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

Spring 2022

INTRODUCTION TO COMPUTER VISION

Atlas Wang
Assistant Professor, The University of Texas at Austin

Visual Informatics Group@UT Austin
https://vita-group.github.io/

Example: Image classification

input **desired output**

apple

pear

tomato

cow

dog

horse

Training data:

apple

pear

tomato

cow

dog

horse

Example 2: Spam filter

Dear Sir.

First, I must solicit your confidence in this transaction, this is by virture of its nature as being utterly confidencial 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, Iknow 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.

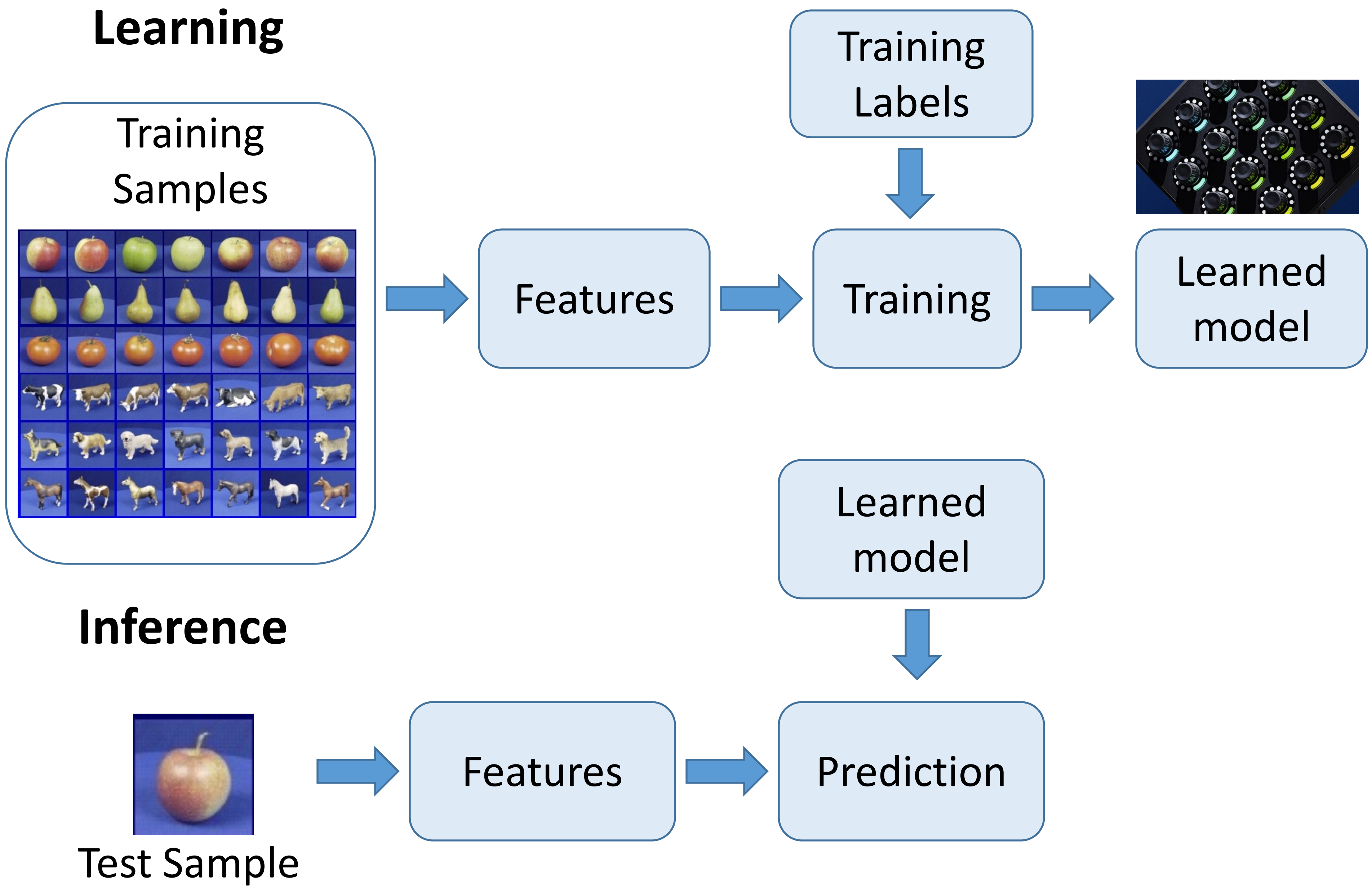
The basic *supervised learning* framework

$$y = f(\mathbf{x})$$

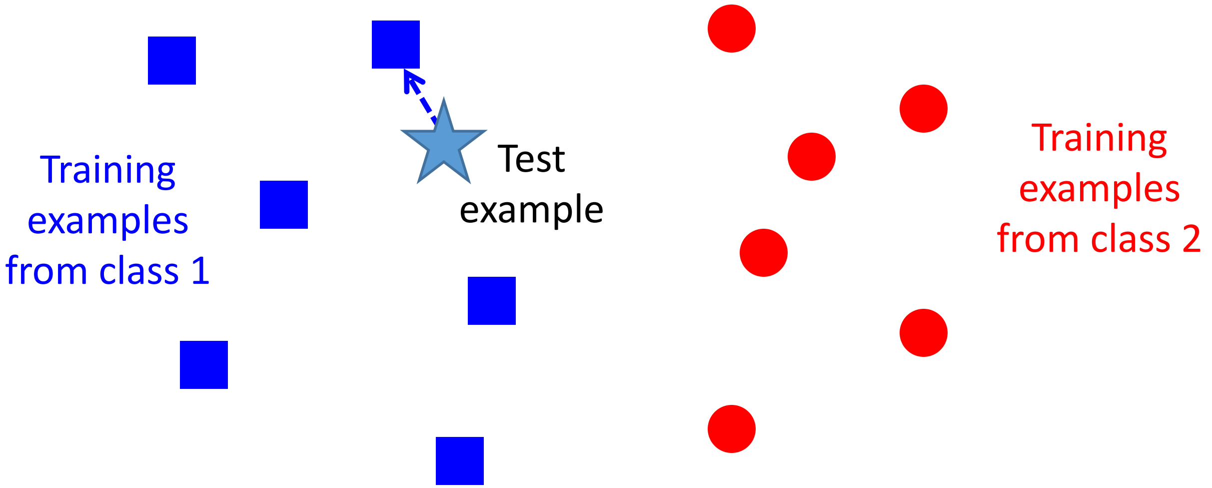
output — classification function — input

- **Learning:** given a *training set* of labeled examples $\{(\mathbf{x}_1,y_1), \ldots, (\mathbf{x}_N,y_N)\}$, estimate the parameters of the prediction function f
- **Inference:** apply f to a never before seen *test example* $\mathbf{x}$ and output the predicted value $y = f(\mathbf{x})$

Learning and inference pipeline



Nearest neighbor classifier

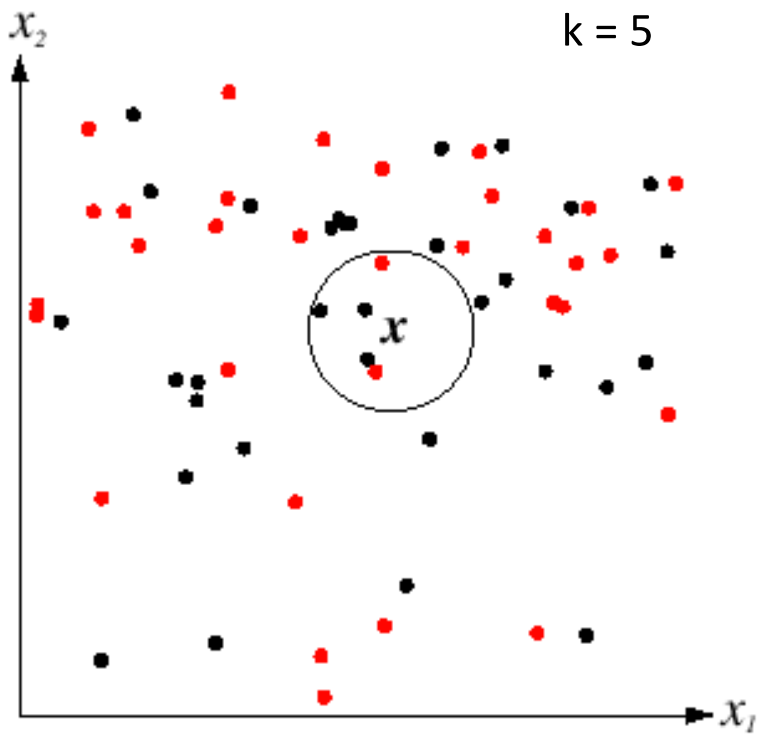


f(**x**) = label of the training example nearest to **x**

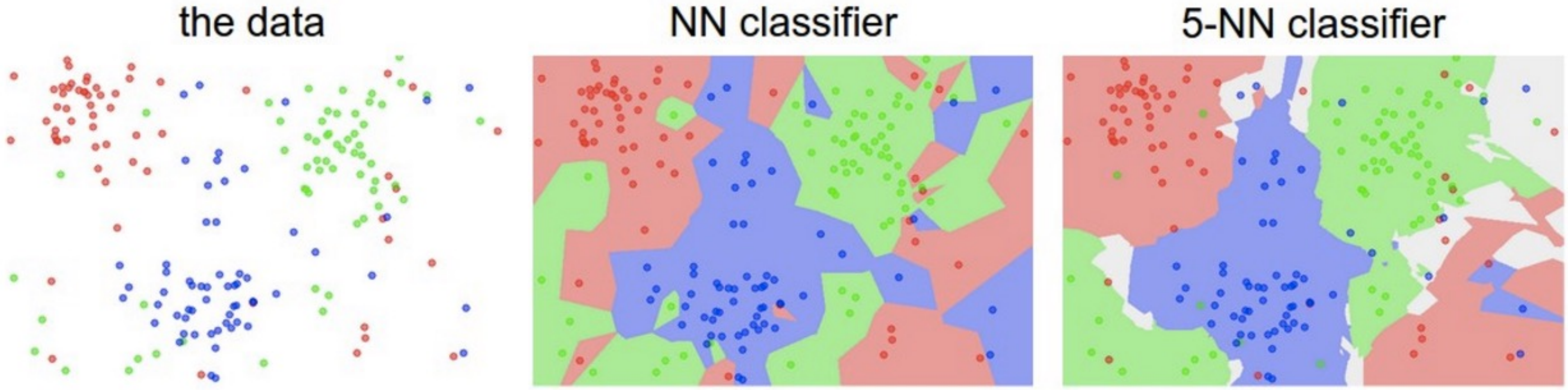
- All we need is a distance function for our inputs
- No training required!

