Lec11 Machine Learning
11-1 Basics of Machine Learning

Yang Shu

School of Data Science and Engineering

yshu@dase.ecnu.edu.cn

[Acknowledgement: Slides are adapted from Machine Learning Course, Mingsheng Long, THU]

Machine Learning

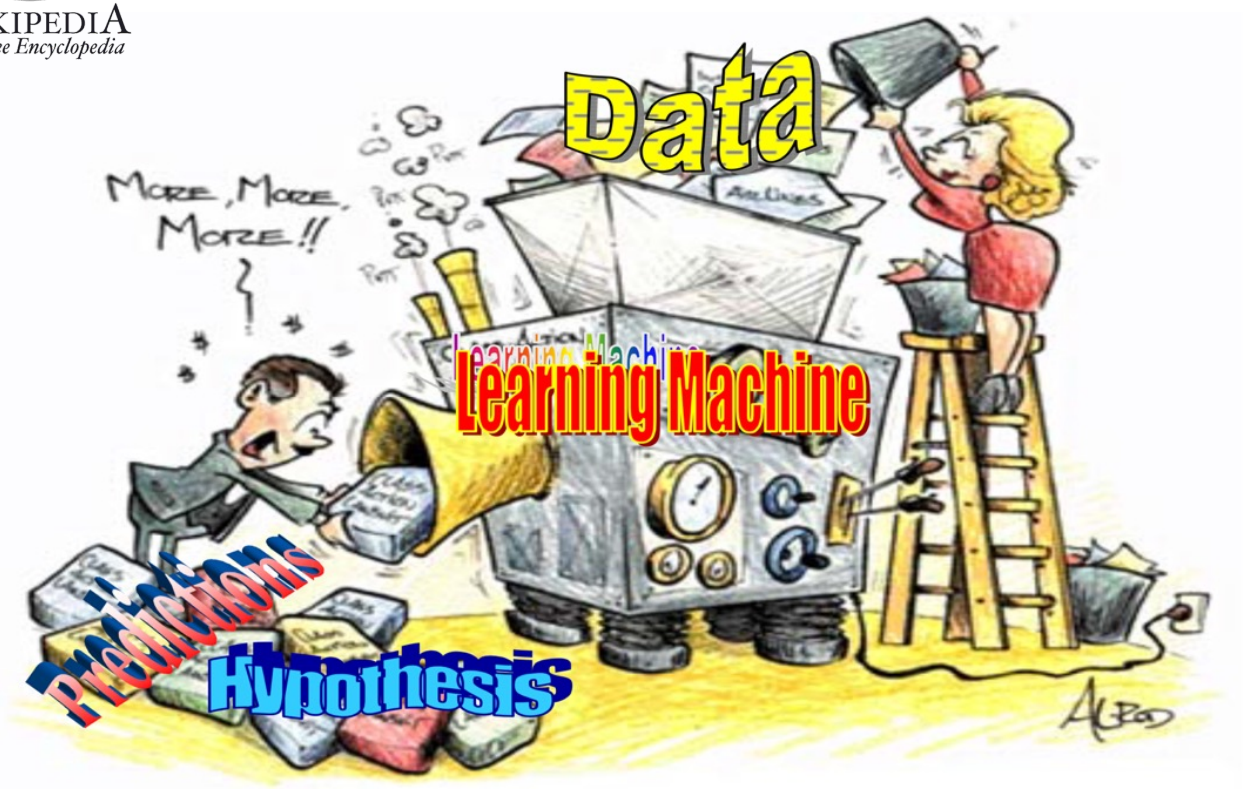
- 统计机器学习 (This lecture)
- 深度学习
- 自然语言处理
- 计算机视觉
- 强化学习与决策

Lec11-1 Basics of Machine Learning

- **Machine Learning**
  - The Framework
  - No Free Lunch Theorem
  - Model Selection
- K-Nearest Neighbors (KNN)
  - Curse of Dimensionality

What is Machine Learning?

Machine learning is a field of study that gives computers the ability to learn from data without being explicitly programmed.



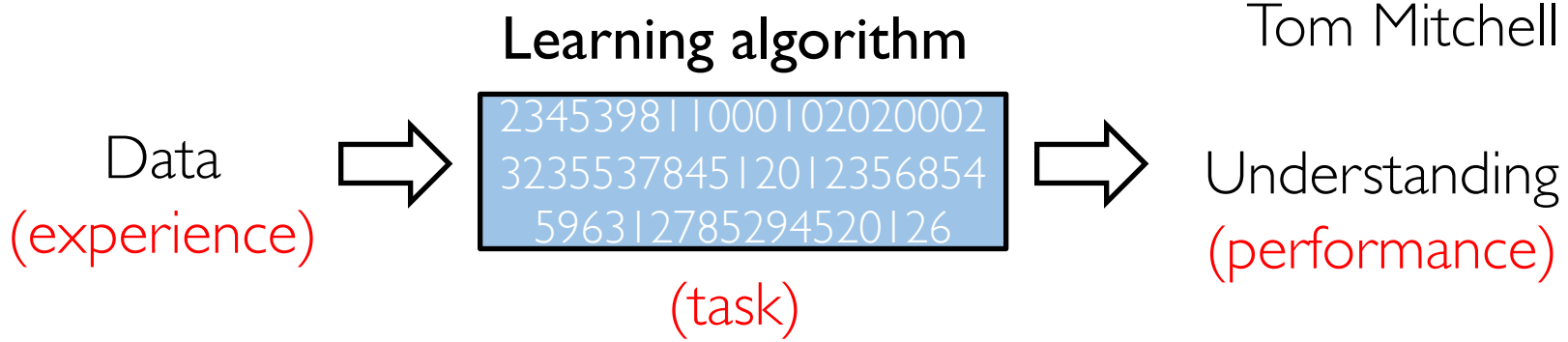
Arthur Samuel

Samuel, Arthur. Some Studies in Machine Learning Using the Game of Checkers. IBM J. R&D 3(3), 1959.

Machine Learning

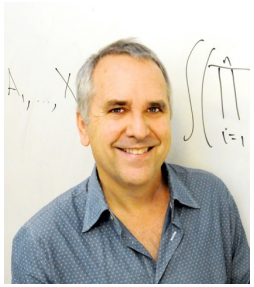
- How to build computers that:
  - (automatically) improve their performance (**P**)
  - at some task (**T**)
  - with experience (**E**)

Tom Mitchell

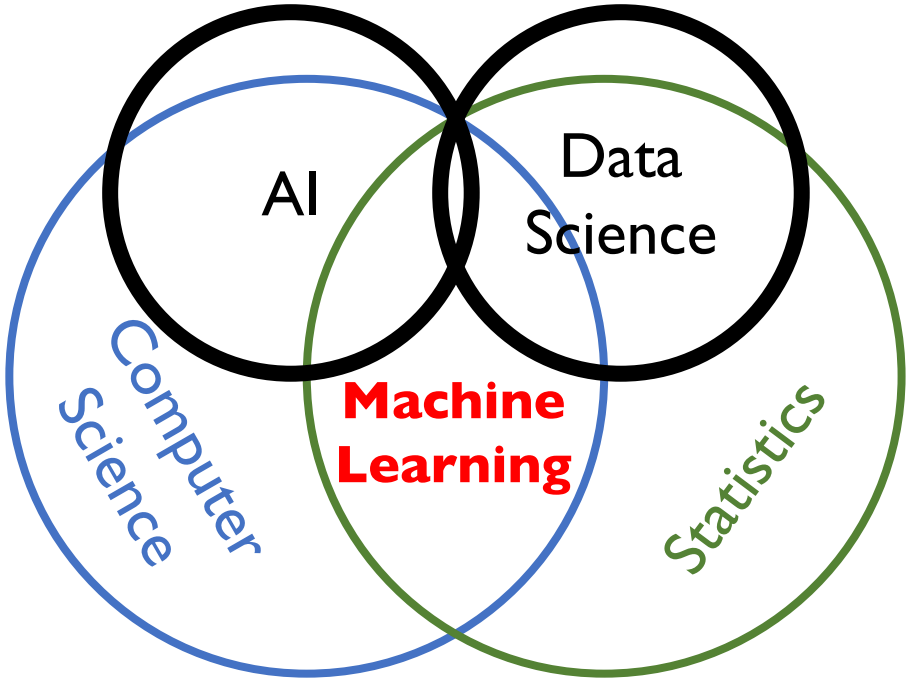


Mitchell, Tom M. Machine learning. 1997. Burr Ridge, IL: McGraw Hill 45.37 (1997): 870-877.

Machine Learning



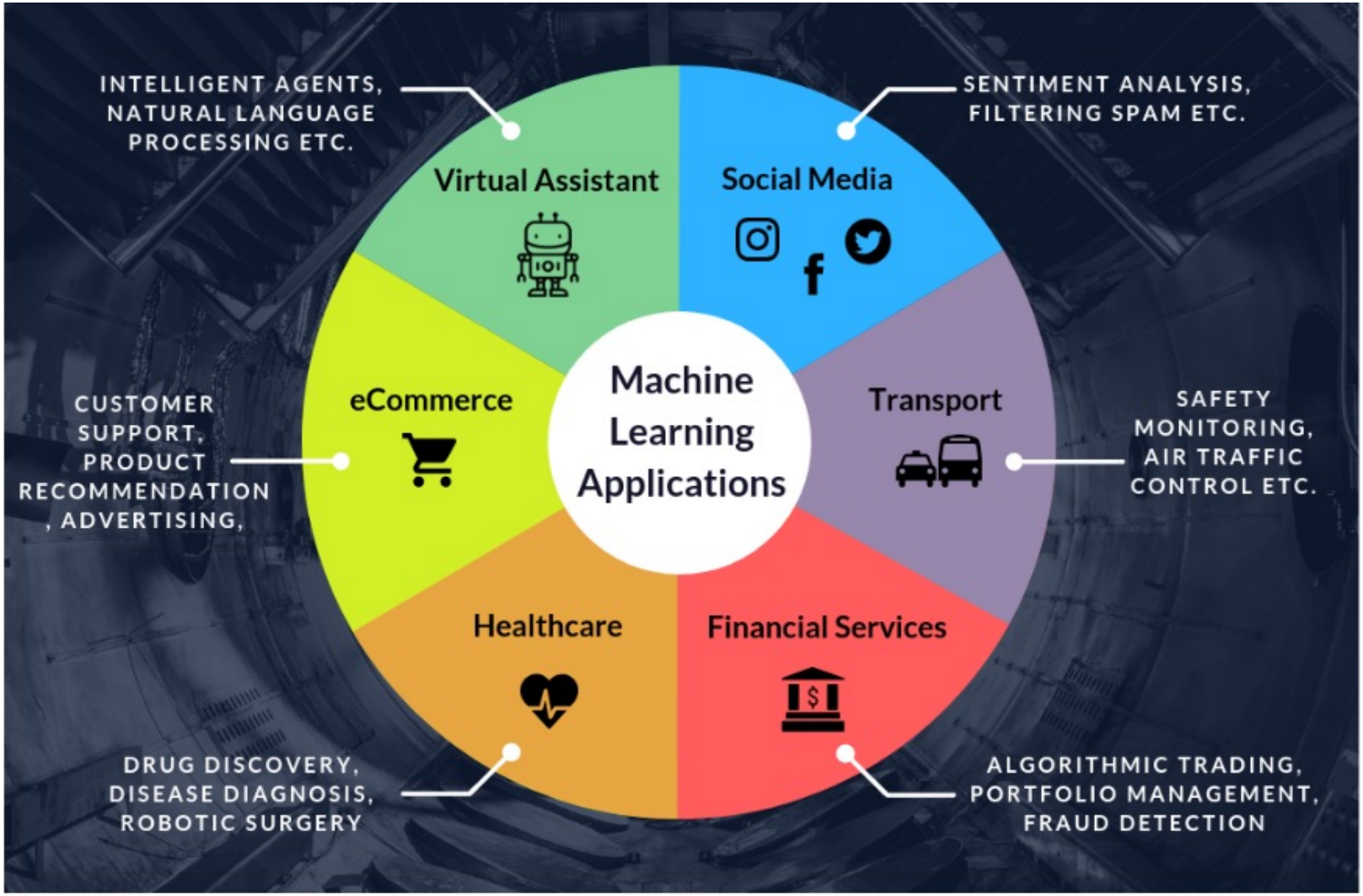
Michael Jordan



At the intersection of computer science and statistics, and at the core of artificial intelligence and data science.

