INTRODUCTION TO

PAC LEARNING

by: *alpaydin@boun.edu.tr*
http://www.cmpe.boun.edu.tr/~ethem/i2ml

Postedited by*: R. Basili*
a.a. 2020-21

Overview

- PAC learnability
- The Vapnik-Chervoniensky dimension
- Model selection in ML methods
  - Error and Model Complexity
- Structural Risk Minimization
- VC-dimension vs. other Model Optimization methods

Learning a Class from Examples

- Class C of a “family car”
  - Prediction: Is car x a “family car”?
  - Knowledge extraction: What do people expect from a family car?

- Output:

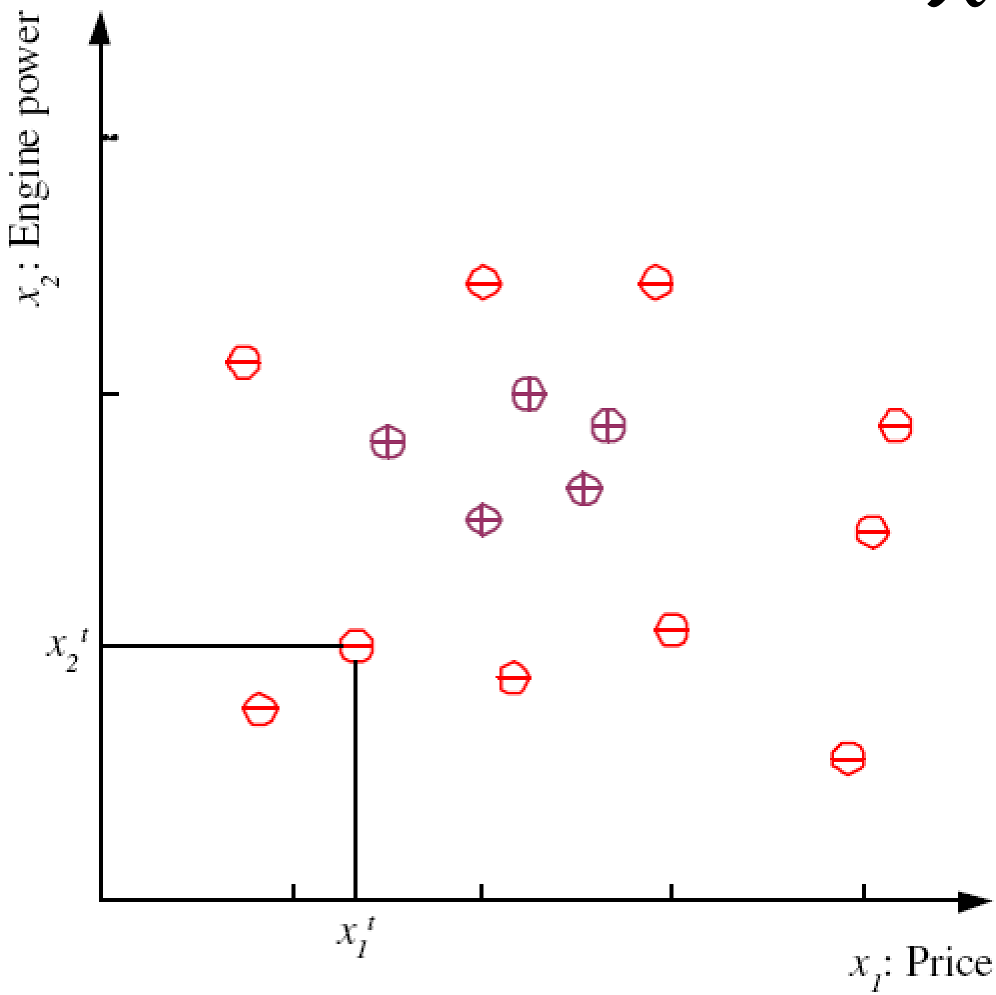
  Positive (+) and negative (–) examples
- Input representation:

  x_1: price, x_2 : engine power

Training set $\mathcal{X}$

$$\mathcal{X} = \{\boldsymbol{x}^t, r^t\}_{t=1}^{N}$$

label

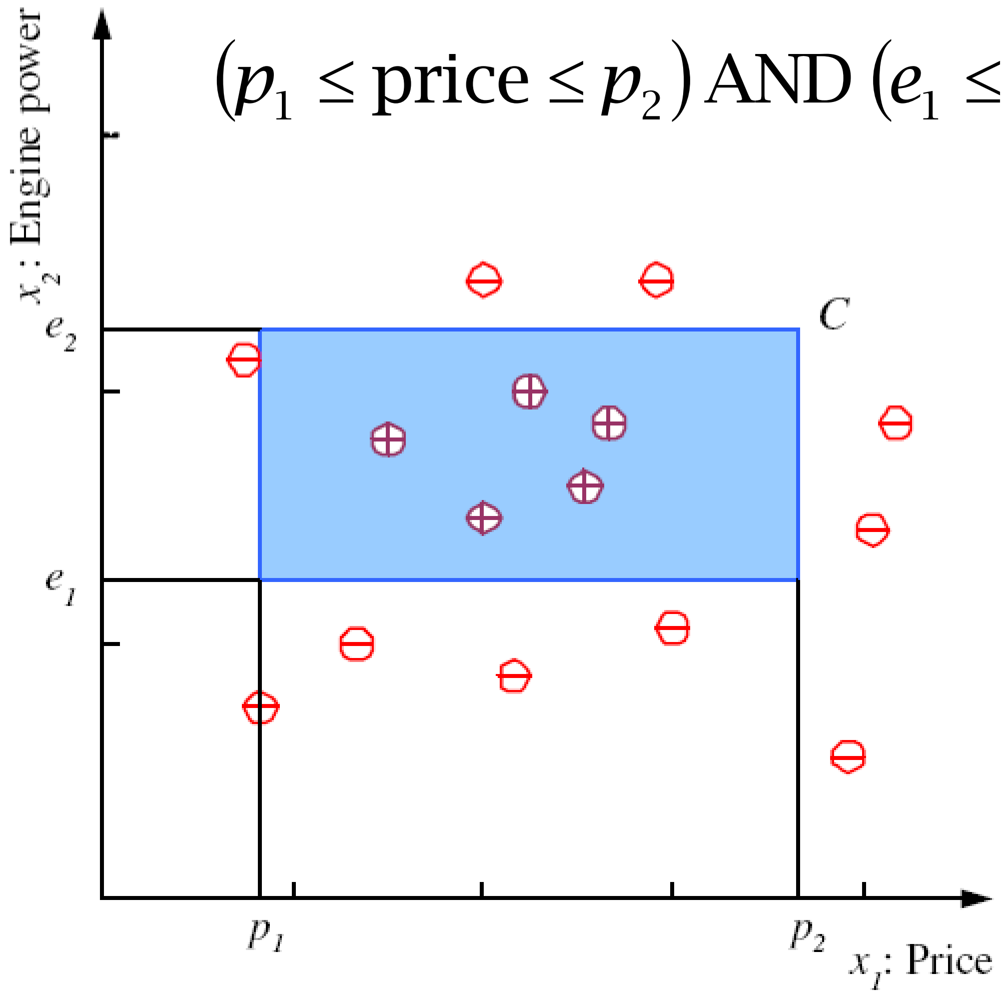


$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$r = \begin{cases} 1 \text{ if } \boldsymbol{x} \text{ is positive} \\ 0 \text{ if } \boldsymbol{x} \text{ is negative} \end{cases}$$

