CMSC726
Lecture 14
Computational Learning Theory

Ben London

October 21, 2010

Introduction

- What is *computational learning theory* (CLT)?
  - Complexity and correctness of machine learning
    - Time complexity
    - Sample complexity
    - Generalization error
    - Robustness (to noise)
    - Hardness

Introduction

- Why should I care about CLT?
  - Having an understanding of learning theory will help you better understand the machine learning
  - Theory can lead to/explain/give bounds for application
  - Theory is fun! (?)

Introduction

- Goals for this lecture:
  - Review basic learning theory concepts
  - Discuss two classic learning paradigms
  - Quantify hypothesis complexity
  - Bound generalization error
  - Inspire you to get interested in CLT!

Terminology

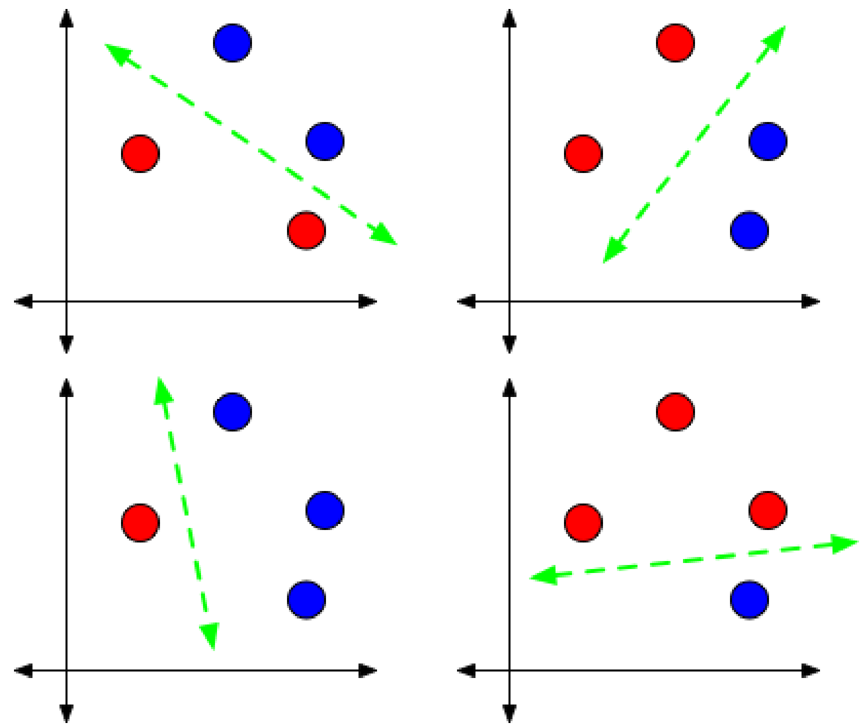
- **Instance space**:
  - Domain (input) of problem
  - Notation: X
  - Can be boolean, integral, real-valued or categorical
    - CLT usually considers boolean
      - Simpler
      - Can represent any other type
  - e.g. boolean strings of length-n

    $$X \in \{0,1\}^n$$

    - size of instance space $|X| = 2^n$

- **Instance**:
  - a.k.a **example, sample**
  - A n-tuple permutation of the instance space

Terminology



- **Concept**:
  - A deterministic partitioning (labeling) of the instance space
  - A function of the instance space
  - Is unique
  - Notation: *c(x)*
- **Concept class**:
  - An abstraction of related concepts
  - e.g. monotone DNFs, CNFs, half-spaces in $\mathbb{R}^d$
  - Notation: *C*
- Note difference: similar to object vs. class in OOP

Terminology

- **Hypothesis**:
  - The learning algorithm's approximation (estimate) to the target concept *c*
  - Notation: by *h(x)*

- **Hypothesis space**:
  - Space of all possible hypotheses for concept class *C*
  - Notation: *H*

- Similar to concept/concept class relationship

Terminology

- **Example Oracle**:
  - Like a teacher, provides examples of a given concept
  - Notation:
    - Malicious: *Ex(c)*
    - Agnostic: *Ex(c,D)*

General Learning Paradigm

- Given:
  - Instance space X
  - Concept class C over X
  - Learning algorithm (learner) A for C
  - (supervised) Oracle for concept c in C, $Ex(c)$ (or $Ex(c,D)$)

- After m examples, learner has hypothesis h in H, such that h is either consistent with c, or within tolerance

Online Learning

- Often referred to as the *Online Mistake Bound* (OLMB) Model
- Sequential, continual learning
- Learning proceeds as a sequence of trials
- "Malicious" Oracle *Ex(c)*: provides sequence of labeled examples (x, $c(x)$) in presumably worst possible order for learning concept c

Online Learning

- Given:
  - Instance space X, concept class C
  - Learning algorithm A for concept class C
  - "Malicious" Oracle $Ex(c)$ for target concept c

- Trial:
  1. Learner gets labeled example (x, $c(x)$) from Oracle
  2. Learner outputs $h(x)$ using current hypothesis
  3. If $h(x) \neq c(x)$, learner incurs a mistake
  4. Learner updates hypothesis given outcome

Online Learning

- **Mistake Bound**:
  - Algorithm *A* has mistake bound *M* if, for any target concept *c* in *C* and any sequence of examples from *Ex(c)*, *A* makes at most *M* mistakes
  - After *M* mistakes, the learner will have hypothesis *h* that is consistent with *c*
  - In other words, mistake bound is a measure of algorithm convergence

Online Learning

- Theorem:
  - For any finite hypothesis space H for concept class C, there exists an algorithm with mistake bound at most $\log|H|$

