

**Spring 2022**

# INTRODUCTION TO COMPUTER VISION

---

**Atlas Wang**

Assistant Professor, The University of Texas at Austin

# Example: Image classification

input	desired output	
	apple	 apple
	pear	 pear
	tomato	 tomato
	cow	 cow
	dog	 dog
	horse	 horse

Training data:

# Example 2: Spam filter



Dear Sir.

First, I must solicit your confidence in this transaction, this is by virtue of its nature as being utterly confidential and top secret. ...



TO BE REMOVED FROM FUTURE  
MAILINGS, SIMPLY REPLY TO THIS  
MESSAGE AND PUT "REMOVE" IN THE  
SUBJECT.

99 MILLION EMAIL ADDRESSES  
FOR ONLY \$99



Ok, I know this is blatantly OT but I'm beginning to go insane. Had an old Dell Dimension XPS sitting in the corner and decided to put it to use, I know it was working pre being stuck in the corner, but when I plugged it in, hit the power nothing happened.

# The basic *supervised learning* framework

$$y = f(x)$$

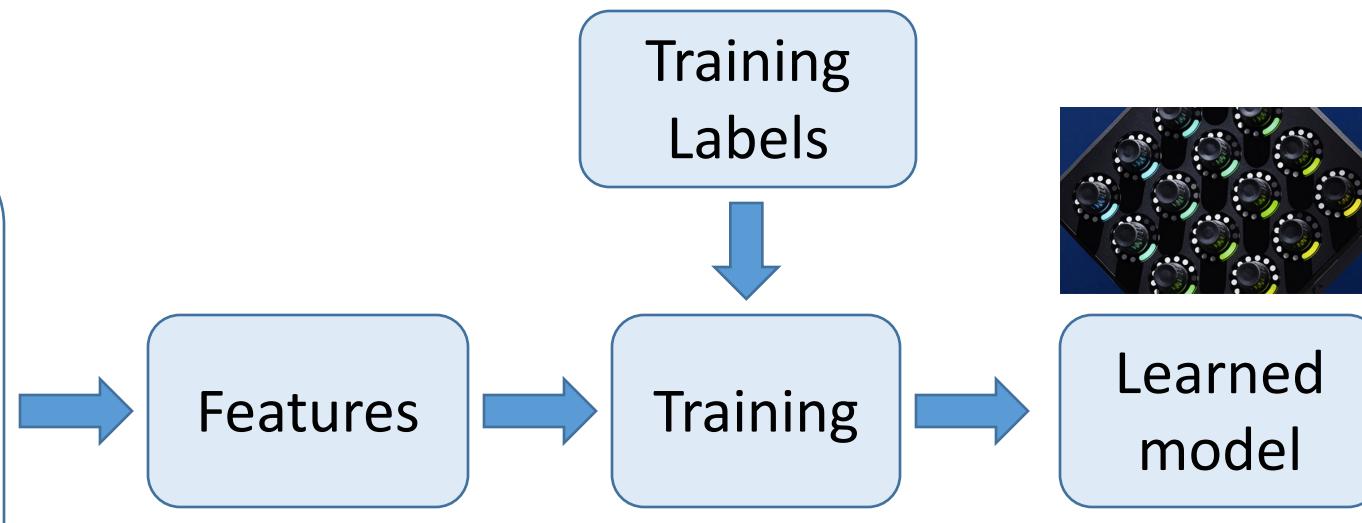
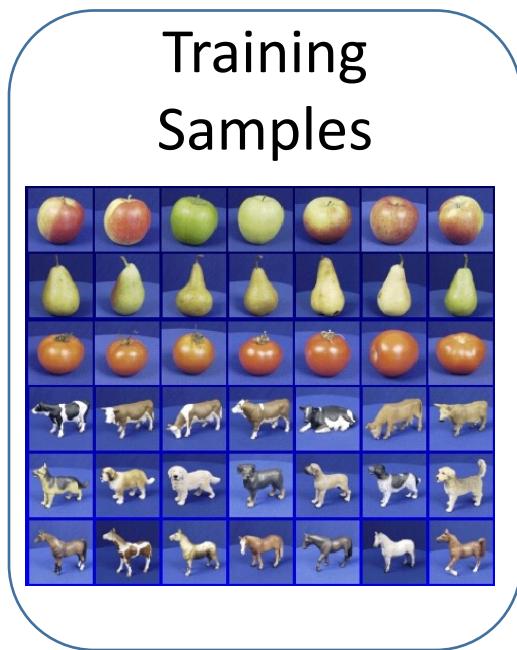
A diagram illustrating the supervised learning framework. At the top is the equation  $y = f(x)$ . Below it, three red arrows point upwards from the words "output", "classification function", and "input" to the corresponding terms in the equation. The word "output" points to the variable  $y$ , "classification function" points to the term  $f$ , and "input" points to the variable  $x$ .

output      classification function      input

- **Learning:** given a *training set* of labeled examples  $\{(x_1, y_1), \dots, (x_N, y_N)\}$ , estimate the parameters of the prediction function  $f$
- **Inference:** apply  $f$  to a never before seen *test example*  $x$  and output the predicted value  $y = f(x)$

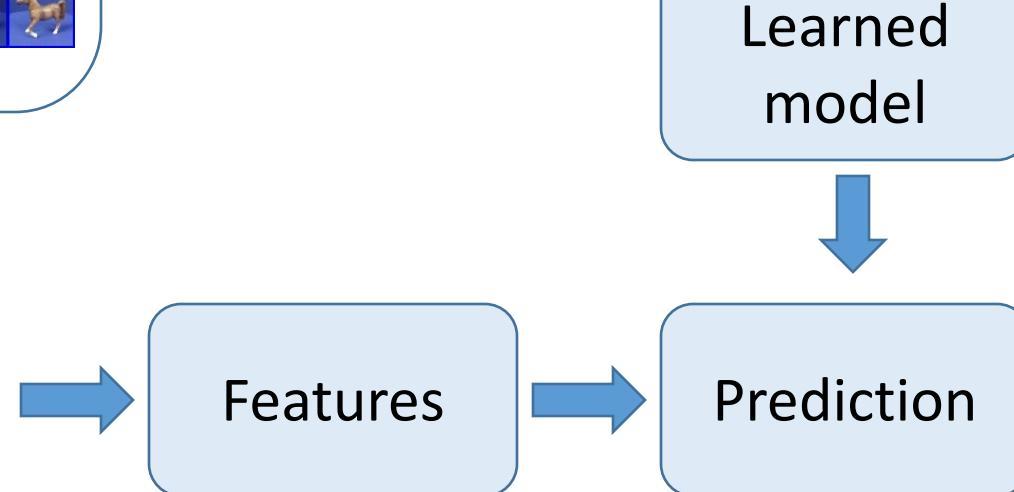
# Learning and inference pipeline

## Learning

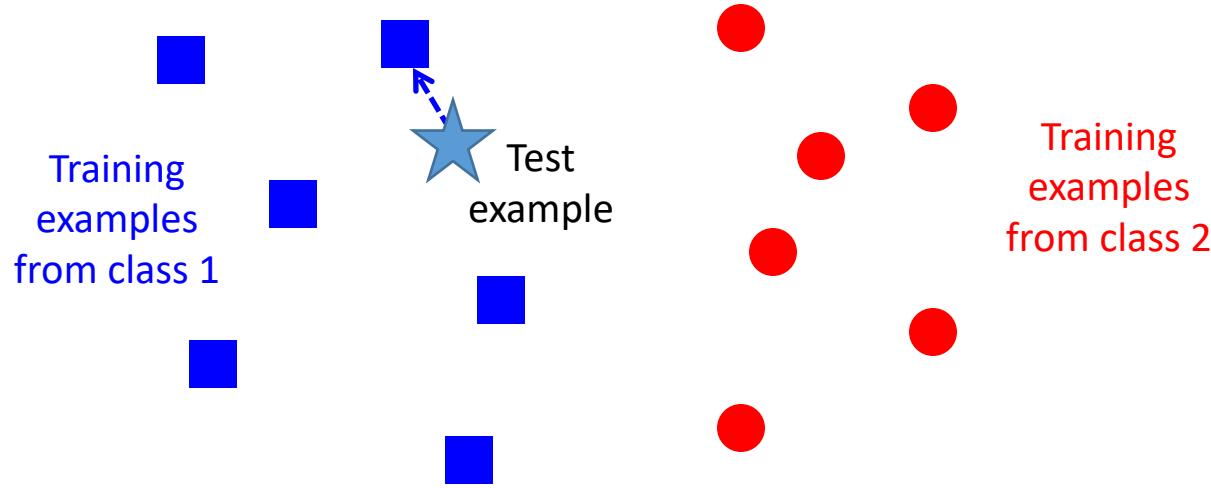


## Inference

Test Sample



# Nearest neighbor classifier

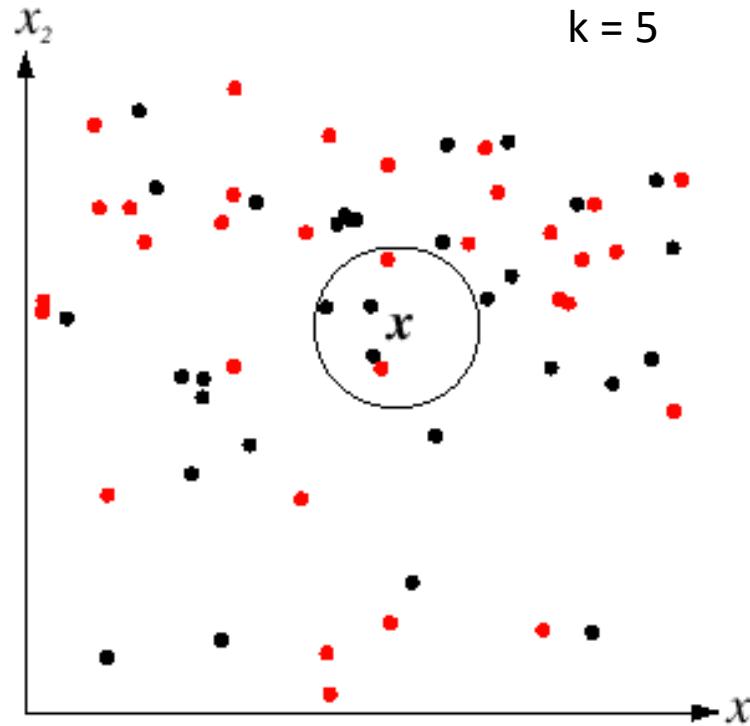


$f(x)$  = label of the training example nearest to  $x$

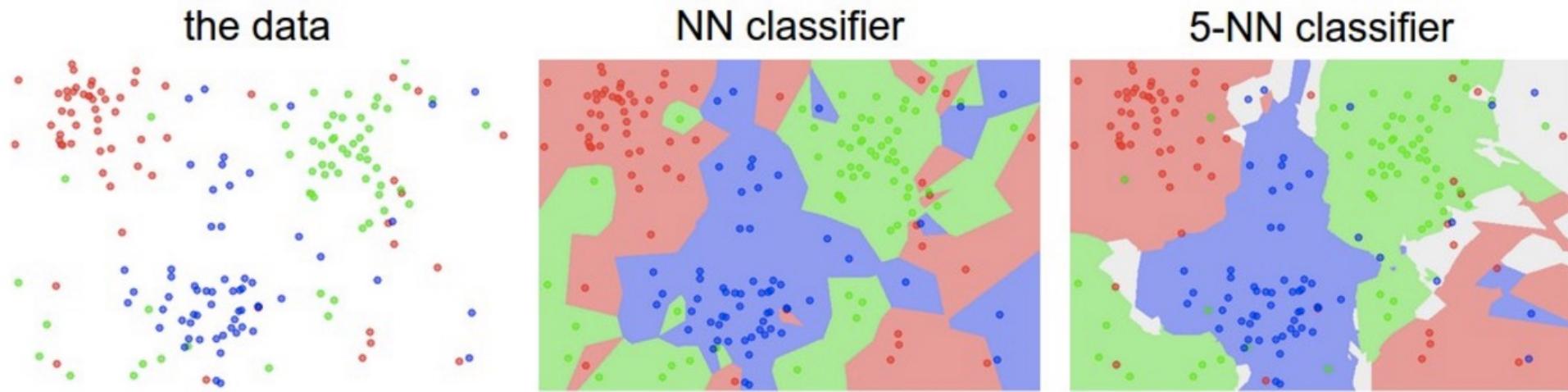
- All we need is a distance function for our inputs
- No training required!

