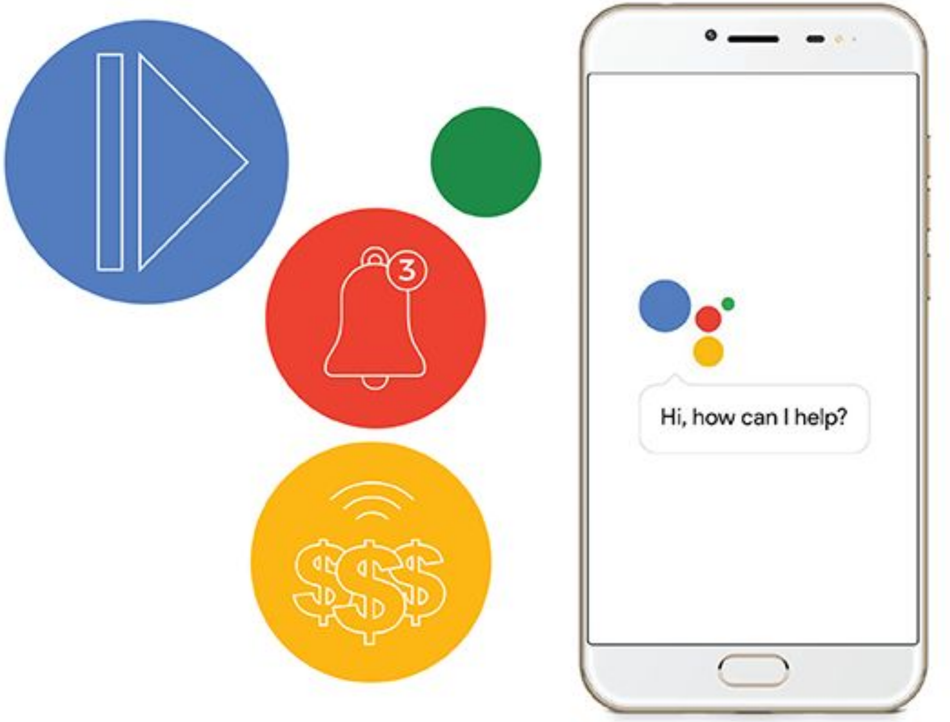
Image Caption Generation

(Vinyals et al, "Show and Tell: A Neural Image Caption Generator", CVPR, 2015.)

Speech recognition

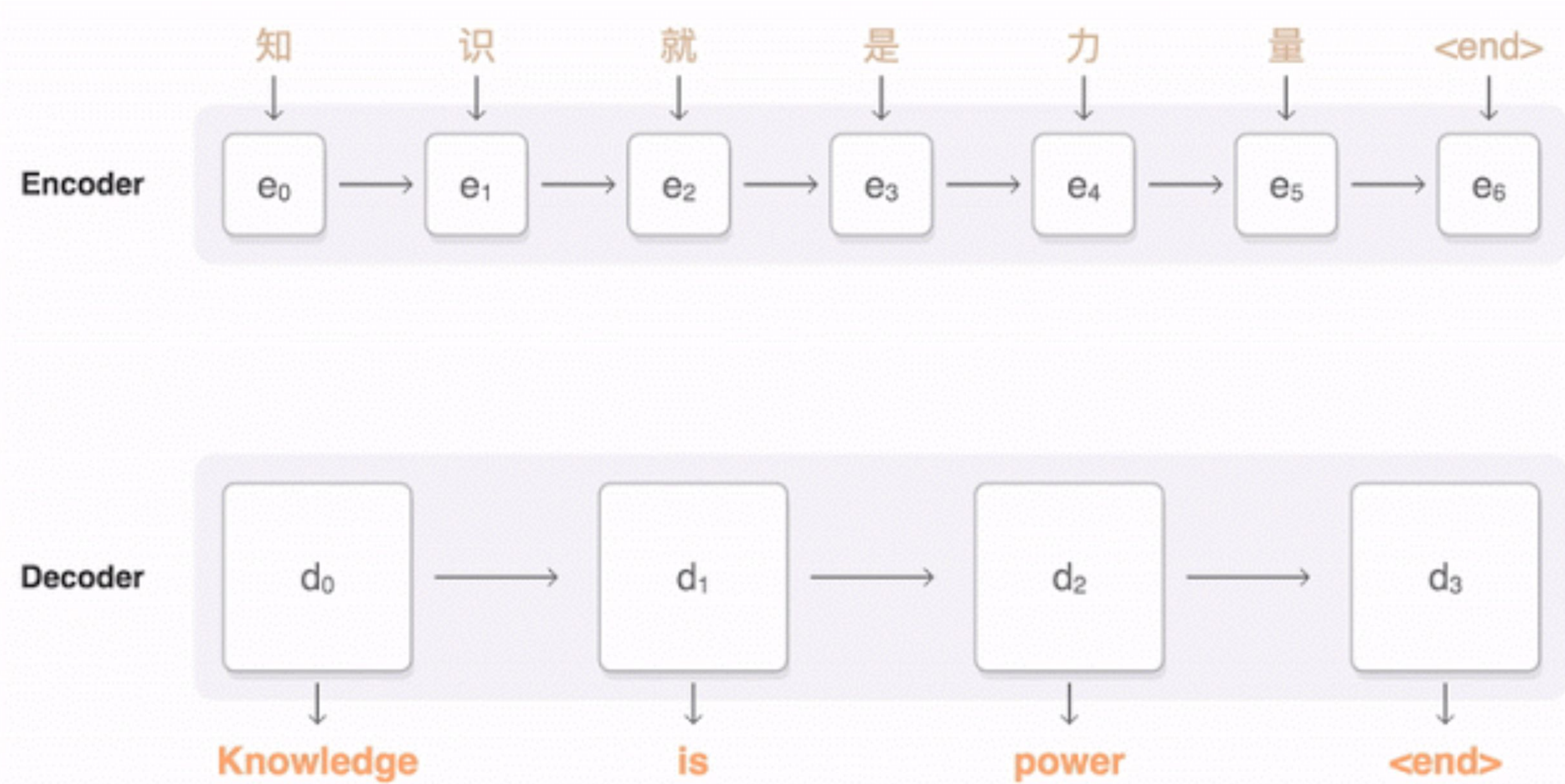
Siri, Google home, and Google assistant achieves commercial-level performance