- Halving Algorithm:
  1. Initialize "working" hypothesis space H' to H
  2. At each trial:
     1. Learner gets labeled example $(x, c(x))$ from Oracle
     2. Learner makes prediction $h(x)$ based on majority vote of all hypotheses in H'
     3. If $h(x) = c(x)$, continue
     4. Else, eliminate all hypotheses from H' that predicted consistent with $h(x)$

Online Learning

- Halving algorithm has good mistake bound but terrible performance
  - e.g. H is all monotone disjunctions of length-n

$$x_1 \vee x_2 \vee \cdots \vee x_n$$
$$|H| = 2^n$$

  - Mistake bound $M \leq n$
  - However, time complexity of the algorithm is $O(2^n)$

- Fact:
  - If a concept class is efficiently learnable in the online mistake bound model, it is efficiently learnable in other models

PAC Learning

- *Probably Approximately Correct* (PAC) Model
- Batch learning: learner trains on a random set of i.i.d. examples *S* drawn from distribution *D*
- Agnostic Oracle *E(c,D):* provides random draws of labeled examples (*x, c(x)*) from distribution *D*
- Learning is approximate: we allow hypothesis to have error ϵ, for $0 \leq \epsilon \leq 0.5$
- We allow learner to fail with probability δ, for $0 \leq \delta \leq 0.5$

PAC Learning

- Given:
  - Distribution D over instance space X
  - Learning algorithm A for concept class C
  - Parameters ϵ and δ
  - Agnostic (Randomized) Oracle $Ex(c,D)$

- Goal:
  - if A is given m examples from $Ex(c,D)$, then, with probability $\geq 1-\delta$, A outputs hypothesis h with error $P_D[h(x) \neq c(x)] \leq \epsilon$

PAC Learning

- Sample complexity:
  - Number of draws from $Ex(c,D)$ necessary to learn h within bounds of given parameters, ϵ and δ
  - Notation: m

  - We want a bound on m, given ϵ and δ

  - Corollary: given m and δ, we can bound ϵ

PAC Learning Bounds

Finite Hypothesis Spaces

- Consistent Hypothesis Finder (CHF):
  - For any sequence *S* of *m* examples labeled according to *c* in *C*, finds a consistent hypothesis *h* from finite hypothesis space *H*
- Theorem:
  - Given a CHF for *C*, can PAC learn any *c* in *C* with sample complexity

$$m = \frac{1}{\epsilon}\left(\ln|H| + \ln\frac{1}{\delta}\right)$$

PAC Learning Bounds
Finite Hypothesis Spaces

- Corollary:
  - If any hypothesis *h*, from a finite hypothesis space *H*, is consistent with *m* examples drawn from distribution *D*, its error w.r.t. *D* can be bound by

$$\Pr_D[h(x) \neq c(x)] \leq \frac{1}{m}\left(\ln|H| + \ln\frac{1}{\delta}\right)$$

PAC Learning Bounds
Finite Hypothesis Spaces

- Imperfect Hypotheses
  - Suppose *h* is not perfect w.r.t. *S*, but has training error ϵ_T
  - We have the following error bound

$$\Pr_D[h(x) \neq c(x)] \leq \epsilon_T + \sqrt{\frac{\ln|H| + \ln\frac{1}{\delta}}{2m}}$$

  - Follows intuition: true error will be greater than the training error

Infinite Hypothesis Spaces

- But what about infinite hypothesis spaces?
  - e.g. half-spaces in $\mathbb{R}^d$
- How do we quantify $|H|$?
- If we can't count the hypotheses directly, perhaps we can count the number of examples that can be discriminated

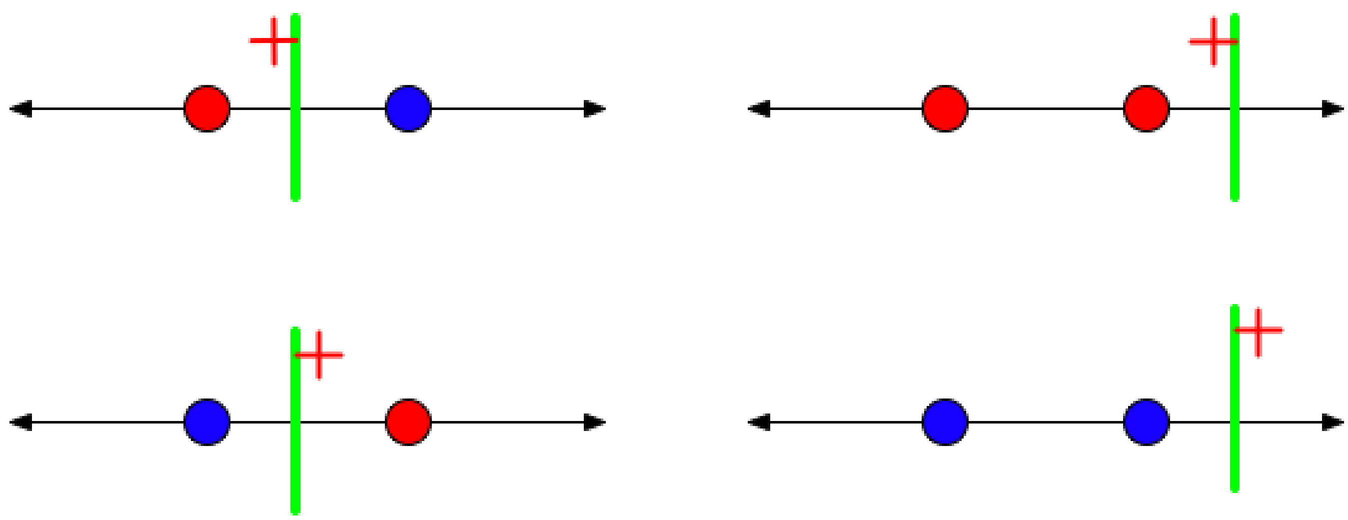
Shattering

- Definition: Given a set *S* of *n* points from *X*, if for any labeling according to *C* there exists some *h* in *H* that induces this labeling, then *H* shatters *S*