Class C

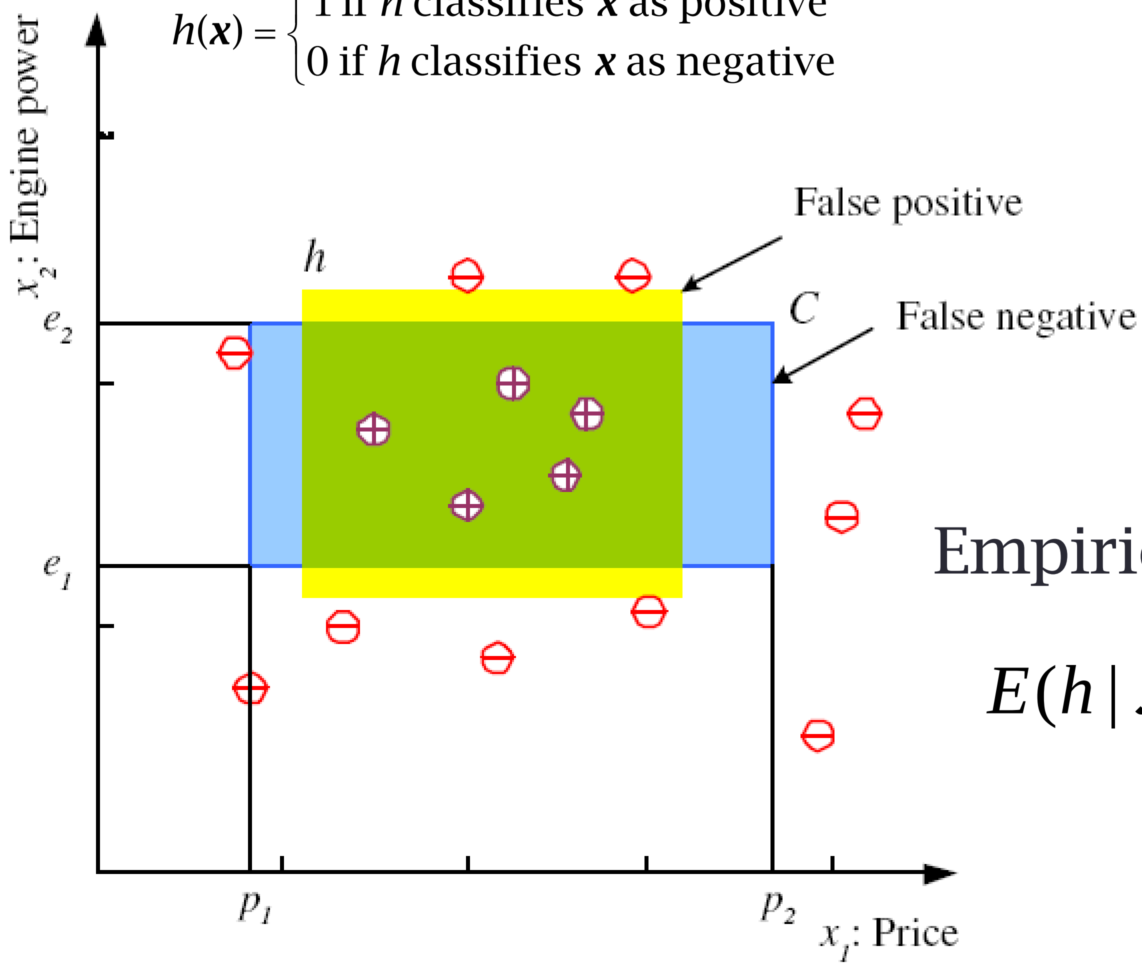
$$(p_1 \le \text{price} \le p_2) \text{ AND } (e_1 \le \text{engine power} \le e_2)$$



In general we do not know C(x).

Hypothesis class $\mathcal{H}$

$$h(\boldsymbol{x}) = \begin{cases} 1 \text{ if } h \text{ classifies } \boldsymbol{x} \text{ as positive} \\ 0 \text{ if } h \text{ classifies } \boldsymbol{x} \text{ as negative} \end{cases}$$



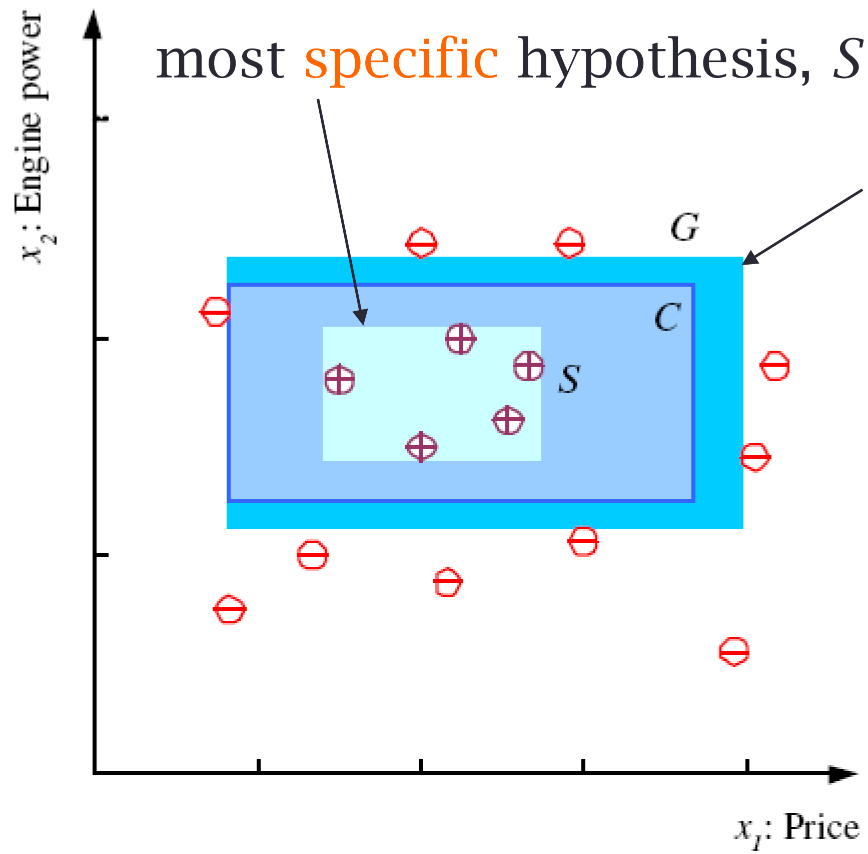
Empirical error:

$$E(h \mid \mathcal{X}) = \sum_{t=1}^{N} 1\left(h\left(\mathbf{x}^t\right) \neq r^t\right)$$

S, G, and the Version Space

most specific hypothesis, S

most general hypothesis, G



$h \in \mathcal{H}$, between S and G is consistent and make up the version space

(Mitchell, 1997)

Probably Approximately Correct (PAC) Learning

- How many training examples are needed so that the tightest rectangle S which will constitute our hypothesis, will probably be approximately correct?
  - We want to be *confident* (*above a level*) that $1-\delta$
  - … the *error probability is bounded* by some value ε

- A concept class *C* is called ***PAC-learnable*** if there exists a PAC-learning algorithm such that, for any $\varepsilon>0$ and $\delta>0$, there exists a fixed sample size such that, for any concept $c \in C$ and for any probability distribution on X, the learning algorithm produces a probably-approximately-correct hypothesis *h*
- a (PAC) *probably-approximately-correct hypothesis h* is one that has error at most ε with probability at least $1-\delta$.

Probably Approximately Correct (PAC) Learning

- In PAC learning, given a class *C* and examples drawn from some unknown but fixed distribution *p(x)*, we want to find the number of examples *N*, such that with probability at least $1 - \delta$, *h* has error at most ε ?

  (Blumer et al., 1989)

$$P(\, C \Delta h \leq \varepsilon \,) \geq 1 - \delta$$

where $C \Delta h$ is (area of the) "the region of difference between *C* and *h*", and $\delta > 0$, $\varepsilon > 0$.

PAC Learning

- How many training examples *m* should we have, such that with probability at least $1 - \delta$, h has error at most ε ?

