

Computational Learning Theory

Some slides were adapted/taken from various sources, including Prof. Andrew Ng's Coursera Lectures, Stanford University, Prof. Kilian Q. Weinberger's lectures on Machine Learning, Cornell University, Prof. Sudeshna Sarkar's Lecture on Machine Learning, IIT Kharagpur, Prof. Bing Liu's lecture, University of Illinois at Chicago (UIC), CS231n: Convolutional Neural Networks for Visual Recognition lectures, Stanford University, Prof. Alexander Ihler and many more. We thankfully acknowledge them. Students are requested to use this material for their study only and **NOT** to distribute it.

Goals of Learning Theory

- Hypothesis Space and Inductive Bias
- Concepts
 - What kinds of tasks are learnable?
 - What kind of data is required for learnability?
 - What are the (space, time) requirements of the learning algorithms?
- Models
 - Develop algorithms that provably meet desired criteria
 - Prove guaranties for success full algorithms

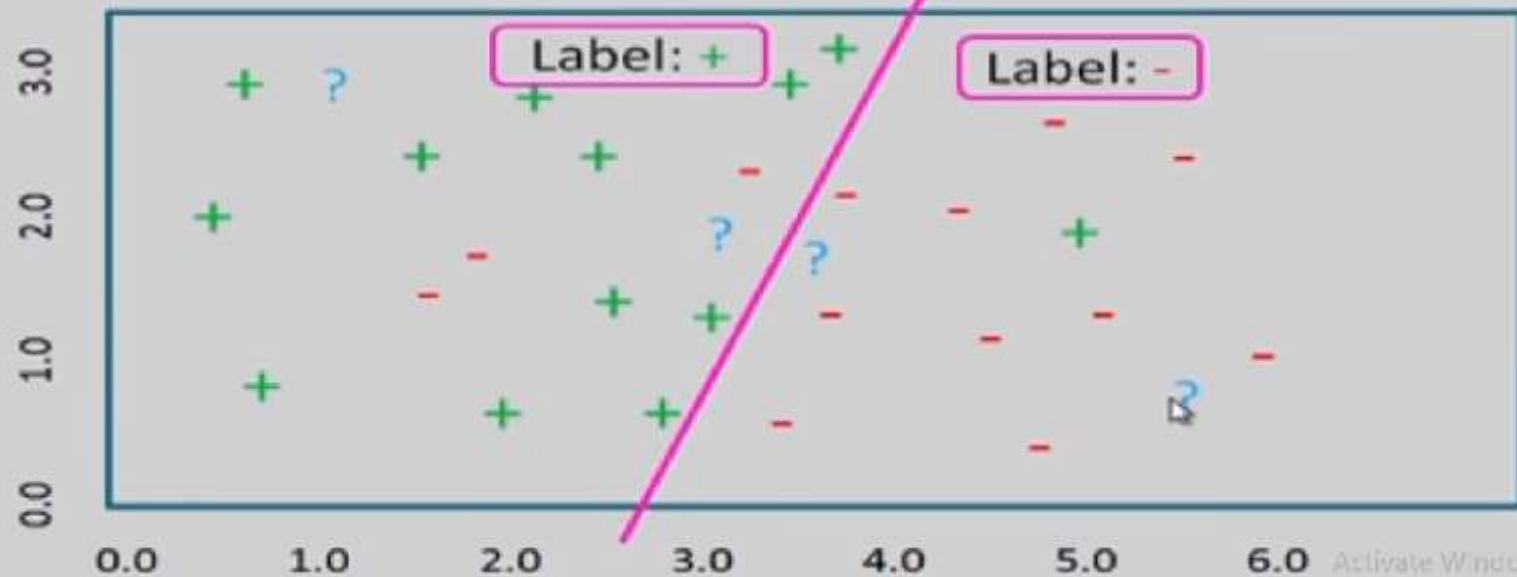
Inductive Learning

- We are given some data, and we are trying to do induction to try to identify a function (f) which can explain the data (\hat{x}).
- $y = f(\hat{x})$
- Examples: (\hat{x}, y) or $(\hat{x}, f(\hat{x}))$
- So induction as opposed to deduction.
- Unless we can see all the instances, all the possible data points or we make some restrictive assumptions about the language in which the hypothesis is expressed or some bias, this problem is not well defined. So that is why it is called an **Inductive Problem**.

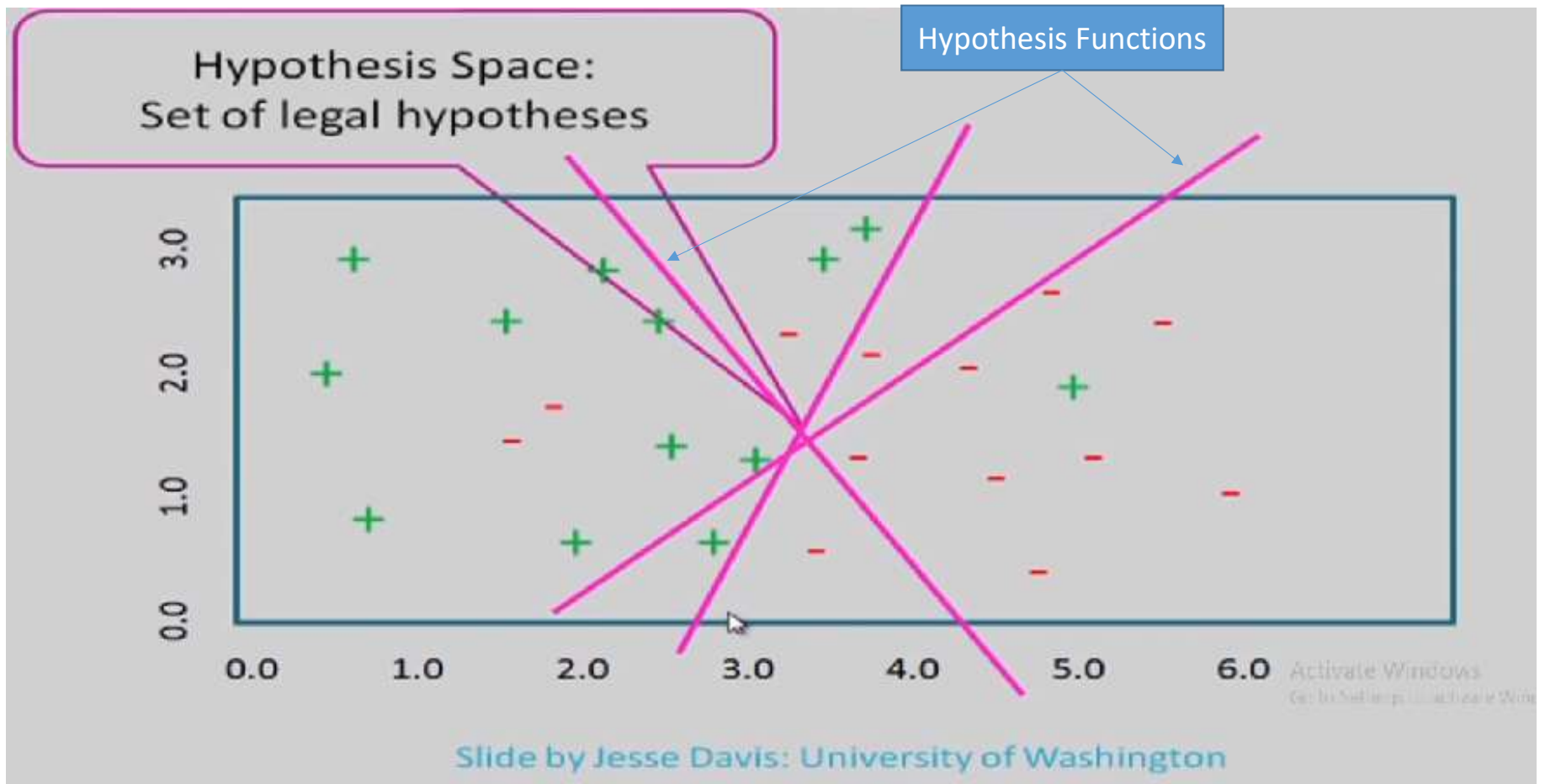
Features

- Classification: $f(\hat{x})$ is discrete
 - Regression: $f(\hat{x})$ is continuous
 - Probability estimation: probability of the \hat{x} .
-
- Features are properties that describe each instances. Each instances can be described in quantitative manner by the features
 - Multiple feature: feature vector

