Foundations of Machine Learning

Module 7: Computational Learning Theory

Part A: Finite Hypothesis Space

Sudeshna Sarkar

IIT Kharagpur

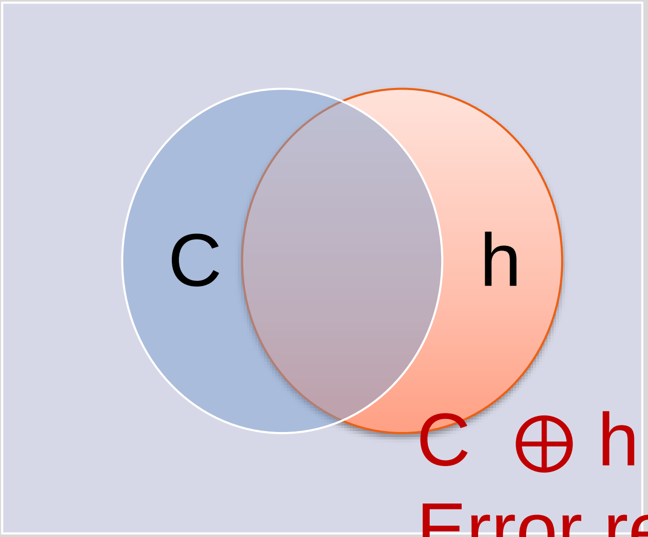
Goal of Learning Theory

- To understand
  - What kinds of tasks are learnable?
  - What kind of data is required for learnability?
  - What are the (space, time) requirements of the learning algorithm.?
- To develop and analyze models
  - Develop algorithms that provably meet desired criteria
  - Prove guarantees for successful algorithms

Goal of Learning Theory

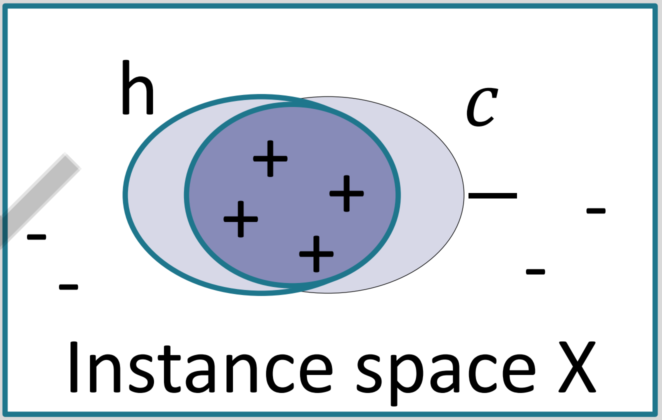
- Two core aspects of ML
  - Algorithm Design. How to optimize?
  - Confidence for rule effectiveness on future data.
- We need particular settings (models)
  - Probably Approximately Correct (PAC)

$$\Pr(P(c \oplus h) \leq \epsilon) \geq 1 - \delta$$



C $\oplus$ h =
Error region

Prototypical Concept Learning Task



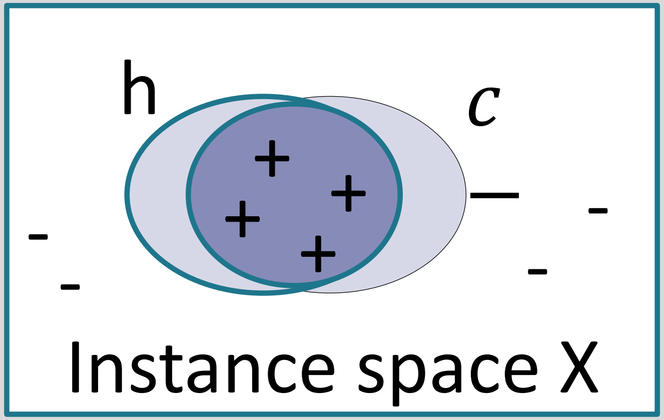
- Given
  - Instances X (e.g., $X = R^d$ or $X = \{0,1\}^d$
  - Distribution $\mathcal{D}$ over X
  - Target function c
  - Hypothesis Space $\mathcal{H}$
  - Training Examples S = $\{(x_i, c(x_i))\}$ x_i i.i.d. from $\mathcal{D}$
- Determine
  - A hypothesis $\mathrm{h} \in \mathcal{H}$ s.t. $h(x) = c(x)$ for all x in S?
  - A hypothesis $\mathrm{h} \in \mathcal{H}$ s.t. $h(x) = c(x)$ for all x in X?
- An algorithm does optimization over S, find hypothesis h.
- Goal: Find *h* which has small error over $\mathcal{D}$

Computational Learning Theory

- Can we be certain about how the learning algorithm generalizes?
- We would have to see all the examples.

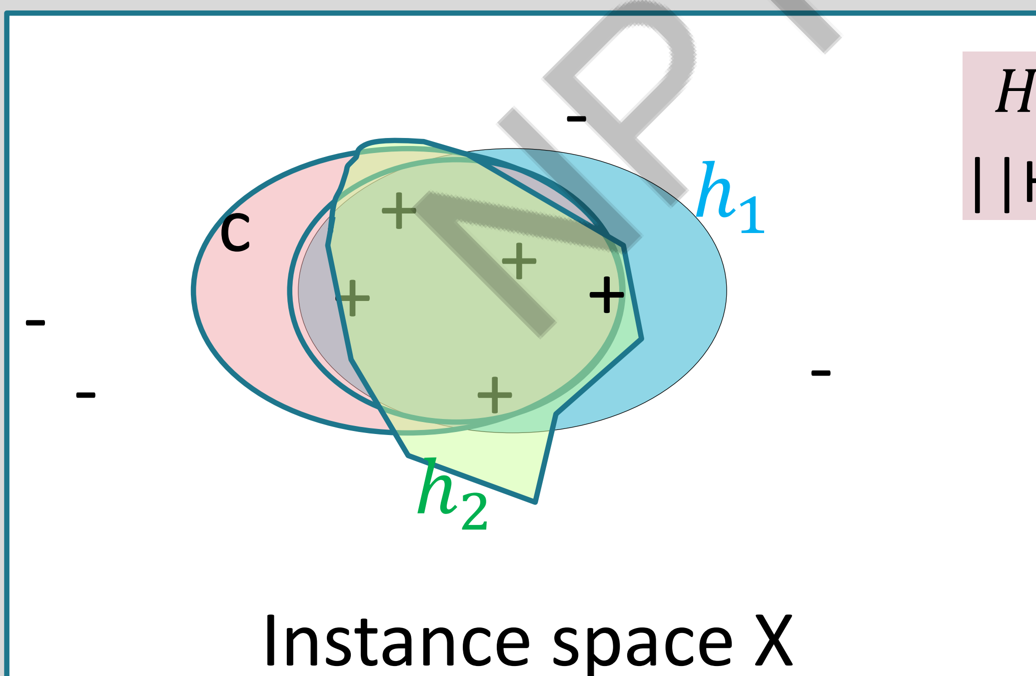
- Inductive inference – generalizing beyond the training data is impossible unless we add more assumptions (e.g., priors over H)