(Blumer et al., 1989)

- Let prob. of a + ex. in each strip be at most $\varepsilon/4$
- Pr that a random ex. misses a strip: $1- \varepsilon/4$
- Pr that *m* random instances miss a strip:

$$(1 - \varepsilon/4)^m$$

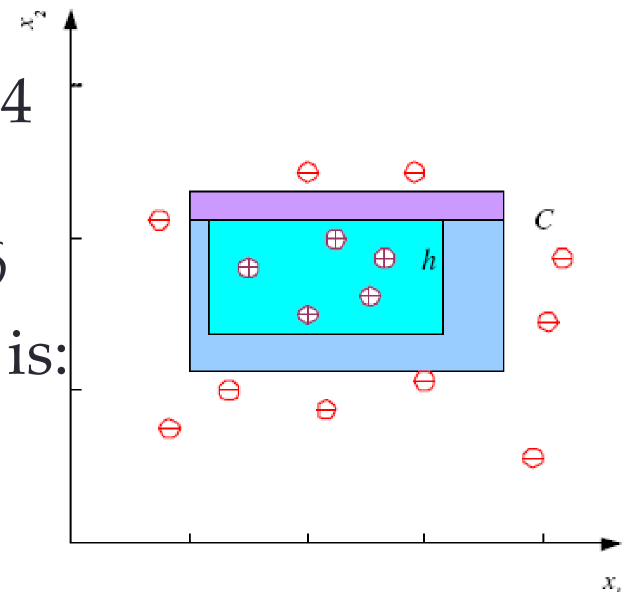
- Pr that *m* random instances instances miss 4 strips: $4(1 - \varepsilon/4)^m$
- We want $1-4(1 - \varepsilon/4)^m \geq 1- \delta$ or $4(1 - \varepsilon/4)^m \leq \delta$
- Using $1-x \leq e^{-x}$ an even stronger condition is:

$[(1-\varepsilon/4) \leq exp(-\varepsilon/4)$ so $(1-\varepsilon/4)^m \leq exp(-\varepsilon/4)^m = exp(-\varepsilon m/4)]$

$$4e^{-\varepsilon m/4} \leq \delta \qquad \text{OR}$$

- Divide by 4, take *ln*... and show that

$$m \geq (4/\varepsilon)ln(4/\delta)$$



Probably Approximately Correct (PAC) Learning

How many training examples m should we have, such that with probability at least 1 - δ, h has error at most ε ?

(Blumer et al., 1989)

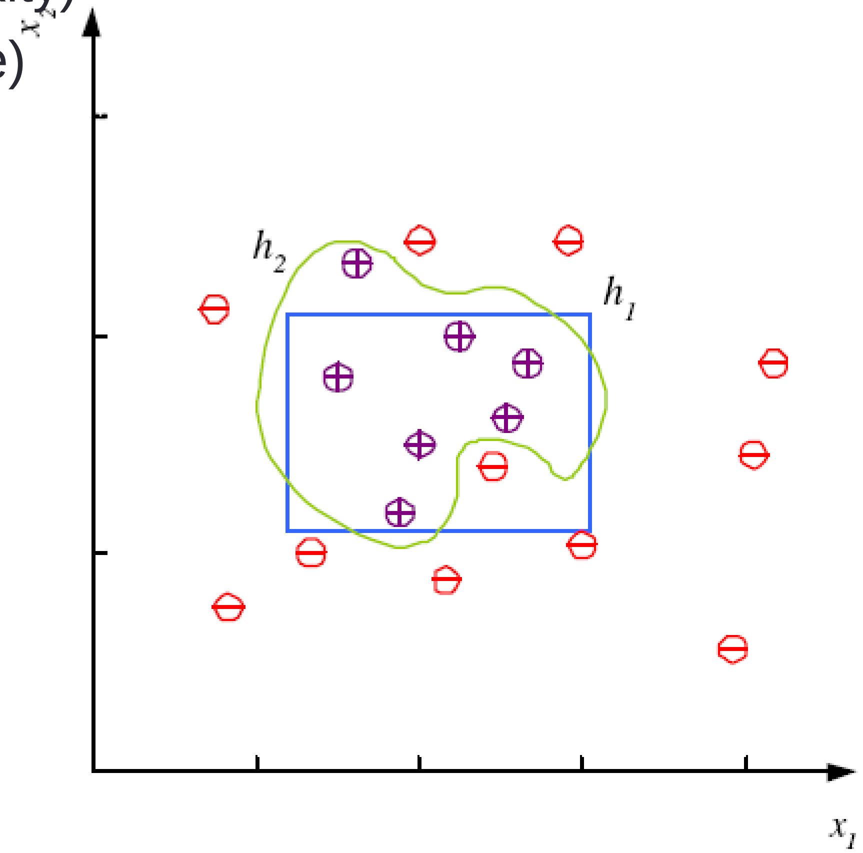
$$m \geq (4/\varepsilon) ln(4/\delta)$$

- m increases slowly with $1/\varepsilon$ and $1/\delta$
- Say ε=1% with confidence 95%, pick $m \geq 1752$
- Say ε=10% with confidence 95%, pick $m \geq 175$

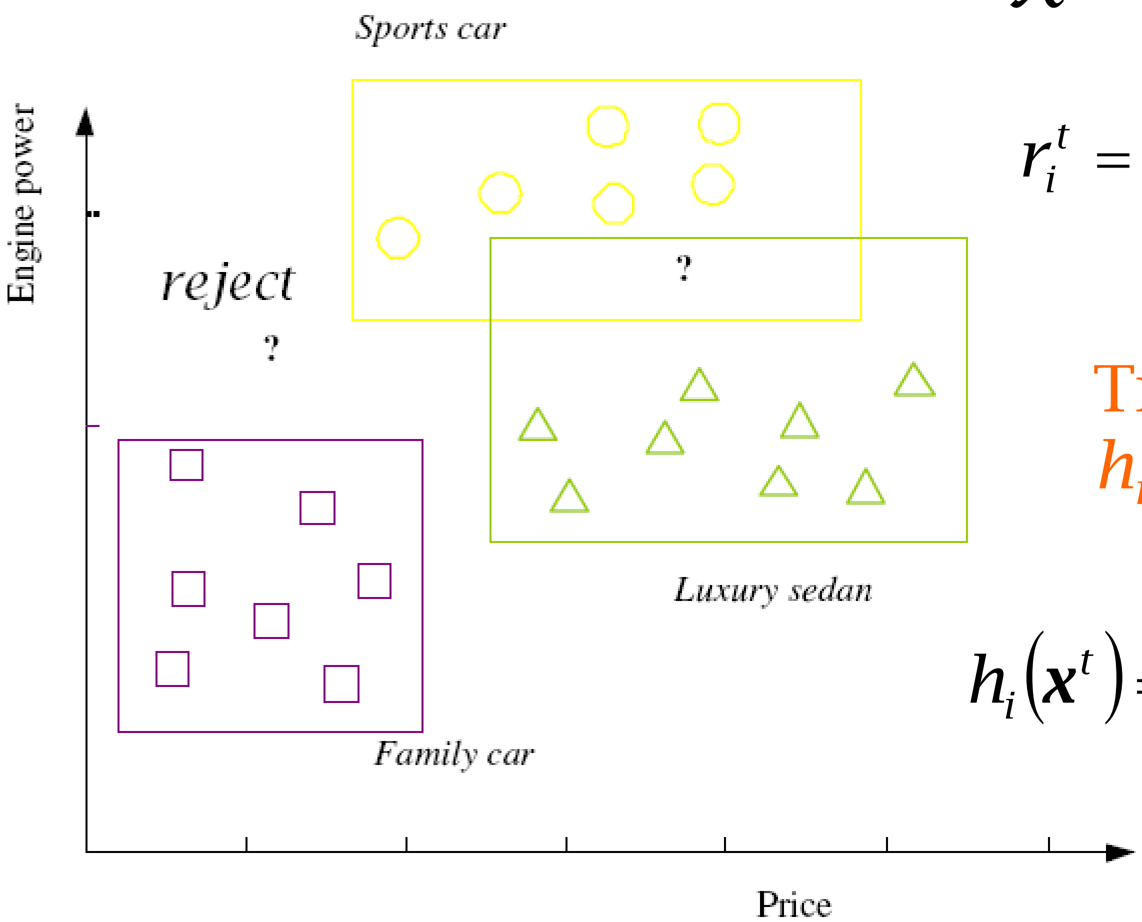
Model Complexity vs. Noise

Use the simpler one because

- Simpler to use (lower computational complexity)
- Easier to train (lower space complexity)
- Easier to explain (more interpretable)
- Generalizes better
  (lower variance - Occam's razor)