# K-nearest neighbor classifier

- For a new point, find the k closest points from training data
- Vote for class label with labels of the k points

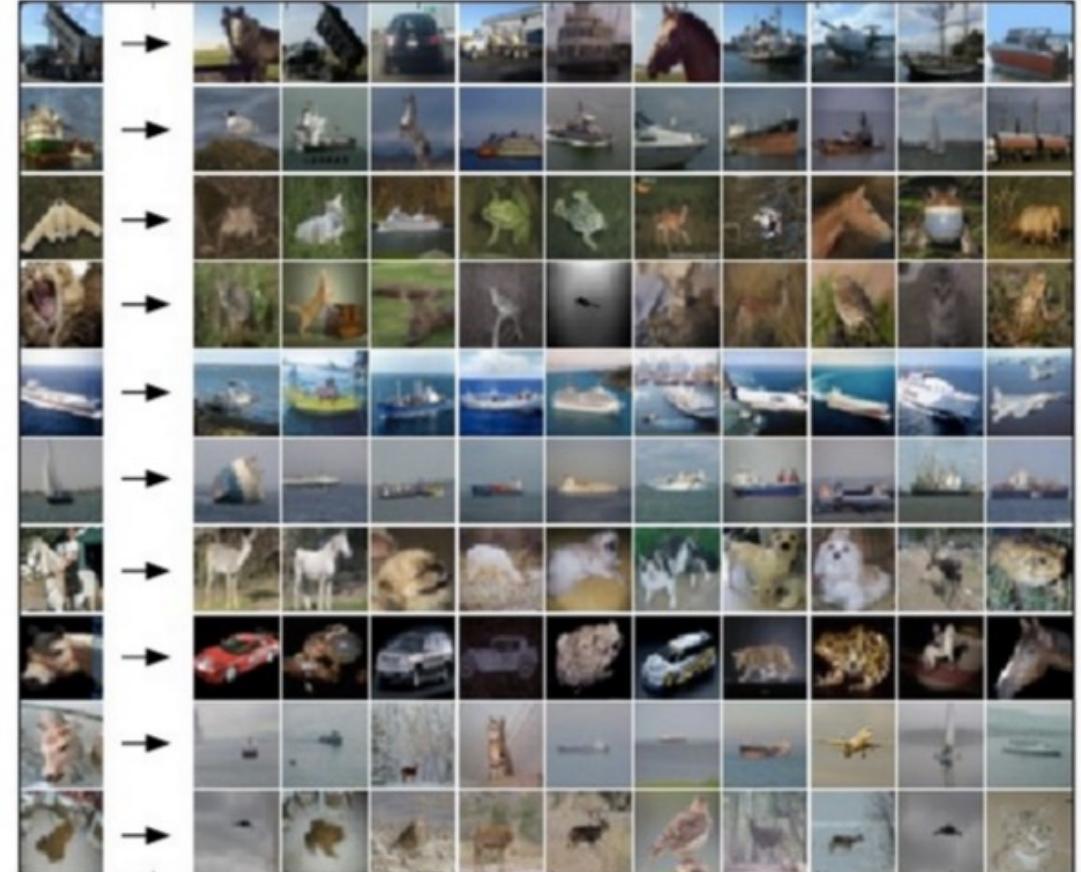
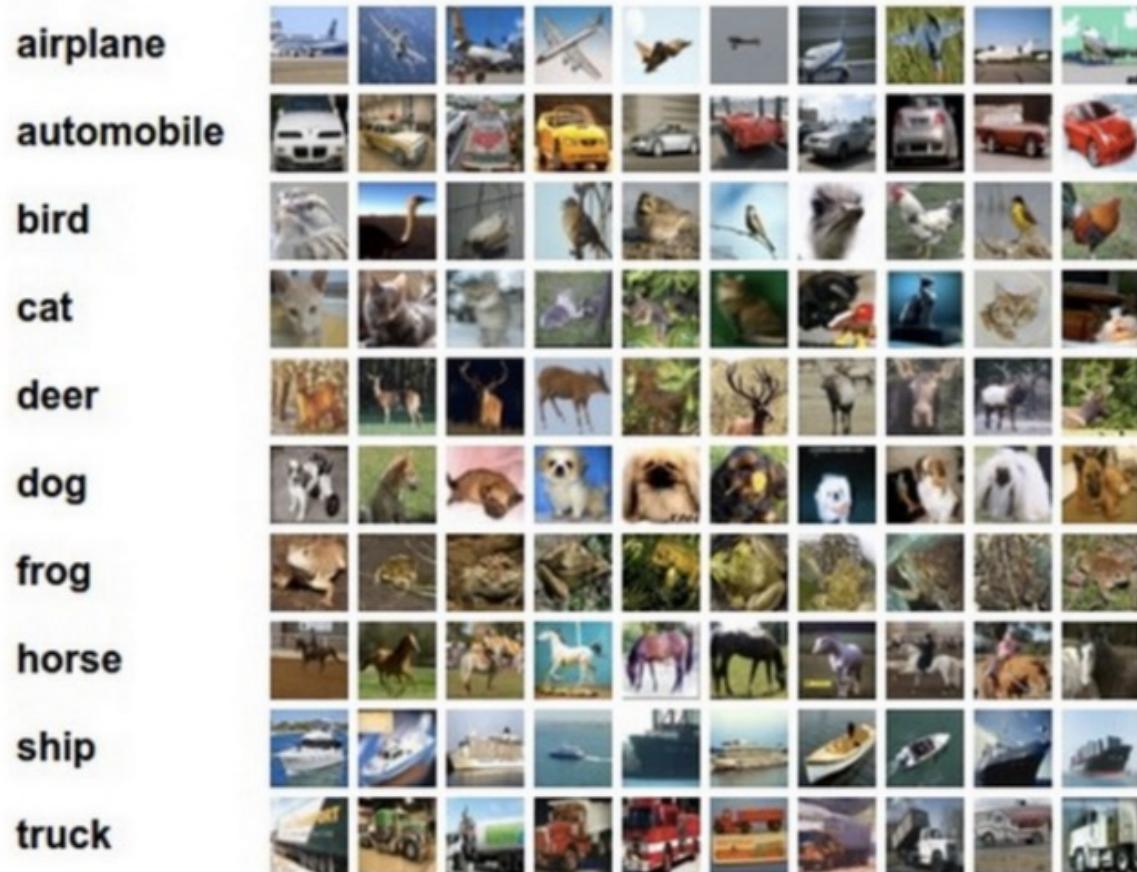


# K-nearest neighbor classifier



- Which classifier is more robust to *outliers*?

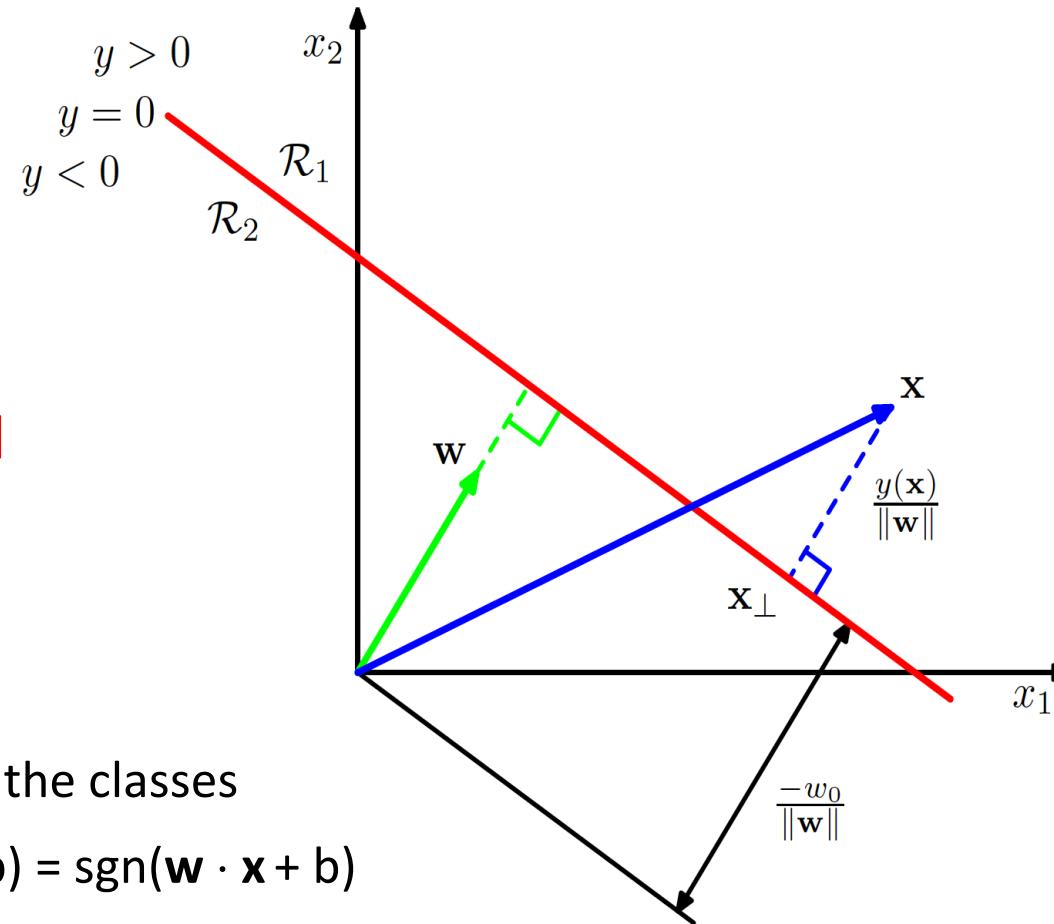
# K-nearest neighbor classifier



Left: Example images from the [CIFAR-10 dataset](#). Right: first column shows a few test images and next to each we show the top 10 nearest neighbors in the training set according to pixel-wise difference.

# Linear classifier

The Classification  
Boundary marked in Red  
(Decision Hyperplane)



- Find a *linear function* to separate the classes

$$f(\mathbf{x}) = \text{sgn}(w_1x_1 + w_2x_2 + \dots + w_Dx_D + b) = \text{sgn}(\mathbf{w} \cdot \mathbf{x} + b)$$

# Logistic Regression

- In two-class problem, the posterior probability of class  $\mathcal{C}_1$  can be written as a logistic sigmoid acting on a linear function of the feature vector  $\phi$  so that

$$p(\mathcal{C}_1|\phi) = y(\phi) = \sigma(\mathbf{w}^T \phi).$$

- $\sigma(\cdot)$  is the logistic sigmoid function.
- $p(\mathcal{C}_2|\phi) = 1 - p(\mathcal{C}_1|\phi)$
- In statistics, the model is known as **logistic regression**.
- The model is for **classification**, not for regression.
- In logistic regression, we estimate the parameter  $\mathbf{w}$  directly.

# Logistic Regression

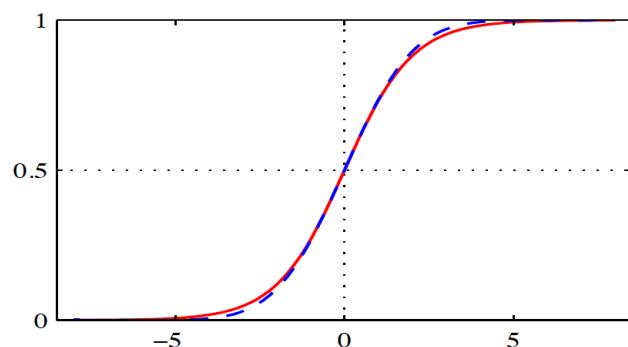
- Definition of **Logistic Sigmoid** function  $\sigma(a)$ :

$$\sigma(a) = \frac{1}{1 + \exp(-a)}$$

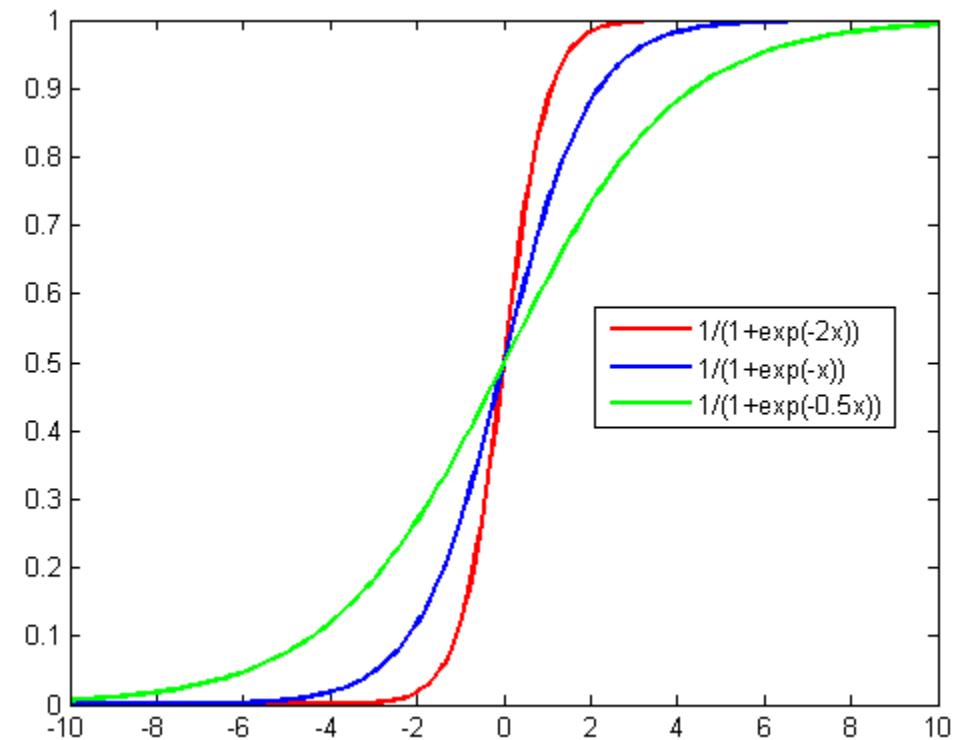
- Properties of Logistic Sigmoid function  $\sigma(a)$ :

$$\sigma(-a) = 1 - \sigma(a)$$

$$\frac{d\sigma}{da} = \sigma(1 - \sigma)$$

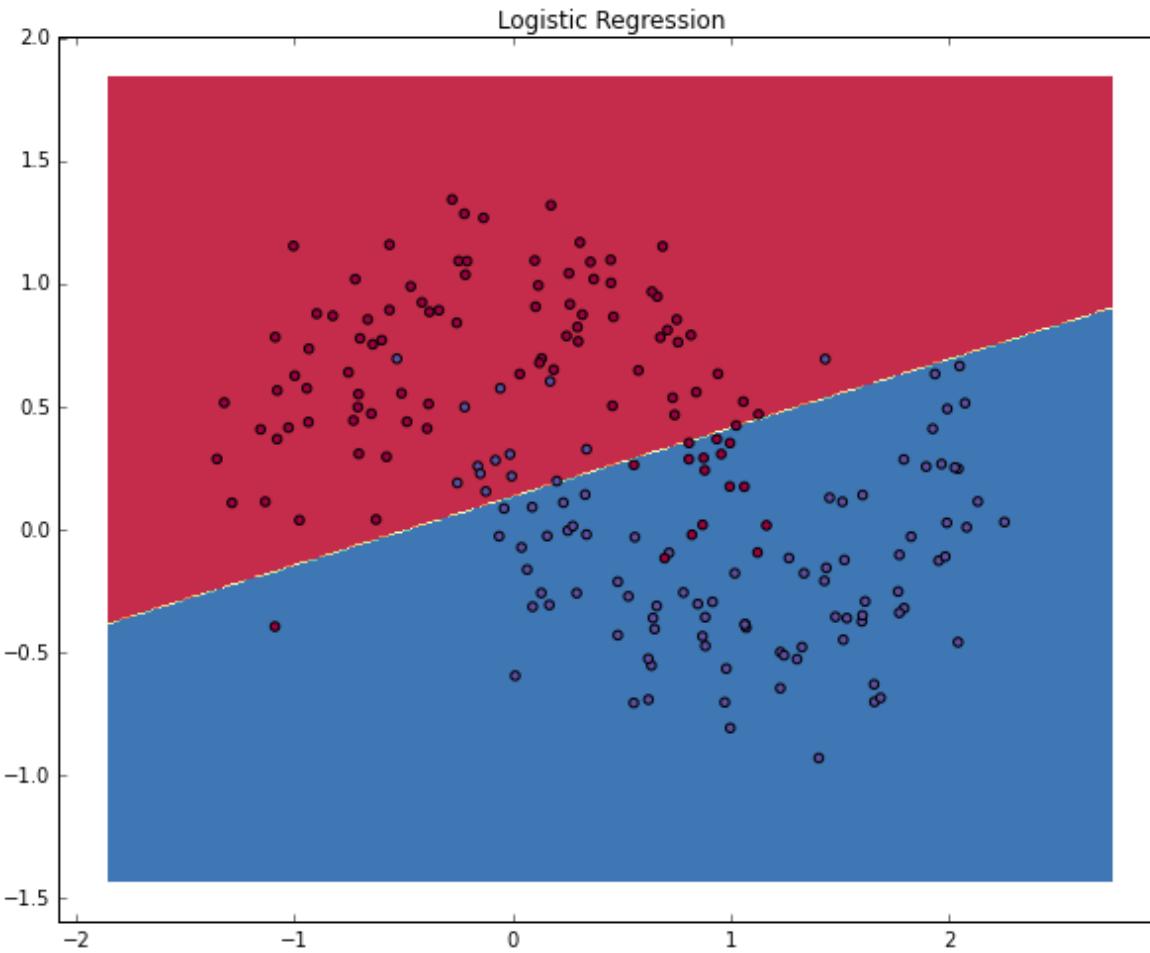


Plot of the logistic sigmoid function  $\sigma(a)$  (Red Line)



A Continuously Differentiable  
Approximation of the 0-1 loss

# Logistic Regression



Linear Decision  
Boundary! Why?

