Machine Learning Overview

Yale

CPSC/AMTH 663

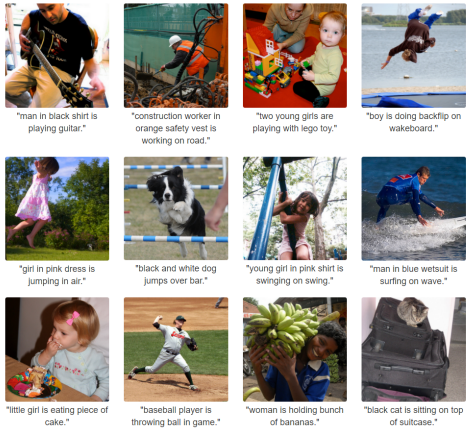
1. What is machine learning?
2. Supervised Learning
   - Regression
   - Complexity, overfitting, and underfitting
   - Regularization
   - Classification
   - Validation techniques
   - Anomaly detection
3. Unsupervised Learning
   - Principal components analysis (PCA)
   - Manifold learning
   - Clustering

xkcd.com/1838

- Machine learning algorithm = algorithm that can learn from data
- What do we mean by "learn from data"?
- "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E." – Mitchell (1997)
- What are E, T, and P?
  - It depends on the problem!

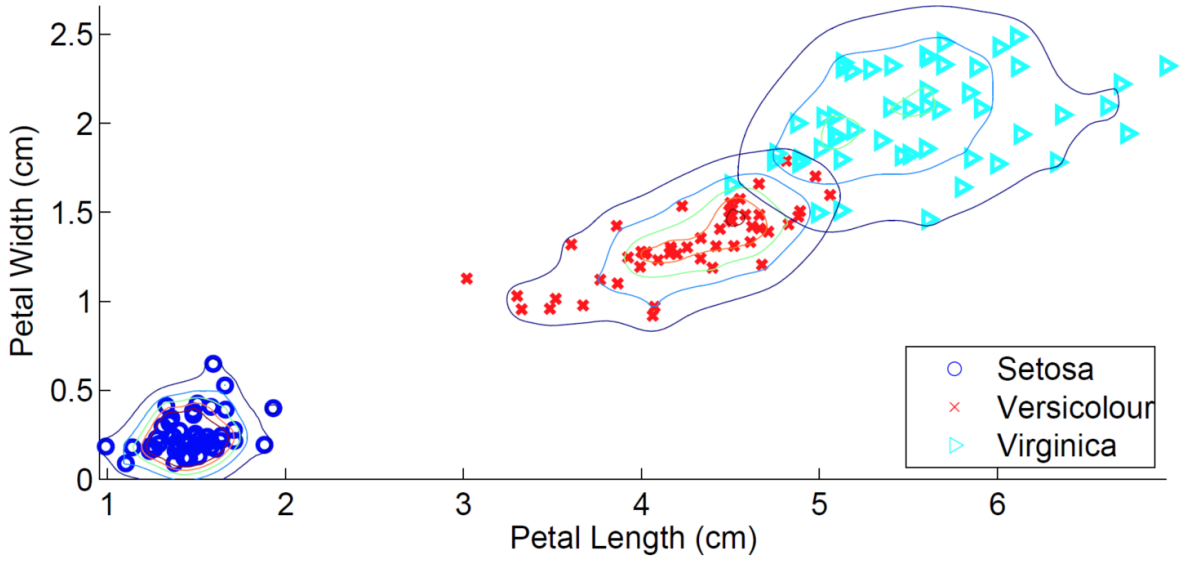
- Machine learning helps us to solve problems that are too difficult for fixed programs
- Learning is the process of attaining the ability to perform the task
- Tasks are typically described in terms of how to process an ***example***
  - An example is a collection of measured ***features***
  - Often represented as a vector $x \in \mathbb{R}^d$ of features
  - E.g. an example includes a single image with the features comprised of the pixels in the image

- Classification
  - Output: a function $f: \mathbb{R}^d \to \{1, \ldots, k\}$
- Regression
  - Output: a function $f: \mathbb{R}^d \to \mathbb{R}$
- Transcription
- Machine translation
- Anomaly detection
- Synthesis and sampling
- Imputation of missing values
- Denoising
- Density estimation

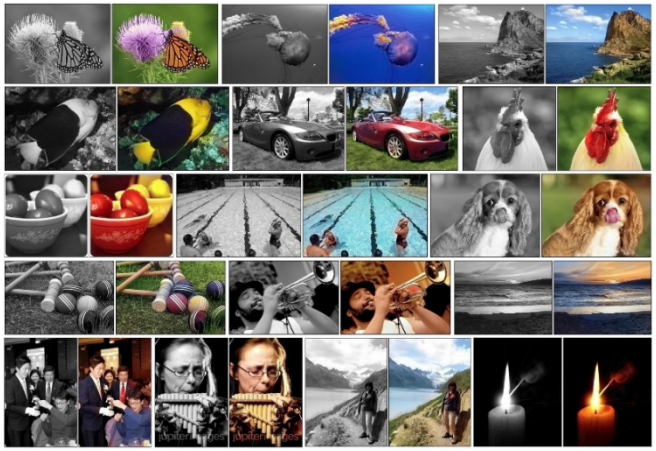


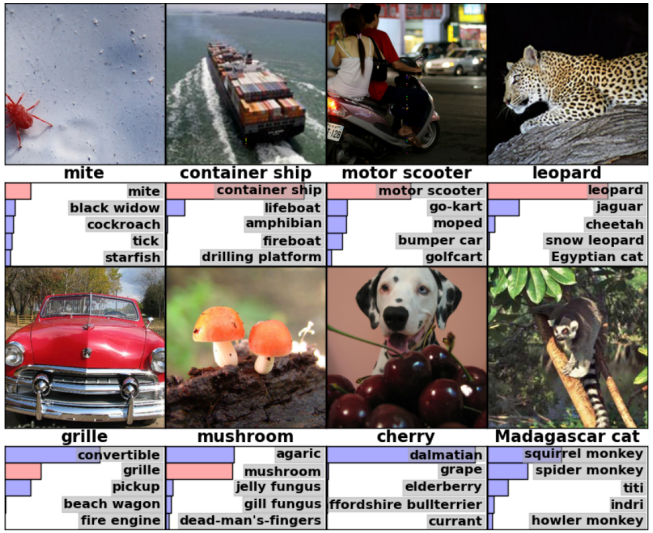
- Need a quantitative measure to evaluate a machine learning algorithm
- Provides a measure to guide the selection of an algorithm
- Examples:
  - **A*ccuracy*** (or equivalently, the ***error rate***) in a classification problem
  - Prediction error
- Performance is typically ***estimated***/evaluated based on a *test set* of data (different from training data)
- Choosing a performance measure isn't always straightforward
  - Transcription
  - Regression

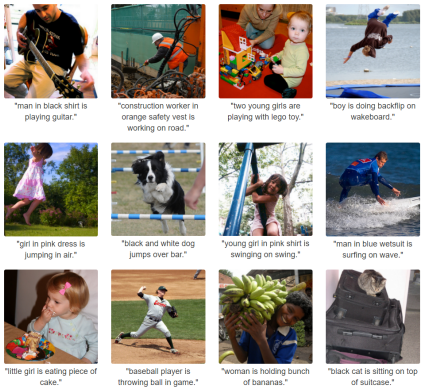
- Most algorithms experience an entire ***dataset*** consisting of many examples or ***data points***
- Example: Iris data set (Fisher, 1936)
  - Four features: sepal length, sepal width, petal length, petal width
  - 3 species/classes, 50 data points per class



- Goal is to build a program that learns from experience E to do well at task T as measured by performance measure P





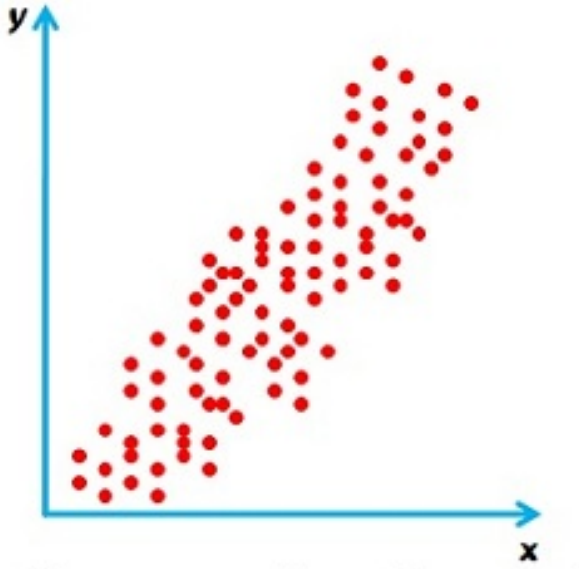


Supervised Learning

- Dataset contains features and a ***label*** or target
  - E.g. classify iris plants into the three different species
- Mathematically, supervised learning observes examples of a random vector $\boldsymbol{x}$ and an associated label $\boldsymbol{y}$
  - Goal is to learn to predict $\boldsymbol{y}$ from $\boldsymbol{x}$
- Term comes from view of the label $\boldsymbol{y}$ coming from an instructor or teacher