Hypothesis:
Function for labeling examples



the hypothesis space is the set of all possible functions that a machine learning algorithm can output, while a function is a specific element of that space that has been learned from the available data. The goal of machine learning is to find the best function within the hypothesis space



hypothesis space (i.e., the set of all possible functions that can be represented by the chosen model or learning algorithm)

Language of Function

Hypothesis Space

- Hypothesis Space:
 - The space of all hypothesis that can in principle be output by a learning algorithm.
 - We can think about a supervised learning machine as a device that explore a hypothesis space.
 - Each setting of the parameters in the machine is a different hypothesis about the function that maps the input vectors to output vectors.

$$h \in \mathcal{H}$$

o/p of learning algo. Hypothesis Space

$$\text{price} = w_1 * \text{size} + w_2 * \text{bedrooms} + b$$

where w_1 and w_2 are the weights or parameters that determine the importance of each input variable in the prediction, and b is the bias or intercept term that determines the base price of a house.

Terminology:

- Example (\hat{x}, y) : instance x with label $y=f(x)$
- Training data S : Collection of training data observed by learning algorithm
- Instance (feature) Space X : Set of all possible objects described by features
- Concept C : Subset of objects from X : $C \subseteq X$ (C is unknown)
- Target function f : maps each instance $x \in X$ to the target label $y \in Y$

Classifier

- Hypothesis h : Find function $h \in \mathcal{H}$ to approximate f .
- Hypothesis Space \mathcal{H} : set of functions we allow for approximating f
- The set of hypothesis that can be produced, can be restricted further by specifying a language bias
 - Constraints
 - Preferences
- Input: Training set $S \subseteq X$
- Output: $h \in \mathcal{H}$

Hypothesis Space: Example

- If there are N (let $N=4$) input features, there are 2^{16} ($2^{2^4} = 2^{2^N}$) possible Boolean functions.
- We can't figure out which one is correct unless we see every possible i/p-o/p pair 2^4 (2^N)
- So for 4 input features ($N=4$), possible instances = 2^4 (2^N).
- Functions will classify some of the points as +ve and some of the points are -ve, out of 16 points.
- So the no. of functions are no. of possible subsets of these $2^4 = 16$ instances. Since binary classification, it is $2^{2^4} = 2^{16}$
- So, it seems hypothesis space is very large.

Inductive Bias

- Since, hypothesis space may be very large, we can select a hypothesis language
- May be un-restricted or restricted
- In case of restriction, a hypothesis language reflects an inductive bias of the learner.

Inductive Bias

- Need to make assumption
 - Experience alone does not allow us to make conclusions about unseen data instances
- Two types of bias(s)
 - Restriction Bias: Limit the hypothesis space
 - Preference Bias: Impose ordering on hypothesis space

Restriction bias: This can be done by constraining the model parameters or by imposing assumptions on the structure or form of the model. By restricting the hypothesis space, the learner can focus on a smaller set of models that are more likely to generalize well and avoid overfitting. However, if the hypothesis space is too restricted, the learner may miss the true solution or be unable to capture the complexity of the underlying data.

Preference bias: based on some prior knowledge or assumption about the problem. This can be done by specifying a loss function or objective that favors certain types of solutions, or by incorporating domain-specific knowledge or constraints into the learning algorithm. By imposing a preference bias, the learner can bias the search towards models that are more likely to be correct or more suitable for the problem at hand. However, if the preference bias is too strong, the learner may miss alternative solutions or be unable to generalize to new data.

Inductive Learning

- Inductive learning: Inducing a general function for training examples.
 - Construct hypothesis h to agree with (Concepts) c on the training examples
 - A hypothesis is consistent if it agrees with all training examples
 - A hypothesis is said to generalize well if it correctly predicts the values of Y for novel examples. (refers to a new or previously unseen instance)
- Inductive learning is an ill-posed problem
 - Unless we see all possible examples the data is not sufficient for an inductive learning algorithm to find a unique solution.
 - Any hypothesis h found to approximate the target function C well over a sufficiently large set of training examples D will also approximate the target function well over unobserved examples.

Learning as Referencing the Hypothesis Space

- Concept learning is task of searching an hypothesis space of possible representations looking for the representation (S) that best fits the data points, given the bias.
- The tendency to prefer the hypothesis over another is called a bias (Preference bias).
- Given representation, data and a bias the problem of learning can be reduced to one of the SEARCH problems.

Occam's Razor

- A classical example of inductive bias
 - The simplest consistent hypothesis about the target function is actually the best.
- Some other types of inductive bias
 - Minimum description length: when forming a hypothesis, attempt to minimize the length of the description of the hypothesis.
 - Maximum Margin: when drawing boundary between two classes, attempt to maximize the width of the boundary (Example: SVM)

Some Important Issues of Machine Learning

- What are the good hypothesis space?
- Algorithm that works with the hypothesis space.
- How to optimize the accuracy over the future data points (handling overfitting)?
- How can we have confidence in the results? (How much training data?)
- Are some training problems computationally intractable?

Goals of Learning Theory

- Hypothesis Space and Inductive Bias
- 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 algorithms?
- To develop and analyze model
 - Develop algorithms that provably meet desired criteria
 - Prove guaranties for success full algorithms

Learning Theory

- What is the task ? (e.g. spam filtering)
- What types of data?
- Resource requirements
- Develop algorithms (under certain constraints) are provable?
- Confidence of the algorithm

Learning algorithms

- Design algorithms
 - In machine learning, we design algorithms that optimizes certain criteria
- Confidence of the algorithms
 - Generalization ability of the algorithm
 - How well this an algorithm works on testing data (future data)



PAC (Probably approximately correct) Models

C is the class, we wish to learn, h is the hypothesis class, h may not fully cover c , so there are some error region,

$$E = C \text{ xor } H$$

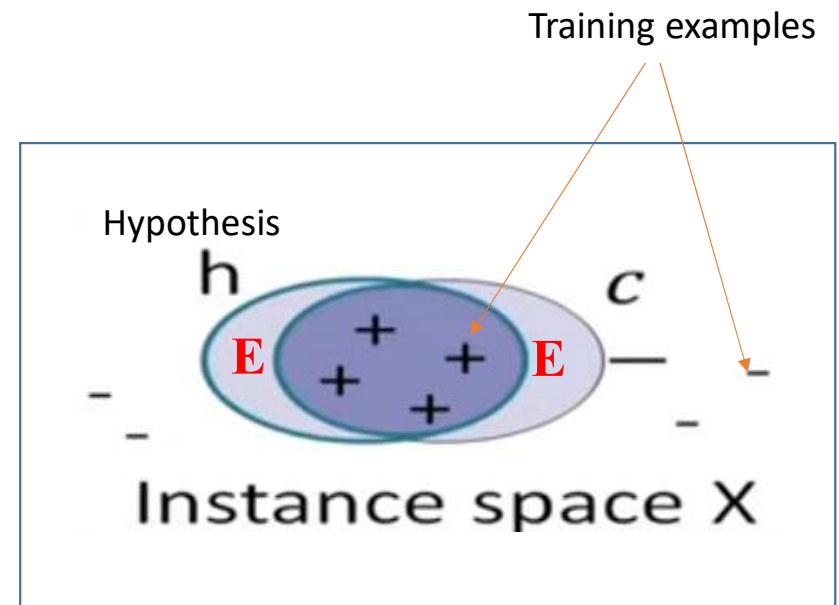
We want the probability of this error region to be small i.e.