# Logistic Regression

- For a training data set  $\{\phi_n, t_n\}$  where  $t_n \in \{0, 1\}$  and  $n = 1, 2, \dots, N$ , the likelihood function is

$$p(\mathbf{t}|\mathbf{w}) = \prod_{n=1}^N y_n^{t_n} (1 - y_n)^{1-t_n} \quad (3)$$

- Definitions of  $t_n$ ,  $\mathbf{t}$  and  $y_n$

$$t_n = \begin{cases} 1 & \text{if } n \in \mathcal{C}_1 \\ 0 & \text{if } n \in \mathcal{C}_2 \end{cases}$$

$$\mathbf{t} = (t_1, t_2, \dots, t_N)^T$$

$$y_n = p(\mathcal{C}_1 | \phi_n) = \sigma(\mathbf{w}^T \phi_n)$$

# Logistic Regression

- The error function is the negative logarithm of the likelihood, namely, **Cross-entropy** error function:

$$E(\mathbf{w}) = -\ln p(\mathbf{t}|\mathbf{w}) = \sum_{n=1}^N \{ t_n \ln y_n + (1 - t_n) \ln(1 - y_n) \}$$

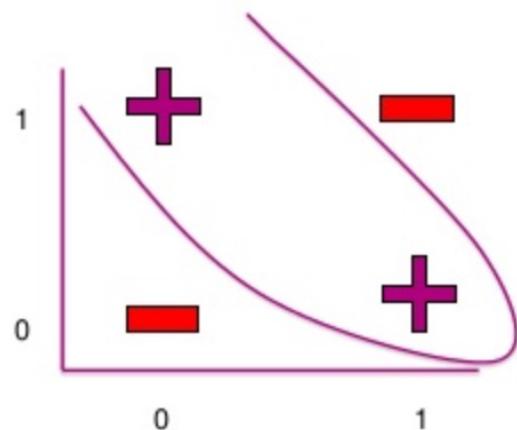
The gradient of cross entropy function with respect to  $\mathbf{w}$  is

$$\nabla E(\mathbf{w}) = \sum_{n=1}^N (y_n - t_n) \phi_n$$

# NN vs. linear classifiers

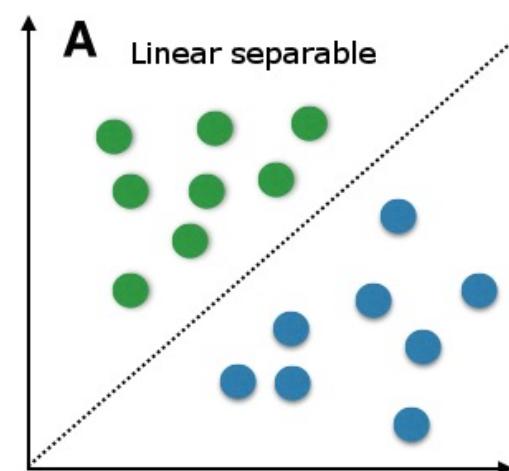
- NN pros:
  - + Simple to implement
  - + Decision boundaries not necessarily linear
  - + Works for any number of classes
  - + *Nonparametric* method
- NN cons:
  - Need good distance function
  - Slow at test time
- Linear pros:
  - + Low-dimensional *parametric* representation
  - + Very fast at test time
- Linear cons:
  - Works for two classes
  - How to train the linear function?
  - What if data is not linearly separable?

# Limitation of Linear Classifiers

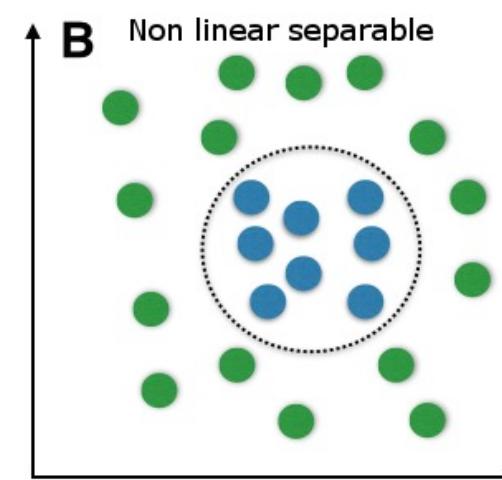


XOR

Need non-linear features



A Linear separable

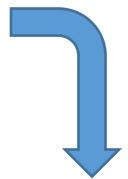


B Non linear separable

# Softmax Regression

- The Multi-Class version of Logistic Regression (popular in deep learning)  
**(Reference:** <http://ufldl.stanford.edu/tutorial/supervised/SoftmaxRegression/>)

$$h_{\theta}(x) = \frac{1}{1 + \exp(-\theta^T x)},$$



$$h_{\theta}(x) = \begin{bmatrix} P(y = 1|x; \theta) \\ P(y = 2|x; \theta) \\ \vdots \\ P(y = K|x; \theta) \end{bmatrix} = \frac{1}{\sum_{j=1}^K \exp(\theta^{(j)\top} x)} \begin{bmatrix} \exp(\theta^{(1)\top} x) \\ \exp(\theta^{(2)\top} x) \\ \vdots \\ \exp(\theta^{(K)\top} x) \end{bmatrix}$$

# Decision tree classifier

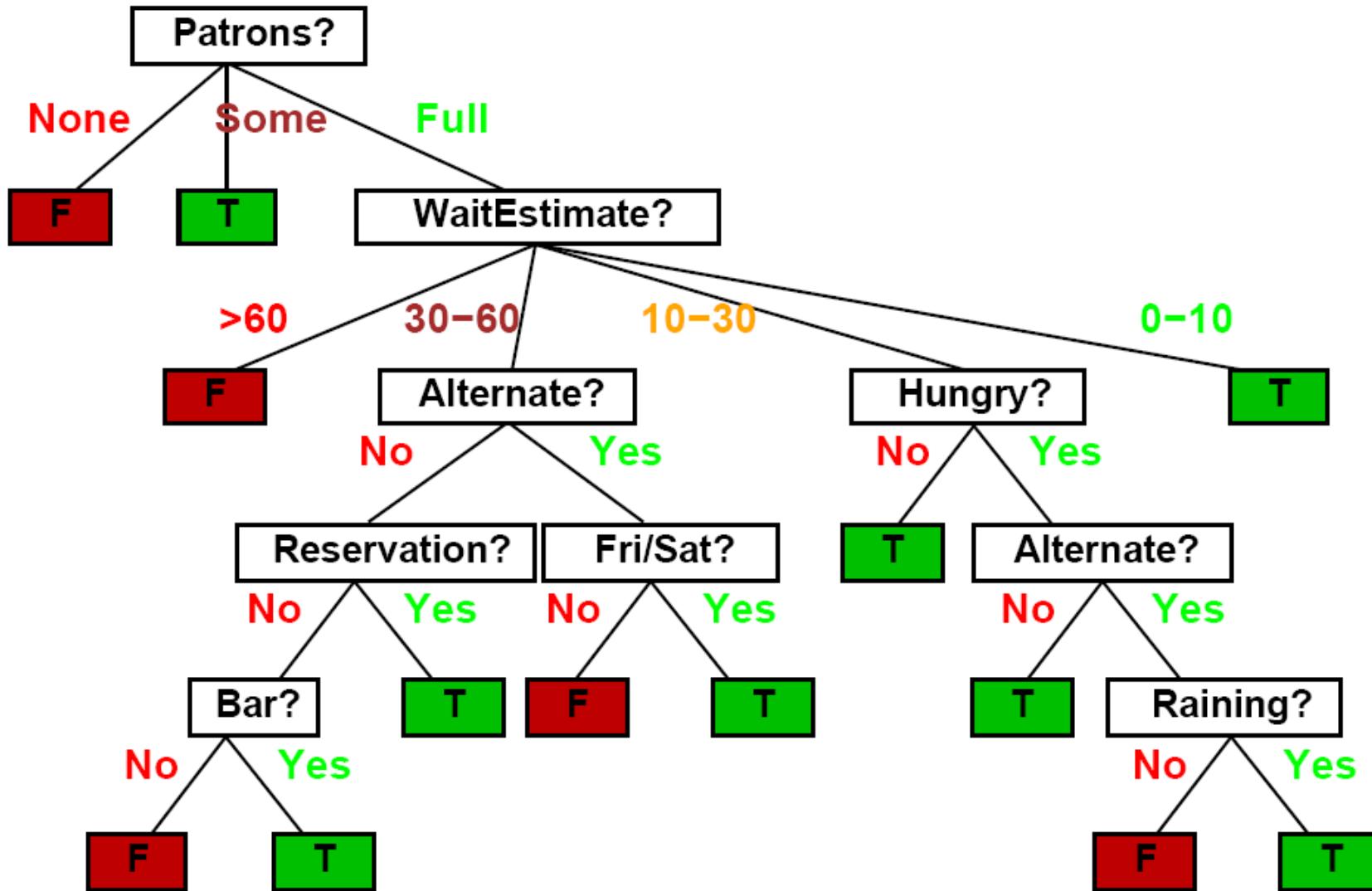
**Example problem:** decide whether to wait for a table at a restaurant, based on the following attributes:

1. **Alternate:** is there an alternative restaurant nearby?
2. **Bar:** is there a comfortable bar area to wait in?
3. **Fri/Sat:** is today Friday or Saturday?
4. **Hungry:** are we hungry?
5. **Patrons:** number of people in the restaurant (None, Some, Full)
6. **Price:** price range (\$, \$\$, \$\$\$)
7. **Raining:** is it raining outside?
8. **Reservation:** have we made a reservation?
9. **Type:** kind of restaurant (French, Italian, Thai, Burger)
10. **WaitEstimate:** estimated waiting time (0-10, 10-30, 30-60, >60)

# Decision tree classifier

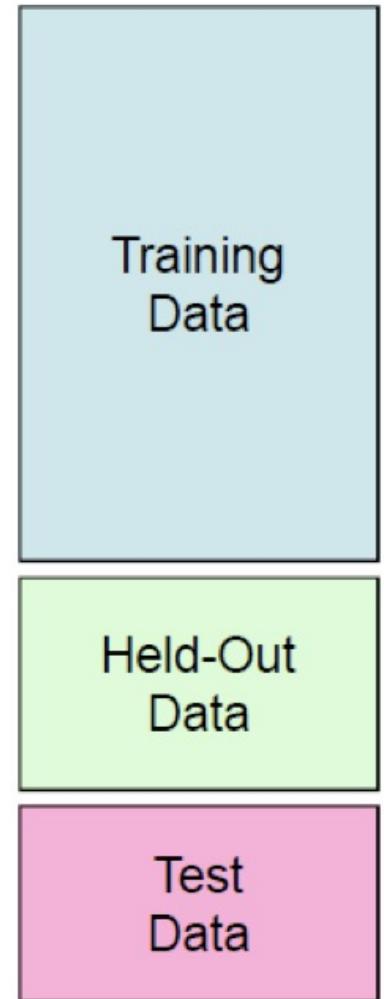
Example	Attributes										Target <i>Wait</i>
	<i>Alt</i>	<i>Bar</i>	<i>Fri</i>	<i>Hun</i>	<i>Pat</i>	<i>Price</i>	<i>Rain</i>	<i>Res</i>	<i>Type</i>	<i>Est</i>	
$X_1$	T	F	F	T	Some	\$\$\$	F	T	French	0–10	T
$X_2$	T	F	F	T	Full	\$	F	F	Thai	30–60	F
$X_3$	F	T	F	F	Some	\$	F	F	Burger	0–10	T
$X_4$	T	F	T	T	Full	\$	F	F	Thai	10–30	T
$X_5$	T	F	T	F	Full	\$\$\$	F	T	French	>60	F
$X_6$	F	T	F	T	Some	\$\$	T	T	Italian	0–10	T
$X_7$	F	T	F	F	None	\$	T	F	Burger	0–10	F
$X_8$	F	F	F	T	Some	\$\$	T	T	Thai	0–10	T
$X_9$	F	T	T	F	Full	\$	T	F	Burger	>60	F
$X_{10}$	T	T	T	T	Full	\$\$\$	F	T	Italian	10–30	F
$X_{11}$	F	F	F	F	None	\$	F	F	Thai	0–10	F
$X_{12}$	T	T	T	T	Full	\$	F	F	Burger	30–60	T