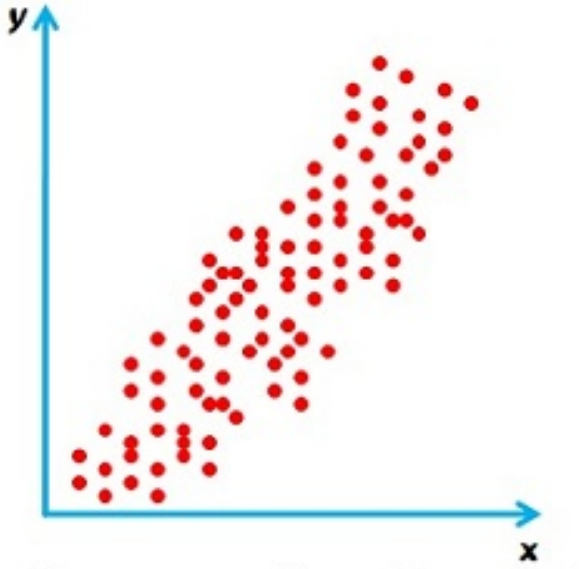
- Input data space $\mathcal{X}$
- Output (label) space $\mathcal{Y}$
- Unknown function $f: \mathcal{X} \rightarrow \mathcal{Y}$
- Dataset $\mathcal{D} = \{(\boldsymbol{x}_1, y_1), \ldots (\boldsymbol{x}_n, y_n)\}$ with $\boldsymbol{x}_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$
- Finite $\mathcal{Y} \Rightarrow$ classification
- Continuous $\mathcal{Y} \Rightarrow$ regression

- We have a set of n observations $(\boldsymbol{x}_i, y_i)$ with $y_i \in \mathbb{R}$
- Goal is to learn a function that predicts label y from sample $\boldsymbol{x}$



Surbhi S, 2016

1. Choose a ***model class*** of functions
2. Choose a performance measure to guide the selection of a function from the model class



Surbhi S, 2016

- We'll begin with a simple model class: linear functions

- We want to fit a linear function to dataset $\mathcal{D} = \{(\boldsymbol{x}_1, y_1), \ldots (\boldsymbol{x}_n, y_n)\}$
- $\hat{y} = \boldsymbol{w} \cdot \boldsymbol{x} = \boldsymbol{w}^T \boldsymbol{x}$
  - $\boldsymbol{w} \in \mathbb{R}^d$ is a vector of parameters
  - Line if $d = 1$
  - Plane if $d = 2$
  - Hyperplane in general d
- How do we determine $\boldsymbol{w}$?
- Based on the performance measure P

- Labels are in $\mathcal{Y}$
  - Discrete (classification)
  - Continuous (regression)
- ***Loss function*** $L: \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}$
- L maps decisions/predictions to a cost.
  - $L(\hat{y}, y)$ is the penalty for predicting $\hat{y}$ when the true label is y
- Standard choice for regression is ***mean squared error*** (MSE)
  - $L(\hat{y}, y) = (\hat{y} - y)^2$
- Absolute loss: $L(\hat{y}, y) = |\hat{y} - y|$

- Parametric function $f(\boldsymbol{x}; \boldsymbol{w})$
  - E.g. $f(\boldsymbol{x}; \boldsymbol{w}) = \boldsymbol{w}^T \boldsymbol{x}$ for regression
- The ***empirical loss*** on a dataset $\mathcal{D}$ is

$$L(\boldsymbol{w}; \mathcal{D}) = \frac{1}{n} \sum_i L(f(\boldsymbol{x}_i; \boldsymbol{w}), y_i)$$

- Least squares minimizes the empirical MSE
- We care about predicting labels for *new* samples
- Does empirical loss minimization help with that?

- Assumption: sample and label pairs $(\boldsymbol{x}, y)$ are drawn from the unknown joint distribution $\mathrm{p}(\boldsymbol{x}, y)$
- Data are sampled i.i.d.
- Empirical loss measured on training data

$$L(\boldsymbol{w}; \mathcal{D}) = \frac{1}{n} \sum_i L(f(\boldsymbol{x}_i; \boldsymbol{w}), y_i)$$

- Goal: minimize the ***expected loss*** (aka ***risk***)

$$R(\boldsymbol{w}) = \mathbb{E}[L(f(\boldsymbol{x}; \boldsymbol{w}), y)]$$

- Empirical loss

$$L(\boldsymbol{w}; \mathcal{D}) = \frac{1}{n} \sum_i L(f(\boldsymbol{x}_i; \boldsymbol{w}), y_i)$$

- Risk

$$R(\boldsymbol{w}) = \mathbb{E}[L(f(\boldsymbol{x}; \boldsymbol{w}), y)]$$

- If training set is representative of $p(\boldsymbol{x}, y)$, then empirical loss is a suitable proxy for the risk
- Essentially, we estimate $p(\boldsymbol{x}, y)$ and $R(\boldsymbol{w})$ from the empirical distribution of the data

1. Choose a model class $\mathcal{H}$ of functions $f: \mathcal{X} \rightarrow \mathcal{Y}$
   - Regression example: linear functions parameterized by $\boldsymbol{w}$: $f(\boldsymbol{x}; \boldsymbol{w}) = \boldsymbol{w}^T \boldsymbol{x}$
2. Select the function in $\mathcal{H}$ that minimizes the empirical risk
   - Linear regression example: minimize empirical squared loss:

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} \sum_{i=1}^{n} (\boldsymbol{w}^T \boldsymbol{x}_i - y_i)^2$$

   - How do we find $\boldsymbol{w}^*$?
   - Use calculus

- Add an offset w_0: $f(\boldsymbol{x}; \boldsymbol{w}) = \boldsymbol{w}^T \boldsymbol{x} + \mathrm{w}_0$

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} \sum_{i=1}^{n} (\boldsymbol{w}^T \boldsymbol{x}_i + w_0 - y_i)^2$$

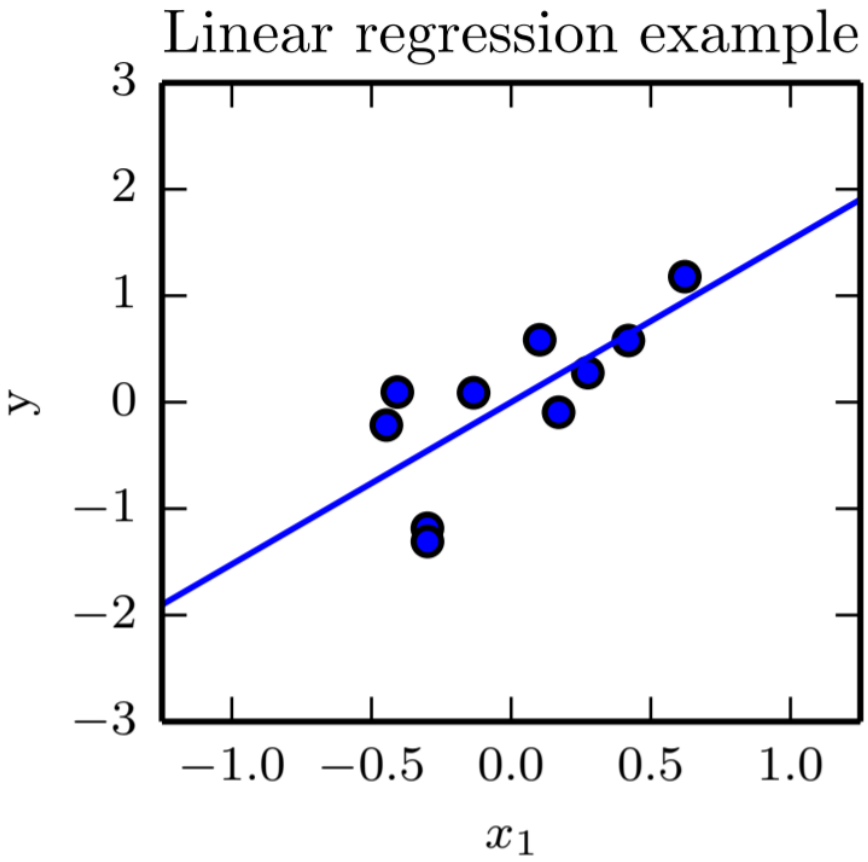
$$= \arg\min_{\boldsymbol{w}} L(\boldsymbol{w}; \mathcal{D})$$

