EECS 498/598: Deep Learning
Lecture 1. Introduction

Honglak Lee

1/11/2019

Outline

- Administrative

- What is deep learning?

Teaching staff

- Instructor: Honglak Lee
  - Email: honglak@eecs.umich.edu
  - Office: CSE 3773
- Graduate Student Instructor:
  - Ruben Villegas (rubville@umich.edu)
  - Lajanugen Logeswaran (llajan@umich.edu)
  - Yijie Guo (guoyijie@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

- Introduction of deep learning
- Foundations of deep learning
  - Mathematical derivation, Implementation of the algorithms
- Applications of deep learning
  - Computer vision, language, speech, robotics, etc.
- Lots of implementation and hands-on experience (via in-class activities, homeworks, projects)
- Open-ended projects on research problems and/or practical applications

About this course

- Our goal is to help you to
  - Understand fundamentals of deep learning
  - Learn technical details of DL algorithms
  - Learn how to implement important algorithms
  - Use deep learning algorithms for your problems/applications of interest.

Topics

- Deep neural networks
- Backpropagation; optimization & regularization
- convolutional neural networks
- Recurrent neural networks
- Representation learning and embedding methods
- Generative models: GANs and VAEs
- Deep reinforcement learning
- Applications to vision, language, and others

Text books

- (Recommended) Ian Goodfellow and Yoshua Bengio and Aaron Courville, Deep Learning, MIT Press.
  - Free online version available at: https://www.deeplearningbook.org/

Prerequisites

- **Machine Learning (EECS 445 or EECS 545)**
- Fluent mathematical skills and knowledge in linear algebra (equivalent to MATH 217 or MATH 417), multivariate calculus, probability (equivalent to EECS 401), and statistics
- Fluent programming skills (we plan to use python-based pytorch as programming tools). We expect solid programming background (covered by EECS 281 and some upper ULCS courses).

* Note: This course will not cover basics of machine learning, but it will assume cumulative knowledge and understanding of ML (at the level of EECS 445 or 545).

Background materials: machine learning

- Chris Bishop, “Pattern Recognition and Machine Learning”. Springer, 2007.
- Kevin Murphy, “Machine Learning: A Probabilistic Perspective”, MIT Press, 2012
- Hastie, Tibshrani, Friedman, "Elements of Statistical Learning," Springer, 2010. (available online)
- David Barber, “Bayesian Reasoning and Machine Learning”, Cambridge University Press, 2012 (available online)
- Sutton and Barto, "Reinforcement Learning: An Introduction," MIT Press, 1998 (available online)
- Boyd and Vandenberghe, "Convex Optimization," Cambridge University Press, 2004. (available online)
- Mackay, "Information Theory, Inference, and Learning Algorithms," Cambridge University Press, 2003. (available online)

Grading policy

- Homework: 40%
- Midterm: 30% (tentative date: late March/early April)
- Project: 30%
  - progress report (10%)
  - final report (20%)

- 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 deep 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
- GSIs will run coding sessions and will provide answers on Piazza

Homework

- There will be 3-4 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 Friday (1/18) – due 2/6, 5pm.

Late days

- 3 maximum late days per assignment
  - No homework will be accepted 3 days after the due date.
- Total 7 late days allowed
- After using up all late days, your assignment will be penalized by 20% from your scores.

Study group

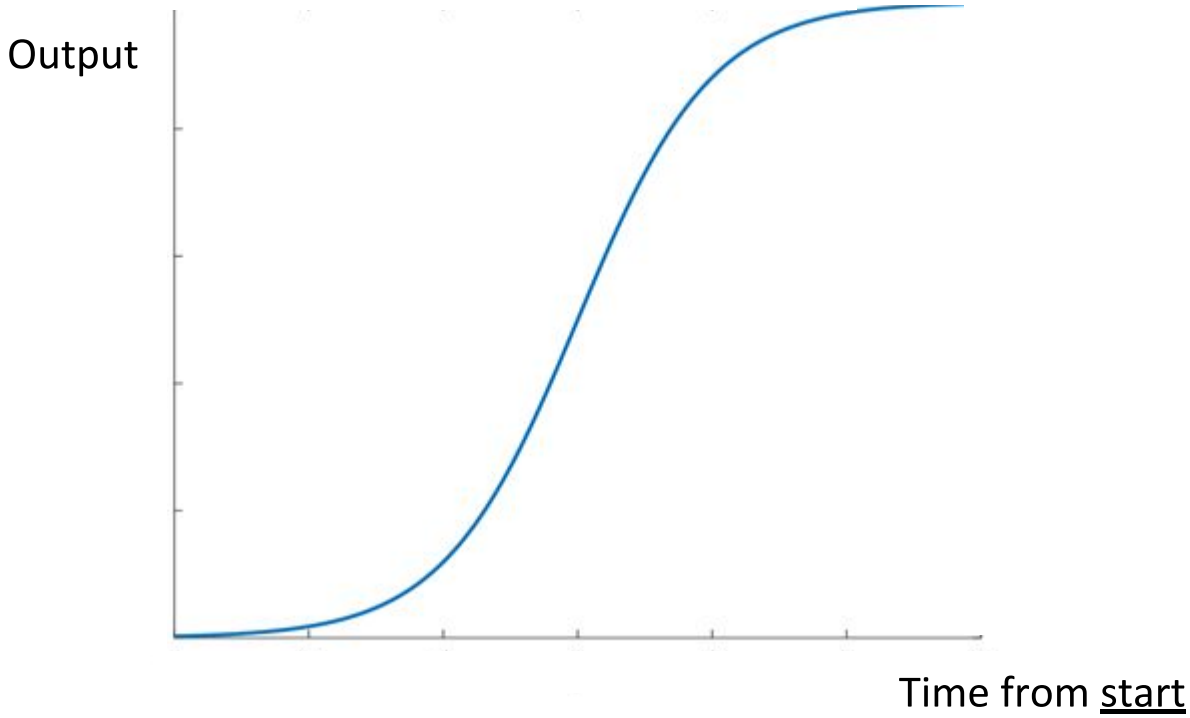
- Form your study group early on!
  - Up to four people are allowed.
- For homework, you may discuss between the study group members, but you should write your own solution independently.
- In the homework submissions, you must put:
  - the names of other people you collaborated.
  - submission time.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

Course Project

- Scope
  - develop new algorithms and theory in deep learning
  - apply existing algorithms to new problems
  - apply to your own problems of interest
- Milestones (tentative)
  - project proposals (tentative due date: 2/13)
  - progress reports (tentative due date: 3/20)
  - poster presentations (tentative date: 4/26)
  - Final project report (tentative due date: 4/30)
- Requirement
  - Write a 8-page paper
  - Submit the final code
  - Give a poster presentation
- Evaluation is based on:
  - novelty, technical quality, and significance 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 session
  - Will hold review sessions on background materials (linear algebra, probability, Matlab, etc.)

- Beginning-of-course survey

Any questions?

Outline

- Administrative

- What is deep learning?

Deep Learning, ML, and AI

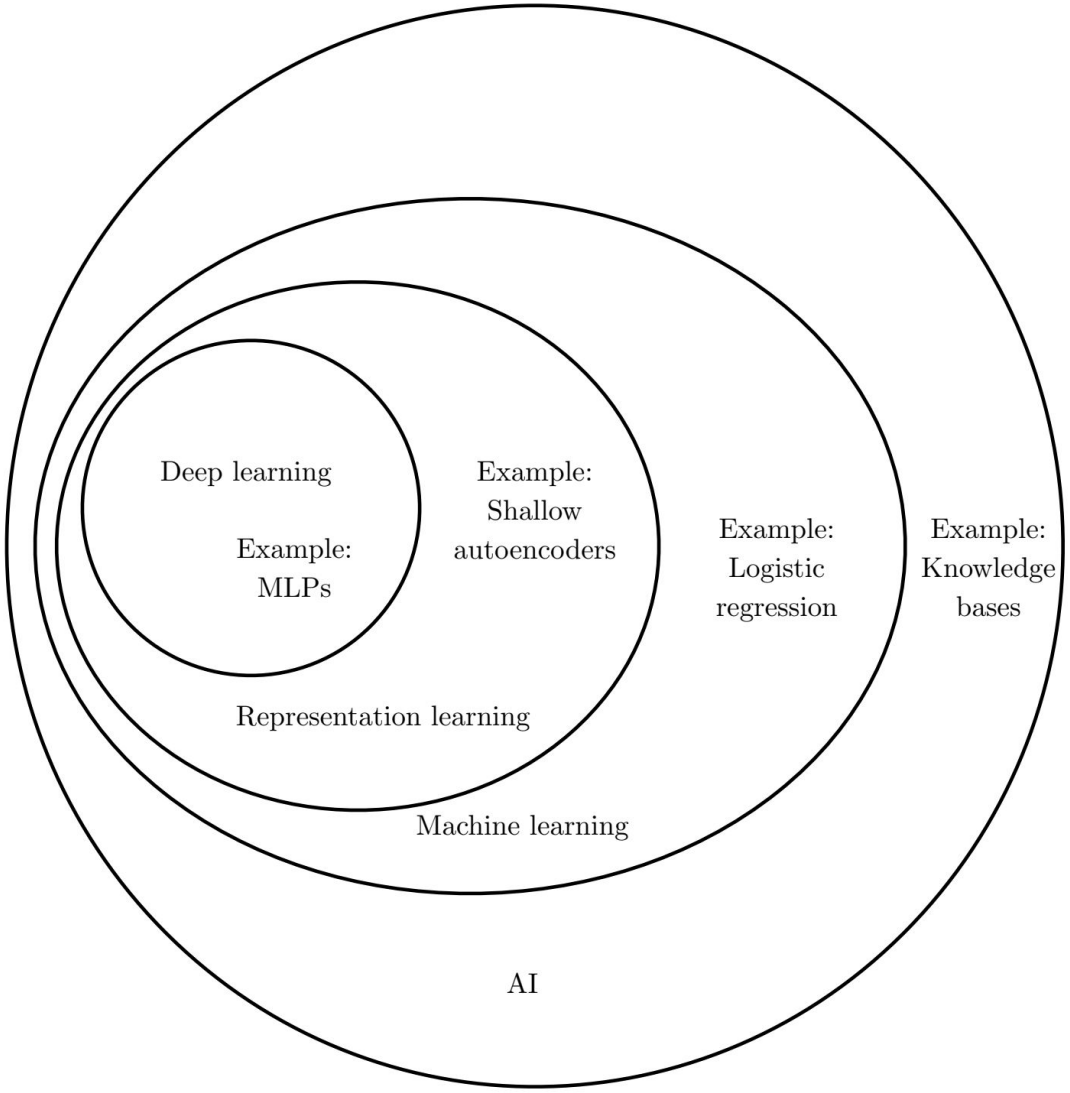


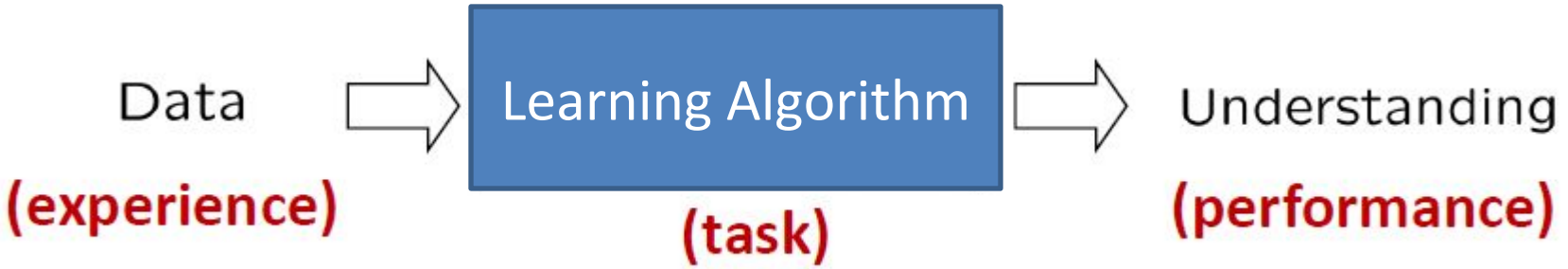
Figure source: Ian Goodfellow, Yoshua Bengio, Aaron Courville. Deep Learning, 2016

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).



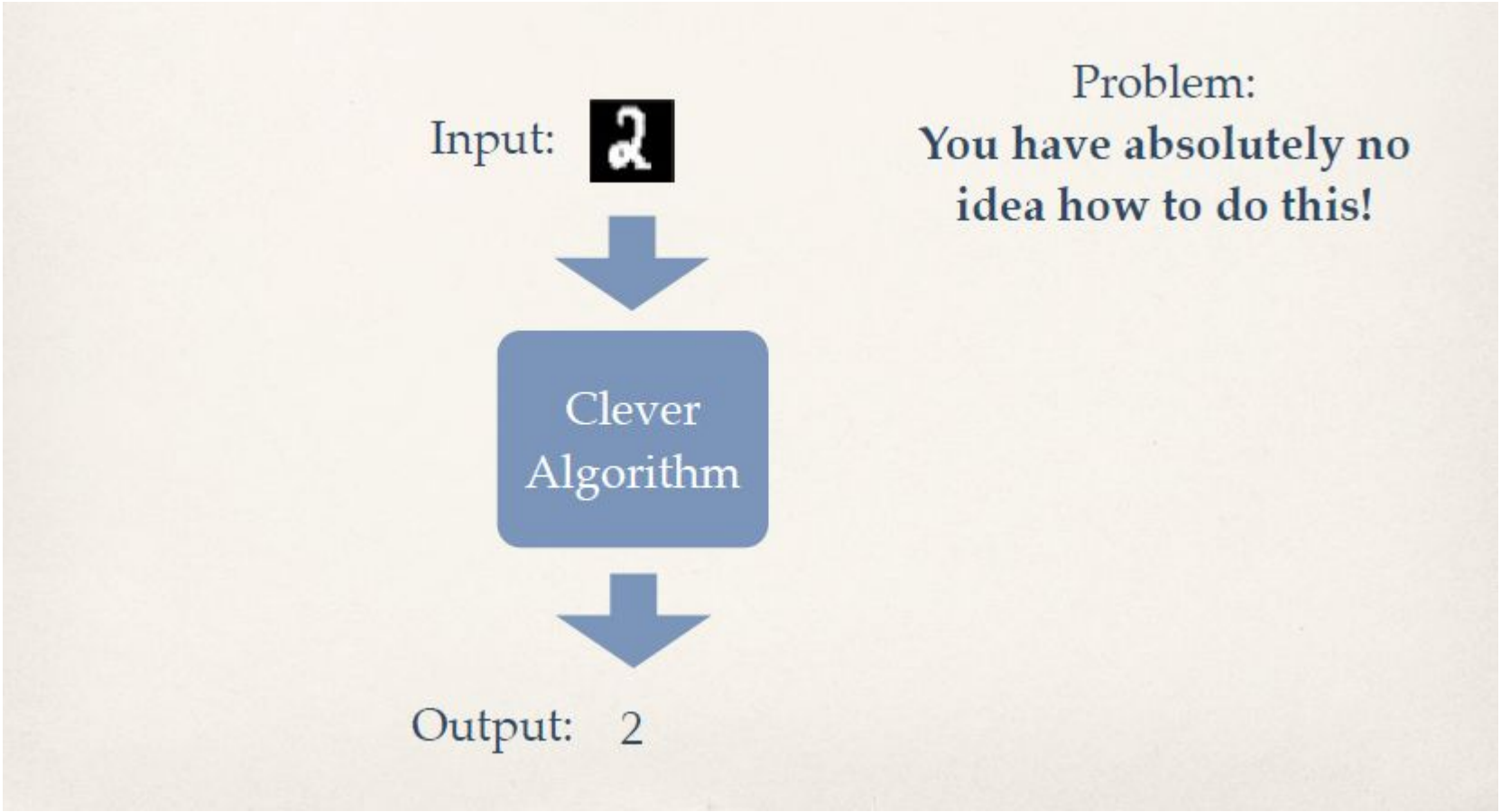
Machine Learning Tasks

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

Deep Learning techniques can be applied to all of the above problem domains.

Example

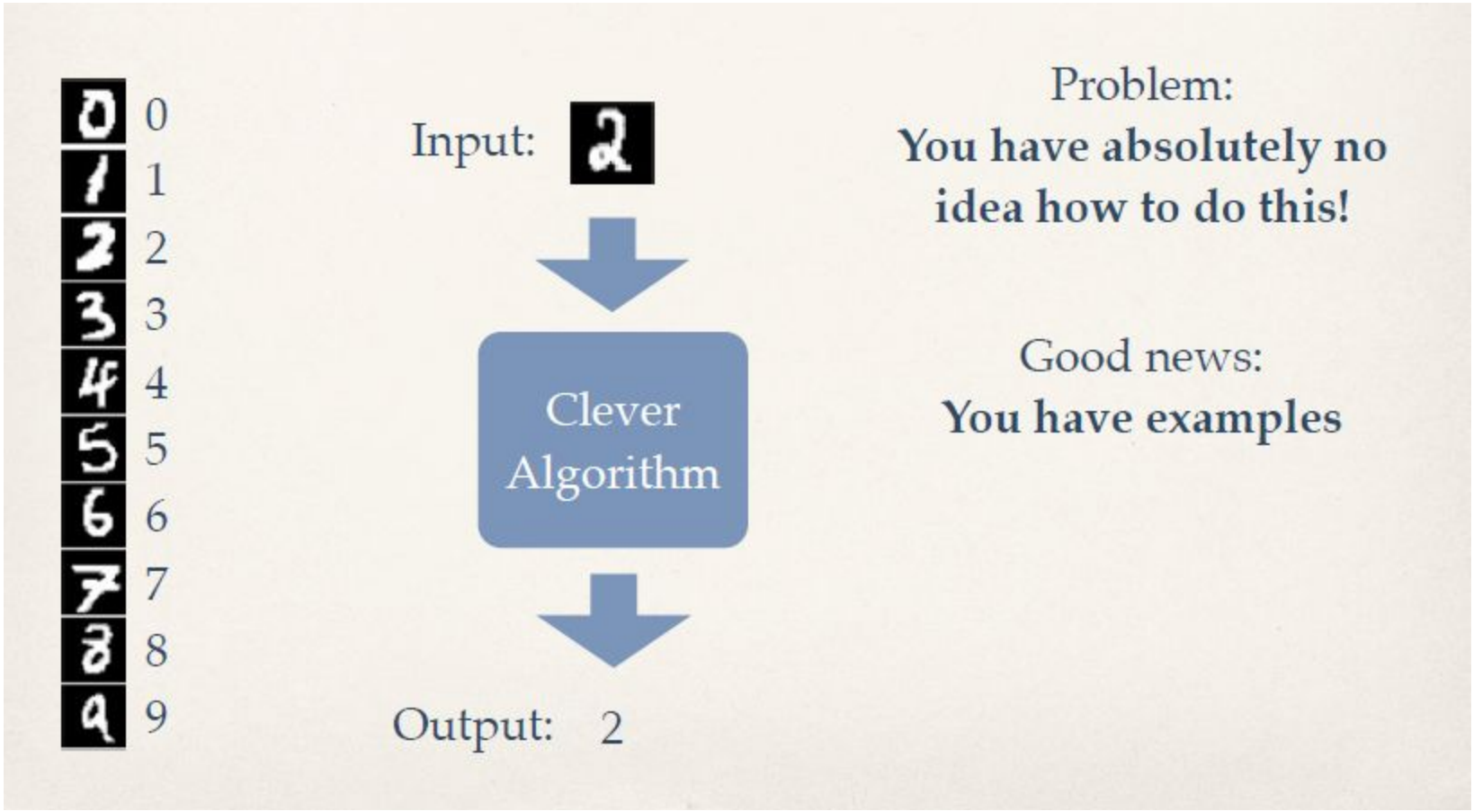
- Problem: Given an image of a handwritten digit, what digit is it?