# Decision tree classifier



# Experimentation cycle

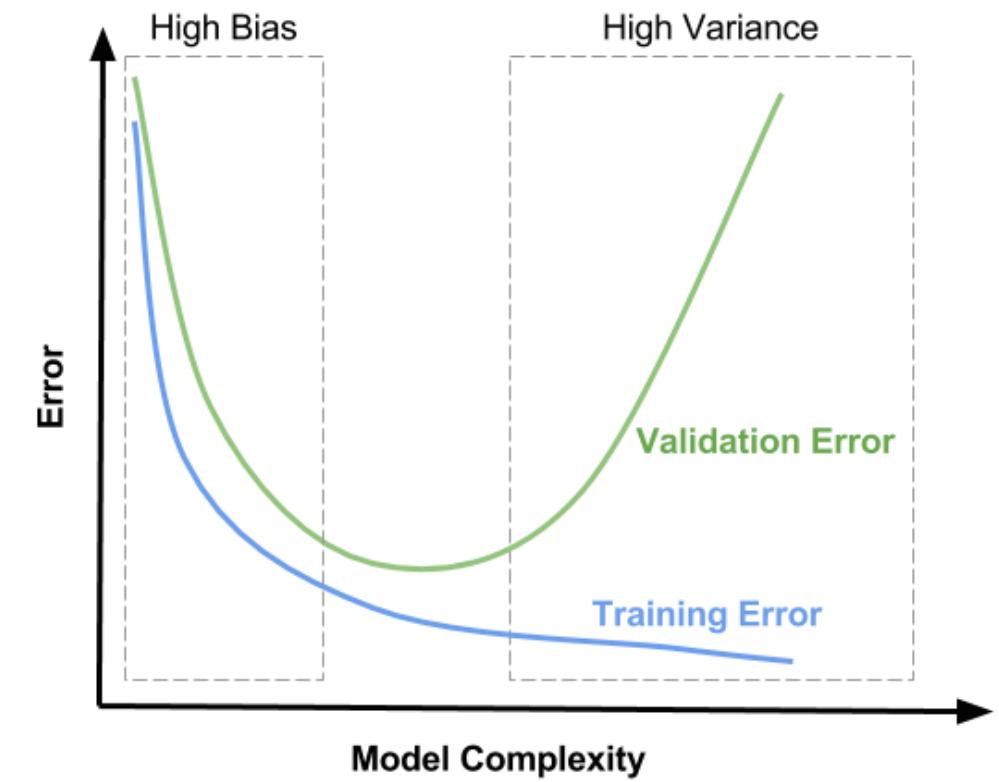
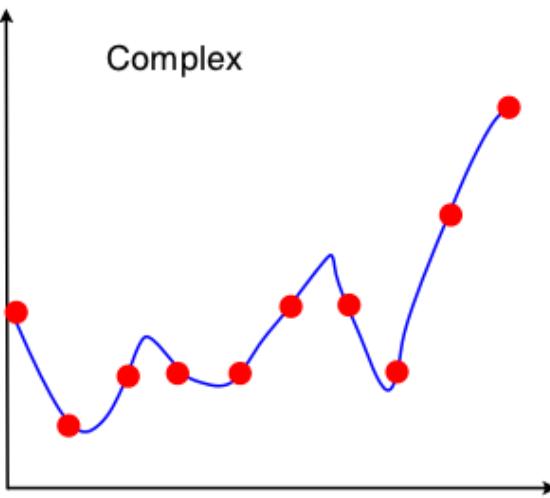
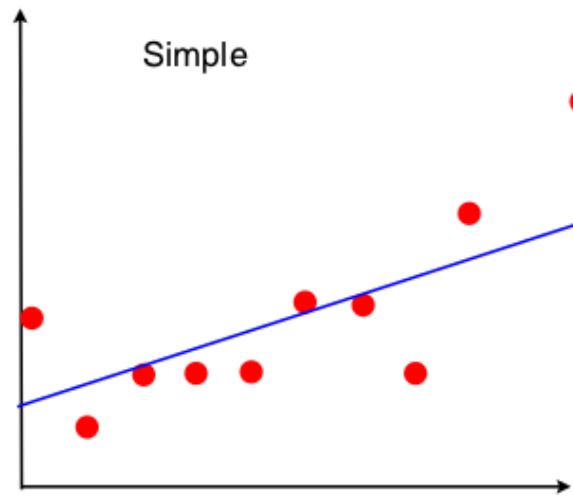
- Learn *parameters* on the *training set*
- Tune *hyperparameters* (implementation choices) on the *held out validation set*
- Evaluate performance on the *test set*
- Do not peek at the test set!
- *Generalization* and *overfitting*
  - Want classifier that does well on never before seen data
  - Overfitting: good performance on the training/validation set, poor performance on test set



# Bias-Variance Tradeoff

- The bias–variance tradeoff is the fundamental dilemma of minimizing between two sources of errors that prevent ML algorithms from generalizing beyond their training set:
  - The bias is error from erroneous assumptions in the learning algorithm. High bias can cause an algorithm to miss the relevant relations between features and target outputs (e.g., model is too simple -> **underfitting**).
  - The variance is error from sensitivity to small fluctuations in the training set. High variance can cause an algorithm to model the random noise in the training data, rather than the intended outputs (e.g., model is too complicated -> **overfitting**).

# Bias-Variance Tradeoff



# Cross Validation

- Given available data in practice, how can we try to find the best-trained algorithm, that will generalize well on unseen testing data?

- Most common approach is  **$K$ -fold cross validation**:

- Partition the training data  $\mathcal{D}$  into  $K$  separate sets of equal size
  - Suppose  $\mathcal{D} = (\mathcal{D}_1, \mathcal{D}_2, \dots, \mathcal{D}_K)$
  - Commonly chosen  $K$ s are  $K = 5$  and  $K = 10$
- For each  $k = 1, 2, \dots, K$ , fit the model  $\hat{f}_{-k}^{(\lambda)}(\mathbf{x})$  to the training set excluding the  $k$ th-fold  $\mathcal{D}_k$
- Compute the fitted values for the observations in  $\mathcal{D}_k$ , based on the training data that excluded this fold
- Compute the cross-validation (CV) error for the  $k$ -th fold:

$$(\text{CV Error})_k^{(\lambda)} = |\mathcal{D}_k|^{-1} \sum_{(\mathbf{x}, y) \in \mathcal{D}_k} (y - \hat{f}_{-k}^{(\lambda)}(\mathbf{x}))^2$$

- The model then has overall cross-validation error:

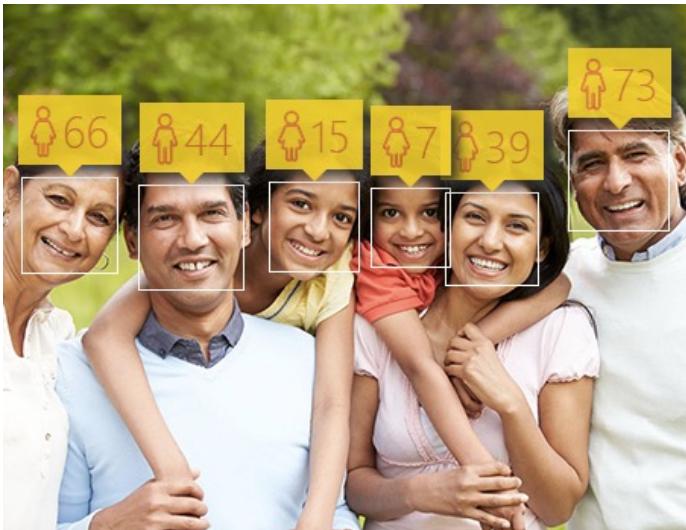
$$(\text{CV Error})^{(\lambda)} = K^{-1} \sum_{k=1}^K (\text{CV Error})_k^{(\lambda)}$$

- Select  $\lambda^*$  as the one with minimum  $(\text{CV Error})^{(\lambda)}$
- Compute the chosen model  $\hat{f}_{-k}^{(\lambda^*)}(\mathbf{x})$  on the entire training set  $T = (\mathcal{D}_1, \mathcal{D}_2, \dots, \mathcal{D}_K)$
- Apply  $\hat{f}_{-k}^{(\lambda^*)}(\mathbf{x})$  to the test set to assess test error

# Beyond supervised classification

- Other prediction scenarios
  - Regression
  - Structured prediction
- Other supervision scenarios
  - Unsupervised learning
  - Self-supervised or predictive learning
  - Active learning
  - Lifelong learning

# Beyond classification: Regression

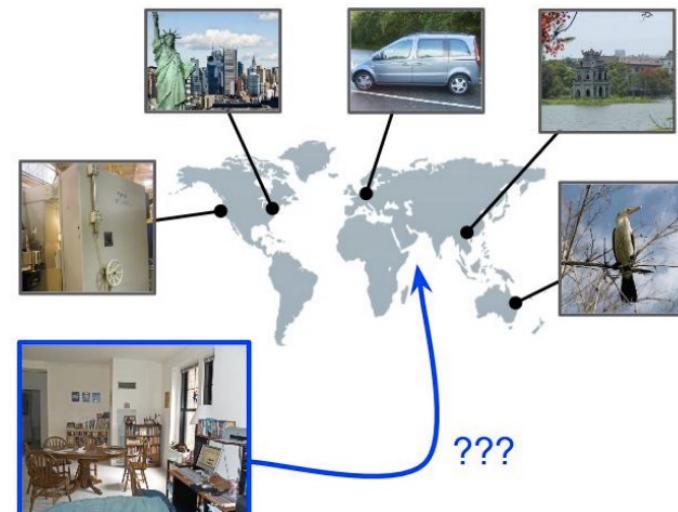


Age estimation

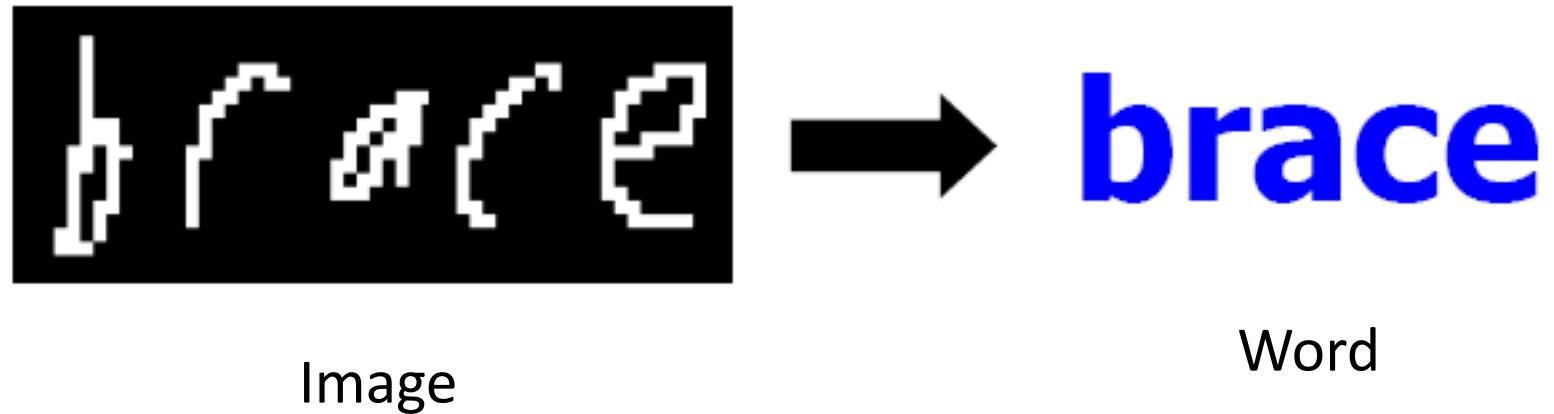


When was that made?