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



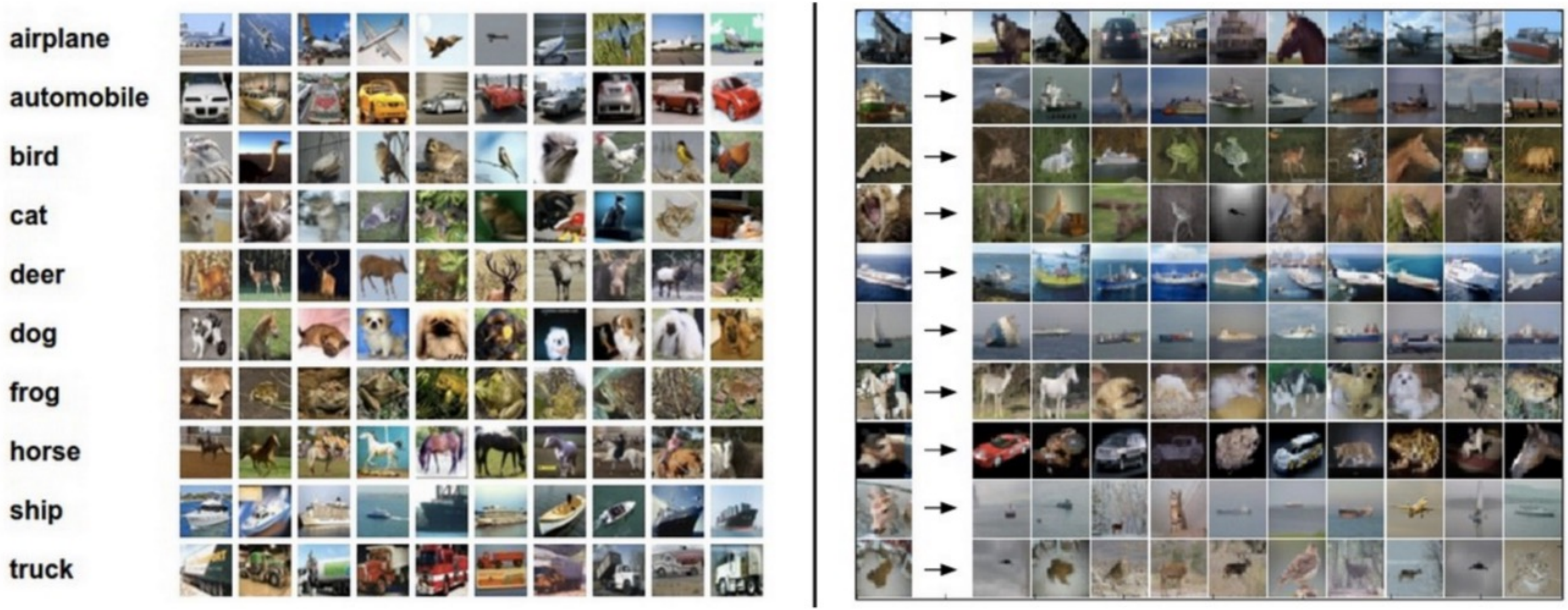
K-nearest neighbor classifier



- Which classifier is more robust to *outliers*?

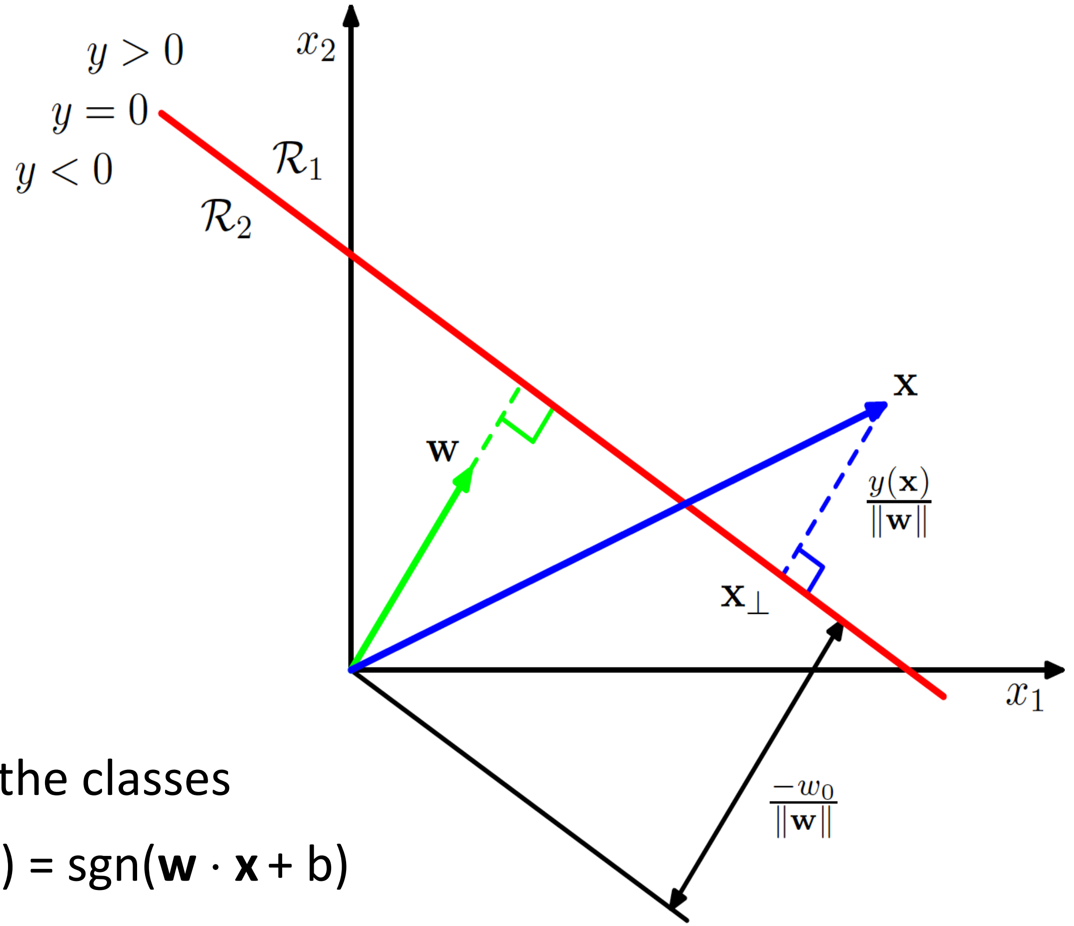
Credit: Andrej Karpathy, http://cs231n.github.io/classification/

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

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

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 ϕ so that

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

- $\sigma(.)$ 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

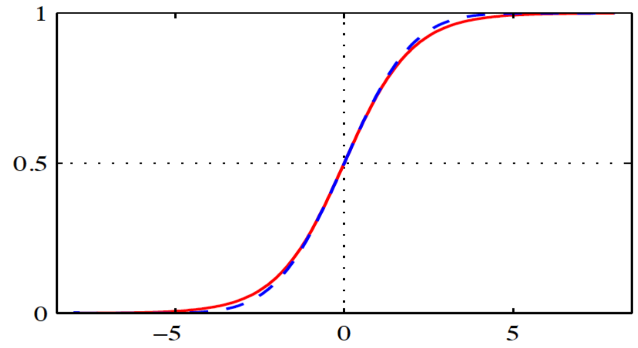
- Definition of **Logistic Sigmoid** function $\sigma(a)$:

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

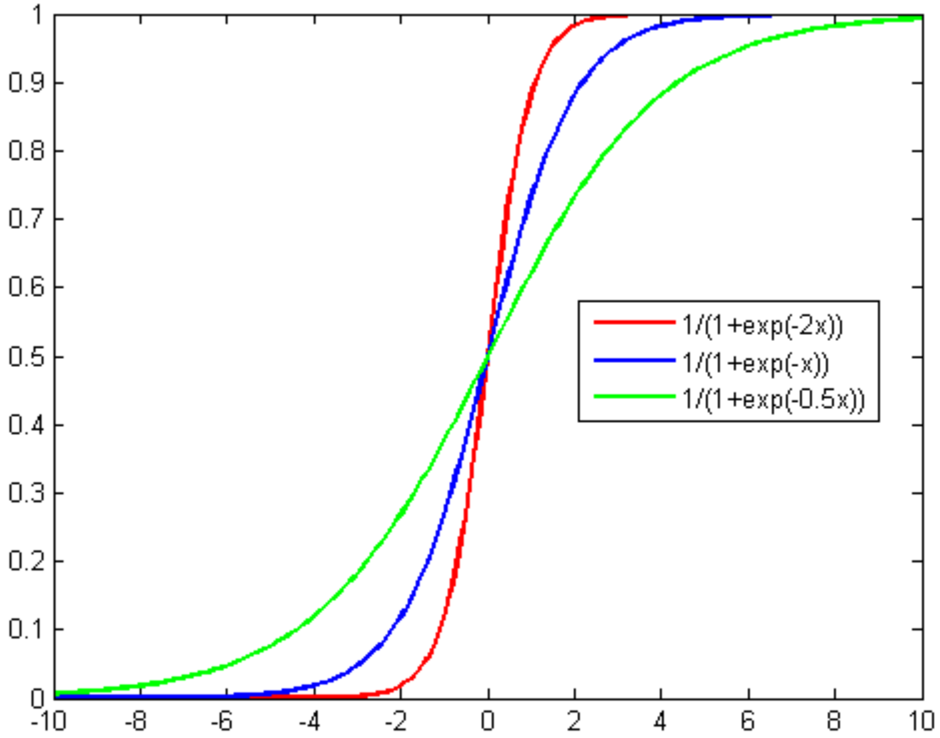
- Properties of Logistic Sigmoid function $\sigma(a)$:

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

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

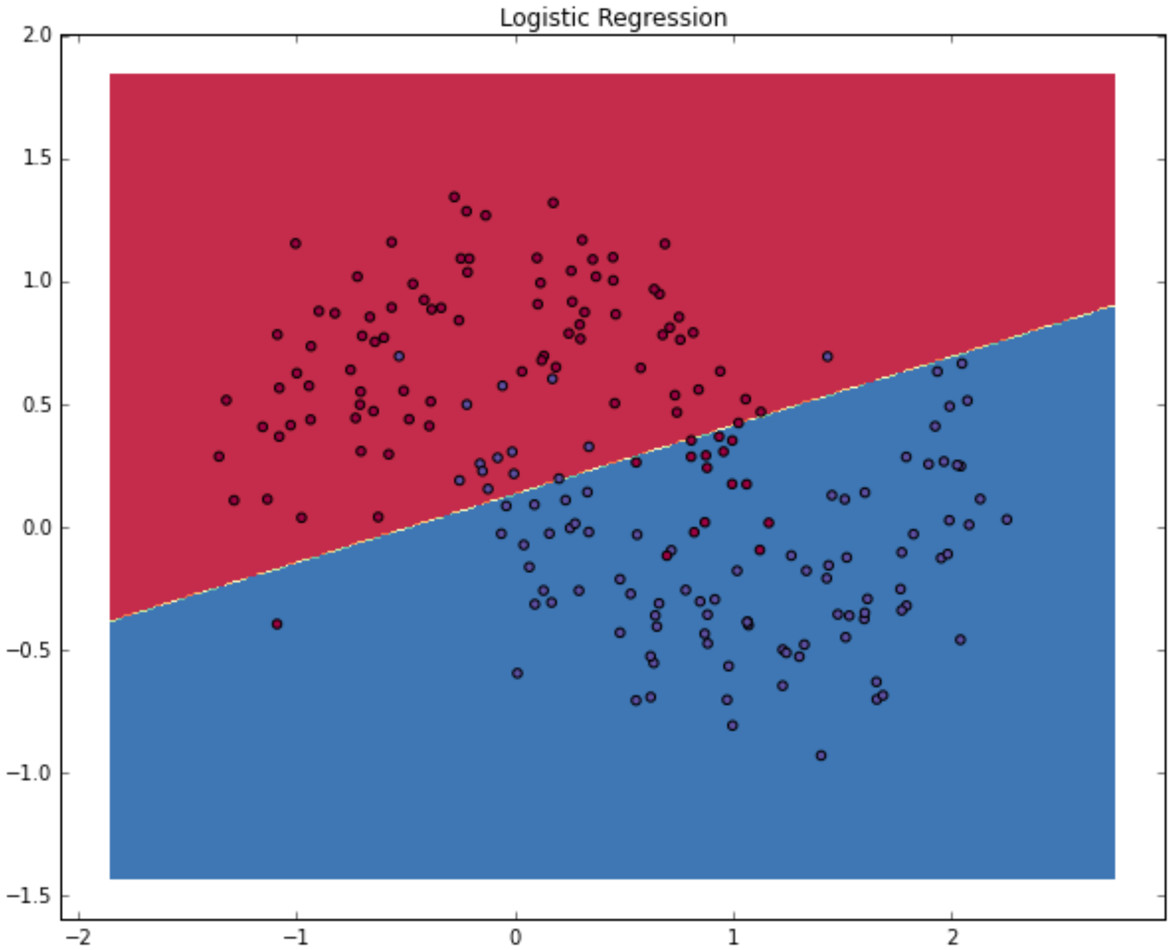


Plot of the logistic sigmoid function $\sigma(a)$ (Red Line)



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 $n = 1, 2, , N$, the likelihood function is

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

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

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

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

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

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 **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?

Softmax Regression

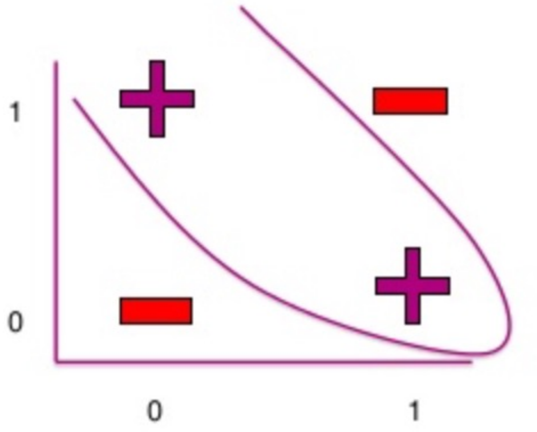
- The Multi-Class version of Logistic Regression (popular in deep learning)

(Reference: http://ufldl.stanford.edu/tutorial/supervised/SoftmaxRegression/)

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

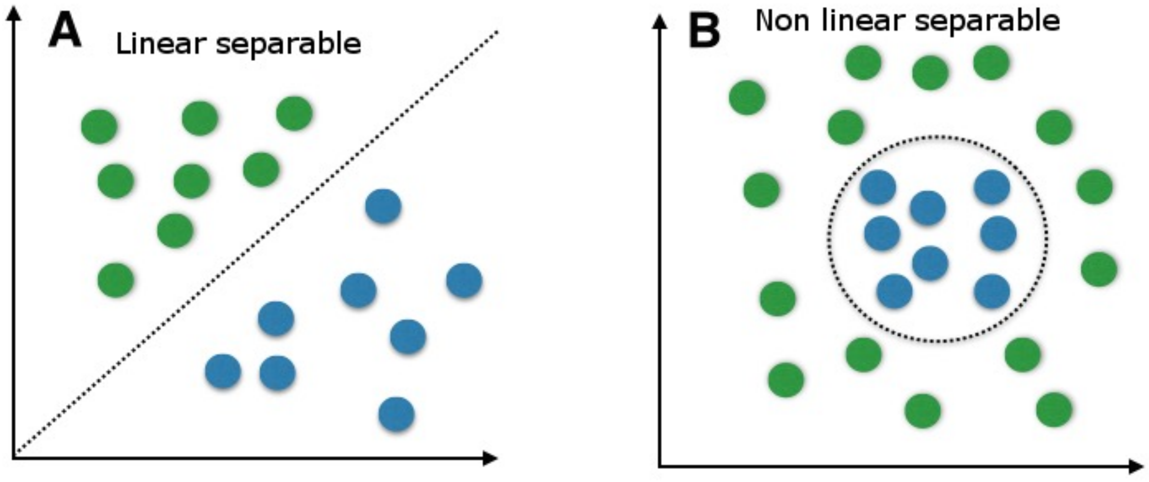
$$h_\theta(x) = \begin{bmatrix} P(y = 1|x;\theta) \\ P(y = 2|x;\theta) \\ \vdots \\ P(y = K|x;\theta) \end{bmatrix} = \frac{1}{\sum_{j=1}^{K} \exp(\theta^{(j)\top} x)} \begin{bmatrix} \exp(\theta^{(1)\top} x) \\ \exp(\theta^{(2)\top} x) \\ \vdots \\ \exp(\theta^{(K)\top} x) \end{bmatrix}$$

Limitation of Linear Classifiers



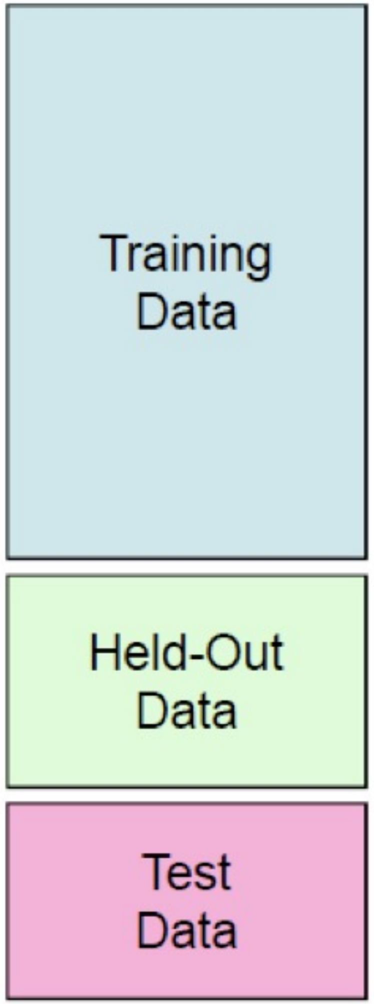
XOR

Need non-linear features



Experimentation cycle

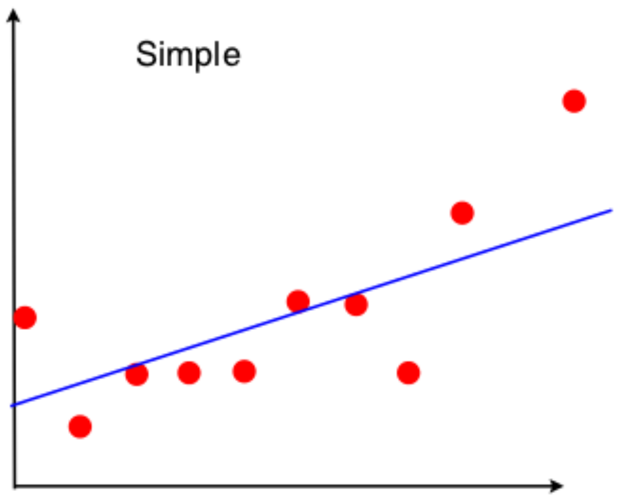
- Learn *parameters* on the *training set*
- Tune *hyperparameters* (implementation choices) on the *held out validation set*
- Evaluate performance on the *test set*
- Do not peek at the test set!
- *Generalization* and *overfitting*
  - Want classifier that does well on never before seen data
  - Overfitting: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