Slide credit: Kilian Weinberger

Example

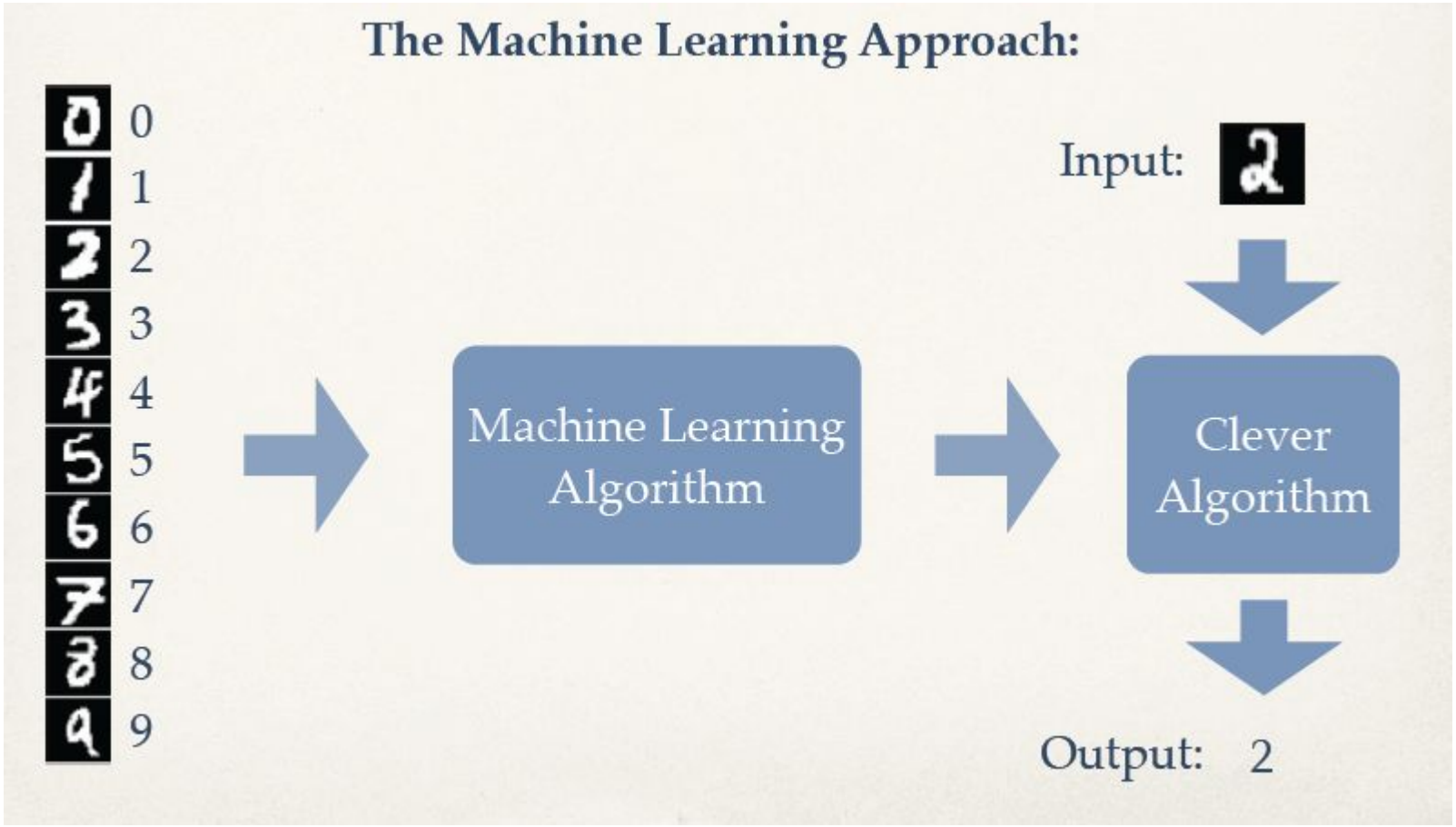
- Problem: Given an image of a handwritten digit, what digit is it?



Slide credit: Kilian Weinberger

Example

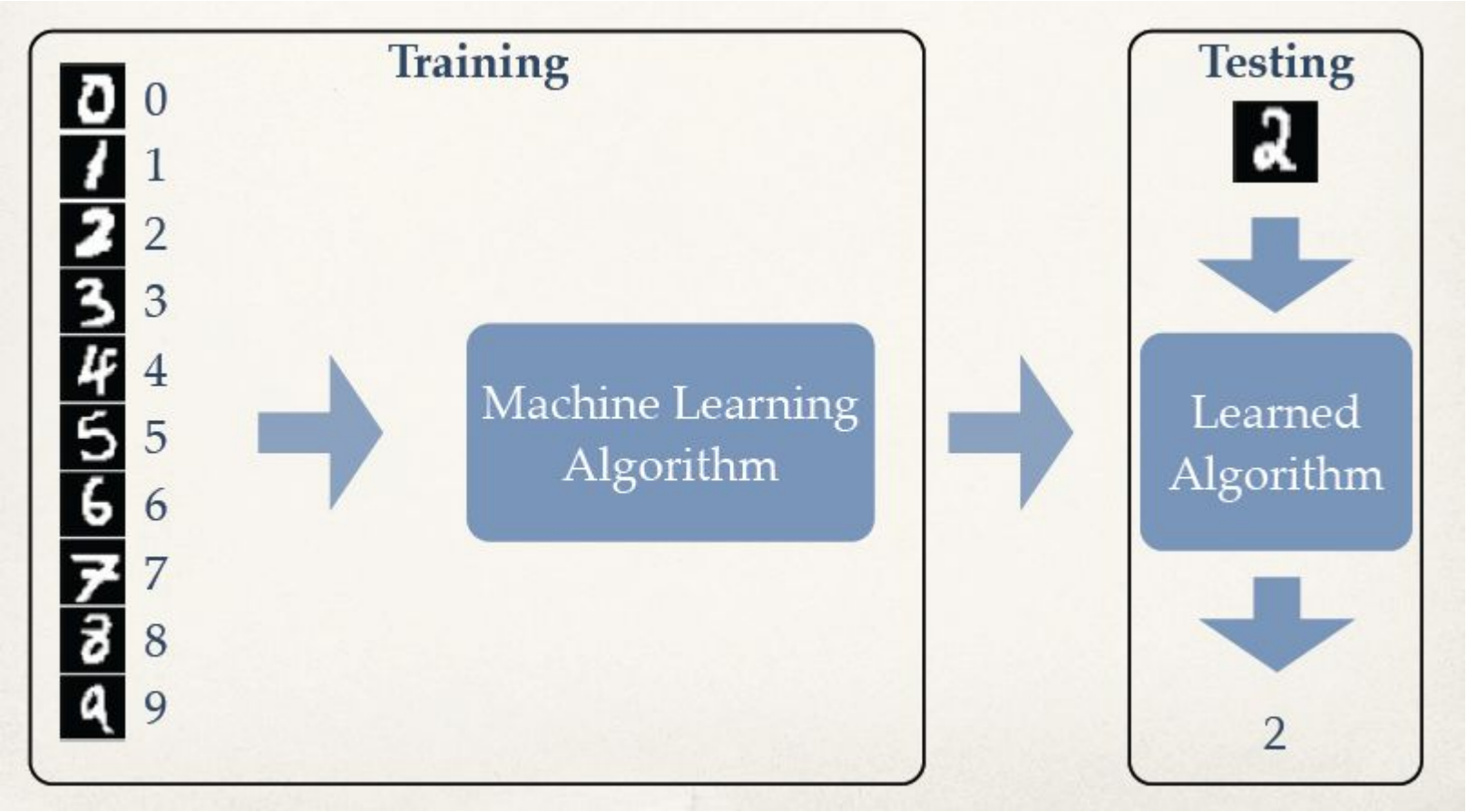
- Problem: Given an image of a handwritten digit, what digit is it?



Slide credit: Kilian Weinberger

Example

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



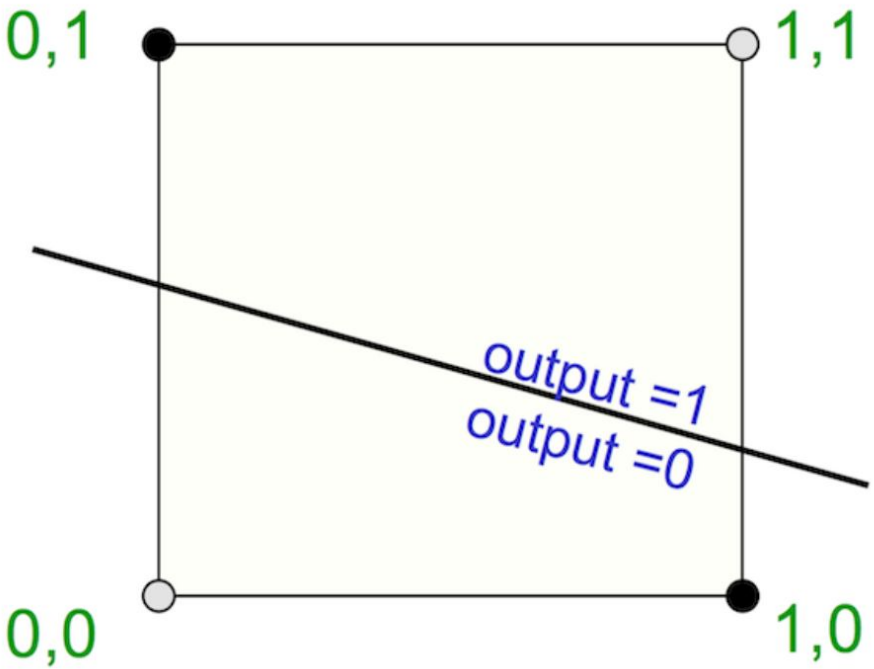
Slide credit: Kilian Weinberger

Limitations of Linear Classifiers

- Linear classifiers (e.g., logistic regression) classify inputs based on linear combinations of input x_i
- Many decisions involve non-linear functions of the input

Limitations of Linear Classifiers

- Canonical example (XOR function)
  - The positive/negative examples are not *linearly separable*.
  - Need to map the input (x_1, x_2) to a feature space where examples are linearly separable.



(Figure from Raquel Urtasun & Rich Zemel)

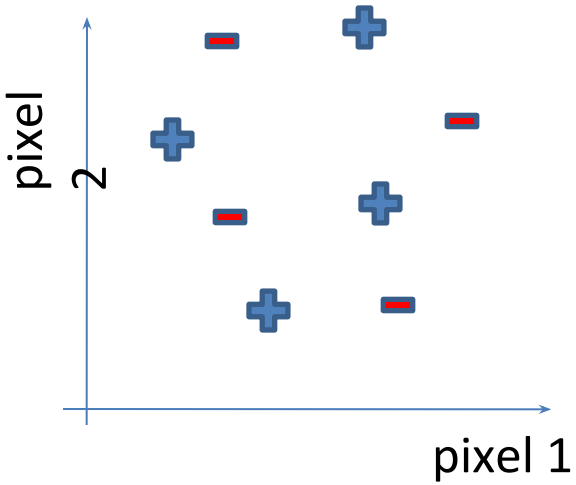
Feature representations

Input

Learning algorithm

Input space

Motorbikes
"Non"-Motorbikes



Feature representations

Input

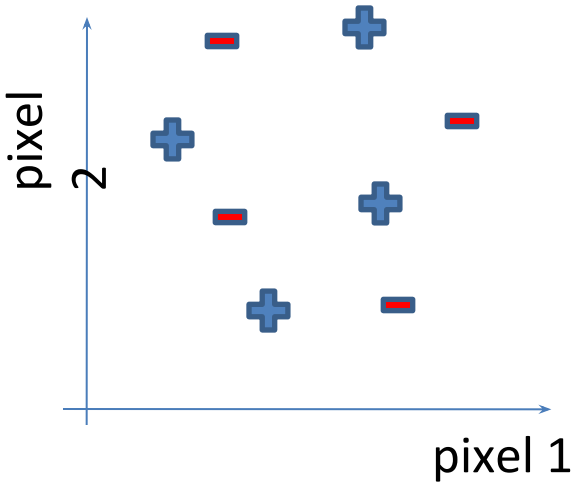
Feature representation

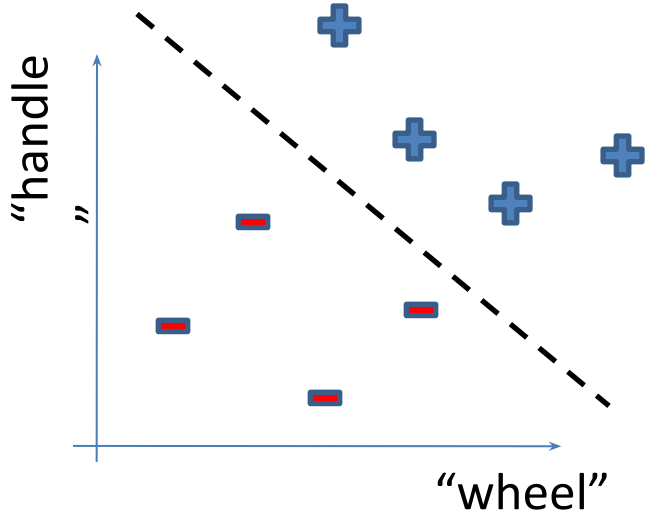
Learning algorithm

Input space

Motorbikes
"Non"-Motorbikes

Feature space

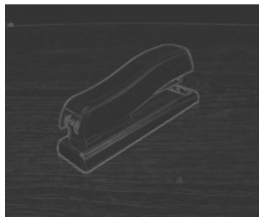




(Traditional) Computer Perception Pipeline

Object detection

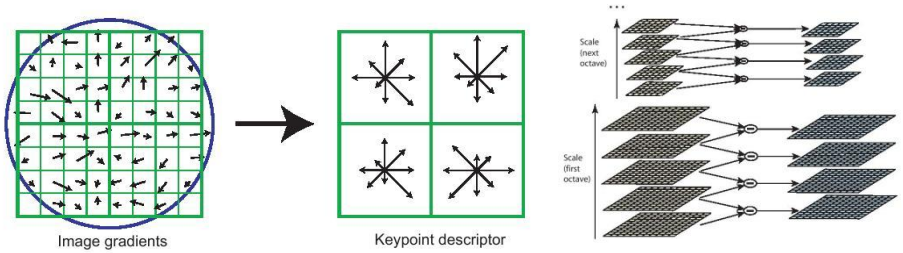
Imag
e



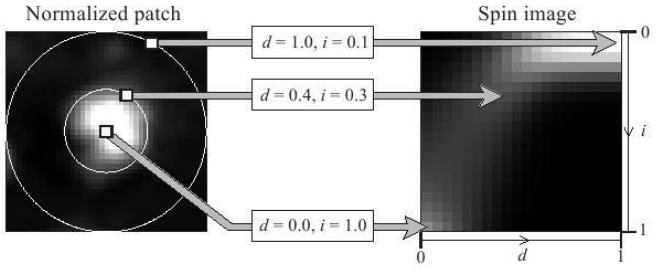
Low-level vision features

Recognitio
n

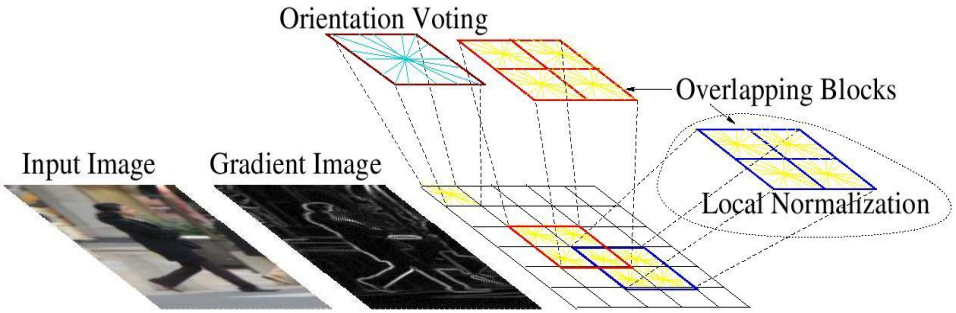
(Traditional) Computer vision features



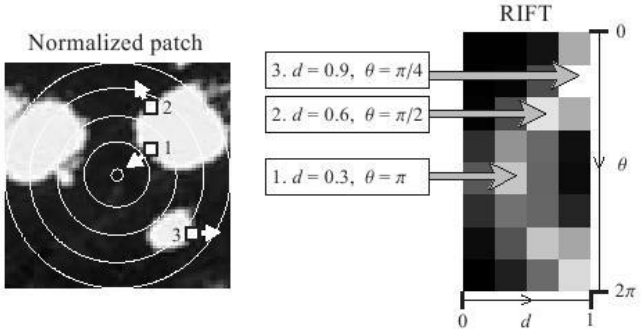
SIFT



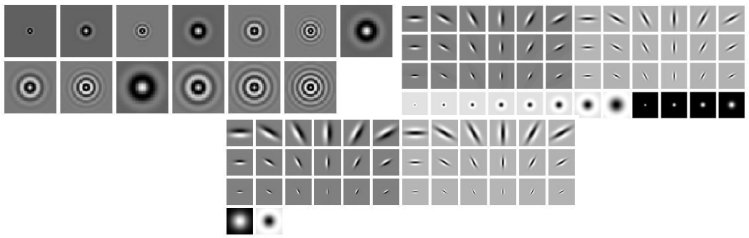
Spin image



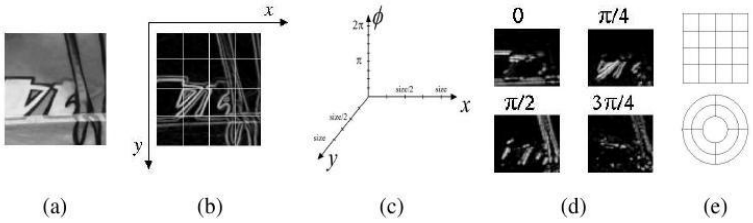
HoG



RIFT



Texton



GLOH

Learning Features

- Most traditional machine learning systems require careful hand-crafted features.