Multiple Classes, C_i i=1,...,K



$$\mathcal{X} = \{\boldsymbol{x}^t, r^t\}_{t=1}^{N}$$

$$r_i^t = \begin{cases} 1 \text{ if } \boldsymbol{x}^t \in C_i \\ 0 \text{ if } \boldsymbol{x}^t \in C_j, j \neq i \end{cases}$$

Train hypotheses $h_i(\boldsymbol{x})$, $i = 1,...,K$:

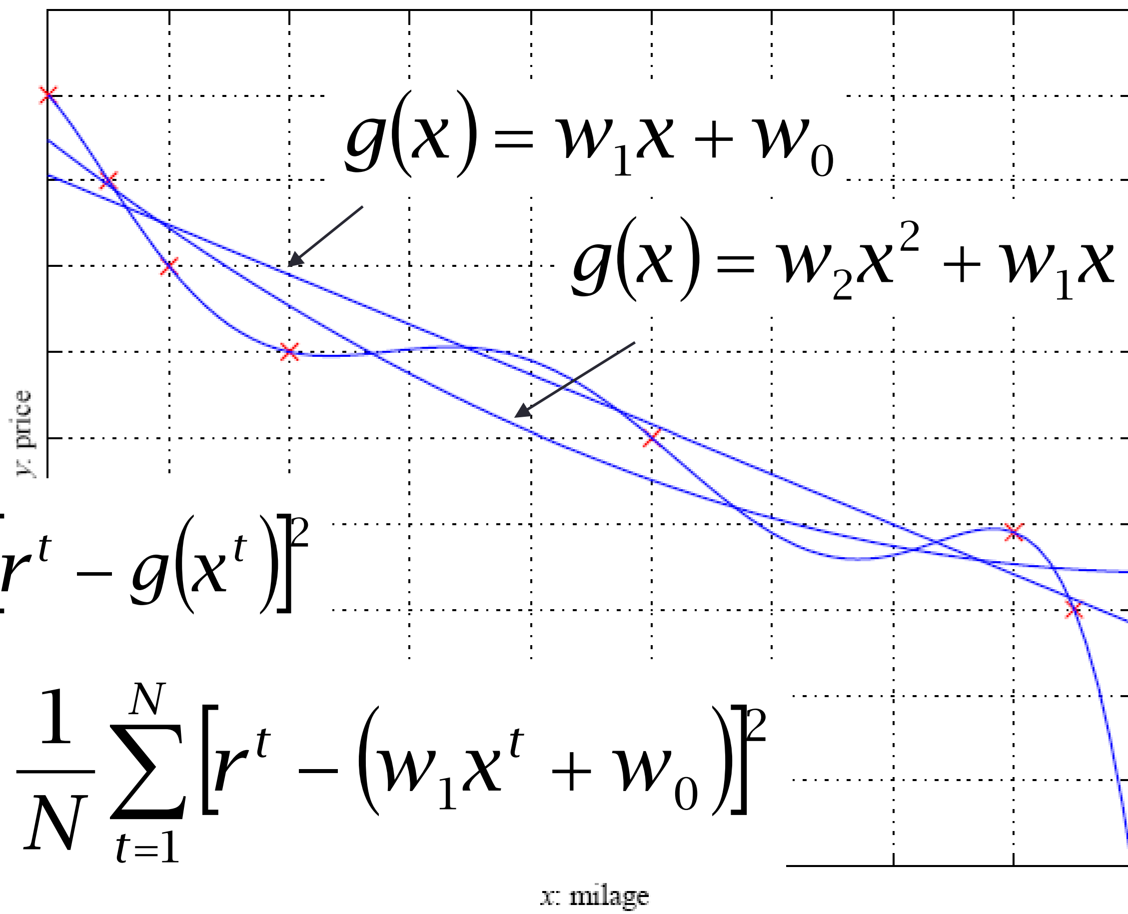
$$h_i(\boldsymbol{x}^t) = \begin{cases} 1 \text{ if } \boldsymbol{x}^t \in C_i \\ 0 \text{ if } \boldsymbol{x}^t \in C_j, j \neq i \end{cases}$$

Regression

$$\mathcal{X} = \left\{x^t, r^t\right\}_{t=1}^{N}$$

$$r^t \in \Re$$

$$r^t = f\left(x^t\right) + \varepsilon$$



$$E(g \mid \mathcal{X}) = \frac{1}{N}\sum_{t=1}^{N}\left[r^t - g\left(x^t\right)\right]^2$$

$$E(w_1, w_0 \mid \mathcal{X}) = \frac{1}{N}\sum_{t=1}^{N}\left[r^t - \left(w_1 x^t + w_0\right)\right]^2$$

VC (Vapnik-Chervonenkis) Dimension

- N points can be labeled in 2^N ways as +/–
- $\mathcal{H}$ shatters N if there exists a set of N points such that $h \in \mathcal{H}$ is consistent with all of these possible labels:
  - Denoted as: $VC(\mathcal{H}) = N$
  - Measures the capacity of H
- Any learning problem definable by N examples can be learned with no error by a hypothesis drawn from H

What is the VC dimension of axis-aligned rectangles?

Formal Definition

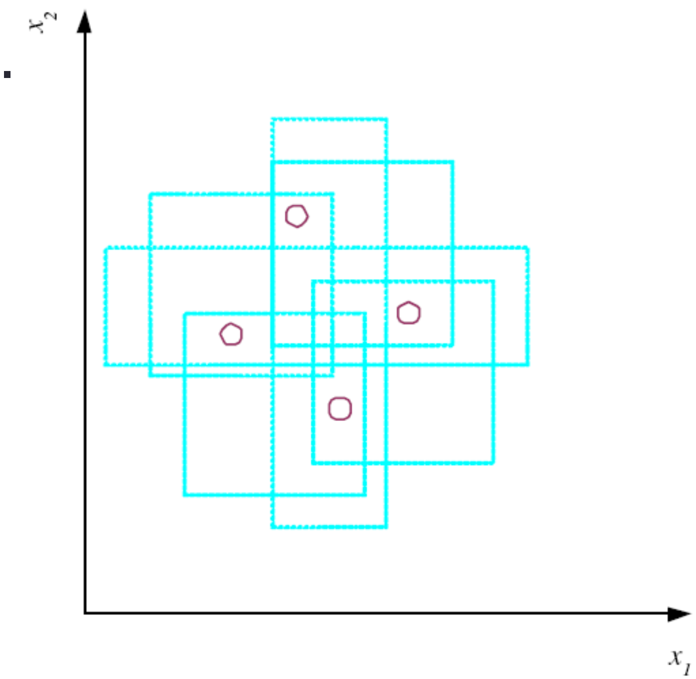
The VC Dimension

Definition: the VC dimension of a set of functions $H = \{h(\mathbf{x}, \alpha)\}$ is d if and only if there exists a set of points $\{x^i\}_{i=1}^{d}$ such that these points can be labeled in all 2^d possible configurations, and for each labeling, a member of set H can be found which correctly assigns those labels, but that no set $\{x^i\}_{i=1}^{q}$ exists where $q > d$ satisfying this property.

VC (Vapnik-Chervonenkis) Dimension

- $\mathcal{H}$ shatters N if there exists N points and $h \in \mathcal{H}$ such that h is consistent for any labelings of those N points.