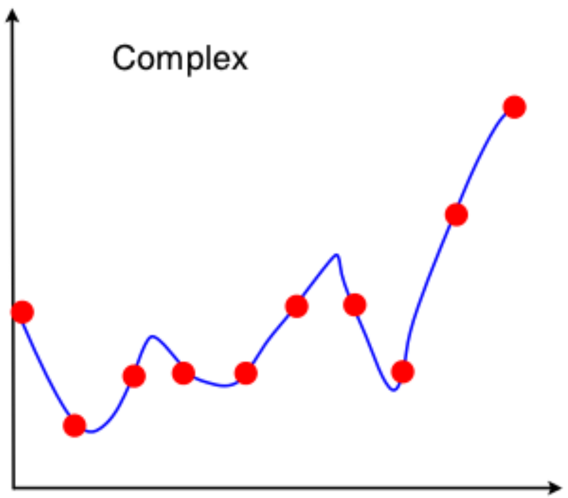


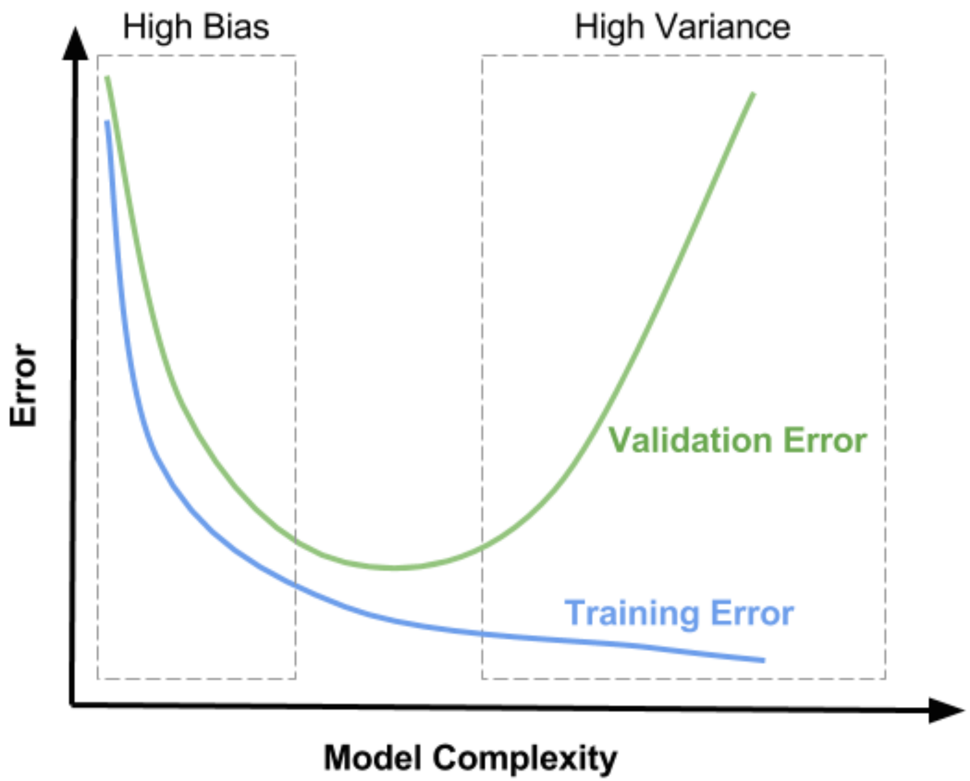
Bias-Variance Tradeoff

- 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**).

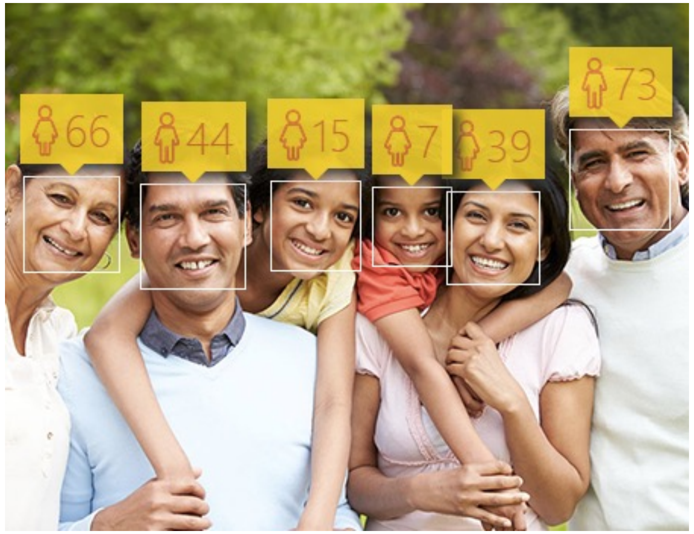
Bias-Variance Tradeoff







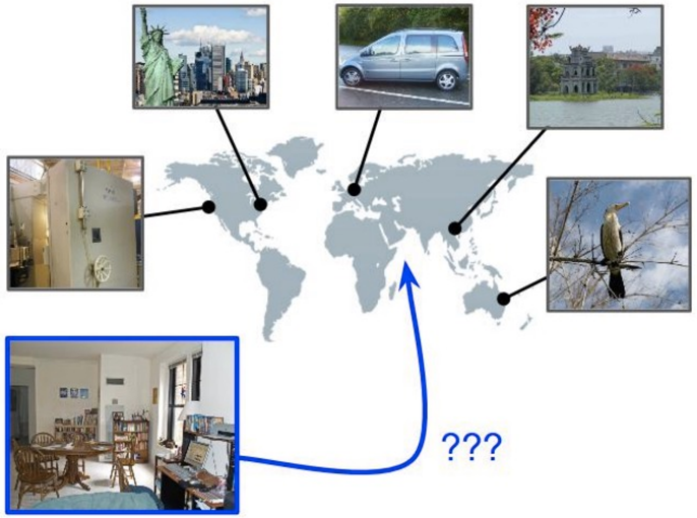
Beyond classification: Regression



[Age estimation](#)

[When was that made?](#)

[IM2GPS](#)