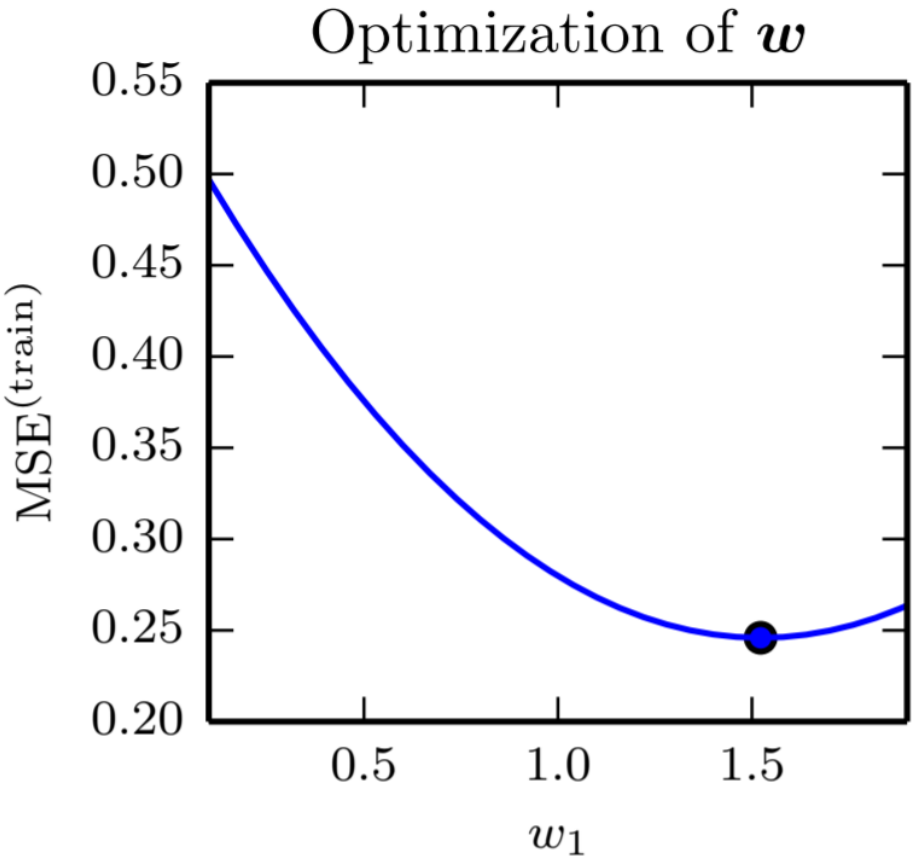
- Set $\frac{\partial L(\boldsymbol{w};\mathcal{D})}{\partial w_i} = 0$ for each i
- Results in $d + 1$ equations and $d + 1$ unknowns

- Switching to vector/matrix notation
  - $X =$ data matrix with an additional column of 1's (for the offset)
  - $\boldsymbol{y} =$ vector of labels
  - Prediction: $\hat{\boldsymbol{y}} = X\boldsymbol{w}$
- Resulting solution is

$$\boldsymbol{w}^* = (X^T X)^{-1} X^T \boldsymbol{y}$$

- Closed form solution!
- $(X^T X)^{-1} X^T$ is the Moore-Penrose pseudoinverse of X

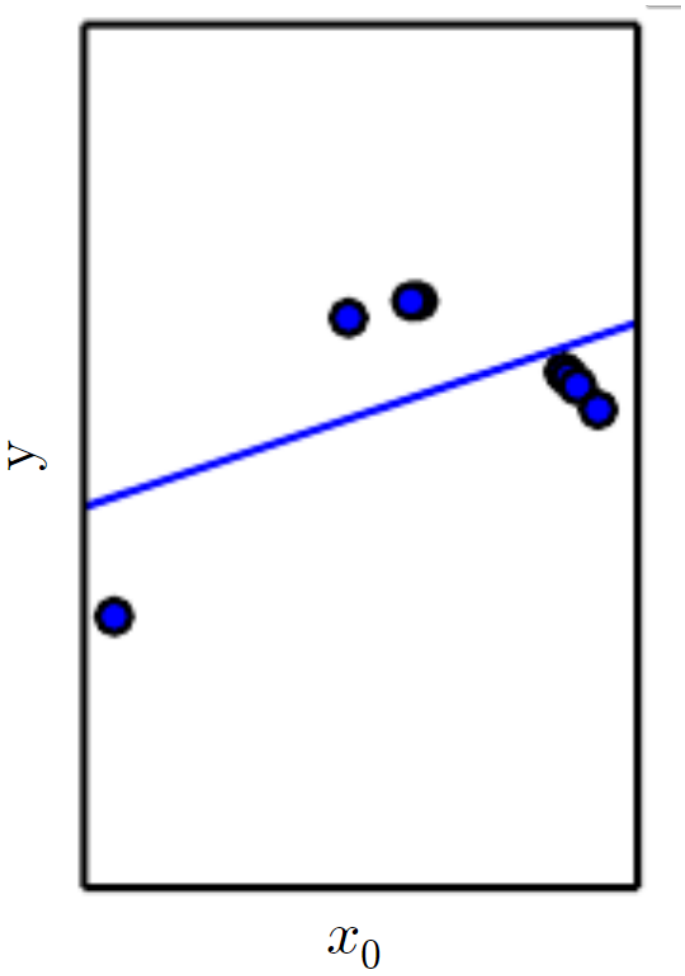




Goodfellow et al., 2016

- Challenge of machine learning: algorithm must perform well on previously unseen inputs
  - Not just training inputs
- In other words, we want the algorithm to ***generalize*** well to unseen data
  - That is, we want the expected error rate on previously unseen inputs to be low
  - This error rate is called the ***generalization error*** or the ***test error***
- We also need to estimate the generalization error
  - Done typically by dividing the total amount of data into two sets: ***training data*** and ***test data***
    - More on how this is done specifically later
  - Model is trained on the training data and then tested on the test data
- Expected test error $\geq$ expected training error

- Many processes are not well modeled by linear functions
  - Biological systems
  - Stock market prices
  - Periodic signals
- In these cases, linear functions can ***underfit*** the data
  - Training error is very high
  - Test error will also be high

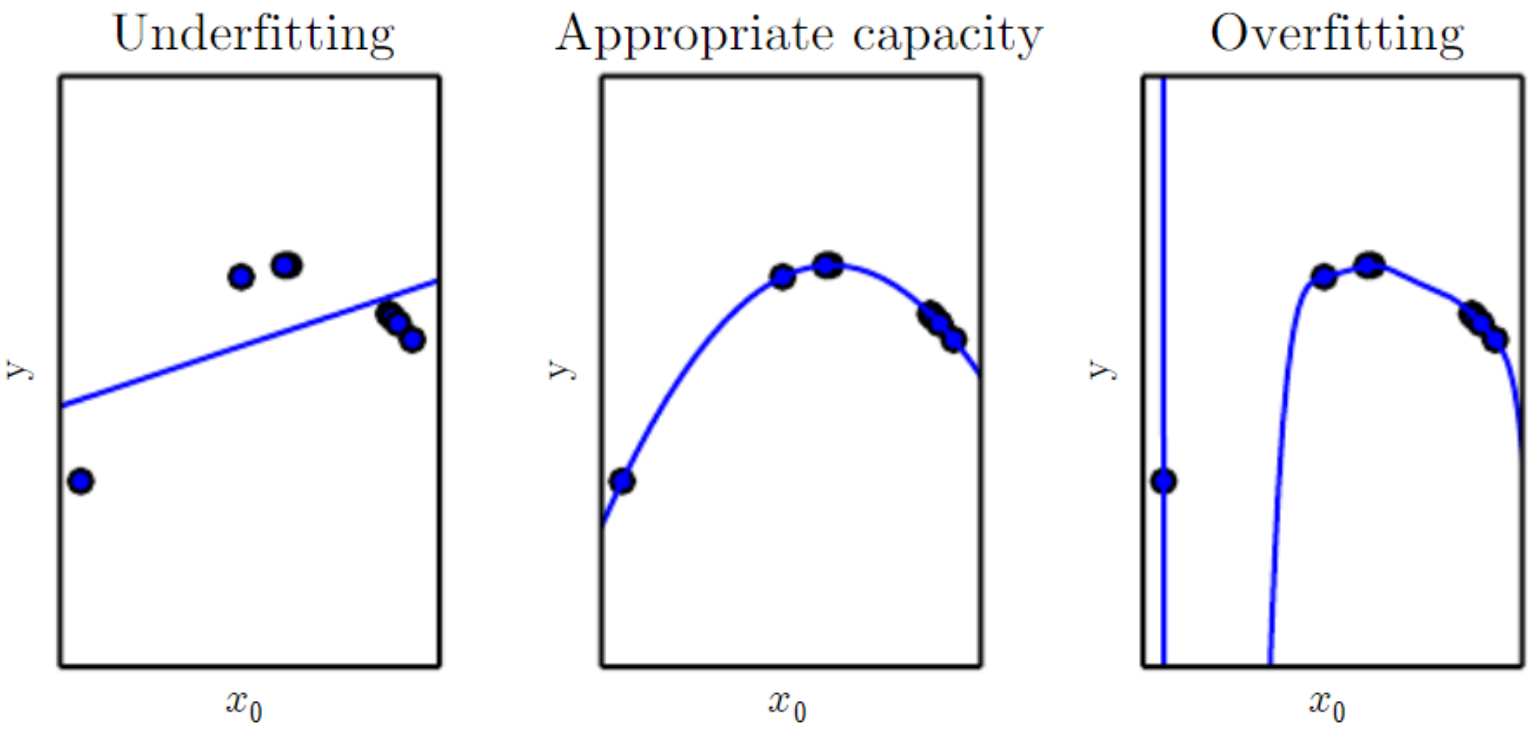


Goodfellow et al., 2016

- Solution: increase ***complexity*** of the model
  - Model complexity is the # of independent parameters in the function model (i.e. “degrees of freedom”)
- We can increase the complexity of our linear function by adding polynomial terms
  - Example: $d = 1$