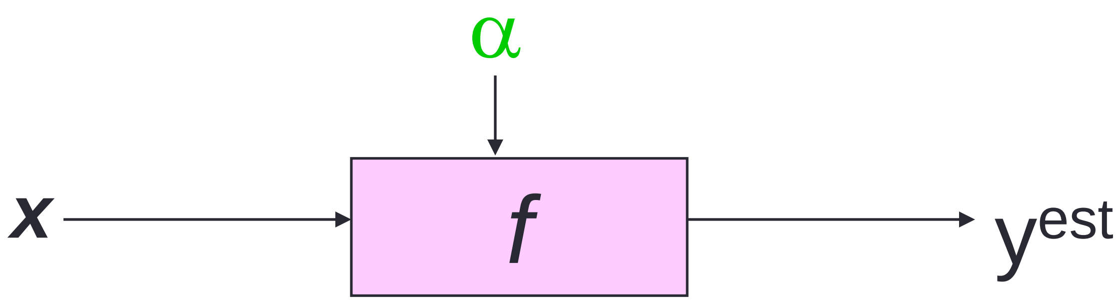
- VC(*axis aligned rectangles*) = 4



VC (Vapnik-Chervonenkis) Dimension

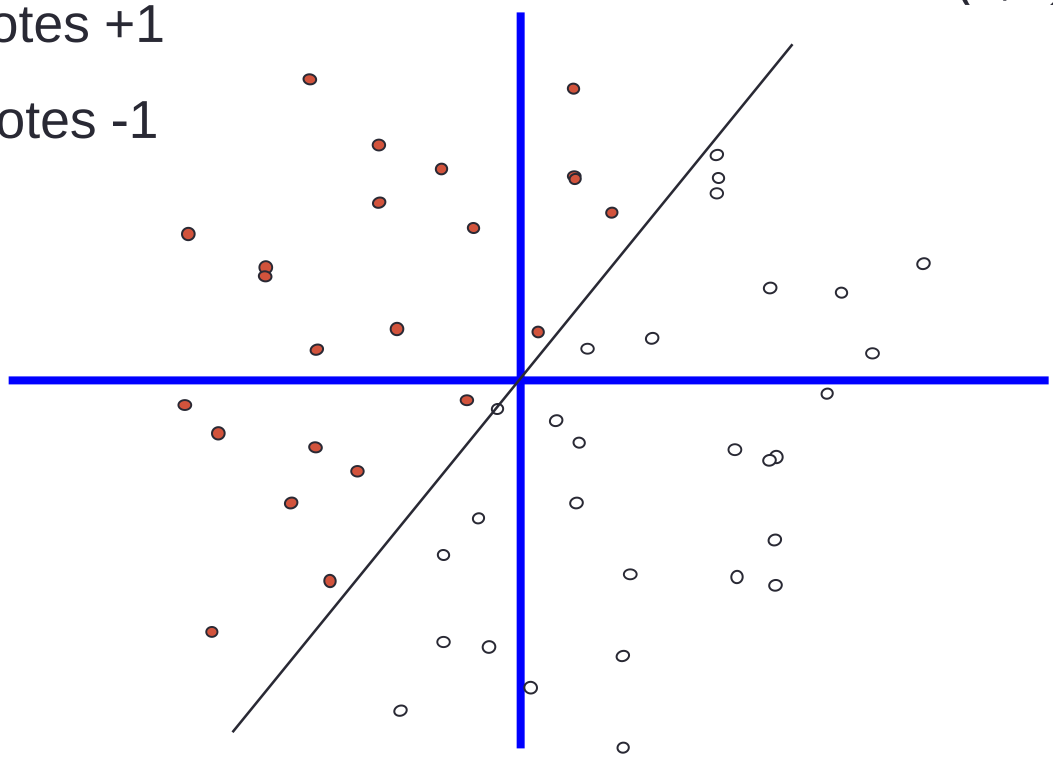
- *What does this say about using rectangles as our hypothesis class?*
- VC dimension is pessimistic: in general we do not need to worry about all possible labelings
- It is important to remember that one can choose the arrangement of points in the space, but then the hypothesis must be consistent with all possible labelings of those fixed points.

Examples

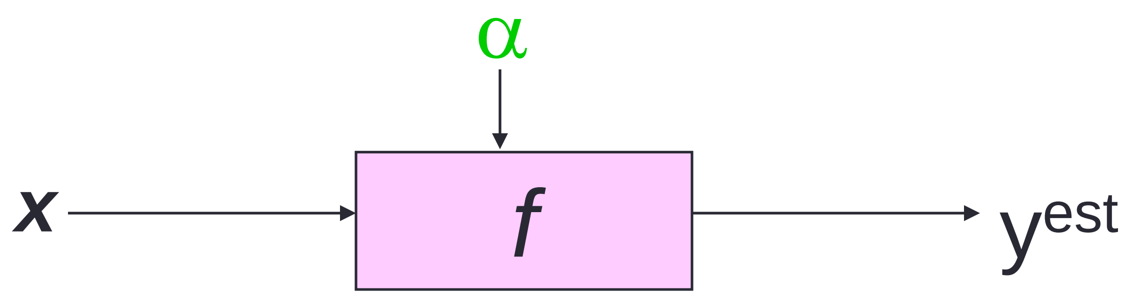


$f(x,w) = sign(x.w)$

- denotes +1
- denotes -1

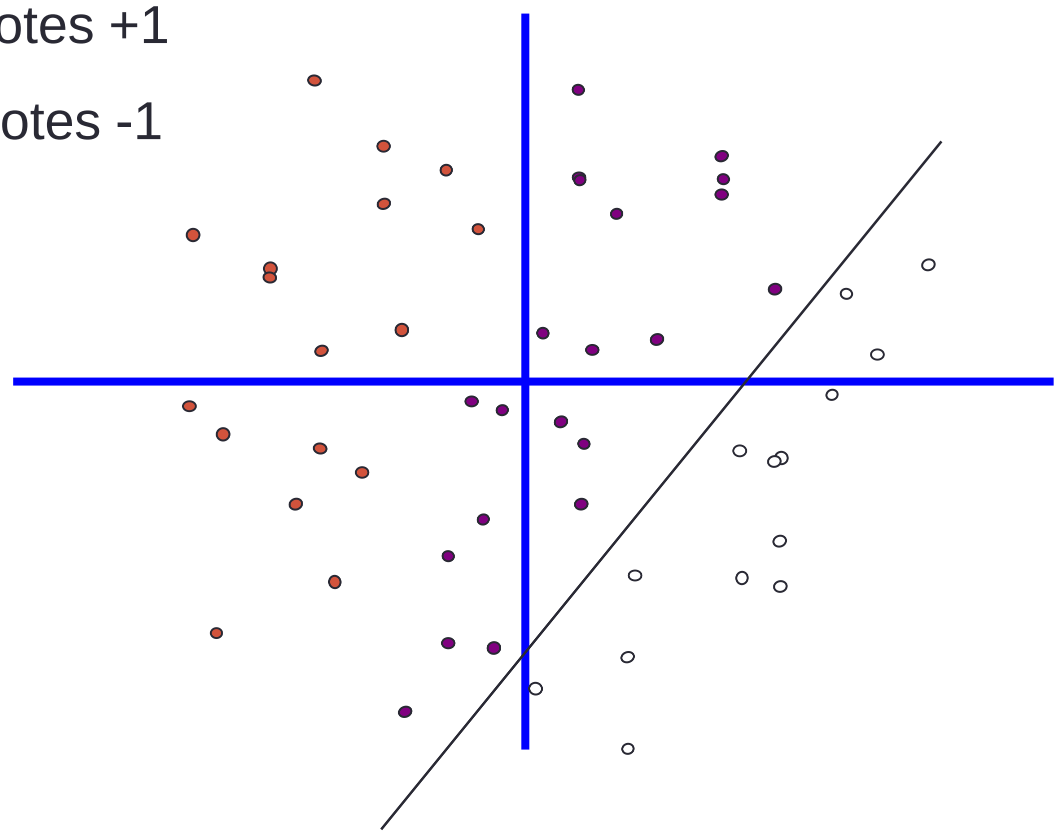


Examples



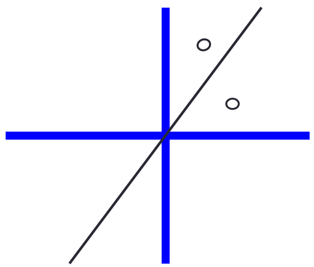
f(x,w,b) = sign(x.w+b)

- denotes +1
- denotes -1



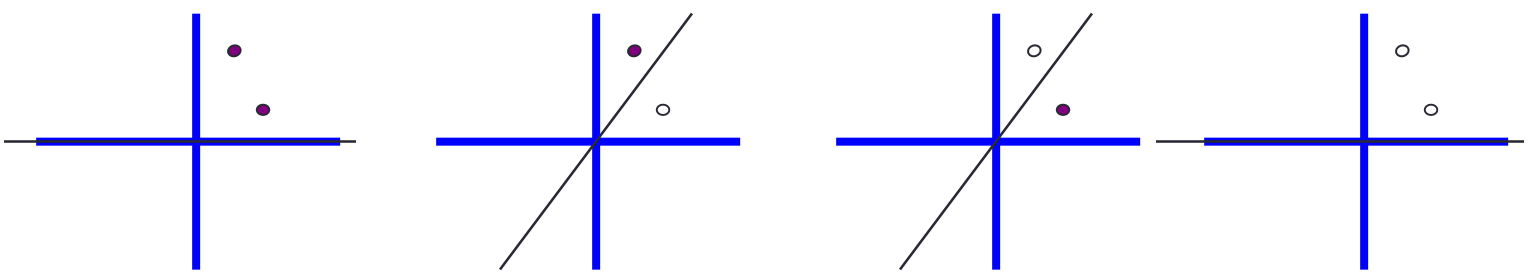
Shattering

- Question: Can the following *f* shatter the following points?