- However, Deep Learning has emerged as an effective way to automatically learning the feature representation, achieving state-of-the-art in many domains
  - such as vision, speech, language, robotics, etc.

Learning feature hierarchies with Deep Learning

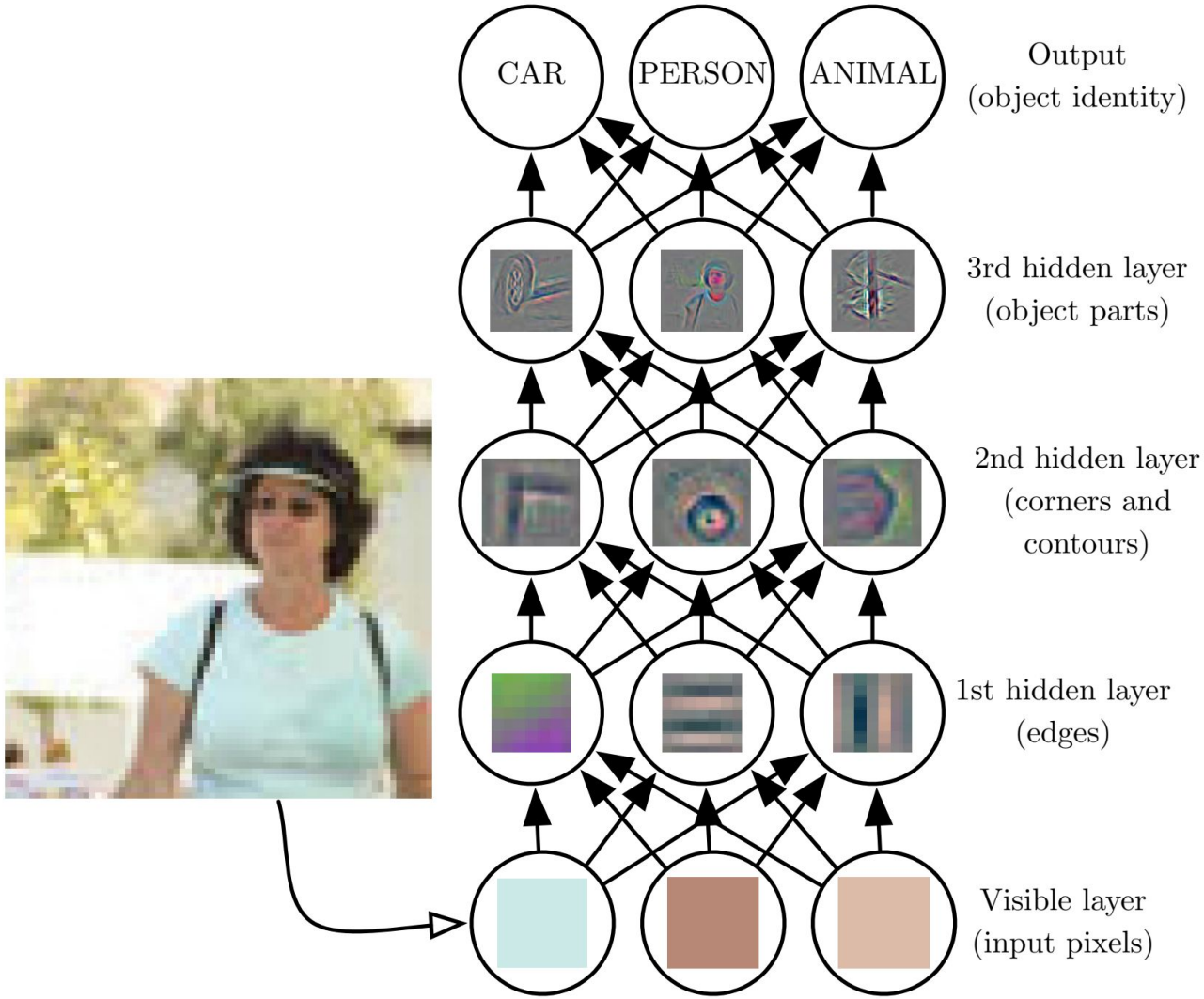
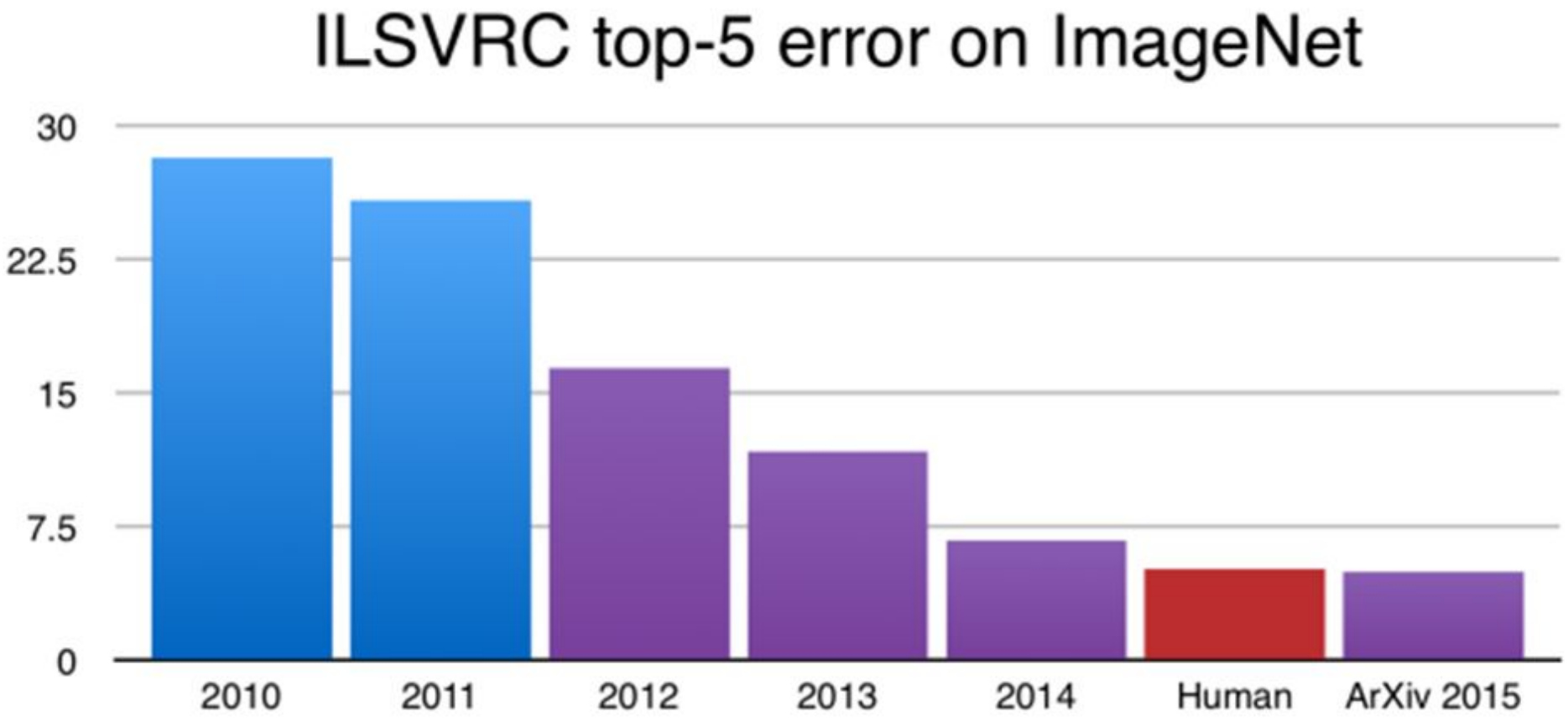


Figure source: Ian Goodfellow, Yoshua Bengio, Aaron Courville. Deep Learning, 2016

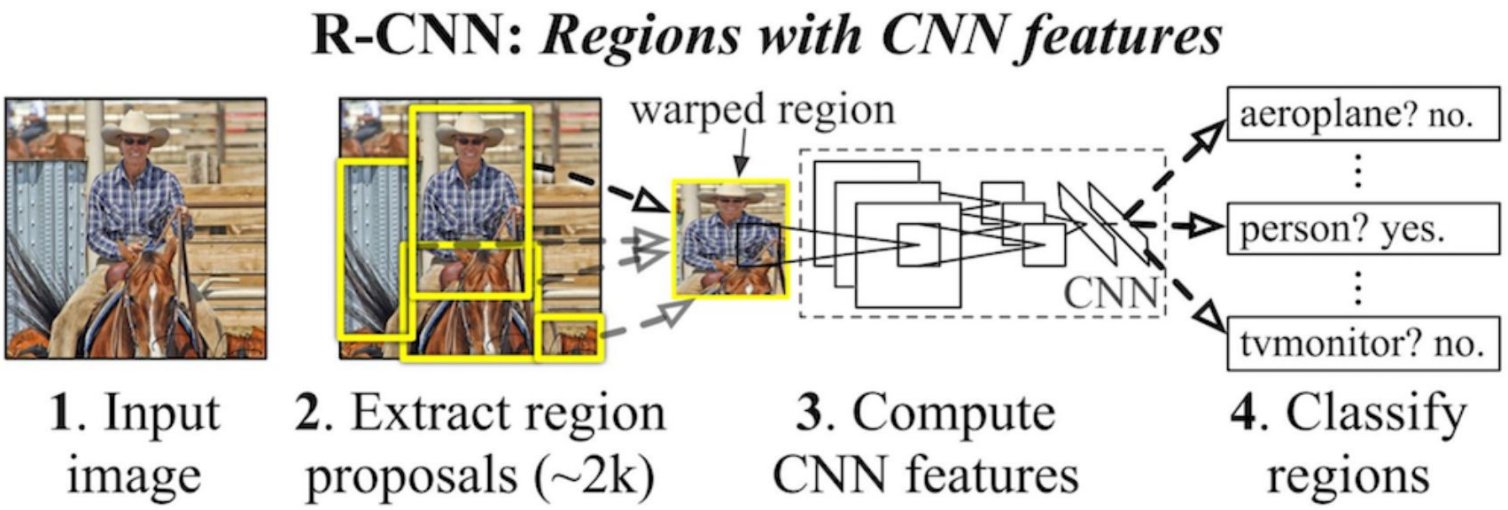
Image classification



Slide credit: Rob Fergus

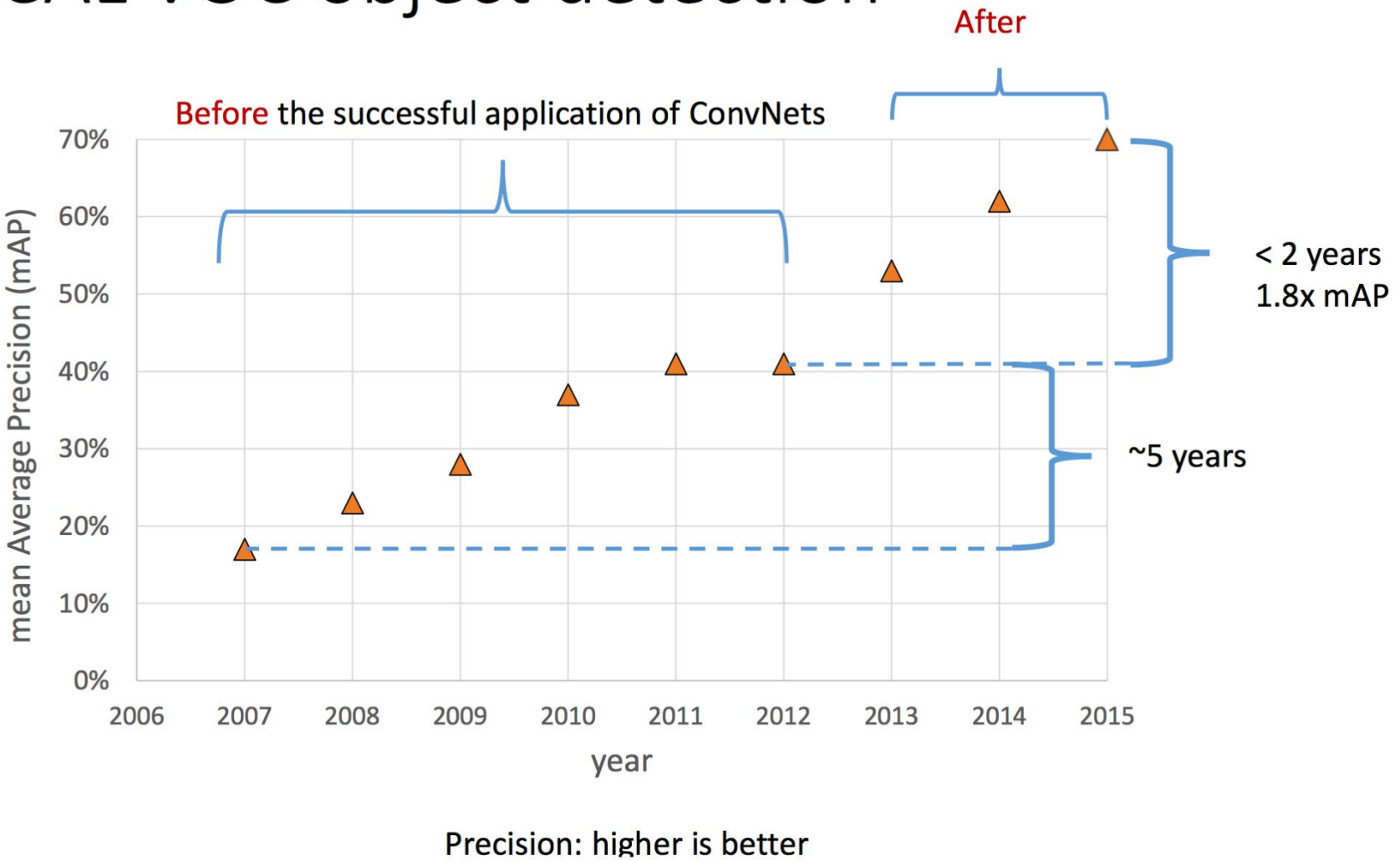
Object Detection

- Goal: find bounding boxes that contain objects in an image and predict their classes.
- CNN approaches have recently achieved state-of-the-art results on object detection task.
- Example: Regions with CNN
    - Use a low-level region proposal methods to generate many candidate bounding boxes
    - Use a pre-trained CNN to classify each region



(Girshick et al, "Reigion-based Convolutional Networks for Accurate Object Detection and Semantic Segmentation", PAMI, 2015.)

PASCAL VOC object detection



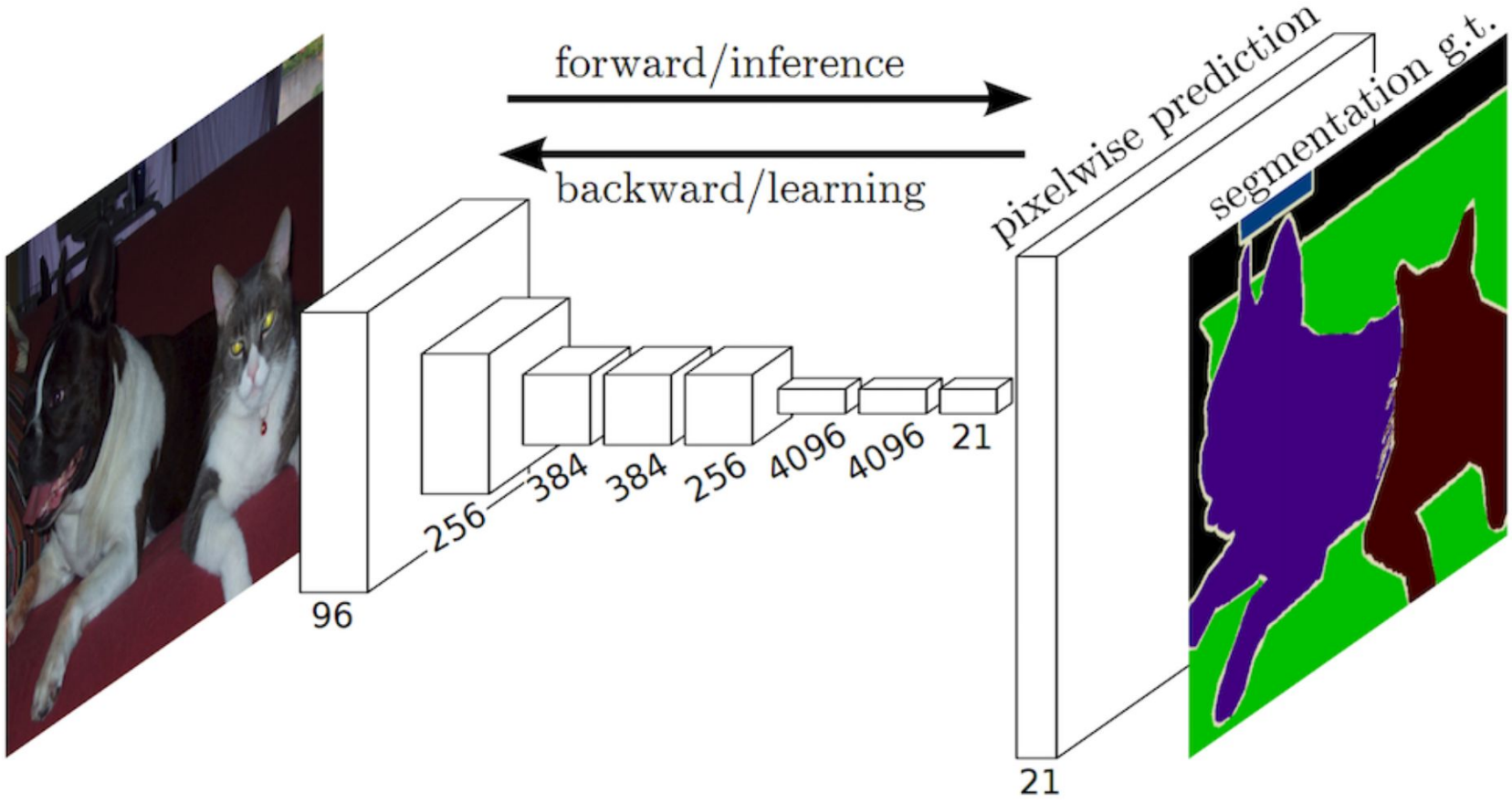
(Figur from Ross Girshick)