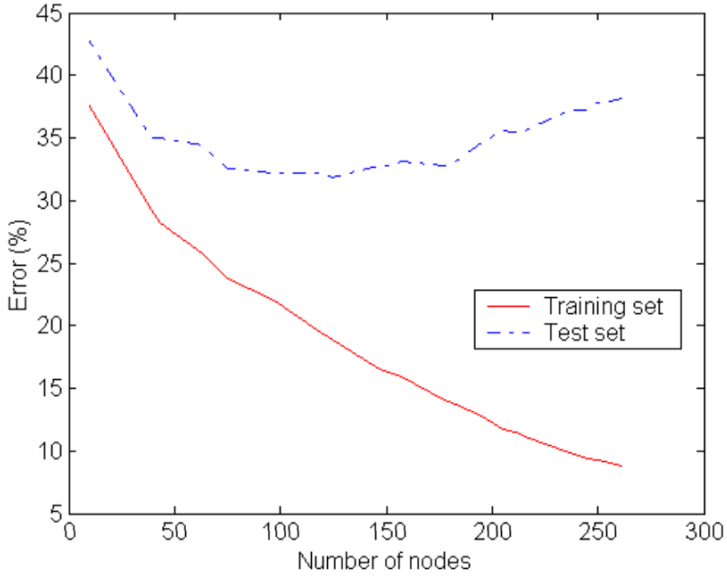
$$f(x; \boldsymbol{w}) = \sum_{j=0}^{m} w_j x^j$$

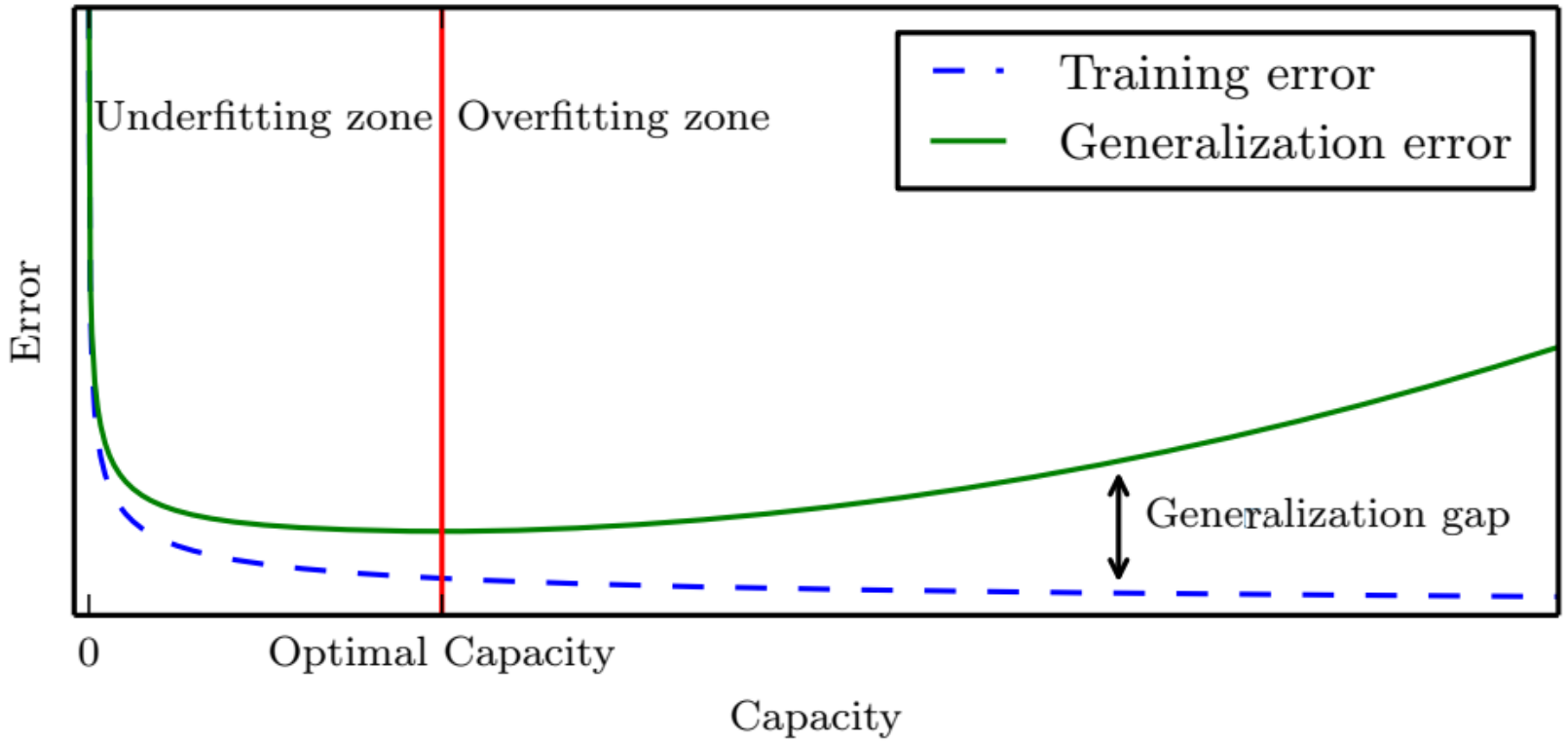
  - Still linear in $\boldsymbol{w}$ ⇒ we can use linear regression!
- Can generalize further by using other nonlinear functions of x (e.g. log or exp)



Goodfellow et al., 2016

- Overfitting occurs when the gap between training error and test error is too large
  - The model is capturing unique characteristics of the training set
  - The model does not generalize well to new samples
  - Occurs when the complexity is too large
- Avoiding overfitting
  - Choose model complexity based on test error
  - Generally requires the use of another test set for evaluation





Goodfellow et al., 2016

- Include a penalty on model complexity directly in the cost function:

$$\sum_i L(f(\boldsymbol{x}_i; \boldsymbol{w}), y_i) + \#\text{parameters}$$

- The addition of the penalty is known as ***regularization***
- Regularization: "any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error." (Goodfellow et al., 2016)
  - Lots of different penalties can be included in regularization
  - Regularization is useful/important in deep learning (and machine learning in general)

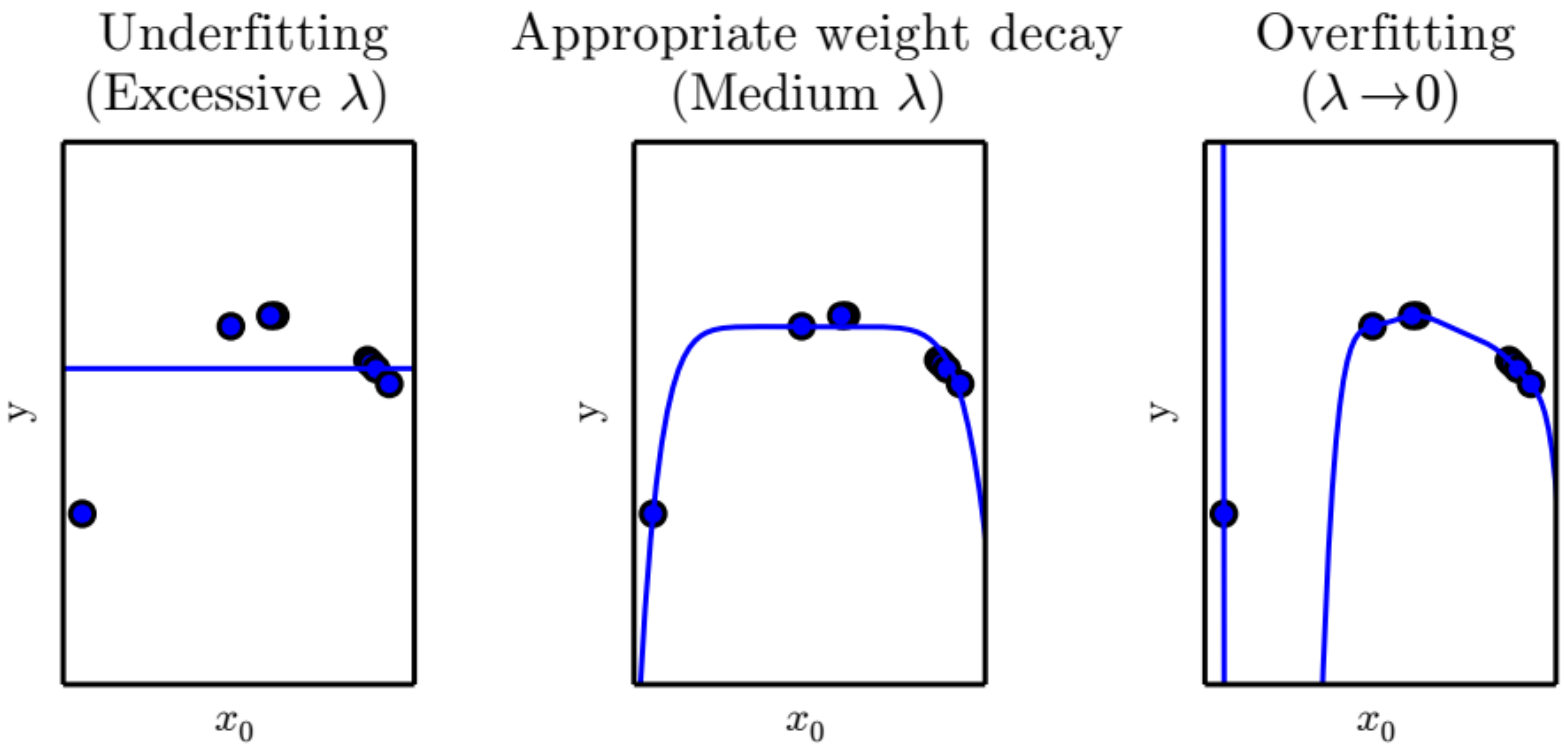
- Ridge regression: penalize with L2 norm