$$P(C \text{ xor } H) \leq \epsilon$$

So, the hypothesis should be approximately correct and the error tolerance

$$0 \leq \epsilon \leq \frac{1}{2}$$

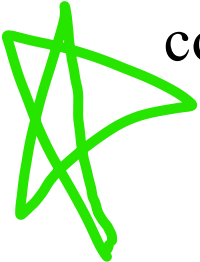
The condition $0 \leq \epsilon \leq 1/2$ means that the target error rate should be less than or equal to half the minimum error rate of any hypothesis in the hypothesis class. This condition ensures that the hypothesis class is rich enough to contain a hypothesis that can achieve the desired level of accuracy.



concept class is the set of all possible solutions to a learning problem, while the hypothesis class is a subset of the concept class that is considered by a learning algorithm.

PAC (Probably approximately correct) Models

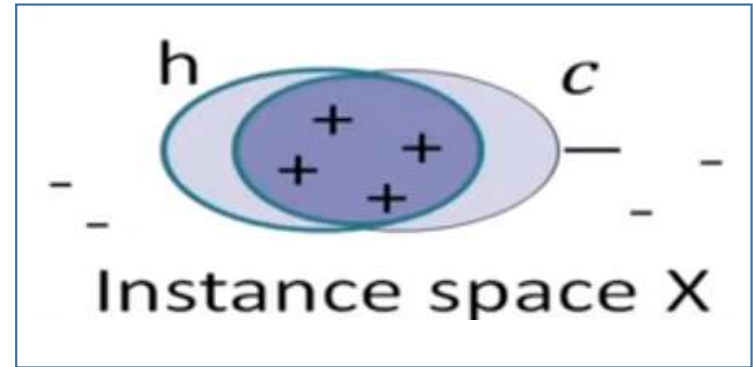
- Now hypothesis function is not always fully correct
- So the probability that the learning algorithm gives us the approximately correct hypothesis


$$P(P(C \text{ xor } H) \leq \epsilon) \geq (1 - \delta) \quad \text{where } \delta \text{ the confidence parameter and } 0 \leq \delta \leq \frac{1}{2}$$

The probability that the learning algorithm gives us an approximately correct hypothesis is denoted by $P(C \text{ xor } H)$, where ϵ is the error tolerance and C is the true concept class. The confidence parameter specifies how confident we want to be that the hypothesis we obtain is approximately correct.

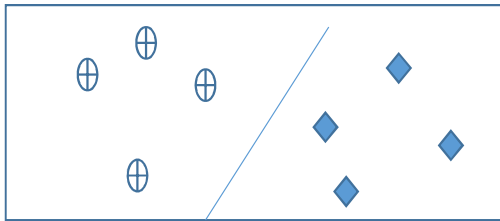
Prototypical Concept Learning Task

- Given
 - Instances X (e.g., $X = \mathbb{R}^d$ or $X = \{0,1\}^d$)
 - Distribution \mathcal{D} over X
 - Target function C
 - Hypothesis space \mathcal{H}
 - Training Examples $S = \{(x_i, c(x_i))\}$ i.i.d from \mathcal{D}
- Determine
 - A hypothesis $h \in \mathcal{H}$ s.t. $h(x) = c(x)$ for all x in S ?
 - A hypothesis $h \in \mathcal{H}$ 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}

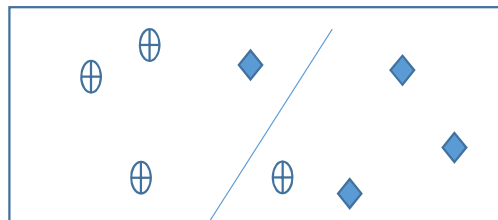


Consistent Hypothesis

- Given the training samples, find the hypothesis h such that it agrees with the training examples, is called **consistent hypothesis**.

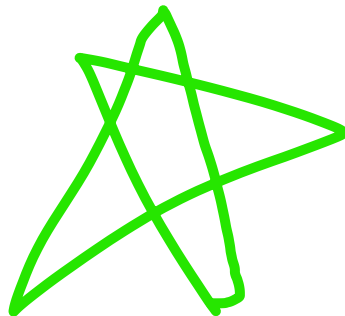


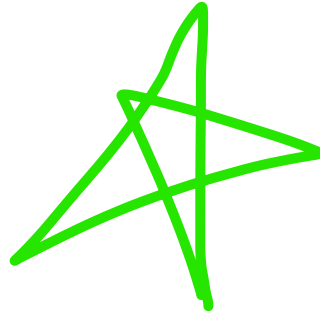
- For certain cases, we can't look for consistent hypothesis, so allow the inconsistent hypothesis.



Restriction Bias:

- Can we be certain about how the learning algorithm generalizes?
- We would have to see all examples.
- Inductive inference
 - Generalize beyond the training data is impossible unless we add more assumptions (e.g. priors over \mathcal{H}).
 - We need a **bias**.
- Restriction Bias: We can put a bias on the form of hypothesis class to allow hypothesis of certain restricted types, what is called **restriction bias**.
- In other words, it restricts the hypothesis space to get **simpler hypothesis**.





Preference Bias:

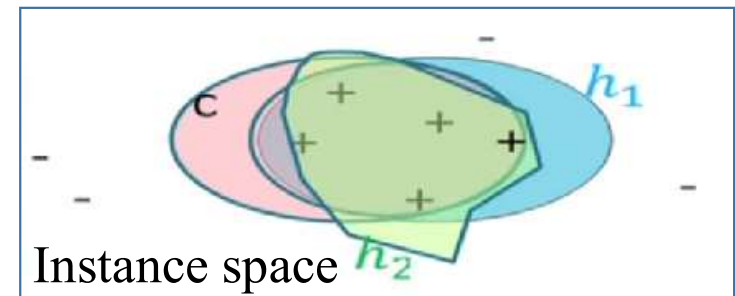
The tendency to prefer a hypothesis over another.

- Prefer some hypothesis over others. Small decision tree over large decision tree.

Function Approximation

- How many labelled examples are necessary in order to determine which of the 2^{2^N} hypothesis is the correct one?
- All 2^N instances are labelled.
- Inductive inference: Generalizing beyond the training data is impossible unless we add more assumption (e.g. bias)
- $H = \{h: X \rightarrow Y\} \quad |H| = 2^{|X|} = 2^{2^N}$
- N features, 2^N instances, so no. of possible partition of the instance space = 2^{2^N}

To find the one correct hypothesis, we have to check every 2^N instances.