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

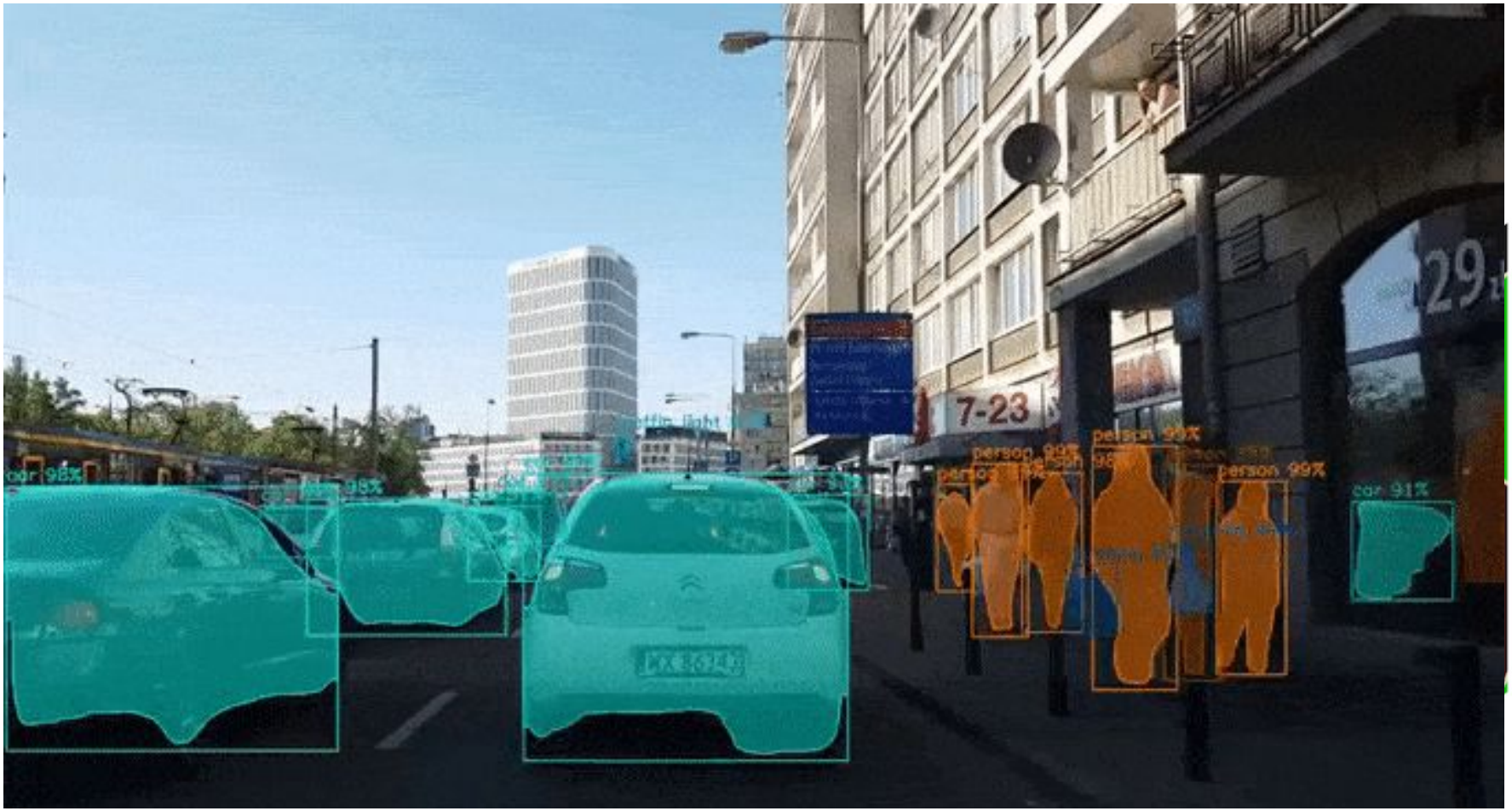
Object Segmentation

- Goal: Segment object regions and predict class labels for each region
- Can be formulated as pixel-wise classification
- CNN is pre-trained on a large-scale classification dataset (ImageNet)



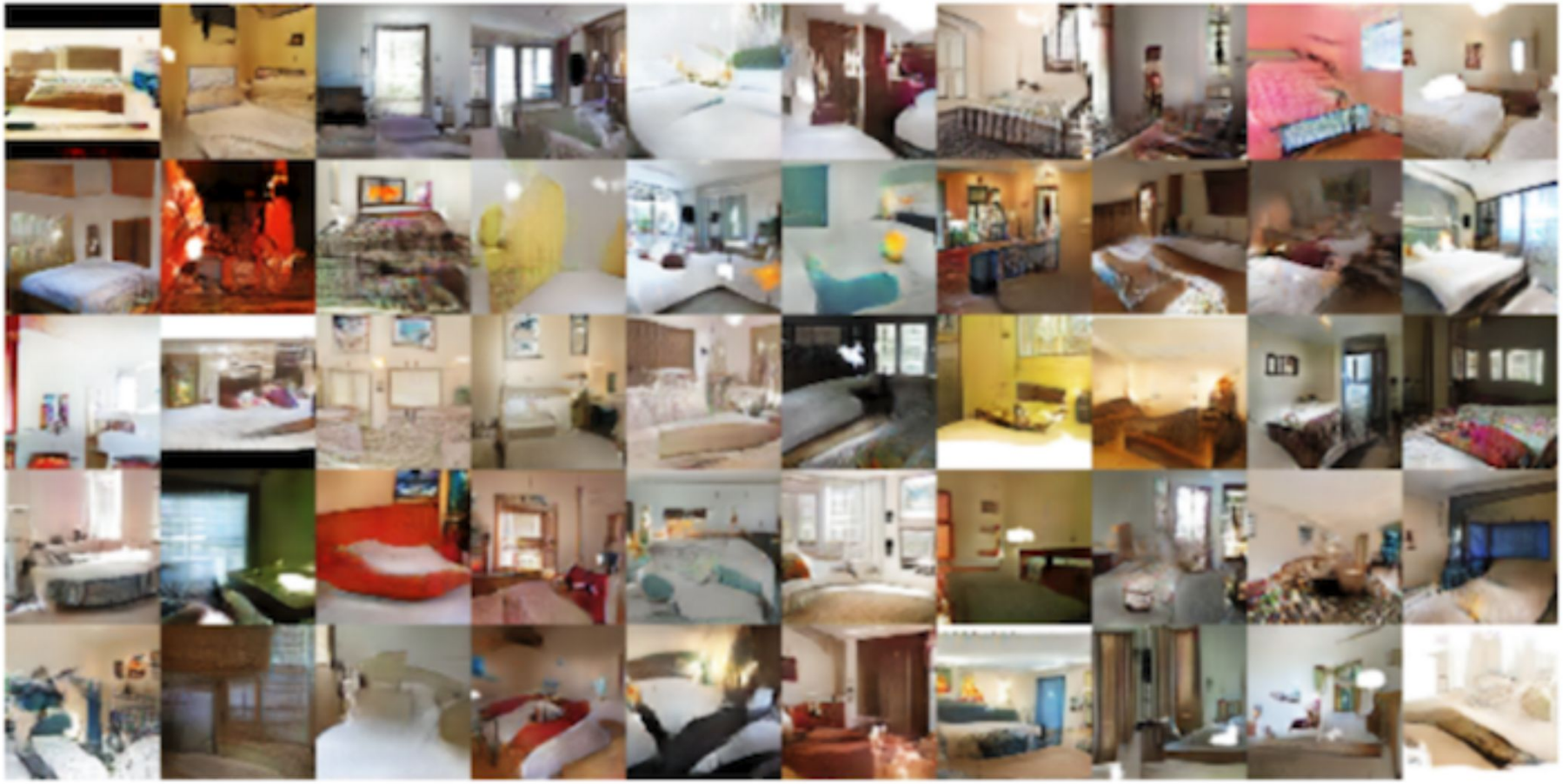
(Long et al, "Fully Convolutional Networks for Semantic Segmentation", CVPR, 2015.)

Object Segmentation



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

Image Generation: Generative Adversarial Network (Radford et al.)



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

Image Generation: Generative Adversarial Network (**Progressive GAN**)

[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]

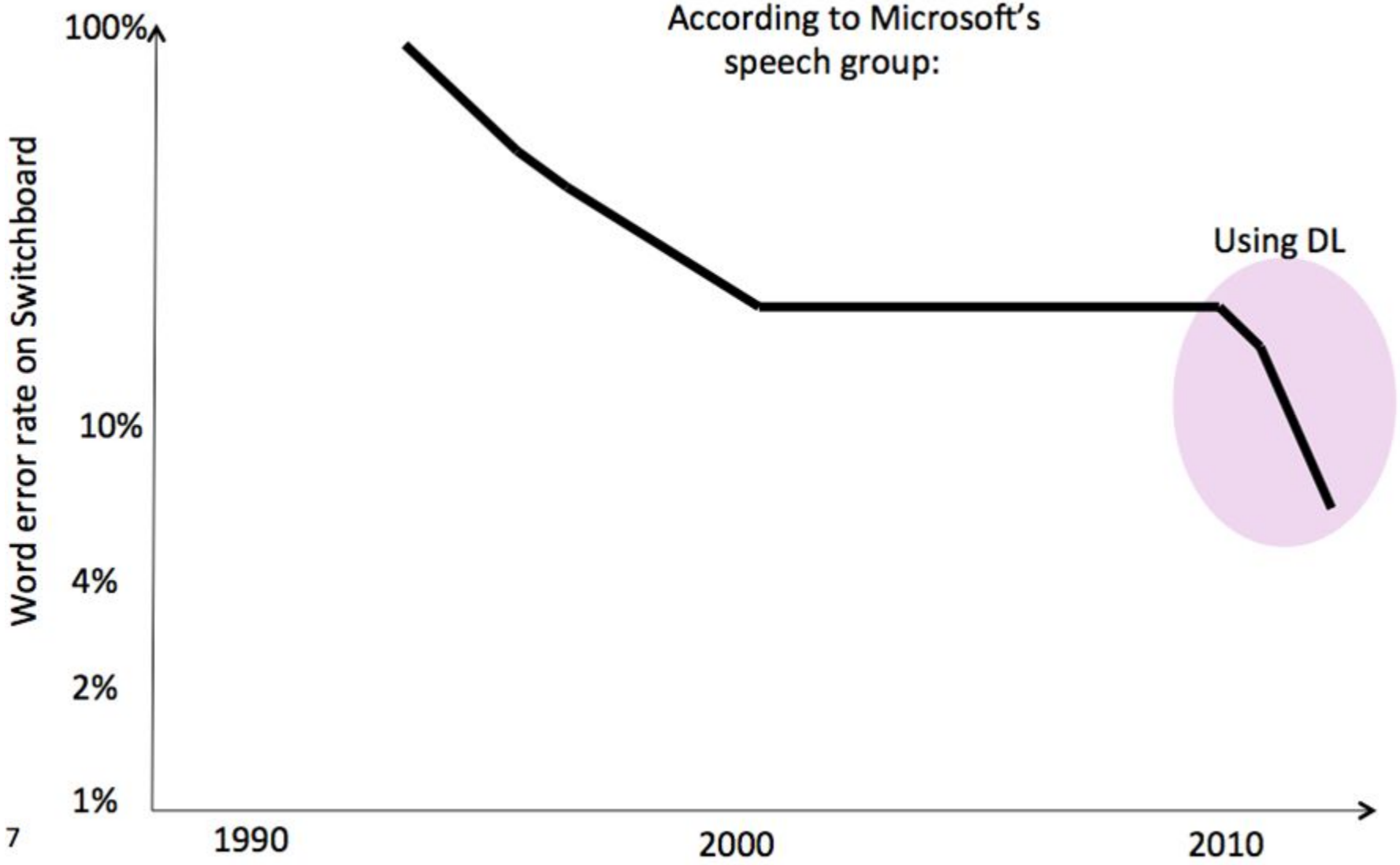
Image Caption Generation (Vinyals et al.)

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

Image Caption Generation (Vinyals et al.)

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

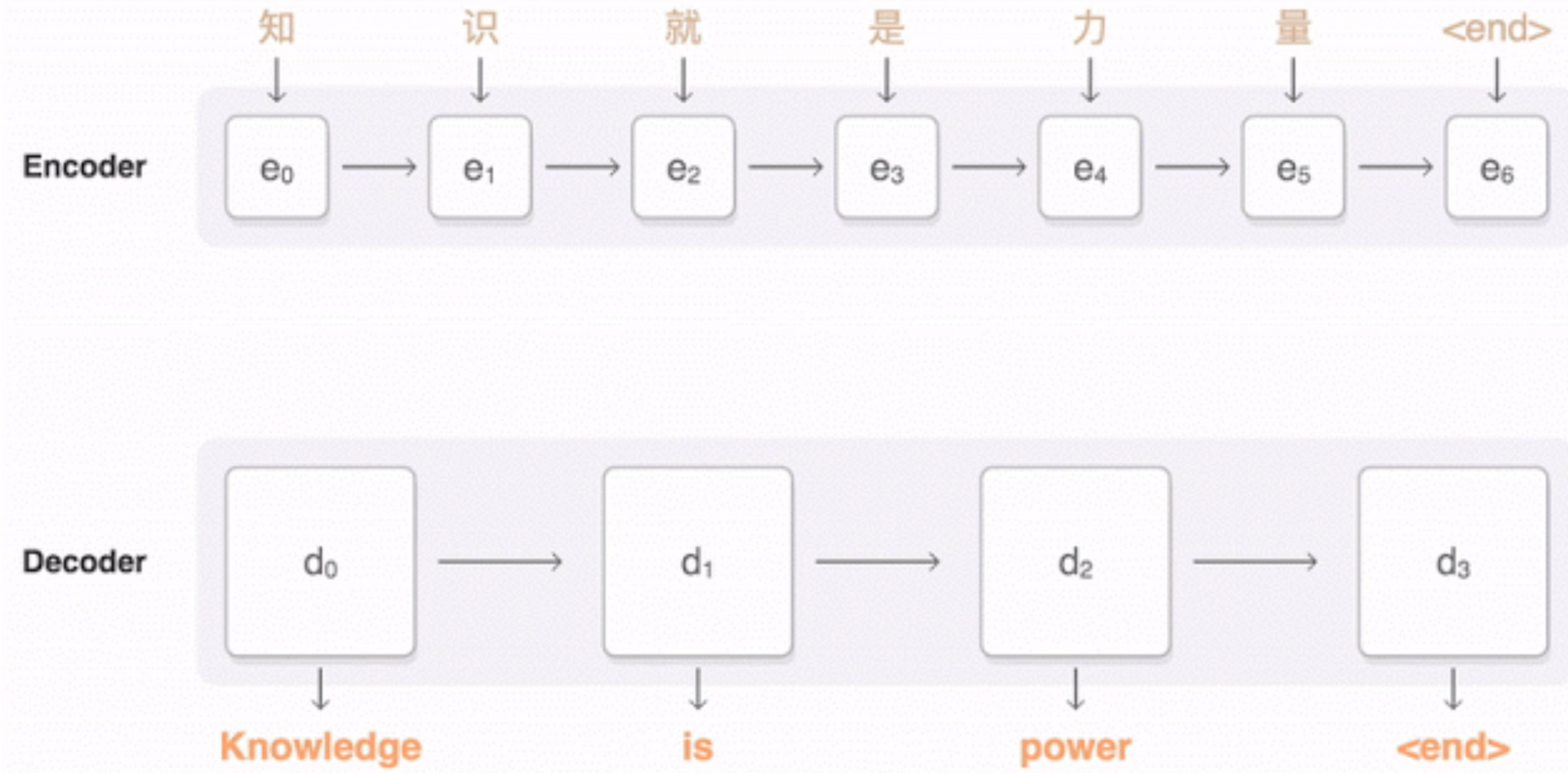
Speech recognition



Slide credit: Rob Fergus

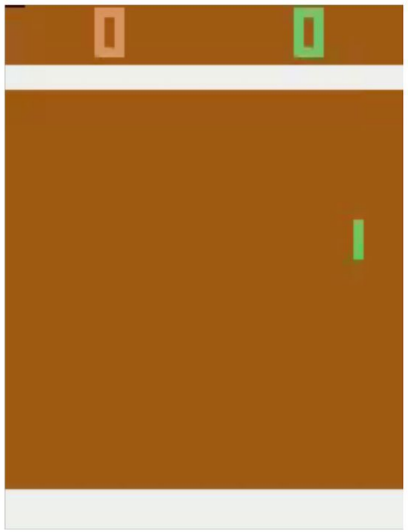
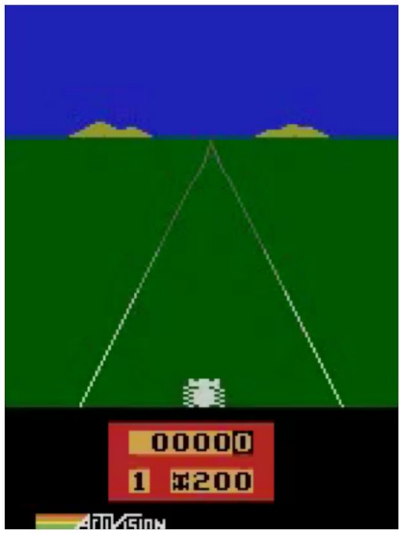
Machine Translation

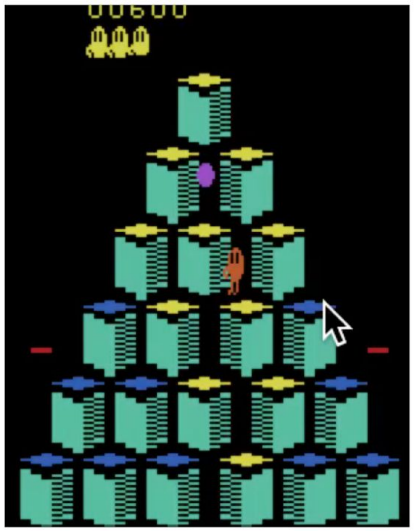
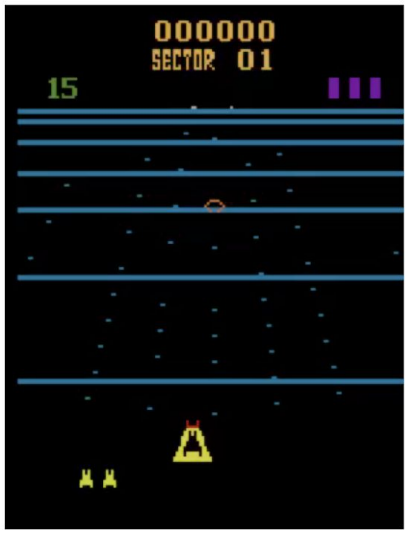
Google Neural Machine Translation (in production)



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 (Silver et al.)