$$\boldsymbol{w}^* = \arg\min \sum_i L(f(\boldsymbol{x}_i; \boldsymbol{w}), y_i) + \lambda \sum_{j=1}^{m} w_j^2$$

  - Closed form solution exists $\boldsymbol{w}^* = (\lambda I + X^T X)^{-1} X^T \boldsymbol{y}$
- LASSO regression: penalize with L1 norm

$$\boldsymbol{w}^* = \arg\min \sum_i L(f(\boldsymbol{x}_i; \boldsymbol{w}), y_i) + \lambda \sum_{j=1}^{m} |w_j|$$

  - No closed form solution but still convex (optimal solution can be found)



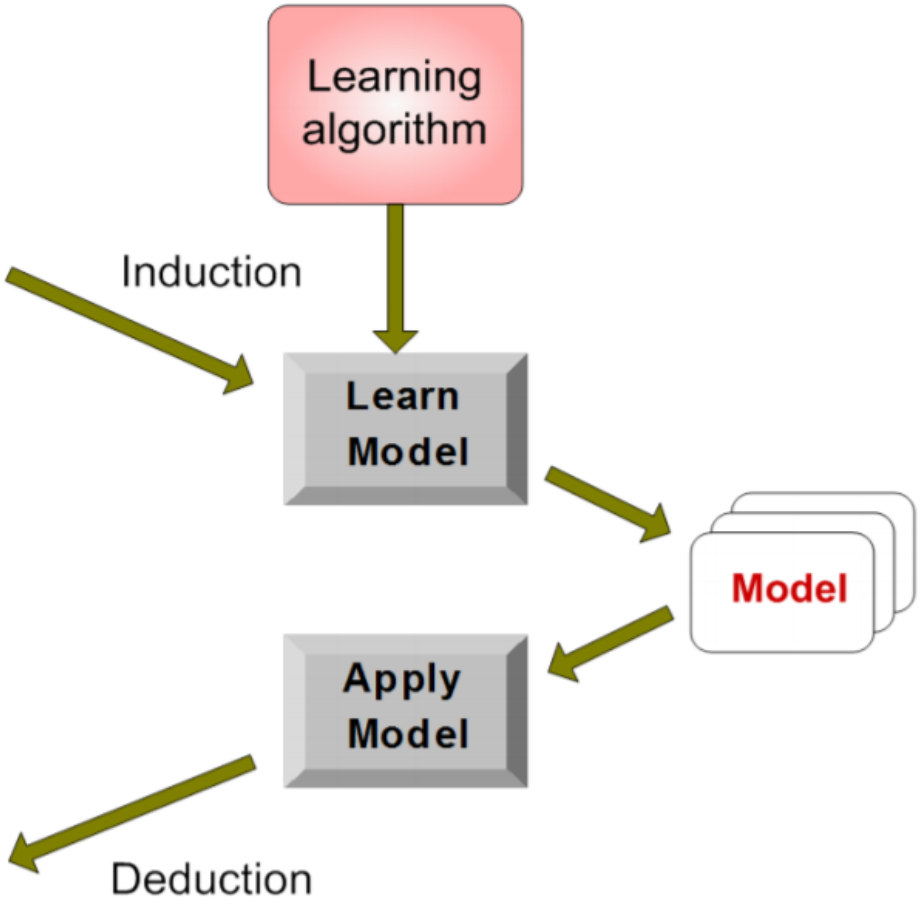
Goodfellow et al., 2016

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

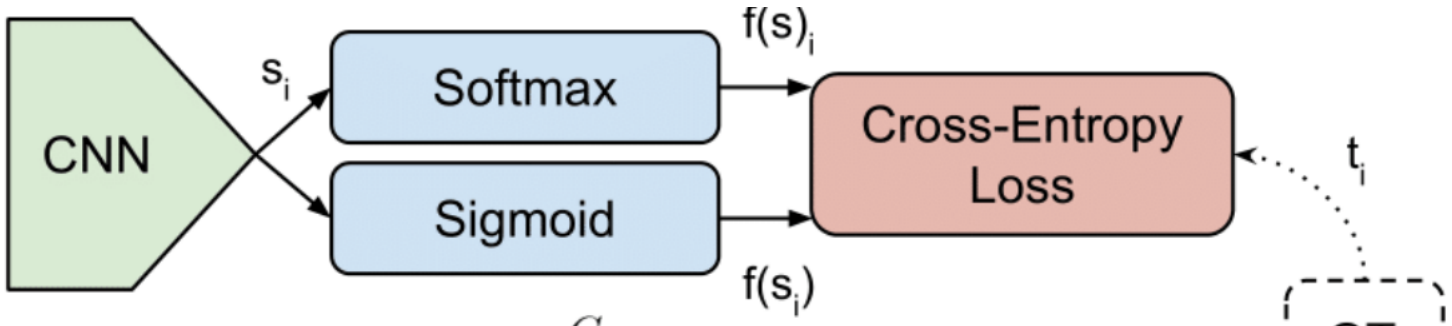
Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set



- Let $f: \mathcal{X} \rightarrow \mathcal{Y}$ be a classifier
- Common loss function is the 0/1 loss:

$$L(f(\boldsymbol{x}), y) = \begin{cases} 0, & f(x) = y \\ 1, & f(x) \neq y \end{cases}$$



$$CE = -\sum_{i}^{C} t_i log(f(s)_i)$$

$$CE = -\sum_{i=1}^{C'=2} t_i log(f(s_i)) = -t_1 log(f(s_1)) + (1 - t_1) log(1 - f(s_1))$$

- For binary case, we get that

$$f(x) = 1 \text{ iff } \log \frac{p(y=1|\boldsymbol{x})}{p(y=0|\boldsymbol{x})} \geq 0$$

- We can model the decision boundary with a linear function:

$$\log \frac{p(y=1|\boldsymbol{x})}{p(y=0|\boldsymbol{x})} = 0 = w_0 + \boldsymbol{w}^T \boldsymbol{x}$$

- It can be shown that

$$p(y=1|\boldsymbol{x}) = \frac{1}{1+\exp(-w_0 - \boldsymbol{w}^T \boldsymbol{x})}$$