- Procedure:
  - Given: *X, C, H, S, n*
  1. Consider all possible configuration of labels over *S*
     (for simplicity, just we'll use +/– labels)
  2. For each configuration, verify that there exists a classifier *h* in *H* that induces it

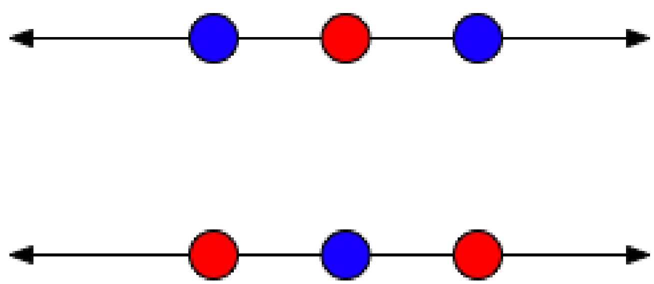
Shattering

- Example 1:
  - X = 1D number line
  - C = +/– labels
  - H = point (separator)
  - $n = 2$



Shattering

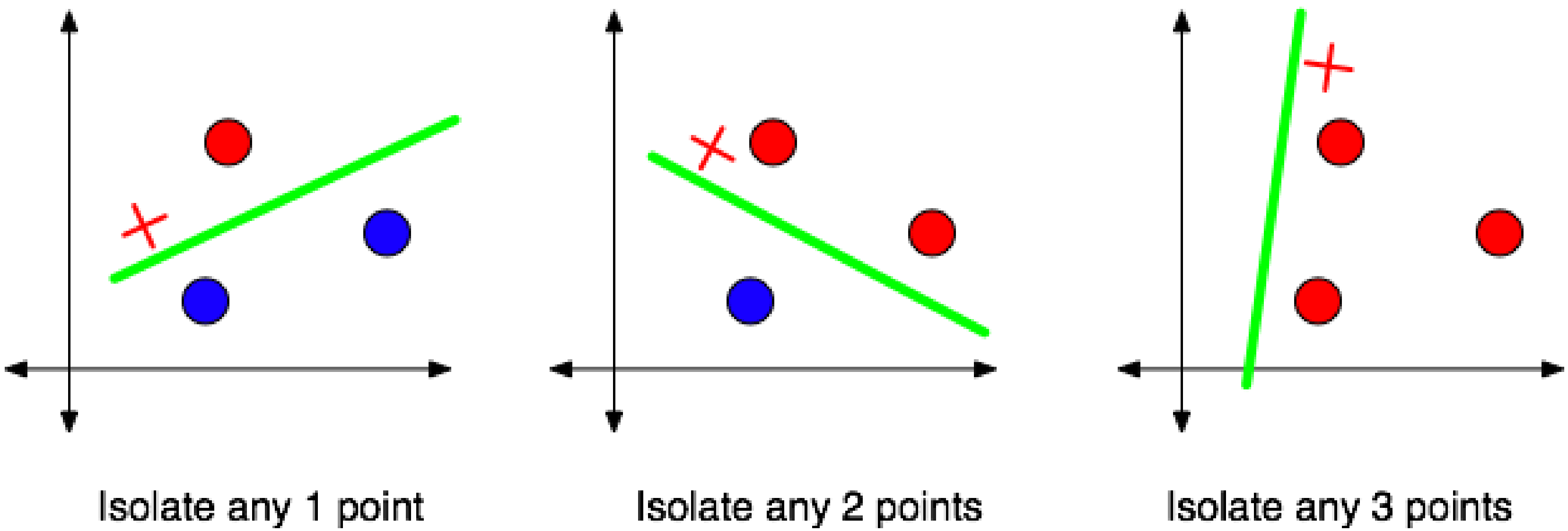
- How about 3 points?



Not
shatterable!

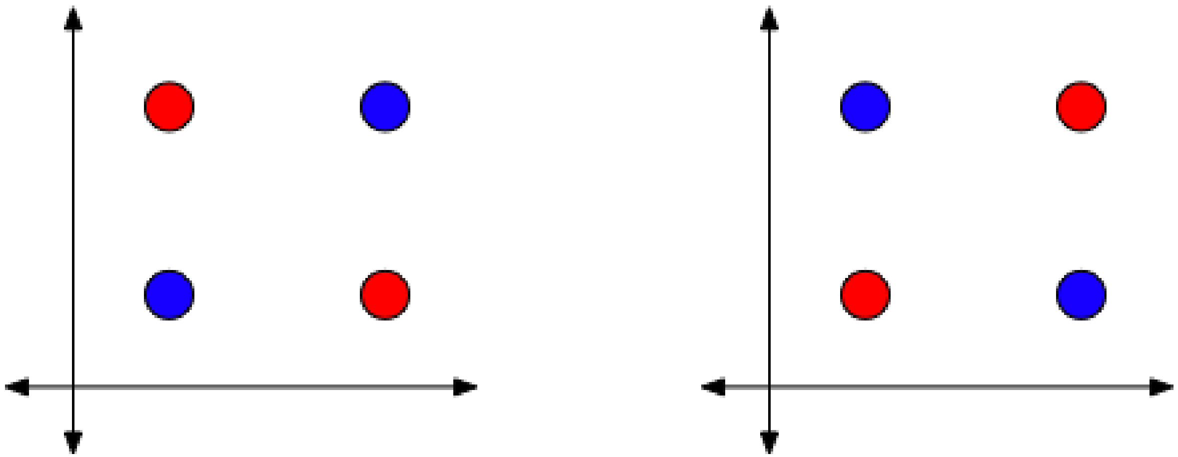
Shattering

- Example 2:
  - X = 2D plane
  - C = +/− labels
  - H = line
  - n = 3



Shattering

- How about 4 points?