- 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

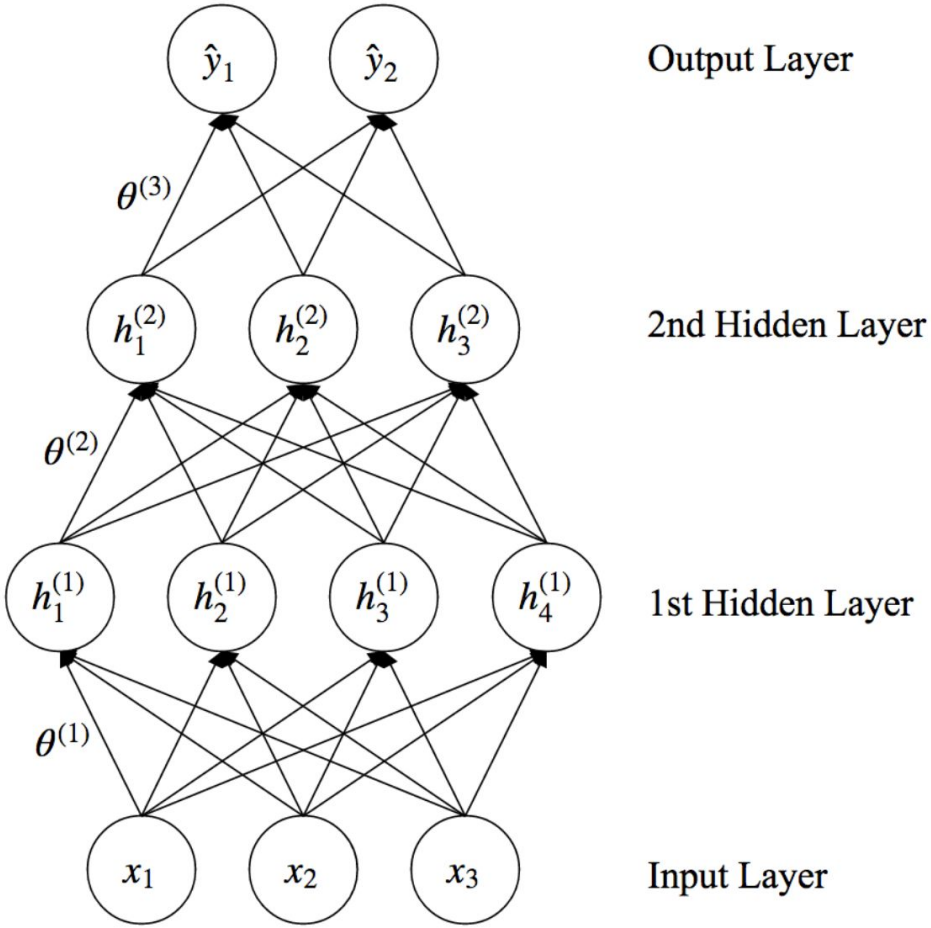
Levine et al., Learning Hand-Eye Coordination for Robotic Grasping. 2016
video: https://www.youtube.com/watch?v=cXaic_k80uM

slide credit: Pieter Abbeel

High-level Overview of Deep Neural Networks

Overview of Neural Networks

- Input Layer: provides input
- Hidden Layer: features extracted from input
- Output Layer: output of the network
- Parameters (or weights) for each layer



Slide credit: Junhyuk Oh, Jacob Abernethy

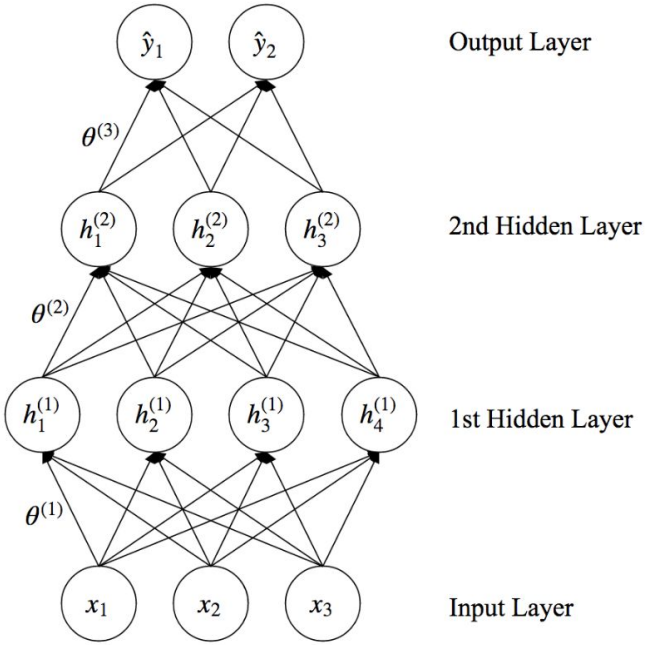
Overview of Neural Networks

- A **loss function** is defined over the *output units* and *desired outputs* (i.e., labels)

$$\mathcal{L}\left(\mathbf{y}, \hat{\mathbf{y}}\right) \text{ where } \hat{\mathbf{y}} = f(\mathbf{x}; \theta)$$

- The parameter of the network is trained to minimize the loss function based on gradient descent methods

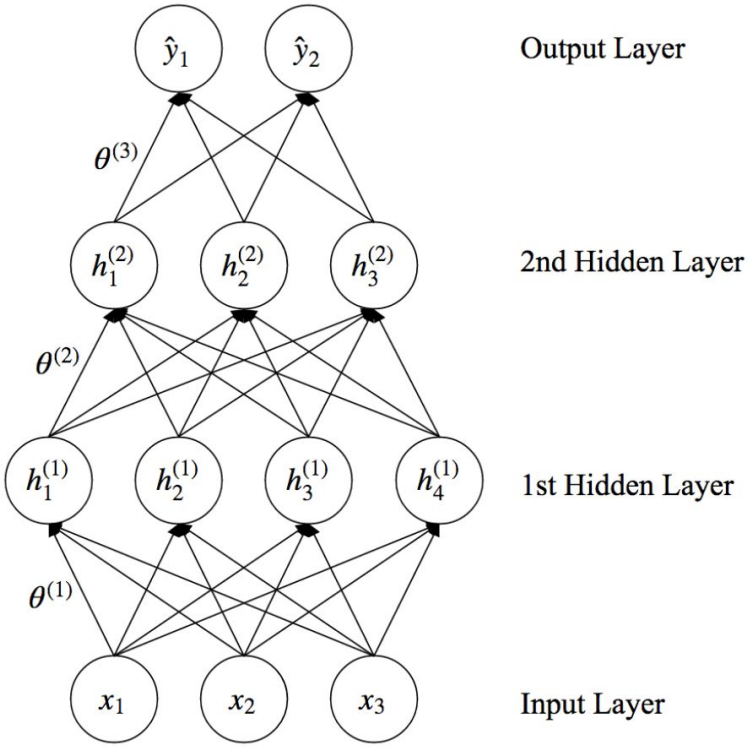
$$\min_{\theta} \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim \text{data}} \left[\mathcal{L}\left(\mathbf{y}, \hat{\mathbf{y}}\right)\right] \text{ where } \hat{\mathbf{y}} = f(\mathbf{x}; \theta)$$



Slide credit: Junhyuk Oh, Jacob Abernethy

Overview of Neural Networks

- Forward Propagation (inference): Compute $\hat{\mathbf{y}} = f(\mathbf{x}; \theta)$ (output given input)
- Backward Propagation (learning): Compute $\nabla_\theta \mathcal{L}$ (gradient of loss w.r.t. parameters)



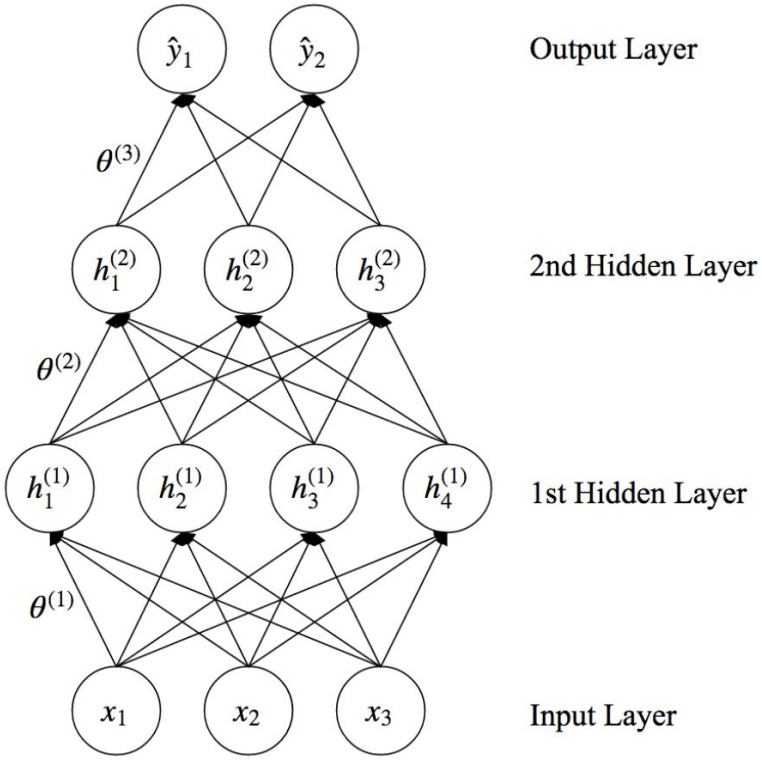
Slide credit: Junhyuk Oh, Jacob Abernethy

Forward Propagation

- The activation of each unit is computed based on **the previous layer** and **parameters (or weights)** associated with edges

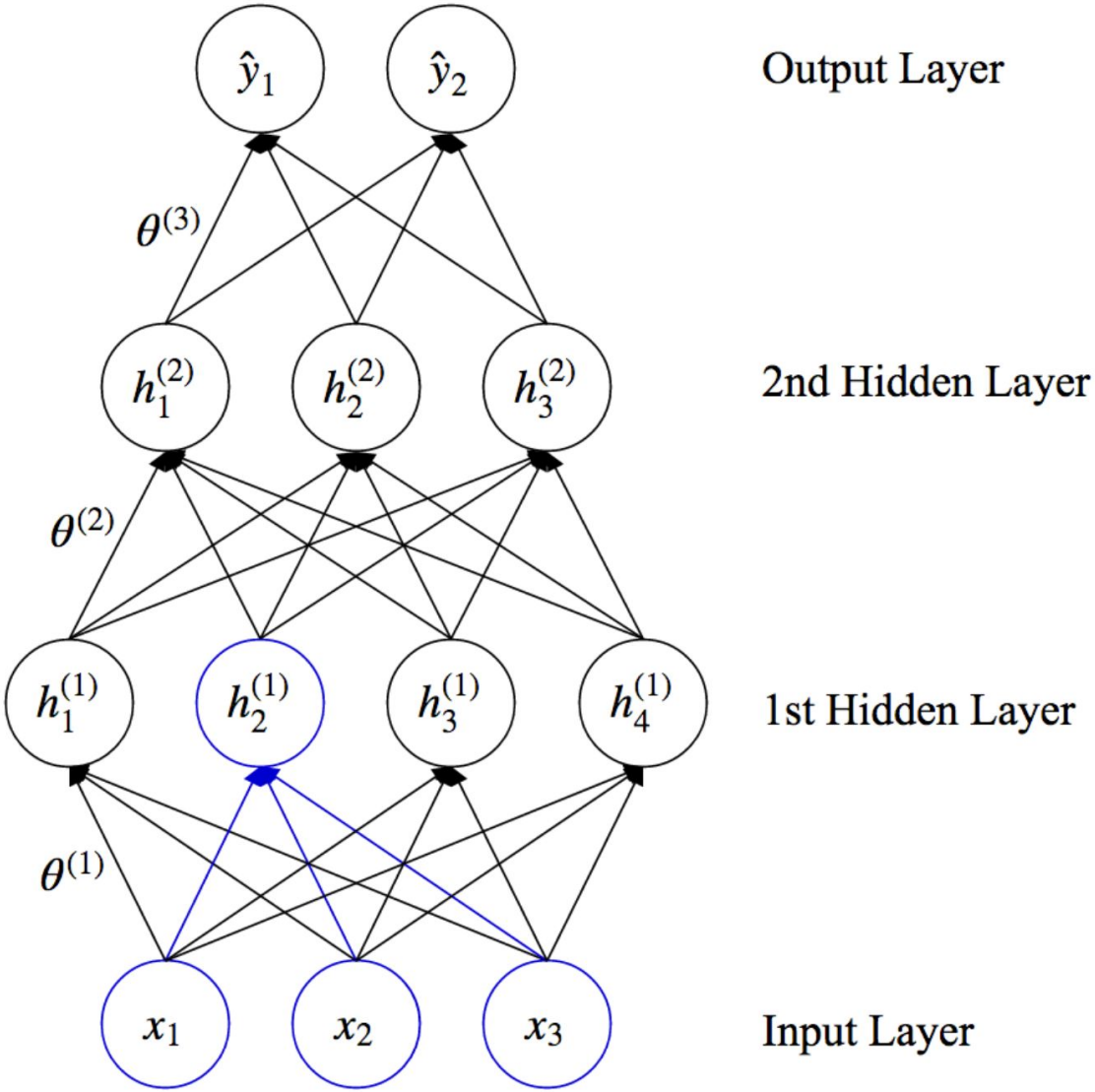
$$\underbrace{\mathbf{h}^{(l)}}_{l\text{-th layer}} = f^{(l)}\Big(\underbrace{\mathbf{h}^{(l-1)}}_{(l-1)\text{-th layer}} ; \underbrace{\theta^{(l)}}_{\text{weights}} \Big) \text{ where } \mathbf{h}^{(0)} \equiv \mathbf{x}, \mathbf{h}^{(L)} \equiv \hat{\mathbf{y}}$$

$$\hat{\mathbf{y}} = f(\mathbf{x}; \theta) = f^{(L)} \circ f^{(L-1)} \cdots f^{(2)} \circ f^{(1)} \left(\mathbf{x}; \theta^{(1)} \right)$$



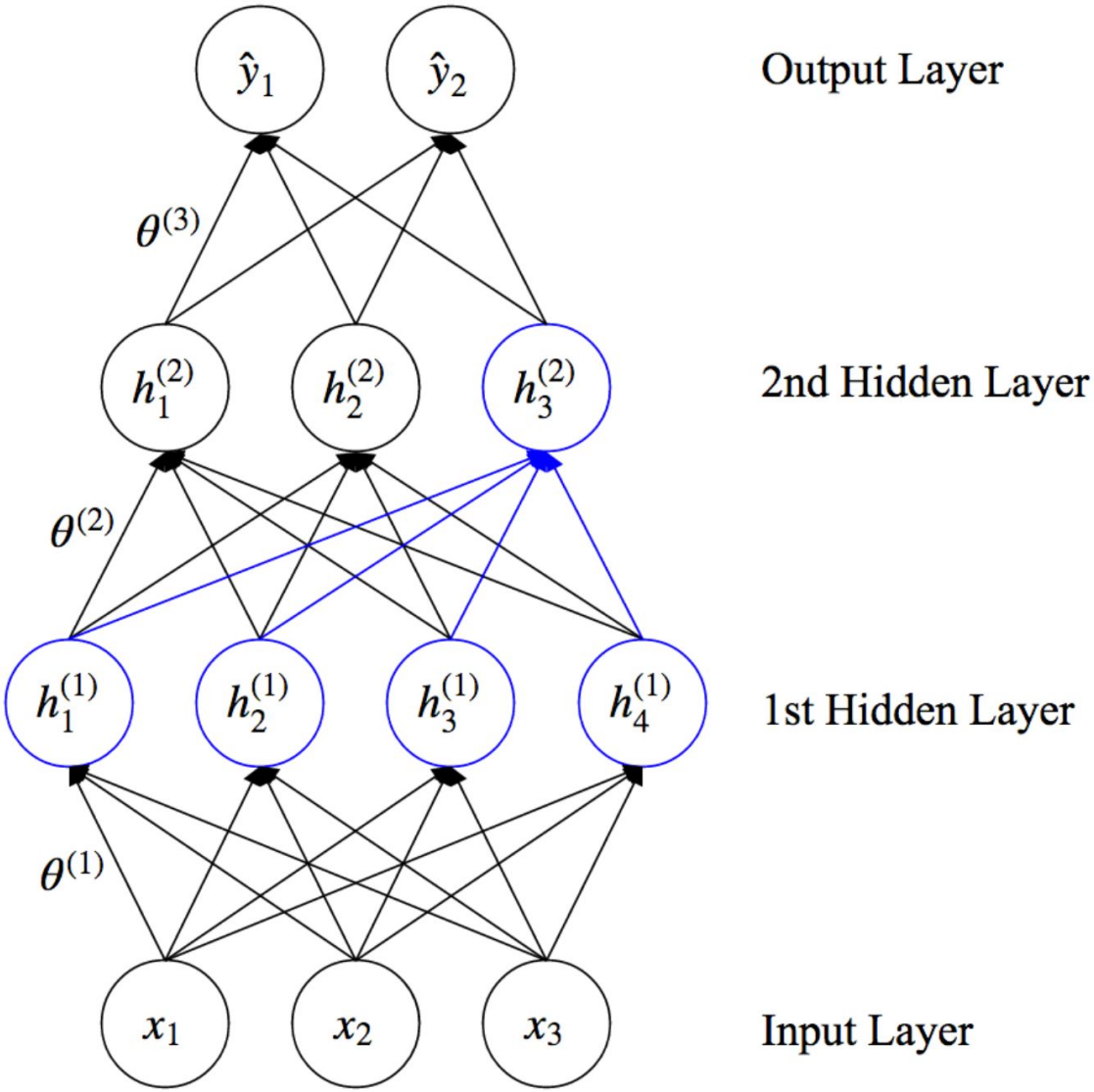
Slide credit: Junhyuk Oh, Jacob Abernethy

Forward Propagation



Slide credit: Junhyuk Oh, Jacob Abernethy

Forward Propagation



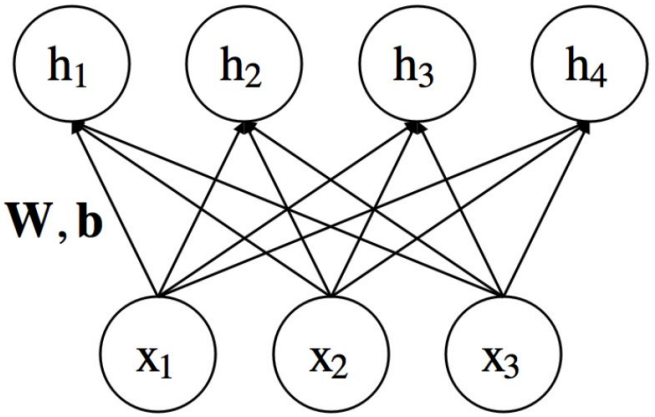
Slide credit: Junhyuk Oh, Jacob Abernethy

Types of Layers: Linear

$$h_i = \sum_j w_{ij} x_j + b_i$$

$$\mathbf{h} = \mathbf{W}\mathbf{x} + \mathbf{b}$$

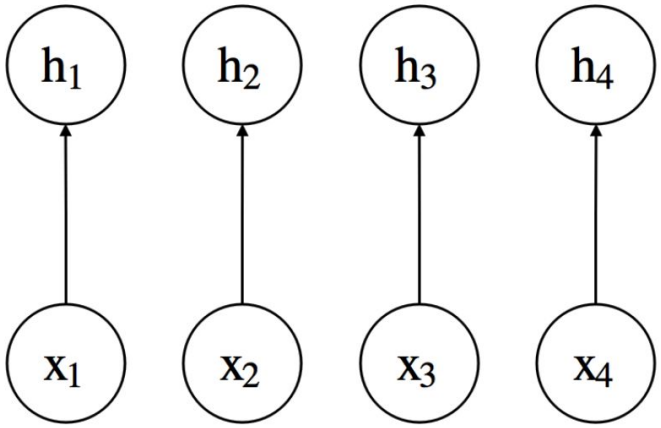
- $\mathbf{x} \in \mathbb{R}^m$: Input, $\mathbf{h} \in \mathbb{R}^n$: Output
- $\mathbf{W} \in \mathbb{R}^{n \times m}$: Weight, $\mathbf{b} \in \mathbb{R}^n$: Bias $\rightarrow$ parameter
- Often called "fully-connected layer"