IM2GPS



# Beyond classification: Structured prediction

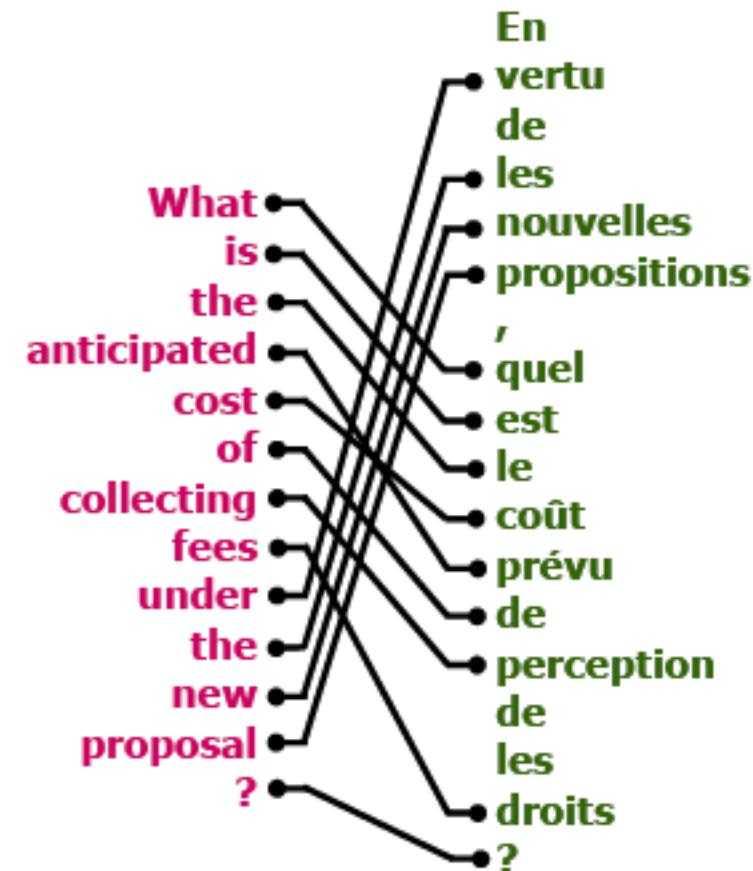


Source: B. Taskar

# Structured Prediction

**What is the anticipated cost of collecting fees under the new proposal?**

**En vertu des nouvelles propositions, quel est le coût prévu de perception des droits?**



Sentence in two languages

Word alignment

Source: B. Taskar

# Structured Prediction

RSCCP CYWG GCPW  
GQNC YPEG CSGPKV



RS**CCPC**YWGG**C**PWGQNCYPEG**C**SGPKV  
1 2 3 4 5 6

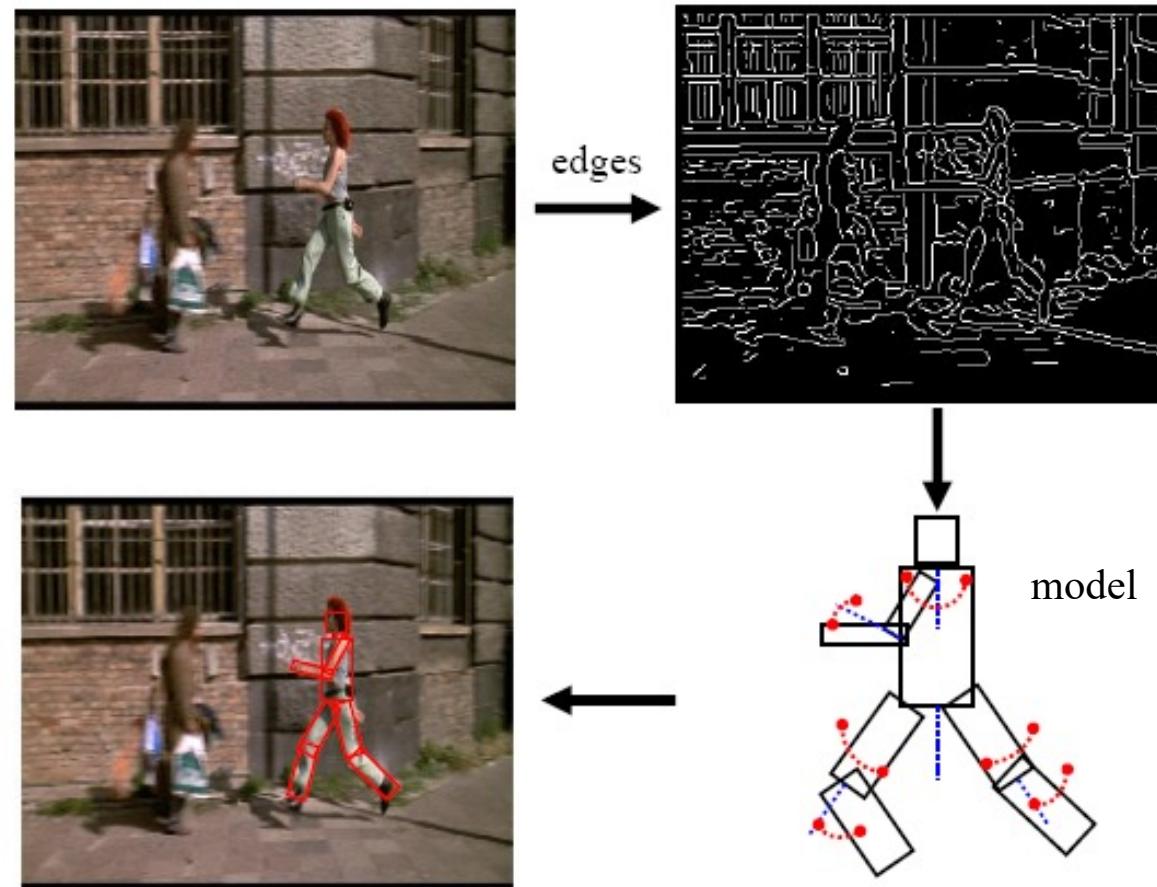
## Amino-acid sequence

# Bond structure

Source: B. Taskar

# Structured Prediction

- Many image-based inference tasks can loosely be thought of as “structured prediction”



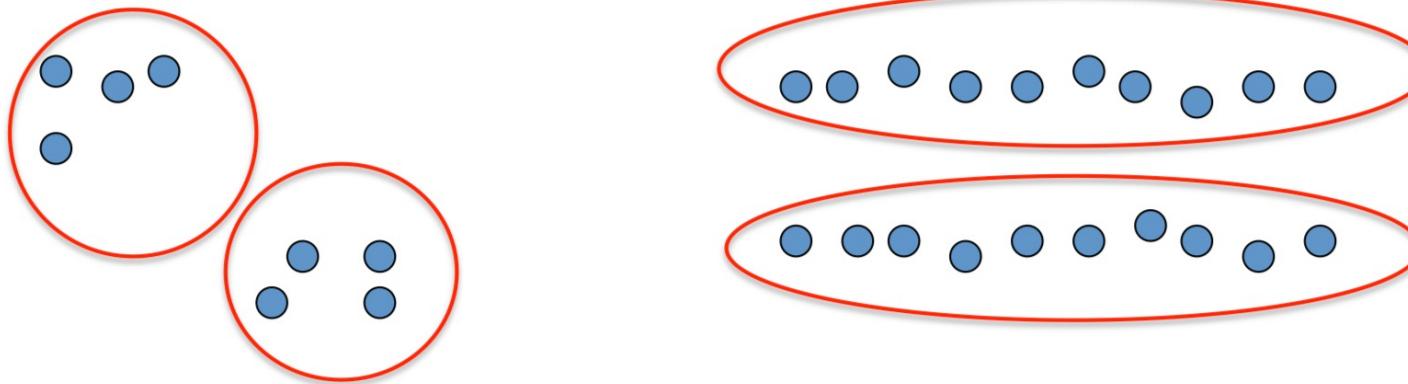
Source: D. Ramanan

# Unsupervised Learning

- Idea: Given only *unlabeled* data as input, learn some sort of structure
- The objective is often more vague or subjective than in supervised learning
- This is more of an exploratory/descriptive data analysis

# Clustering

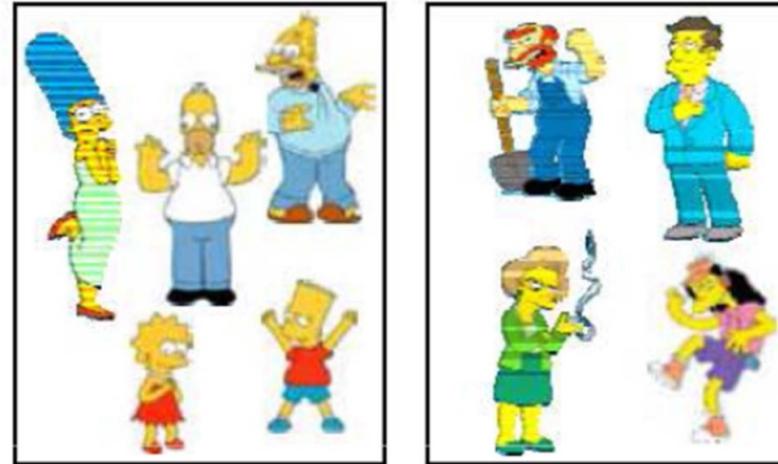
- Basic idea: group together similar instances
- Example: 2D point patterns



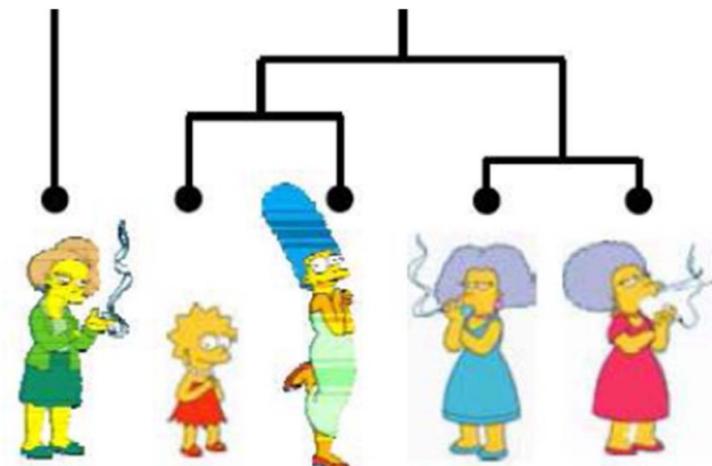
- What could “similar” mean?
  - One option: small Euclidean distance (squared)
$$\text{dist}(\vec{x}, \vec{y}) = \|\vec{x} - \vec{y}\|_2^2$$
  - Clustering results are crucially dependent on the measure of similarity (or distance) between “points” to be clustered

# Clustering Algorithms

- Partition algorithms (Flat)
  - K-means
  - Mixture of Gaussian
  - Spectral Clustering

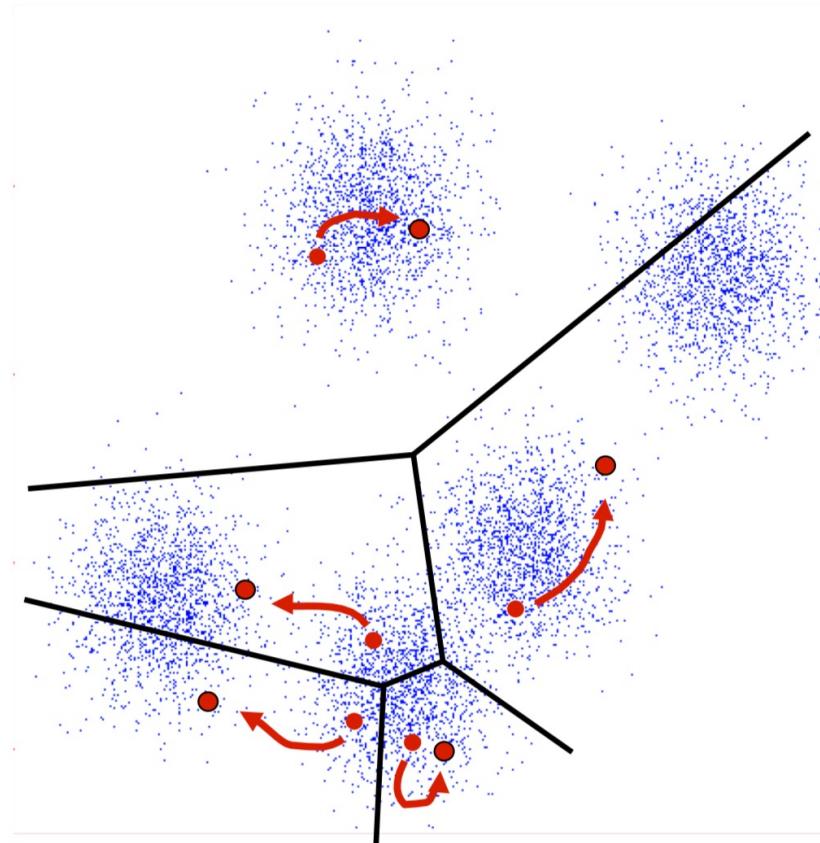


- Hierarchical algorithms
  - Bottom up – agglomerative
  - Top down – divisive



# K-Means

- An iterative clustering algorithm
  - **Initialize:** Pick  $K$  random points as cluster centers
  - **Alternate:**
    1. Assign data points to closest cluster center
    2. Change the cluster center to the average of its assigned points
  - Stop when no points' assignments change



# K-Means: Local Convergence

## Objective

$$\min_{\mu} \min_C \sum_{i=1}^k \sum_{x \in C_i} |x - \mu_i|^2$$

1. Fix  $\mu$ , optimize  $C$ :

$$\min_C \sum_{i=1}^k \sum_{x \in C_i} |x - \mu_i|^2 = \min_C \sum_i^n |x_i - \mu_{x_i}|^2$$

**Step 1 of kmeans**

2. Fix  $C$ , optimize  $\mu$ :

$$\min_{\mu} \sum_{i=1}^k \sum_{x \in C_i} |x - \mu_i|^2$$

- Take partial derivative of  $\mu_i$  and set to zero, we have

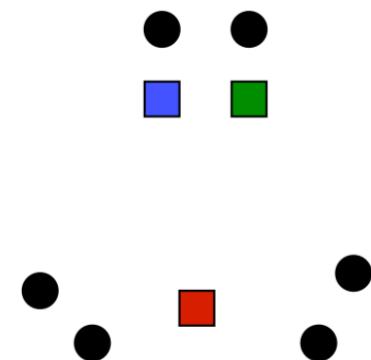
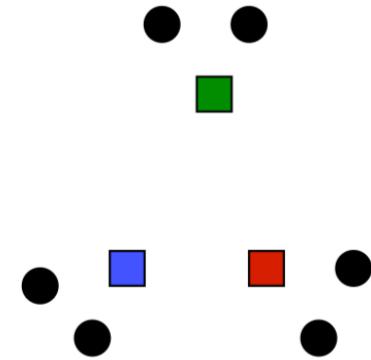
$$\mu_i = \frac{1}{|C_i|} \sum_{x \in C_i} x$$

**Step 2 of kmeans**

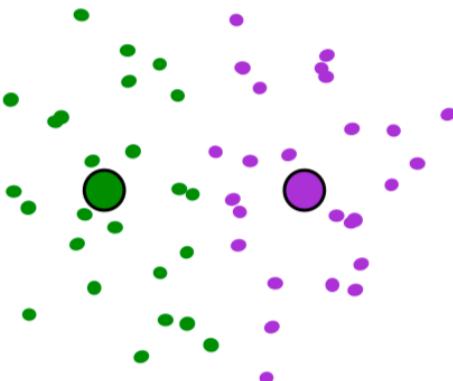
Kmeans takes an alternating optimization approach, each step is guaranteed to decrease the objective – thus guaranteed to converge

# K-Means is Fragile

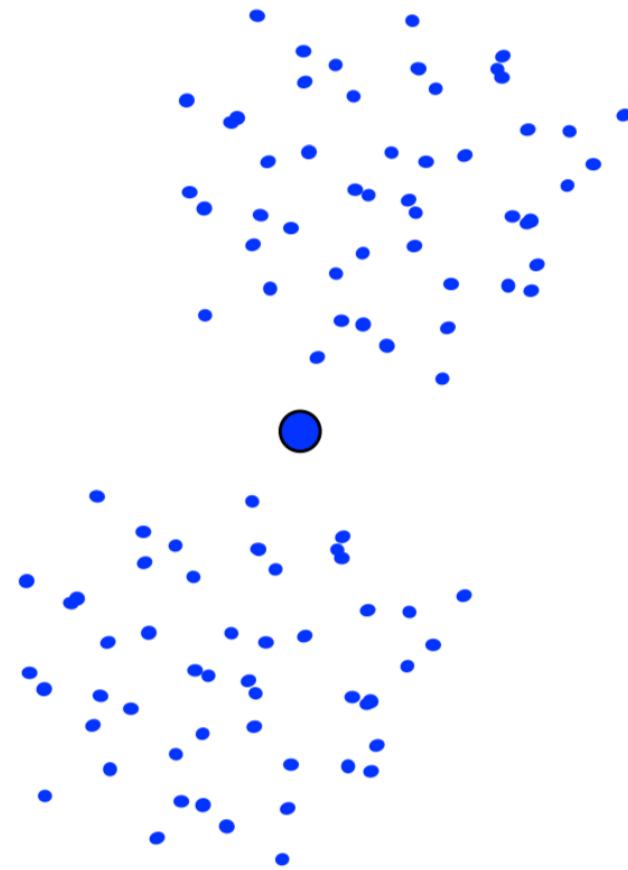
- K-means **algorithm** is a heuristic
  - Requires initial means
  - It does matter what you pick!
  - What can go wrong?
  - Various schemes for preventing this kind of thing: variance-based split / merge, initialization heuristics