  We need a bias!



Function Approximation

- How many labeled examples in order to determine which of the 2^{2^N} hypothesis is the correct one?
- All 2^N instances in X must be labeled!
- Inductive inference: generalizing beyond the training data is impossible unless we add more assumptions (e.g., bias)



$H = \{h: X \rightarrow Y\}$

$||H|=2^{|X|} = 2^{2^N}$

Error of a hypothesis

The ***true error*** of hypothesis *h,* with respect to the target concept *c* and observation distribution $\mathcal{D}$ is the probability that *h* will misclassify an instance drawn according to $\mathcal{D}$

$$error_{\mathcal{D}}(h) Pr_{x \sim \mathcal{D}}[c(x) \neq h(x)]$$

In a perfect world, we'd like the true error to be 0.

Bias: Fix hypothesis space H

c may not be in H => Find h close to c

A hypothesis *h* is approximately correct if

$$error_{\mathcal{D}}(h) \leq \varepsilon$$

PAC model

- Goal: h has small error over D.
- True error: $error_D(h) = \Pr_{x \sim D}\left(h(x) \neq c^*(x)\right)$
- How often $h(x) \neq c^*(x)$ over future instances drawn at random from D
- But, can only measure:

  Training error: $error_S(h) = \frac{1}{m}\sum_i I(h(x_i) \neq c^*(x))$

  How often $h(x) \neq c^*(x)$ over training Instances

- Sample Complexity: bound $error_D(h)$ in terms of $error_S(h)$

Probably Approximately Correct Learning

- PAC Learning concerns efficient learning
- We would like to prove that
  - With high probability an (efficient) learning algorithm will find a hypothesis that is approximately identical to the hidden target concept.
- We specify two parameters, ε and δ and require that with probability at least $(1-\delta)$ a system learn a concept with error at most ε.

Sample Complexity for Supervised Learning

Theorem

$$m \geq \frac{1}{\epsilon}\left[In(|H|) + In\left(\frac{1}{\delta}\right)\right]$$

labeled examples are sufficient so that with prob. $1-\delta$, all $h \in H$ with $error_D(h) \geq \epsilon$ have $error_S(h) > 0$.

- inversely linear in ϵ
- logarithmic in |H|
- ϵ error parameter: D might place low weight on certain parts of the space
- δ confidence parameter: there is a small chance the examples we get are not representative of the distribution

Sample Complexity for Supervised Learning

Theorem: $m \geq \frac{1}{\in}\left[In(|H|) + In\left(\frac{1}{\delta}\right)\right]$ labeled examples are sufficient so that with prob. $1 - \delta$, all $h \in H$ with $error_D(h) \geq \in$ have $error_S(h) > 0$.

Proof: Assume k bad hypotheses $H_{\text{bad}}=\{h_1, h_2, \dots, h_k\}$ with

$$err_D(h_i) \geq \in$$

- Fix h_i. Prob. h_i consistent with first training example is $\leq 1 - \in$. Prob. h_i consistent with first m training examples is $\leq (1 - \in)^m$.
- Prob. that at least one h_i consistent with first m training examples is
$$\leq k(1 - \in)^m \leq |H|(1 - \in)^m.$$
- Calculate value of m so that $|H|(1 - \in)^m \leq \delta$
- Use the fact that $1 - x \leq e^{-x}$, sufficient to set $|H|e^{-\in m} \leq \delta$

Sample Complexity: Finite Hypothesis Spaces Realizable Case

PAC: How many examples suffice to guarantee small error whp.

Theorem

$$m \geq \frac{1}{\epsilon}\left[In(|H|) + In\left(\frac{1}{\delta}\right)\right]$$

labeled examples are sufficient so that with prob. $1-\delta$, all $h \in H$ with $err_D(h) \geq \epsilon$ have $err_S(h) > 0$.

Statistical Learning Way:

With probability at least $1-\delta$, all $h \in H$ s.t. $err_S(h) = 0$ we have

$$err_D(h) \leq \frac{1}{m}\left[In(|H|) + In\left(\frac{1}{\delta}\right)\right]$$

$$P(\text{consist}(H_{bad}, D)) \leq |H| e^{-\varepsilon m} \leq \delta$$

$$e^{-\varepsilon m} \leq \frac{\delta}{|H|}$$

$$-\varepsilon m \leq \ln(\frac{\delta}{|H|})$$

$$m \geq \left(-\ln \frac{\delta}{|H|}\right) / \varepsilon \quad (\text{flip inequality})$$

$$m \geq \left(\ln \frac{|H|}{\delta}\right) / \varepsilon$$

$$m \geq \left(\ln \frac{1}{\delta} + \ln|H|\right) / \varepsilon$$

Sample complexity: inconsistent finite $|\mathcal{H}|$

- For a single hypothesis to have misleading training error

$$\Pr[error_{\mathcal{D}}(f) \le \varepsilon + error_{D}(f)] \le e^{-2m\varepsilon^2}$$