Beyond classification: Structured prediction

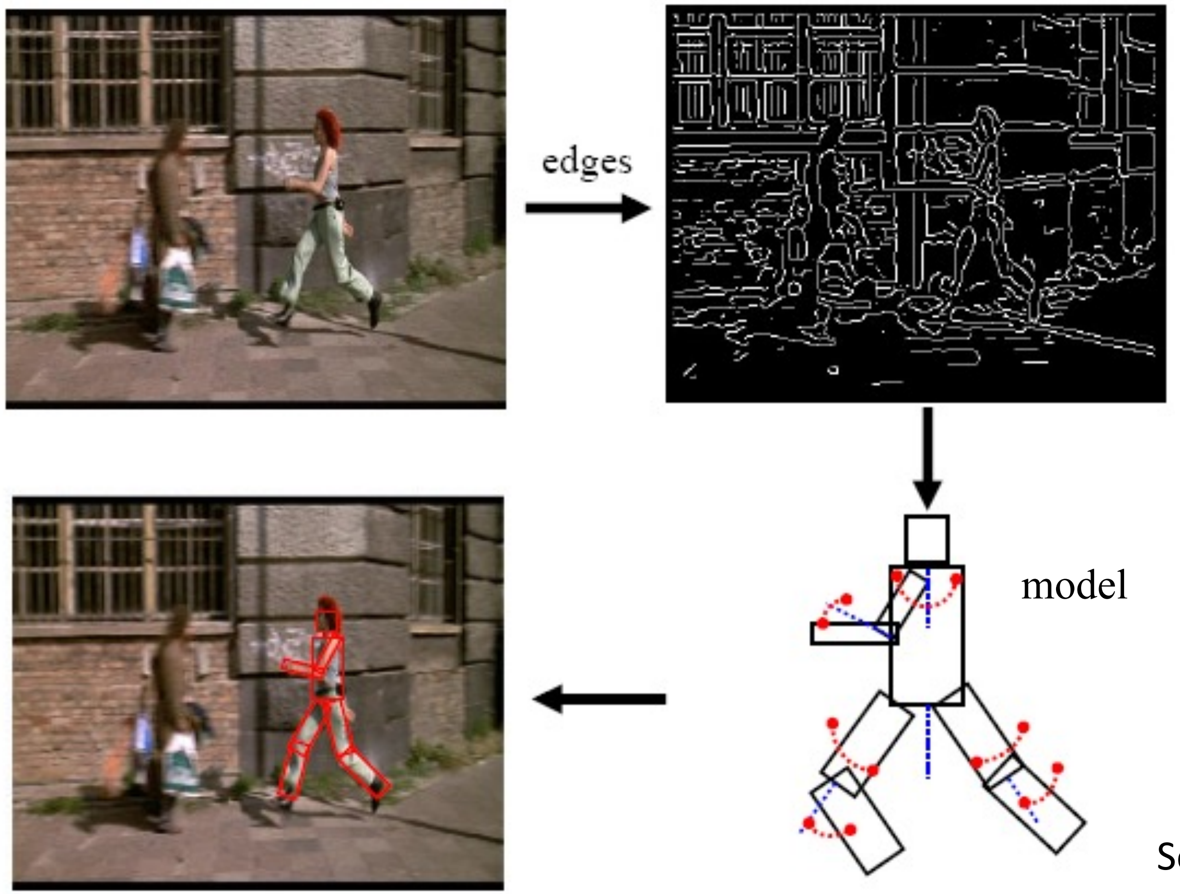
Image

Word

Source: B. Taskar

Structured Prediction

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



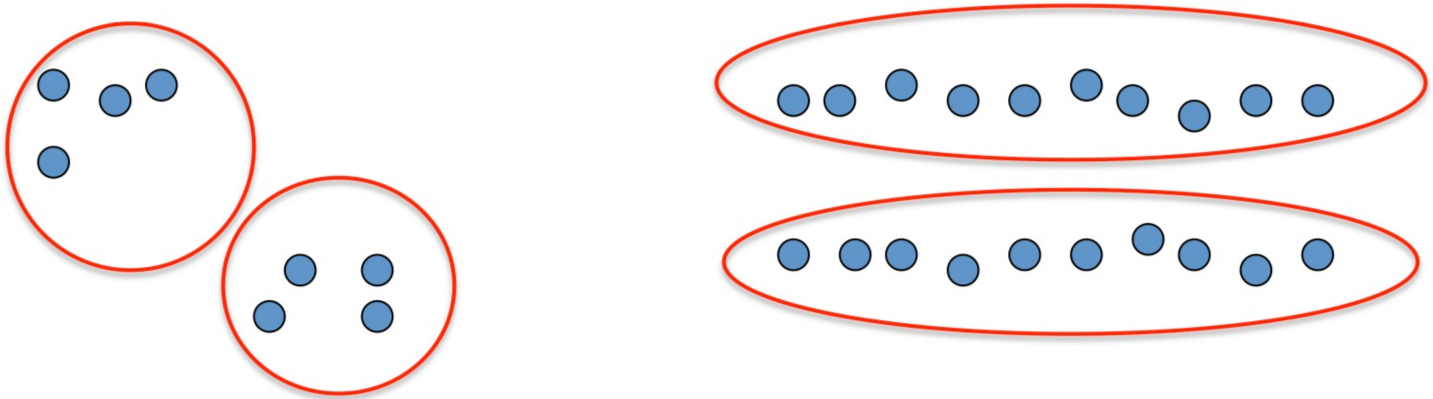
Source: D. Ramanan

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



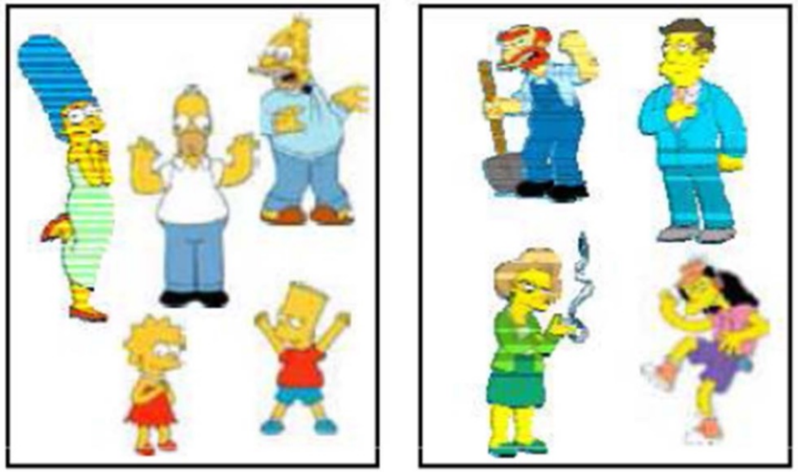
- What could "similar" mean?
  - One option: small Euclidean distance (squared)

$$\text{dist}(\vec{x}, \vec{y}) = ||\vec{x} - \vec{y}||_2^2$$

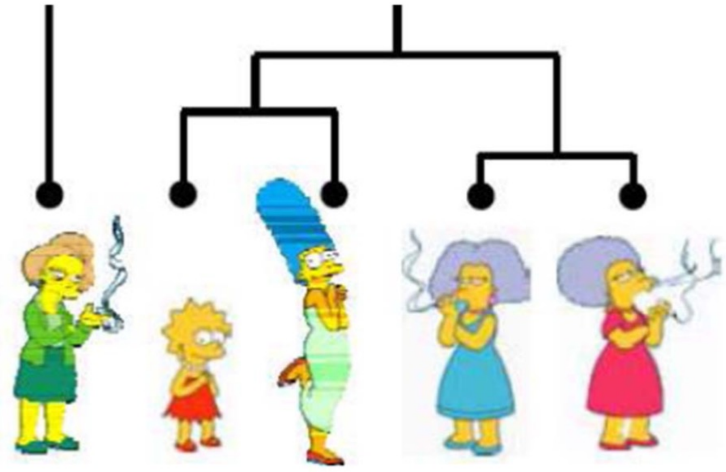
  - Clustering results are crucially dependent on the measure of similarity (or distance) between "points" to be clustered

Clustering Algorithms

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

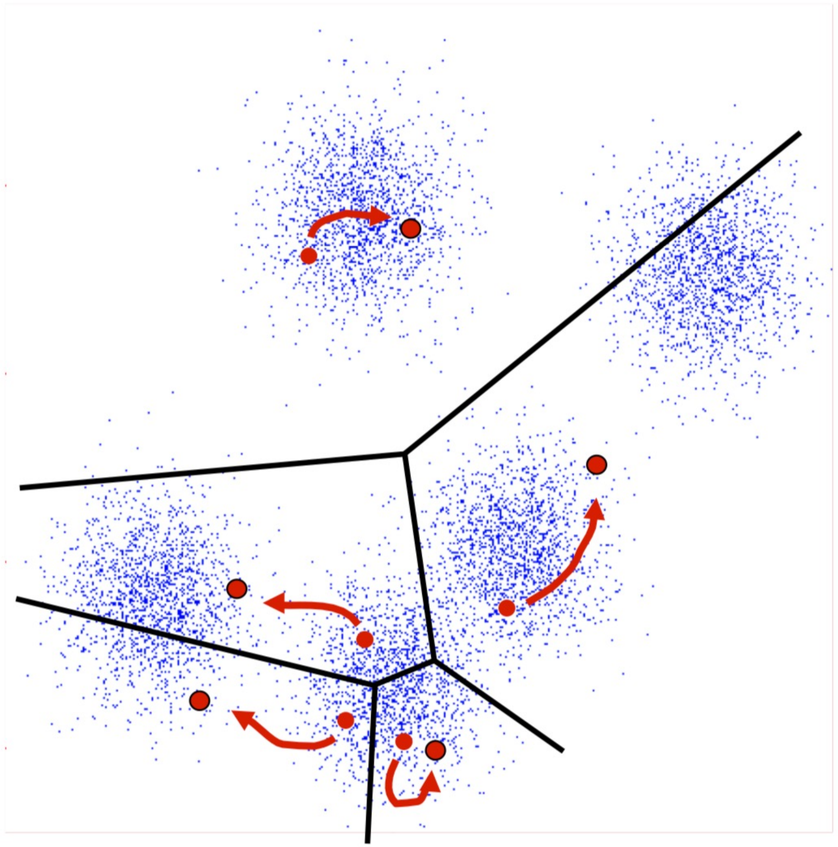


- Hierarchical algorithms
  - Bottom up – agglomerative
  - Top down – divisive



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 μ, optimize *C:*

Step 1 of kmeans

$$\min_{C} \sum_{i=1}^{k} \sum_{x \in C_i} |x - \mu_i|^2 = \min_{c} \sum_{i}^{n} \left|x_i - \mu_{x_i}\right|^2$$

2. Fix *C,* optimize μ:

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

   – Take partial derivative of μ_i and set to zero, we have

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

Step 2 of kmeans

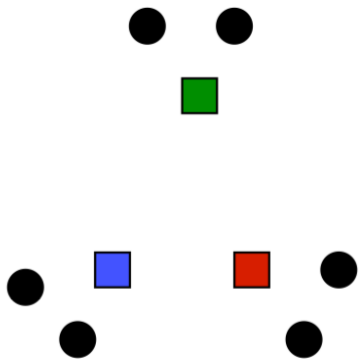
Kmeans takes an alternating optimization approach, each step is guaranteed to decrease the objective – thus guaranteed to converge

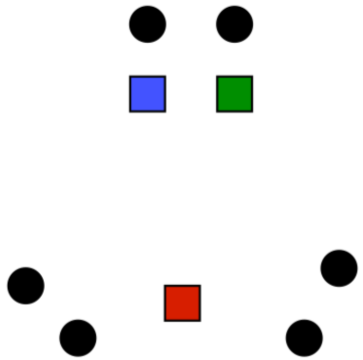
K-Means is Fragile

- K-means **algorithm** is a heuristic
  - Requires initial means
  - It does matter what you pick!

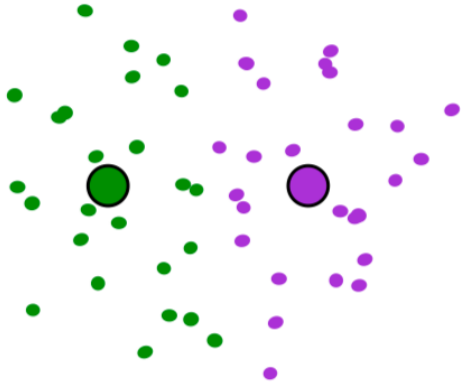
  - What can go wrong?

  - Various schemes for preventing this kind of thing: variance-based split / merge, initialization heuristics