M. I. Jordan1, T. M. Mitchell. Machine learning: Trends, perspectives, and prospects. **Science**. 2015.

Explosive Applications of Machine Learning



Outline

- **Machine Learning**
  - **The Framework**
  - No Free Lunch Theorem
  - Model Selection
- K-Nearest Neighbors (KNN)
  - Curse of Dimensionality

Components of Learning

Formalization:

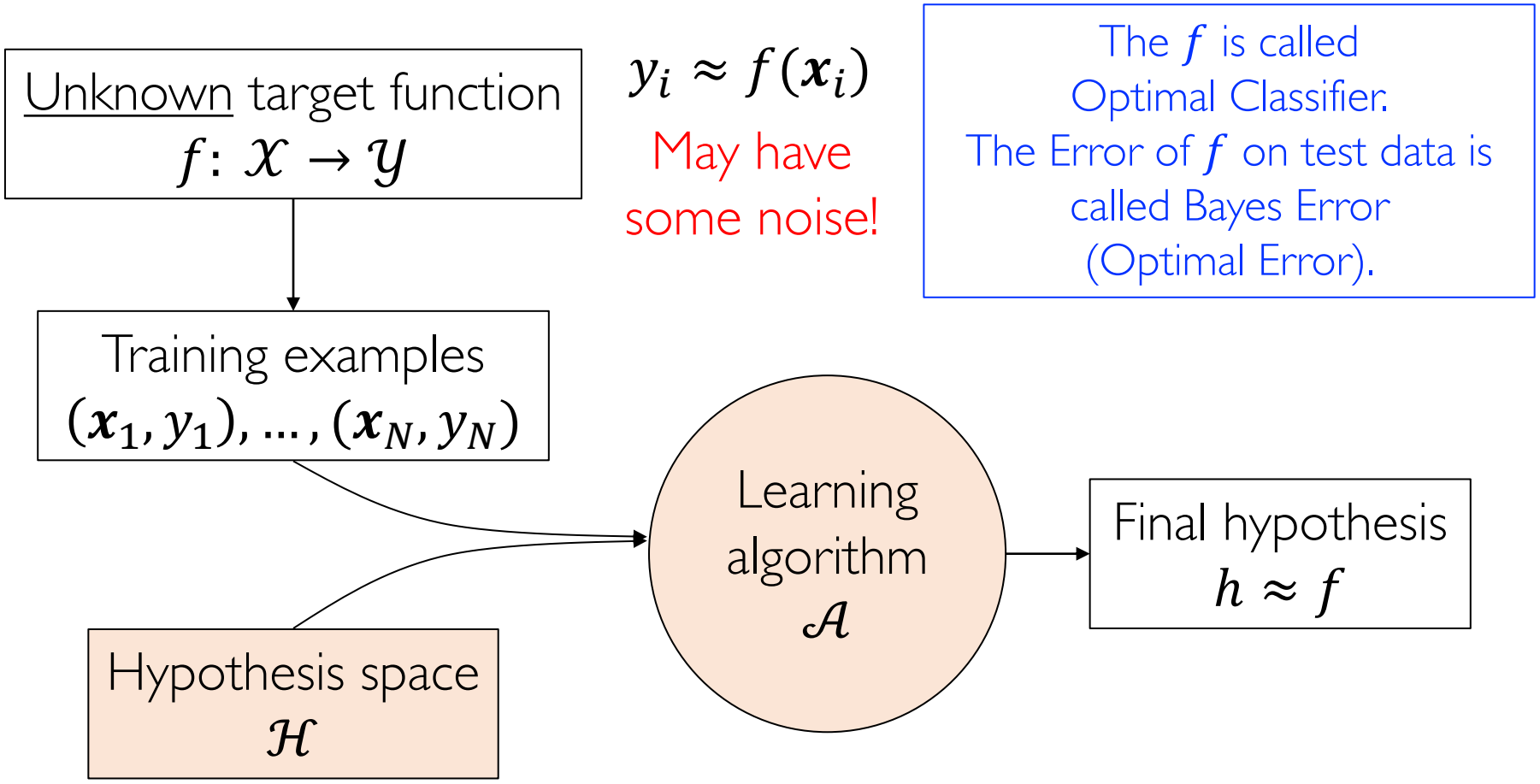
- Input: $\boldsymbol{x} \in \mathbb{R}^d$ (Email Features)
- Output: $y = \{0,1\}$ (Email Classes)
  - 0 for Not-spam and 1 for Spam
- Target function: $f: \mathcal{X} \rightarrow \mathcal{Y}$ (Ideal Spam Filter → unknown)
- Data: $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)$ (Dataset of Spam/Not-Spam emails)

Training ↓

Supervised Learning

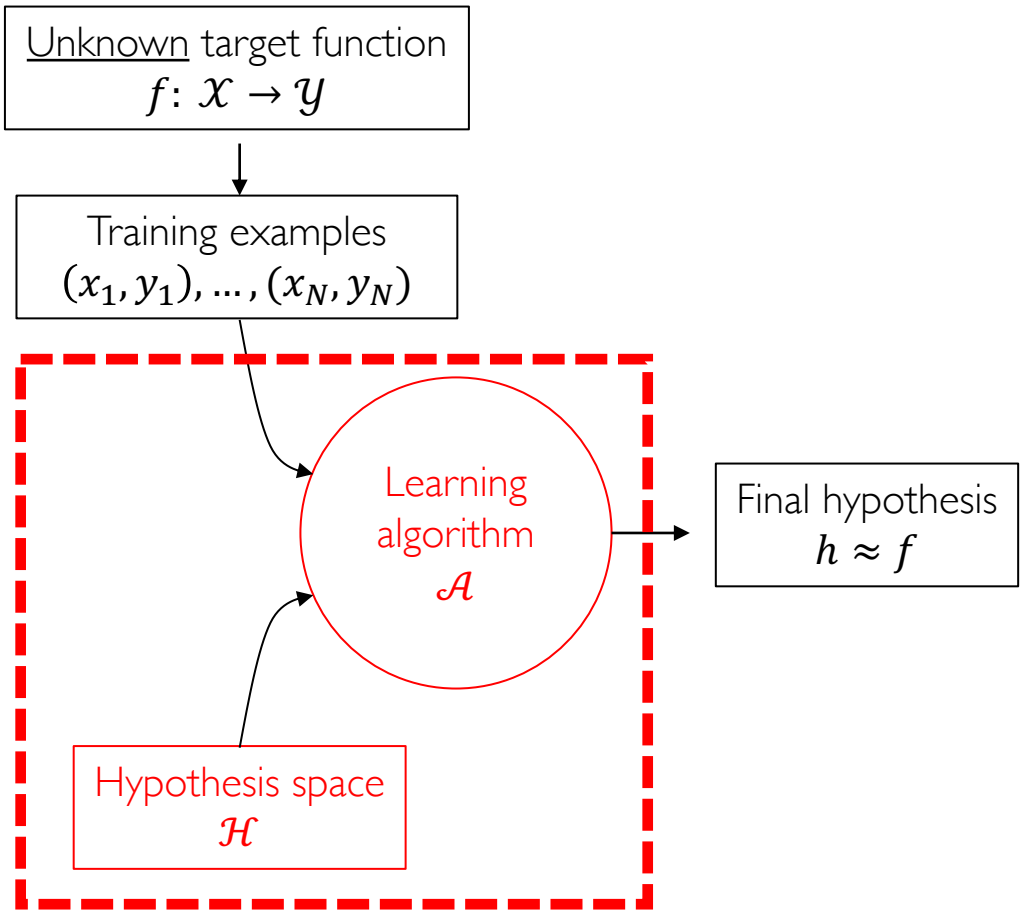
- Hypothesis (假设): $h: \mathcal{X} \rightarrow \mathcal{Y}$ (Formula to be used for prediction)

Components of Learning



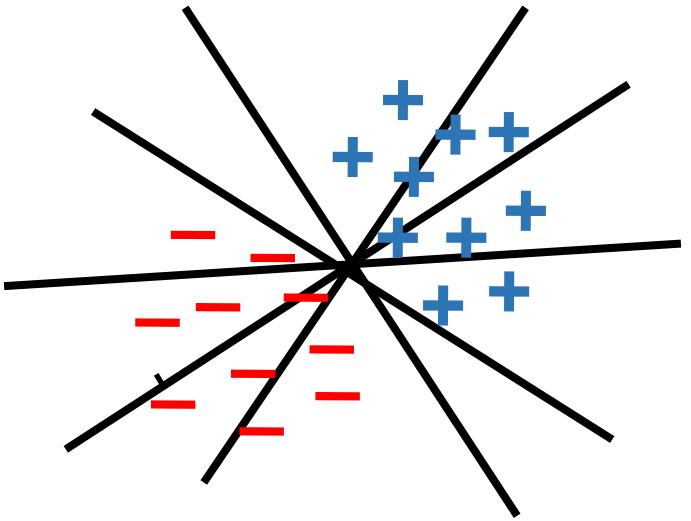
Components of Learning

- The two core components of the learning problem:
  - The hypothesis space
    $\mathcal{H} = \{h\} \quad h \in \mathcal{H}$
  - The learning algorithm $\mathcal{A}$
    - To find the best h
- They are together referred to as the learning model



Hypothesis Space (假设空间)

- A hypothesis space $\mathcal{H}$ is a set of functions that maps $\mathcal{X} \rightarrow \mathcal{Y}$.
  - It is the collection of prediction functions we are choosing from.
- We want hypothesis space that…
  - Includes only those functions that have desired regularity (正则性)
    - Continuity (连续性)
    - Smoothness (光滑性)
    - Simplicity (简单性)
  - Easy to work with
- An example hypothesis space:
  - All linear hyperplanes for classification



How to find the best?

Loss Function (损失函数)

- Loss function: $\ell: \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$ measures the difference between a prediction $h(\boldsymbol{x})$ and an actual output y:

$$\ell(y, h(\boldsymbol{x})) = (y - h(\boldsymbol{x}))^2 \text{ (Regression)}$$

$$\ell(y, h(\boldsymbol{x})) = \mathbf{1}[y \neq h(\boldsymbol{x})] \text{ (Classification)}$$

- The canonical training procedure of machine learning:

Fit dataset with best hypothesis

$$\hat{\epsilon}(h) = \min_{\theta} \sum_{i=1}^{m} \ell(h_\theta(\boldsymbol{x}_i), y_i)$$

θ is parameters of the hypothesis h

- Virtually every machine learning algorithm has this form, just specify
  - What is the hypothesis function?
  - What is the loss function?
  - How do we solve the training problem?