- We want to ensure that the best hypothesis has error bounded in this way
  - So consider that any one of them could have a large error

$$\Pr[(\exists f \in \mathcal{H}) error_{\mathcal{D}}(f) \le \varepsilon + error_{D}(f)] \le |\mathcal{H}| e^{-2m\varepsilon^2}$$

- From this we can derive the bound for the number of samples needed.

$$m \ge \frac{1}{2\varepsilon^2}(\ln|\mathcal{H}| + \ln(\frac{1}{\delta}))$$

Sample Complexity: Finite Hypothesis Spaces

Consistent Case

Theorem

$$m \geq \frac{1}{\in}\left[In(|H|) + In\left(\frac{1}{\delta}\right)\right]$$

labeled examples are sufficient so that with prob. $1-\delta$, all $h \in H$ with $err_D(h) \geq \in$ have $err_S(h) > 0$.

Inconsistent Case

What if there is no perfect h?

Theorem: After m examples, with probability $\geq 1-\delta$, all $h \in H$ have $|err_D(h) - err_S(h)| < \in$, for

$$m \geq \frac{2}{2 \in^2}\left[In(|H|) + In\left(\frac{2}{\delta}\right)\right]$$

Sample complexity: example

- $\mathcal{C}$: Conjunction of n Boolean literals. Is $\mathcal{C}$ PAC-learnable?

$$|\mathcal{H}| = 3^n$$

$$m \geq \frac{1}{\varepsilon}(n\ \ln 3 + \ln(\frac{1}{\delta}))$$

- Concrete examples:
  - δ=ε=0.05, *n*=10 gives 280 examples
  - δ=0.01, ε=0.05, *n*=10 gives 312 examples
  - δ=ε=0.01, *n*=10 gives 1,560 examples
  - δ=ε=0.01, *n*=50 gives 5,954 examples
- Result holds for any consistent learner, such as FindS.

Sample Complexity of Learning Arbitrary Boolean Functions

- Consider any boolean function over *n* boolean features such as the hypothesis space of DNF or decision trees. There are 2^{2^n} of these, so a sufficient number of examples to learn a PAC concept is:

$$m \geq \frac{1}{\varepsilon}(\ln 2^{2^n} + \ln(\frac{1}{\delta})) = \frac{1}{\varepsilon}(2^n \ln 2 + \ln(\frac{1}{\delta}))$$

- $\delta=\varepsilon=0.05$, $n=10$ gives 14,256 examples
- $\delta=\varepsilon=0.05$, $n=20$ gives 14,536,410 examples
- $\delta=\varepsilon=0.05$, $n=50$ gives 1.561×10^{16} examples

Thank You

Concept Learning Task

"Days in which Aldo enjoys swimming"

Example	Sky	AirTemp	Humidity	Wind	Water	Forecast	EnjoySport
1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	High	Strong	Cool	Change	Yes

- Hypothesis Representation: Conjunction of constraints on the 6 instance attributes
    - "?" : any value is acceptable
    - specify a single required value for the attribute
    - "∅" : that no value is acceptable

Concept Learning

h = (?, Cold, High, ?, ?, ?)

indicates that Aldo enjoys his favorite sport on cold days with high humidity

Most general hypothesis: (?, ?, ?, ?, ?, ?)

Most specific hypothesis: (∅, ∅, ∅, ∅, ∅, ∅)

Find-S Algorithm

1. Initialize h to the most specific hypothesis in $\mathcal{H}$
2. For each positive training instance x

   For each attribute constraint a_i in h

   IF the constraint a_i in h is satisfied by x

   THEN do nothing

   ELSE replace a_i in h by next more general constraint satisfied by x
3. Output hypothesis h

Concept Learning

Example	Sky	AirTemp	Humidity	Wind	Water	Forecast	EnjoySport
1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	High	Strong	Cool	Change	Yes

Finding a Maximally Specific Hypothesis

Find-S Algorithm

$h_1 \leftarrow (\varnothing, \varnothing, \varnothing, \varnothing, \varnothing, \varnothing)$

$h_2 \leftarrow$ (Sunny, Warm, Normal, Strong, Warm, Same)

$h_3 \leftarrow$ (Sunny, Warm, ?, Strong, Warm, Same)

$h_4 \leftarrow$ (Sunny, Warm, ?, Strong, ?, ?)

Back

Thank You

Foundations of Machine Learning

Module 7: Computational Learning Theory
Part A

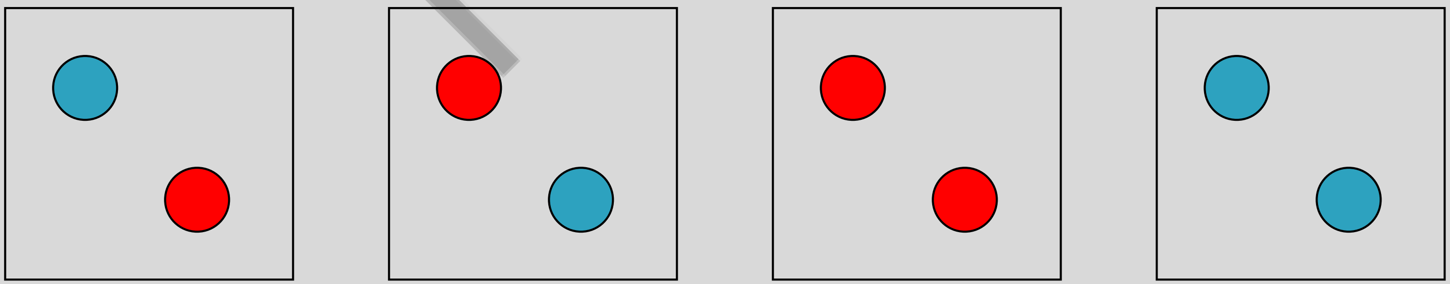
Sudeshna Sarkar
IIT Kharagpur

Sample Complexity: Infinite Hypothesis Spaces

- Need some measure of the expressiveness of infinite hypothesis spaces.
- The *Vapnik-Chervonenkis (VC) dimension* provides such a measure, denoted VC(H).
- Analagous to $\ln|H|$, there are bounds for sample complexity using VC(H).