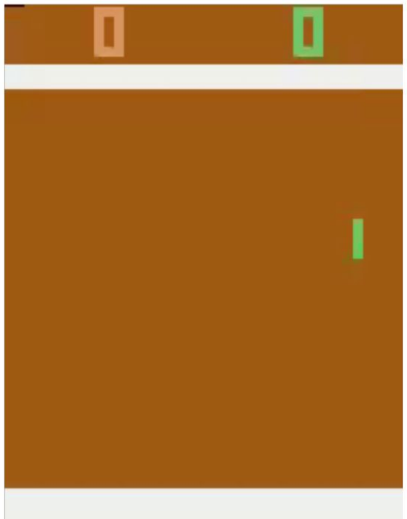
Machine Translation

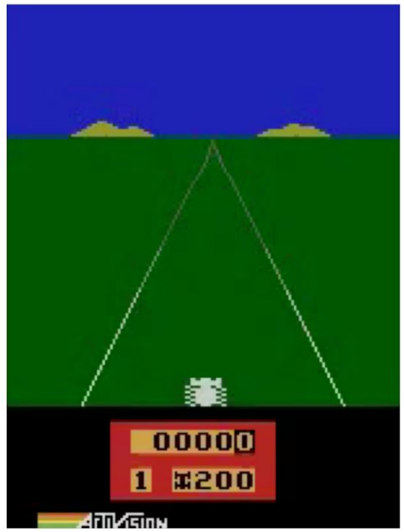
Google Neural Machine Translation (in production)

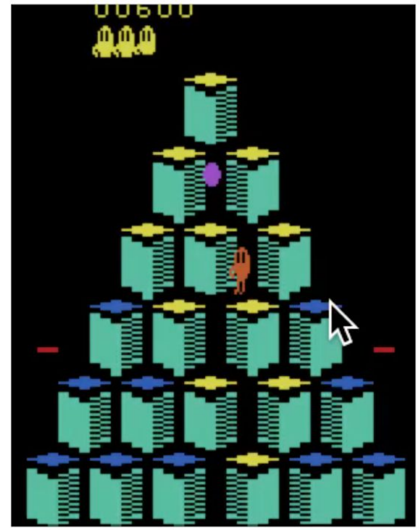


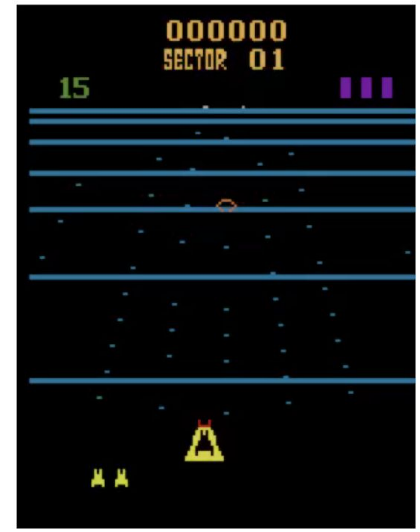
Slide credit: Pieter Abbeel

RL success stories: playing ATARI games









DQN Mnih et al, NIPS 2013 / Nature 2015;

MCTS Guo et al, NIPS 2014; **TRPO** Schulman, Levine, Moritz, Jordan, Abbeel, ICML 2015; **A3C** Mnih et al, ICML 2016; **Dueling DQN** Wang et al ICML 2016; **Double DQN** van Hasselt et al, AAAI 2016; **Prioritized Experience Replay** Schaul et al, ICLR 2016; **Bootstrapped DQN** Osband et al, 2016; **Q-Ensembles** Chen et al, 2017; **Rainbow** Hessel et al, 2017; ...

