FUNDAMENTALS OF MACHINE LEARNING

SUPERVISED LEARNING

CSCI3320

Prof. John C.S. Lui, CSE Department, CUHK
Introduction to Machine Learning

Objectives

- Be familiar about notations of ML
- How many sample points we need to guarantee high accuracy ?
- Underfitting and overfitting (VC dimension)

Learning a Class from Examples

- Class C of a "family car"
- We can apply this learning in two ways
  - **Prediction:** when a new point x comes in, is x a family car?
  - **Knowledge extraction**: What do people expect from a family car?
- **Output:**

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

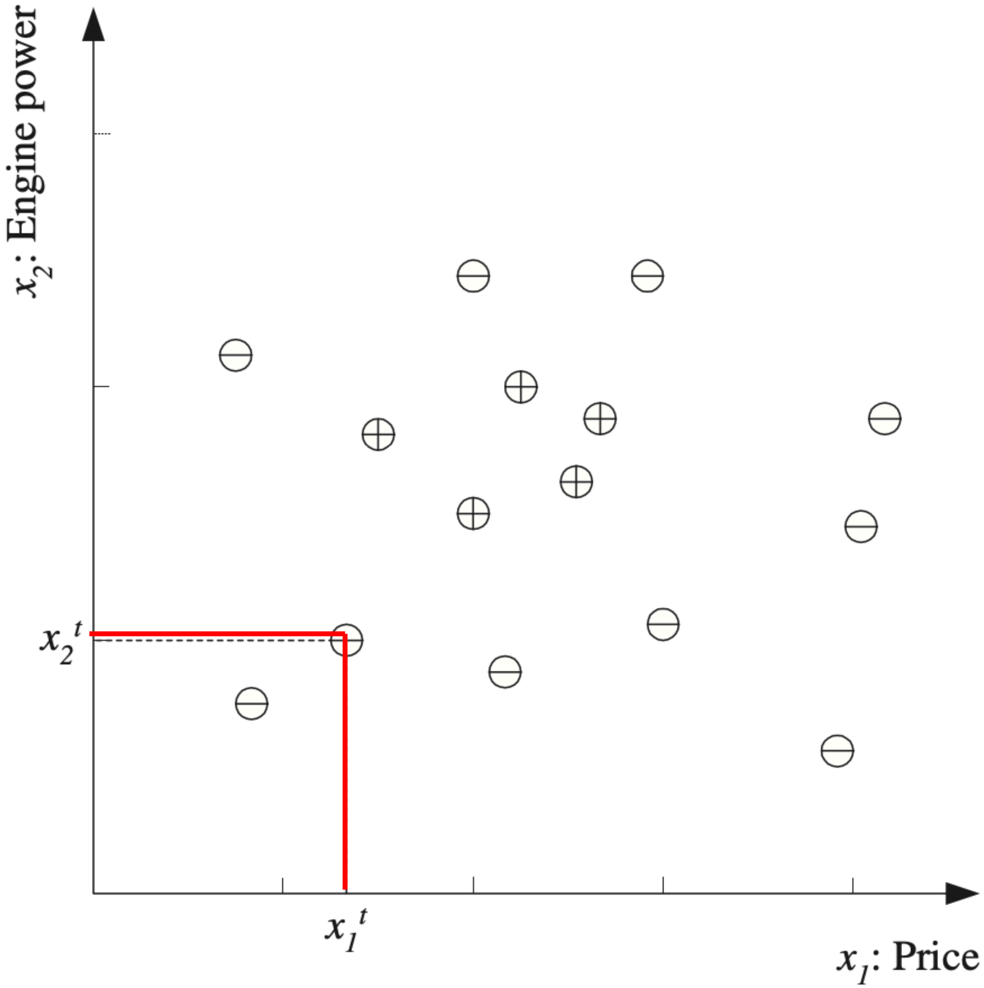
  x_1: price

  x_2 : engine power

  r : label data (+ or –, 1 or 0)

Training set χ

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad r = \begin{cases} 1 & \text{if } \boldsymbol{x} \text{ is a positive example} \\ 0 & \text{if } \boldsymbol{x} \text{ is a negative example} \end{cases}$$