# Stuck at Poor Local Minimum

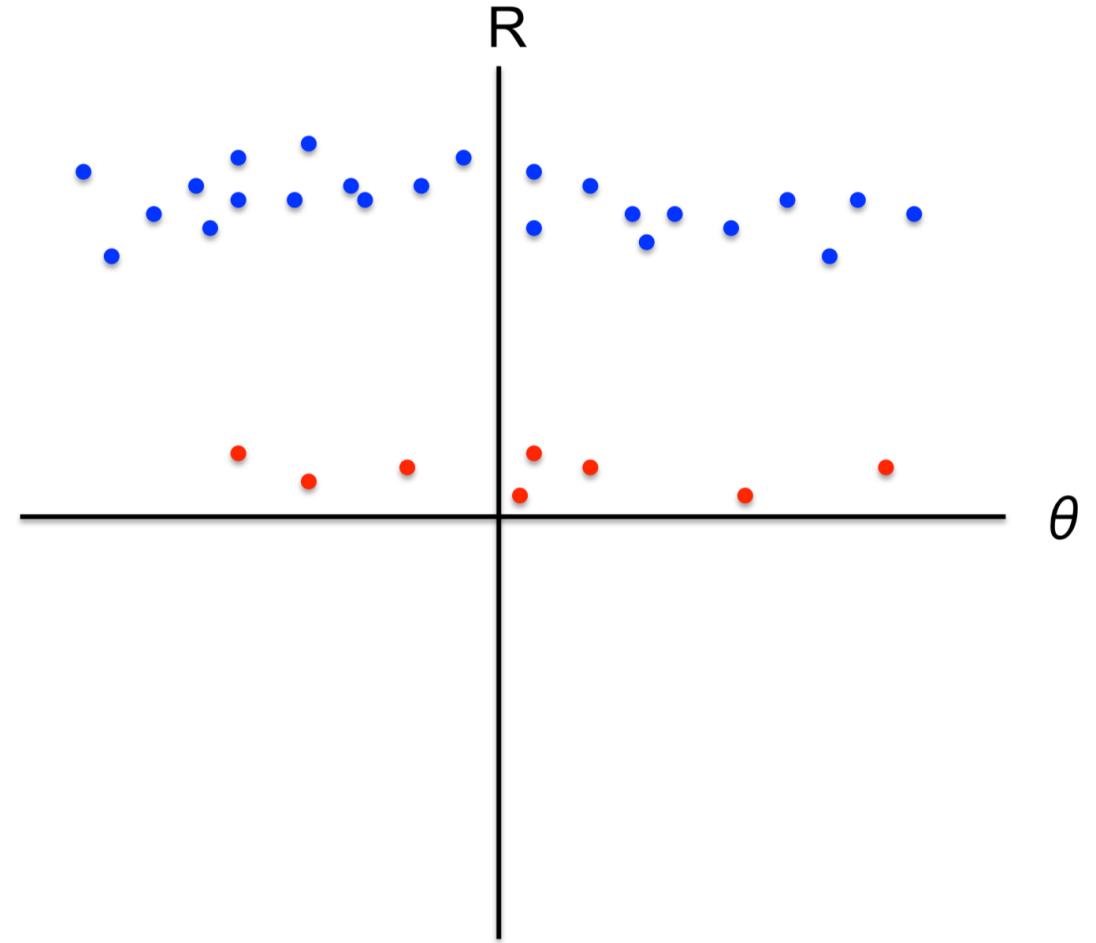
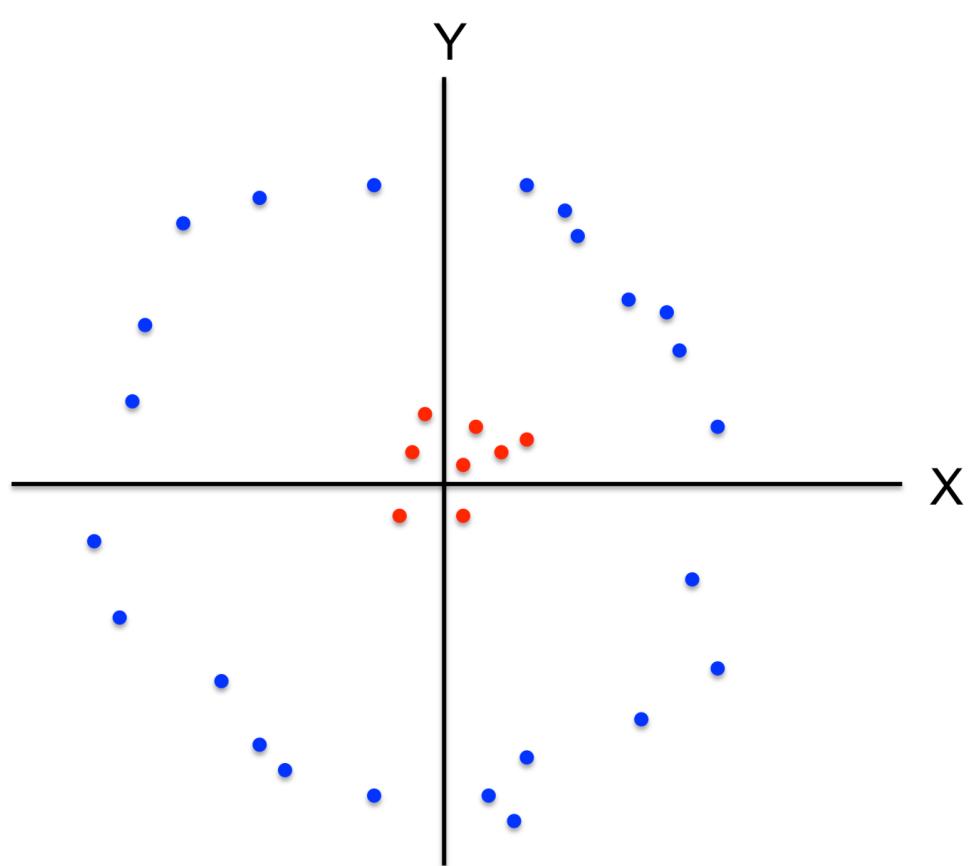


Would be better to have  
one cluster here

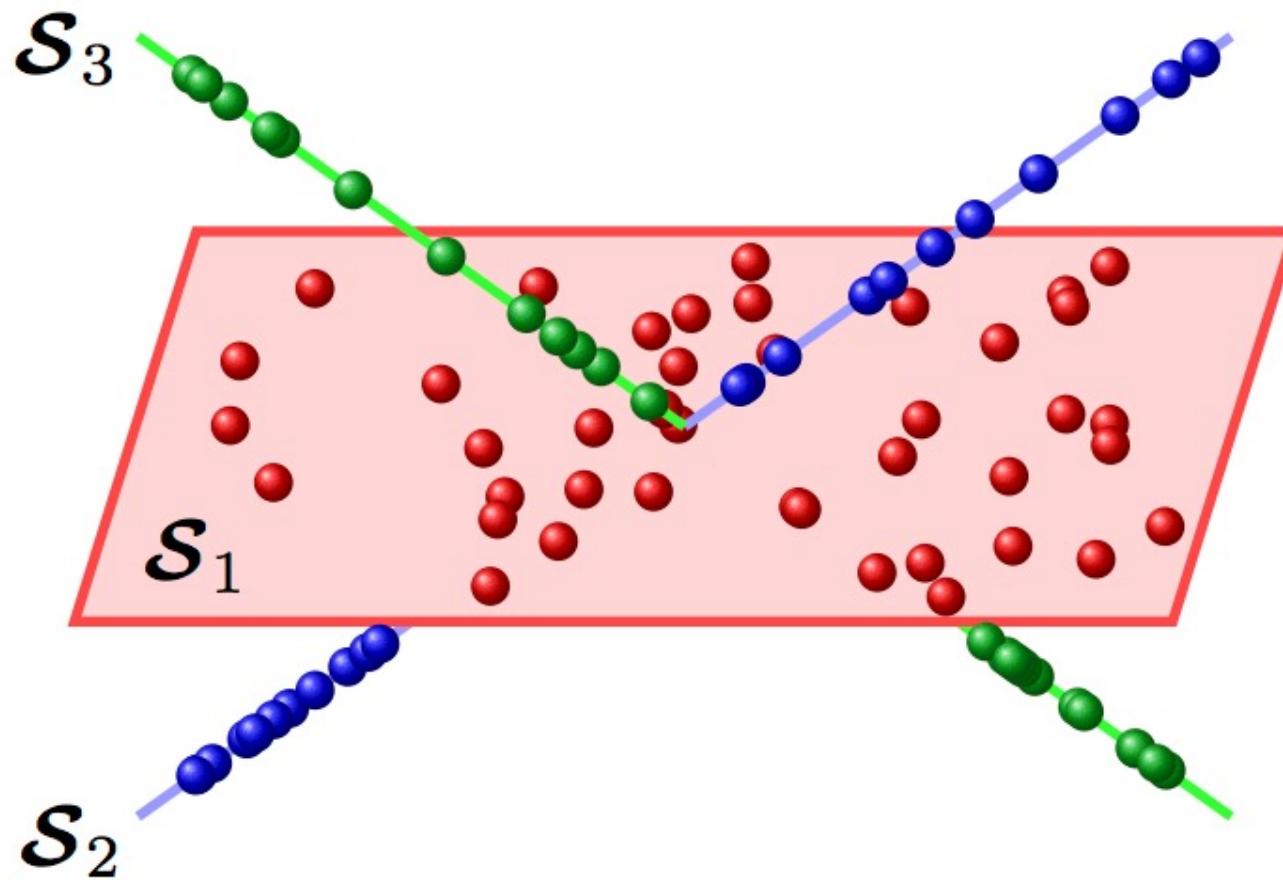


... and two clusters here

# Euclidean Distance Might Not be Proper



Do not underestimate clustering!!



cute rabbit bunny animal  
baby adorable pet  
funny animals



cheerleader football girls  
basketball girls dance  
university sports college



bird birds nature wildlife  
animal booby eagle  
hawk flight



nature macro flower  
closeup green insect  
bravo red yellow



music concert rock live  
festival band scientists  
dance drum



city urban manhattan new  
building downtown night  
architecture buildings



home design office house  
interior kitchen fashion  
work room



portrait face self girl  
woman eyes smile  
child portraits



abandoned decay old  
urban rust industrial  
factory jail rusty



underwater fish diving  
scuba coral sea  
ocean reef dive



autumn trees tree  
park fall leaves  
forest fog mist

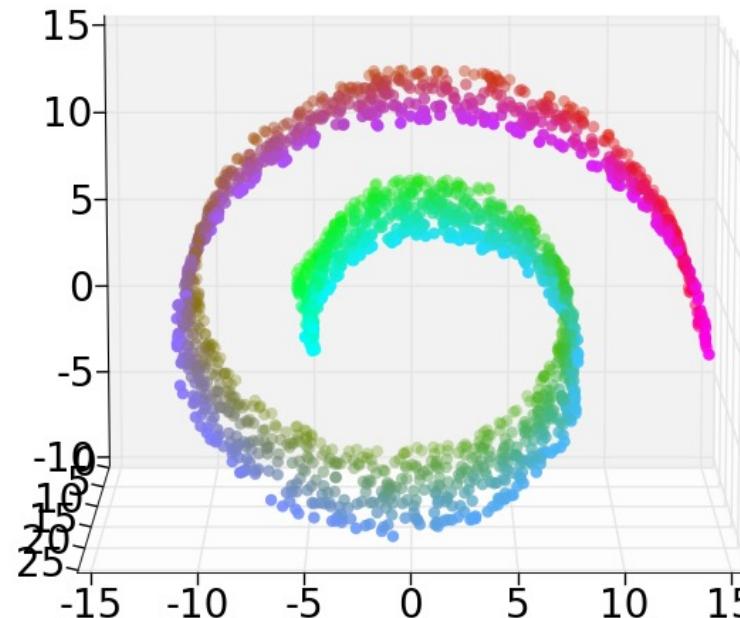


snow winter ice cold  
nature trees mountains  
white mountain



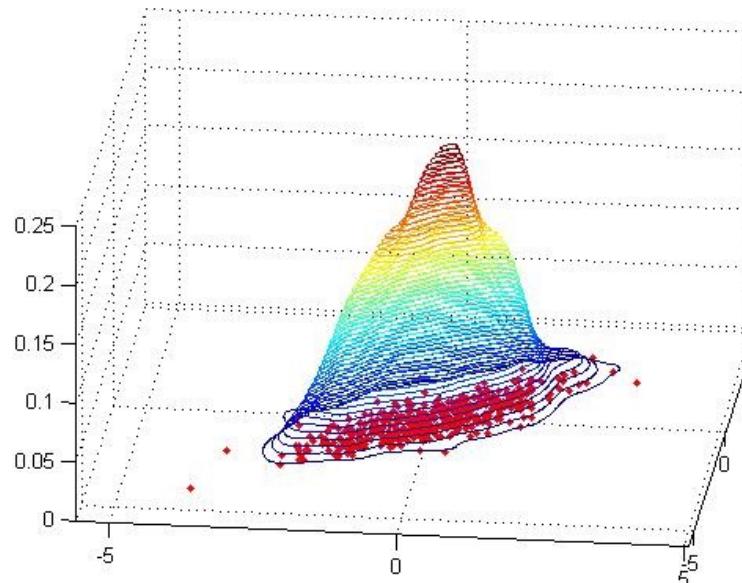
# Unsupervised Learning

- **Dimensionality reduction, manifold learning**
  - Discover a lower-dimensional surface on which the data lives