Error of a hypothesis

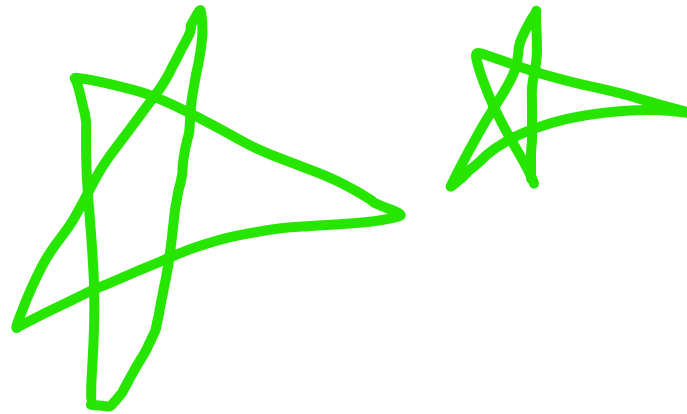
- h measure the error (sample error) of h on sample S .
- The true error of the hypothesis h , with respect to the target concept c observation distribution \mathcal{D} is the probability that h will misclassify an instance drawn according to the D

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

- In a perfect situation, we'd like the true error to be 0.
- Bias: Fix hypothesis space \mathcal{H} such that $h \in \mathcal{H}$
- C may not be in $\mathcal{H} \rightarrow$ Find h close to c
- A hypothesis h is approximately correct if

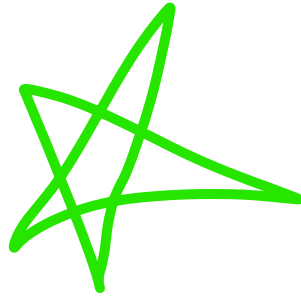
$$error_{\mathcal{D}}(h) \leq \varepsilon$$

PAC Model



- Goal: h has small error over D .
- True error: $error_{\mathcal{D}}(h) = Pr_{x \sim \mathcal{D}} [h(x) \neq c^*(x)]$ the true error is not known, as we typically don't have access to the true concept class or the observation distribution.
- How often $h(x) \neq c^*(x)$ over future instances drawn at random from \mathcal{D}
- But, we can only measure:
- Training error : $error_S(h) = \frac{1}{m} \sum_i I(h(x_i) \neq c^*(x))$ since, we cant access true error.
- How often $h(x) \neq c^*(x)$ over training instances
- Sample complexity: how to bound $error_{\mathcal{D}}(h)$ in terms of $error_S(h)$

PAC Learning



- PAC learning concerns **efficient learning**.
- We would like to prove that with high probability ($P(1 - \delta)$) an efficient (**space and time complexity is in order of polynomial time**) learning algorithm will find a hypothesis that is approximately identical (**$\text{error} \leq \epsilon$**) 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 of the supervised learning

- Theorem: If the size of the training sample (m) is greater or equal to Z then it is sufficient to be able to PAC learner hypothesis where

$$Z = \frac{1}{\varepsilon} [\ln(|\mathcal{H}|) + \ln\left(\frac{1}{\delta}\right)] \quad \text{i.e.} \quad m \geq \frac{1}{\varepsilon} [\ln(|\mathcal{H}|) + \ln\left(\frac{1}{\delta}\right)]$$

Labeled example are sufficient so that with probability $1-\delta$, all $h \in \mathcal{H}$ with $error_D(h) \geq \varepsilon$ have $error_S(h) \geq 0$ [i.e. PAC]

- inversely linear in ε
- logarithmic to $|\mathcal{H}|$.
- ε error parameters: D might place low weight on certain parts of the space
- δ confidence parameters: there is a small chance the examples we get are not representative of the distribution.

proof?

Sample Complexity of the Supervised Learning

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

Proof: Assume k bad hypotheses $H_{\text{bad}} = \{h_1, h_2, \dots, h_k\}$ with $error_D(h_i) \geq \epsilon$

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

Finite Hypothesis Space Realizable Case

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

Theorem

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

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

$$err_D(h) \leq \frac{1}{m} \left[\ln(|H|) + \ln\left(\frac{1}{\delta}\right) \right] \quad \text{Bounds on True error}$$

Summary

- We have looked at the case where the learner looks at m training examples, outputs a consistent hypothesis.
- We derived bounds on how many examples the learner has to see so that the consistent hypothesis is a PAC hypothesis.
- In other words, we have seen, how many examples are needed to be seen in order to guarantee that consistent hypothesis has $\text{error} \leq \epsilon$ and we want to do this with the confidence $(\geq 1 - \delta)$.
- But there are scenarios where $\text{we can't get consistent hypothesis}$ but the hypothesis that the learner outputs is inconsistent that is it has error in the training sets.

Finite Hypothesis Space: Inconsistent hypothesis

- The hypothesis output by the learner has non zero errors in the training sets.

Finite Hypothesis Space Realizable Case

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

Theorem

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

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

$$err_D(h) \leq \frac{1}{m} \left[\ln(|H|) + \ln\left(\frac{1}{\delta}\right) \right] \quad \text{Bounds on True error}$$

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

- For a single hypothesis to have misleading training error

$$\Pr[\text{error}_{\mathcal{D}}(f) \leq \varepsilon + \text{error}_D(f)] \leq e^{-2m\varepsilon^2}$$

- 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

$$\Pr[(\exists f \in \mathcal{H}) \text{error}_{\mathcal{D}}(f) \leq \varepsilon + \text{error}_D(f)] \leq |\mathcal{H}|e^{-2m\varepsilon^2}$$

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

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

Sample Complexity: Finite Hypothesis Space

Consistent Case

Theorem

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

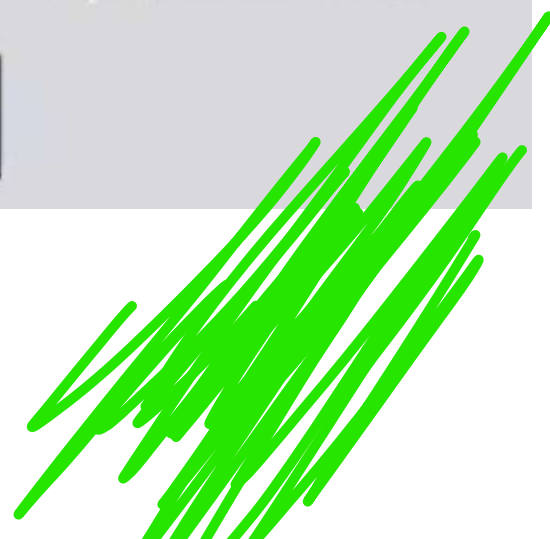
labeled examples are sufficient so that with prob. $1 - \delta$, all $h \in H$ with $err_D(h) \geq \epsilon$ have $err_S(h) > 0$.

Inconsistent Case

What if there is no perfect h ?

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

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



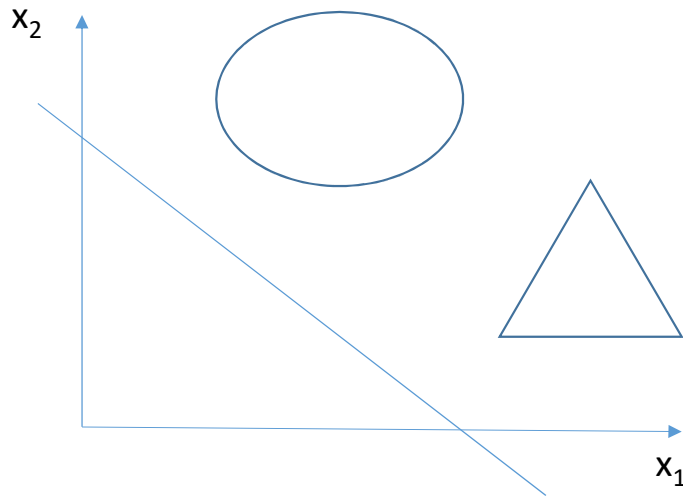
VC Dimension

•

Sample Complexity: Infinite Hypothesis Space

- Hypothesis Space is **infinite**.
- Need some measure of expressiveness of infinite hypothesis spaces.
- The **Vapnik-Chervonenkis (VC) dimension** provides such measures, denoted $VC(H)$.
- Analogous to $\ln(|\mathcal{H}|)$, there are bounds for sample complexity using $VC(H)$