Slide credit: Junhyuk Oh, Jacob Abernethy

Types of Layers: Non-linear Activation Function

- Applies a non-linear function to individual units.
- There is no weight.
- Allows neural networks to learn non-linear features.
- ex) Sigmoid, Hyperbolic Tangent, Rectified Linear Function



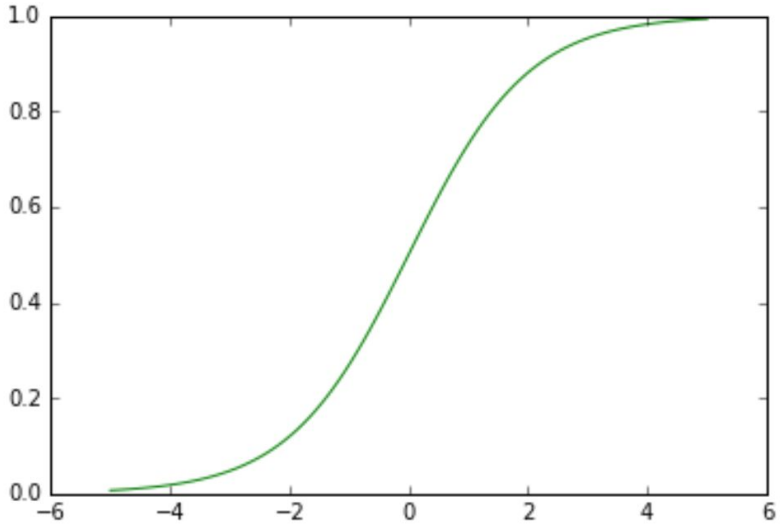
Slide credit: Junhyuk Oh, Jacob Abernethy

Non-linear Activation: Sigmoid

$$h_i = \sigma(x_i) = \frac{1}{1 + \exp(-x_i)}$$
$$\mathbf{h} = \sigma(\mathbf{x})$$

In [3]:
```
xx = np.linspace(-5, 5, 100)
plt.plot(xx, 1/(1 + np.exp(-1 * xx)), '-g')
```

Out[3]:
```
[<matplotlib.lines.Line2D at 0x1054fff10>]
```



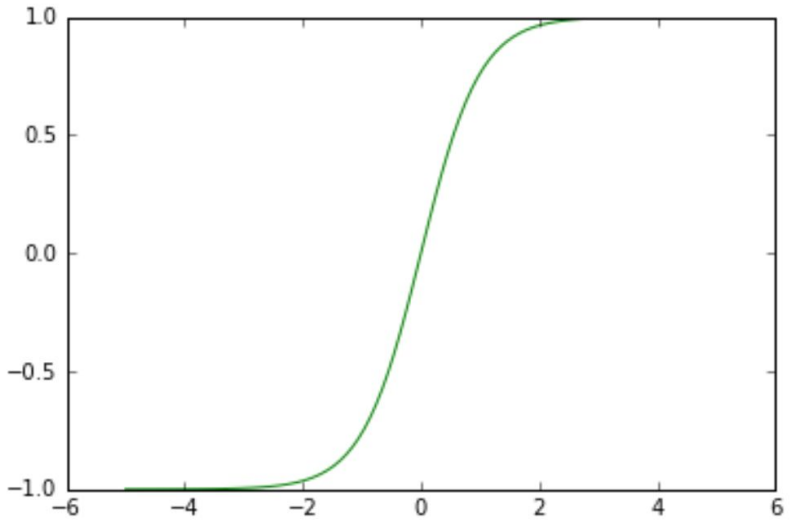
Slide credit: Junhyuk Oh, Jacob Abernethy

Non-linear Activation: Hyperbolic Tangent (Tanh)

$$h_i = \tanh(x_i) = \frac{\exp(x_i) - \exp(-x_i)}{\exp(x_i) + \exp(-x_i)}$$
$$\mathbf{h} = \tanh(\mathbf{x})$$

In [4]:
```
xx = np.linspace(-5, 5, 100)
plt.plot(xx, (np.exp(xx) - np.exp(-1 * xx))/(np.exp(xx) + np.exp(-1 * xx)), '-g')
```

Out[4]: [<matplotlib.lines.Line2D at 0x10568f850>]



Slide credit: Junhyuk Oh, Jacob Abernethy

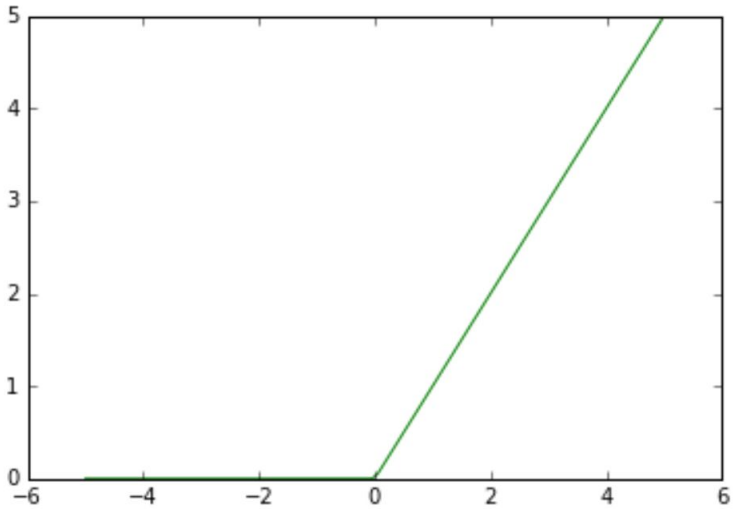
Non-linear Activation: Rectified Linear (ReLU)

$$h_i = \mathrm{ReLU}(x_i) = \max(x_i, 0)$$
$$\mathbf{h} = \mathrm{ReLU}(\mathbf{x})$$

- Easier to optimize

In [5]:
```
xx = np.linspace(-5, 5, 100)
plt.plot(xx, xx * (xx > 0).astype(np.int), '-g')
```

Out[5]:
```
[<matplotlib.lines.Line2D at 0x105872490>]
```

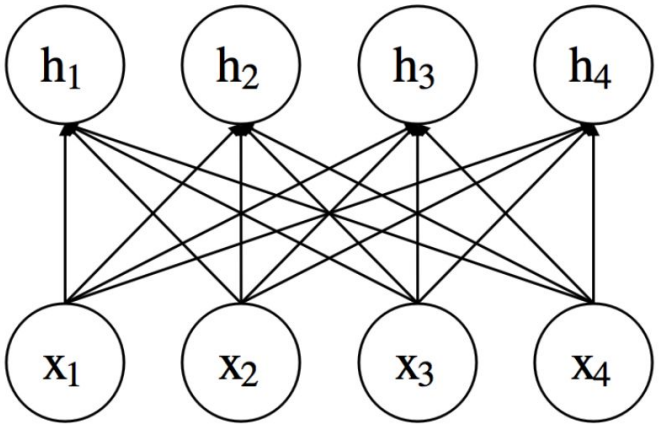


Slide credit: Junhyuk Oh, Jacob Abernethy

Types of Layers: Softmax

$$h_i = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$$
$$\mathbf{h} = \mathrm{Softmax}(\mathbf{x})$$

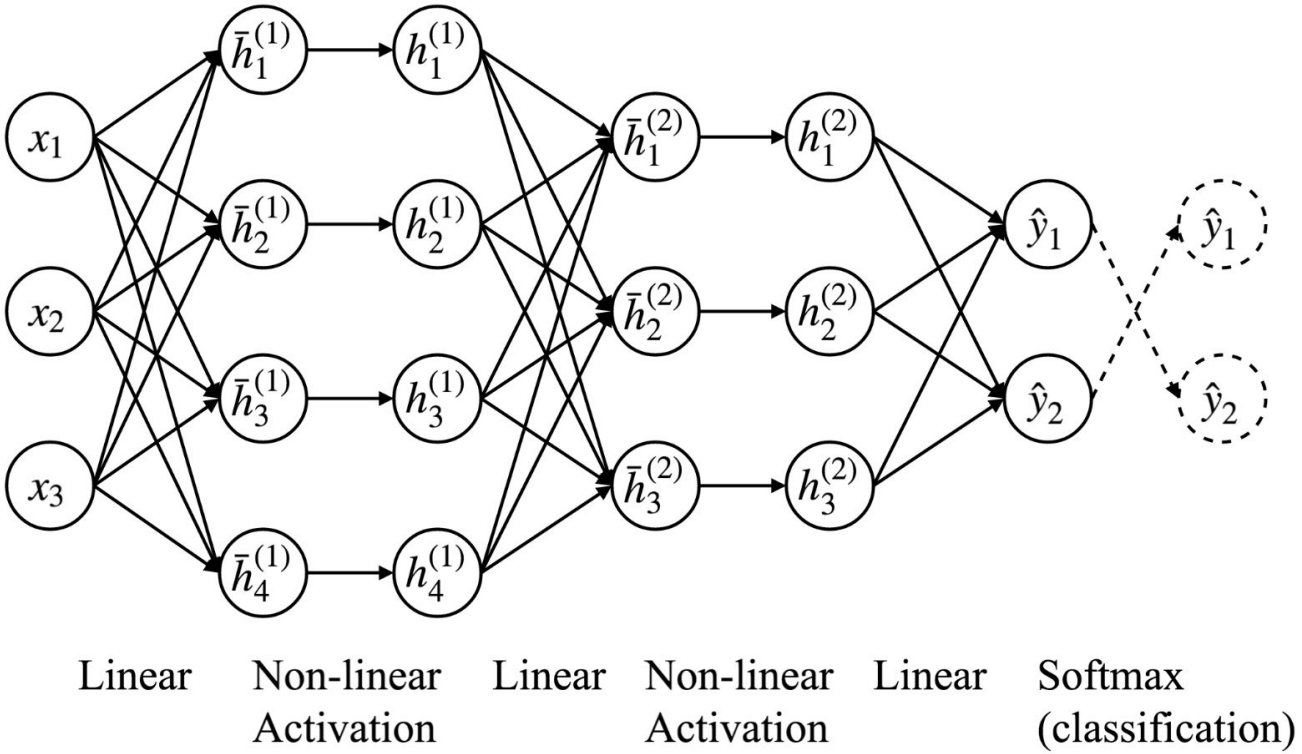
- Note: $h_i \geq 0$ and $\sum_i h_i = 1$
- Useful for generating a multinomial distribution (classification)



Slide credit: Junhyuk Oh, Jacob Abernethy

Multi-layer Neural Network

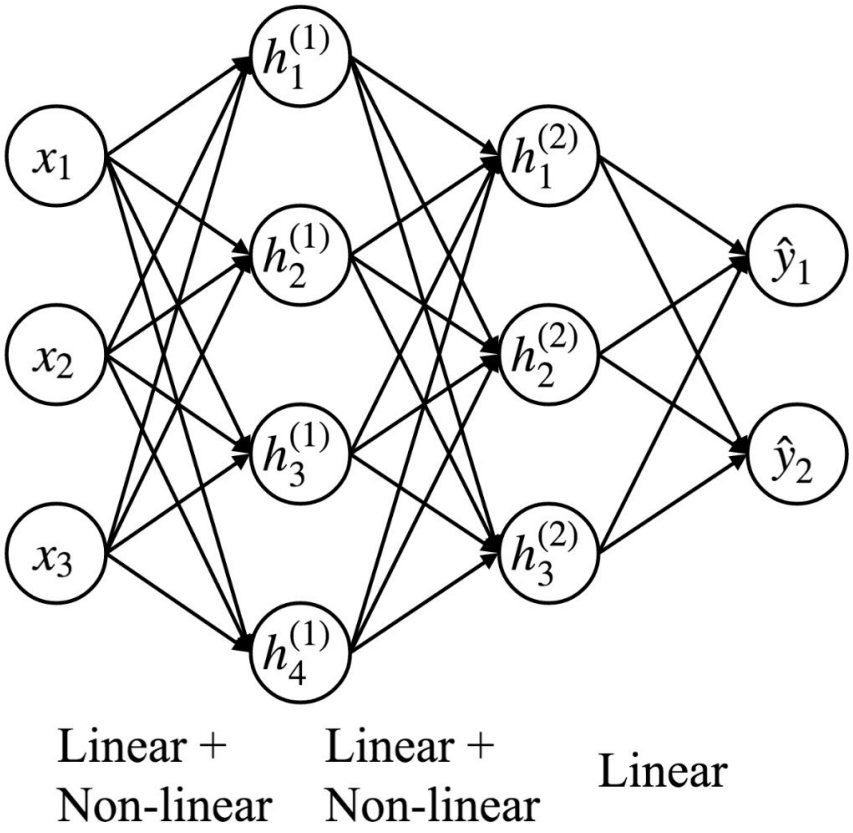
- Consists of multiple (linear + non-linear activation) layers.
- Each layer learns non-linear features from its previous layer.
- Often called Multi-Layer Perceptron (MLP).
- 2-layer MLP with infinite number of hidden units can approximate any functions.



Slide credit: Junhyuk Oh, Jacob Abernethy

Multi-layer Neural Network

- Simplified illustration that only shows edges with weights.
- We assume that each layer is followed by a non-linear activation function (except for the output layer).



Slide credit: Junhyuk Oh, Jacob Abernethy

Training Neural Networks

- Repeat until convergence
  - $(\mathbf{x}, \mathbf{y}) \leftarrow$ Sample an example (or a mini-batch) from data
  - $\hat{\mathbf{y}} \leftarrow f(\mathbf{x}; \theta)$ Forward propagation
  - Compute $\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})$
  - $\nabla_\theta \mathcal{L} \leftarrow$ Backward propagation
  - Update weights using (stochastic) gradient descent
    - $\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}$

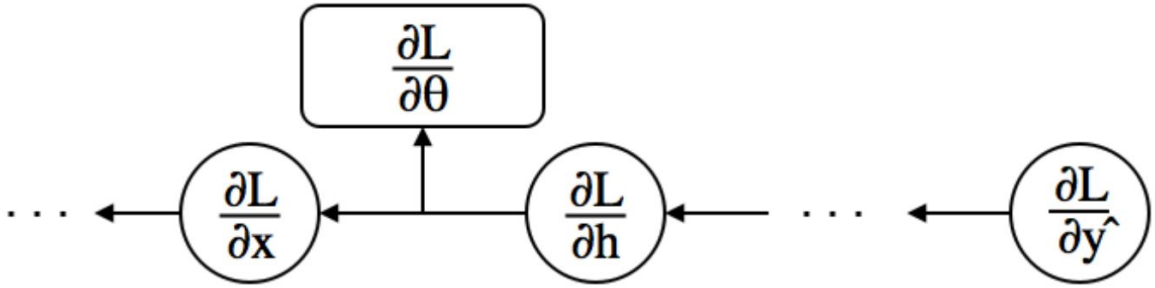
Slide credit: Junhyuk Oh, Jacob Abernethy

Idea of Back-Propagation

- Assuming that $\frac{\partial \mathcal{L}}{\partial h}$ is given, use the **chain rule** to compute the gradients

$$\frac{\partial \mathcal{L}}{\partial \theta} = \underbrace{\frac{\partial \mathcal{L}}{\partial h}}_{\text{given}} \underbrace{\frac{\partial h}{\partial \theta}}_{\text{easy}}, \quad \frac{\partial \mathcal{L}}{\partial x} = \underbrace{\frac{\partial \mathcal{L}}{\partial h}}_{\text{given}} \underbrace{\frac{\partial h}{\partial x}}_{\text{easy}}$$

- We need only $\frac{\partial \mathcal{L}}{\partial \theta}$ for gradient descent. Why compute $\frac{\partial \mathcal{L}}{\partial x}$?
  - The previous layer needs it because x is the output of the previous layer.



Applications

Next class

- Deep Neural Network Basics
- Backpropagation
- Optimization and Regularization of Neural Networks

Reminder

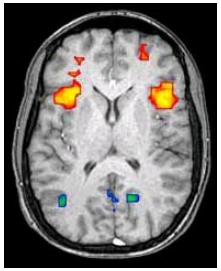
- Check syllabus at Canvas

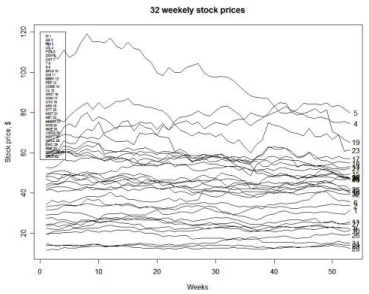
- For all questions, please use Piazza (linked to Canvas)

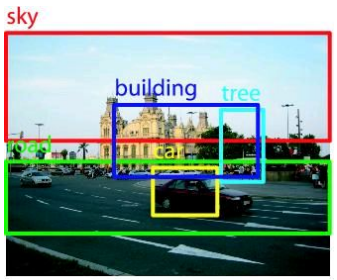
Questions?

Examples of ML applications

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

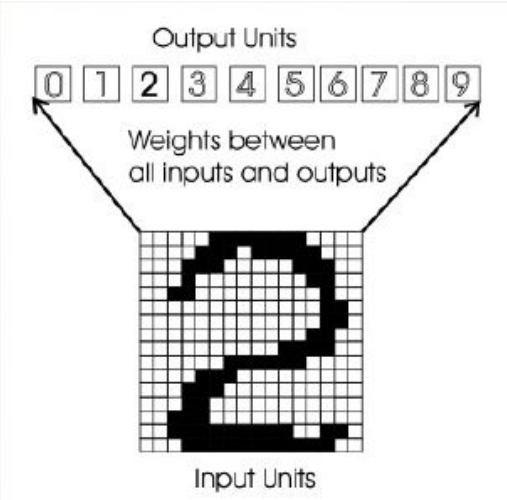






ML application: Computer Vision

- Handwritten digit recognition
  - LeCun et al., 1989



- Face recognition
  - Viola & Jones face detector (2001)

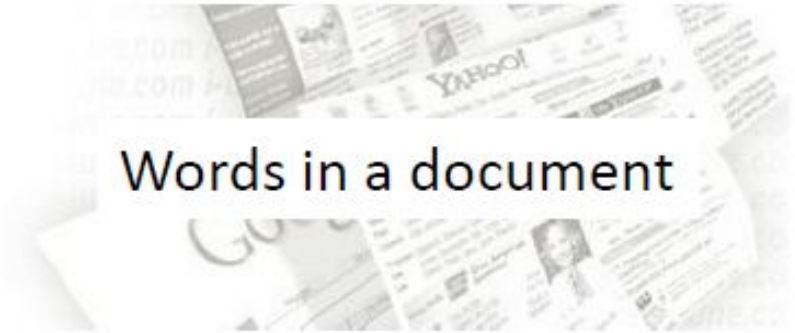
ML applications: speech recognition

- Voice search (e.g., Google)

- Speech transcription
  - http://www.youtube.com/watch?v=W3DhnpLIKCQ

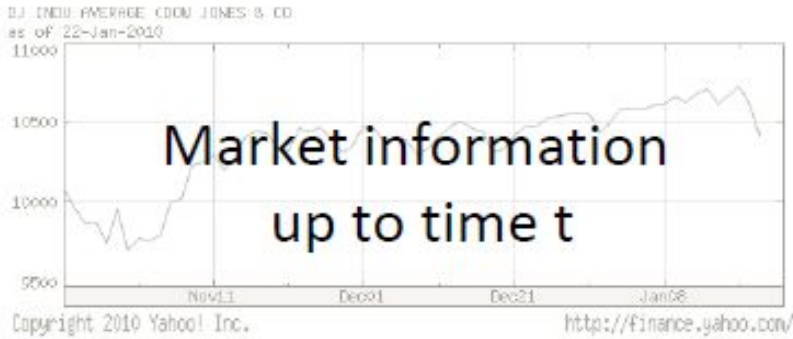
Supervised Learning

Feature Space $\mathcal{X}$



Label Space $\mathcal{Y}$

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



Share Price
"$ 24.50"

Task: Given $X \in \mathcal{X}$, predict $Y \in \mathcal{Y}$.