Machine Learning Paradigms

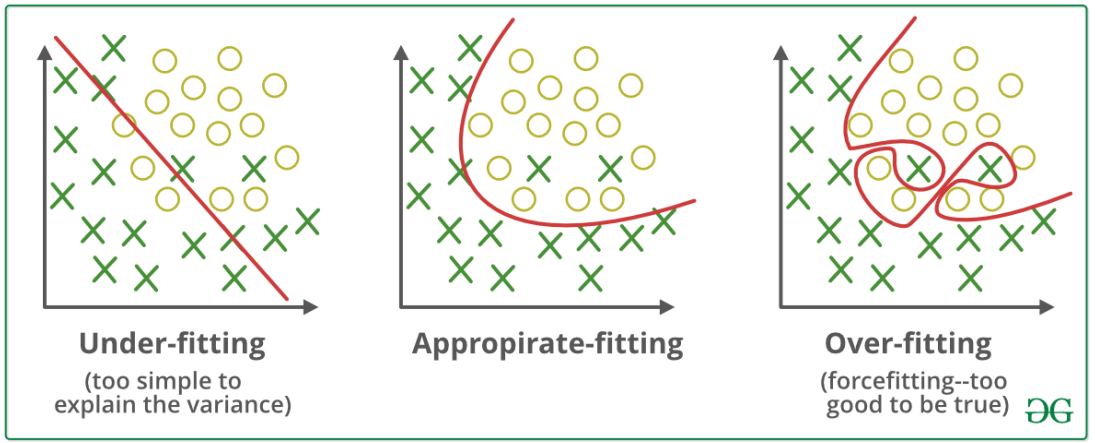
	Supervised Learning	Unsupervised Learning
Discreet Output	Classification (分类)	Clustering (聚类)
Continuous Output	Regression (回归)	Embedding (嵌入)

Supervised Learning

Known y at training

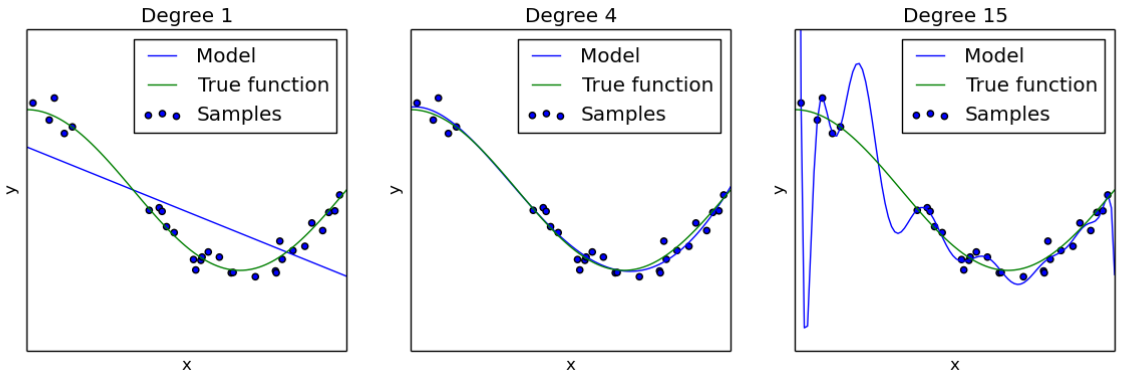
Learner: $h(\boldsymbol{x}) = y$

- Classification:



Most successful in practice

- Regression:

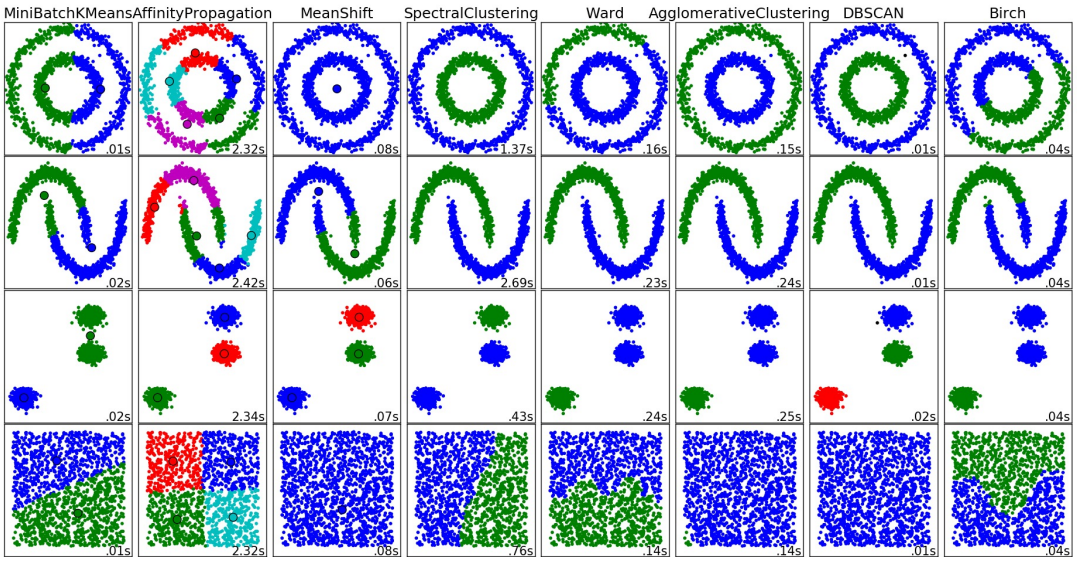


Unsupervised Learning

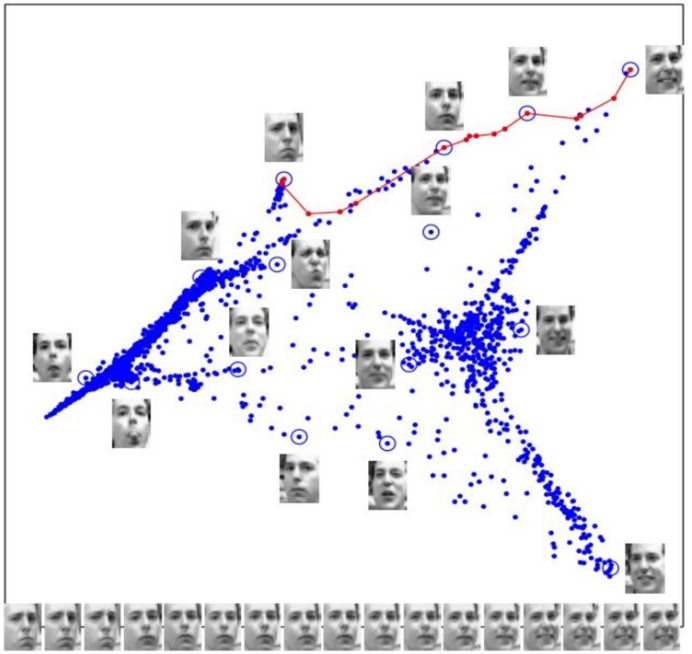
Learner: $h(\boldsymbol{x}) = y$

Unknown y at training

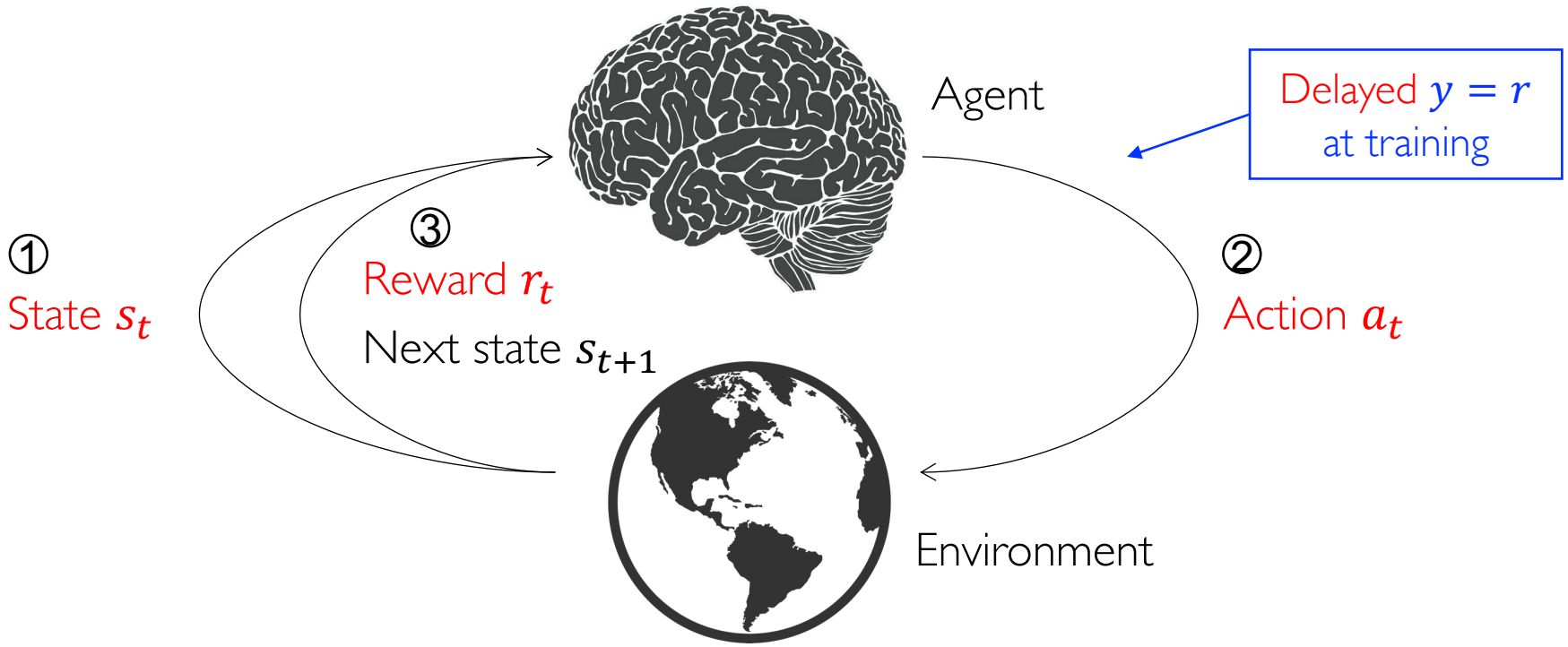
- Clustering:



- Embedding



Reinforcement Learning



At each step t the agent:

- Executes action a_t
- Receives state s_t
- Receives scalar reward r_t

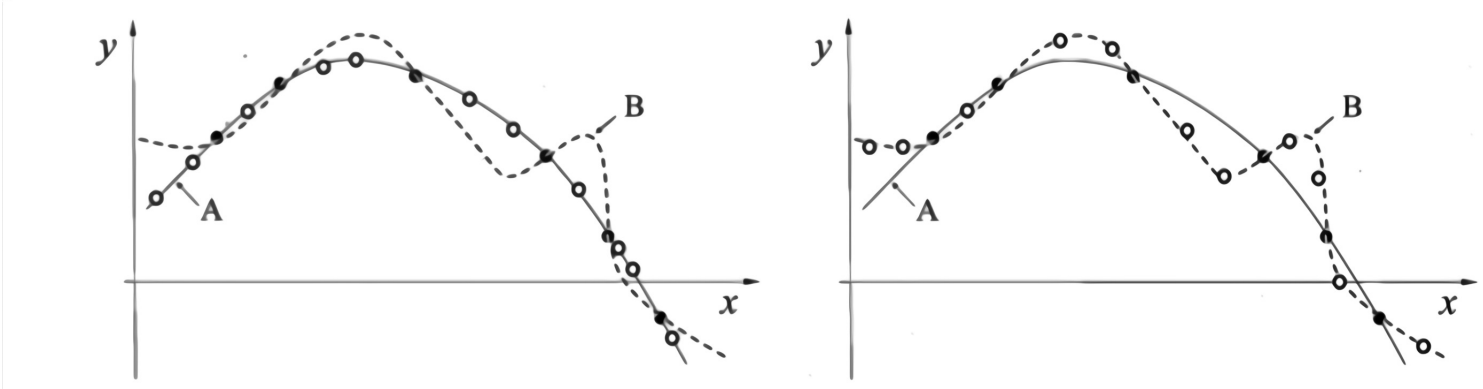
At each step t the environment:

- Receives action a_t
- Emits state s_{t+1}
- Emits scalar reward r_{t+1}

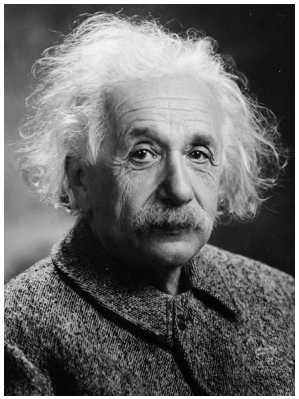
Outline

- **Machine Learning**
  - The Framework
  - **No Free Lunch Theorem**
  - Model Selection
- K-Nearest Neighbors (KNN)
  - Curse of Dimensionality

Ockham's Razor



Sometimes, A is better than B.