slide credit: Pieter Abbeel

AlphaGo

- Another breakthrough from Google DeepMind
- Combines Monte-Carlo Tree Search (MTCS) with deep neural networks

AlphaGo Silver et al, Nature 2015
AlphaGoZero Silver et al, Nature 2017
AlphaZero Silver et al, 2017
Tian et al, 2016; Maddison et al, 2014; Clark et al, 2015

slide credit: Pieter Abbeel

OpenAI’s 1v1 Dota [2017] and 5v5 [2018]

Super-human agent on a competitive game, enabled by

- Reinforcement learning
- Self-play
- Enough computation

Cooperation emerges

slide credit: Pieter Abbeel

Robot learning

Levine et al., Learning Hand-Eye Coordination for Robotic Grasping. 2016
video: https://www.youtube.com/watch?v=cXaic_k80uM

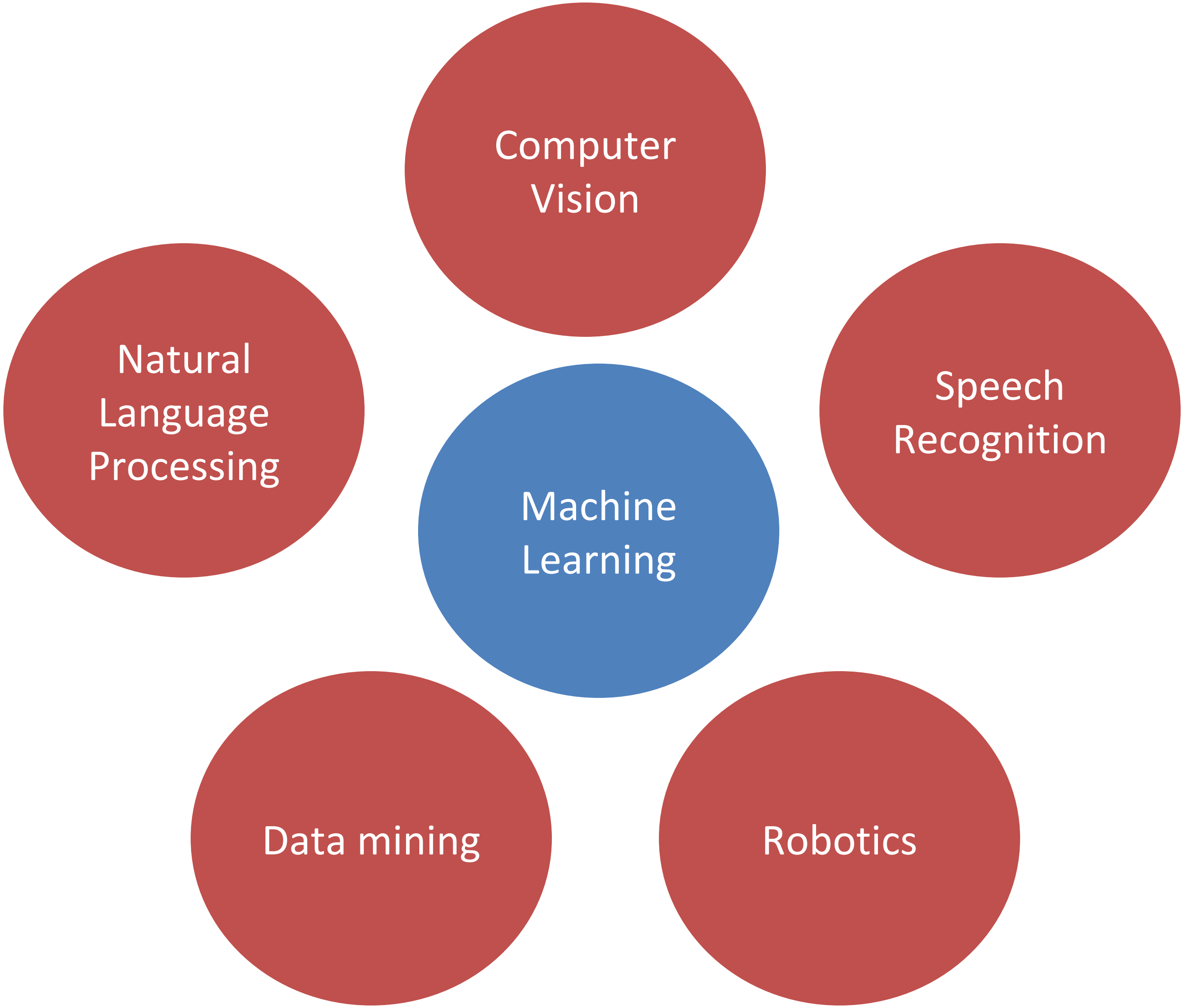
slide credit: Pieter Abbeel

Self-driving cars

https://youtu.be/O6DRfAC1JXA

See also: Chris Urmson: How a driverless car sees the road https://youtu.be/tiwVMrTLUWg

Machine Learning and other fields



Next class

- Supervised learning
  - Linear regression

Reminder

- Check syllabus at Canvas
- For all questions, please use Piazza (linked to Canvas)

Questions?