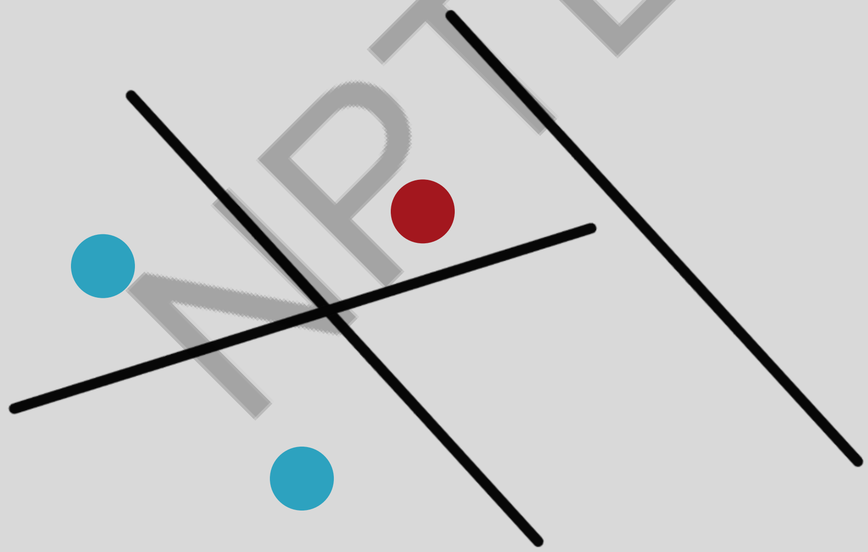
Shattering

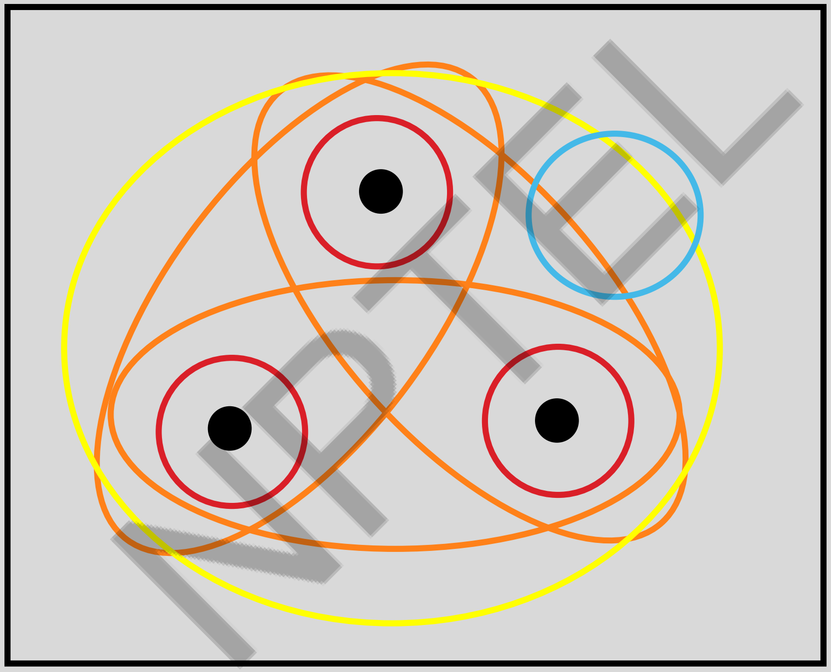
- Consider a hypothesis for the 2-class problem.
- A set of N points (instances) can be labeled as $+$ or $-$ in 2^N ways.
- If for every such labeling a function can be found in $\mathcal{H}$ consistent with this labeling, we set that the set of instances is shattered by $\mathcal{H}$.



Three points in R^2

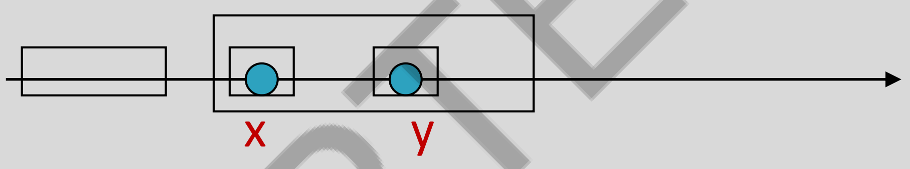
- It is enough to find one set of three points that can be shattered.
- It is not necessary to be able to shatter **every possible set of three points** in 2 dimensions





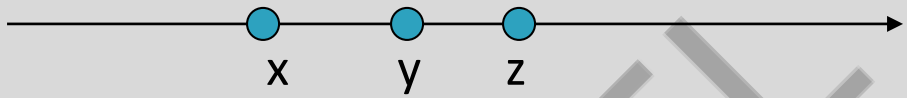
Shattering Instances

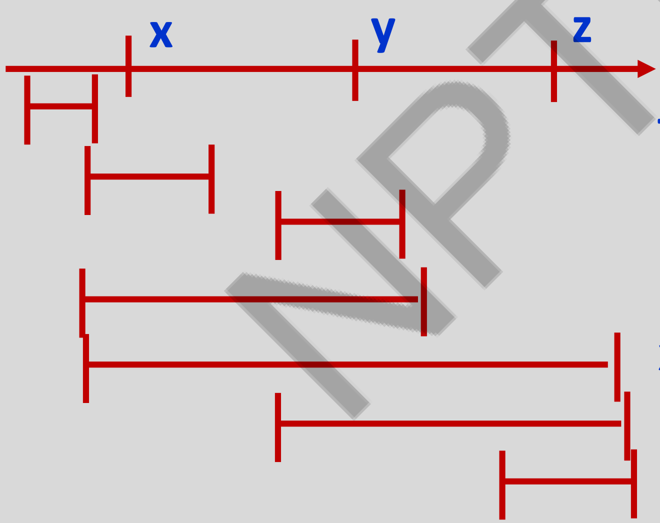
- Consider 2 instances described using a single real-valued feature being shattered by a single interval.