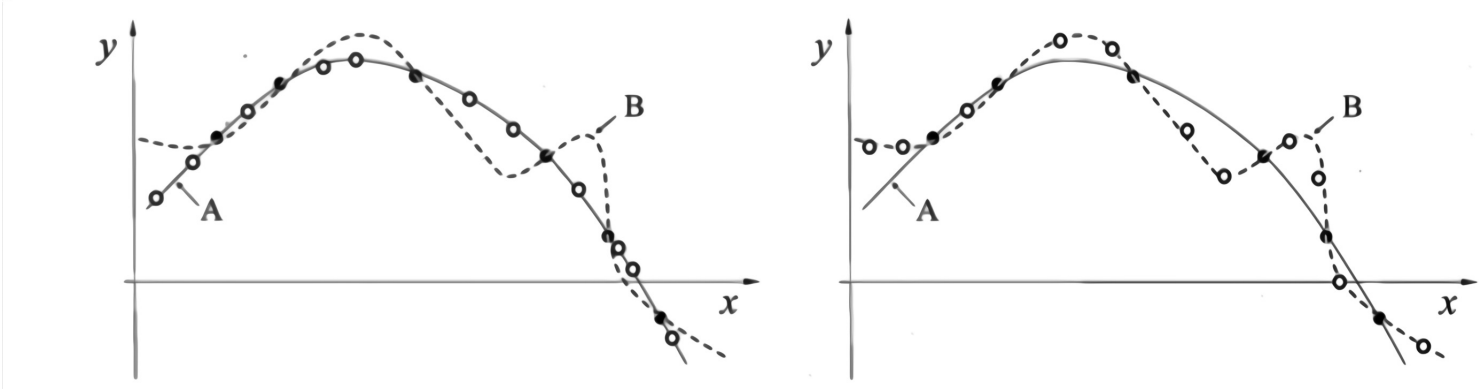
Everything should be made as simple as possible, but no simpler.

—Albert Einstein

Plurality must never be posited without necessity.

—William of Ockham

No Free Lunch Theorem



Sometimes, A is better than B.

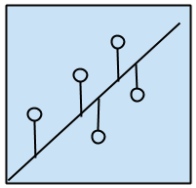
Now, B is better. 😅

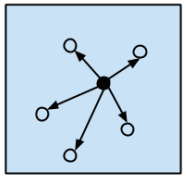
- Theorem: There are no universally best models.
  - Specific assumption to find a specific model for a specific problem!

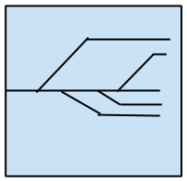
Averaged over all problems, no machine algorithm is expected to perform better than any other algorithm.

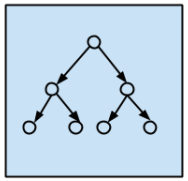
Wolpert. The Lack of A Priori Distinctions Between Learning Algorithms. 1996.

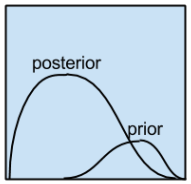
All Models Are Wrong…


Regression Algorithms

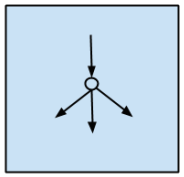

Instance-based Algorithms

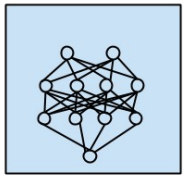

Regularization Algorithms

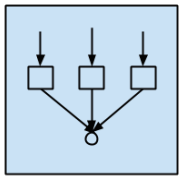

Decision Tree Algorithms



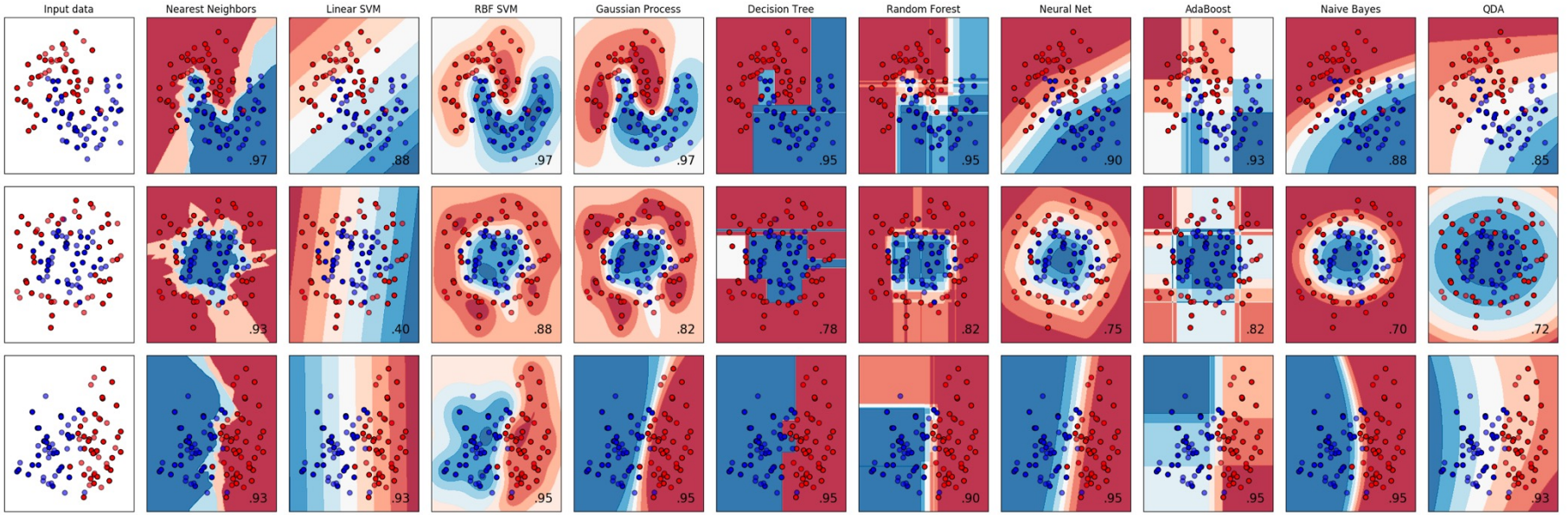
Bayesian Algorithms


Artificial Neural Network Algorithms


Deep Learning Algorithms


Ensemble Algorithms

All models are wrong, but some are useful (with prior knowledge).
-G. E. P. Box

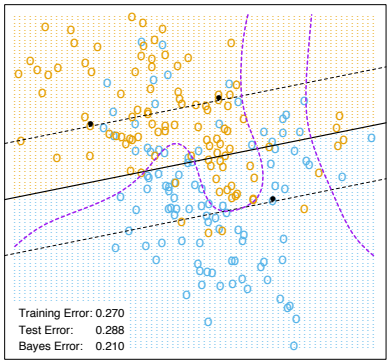


Outline

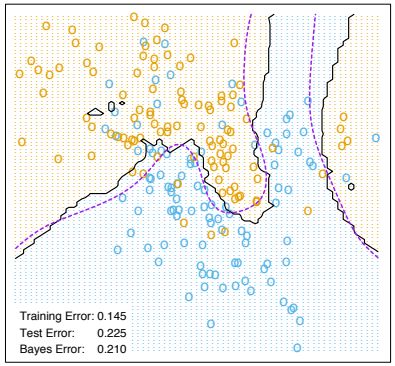
- **Machine Learning**
  - The Framework
  - No Free Lunch Theorem
  - **Model Selection**
- K-Nearest Neighbors (KNN)
  - Curse of Dimensionality

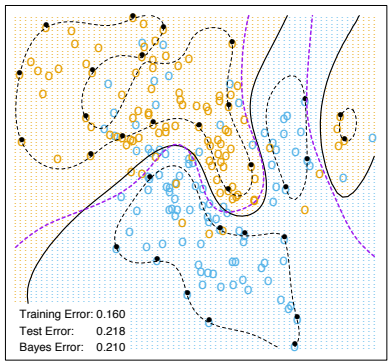
Model Selection

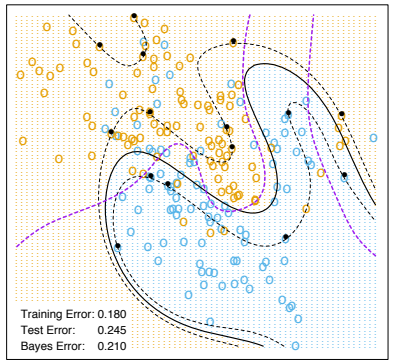
- There are too many candidate learning algorithms or models.
  - Each of which can be fit well to data and used for prediction.



$C = 10000$







Different Algorithms

Linear Classifier K-nearest-neighbor

Different Hyperparameters

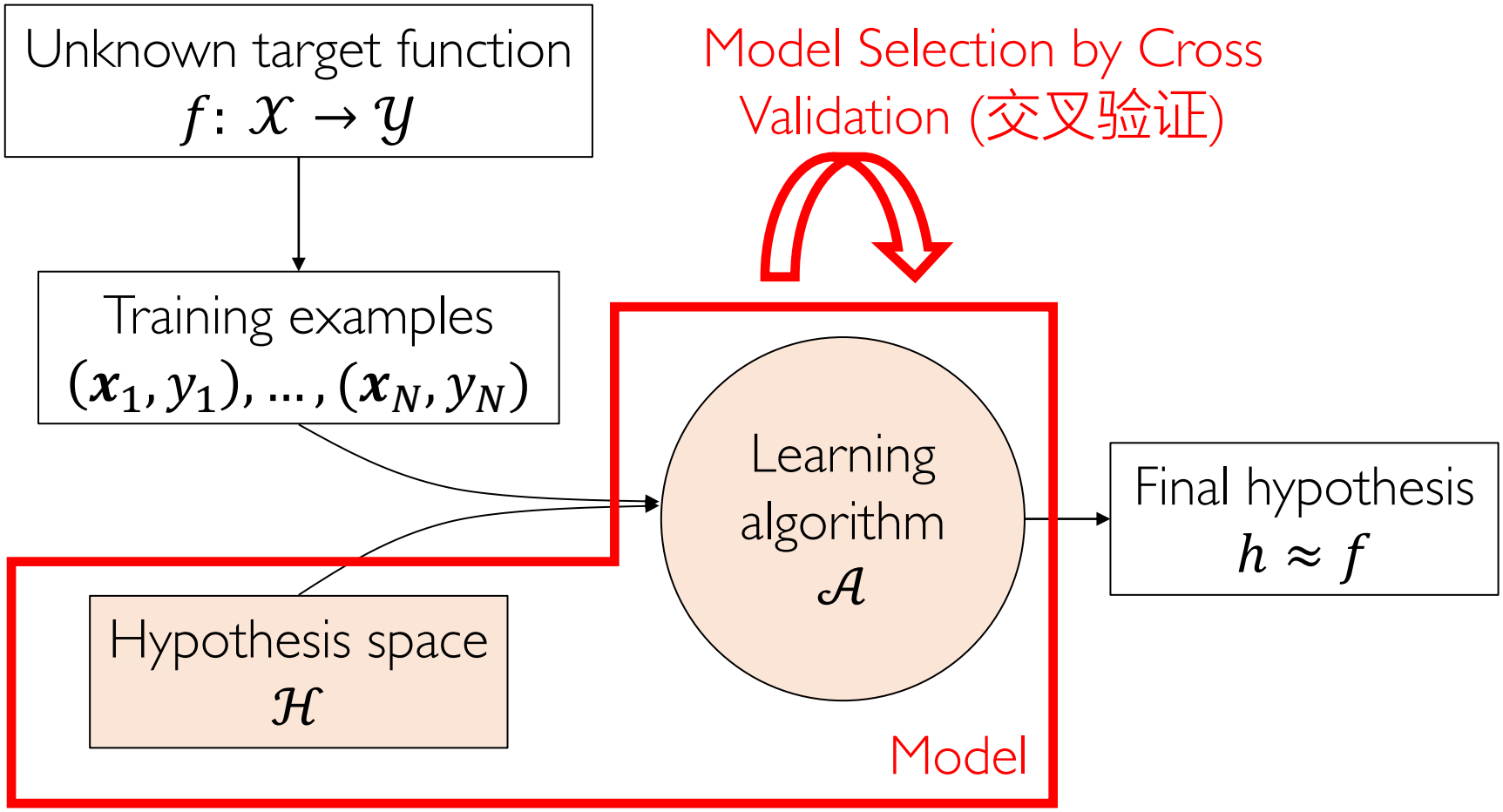
Kernel SVM (same algorithm)

  - How can we decide which is best?
  - Which part of features should we use?