$f(x,w) = sign(x.w)$

- Answer: Yes. There are four possible training set types to consider:



w=(0,1) w=(-2,3) w=(2,-3) w=(0,-1)

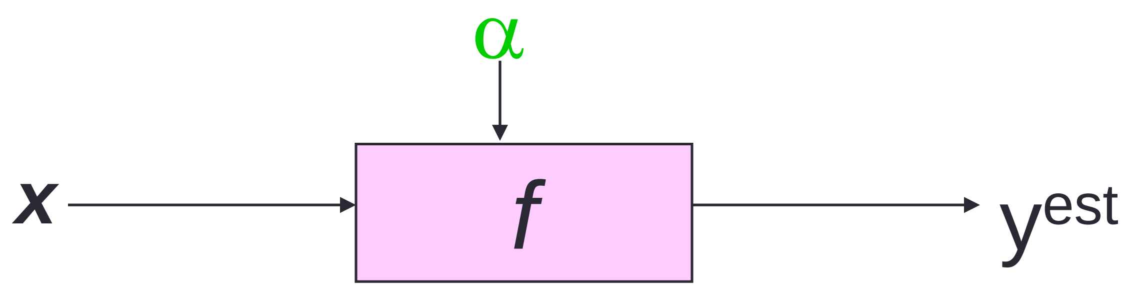
VC dim of linear classifiers in m-dimensions

If input space is *m-dimensional* and if **f** is *sign*($\mathbf{w}.\mathbf{x}-b$), what is the VC-dimension?

$$h=m+1$$

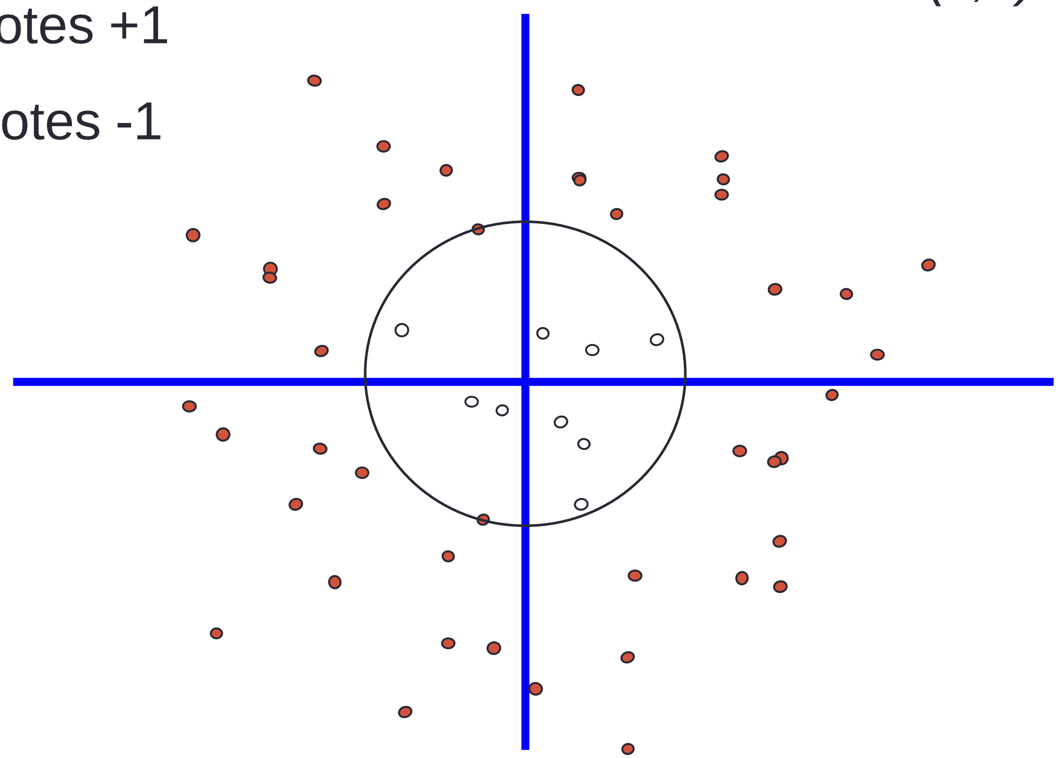
- Lines in 2D can shatter 3 points
- Planes in 3D space can shatter 4 points
- ...

Examples



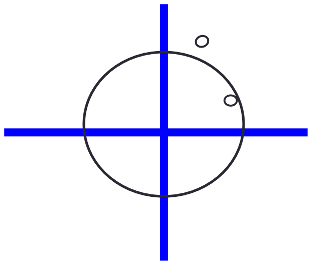
$f(x,b) = sign(x.x - b)$

- denotes +1
- denotes -1



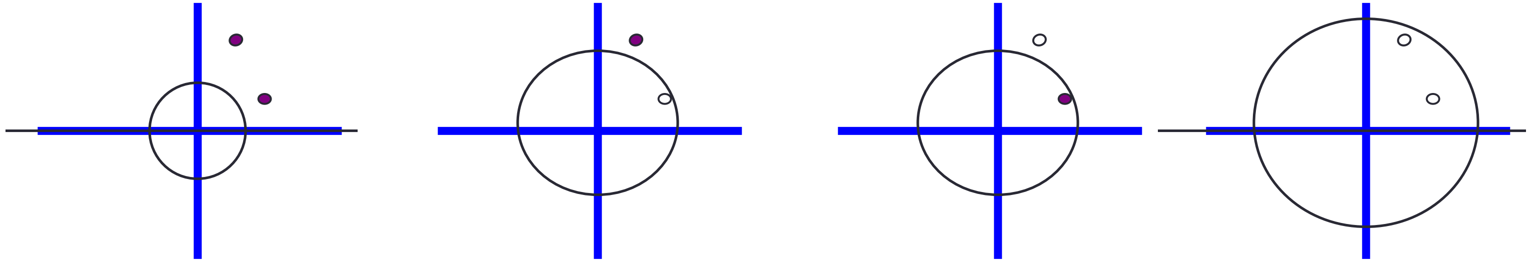
Shattering

- Question: Can the following *f* shatter the following points?



f(x,b) = sign(x.x-b)

Answer: Yes. Hence, the VC dimension of circles on the origin is at least 2.



- Note that if we pick two points at the same distance to the origin, they cannot be shattered. But we are interested if all possible labellings of some n-points can be shattered.

- How about 3 for circles on the origin (Can you find 3 points such that all possible labellings can be shattered?)?