Shattering Instances (cont)

But 3 instances cannot be shattered by a single interval.





	+	−
	–	x,y,z
	x	y,z
	y	x,z
	x,y	z
	x,y,z	
	y,z	x
	z	x,y
Cannot do	x,z	y

VC Dimension

- The Vapnik-Chervonenkis dimension, VC(*H*). of hypothesis space *H* defined over instance space *X* is the size of the largest finite subset of *X* shattered by *H*. If arbitrarily large finite subsets of *X* can be shattered then VC(*H*) = ∞

- If there exists at least one subset of *X* of size *d* that can be shattered then VC(*H*) ≥ *d*.
- If no subset of size *d* can be shattered, then VC(*H*) < *d.*

- For a single intervals on the real line, all sets of 2 instances can be shattered, but no set of 3 instances can, so VC(*H*) = 2.

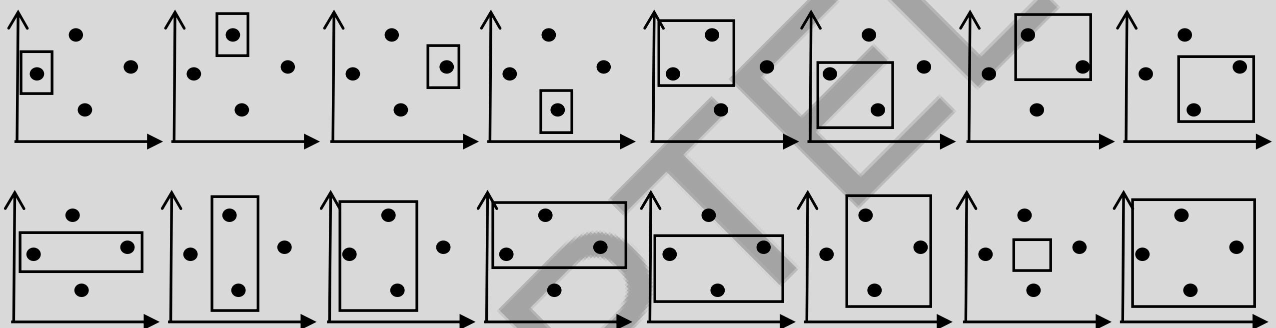
VC Dimension

- An unbiased hypothesis space shatters the entire instance space.
- The larger the subset of X that can be shattered, the more expressive (and less biased) the hypothesis space is.
- The VC dimension of the set of oriented lines in 2-d is three.

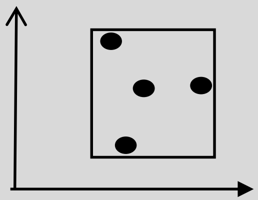
- Since there are 2^m partitions of m instances, in order for H to shatter instances: $|H| \geq 2^m$.
- Since $|H| \geq 2^m$, to shatter m instances, $VC(H) \leq \log_2|H|$

VC Dimension Example

Consider axis-parallel rectangles in the real-plane, i.e. conjunctions of intervals on two real-valued features. Some 4 instances can be shattered.

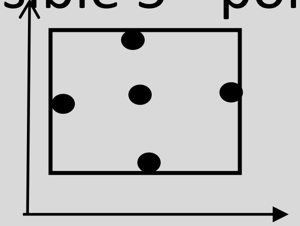


Some 4 instances cannot be shattered:



VC Dimension Example (cont)

- No five instances can be shattered since there can be at most 4 distinct extreme points (min and max on each of the 2 dimensions) and these 4 cannot be included without including any possible 5th point.



- Therefore VC(H) = 4
- Generalizes to axis-parallel hyper-rectangles (conjunctions of intervals in n dimensions): VC(H)=2n.

Upper Bound on Sample Complexity with VC

- Using VC dimension as a measure of expressiveness, the following number of examples have been shown to be sufficient for PAC Learning (Blumer *et al*., 1989).

$$\frac{1}{\varepsilon}\left(4\log_2\left(\frac{2}{\delta}\right)+8VC(H)\log_2\left(\frac{13}{\varepsilon}\right)\right)$$

- Compared to the previous result using $\ln|H|$, this bound has some extra constants and an extra $\log_2(1/\varepsilon)$ factor. Since $VC(H) \leq \log_2|H|$, this can provide a tighter upper bound on the number of examples needed for PAC learning.

Sample Complexity Lower Bound with VC

- There is also a general lower bound on the minimum number of examples necessary for PAC learning (Ehrenfeucht, *et al.*, 1989):

  Consider any concept class *C* such that $VC(H) > 2$, any learner L and any $0 < \varepsilon < {}^1/_8 , 0 < \delta < {}^1/_{100}$.
  Then there exists a distribution *D* and target concept in *C* such that if *L* observes fewer than:

$$\max\left(\frac{1}{\varepsilon}\log_2\left(\frac{1}{\delta}\right), \frac{VC(C)-1}{32\varepsilon}\right)$$

  examples, then with probability at least δ, *L* outputs a hypothesis having error greater than ε.

- Ignoring constant factors, this lower bound is the same as the upper bound except for the extra $\log_2(1/\varepsilon)$ factor in the upper bound.

Thank You

Foundations of Machine Learning

Module 8: Ensemble Learning

Part A

Sudeshna Sarkar

IIT Kharagpur

What is Ensemble Classification?

- Use multiple learning algorithms (classifiers)
- Combine the decisions
- Can be more accurate than the individual classifiers
- Generate a group of base-learners
- Different learners use different
  - Algorithms
  - Hyperparameters
  - Representations (Modalities)
  - Training sets

Why should it work?