Tuner?
(调参侠)

- We need model selection.

Components of Learning



Training and Test Data

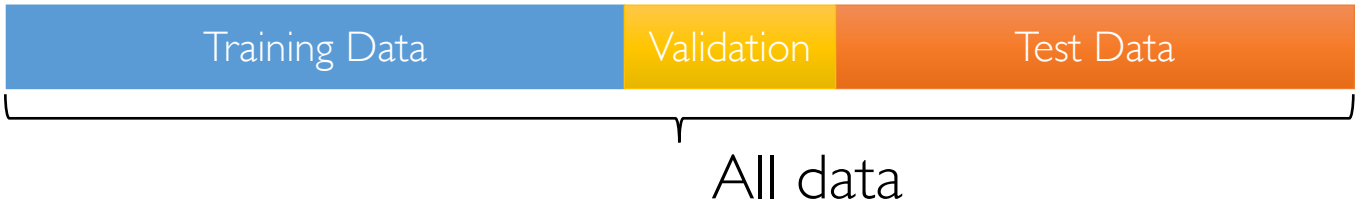
- How can we decide which algorithm or model is best?

- Train the parameters θ of each model h_θ from the training data.
- Evaluate on the test data, and pick the best performer.
- Problems:
  - Over-estimates the test performance ("lucky" model)
  - This is the most common but wrong practice of machine learning.

The learning algorithms should never ever have access to test data!

Training and Test Data

- Reserve some data for validation.



- Train the parameters θ of each model h_θ from the training data.
- Evaluate on the validation data, and pick the best performer.
- Reserve test data to benchmark the chosen model.
- Problems:
  - Wasteful of training data (learning cannot use validation data).
  - May bias the selection towards overly simple models.

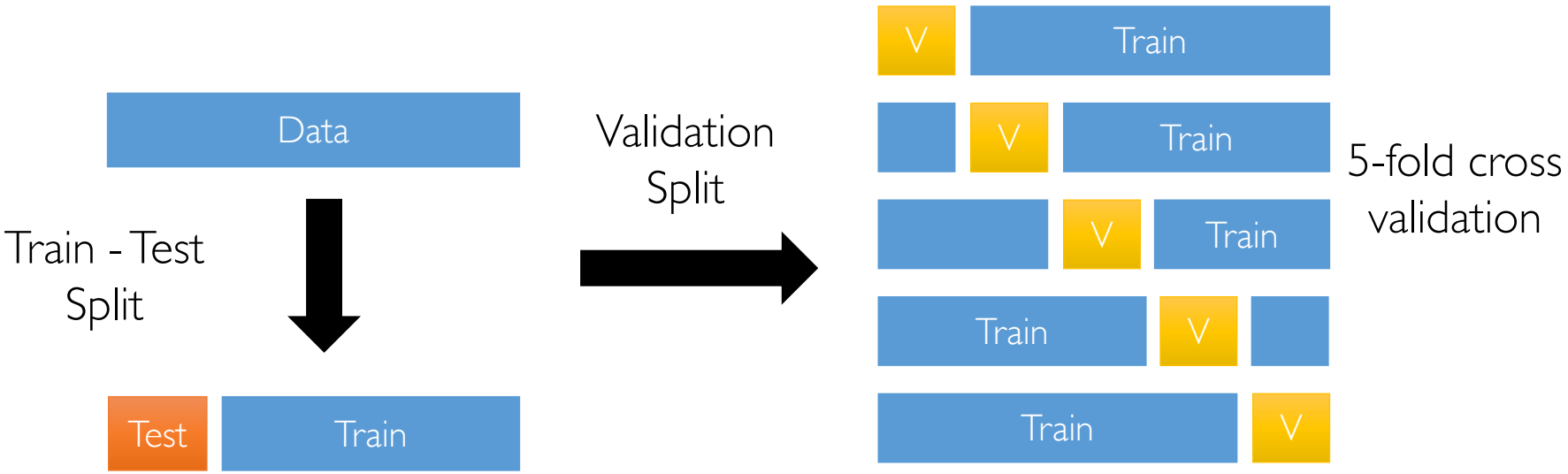
The learning algorithms should never ever have access to test data!

Cross Validation (交叉验证)

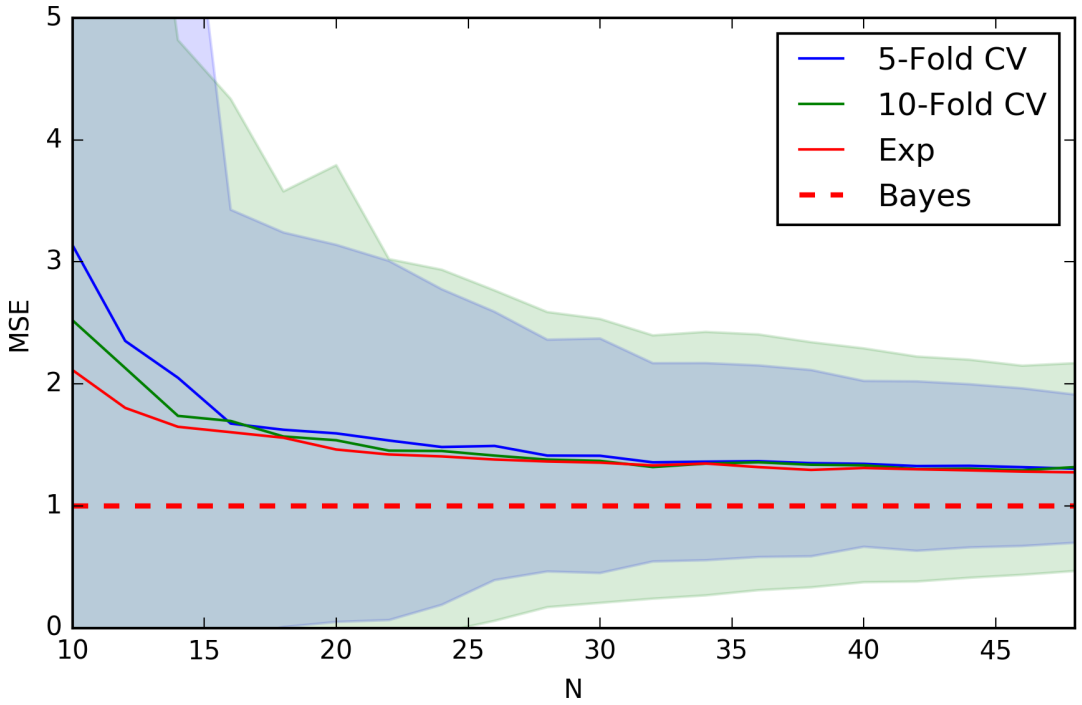
- What if we get an unfortunate split?

Used in small-data regime

- K-fold Cross Validation:
  - Split the data set $\mathcal{D}$ into K subsets $\mathcal{D}^{(i)}$ (called folds)
  - For $i = 1, \ldots, K$, train $h^{(i)}$ using all data but the i-*th* fold ($\mathcal{D} \backslash \mathcal{D}^{(i)}$).
  - Cross-validation error by averaging all validation errors $\hat{\epsilon}_{\mathcal{D}^{(i)}}(h^{(i)})$



Cross Validation



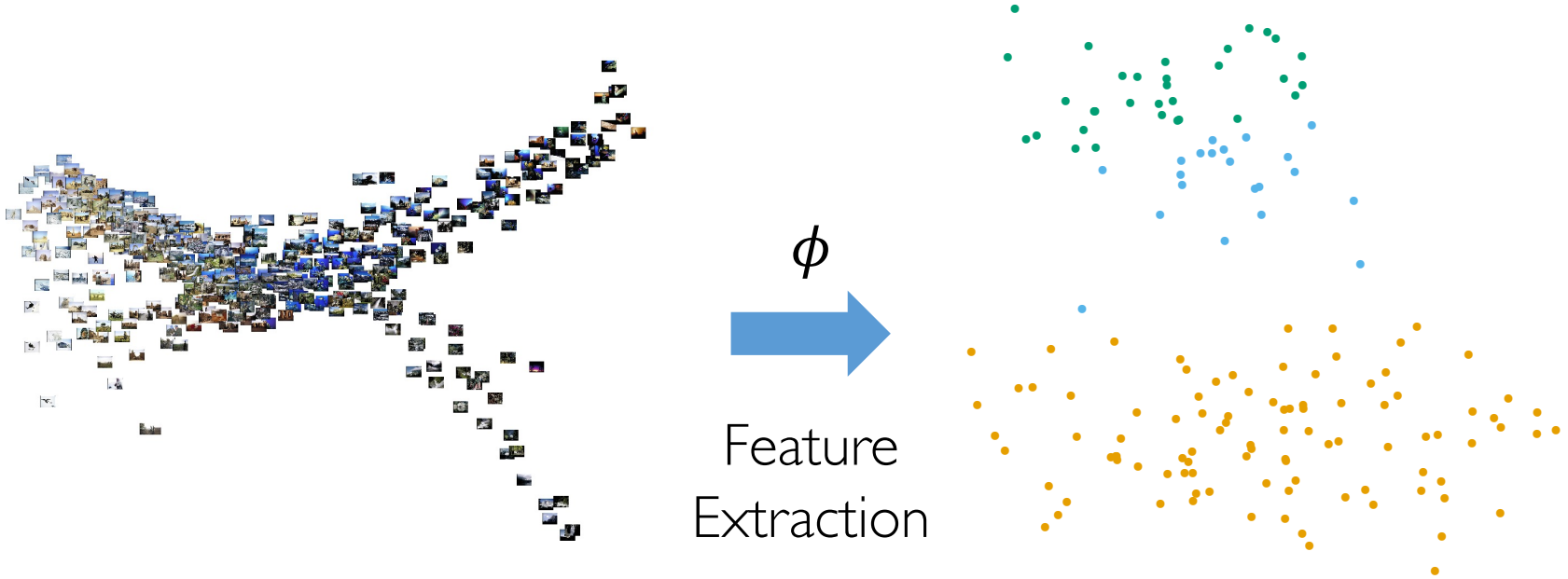
The number of runs scales linearly with K

- In practice, we always choose K as 5 or 10.
- If we choose K as N (the size of $\mathcal{D}$), we get Leave-One-Out Cross Validation (LOOCV) – only for small data, too costly for big data.