- Known as the ***logistic function***

- Logistic function

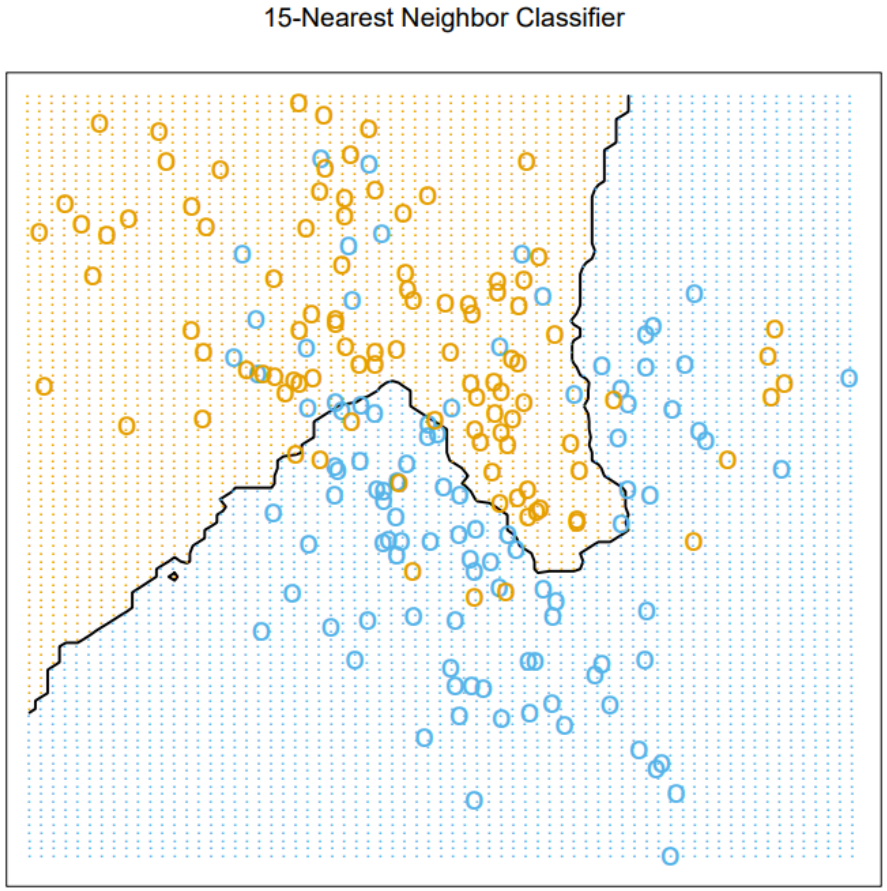
$$p(y = 1|\boldsymbol{x}) = \frac{1}{1 + \exp(-w_0 - \boldsymbol{w}^T \boldsymbol{x})}$$

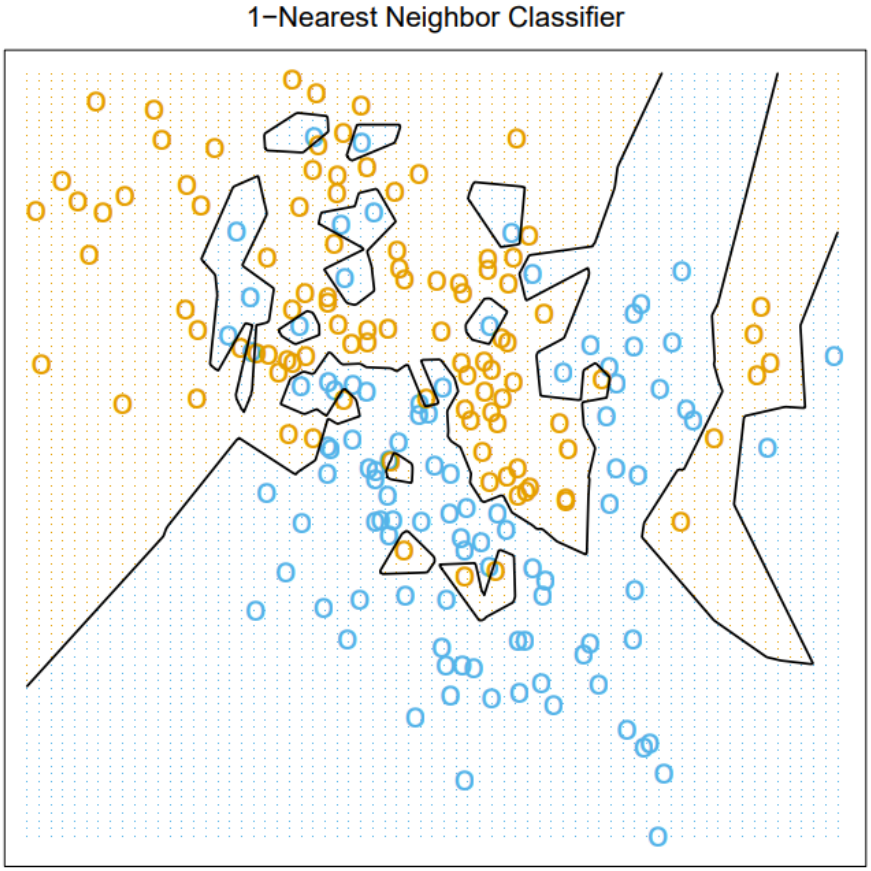
- Optimal parameters can be found using ***Maximum Likelihood***
- Constructs a linear decision boundary with linear model
  - Can generalize to nonlinear boundaries
- This classifier is known as ***Logistic Regression***

Hastie et al., 2009

- No training required

1. Find the k-nearest neighbors of the test point x in the training data
   - Call this set $\mathcal{N}_k(x)$
2. Choose $\hat{y}$ according to a majority vote of the corresponding labels of the neighbors in $\mathcal{N}_k(x)$





Hastie et al., 2009

- Decision trees and random forests
- Bayesian classification (Naïve Bayes and Bayesian Belief Networks)
- Deep learning!
- Support Vector Machines
  - Kernel methods

- Recall that we consider both training error and test error
- How do we select the data for estimating the two?
- Common approaches:
  - **Holdout validation**: Randomly split the data into two disjoint sets. Train the model on one set and compute test error on the other.
  - **Random subsampling**: Repeat the holdout validation k times independently and compute the mean test error
  - Typically these approaches are not very robust

- Cross validation averages evaluation scores over several different splits to provide a more realistic assessment
- Types of cross validation
  - **Leave-one-out**: Use all data points except one for training and validate on the remaining point.
  - **2-fold cross validation**: Randomly split the data into two halves and run two validation iterations, each using one half for training and the other for validation
  - **k-fold cross validation**: Randomly partition the data into k equal size parts and run k validation iterations, each using one part for validation and the rest for training
- In all cases, the validation scores are averaged
- Each data point is only used once in testing but many times in training

- Goal: detect significant deviations from normal behavior
- Examples
  - Credit card fraud detection
  - Cybersecurity intrusion detection
  - Detecting bot traffic in online advertising
  - Malfunction detection in process monitoring
- Approaches
  - 1-class classification
  - Density estimation
  - Entropy estimation
  - Others…

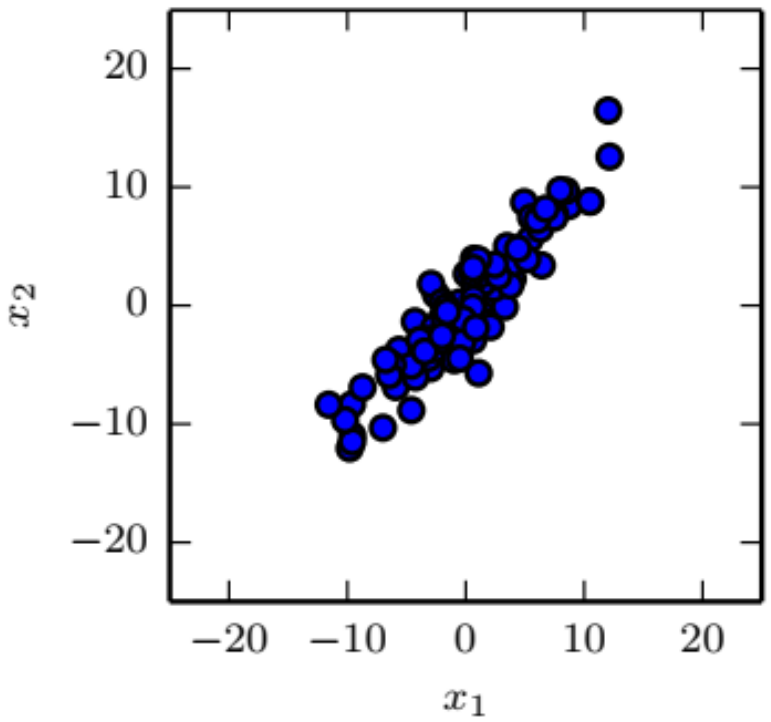
Unsupervised learning

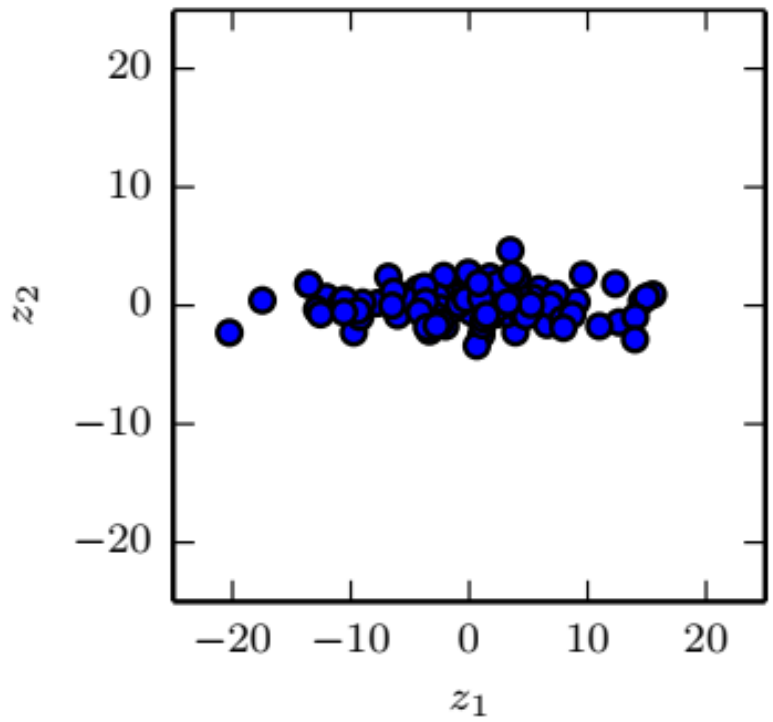
- Dataset contains only features and no labels
- Goal is to find hidden patterns in the unlabeled data
- Examples
  - Density estimation
  - Data generation
  - Clustering
  - Data denoising
  - Representation learning
- Aside: ***Semi-supervised learning*** attempts to combine information from both labeled and unlabeled data to deduce information