Model Selection & Generalization

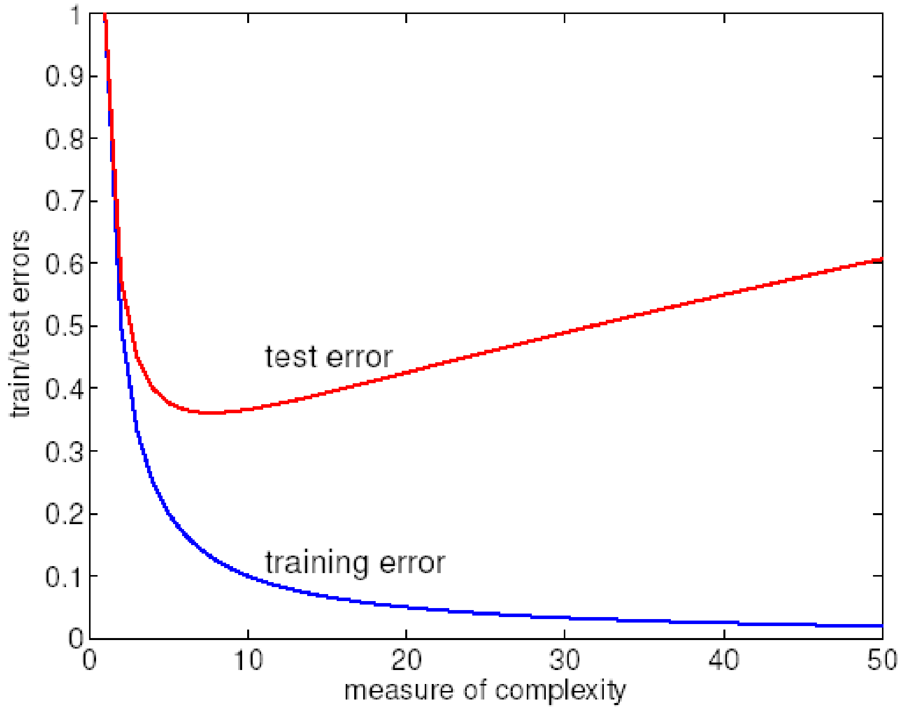
- Learning is an ill-posed problem; data is not sufficient to find a unique solution
- The need for inductive bias, assumptions about $\mathcal{H}$
- **Generalization:** How well a model performs on new data
- Different machines have different amounts of "power".
  Tradeoff between:
  - More power: Can model more complex classifiers but might overfit.
  - Less power: Not going to overfit, but restricted in what it can model.
  - **Overfitting**: $\mathcal{H}$ more complex than C or f
  - **Underfitting**: $\mathcal{H}$ less complex than C or f

Triple Trade-Off

- There is a trade-off between three factors (Dietterich, 2003):
  1. Complexity of $\mathcal{H}$, $c(\mathcal{H})$,
  2. Training set size, N,
  3. Generalization error, E, on new data

- As $N\uparrow$, $E\downarrow$
- As $c(\mathcal{H})\uparrow$, first $E\downarrow$ and then $E\uparrow$

Why Care about Complexity?



- A quantitative measure of complexity is useful to determine the relationship between thetraining error (that we can observe during training) and the test error (which we want to minimize)

Complexity

- "Complexity" is a measure of a family of classifiers, not of any specific (fixed) classifier

- There are many possible measures for complexity
  - degrees of freedom (e.g. number of parameters in polinomials)
  - description length
  - Vapnik-Chervonenkis (VC) dimension
  - etc.

Expected and Empirical error

$$\hat{\mathcal{E}}_n(i) = \frac{1}{n}\sum_{t=1}^{n} \overbrace{\mathsf{Loss}(y_t, h_i(\mathbf{x}_t))}^{=0,1} = \text{empirical error of } h_i(\mathbf{x})$$

$$\mathcal{E}(i) = E_{(\mathbf{x},y)\sim P}\{\, \mathsf{Loss}(y, h_i(\mathbf{x}))\,\} = \text{expected error of } h_i(\mathbf{x})$$

Learning and the VC dimension

- Let d_{VC} be the VC-dimension of our set of classifiers F.

  Theorem: With probability at least $1 - \delta$ over the choice of the training set, for all $h \in F$

$$\mathcal{E}(h) \leq \hat{\mathcal{E}}_n(h) + \epsilon(n, d_{VC}, \delta)$$

Expected (or Test) Error

Empirical Error

where

$$\epsilon(n, d_{VC}, \delta) = \sqrt{\frac{d_{VC}(\log(2n/d_{VC}) + 1) + \log(1/(4\delta))}{n}}$$

Model Selection

- We try to find the model with the best balance of complexity and the fit to the training data
- Ideally, we would select a model from a nested sequence of models of increasing complexity (VC-dimension)

  Model 1 d_1
  Model 2 d_2
  Model 3 d_3

  where $d_1 \leq d_2 \leq d_3 \leq \ldots$
- The model selection criterion is: find the model class that achieves the lowest upper *bound* on the expected loss

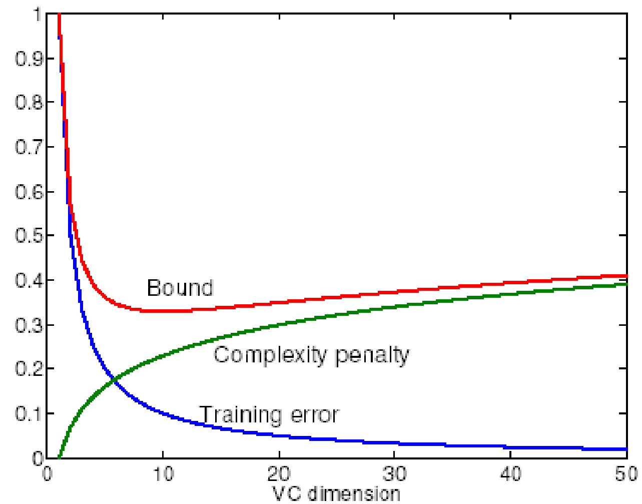
  Expected error $\leq$ Training error $+$ Complexity penalty

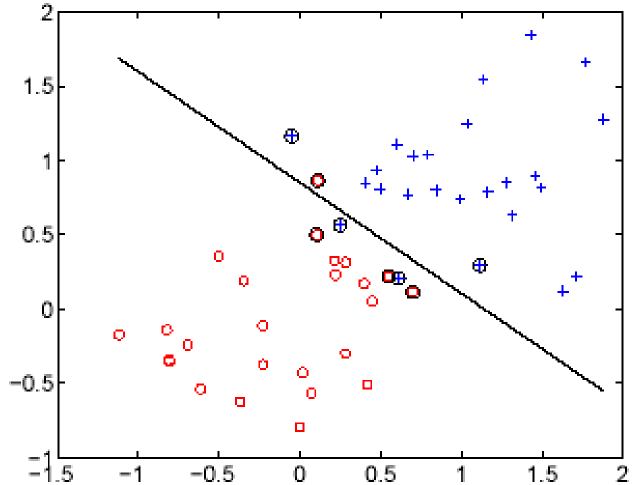
VC dimension and Structural Risk Minimization

- We choose the model class F_i that minimizes the upper bound on the expected error:

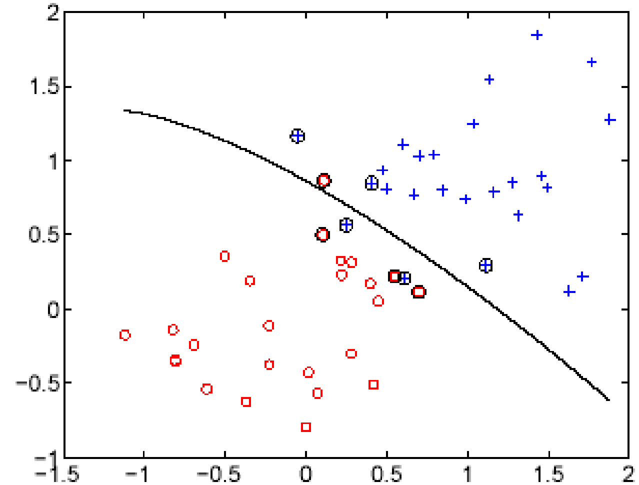
$$\mathcal{E}(\hat{h}_i) \leq \hat{\mathcal{E}}_n(\hat{h}_i) + \sqrt{\frac{d_i(\log(2n/d_i)+1)+\log(1/(4\delta))}{n}}$$

where $\hat{h}_i$ is the best classifier from F_i selected on the basis of the training set.