Not
shatterable!

VC Dimension

- Vapnik-Chervonenkis Dimension
- Determines capacity of a (possibly infinite) hypothesis space
- Definition: the largest number of points that can be shattered by hypothesis space *H*
- Notation: $\mathrm{VCDim}(H)$
  - Alt $\mathrm{VCDim}(C)$
- Examples:
  - $C = +/-$ labels of points in 1-D $\mathrm{VCDim}(H) = 2$
  - $C = +/-$ labels of points in 2-D $\mathrm{VCDim}(H) = 3$

  - C = +/- labels of points in *n*-D $\mathrm{VCDim}(H) = n + 1$

VC Dimension

- Typical proof:
  - Given: concept class C, hypothesis space H
  1. Lower bound: show that there exists some set of at least n points that can be shattered by H
  2. Upper bound: show that there does not exist any set of $n+1$ points that can be shattered by H

  - Proof is usually analytic and/or geometric

VC Dimension

- VC Dimension gives a "reasonable" estimate of the size (capacity, expressivity) of infinite hypothesis spaces
- If we can induce $2^{\mathrm{VCDim}(H)}$ labelings, then there are at least $2^{\mathrm{VCDim}(H)}$ possible hypotheses

VC Dimension
Simple PAC Bounds for Binary Labels

- Note: $$\ln 2^{\mathrm{VCDim}(H)} = O(\mathrm{VCDim}(H))$$
- Sample complexity:

$$m = \frac{1}{\epsilon}\left(O(\mathrm{VCDim}(H)) + \ln\frac{1}{\delta}\right)$$

- Error bound:

$$\Pr_D[h(x) \neq c(x)] \leq \epsilon_T + \sqrt{\frac{O(\mathrm{VCDim}(H) + \log\frac{1}{\delta}}{2m}}$$

VC Dimension
General PAC Bounds

- Sample complexity:

$$m = O\left(\frac{d}{\epsilon}\log\frac{1}{\epsilon} + \frac{1}{\epsilon}\log\frac{1}{\delta}\right) \qquad d = VCDim(H)$$

- Error bound:

$$\Pr_D[h(x) \neq c(x)] \leq \epsilon_T + \sqrt{\frac{d\left(\log\frac{2m}{d} + 1\right) + \log\frac{4}{\delta}}{m}}$$

Proof in Kearns & Vazirani, ch. 3

VC Dimension

- Warnings:
  - VC dimension can be *infinite*
    - If VC dimension is infinite, then no finite set of examples is enough to PAC learn a concept class
  - VC dimension is *distribution free*: it makes no assumptions about the underlying distribution of the data
    - Bounds can be overly pessimistic (but still useful)

Interpretation

- Expressivity vs. Generality:
  - A more expressive model can capture more detail, but it may also overfit the training data
  - A more general model is less likely to overfit, but it can't capture as much detail

- From VC Dimension perspective:
  - Increasing VCDim(H) may decrease empirical error but increase generalization error
  - Decreasing VCDim(H) may decrease generalization error but increase empirical error

- Empirical risk vs. true risk minimization

Risk Minimization

- Risk = Error
  - Expectation of a loss function $L(y,x,\theta)$

$$R(\theta) = \mathbb{E}[L(y_i, x_i, \theta)] = \int_X \int_Y \Pr[x,y] L(y,x,\theta)\,dx\,dy$$

- But we don't know *P(X,Y)*!
  - (Unless we have infinite data)