Outline

- Machine Learning
  - The Framework
  - No Free Lunch Theorem
  - Model Selection
- **K-Nearest Neighbors (KNN)**
  - Curse of Dimensionality

Machine Learning: Geometric View

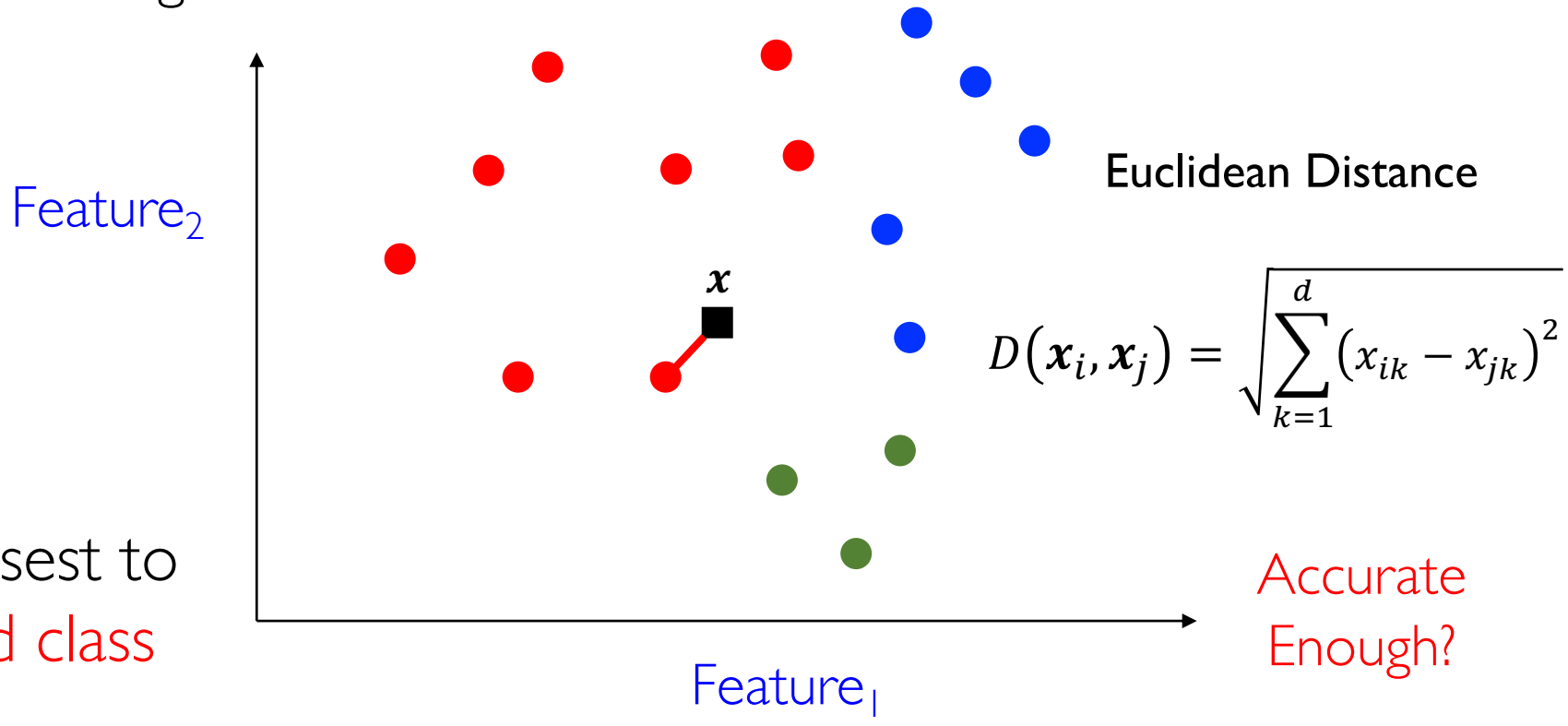


- We can view examples as points in an d-dimensional space where d is the number of features.
- **Assumption:** Closer points in feature space have similar semantics.

Birds of a feather flock together. (物以类聚)

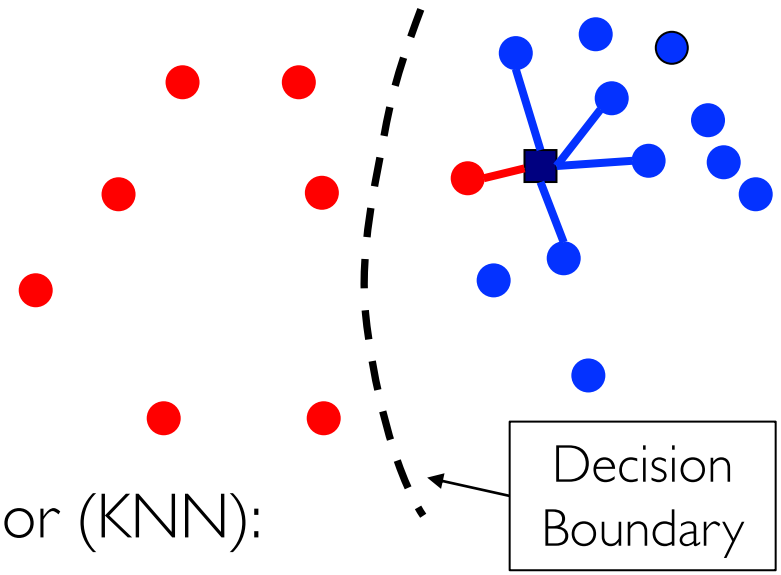
Nearest-Neighbor (1NN)

- To classify a new example $\boldsymbol{x}$:
  - Label $\boldsymbol{x}$ with the label of the closest example to $\boldsymbol{x}$ in the training set.



K-Nearest-Neighbors (KNN, K近邻)

- $\boldsymbol{x}$ is close to a red point.
- But most of the next closest points are blue.



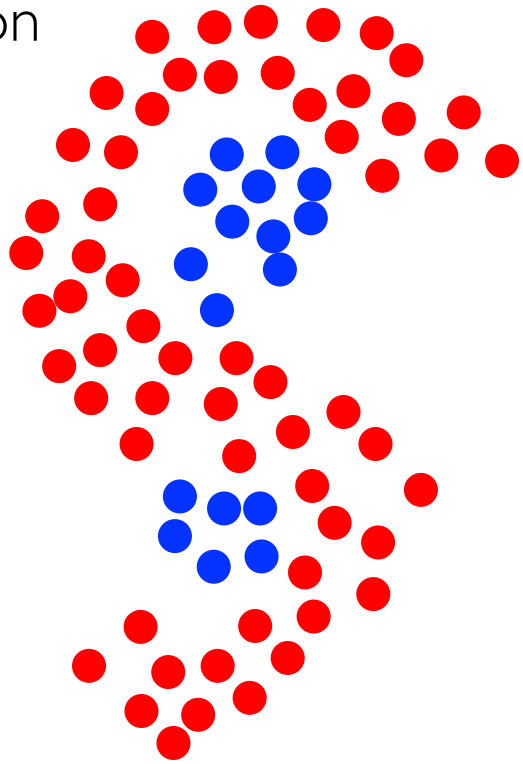
Model-free!

- K-Nearest-Neighbor (KNN):
  - Find $\boldsymbol{k}$ nearest neighbors of $\boldsymbol{x}$.
  - Label $\boldsymbol{x}$ with the majority label within the $\boldsymbol{k}$ nearest neighbors
  - Consider: How can we handle ties for even values of $\boldsymbol{k}$?

K-Nearest-Neighbors (KNN)

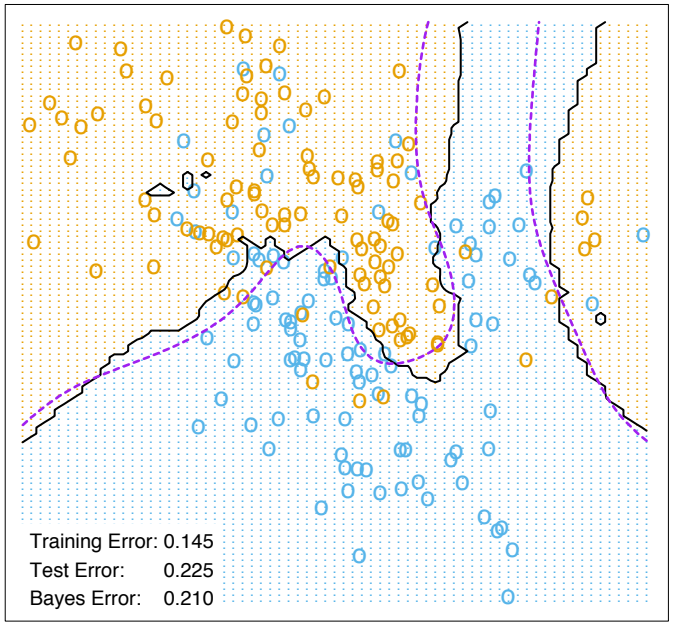
- KNN rule is certainly simple and intuitive, but does it work?
- KNN is a Good Approximator on any smooth distribution:
  - Converges to perfect solution if clear separation
  - $\lim_{n\to\infty}\epsilon_{\mathbf{KNN}}(n) \leq 2\epsilon^*(1-\epsilon^*)$

  (Cover & Hart, 1969) Bayes Error
- KNN is Good Model on complex distributions.
  - Most parametric distributions would not work for this 2-class classification problem (right)
  - Nearest neighbors will do reasonably well, provided that we have enough samples.

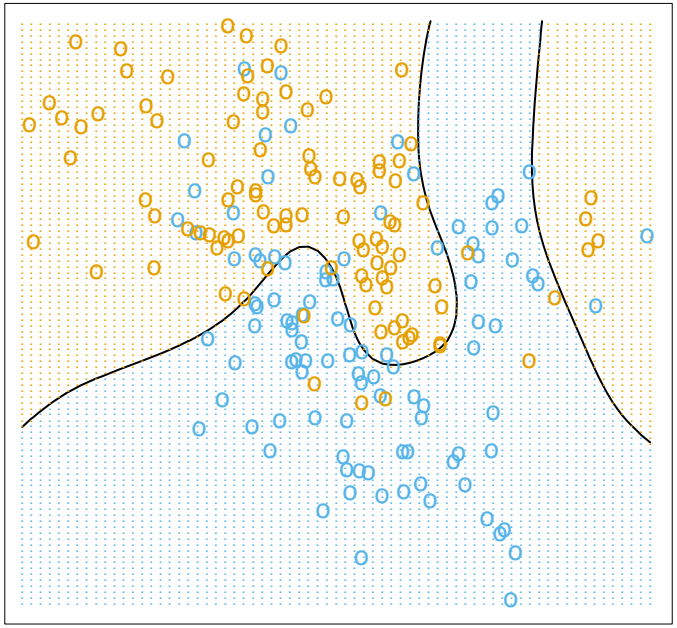


KNN: Empirical Efficacy

KNN is a Good Approximator for complex distributions.