# Unsupervised Learning

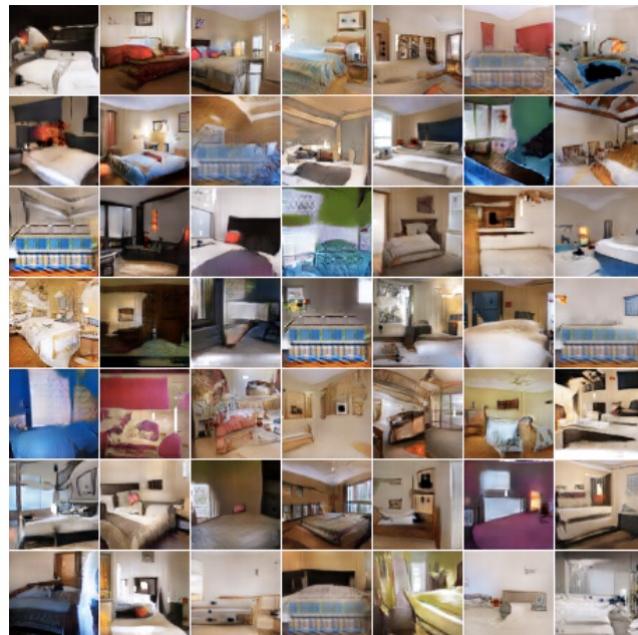
- **Density estimation**
  - Find a function that approximates the probability density of the data (i.e., value of the function is high for “typical” points and low for “atypical” points)
  - Can be used for **anomaly detection**



# Unsupervised Learning

- **Density estimation**
  - Produce samples from a data distribution that mimics the training set

“Bedroom”

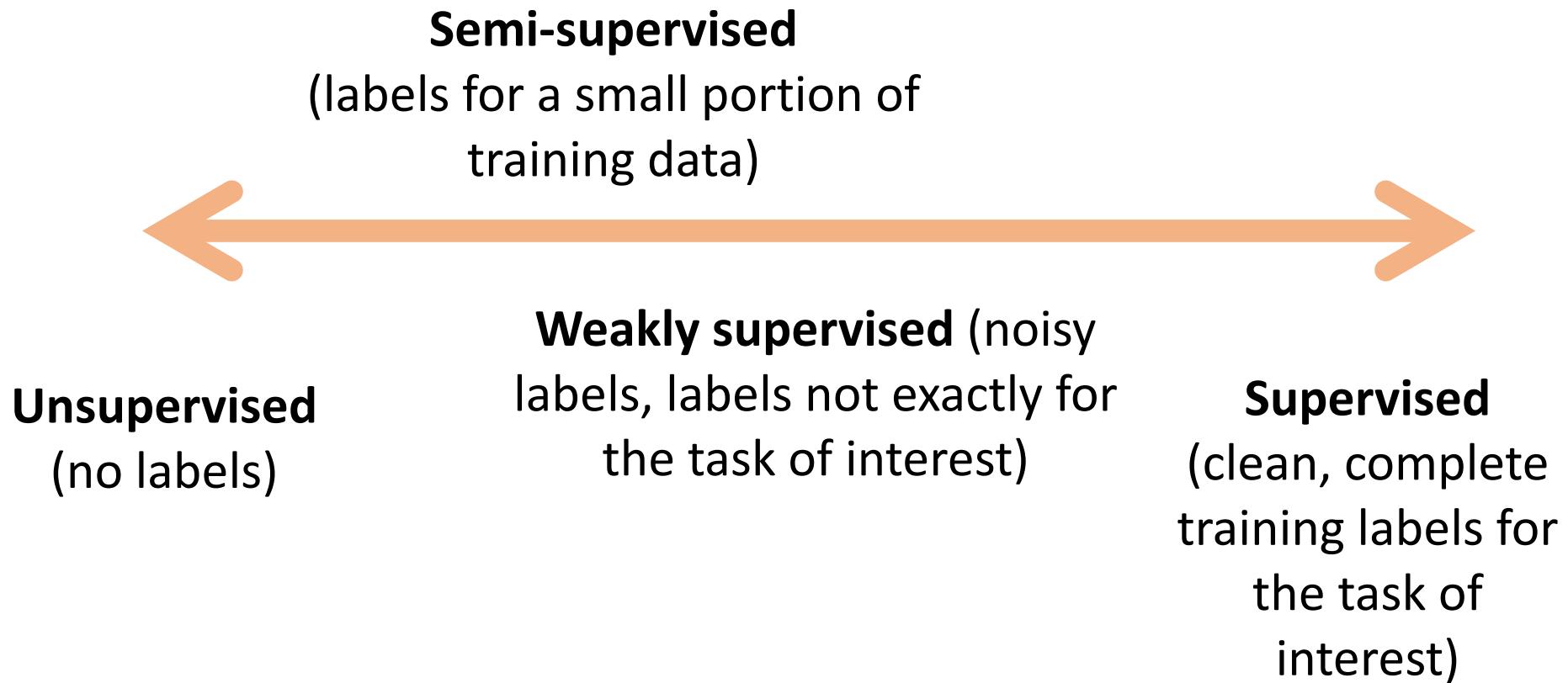


“Face”



Generative adversarial networks

# Continuum of supervision



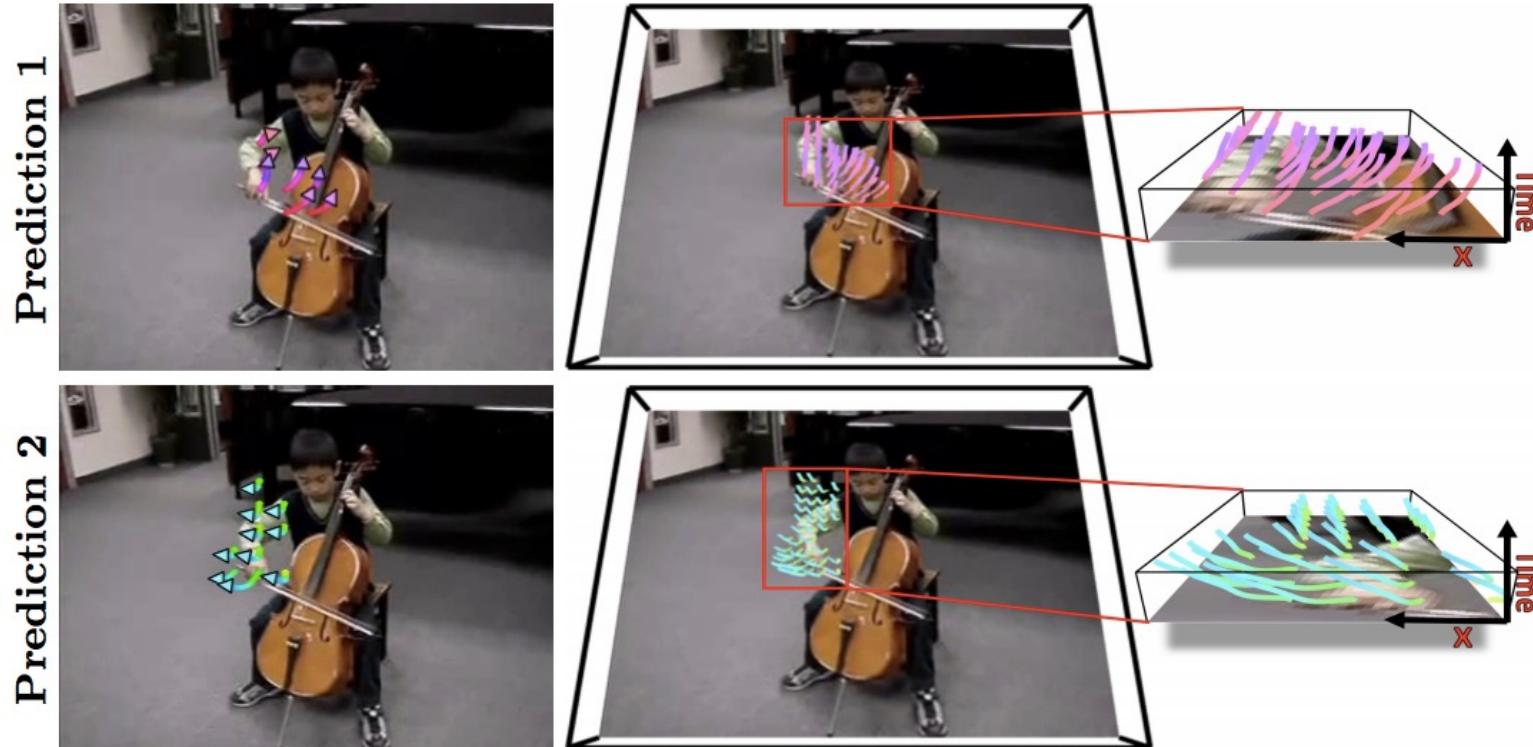
# Self-supervised or predictive learning

- Use part of the data to predict other parts of the data
  - Image colorization



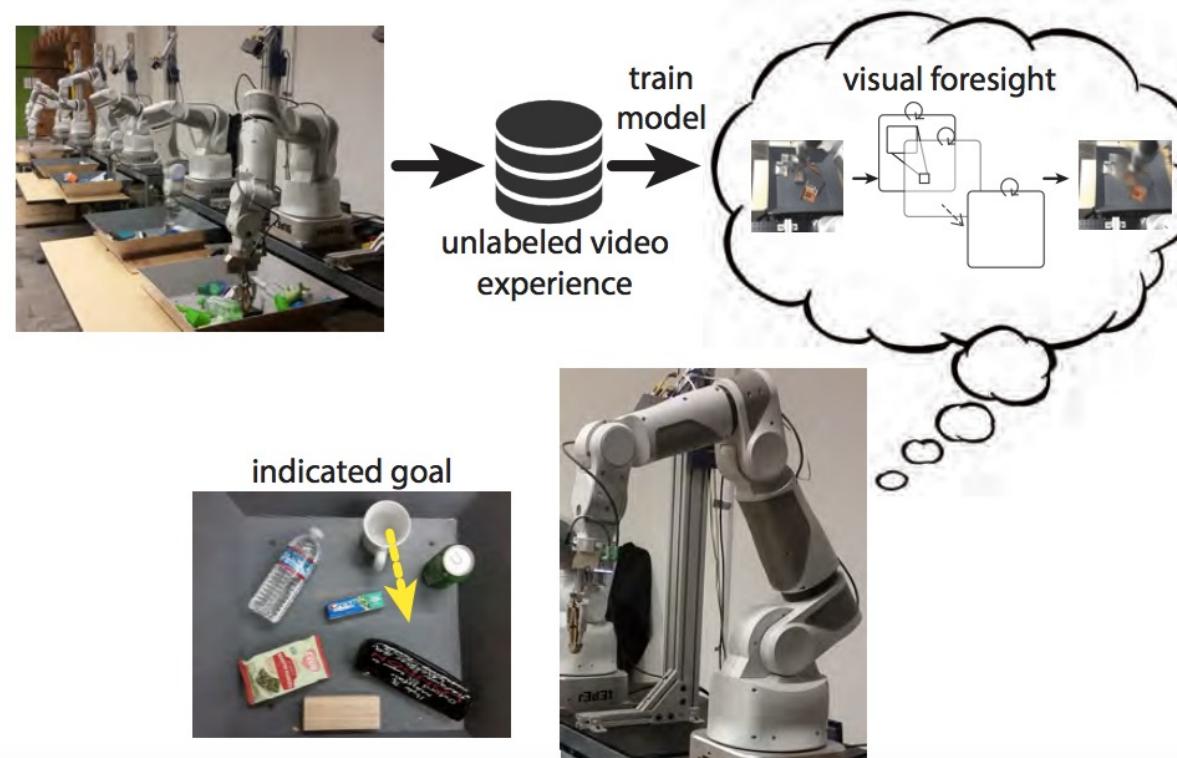
# Self-supervised or predictive learning

- Use part of the data to predict other parts of the data
  - Future prediction



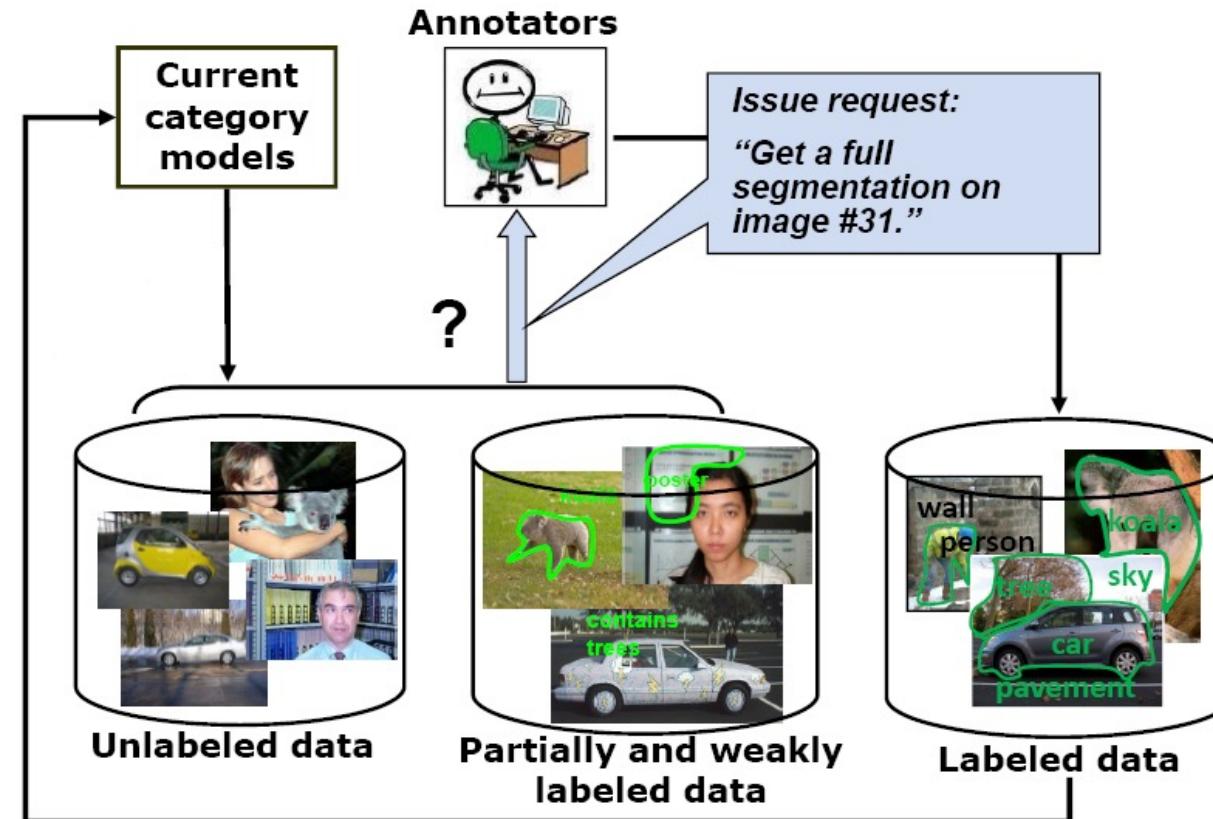
# Self-supervised or predictive learning