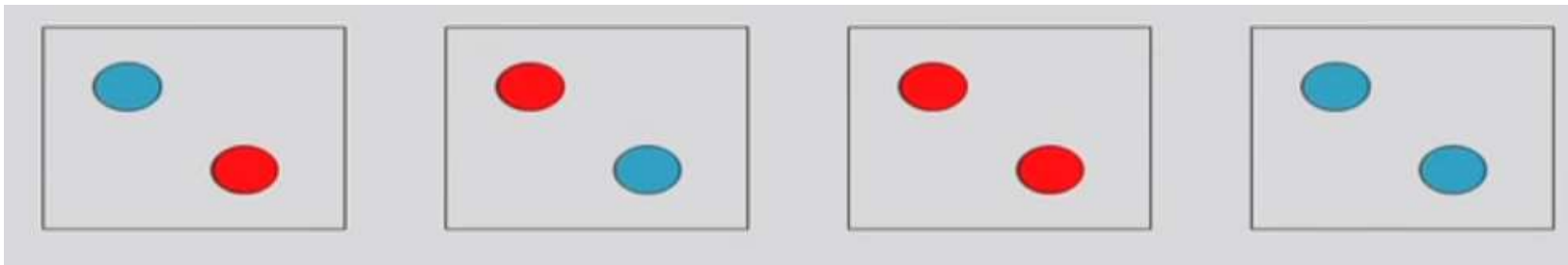
Examples of Infinite Hypothesis Space



x_1 and x_2 are real valued attributes, then the number of linear functions can be infinite.

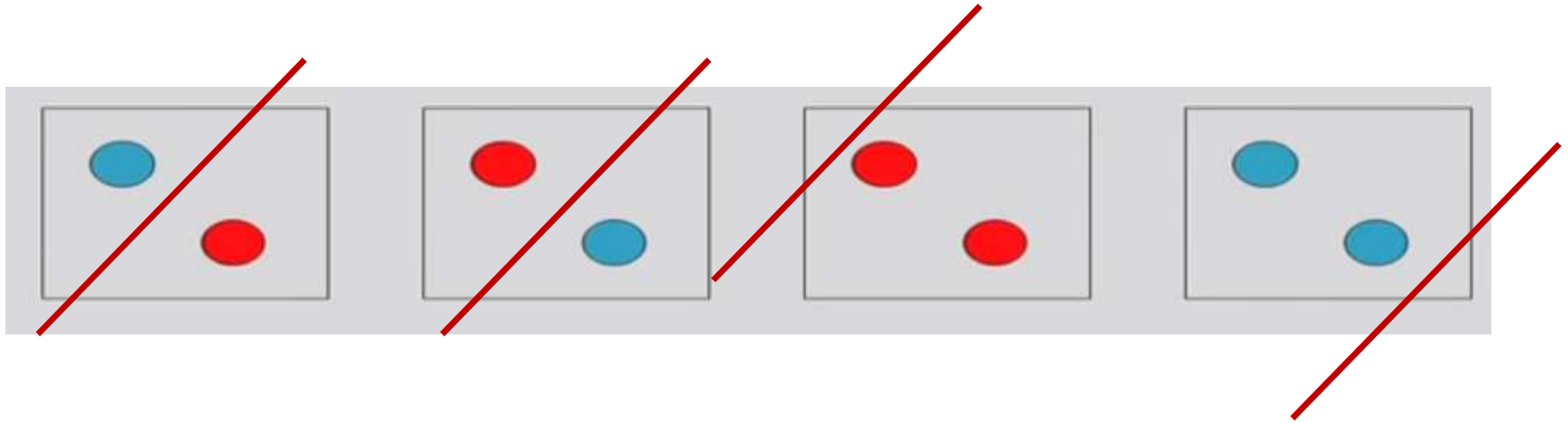
Shattering

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



a set of instances S is shattered by a hypothesis class H if and only if for every binary labeling function $f : S \rightarrow \{0, 1\}$, there exists a hypothesis h in H such that $h(x) = f(x)$ for all x in S .

Shattering

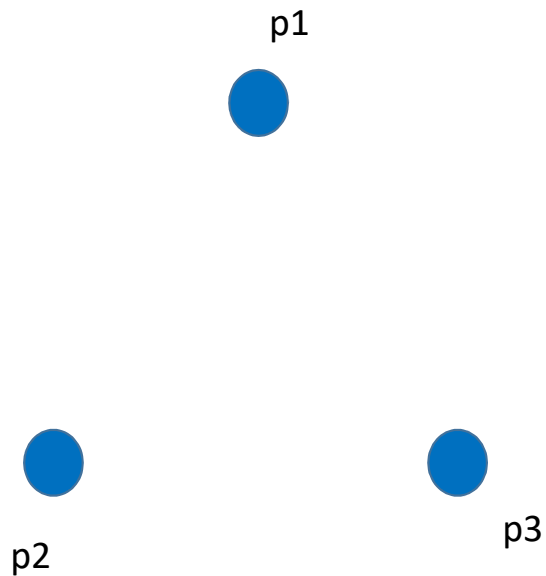


So all possible labeling of these two points, we can find a hypothesis from the hypothesis space which is consistent with the labeling.

for every possible binary labeling of the instances, there exists a hypothesis in the hypothesis class that can perfectly separate the instances according to their labels.

For binary classification, how many possible ways we can partition this space? It is $2^3=8$.

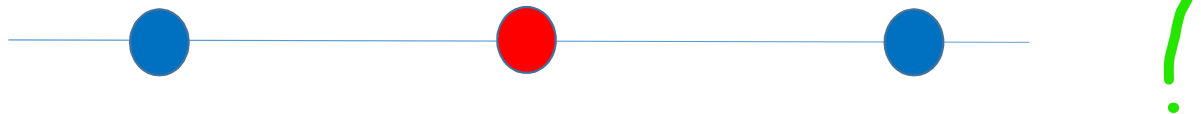
And for every possible such labeling, we can find a linear separator those points.



Can a linear separator partition following setting?