- Find the “best” representation of the data
  - I.e., find a representation of the data that preserves as much information as possible while obeying some penalty or constraint that simplifies the representation
- How do we define simple?
  - Low-dimensional
  - Sparse
  - Independent components
- Above criteria aren’t necessarily mutually exclusive
  - E.g., low-dimensional representations often have independent or weakly dependent components

- An unsupervised representation learning algorithm
  - Learns a lower dimensional representation
  - Elements are linearly uncorrelated with each other
  - Linear method
- PCA finds a low-dimensional linear transformation of the data that best reconstructs the original data (in the mean squared error sense)
  - X is the data matrix
  - $\hat{X}_k$ is the reconstructed matrix from the k-dimensional representation
  - PCA minimizes $\left\|X - \hat{X}_k\right\|_2^2$

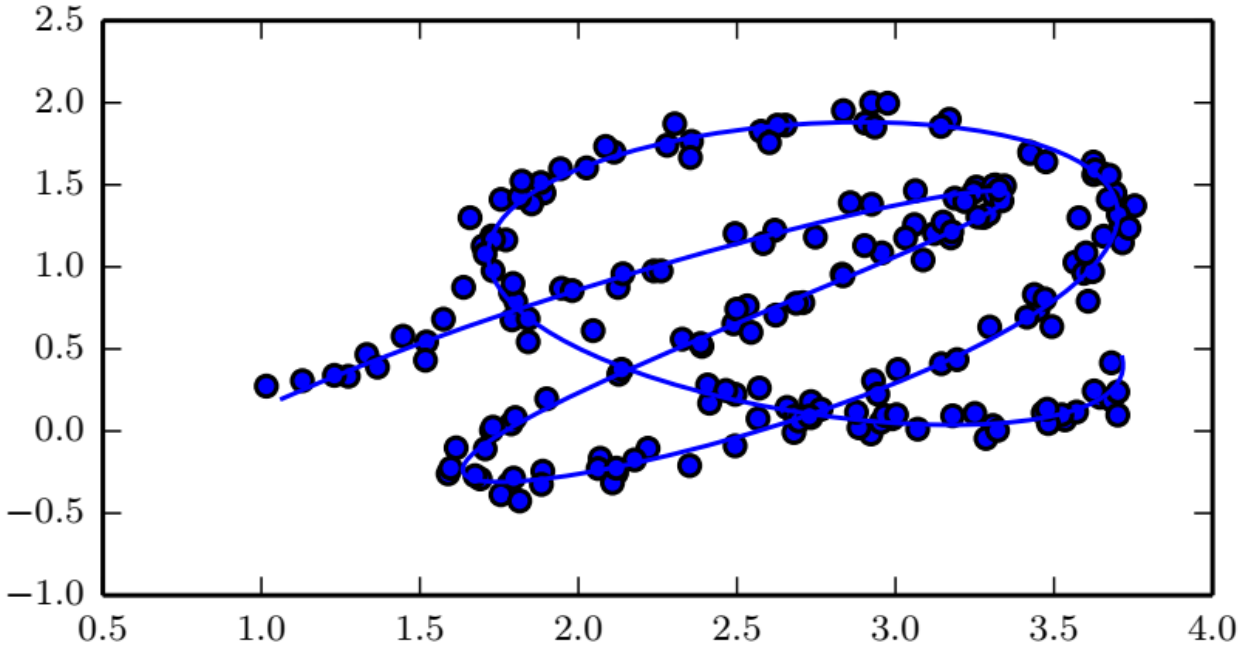
- PCA minimizes $\left\|X - \hat{X}_k\right\|_2^2$
- Corresponds to selecting the first k eigenvectors and eigenvalues of $X^T X$
  - $X^T X = W \Lambda W^T$
  - $W_k =$ the first k principal components, or the eigenvectors corresponding to the k largest eigenvalues
  - W_k gives the low dimensional representation
  - $\hat{X}_k = X W_k W_k^T$ gives the reconstruction