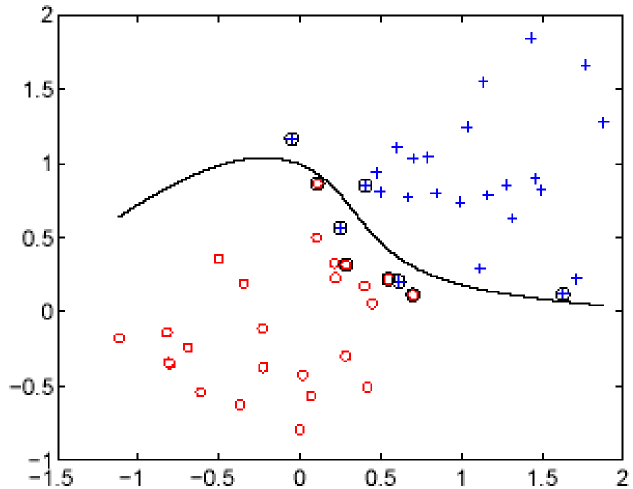




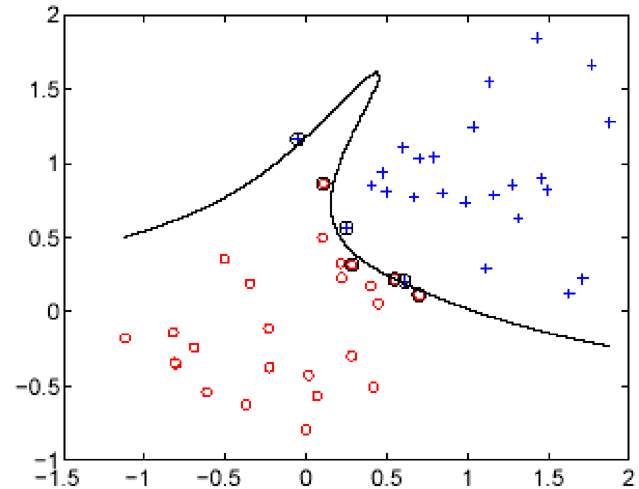
linear



2^{nd} order polynomial



4^{th} order polynomial



8^{th} order polynomial

Structural Risk Minimization

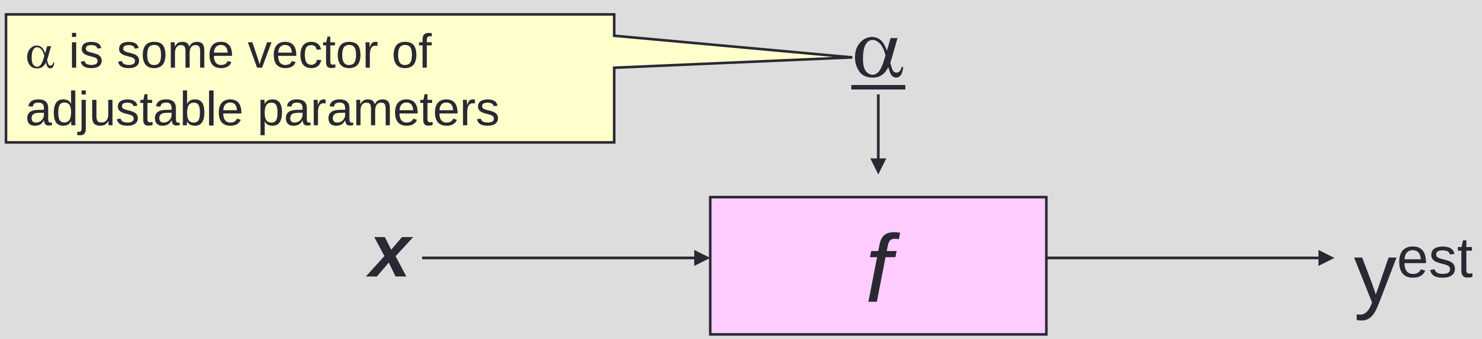
- Number of training examples $n = 50$, confidence parameter $\delta = 0.05$.

Model	d_{VC}	Empirical fit	$\epsilon(n, d_{VC}, \delta)$
1^{st} order	3	0.06	0.5501
2^{nd} order	6	0.06	0.6999
4^{th} order	15	0.04	0.9494
8^{th} order	45	0.02	1.2849

- Structural risk minimization would select the simplest (linear) model in this case.

Summary: a learning machine

- A learning machine ***f*** takes an input ***x*** and transforms it, somehow using factors (as weights) $\underline{\alpha}$, into a predicted output y^{est} = +/- 1



Back to test and empirical (training) error

- Given some machine **f**
- Define:

$$R(\vec{\alpha}) = \text{TESTERR}(\vec{\alpha}) = E\left[\frac{1}{2}\left|y - f(x,\vec{\alpha})\right|\right] = \begin{array}{c}\text{Probability of}\\ \text{Misclassification}\end{array}$$

$$R^{emp}(\vec{\alpha}) = \text{TRAINERR}(\vec{\alpha}) = \frac{1}{R}\sum_{k=1}^{R}\frac{1}{2}\left|y_k - f(x_k,\vec{\alpha})\right| = \begin{array}{c}\text{Fraction Training}\\ \text{Set misclassified}\end{array}$$

R = #training set data points

Vapnik-Chervonenkis dimension

$$\mathrm{TESTERR}(\vec{\alpha}) = E\left[\frac{1}{2}\left|y - f(x,\vec{\alpha})\right|\right] \qquad \mathrm{TRAINERR}(\vec{\alpha}) = \frac{1}{R}\sum_{k=1}^{R}\frac{1}{2}\left|y_k - f(x_k,\vec{\alpha})\right|$$

- Given some machine **f**, let *h* be its VC dimension (*h* does not depend on the choice of training set)
- **Let R be the number of training examples**
- Vapnik showed that with probability 1-η

$$\mathrm{TESTERR}(\vec{\alpha}) \le \mathrm{TRAINERR}(\vec{\alpha}) + \sqrt{\frac{h(\log(2R/h)+1)-\log(\eta/4)}{R}}$$

This gives us a way to estimate the error on future data based only on the training error and the VC-dimension of ***f***

VC-dimension as measure of complexity

$$\mathrm{TESTERR}(\vec{\alpha}) \leq \mathrm{TRAINERR}(\vec{\alpha}) + \sqrt{\frac{h(\log(2R/h)+1)-\log(\eta/4)}{R}}$$

i	f_i	TRAINERR	VC-Conf	Probable upper bound on TESTERR	Choice
1	f_1				
2	f_2				
3	f_3				⌧
4	f_4				
5	f_5				
6	f_6				

Using VC-dimensionality

People have worked hard to find VC-dimension for ...

- Decision Trees
- Perceptrons
- Neural Nets
- Decision Lists
- Support Vector Machines
- …and many many more

All with the goals of

1. Understanding which learning machines are more or less powerful under which circumstances
2. Using Structural Risk Minimization for to choose the best learning machine

Alternatives to VC-dim-based model selection

Cross Validation

- To estimate generalization error, we need data unseen during training. Given any two models, M_1 M_2, we split the data as:
  - Training set (50%) train(M_2) < train(M_1)
  - Validation set (25%) $test_{Vs}(M_1)=P_1$ $test_{Vs}(M_2)=P_2$ $P_2>P_1$
  - Test (publication) set (25%)

- Resampling when there is few data
  - N-fold cross-validation: N-2 fold for training, 1 fold as validation set and 1 fold for testing (N*(N-1) tests)

Alternatives to VC-dim-based model selection

i	f_i	TRAINERR	10-FOLD-CV-ERR	Choice
1	f_1			
2	f_2			
3	f_3			⌧
4	f_4			
5	f_5			
6	f_6			

Extra Comments

- An excellent tutorial on VC-dimension and Support Vector Machines

C.J.C. Burges. A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery, 2(2):955-974, 1998.

What you should know

- Definition of PAC learning
- The definition of a learning machine: $\boldsymbol{f}(\boldsymbol{x},\alpha)$
- The definition of Shattering
- Be able to work through simple examples of shattering
- The definition of VC-dimension
- Be able to work through simple examples of VC-dimension
- Structural Risk Minimization for model selection
- Awareness of other model selection methods