Slide credit: Aarti Singh

Supervised Learning - Classification

Feature Space $\mathcal{X}$

Label Space $\mathcal{Y}$

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

Discrete Labels

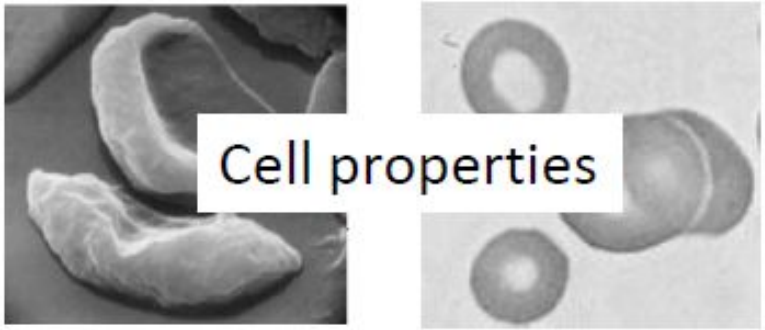
Slide credit: Aarti Singh

Supervised Learning - Classification

Feature Space $\mathcal{X}$

Label Space $\mathcal{Y}$

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

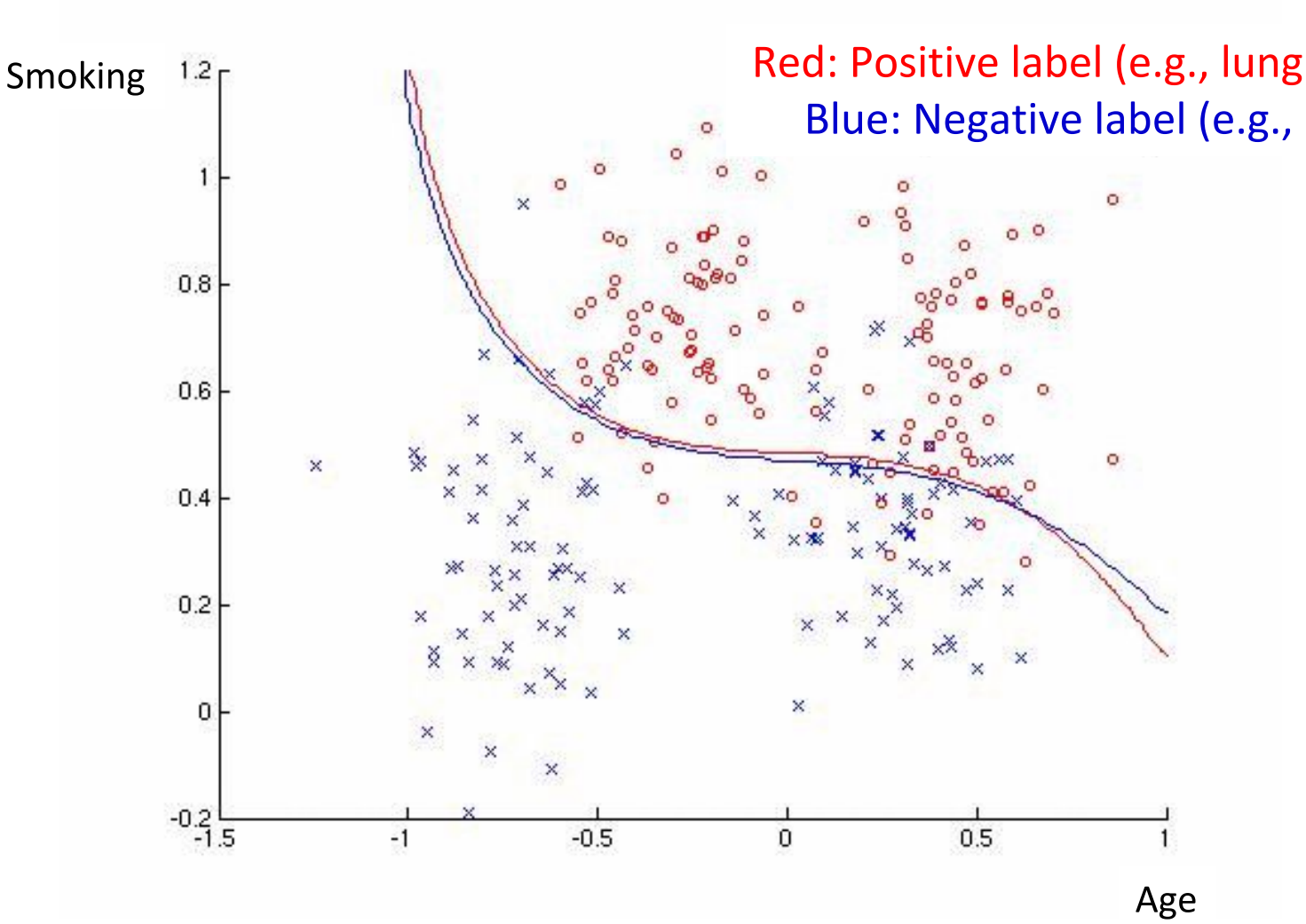


"Anemic cell"
"Healthy cell"

Discrete Labels

Slide credit: Aarti Singh

Supervised Learning - Classification

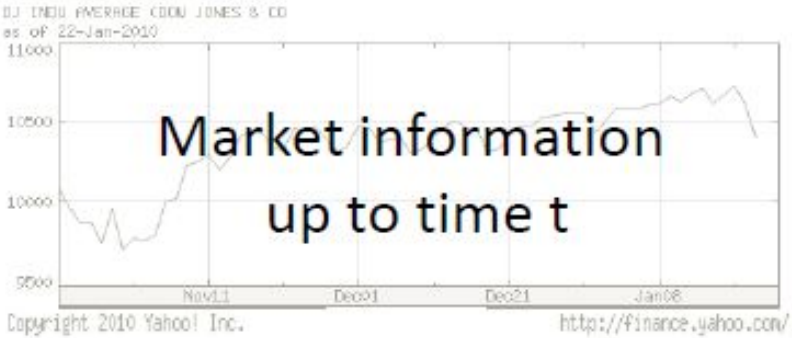


"Learning decision boundaries"

Supervised Learning - Regression

Feature Space $\mathcal{X}$

Label Space $\mathcal{Y}$



Share Price
"$ 24.50"

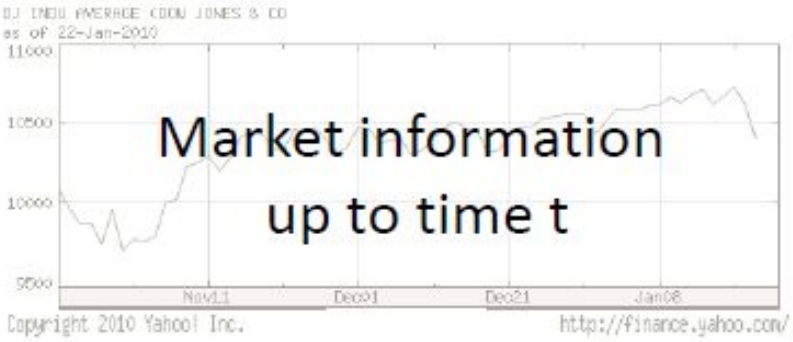
Continuous Labels

Slide credit: Aarti Singh

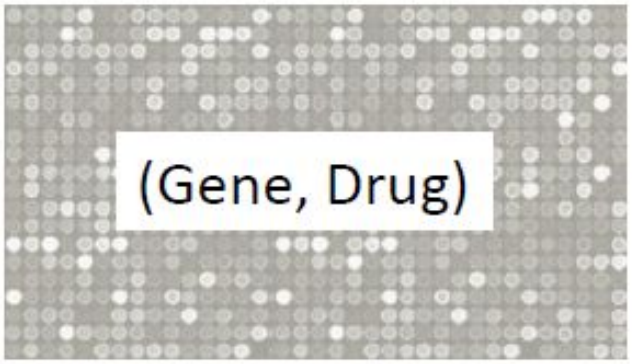
Supervised Learning - Regression

Feature Space $\mathcal{X}$

Label Space $\mathcal{Y}$



Share Price
"$ 24.50"

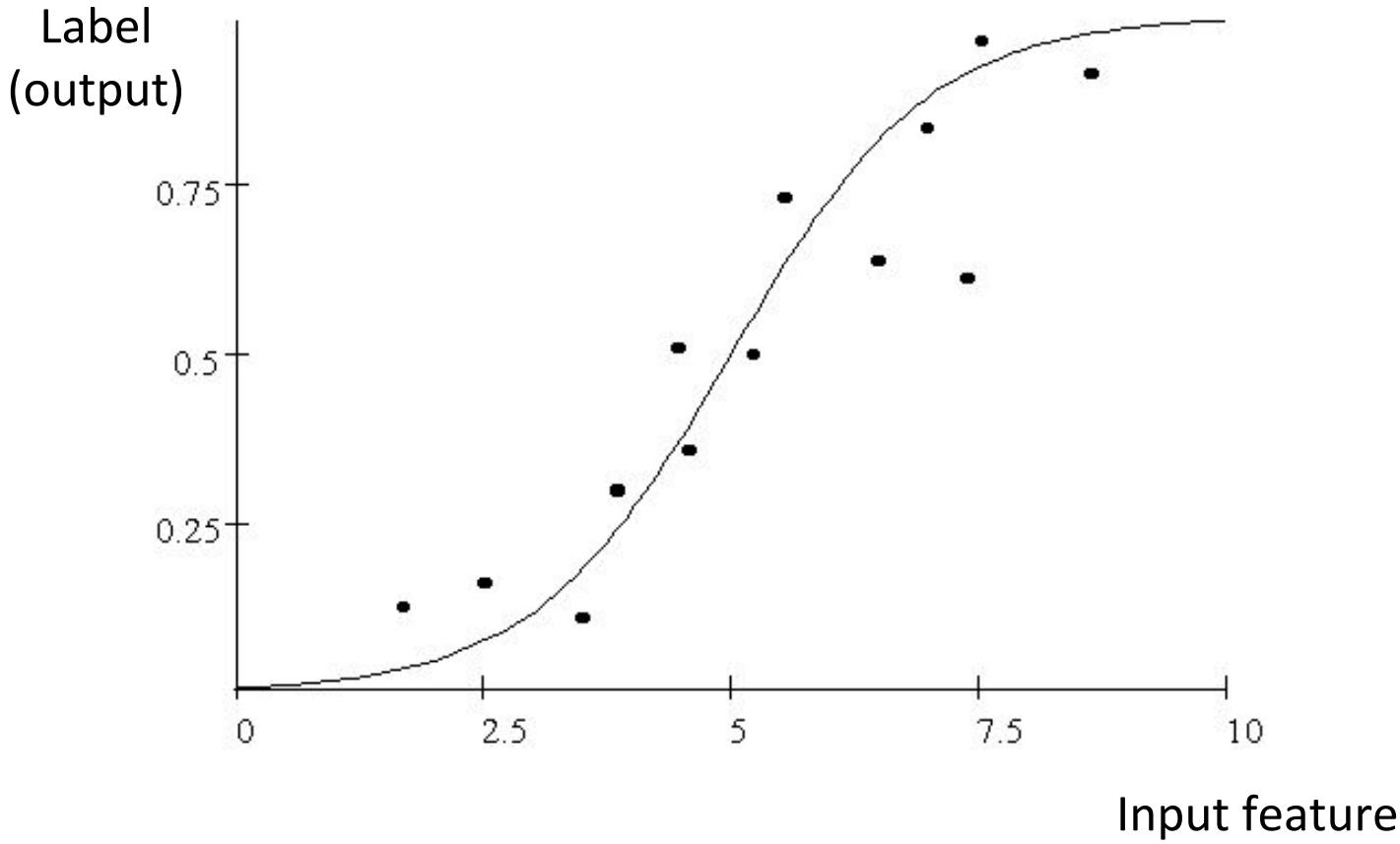


Expression level
"0.01"

Continuous Labels

Slide credit: Aarti Singh

Supervised Learning - Regression



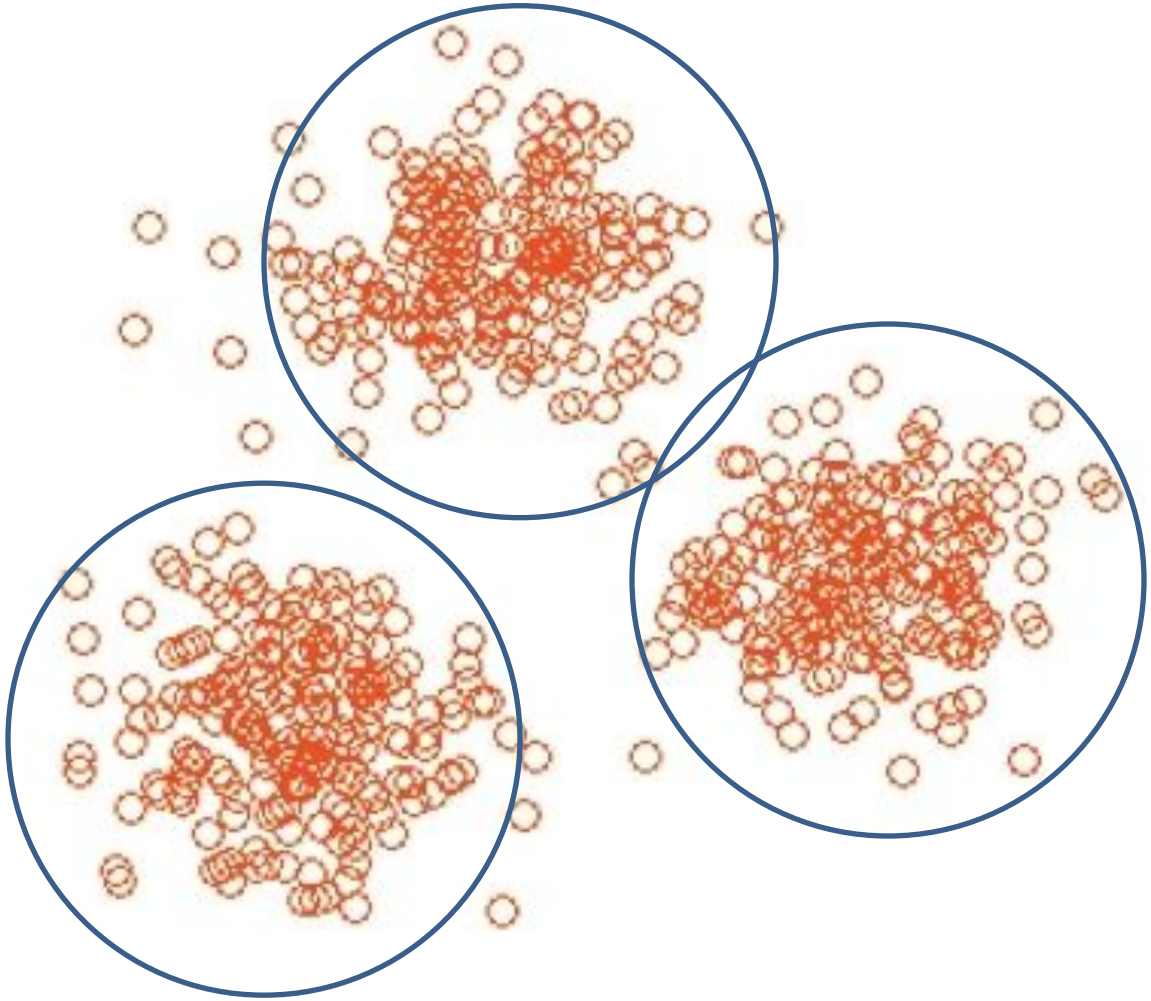
"Learning regression function f(X)"

Unsupervised Learning

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

Unsupervised Learning – Clustering

- "Grouping into similar examples"



Unsupervised Learning – Clustering

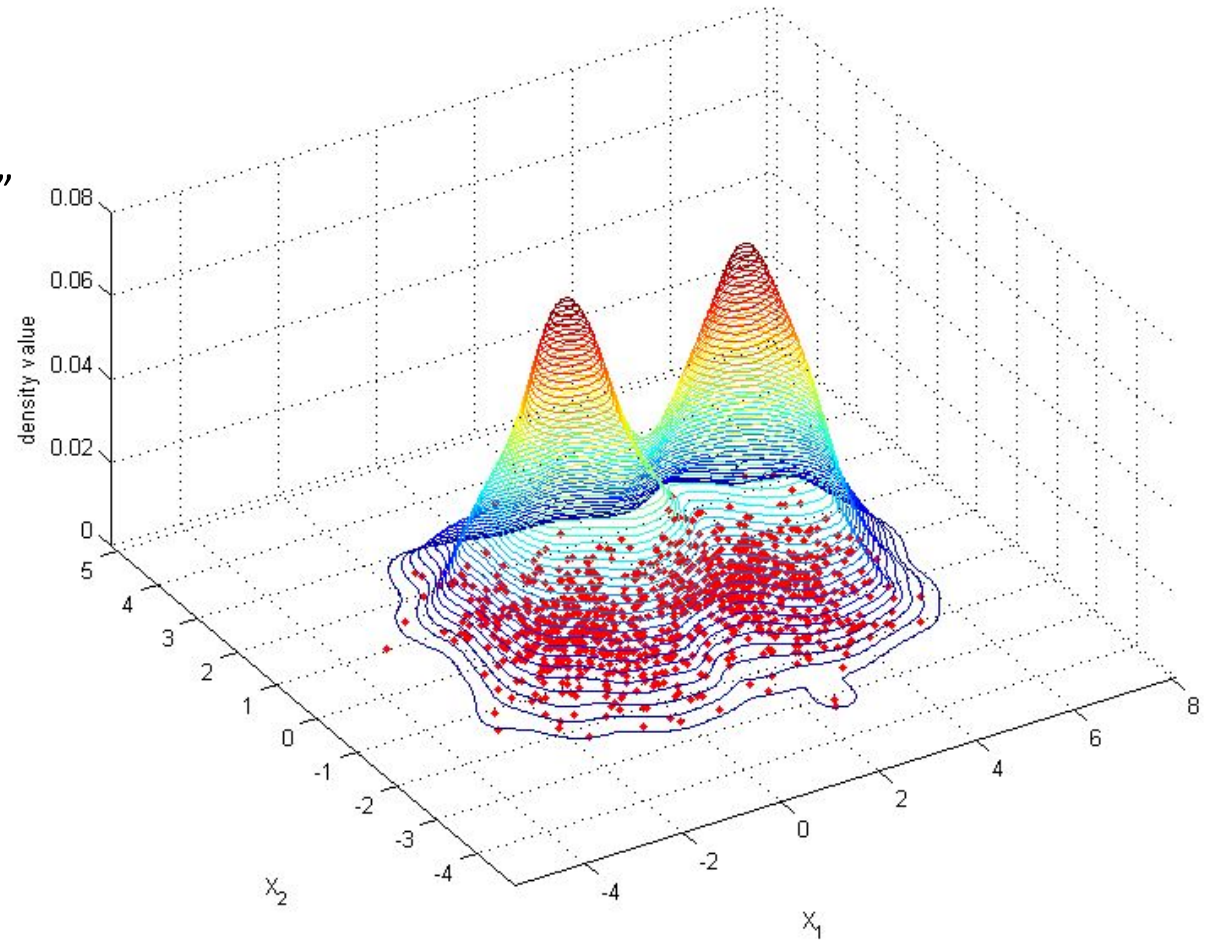
Group similar things e.g. images

[Goldberger et al.]