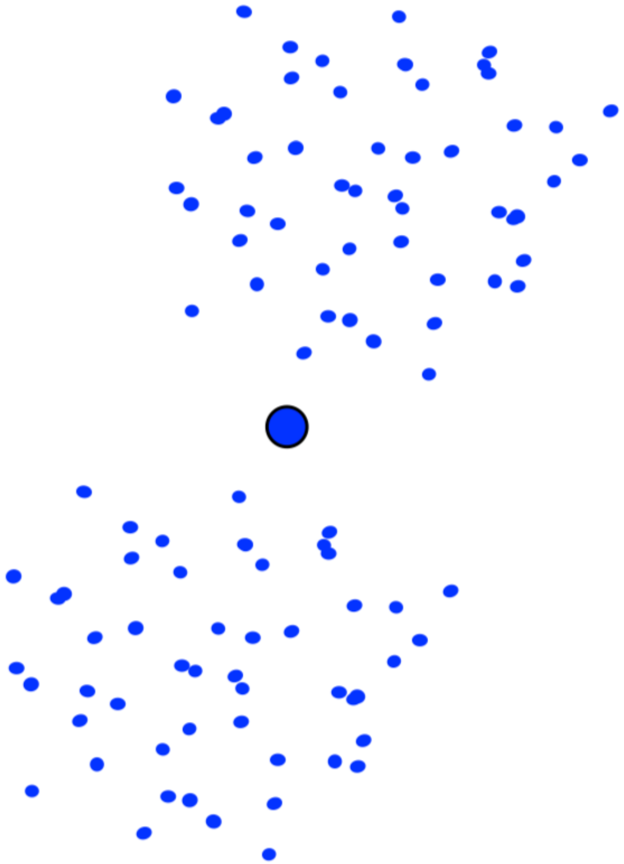




Stuck at Poor Local Minimum

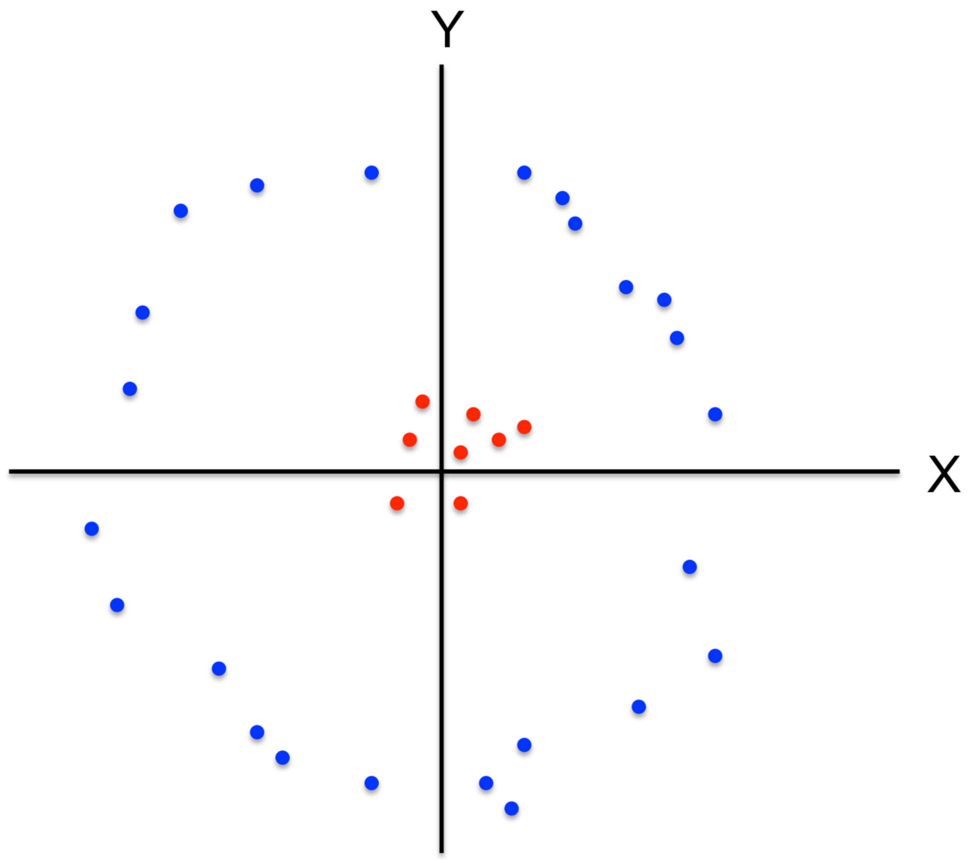


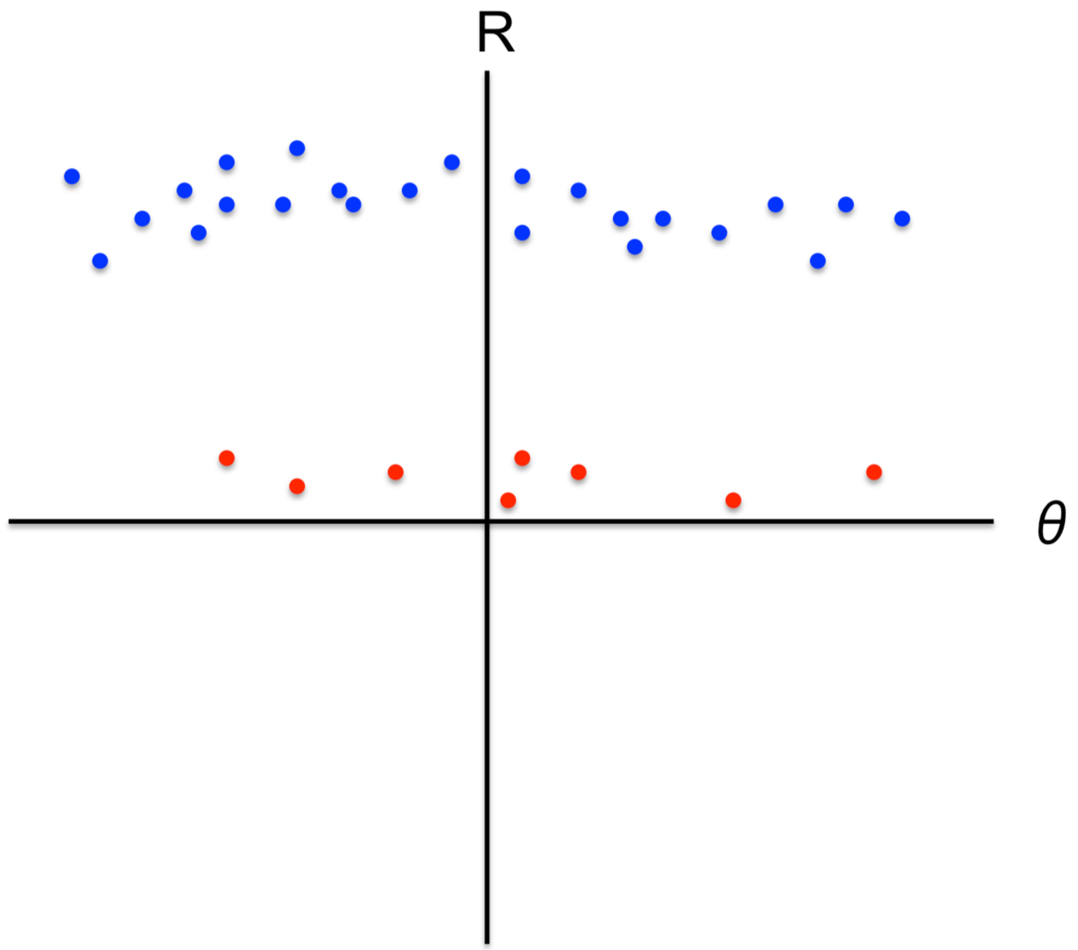
Would be better to have
one cluster here