- Best we can do is approximate true risk

$\Rightarrow$ Empirical risk

Empirical Risk Minimization

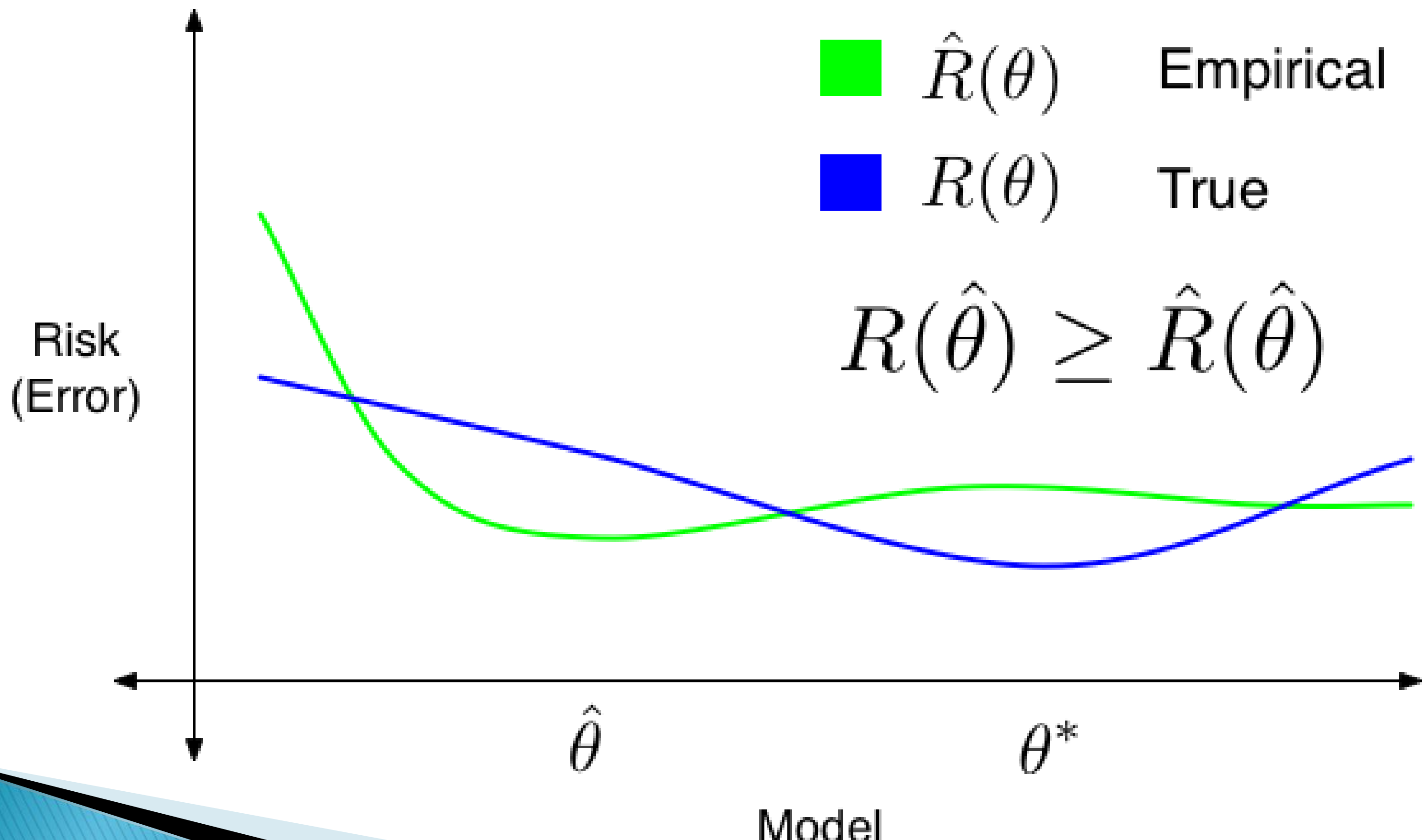
- Most learning algorithms minimize empirical risk:

$$\hat{R}(\theta) = \frac{1}{m} \sum_{i=1}^{m} L(y_i, x_i, \theta))$$

  - Quadratic: $L(y, x, \theta) = \frac{1}{2}(y - f(x;\theta))^2$ Regression
  - Binary: $L(y, x, \theta) = \mathrm{sign}(-y f(x;\theta))$ Perceptron

Empirical Risk Minimization

- Problem: empirical risk is not true risk



Structural Risk Minimization

- We want a guaranteed risk $J(\hat{\theta})$
- Solution: add confidence bound to empirical risk $J(\theta) = \hat{R}(\theta) + C(\theta)$
- If $\forall \theta, R(\theta) \leq J(\theta)$, then we have true risk bound
- If we select the $\hat{\theta} = \mathrm{argmin}_\theta J(\theta)$, then we can guarantee that the true error $R(\hat{\theta}) \leq J(\hat{\theta})$

Structural Risk Minimization

- Recall the PAC error bound

$$\Pr_D[h(x) \neq c(x)] \leq \epsilon_T + \sqrt{\frac{d\left(\log \frac{2m}{d} + 1\right) + \log \frac{4}{\delta}}{m}}$$