≈



7-Nearest-Neighbor

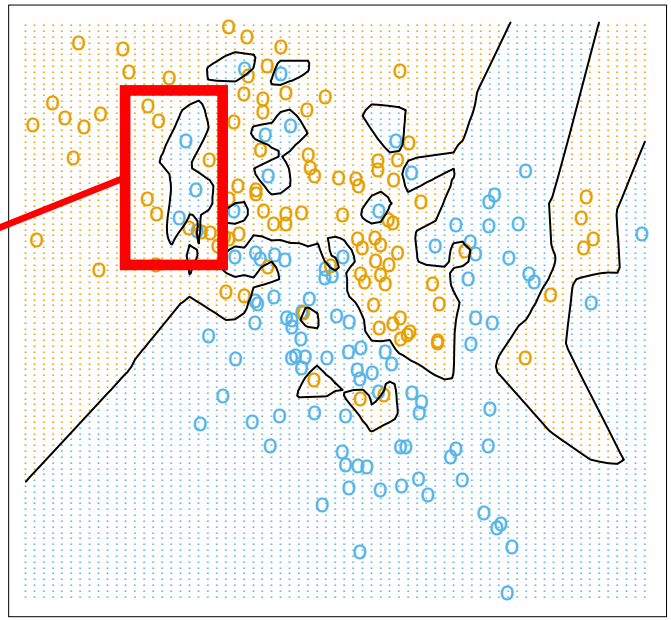
Target Decision Boundary

The Effect of K

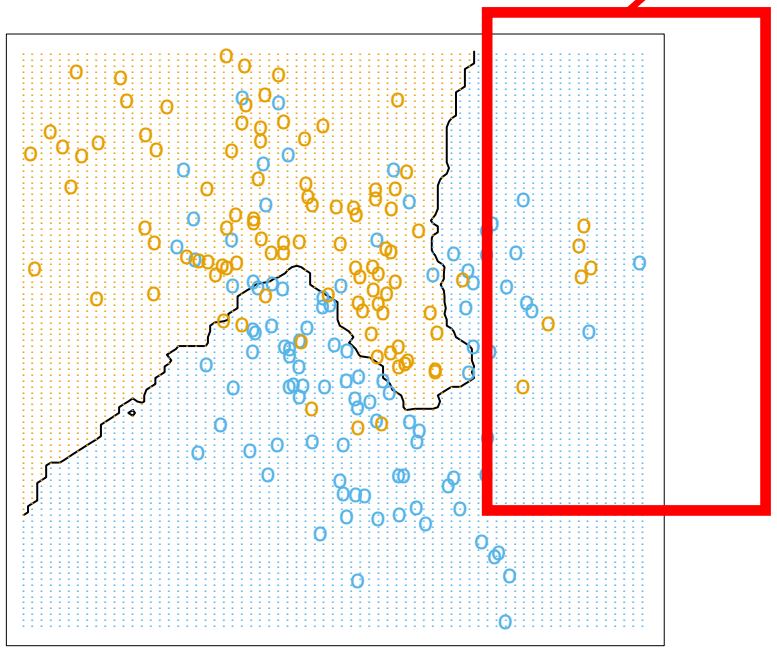
- Increasing k simplifies the decision boundary
  - Majority voting means less emphasis on individual points

Smooth Boundary

Fit Noise

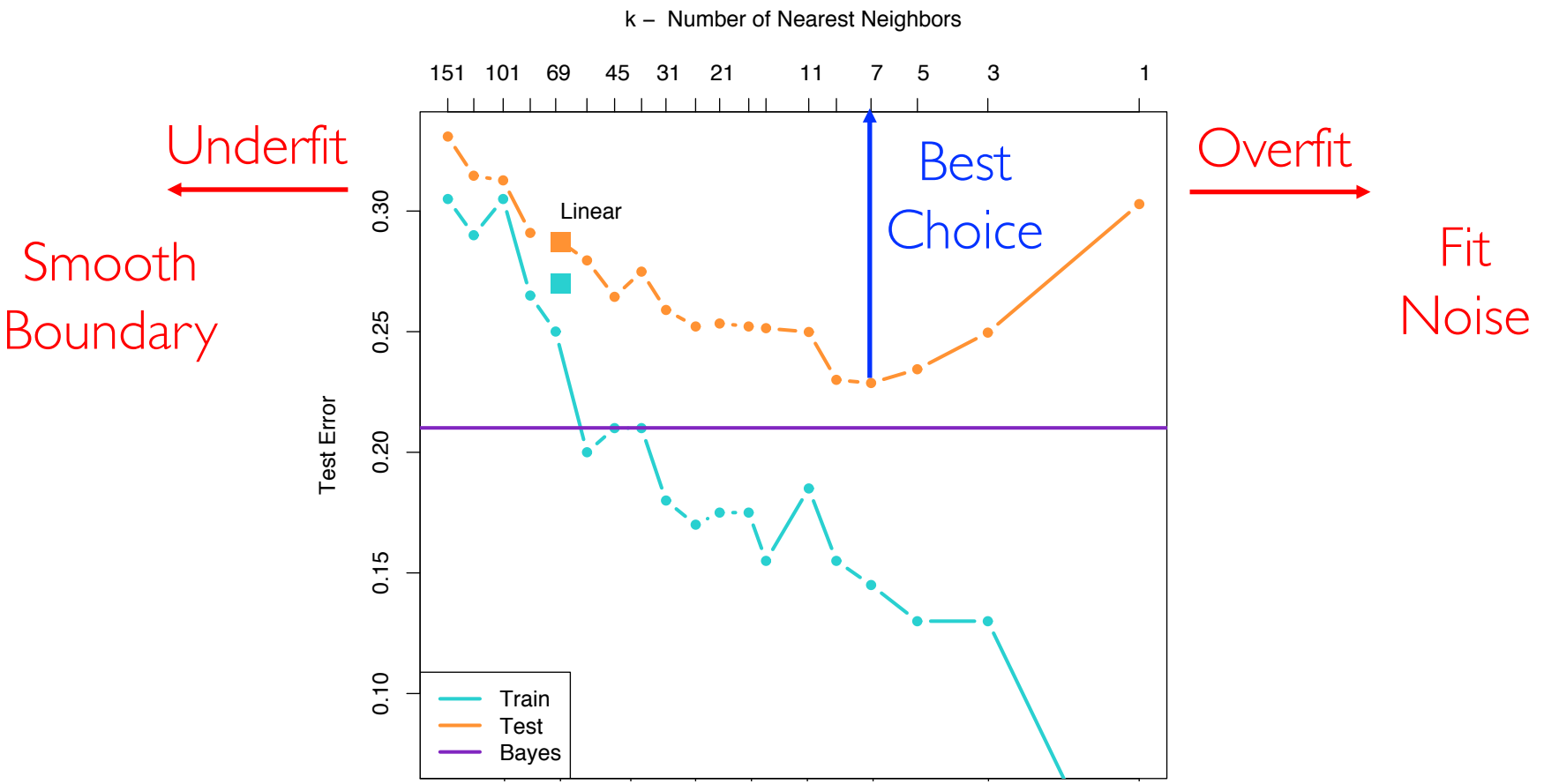


1-Nearest-Neighbor



15-Nearest-Neighbors

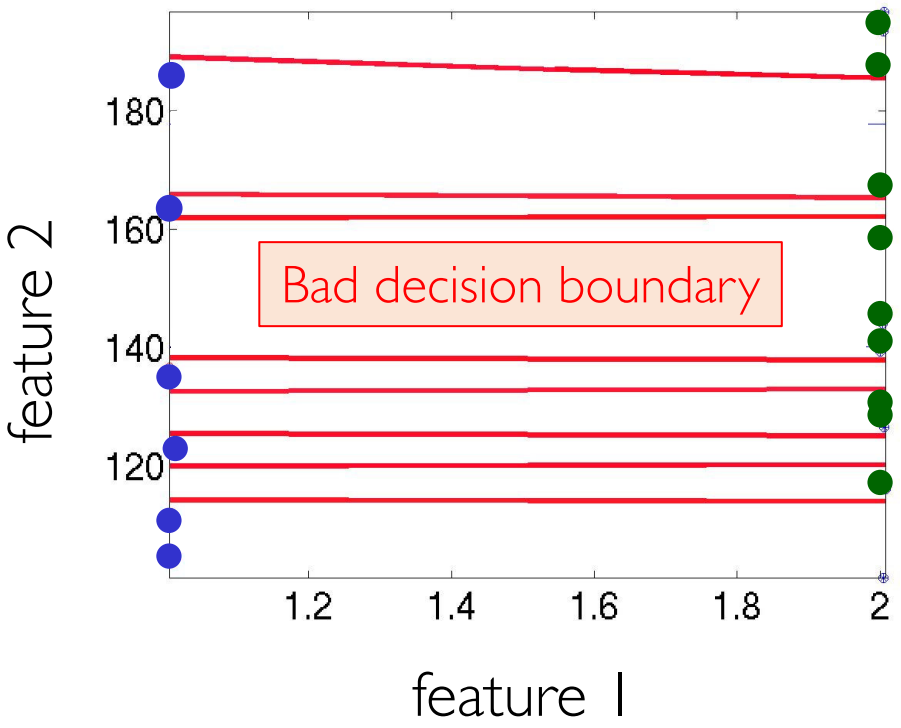
The Effect of K



- Choose best k on the validation set. (Often set k as an odd number).

KNN: Inductive Bias

- Inductive bias (归纳偏好):
  - Similar points have similar labels. All dimensions are created equal!



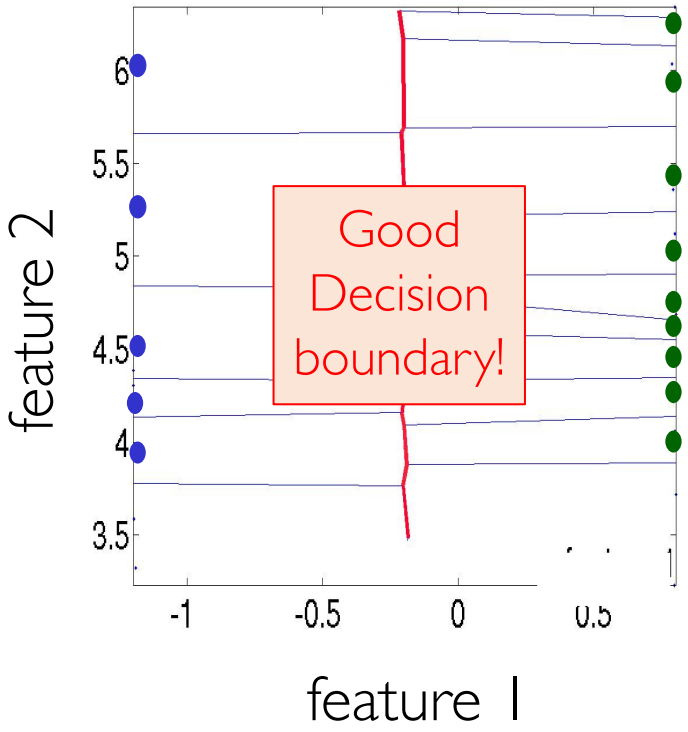
Why?

Feature scale could dramatically influence classification results ☹

Feature Normalization

- We could normalize each feature to be of mean 0 and variance 1.
- **Z-score normalization:**
  - For each feature dimension j, compute based on its samples:
  - Mean $\mu_j = \frac{1}{N}\sum_{i=1}^{N} x_{ij}$
  - Variance $\sigma_j = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_{ij} - \mu_j)^2}$
  - Normalize the feature into a new one:

$$\hat{x}_{ij} \leftarrow \frac{x_{ij} - \mu_j}{\sigma_j}$$



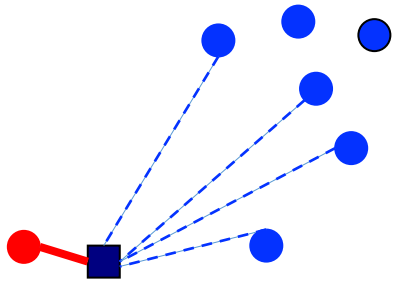
Distance Selection

- Cosine Distance: $\cos(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle}{\|\boldsymbol{x}_i\| \|\boldsymbol{x}_j\|}$
- Minkowski Distance: $D_p(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sqrt[p]{\sum_{k=1}^{d} |x_{ik} - x_{jk}|^p}$
- Mahalanobis Distance: $D_M(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sqrt{(\boldsymbol{x}_i - \boldsymbol{x}_j)^{\mathrm{T}} \boldsymbol{M} (\boldsymbol{x}_i - \boldsymbol{x}_j)}$
  - $\boldsymbol{M}$ should be specified a priori
  - $\boldsymbol{M}$ can also be learned from data → metric learning
- And many other distances (Wikipedia: https://en.wikipedia.org/wiki/Distance)
  - No distance actually performs universally better than the others.