- Works well only if the individual classifiers disagree
  - Error rate < 0.5 and errors are independent
  - Error rate is highly correlated with the correlations of the errors made by the different learners

Bias vs. Variance

- We would like low bias error and low variance error
- Ensembles using multiple trained (high variance/low bias) models can average out the variance, leaving just the bias
  - Less worry about overfit (stopping criteria, etc.) with the base models

Combining Weak Learners

- Combining weak learners
  - Assume *n* independent models, each having accuracy of 70%.
  - If all *n* give the same class output then you can be confident it is correct with probability $1-(1-.7)^n$.
  - Normally not completely independent, but unlikely that all *n* would give the same output
    - Accuracy better than the base accuracy of the models by using the majority output.
  - If *n1* models say class 1 and *n2*<*n1* models say class 2, then P(class1) = 1 – Binomial(*n*, n2, .7)

$$P(r) = \frac{n!}{r!(n-r)!} p^r (1-p)^{n-r}$$

Ensemble Creation Approaches

- Get less correlated errors between models
  - Injecting randomness
    - initial weights (eg, NN), different learning parameters, different splits (eg, DT) etc.
  - Different Training sets
    - Bagging, Boosting, different features, etc.
  - Forcing differences
    - different objective functions
  - Different machine learning model

Ensemble Combining Approaches

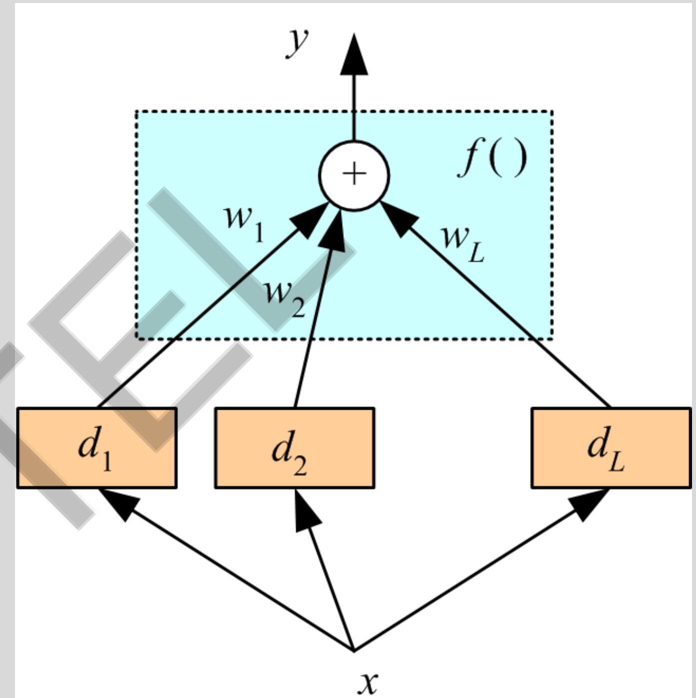
- Unweighted Voting (e.g. Bagging)
- Weighted voting – based on accuracy (e.g. Boosting), Expertise, etc.
- Stacking - Learn the combination function

Combine Learners: Voting

- Unweighted voting
- Linear combination (weighted vote)
  - weight ∝ accuracy
  - weight ∝ $^1/_{\text{variance}}$

$$y = \sum_{j=1}^{L} w_j d_j$$

$$w_j \geq 0 \text{ and } \sum_{j=1}^{L} w_j = 1$$



- Bayesian

$$P(C_i|x) = \sum_{\text{all models } \mathcal{M}_j} P(C_i|x, \mathcal{M}_j) P(\mathcal{M}_j)$$

Fixed Combination Rules

Rule	Fusion function $f(\cdot)$
Sum	$y_i = \frac{1}{L}\sum_{j=1}^{L} d_{ji}$
Weighted sum	$y_i = \sum_j w_j d_{ji}, w_j \geq 0, \sum_j w_j = 1$
Median	$y_i = \text{median}_j d_{ji}$
Minimum	$y_i = \min_j d_{ji}$
Maximum	$y_i = \max_j d_{ji}$
Product	$y_i = \prod_j d_{ji}$

	C_1	C_2	C_3
d_1	0.2	0.5	0.3
d_2	0.0	0.6	0.4
d_3	0.4	0.4	0.2
Sum	0.2	**0.5**	0.3
Median	0.2	**0.5**	0.4
Minimum	0.0	**0.4**	0.2
Maximum	0.4	**0.6**	0.4
Product	0.0	**0.12**	0.032

Bayes Optimal Classifier

- The Bayes Optimal Classifier is an ensemble of all the hypotheses in the hypothesis space.
- On average, no other ensemble can outperform it.
- The vote for each hypothesis
  - proportional to the likelihood that the training dataset would be sampled from a system if that hypothesis were true.
  - is multiplied by the prior probability of that hypothesis.

$$y = \text{argmax}_{c_j \in C} \sum_{h_i \in H} P(c_j|h_i)P(T|h_i)P(h_i)$$

$$y = \operatorname{argmax}_{c_j \in C} \sum_{h_i \in H} P(c_j|h_i)P(T|h_i)P(h_i)$$

- y is the predicted class,
- C is the set of all possible classes,
- H is the hypothesis space,
- T is the training data.

The Bayes Optimal Classifier represents a hypothesis that is not necessarily in H.

But it is the optimal hypothesis in the ensemble space.

Practicality of Bayes Optimal Classifier

- Cannot be practically implemented.
- Most hypothesis spaces are too large
- Many hypotheses output a class or a value, and not probability
- Estimating the prior probability for each hypothesizes is not always possible.

BMA

- All possible models in the model space used weighted by their probability of being the "Correct" model
- Optimal given the correct model space and priors

Why are Ensembles Successful?