- This gives us

True risk

$$R(\theta) = \Pr_D[h(x) \neq c(x)]$$

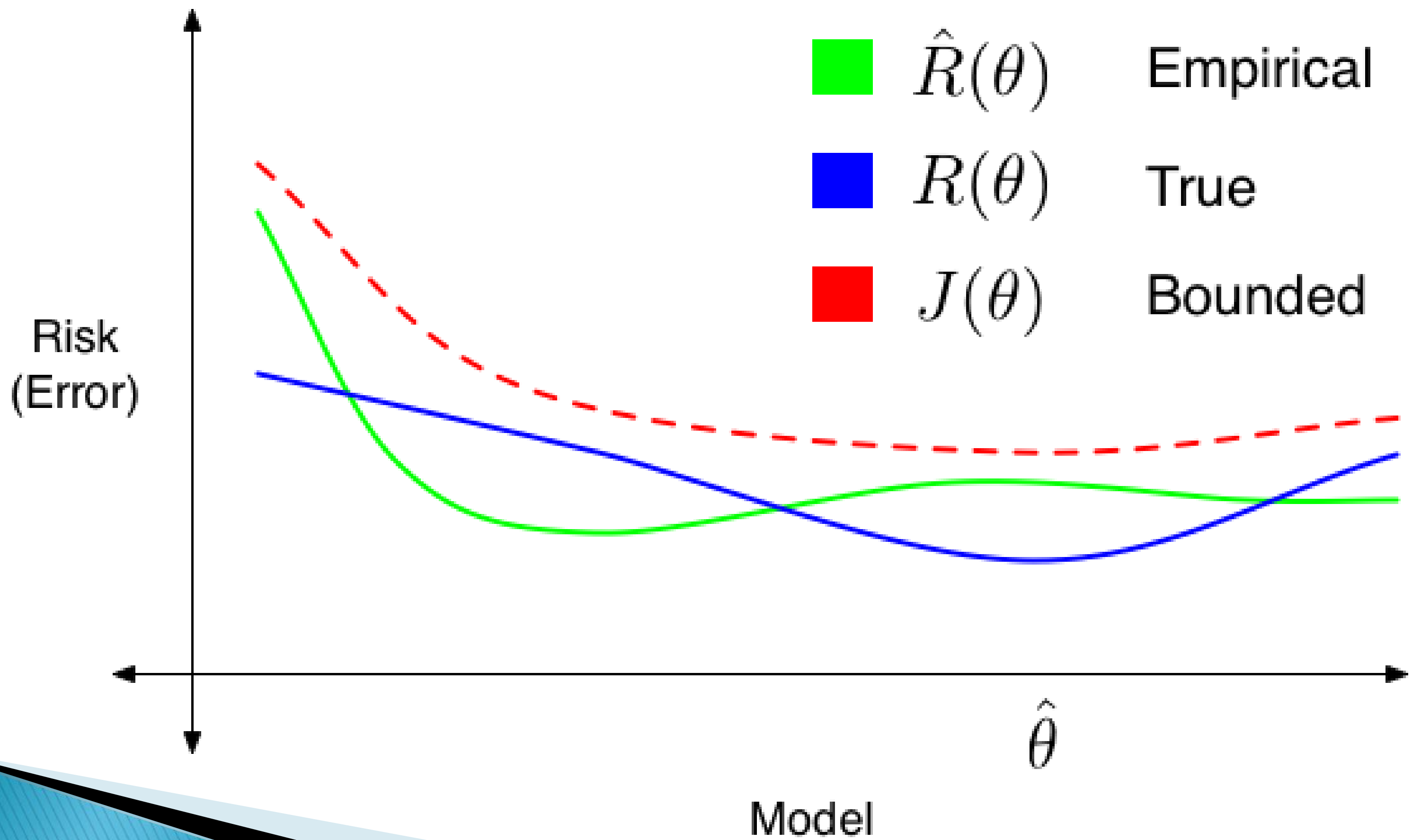
Empirical risk

$$\hat{R}(\theta) = \epsilon_T$$

Model complexity $$C(\theta) = \sqrt{\frac{d\left(\log \frac{2m}{d} + 1\right) + \log \frac{4}{\delta}}{m}}$$