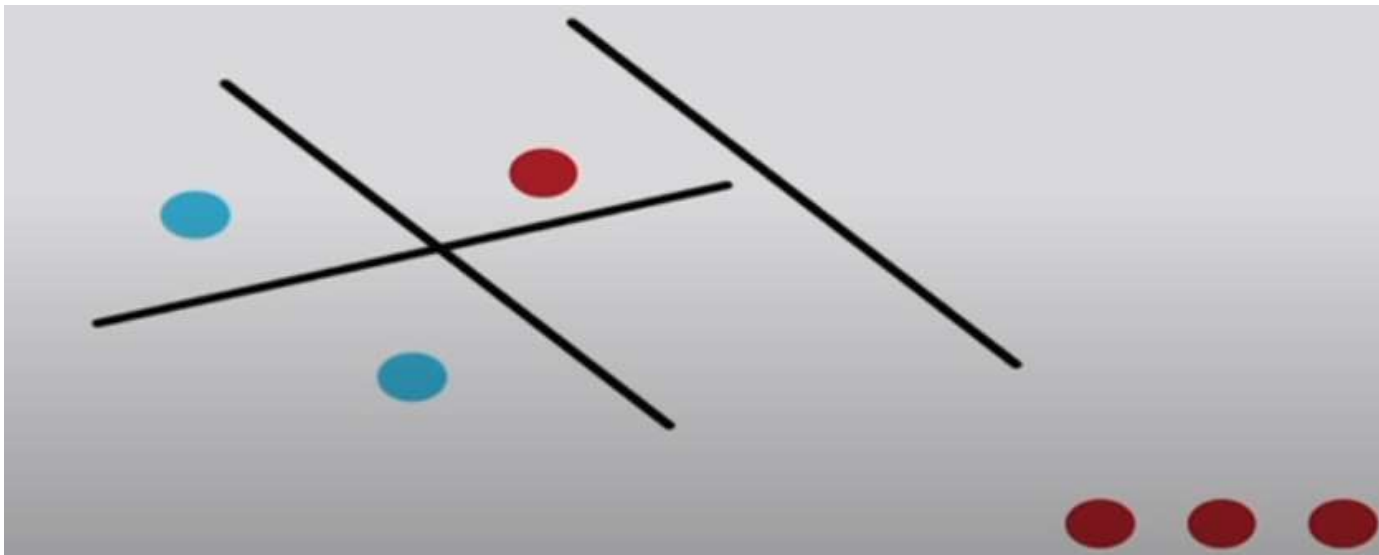
So, this particular setting of three points can not be shattered by a hypothesis space comprises of straight lines.

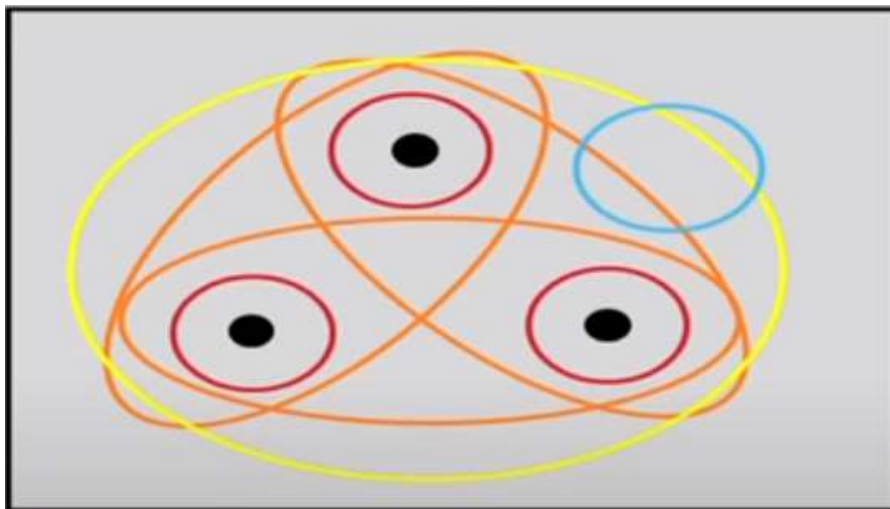
i.e, the hypothesis class $H(\text{linear func})$ can't shatter this labelling.



Three points in \mathbb{R}^2

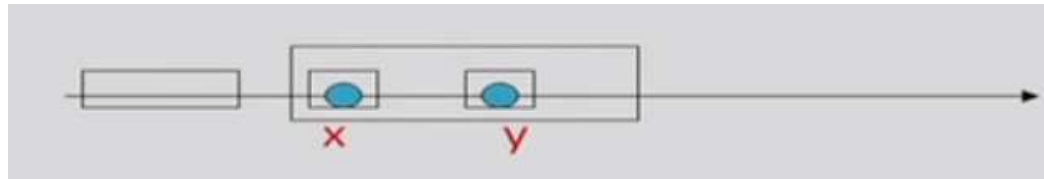
- It is enough to find one set of three points that can be shattered.
- It is not necessary to be able to shatter every possible set of three points in 2 dimension.





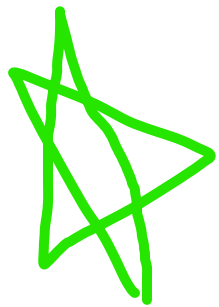
Shattering Instances

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

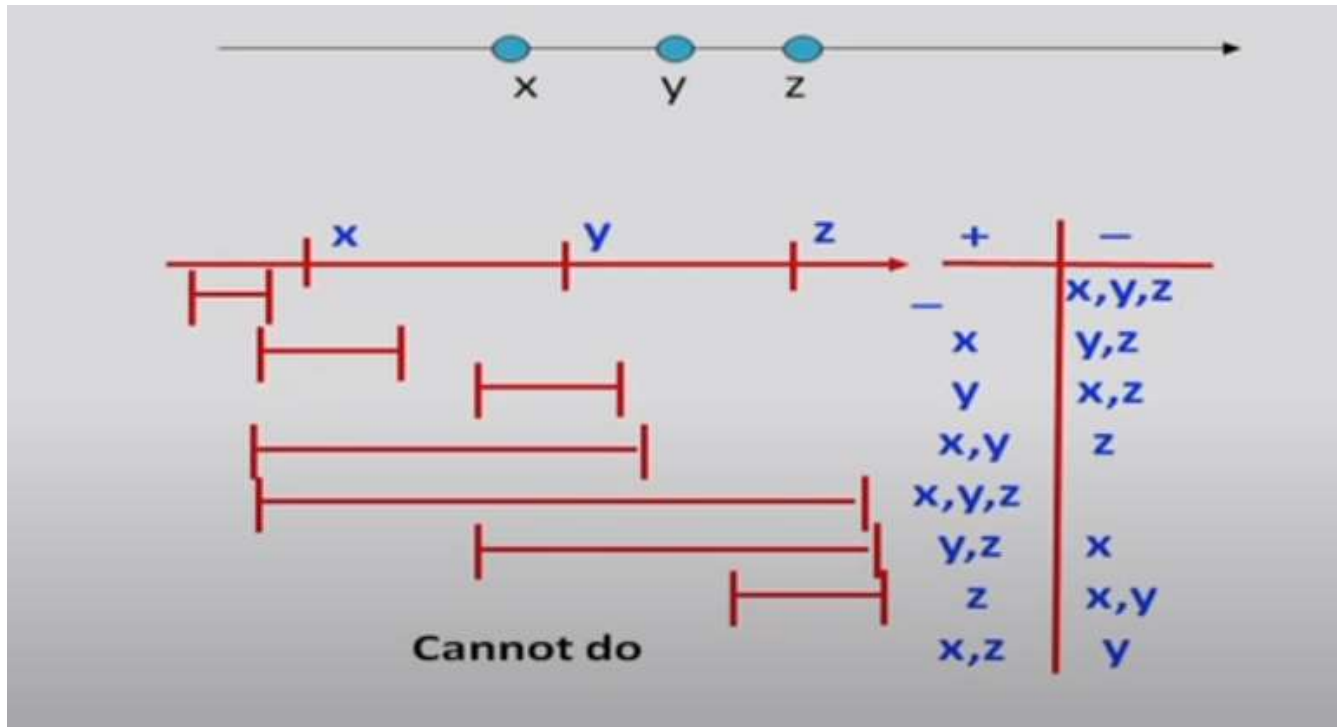


Shattering Instances

But three instances can not be shattered by a single interval.



same as above fig.