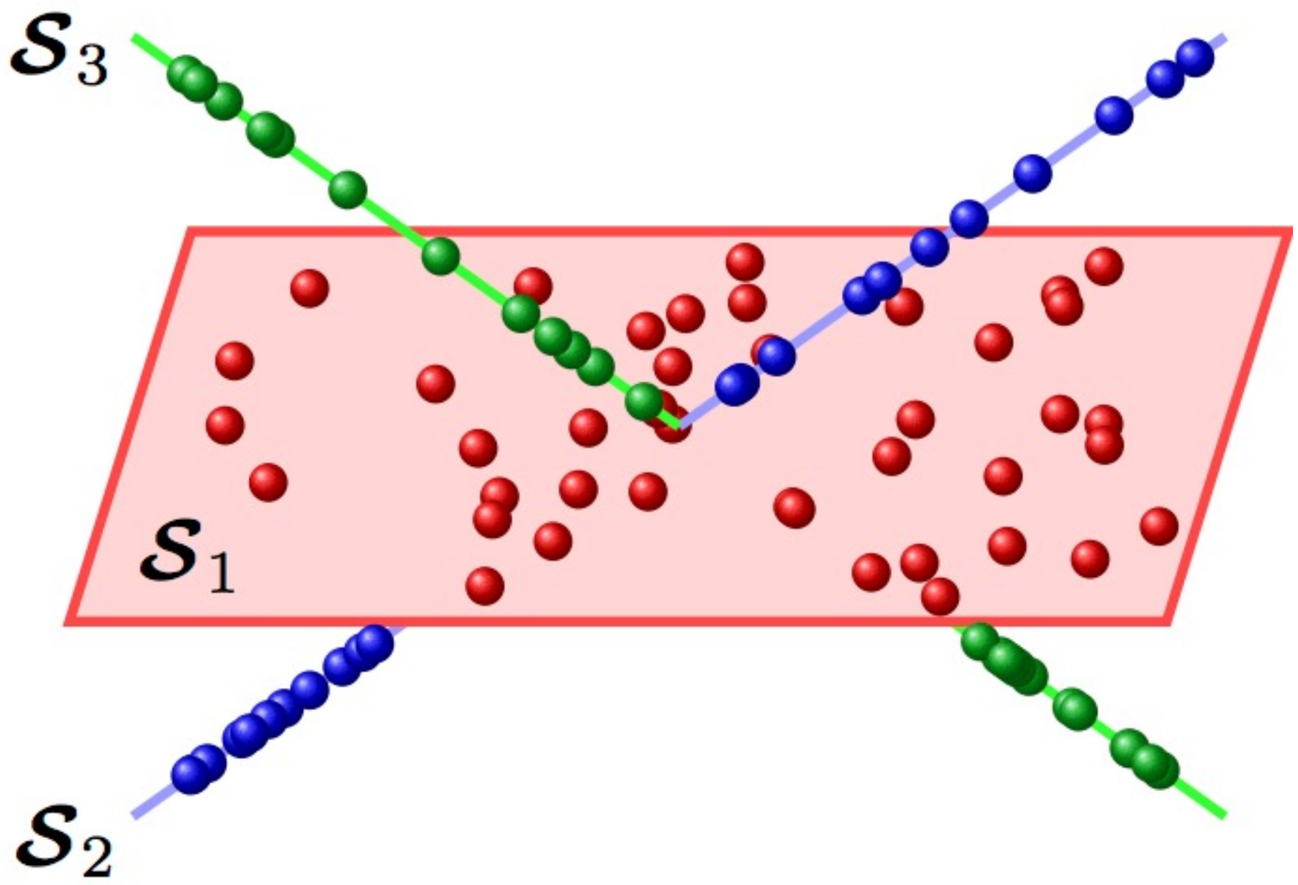
… and two clusters here

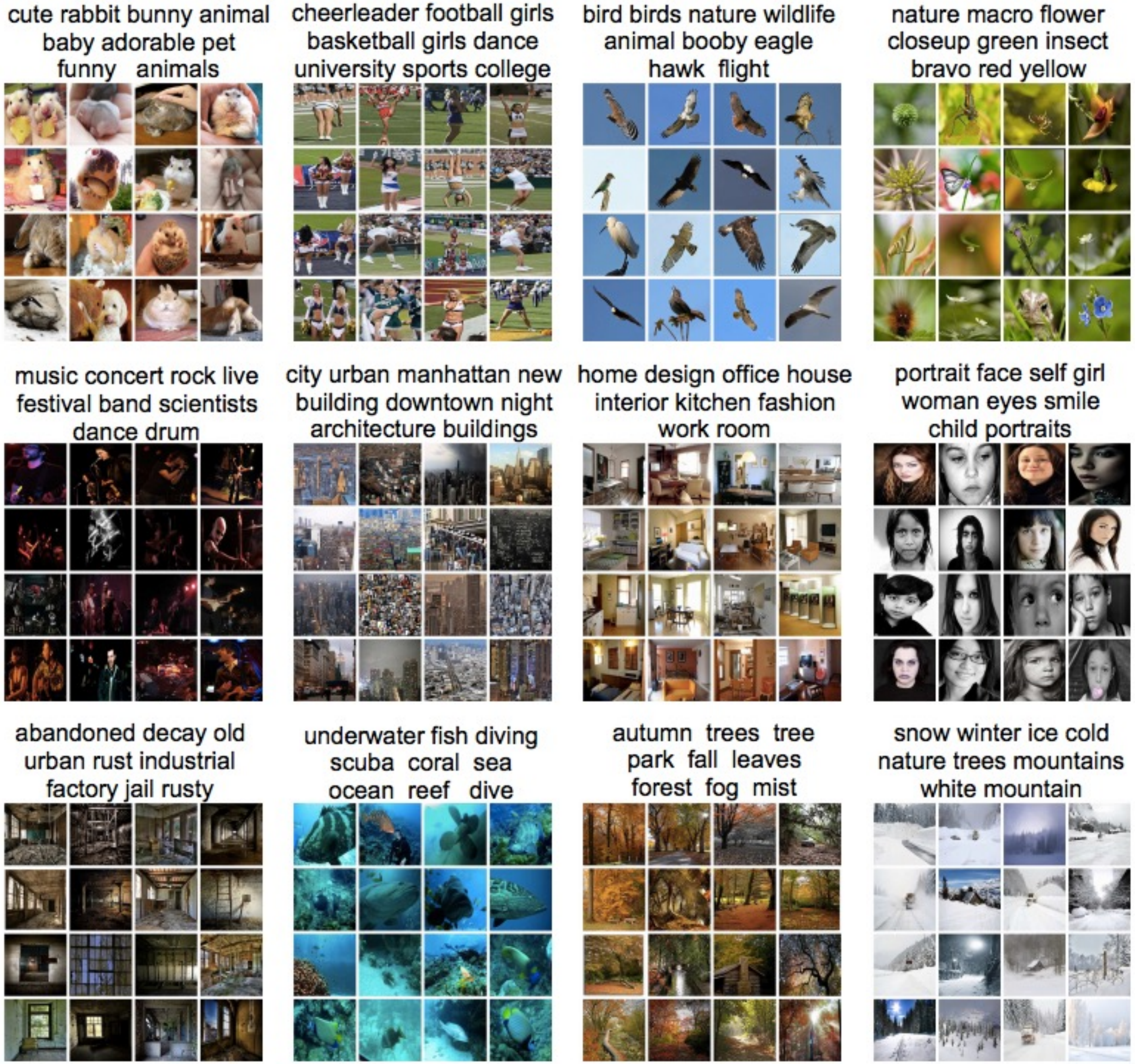
Euclidean Distance Might Not be Proper





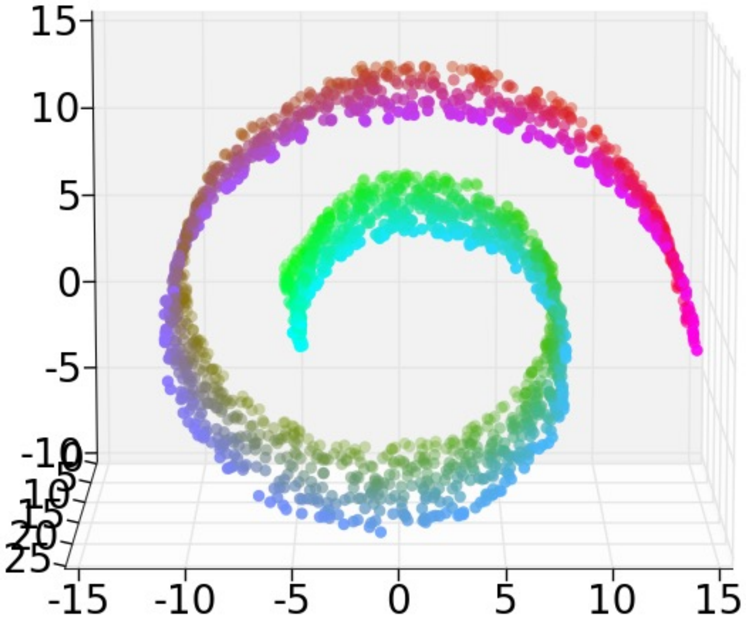
Do not underestimate clustering!!




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

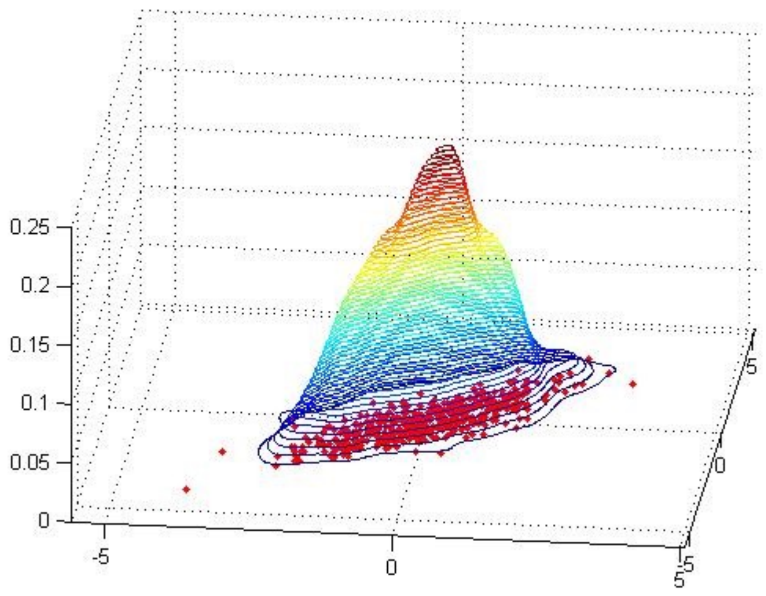
Unsupervised Learning

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



Unsupervised Learning

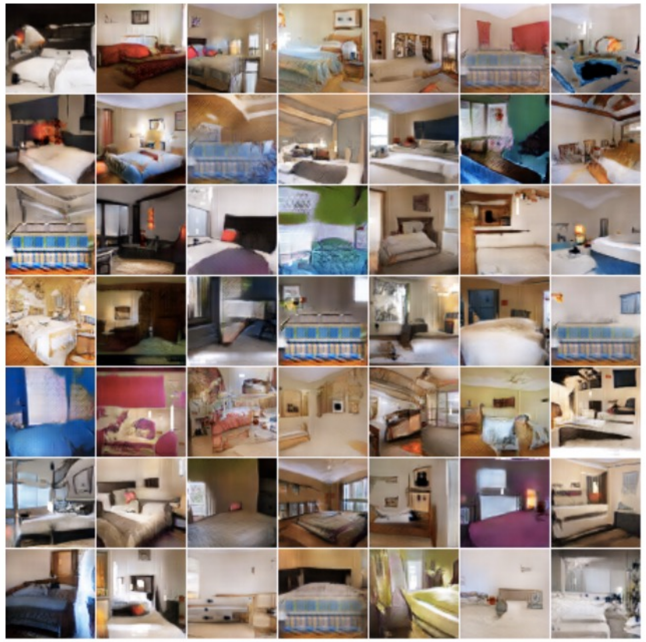
- **Density estimation**
  - Find a function that approximates the probability density of the data (i.e., value of the function is high for “typical” points and low for “atypical” points)
  - Can be used for **anomaly detection**



Unsupervised Learning

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

"Bedroom"



"Face"

Generative adversarial networks

Continuum of supervision

Semi-supervised
(labels for a small portion of training data)

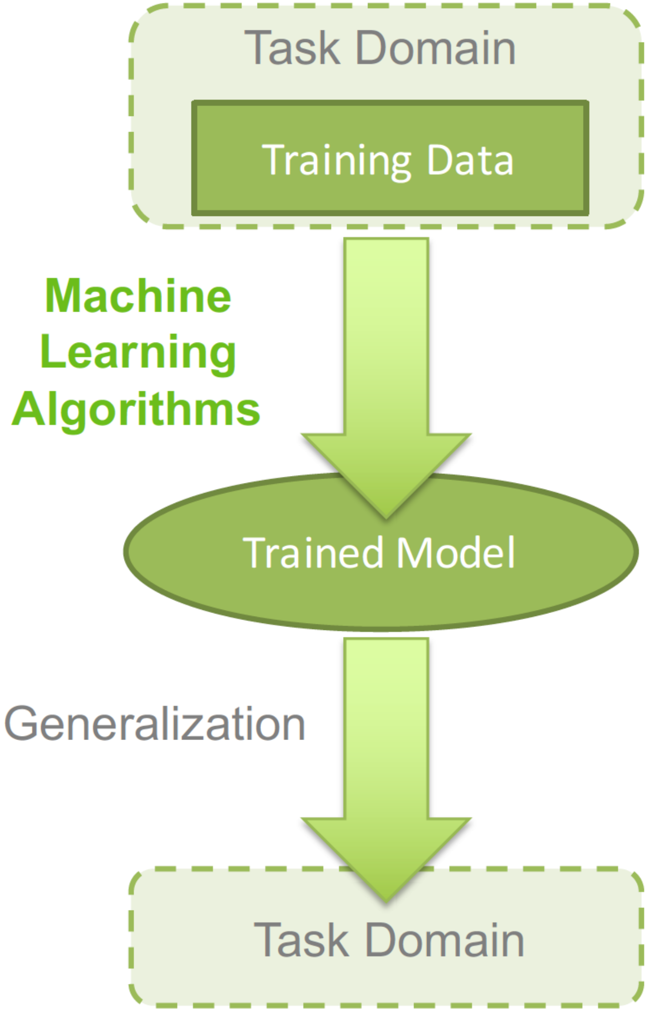
Unsupervised
(no labels)