- Use part of the data to predict other parts of the data
  - Future prediction



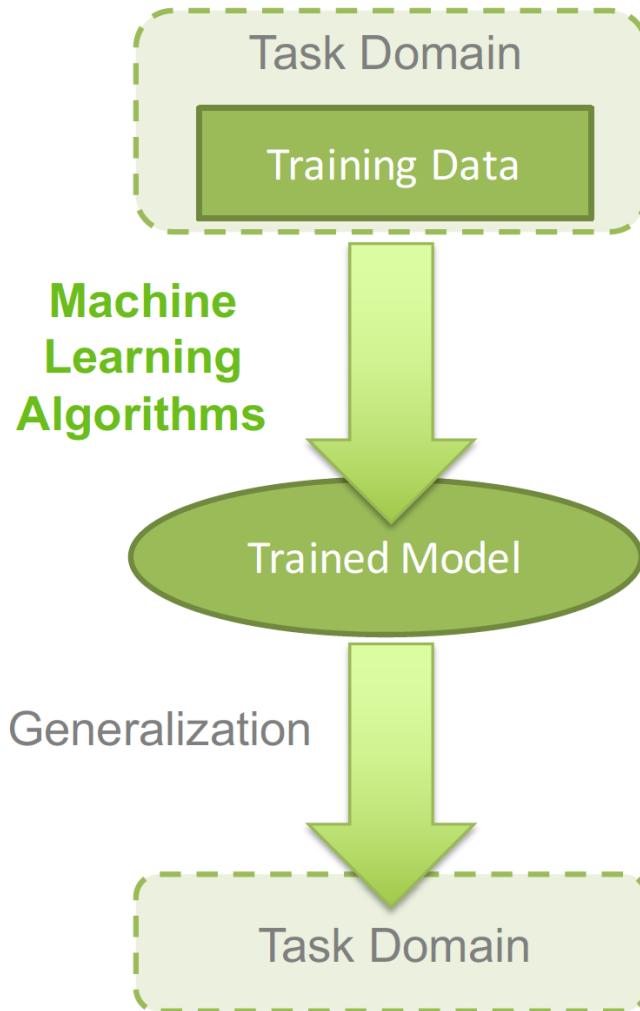
# Active learning

- The learning algorithm can choose its own training examples, or ask a “teacher” for an answer on selected inputs



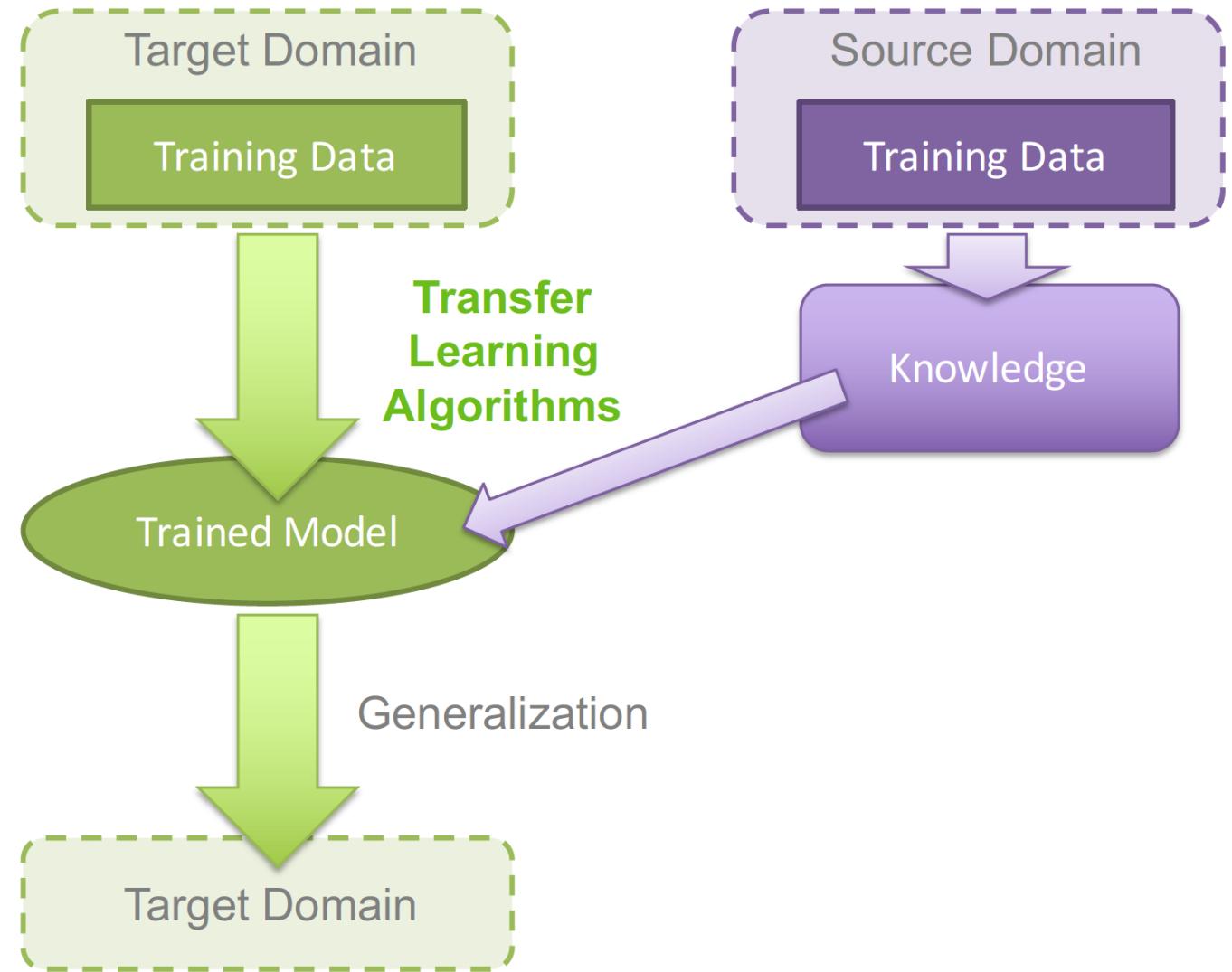
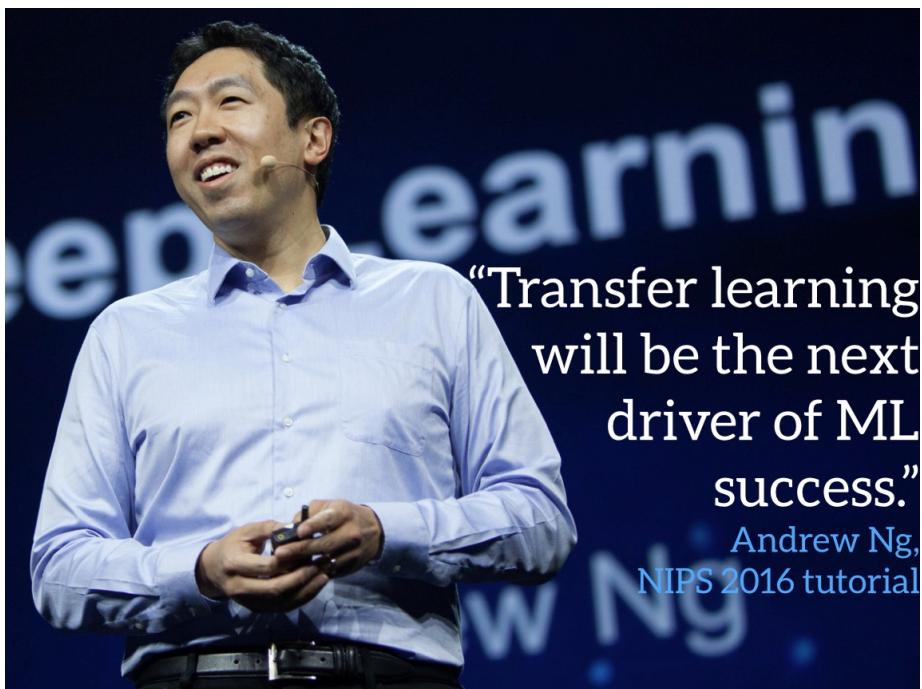
# Machine Learning for Single Task

- Elements of machine learning on single task
  - The problem (**task/domain**)
  - Training data
  - Learning algorithms
  - Trained model
  - Applying model on unseen data (**generalization**)

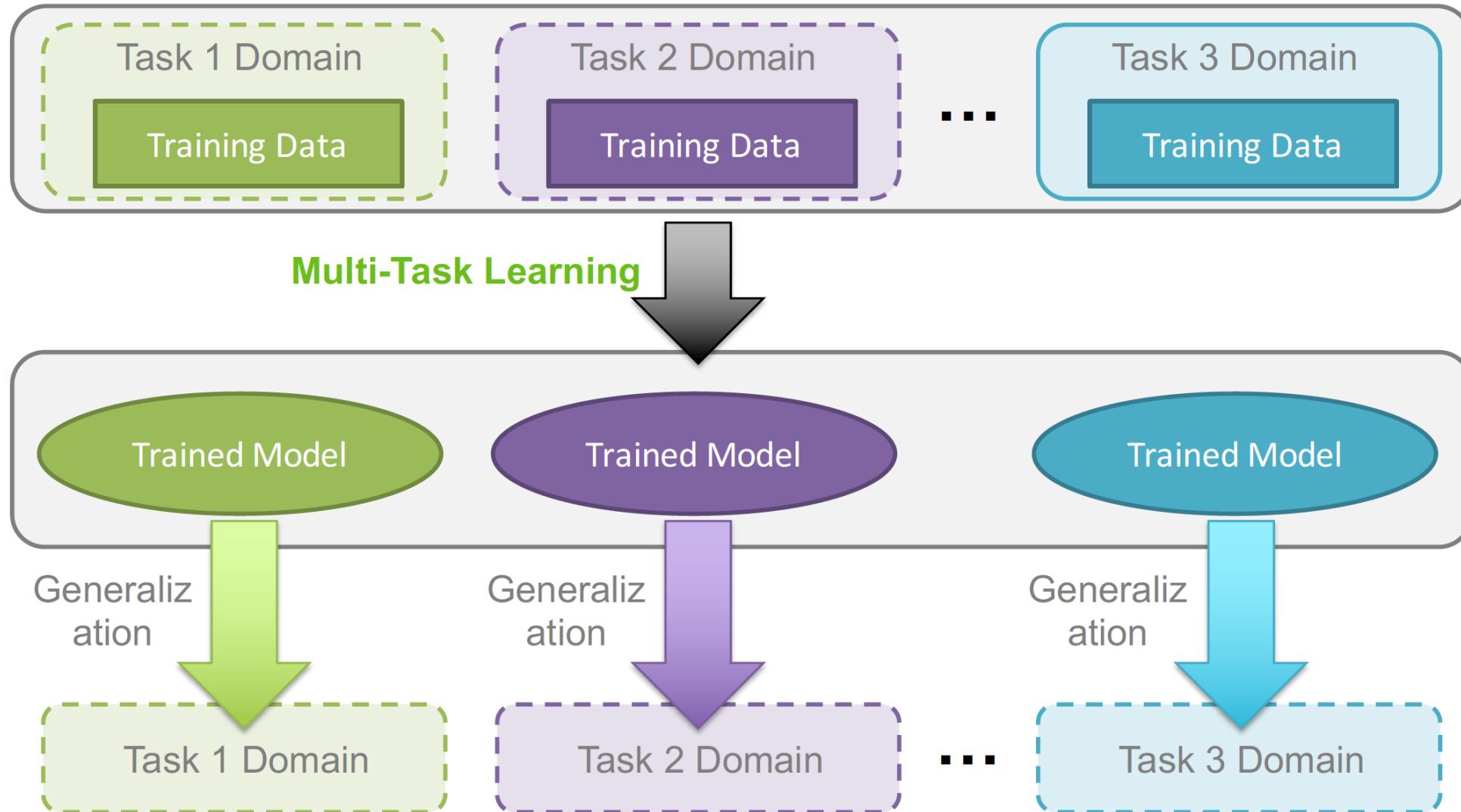


# Transfer Learning

Improve Learning New Task  
by Learned Task



# Multi-Task Learning



# Lifelong learning

## Read the Web

Research Project at Carnegie Mellon University

Home    Project Overview    Resources & Data    Publications    People

### NELL: Never-Ending Language Learning

Can computers learn to read? We think so. "Read the Web" is a research project that attempts to create a computer system that learns over time to read the web. Since January 2010, our computer system called NELL (Never-Ending Language Learner) has been running continuously, attempting to perform two tasks each day:

- First, it attempts to "read," or extract facts from text found in hundreds of millions of web pages (e.g., `playsInstrument(George_Harrison, guitar)`).
- Second, it attempts to improve its reading competence, so that tomorrow it can extract more facts from the web, more accurately.

So far, NELL has accumulated over 50 million candidate beliefs by reading the web, and it is considering these at different levels of confidence. NELL has high confidence in 2,033,557 of these beliefs — these are displayed on this website. It is not perfect, but NELL is learning. You can track NELL's progress below or [@cmunell on Twitter](#), browse and download its [knowledge base](#), read more about our [technical approach](#), or join the [discussion group](#).



Browse the Knowledge Base!



The University of Texas at Austin  
**Electrical and Computer  
Engineering**  
*Cockrell School of Engineering*