VC Dimension

- The Vapnin-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 subset of X can be shattered then $VC(H)=\infty$. *not alpha, it is infinite.*
- If there exist at least of subset of X of size d that can be shattered by H then $VC(H)\geq d$.
- If no subset can be shattered by H , then $VC(H)<d$.
- *For a single intervals on the real line, all sets of 2 instances can be shattered, no set of three instances can, $VC(H)=2$.*

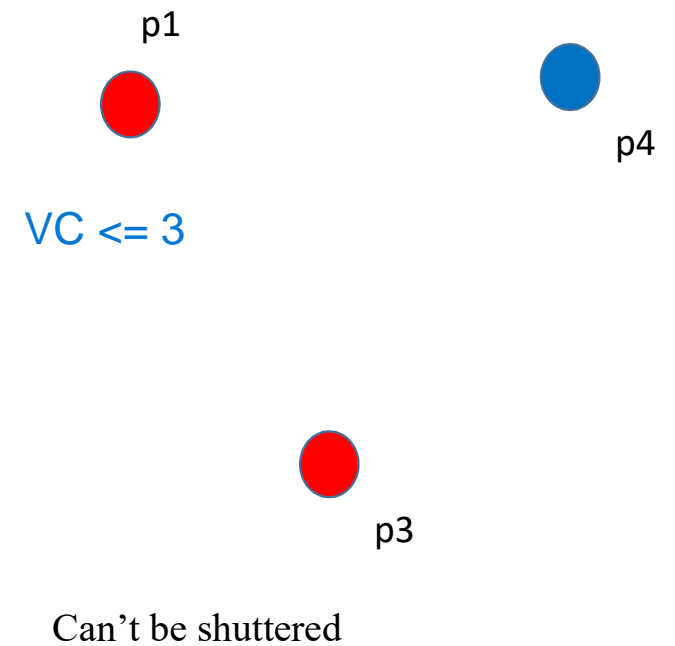
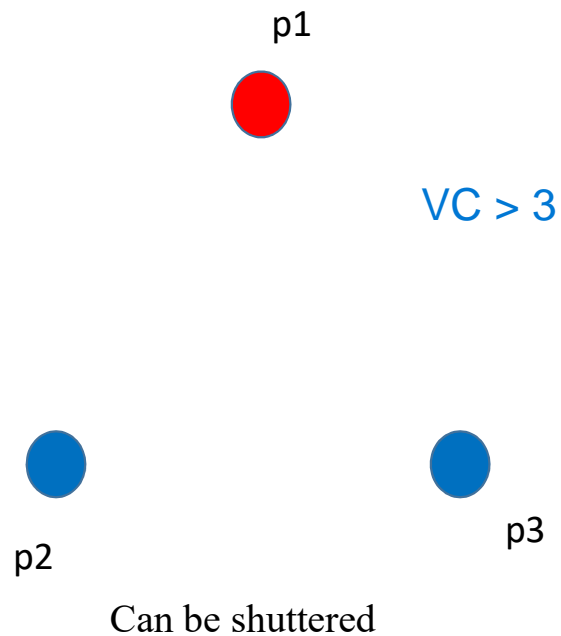
the VC dimension of a hypothesis space that consists of all possible linear classifiers in 2D is 3. This means that there exists a set of three points that can be shattered by the hypothesis space, but no set of four points can be shattered.

The VC dimension of a single interval on the real line, which is 2. This means that any set of two points on the real line can be shattered by this hypothesis space, but no set of three points can be shattered.

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, $VC(H) \leq \log_2 |H|$

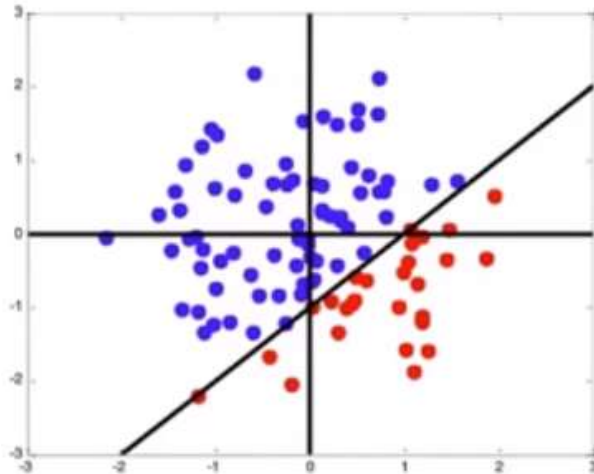
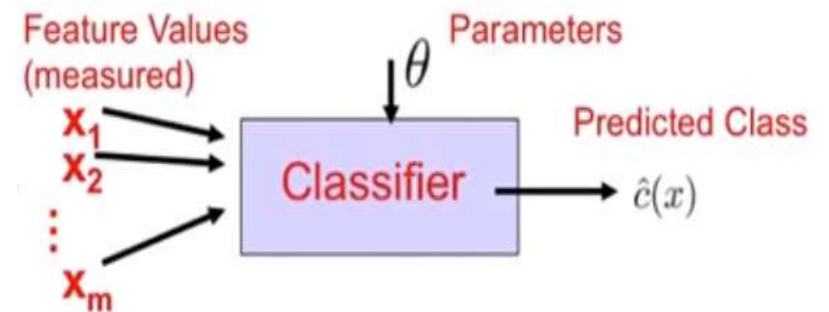
For Binary Classification



So, the VC dimension of the set of oriented lines in 2-D is three.

Learners and Complexity

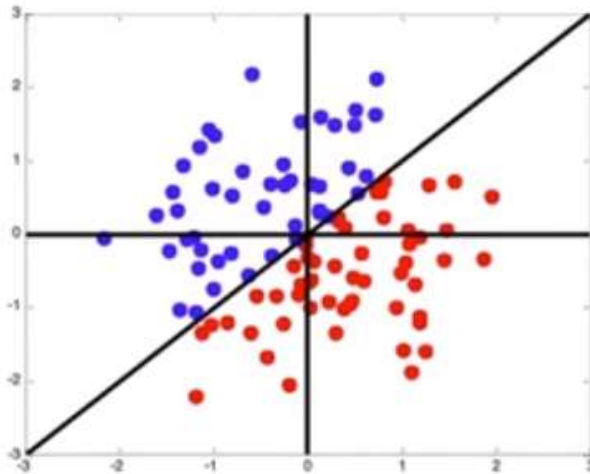
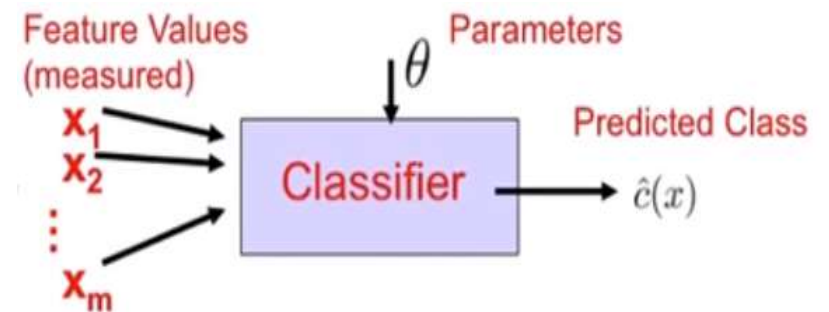
- We have seen many versions of underfit/overfit trade-off.
 - Complexity of the learner
 - Representational power
- Different learners have different power