Structural Risk Minimization

- Bounding true risk



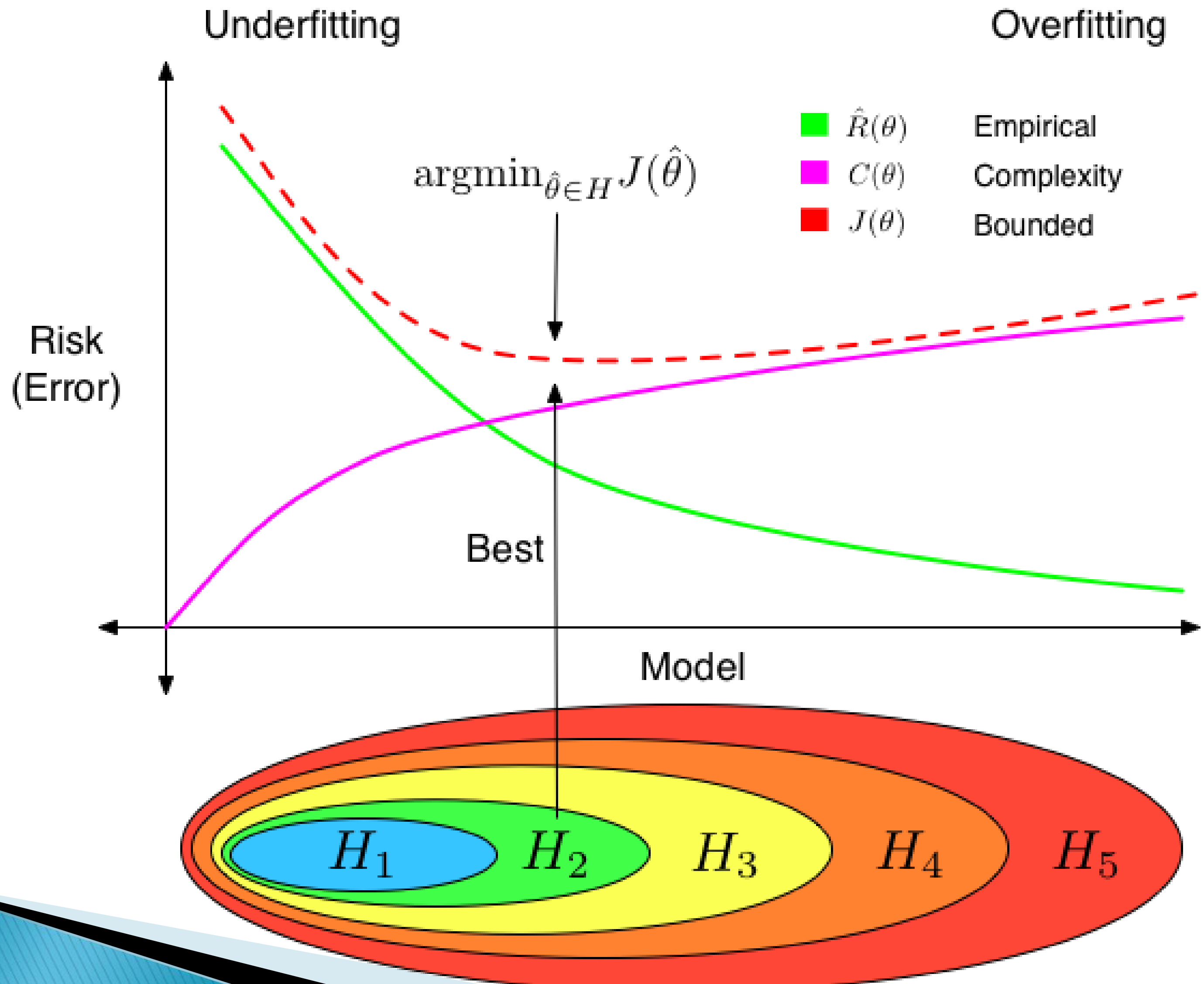
Structural Risk Minimization

- Goal: choose the hypothesis that minimizes error bound $J(\theta)$

- Procedure:
  1. Select set of "reasonable" hypothesis spaces using *a priori* knowledge of the problem
  2. For each hypothesis space:
     1. Train hypothesis that minimizes empirical error
     2. Compute error bound $J(\hat{\theta}) = \hat{R}(\hat{\theta}) + C(\hat{\theta})$
  3. Select hypothesis that minimizes error bound

$$\theta^* = \mathrm{argmin}_{\hat{\theta}} J(\hat{\theta})$$

Structural Risk Minimization



Philosophical Interpretation:
Occam's Razor

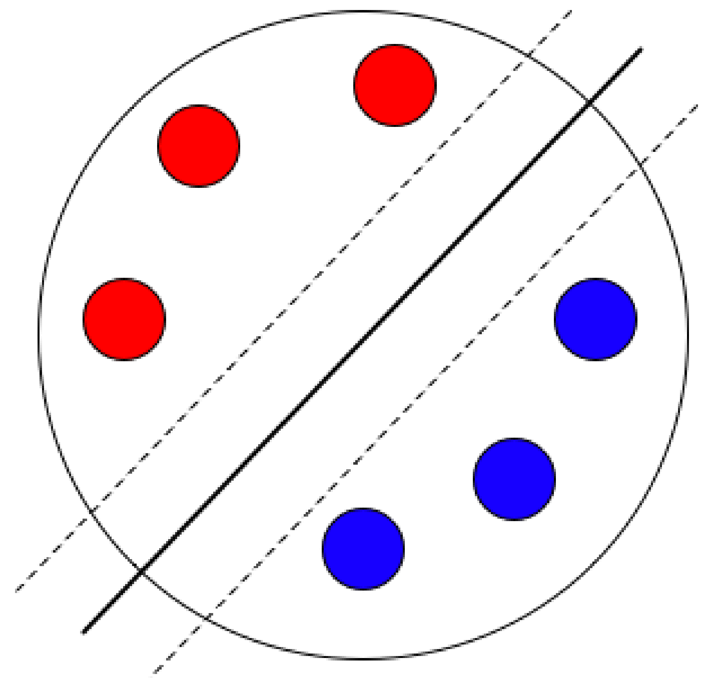
- "Entities must not be multiplied beyond necessity"
  – William of Ockham (14th cent. AD)

- Given many potential hypotheses, choose the one that makes the fewest assumptions