Training set

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

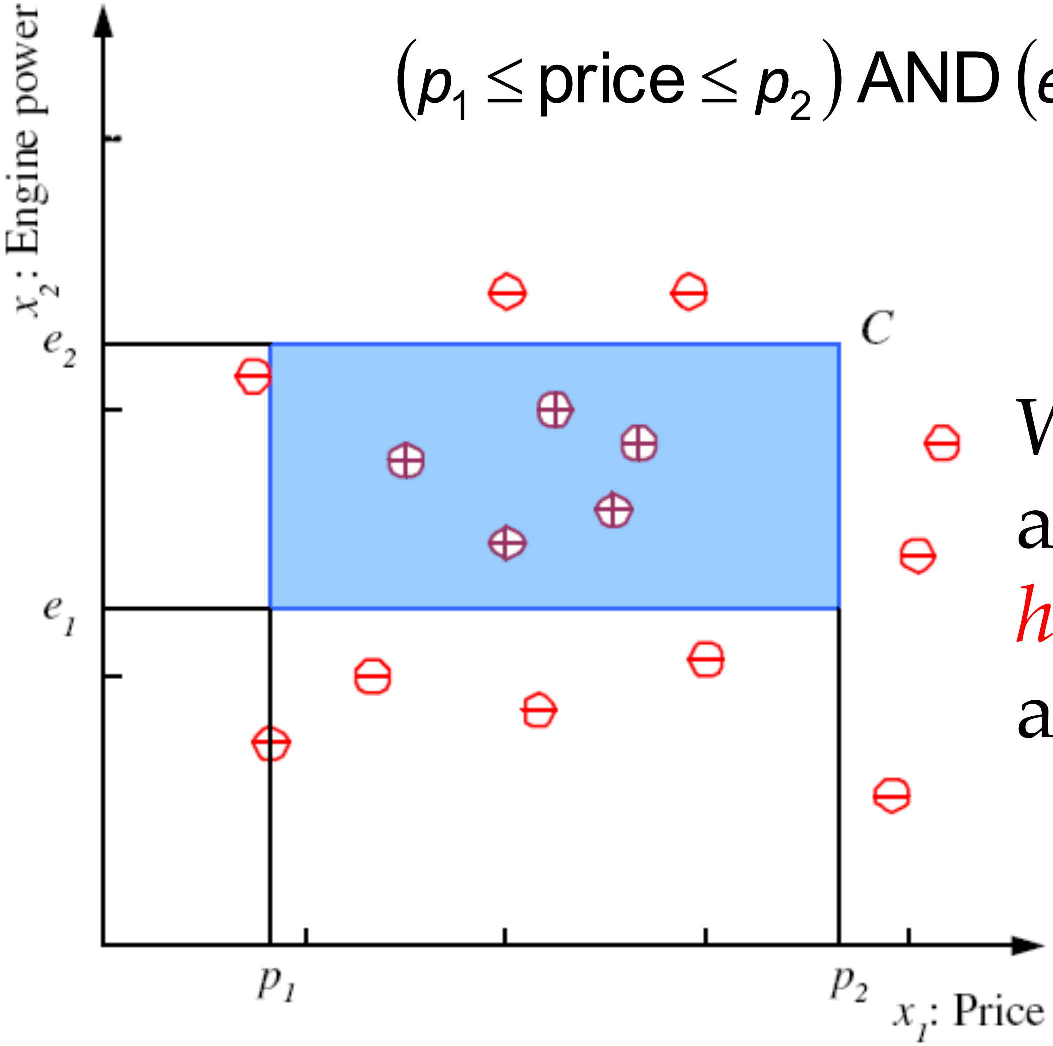
Fundamental Questions

- How many samples do we need to learn things accurately?
  - **Probably approximately correct** (PAC)
  - **VC Dimension** (relate learner complexity to errors)

Class C *(or target hypothesis)*

Hypothesis class $\mathcal{H}$ from which C is drawn

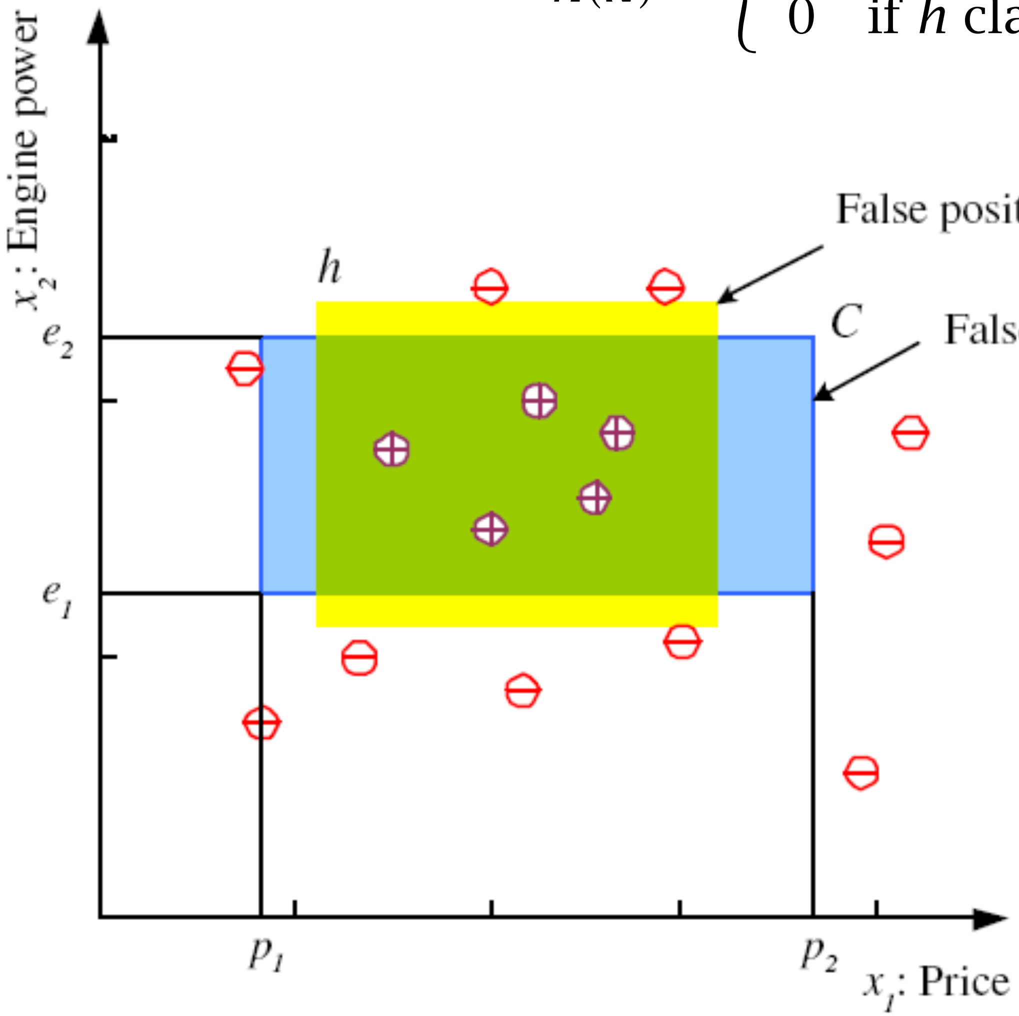
$$\left(p_1 \le \text{price} \le p_2\right) \text{AND} \left(e_1 \le \text{engine power} \le e_2\right)$$