- Bayesian perspective:

$$P(C_i \mid x) = \sum_{\text{all models } \mathcal{M}_j} P(C_i \mid x, \mathcal{M}_j) P(\mathcal{M}_j)$$

- If d_j are independent

$$\text{Var}(y) = \text{Var}\left(\sum_j \frac{1}{L} d_j\right) = \frac{1}{L^2} \text{Var}\left(\sum_j d_j\right) = \frac{1}{L^2} L \cdot \text{Var}(d_j) = \frac{1}{L} \text{Var}(d_j)$$

Bias does not change, variance decreases by L

- If dependent, error increase with positive correlation

$$\text{Var}(y) = \frac{1}{L^2} \text{Var}\left(\sum_j d_j\right) = \frac{1}{L^2}\left[\sum_j \text{Var}(d_j) + 2\sum_j \sum_{i<j} Cov(d_i, d_j)\right]$$

Challenge for developing Ensemble Models

- The main challenge is to obtain base models which are independent and make independent kinds of errors.
- Independence between two base classifiers can be assessed in this case by measuring the degree of overlap in training examples they misclassify
  ($|A \cap B|/|A \cup B|$)

Thank You

Foundations of Machine Learning

Module 8: Ensemble Learning
Part B: Bagging and Boosting

Sudeshna Sarkar
IIT Kharagpur

Bagging

- Bagging = “bootstrap aggregation”
  - Draw N items from X with replacement
- Desired learners with high variance (unstable)
  - Decision trees and ANNs are unstable
  - K-NN is stable
- Use bootstrapping to generate *L* training sets and train one base-learner with each (Breiman, 1996)
- Use voting

Bagging

- Sampling with replacement
- Build classifier on each bootstrap sample
- Each sample has probability $(1 – 1/n)^n$ of being selected

Boosting

- An iterative procedure. Adaptively change distribution of training data.
  - Initially, all N records are assigned equal weights
  - Weights change at the end of boosting round
- On each iteration t:
  - Weight each training example by how incorrectly it was classified
  - Learn a hypothesis: h_t
  - A strength for this hypothesis: α_t
- Final classifier:
  - A linear combination of the votes of the different classifiers weighted by their strength
- “weak” learners
  - P(correct) > 50%, but not necessarily much better

Adaboost

- Boosting can turn a weak algorithm into a strong learner.
- Input: S=$\{(x_1, y_1), \ldots, (x_m, y_m)\}$
- $D_t(i)$: weight of i th training example
- Weak learner A
- For $t = 1,2, \ldots, T$
  - Construct D_t on $\{x_1, x_2 \ldots\}$
  - Run A on D_t producing $h_t: X \rightarrow \{-1,1\}$

    ϵ_t =error of h_t over D_t

Given: $(x_1, y_1), \dots, (x_m, y_m)$ where $x_i \in X, y_i \in Y = \{-1, +1\}$

Initialize $D_1(i) = 1/m$.

For $t = 1, \dots, T$:

- Train weak learner using distribution D_t.
- Get weak classifier $h_t: X \to \mathbb{R}$.
- Choose $\alpha_t \in \mathbb{R}$.
- Update:

$$D_t + 1(i) = \frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

Where Z_t is a normalization factor

$$Z_t = \sum_{i=1}^{m} D_t(i) exp\,(-\alpha_t y_i h_t(x_i))$$

Output the final classifier:

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right).$$

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where
$x_i \in X, y_i \in Y = \{-1, +1\}$

Initialize $D_1(i) = 1/m$.

For $t = 1, \ldots, T$:

- Train weak learner using distribution D_t.
- Get weak classifier $h_t: X \to \mathbb{R}$.
- Choose $\alpha_t \in \mathbb{R}$.
- Update:

$$D_t + 1(i) = \frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

Where Z_t is a normalization factor

$$Z_t = \sum_{i=1}^{m} D_t(i) exp\,(-\alpha_t y_i h_t(x_i))$$

Output the final classifier:

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right).$$

Choose α_t to minimize training error

$$\alpha_t = \frac{1}{2} In\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$$

where

$$\epsilon_t = \sum_{i=1}^{m} D_t(i)\delta(h_t(x_i) \neq y_i)$$

Strong weak classifiers

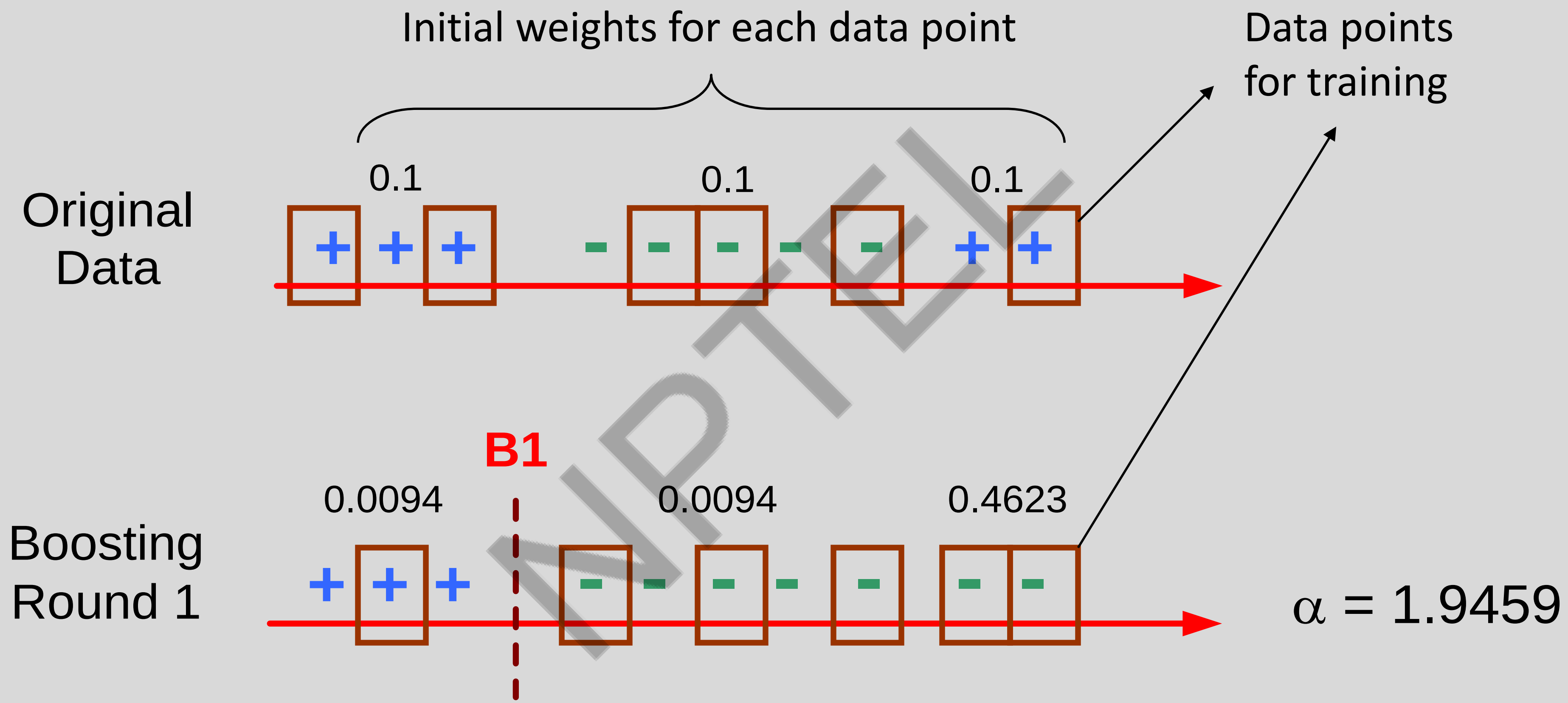
- If each classifiers is (at least slightly) better than random