Weighted KNN

- Vanilla KNN uses majority vote to predict discrete outputs.
- Weighted KNN assigns greater weights to closer neighbors.
  - **Weighting** is a common approach used in machine learning.
- Distance-weighted Classification:

$$\hat{y} = \text{argmax}_{c \in y} \sum_{\boldsymbol{x}' \in \text{KNN}_c(\boldsymbol{x})} \frac{1}{D(\boldsymbol{x}, \boldsymbol{x}')^2}$$



- Distance-weighted Regression:

$$\hat{y} = \sum_{\boldsymbol{x}' \in \text{KNN}(\boldsymbol{x})} \frac{1}{D(\boldsymbol{x}, \boldsymbol{x}')^2} y'$$

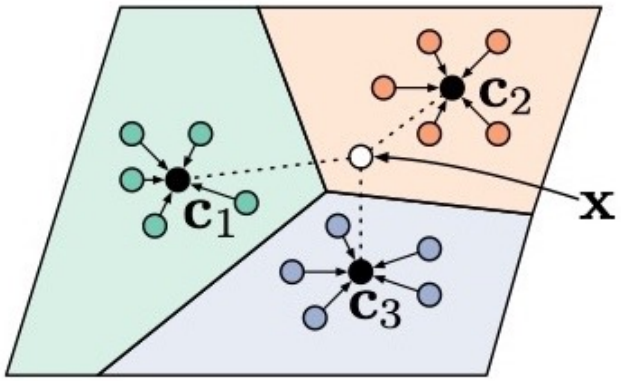
- Other weighting function (e.g. exponential family) can also be used.

Nearest Centroid Classifier

- A simple modification of the k-nearest-neighbor algorithm.
- Given dataset $\{\boldsymbol{x}_i, y_i\}_{i=1}^N$ with class label $y_i \in \mathcal{Y}$:
  - Compute per-class centroids $\boldsymbol{\mu}_c = \frac{1}{|N_c|}\sum_{y_i=c} \boldsymbol{x}_i$.
  - Prediction function: $\hat{y} = \mathbf{argmin}_{c \in \mathcal{Y}} \|\boldsymbol{x} - \boldsymbol{\mu}_c\|$

Nearest Centroid

- $O(Nd)$ training and $O(Cd)$ testing.
  - Faster!
- Gaussian assumption for each class.
  - Cannot tackle complex distributions.
  - Very good for Few-Shot Learning (FSL).

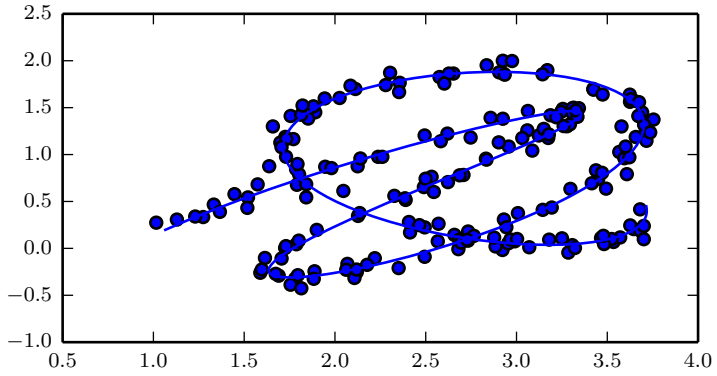


Outline

- Machine Learning
  - The Framework
  - No Free Lunch Theorem
  - Model Selection
- **K-Nearest Neighbors (KNN)**
  - **Curse of Dimensionality**

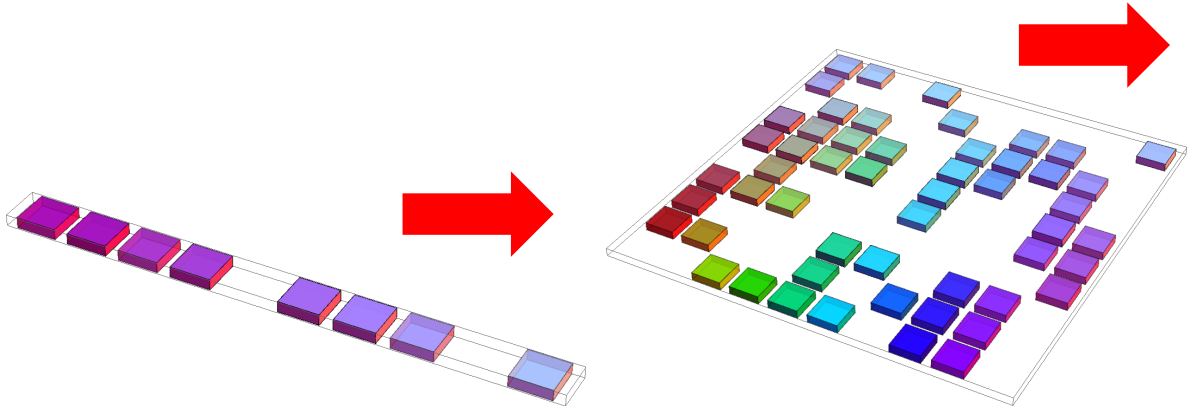
Curse of Dimensionality (维度灾难)

- Data distributed in low-dimension manifold of high-dimension space.



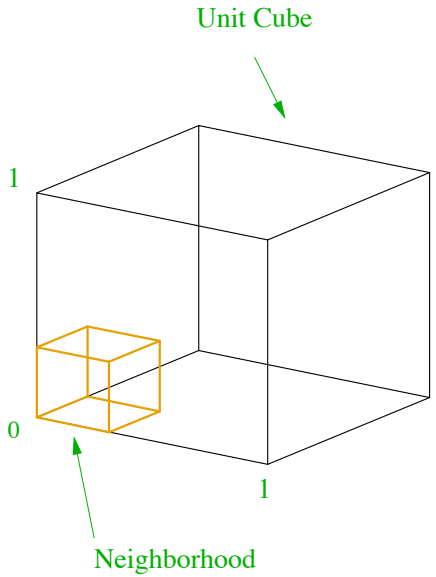
Manifold

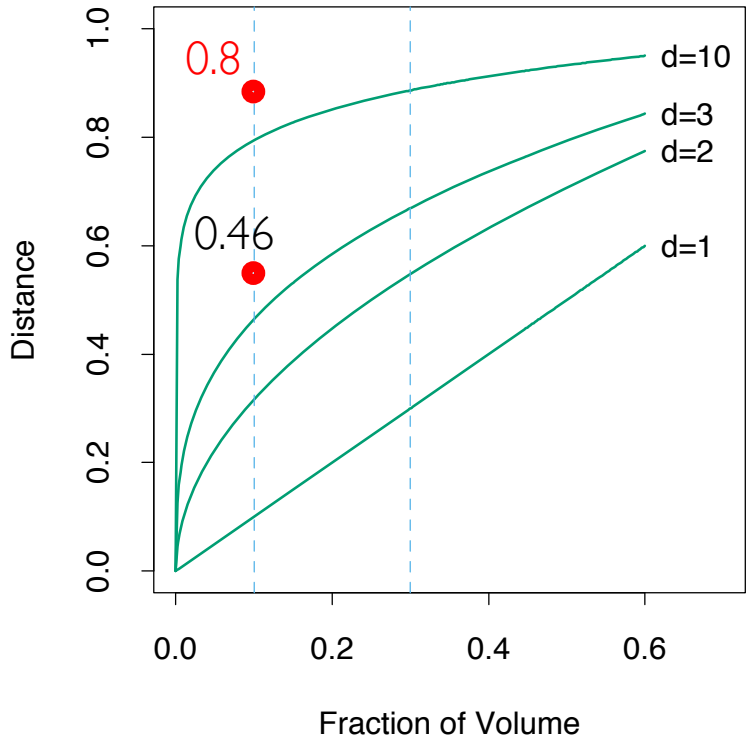
We only have data in the high-dimension space
We do not know the manifold (prior knowledge)



Curse of Dimensionality

- Assume your data lives in $[0,1]^d$.
  - To capture a neighborhood which represents a fraction s of the hypercube volume, you need the edge length to be $s^{1/d}$.
- $s = 0.1, d = 10, s^{1/d} = 0.8$

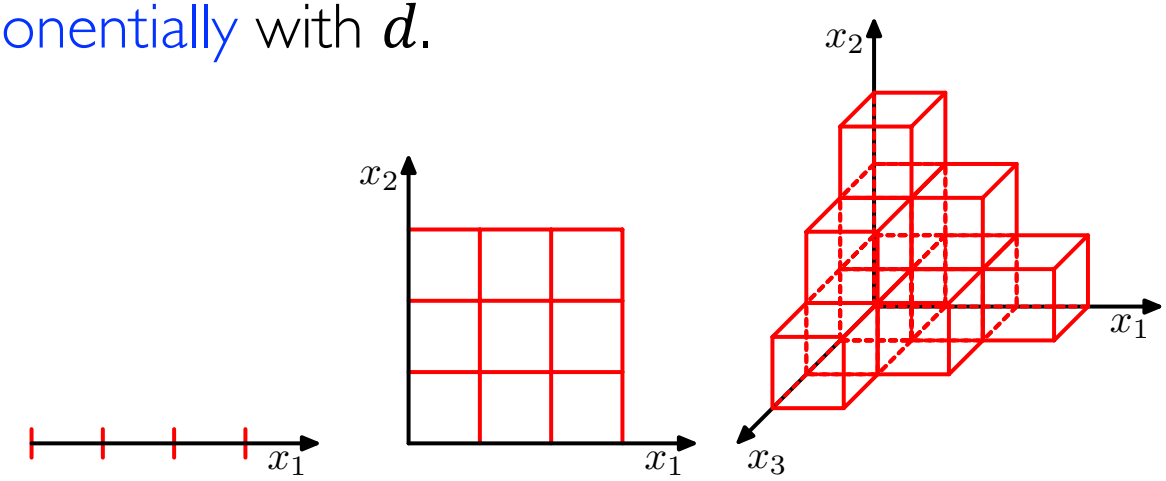




- Neighbors are no longer local!

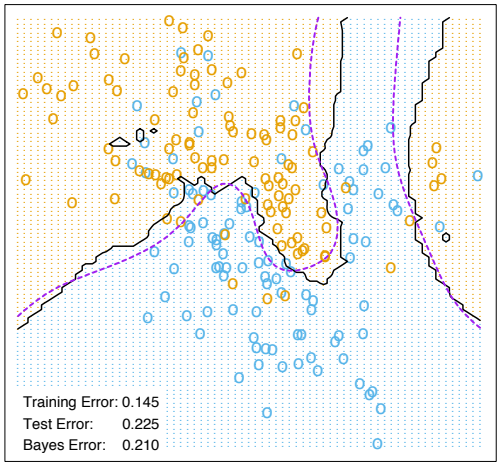
Curse of Dimensionality

- The volume of an hypercube with an edge length of $r = 0.1$ is 0.1^d.
  - When d grows, it quickly becomes so small that the probability to capture points becomes very small.
  - Points in high-dimensional spaces are isolated (very sparse).
- To overcome this limitation, you need a number of samples which grows exponentially with d.



KNN Summary

- **When to use:**
  - Few attributes per instance (expensive computation)
  - Lots of training data (curse of dimensionality)
- **Advantages:**
  - Agnostically learn complex target functions
  - Do not lose information (store original data)
  - Data number can be very large (big pro!)
  - Class number can be very large (biggest pro!)
    - All other ML algorithms may fail here!



- **Disadvantages:**
  - Slow at inference time (acceleration a must)
  - Ineffective in high dimensions (curse of dimensionality)
  - Fooled easily by irrelevant attributes (feature engineering crucial)