http://en.wikipedia.org/wiki/File:William_of_Ockham.png

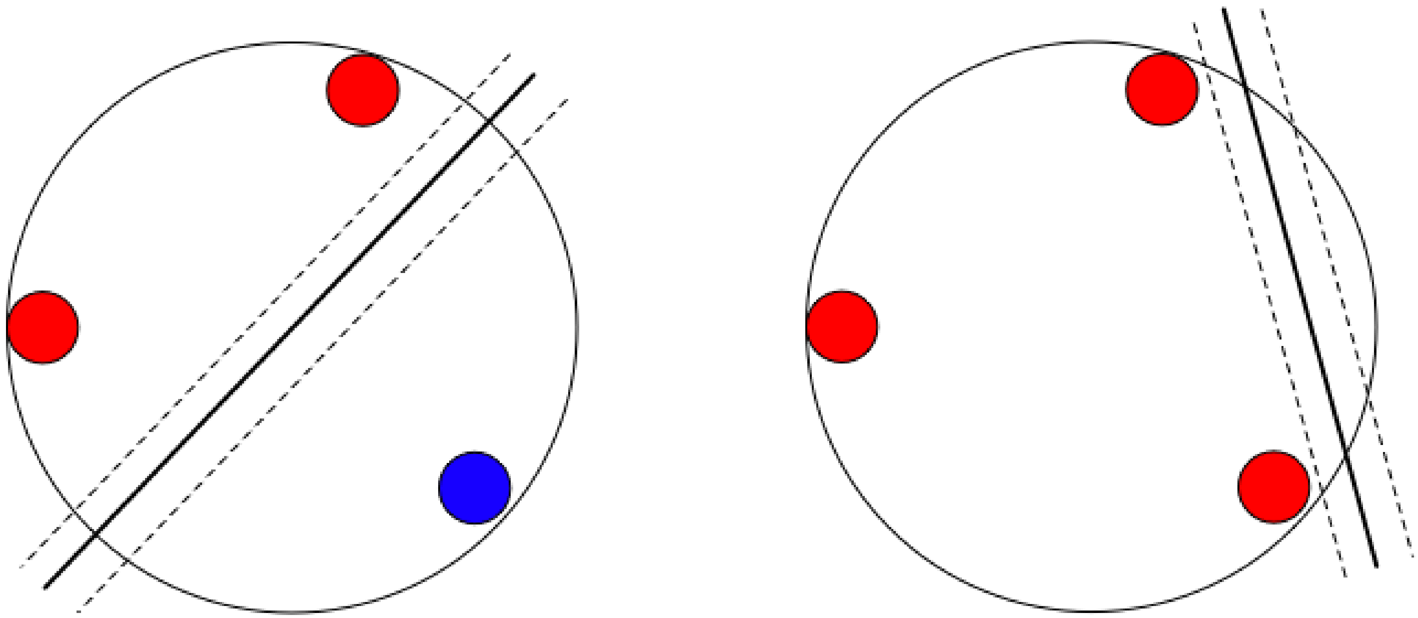
VC Dimension & Large Margins

- Arbitrary linear classifiers are too flexible
- Can reduce VC dimension if we restrict them
- Reasonable assumption: constrain data to living inside a circle/sphere
  - dimension *d*
  - diameter *D*
- Apply linear classifier with margin *M*



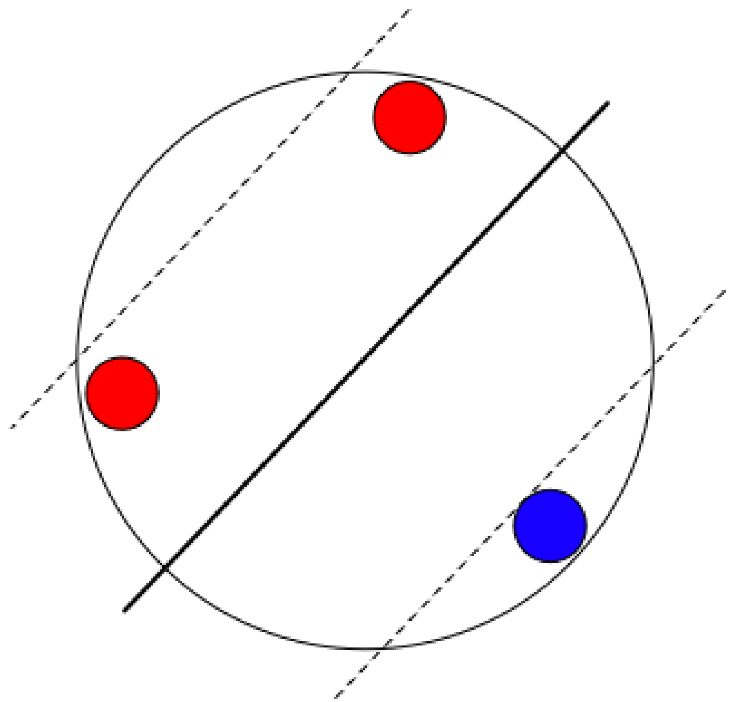
VC Dimension & Large Margins

- If M is small, can shatter 3 points

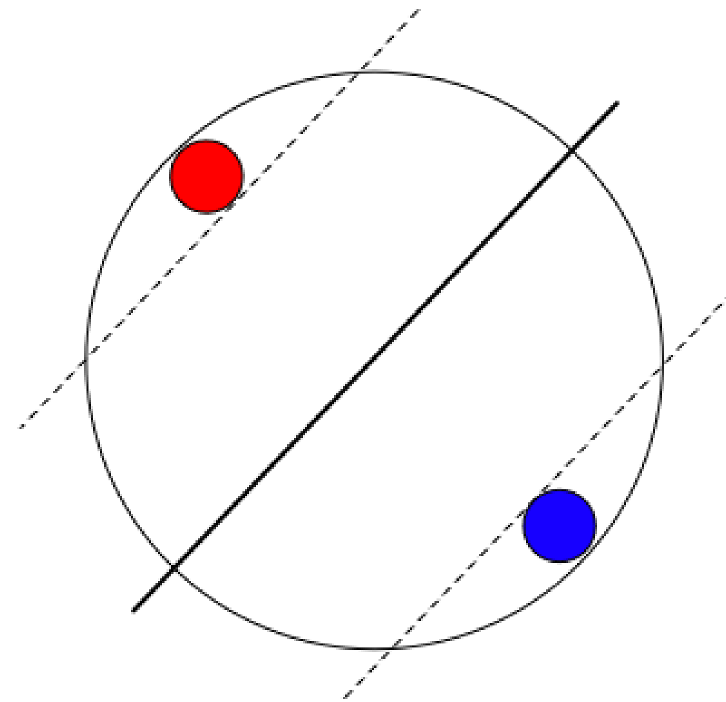


VC Dimension & Large Margins

- If M is large enough, can only shatter 2 points



Can’t shatter



Can shatter

VC Dimension & Large Margins

- For hyperplanes, as M grows relative to D, VC dimension goes down

- General formula $\mathrm{VCDim}(H) = \min\left(\left\lceil \frac{D^2}{M^2} \right\rceil, d\right) + 1$
  - Arbitrary data: $\mathrm{VCDim}(H) \to d + 1$
  - Inside sphere: $\mathrm{VCDim}(H) \to \lceil D^2/M^2 \rceil + 1$
    - Typical data lives inside sphere

- $\therefore$ Larger margins decrease VC dimension $\Rightarrow$ Decrease generalization error!

Relation to SVMs

- SVMs maximize margin,
  $\Rightarrow$ minimize generalization error

- SVMs naturally perform SRM

- CLT explains why SVMs are so good

Research Topics in CLT

- Learning models
  - Online
  - PAC
  - Active
  - Statistical Query
  - Boolean Hypercube
  - Evolvability
- Cryptography/hardness results
- Boosting/ensemble methods