$$\epsilon_t < 0.5$$

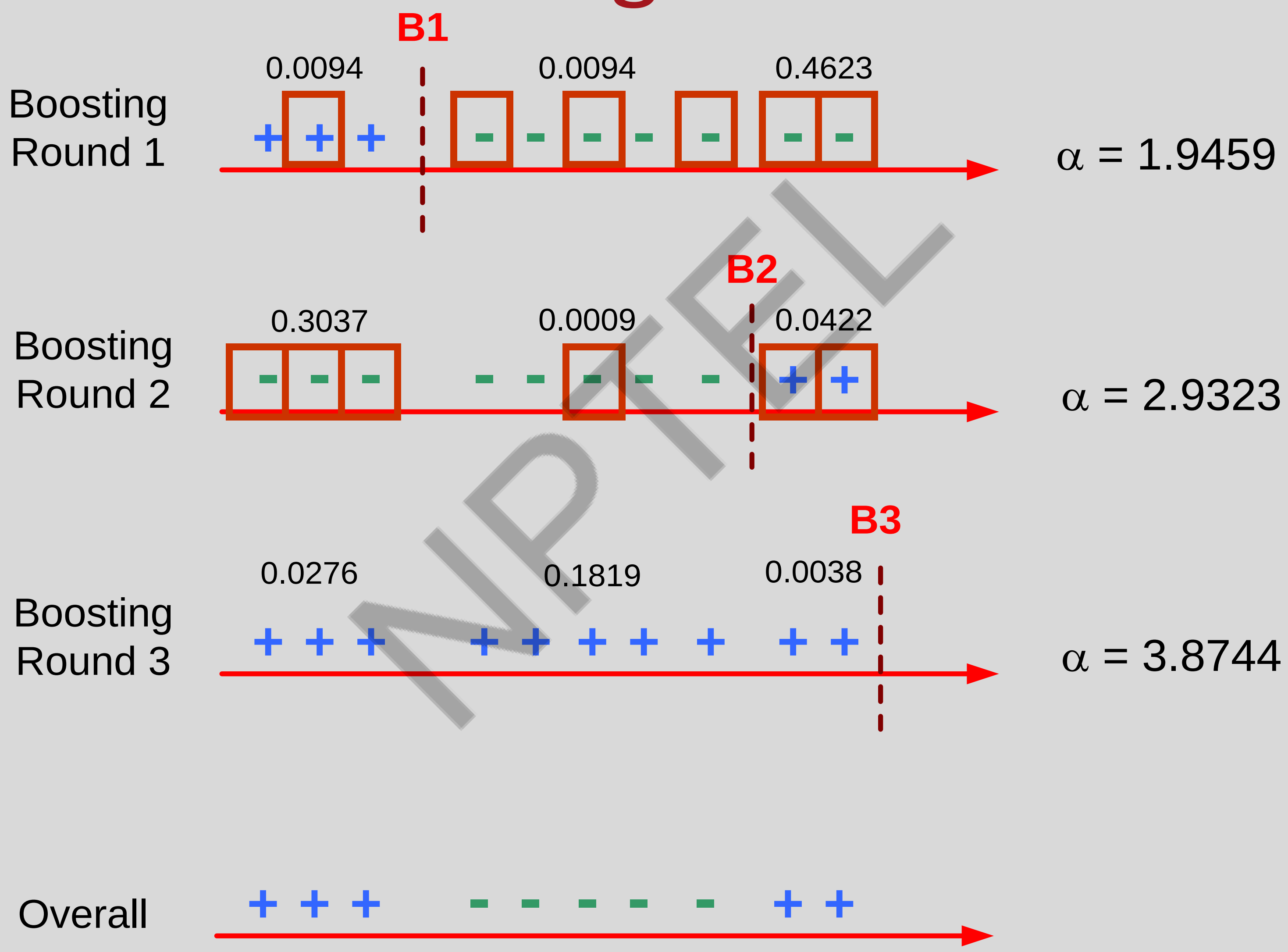
- Ican be shown that AdaBoost will achieve zero training error (expotentially fast):

$$\frac{1}{m}\sum_{i=1}^{m}\delta(H(x_i) \neq y_i) \leq \prod_t Z_t \leq exp\left(-2\sum_{t=1}^{T}(1/2 - \epsilon_t)^2\right)$$

Illustrating AdaBoost



Illustrating AdaBoost



Thank You