Example: $\hat{c}(x) = \text{sign}(\theta_1 x_1 + \theta_2 x_2 + \theta_0)$

Learners and Complexity

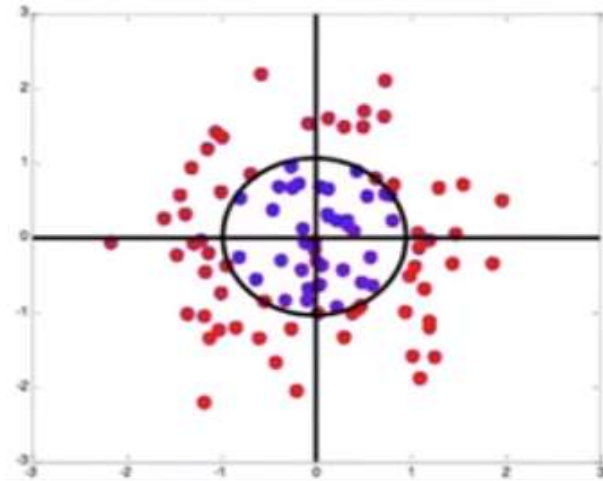
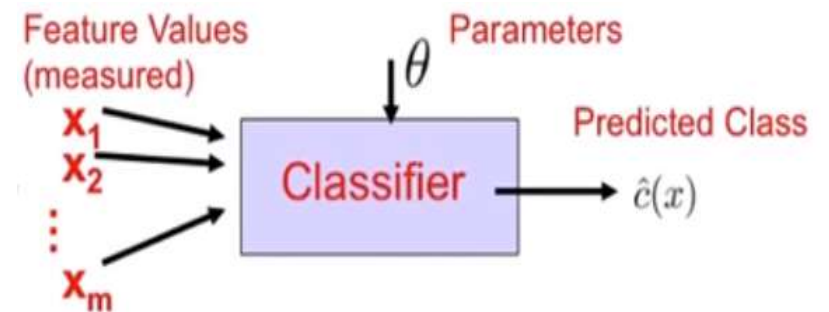
- We have seen many versions of underfit/overfit trade-off.
 - Complexity of the learner
 - Representational power
- Different learners have different power



Example: $\hat{c}(x) = \text{sign}(\theta_1 x_1 + \theta_2 x_2)$

Learners and Complexity

- We have seen many versions of underfit/overfit trade-off.
 - Complexity of the learner
 - Representational power
- Different learners have different power



Example: $\hat{c}(x) = \text{sign}(x^T x - \theta_0)$

Learners and Complexity

- We have seen many versions of underfit/overfit trade-off.
 - Complexity of the learner
 - Representational power
- Different learners have different power
- Usual trade-off:
 - More power = represent more complex systems, might overfit
 - Less power = won't overfit, but may not find best learner
- How can we quantify the representational power?
 - Not easily...
 - One solution is **VC (Vapnik-Chervonenkis) dimension**

Preliminaries

- Let's assume our training data are iid (**independent and identically distributed**) from some distribution $p(x)$.

- Define “risk” and “empirical risk”

- These are just “long term” test and observed training error

$$R(\theta) = \text{TestError} = E[\delta(c \neq \hat{c}(x; \theta))]$$

$$R^{emp}(\theta) = \text{TrainError} = \frac{1}{m} (\sum_i \delta(c^{(i)} \neq \hat{c}(x^{(i)}; \theta)))$$

- How are these related? Depends on overfitting...

- Under-fitting domain: pretty similar
 - Overfitting domain: test results might be lots worse! **good train error, but worst test error**


VC Dimension and Risk

- Given some classifier, let H be its VC dimension
 - Represents the “representational power” of the classifier.

$$R(\theta) = \text{TestError} = E[\delta(c \neq \hat{c}(x; \theta))]$$

$$R^{\text{emp}}(\theta) = \text{TrainError} = \frac{1}{m} (\sum_i \delta(c^{(i)} \neq \hat{c}(x^{(i)}; \theta)))$$

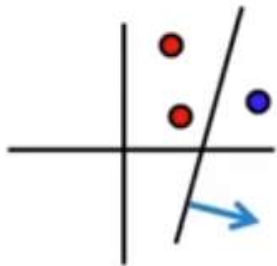
- With high probability $(1-\eta)$, Vapnik showed


$$\text{Test Error} \leq \text{Train Error} + \sqrt{\frac{H \log\left(\frac{2m}{H}\right) + H - \log\left(\frac{\eta}{4}\right)}{m}}$$

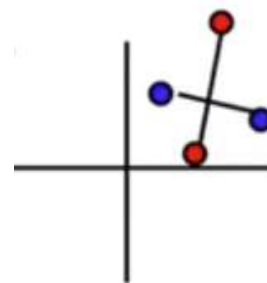
VC Dimension: Examples

- What is the VC dimension of the two dimensional line,

$$f(x; \theta) = \text{sign}(\theta_1 x_1 + \theta_2 x_2 + \theta_0)?$$



VC dimension is ≥ 3 YES



VC dimension is ≥ 4 : NO

It turns out, for a general linear classifier (perceptron) in d dimensional space with a constant term, the VC dimension is $d+1$

VC Dimension

- VC dimension measures the power of the learner.
- Does **not** necessarily equal to the # of parameters.
- Number of parameters does not necessarily equal complexity
 - Can define a classifier with lot of parameters but not much power (how?)
 - Can define a classifier with one parameter but lots of power (how?)
- Lots of work to determine what the VC dimension of various learners is... it is a challenging task to calculate VC of a learner.

For example, a decision tree with a small number of parameters may have a high VC dimension because it can represent very complex functions that can shatter a large number of points in the instance space. On the other hand, a neural network with a large number of parameters may have a lower VC dimension because it is more constrained by its architecture and regularization methods.

Using VC dimension

- Used validation/cross-validation to select complexity
 - Fundamental problem of model selection

The VC dimension can be used to help select the appropriate model complexity. It provides a measure of the expressive power of a given hypothesis space, which can be used to determine the optimal complexity of the model.



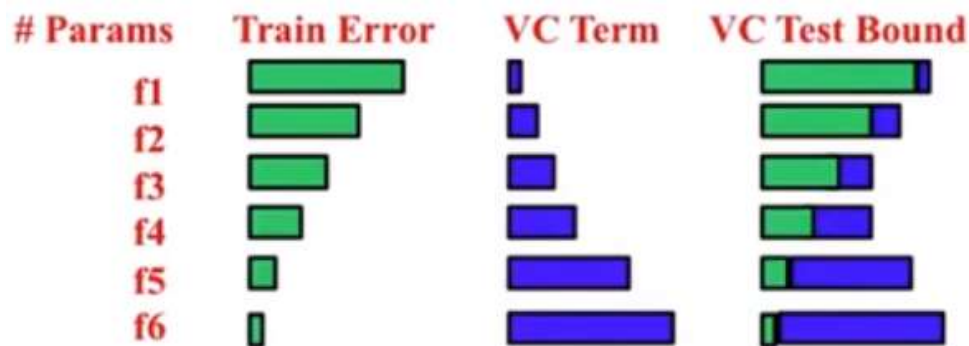
One approach to model selection using the VC dimension is to use validation or cross-validation techniques to estimate the generalization error of different models. This involves dividing the data into training and validation sets and evaluating the performance of different models on the validation set. The model with the lowest validation error is then selected as the best model.

Using VC dimension

- Used validation/cross-validation to select complexity
- Use VC dimension based bound on test error similarity

The use of VC dimension-based bounds in SRM provides a way to control overfitting and ensure that the selected model is not too complex for the available data.

- Structural risk minimization (SRM)



The basic idea of SRM is to find the model that minimizes the expected risk (sum of the training error and the complexity penalty). complexity penalty is a function of the VC dimension of the model, which is a measure of its complexity.

The SRM principle suggests that the best model is the one that minimizes the expected risk, subject to a constraint on the model's complexity. The constraint is typically expressed as a bound on the VC dimension of the model, which limits the model's ability to fit the data. By selecting the model with the smallest VC dimension that satisfies the constraint, we can find a good trade-off between bias and variance and minimize the generalization error.

to continue...