Slide credit: Aarti Singh

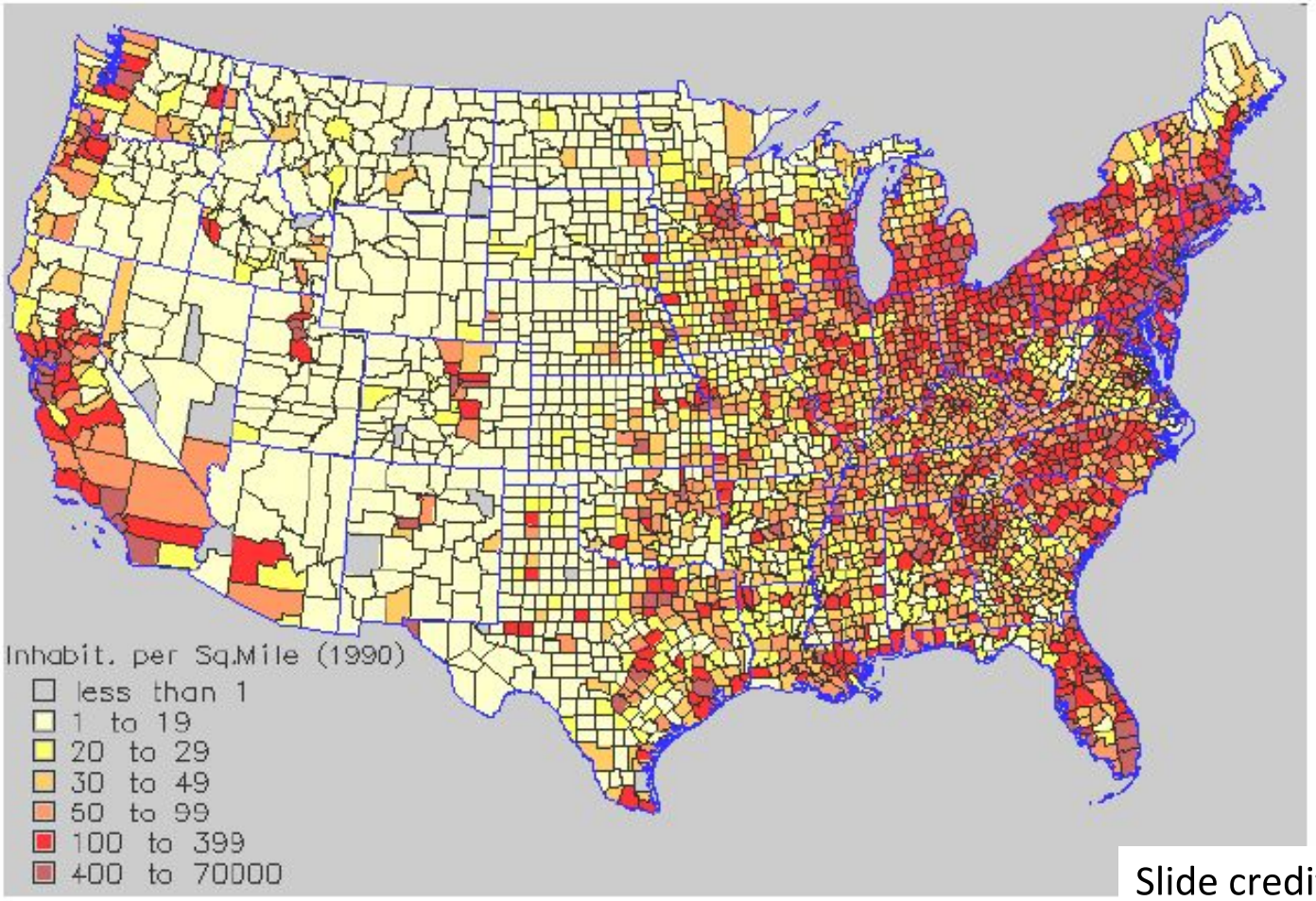
Unsupervised Learning – Density estimation

P(X1, X2)
"Probability"



Unsupervised Learning – Density estimation

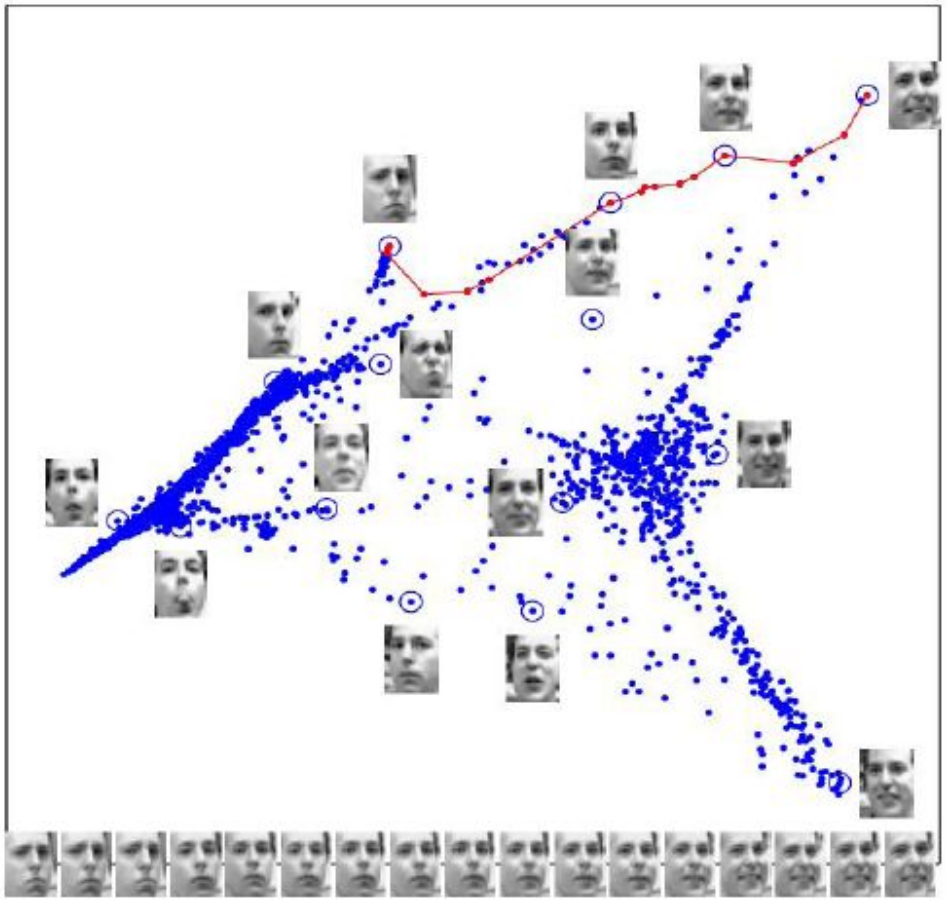
Population density



Slide credit: Aarti Singh

Unsupervised Learning-Embedding and Dimensionality reduction

- E.g., Reducing pixel images (several thousand pixels) into low dimensional coordinates



[Saul and Roweis, 03]

Reinforcement Learning

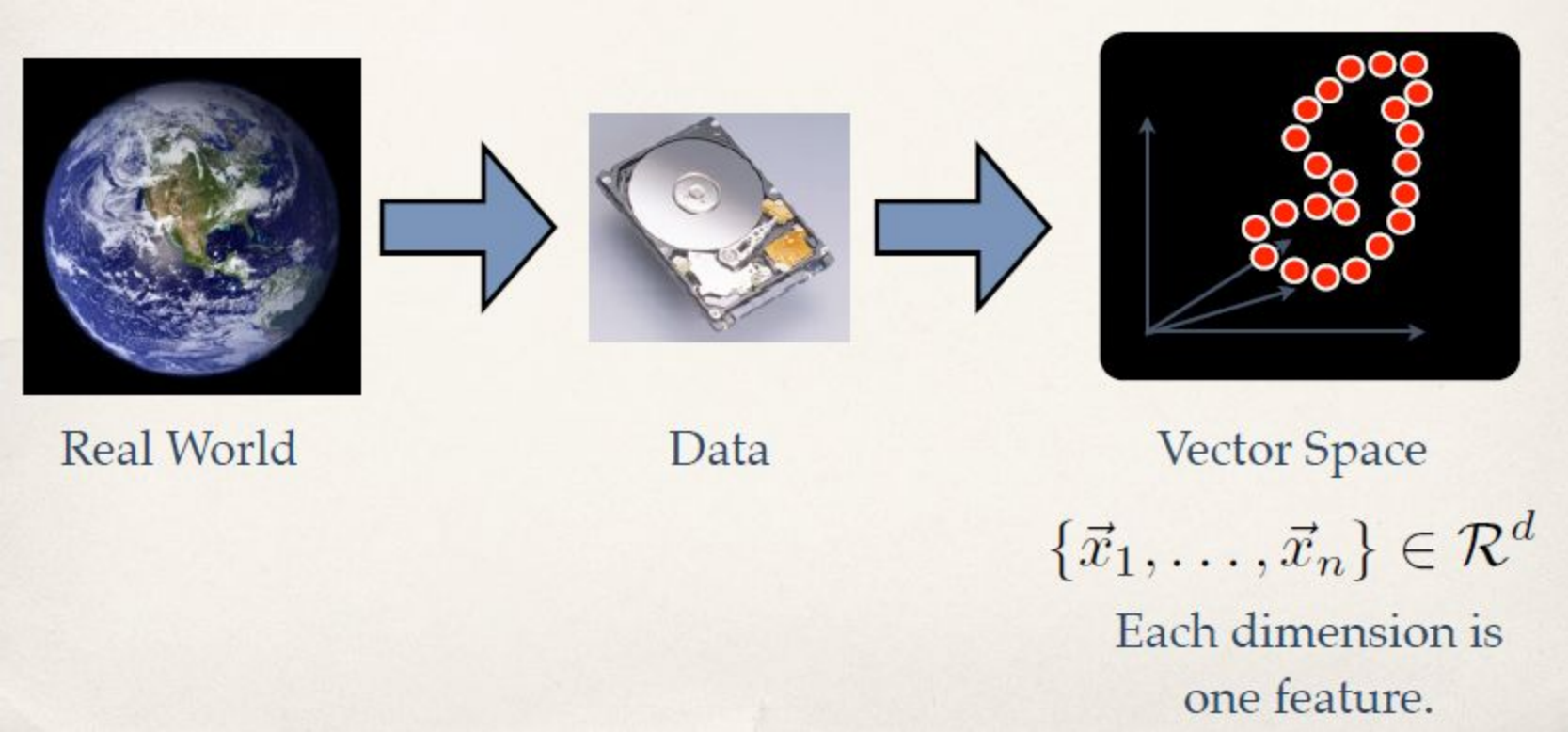
- Setting
  - Given sequence of states X and "rewards" (e.g., delayed labels)
  - Agent has to take actions A for each time step
- Goal:
  - How to "learn to act" or "make decisions" to maximize the sum of future rewards
- Example: Robot navigation task
  - Input: Dynamical environment + sensor input
  - 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 Extraction

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



Slide credit: Kilian Weinberger

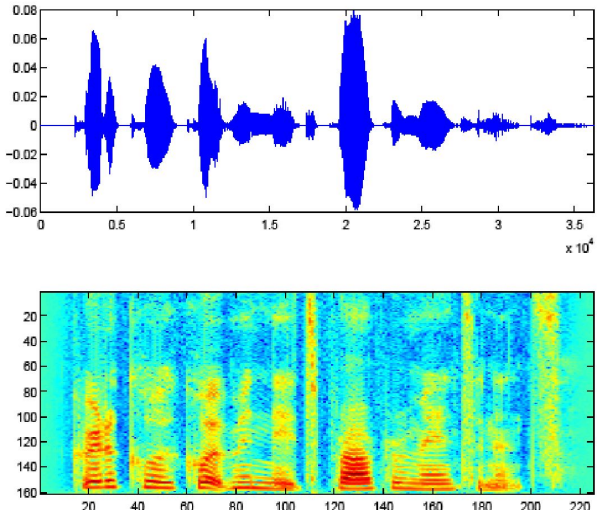
Examples of features: Housing data

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

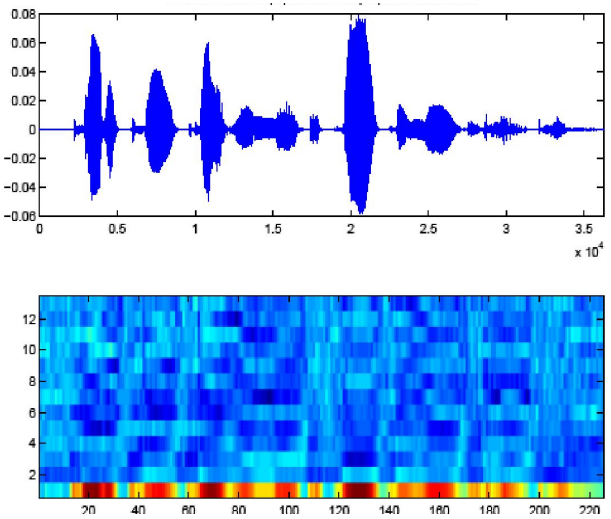
Examples of features: Recognizing handwritten-digits

- 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
  - ….

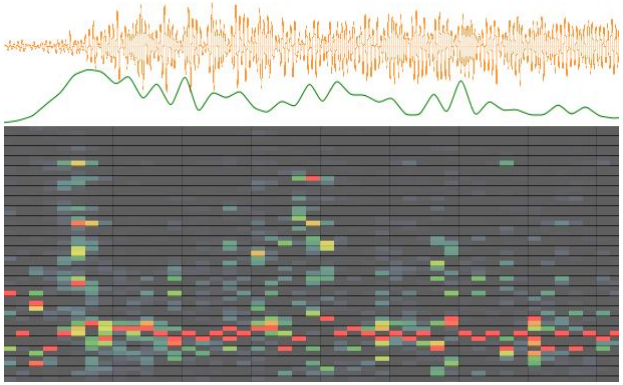
Audio features



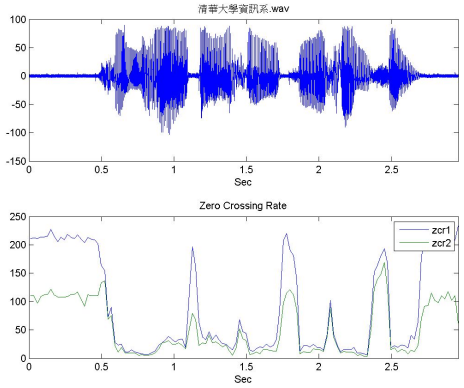
Spectrogram



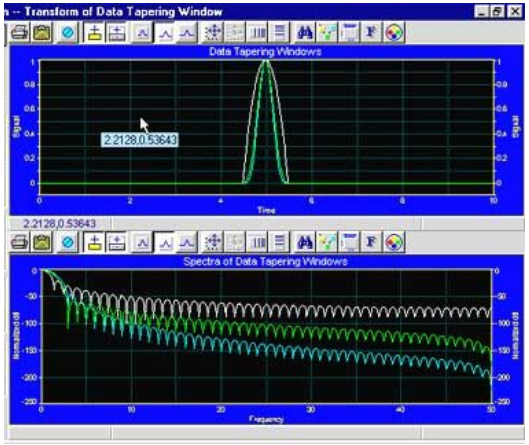
MFCC



Flux



ZCR

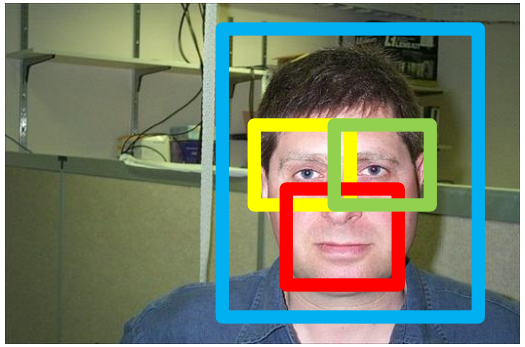


Rolloff

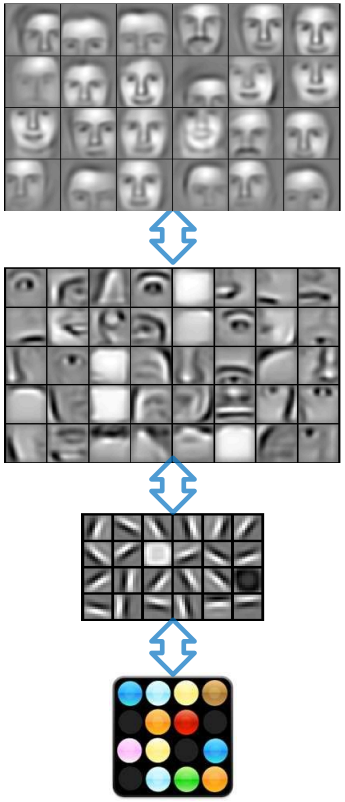
Learning hierarchy of features

1. Learn high-level “structures” from data.

Input data



Feature representation



2. Use these features for ML tasks.

ML applications