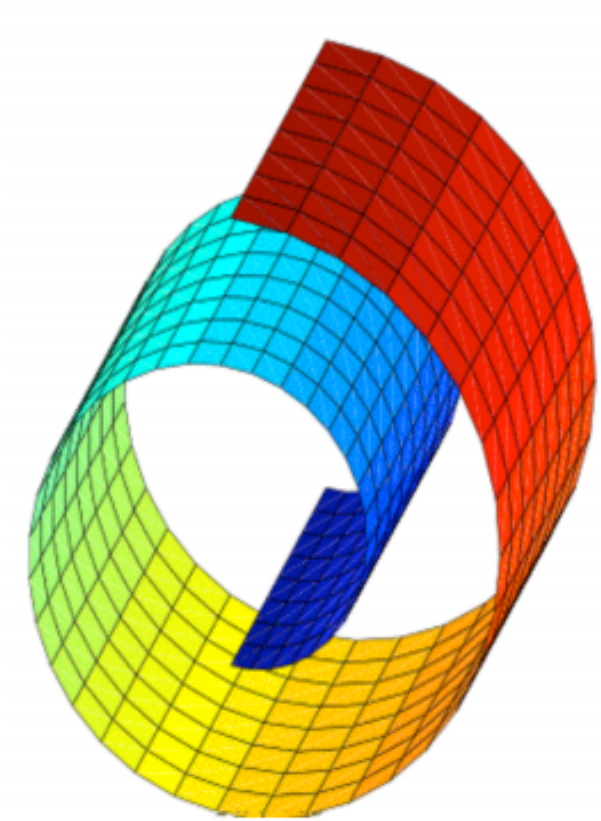


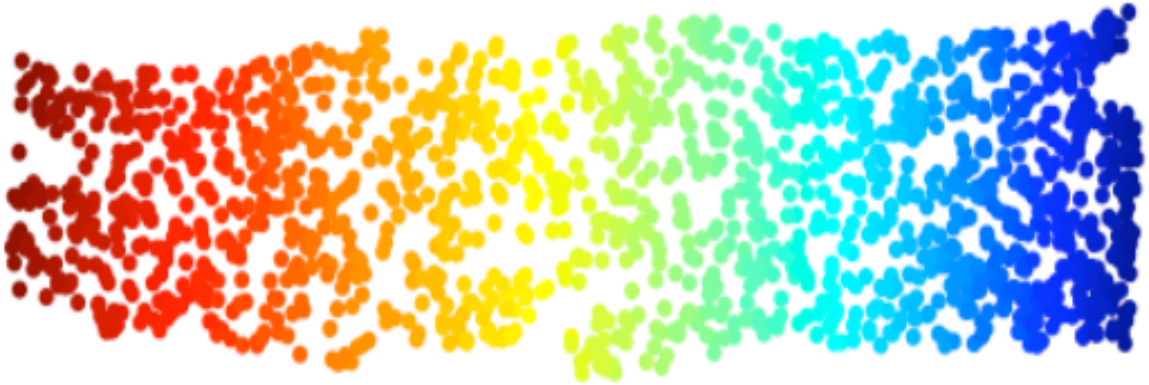
Goodfellow et al., 2016

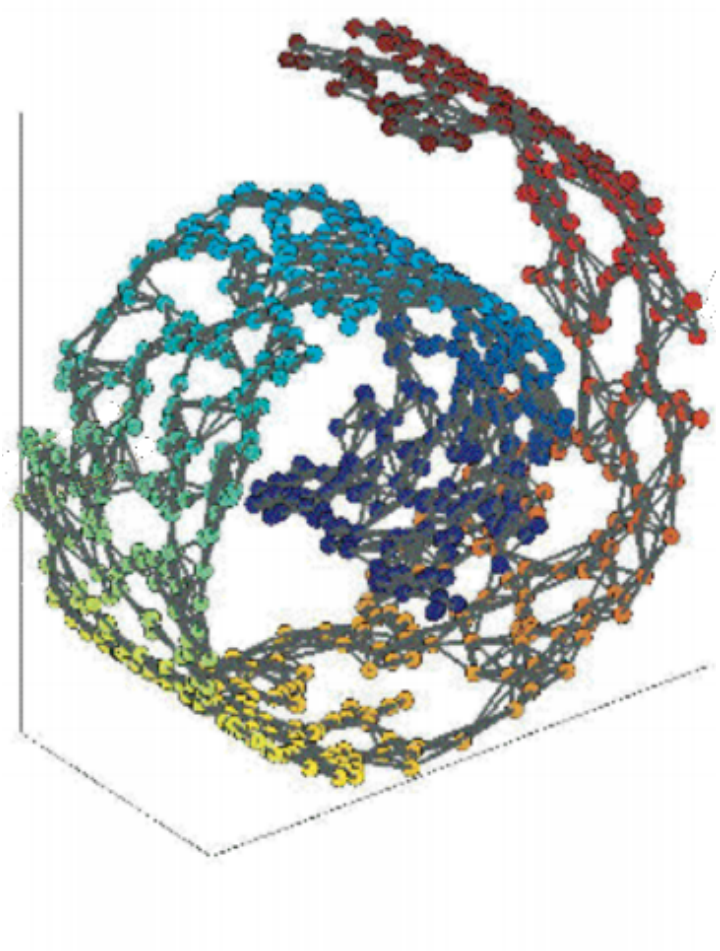
- Informally, a manifold in machine learning loosely means a connected set of points in high dimensions that can be approximated well by a small number of dimensions
- Generally a good assumption in many machine learning tasks (see Goodfellow book chapter 5.11.3)

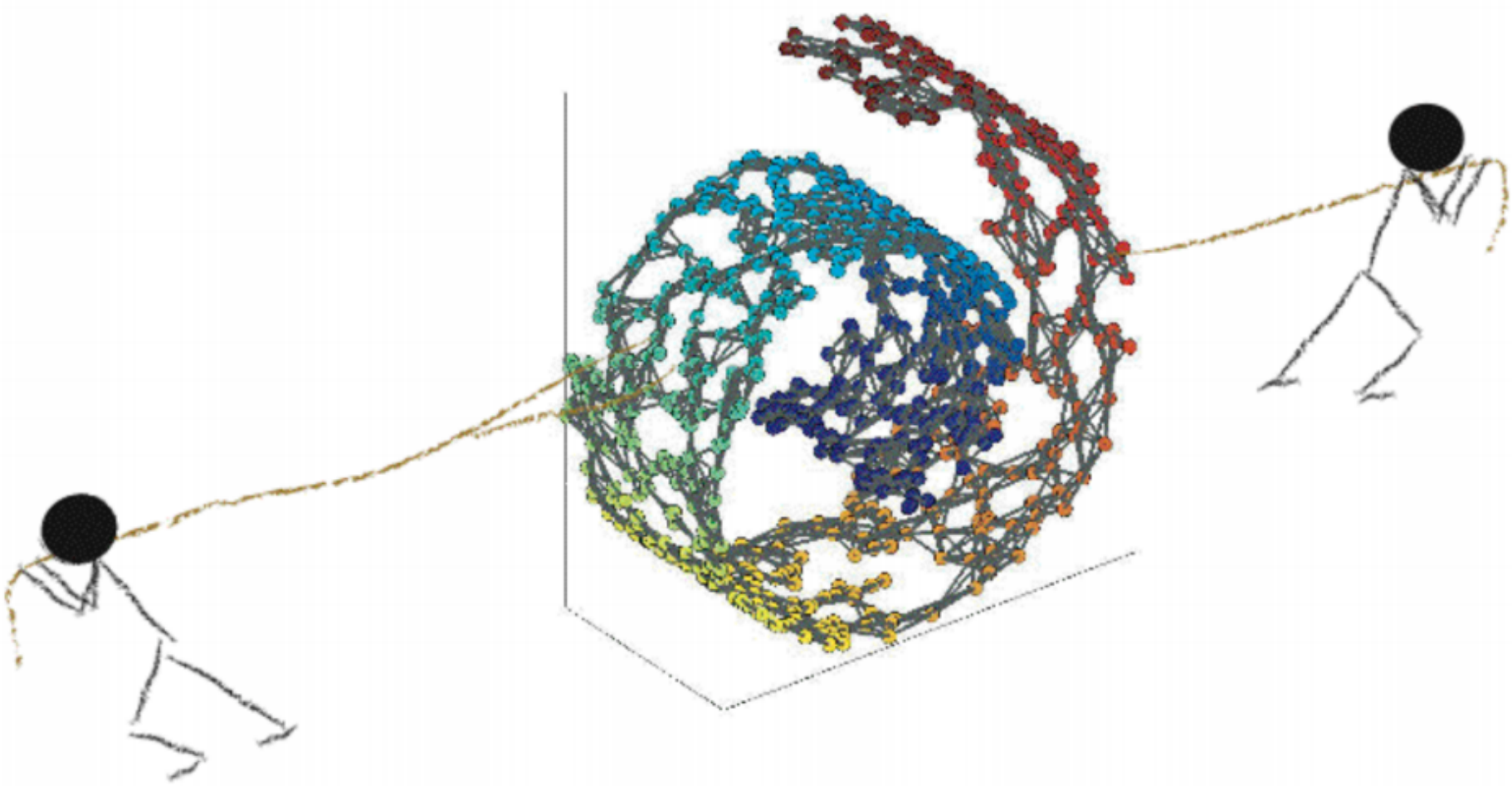


Goodfellow et al., 2016



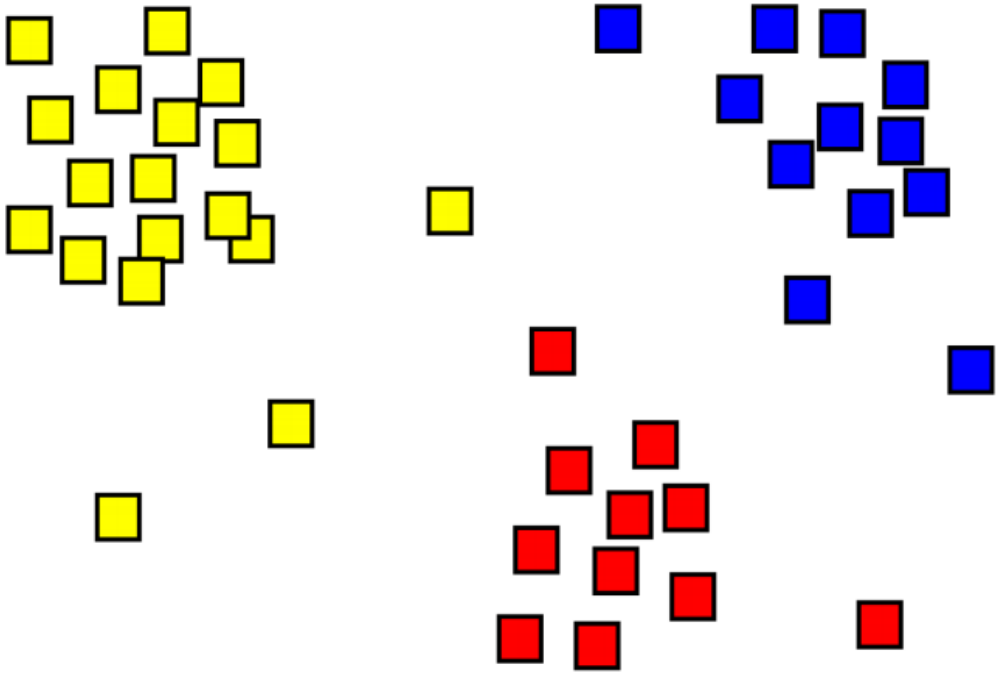






- Isomap
- Diffusion Maps (developed at Yale)
- Locally linear embedding
- Others

- Group objects so that objects within a cluster are similar to each other and dissimilar to objects in another cluster
  - Ill-posed problem ⇒ MANY clustering algorithms



- Suppose we have cars and trucks that are either red or gray
- How do you group these?
  - By color?
  - By vehicle type?
  - By both?
- Different clustering algorithms may give different results
  - Be careful which algorithm you choose
- Clustering quality depends on interpretability

- Examples
  - Clustering stocks to diversify investment
  - Community detection in social networks
  - Clustering genes and cells to uncover activities, reactions, and interactions
  - Network activity profiling by clustering packets/session
- Approaches
  - k-means
  - Hierarchical clustering
  - Spectral clustering
  - Variations on these using different metrics

- Chapter 5 in Goodfellow book
- *Elements of Statistical Learning* by Hastie, Tibshirani, & Friedman
- *Pattern Recognition and Machine Learning* by Bishop
- Online lecture notes from various sources, etc.