Weakly supervised (noisy labels, labels not exactly for the task of interest)

Supervised
(clean, complete training labels for the task of interest)

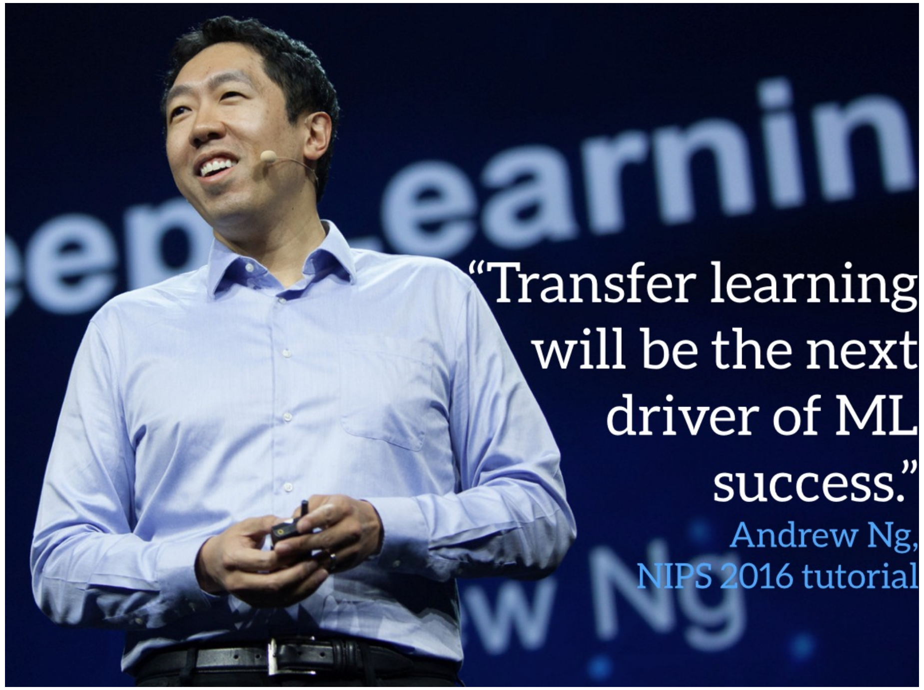
Machine Learning for Single Task

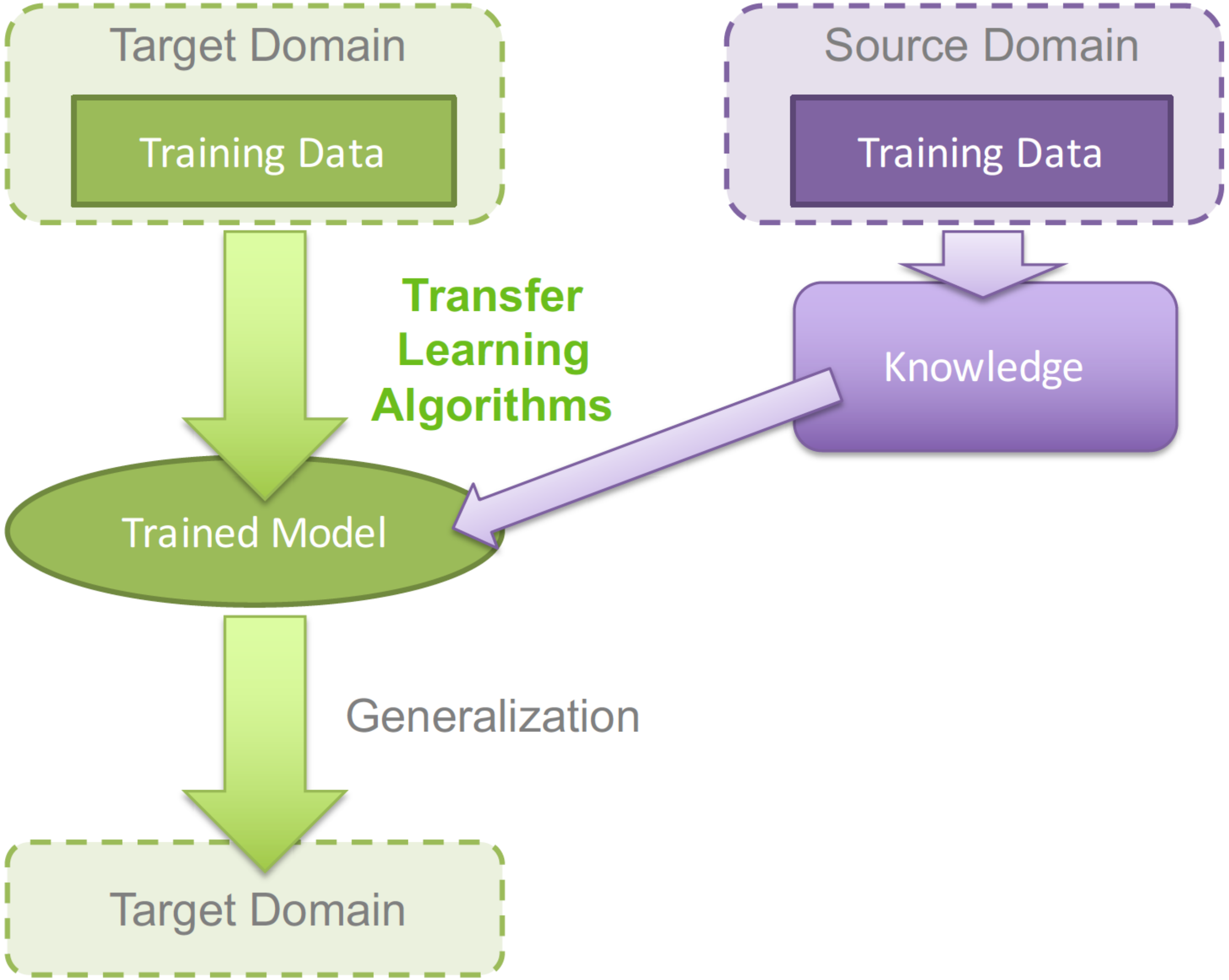
- Elements of machine learning on single task
  - The problem (task/domain)
  - Training data
  - Learning algorithms
  - Trained model
  - Applying model on unseen data (generalization)



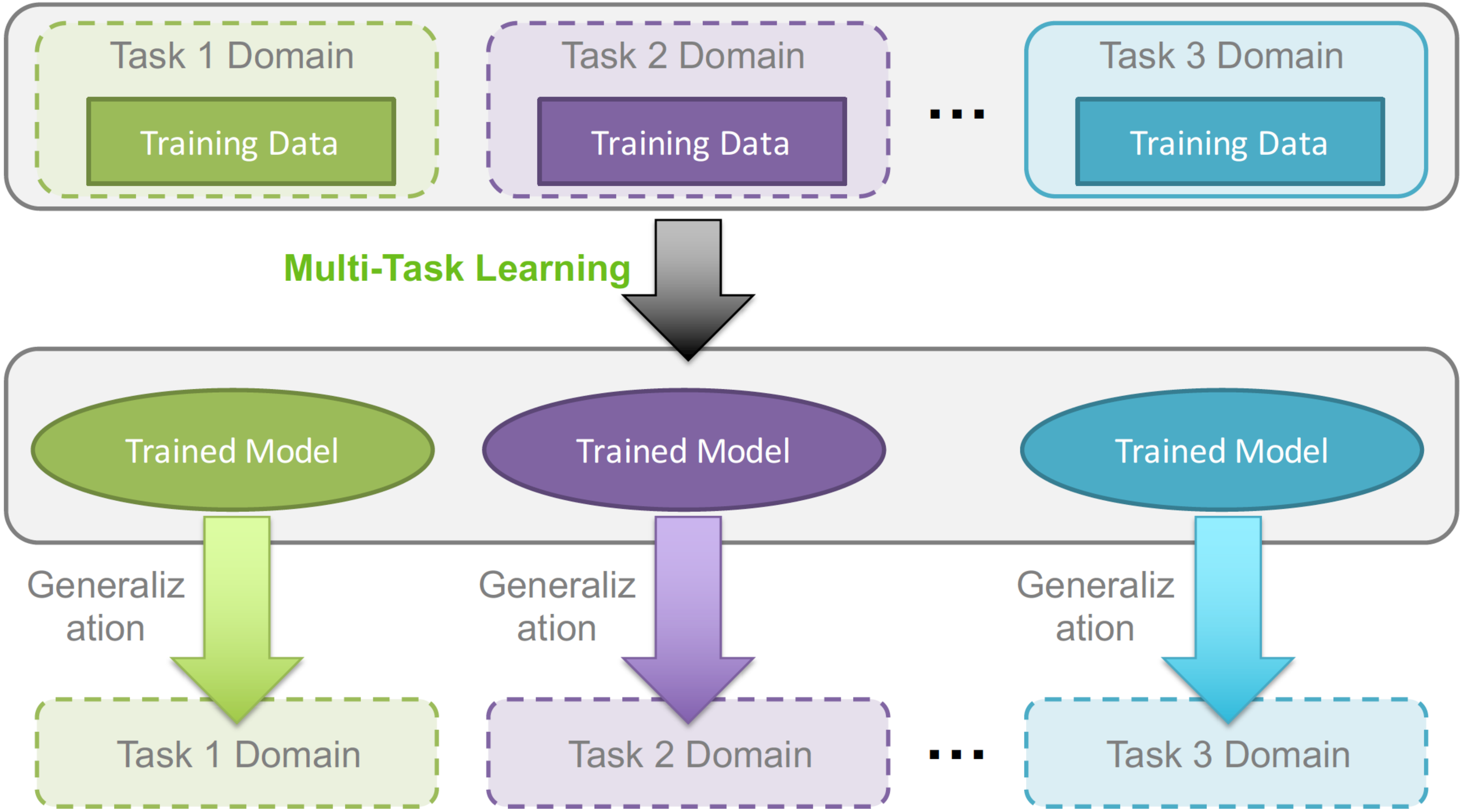
Transfer Learning

Improve Learning New Task
by Learned Task





Multi-Task Learning



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