We need to find an algorithm to find $h \in \mathcal{H}$ to approximate C as close as possible

Hypothesis class $\mathcal{H}$

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



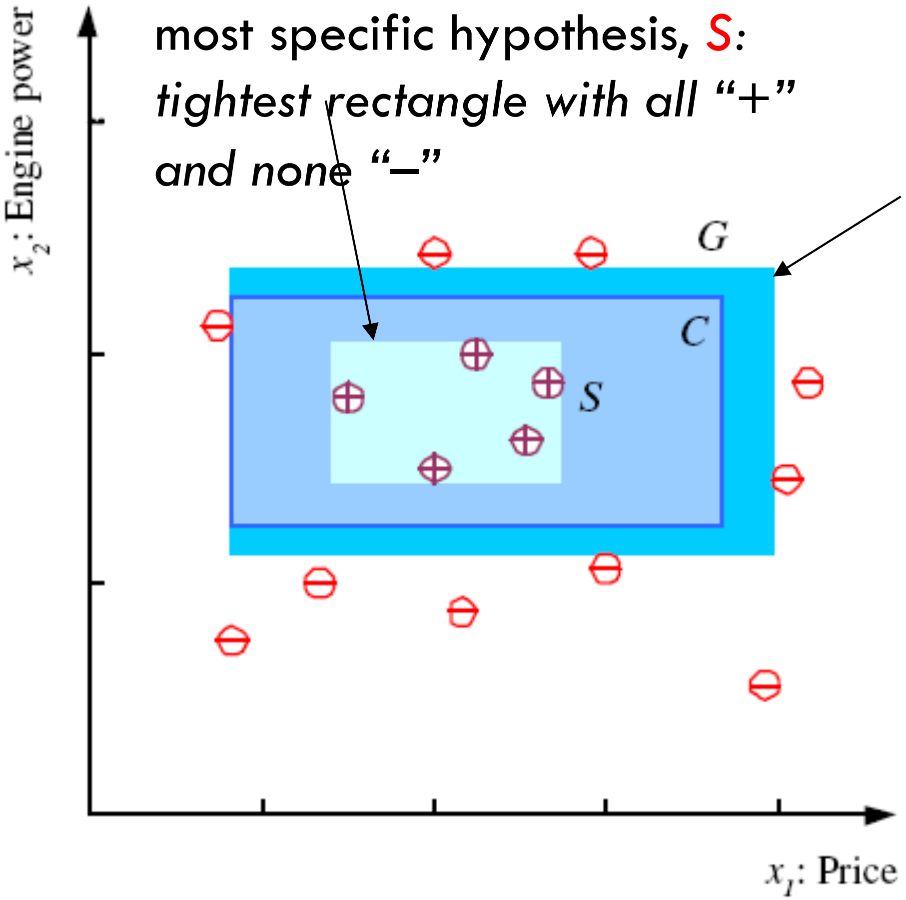
Error of h on $\mathcal{H}$

$$E(h|\mathcal{X}) = \sum_{t=1}^{N} 1(h(\boldsymbol{x}^t) \neq r^t)$$

S, G, and the Version Space

Possibilities of *h*: or the "***generalization power***"

most specific hypothesis, S:
tightest rectangle with all "+" and none "–"



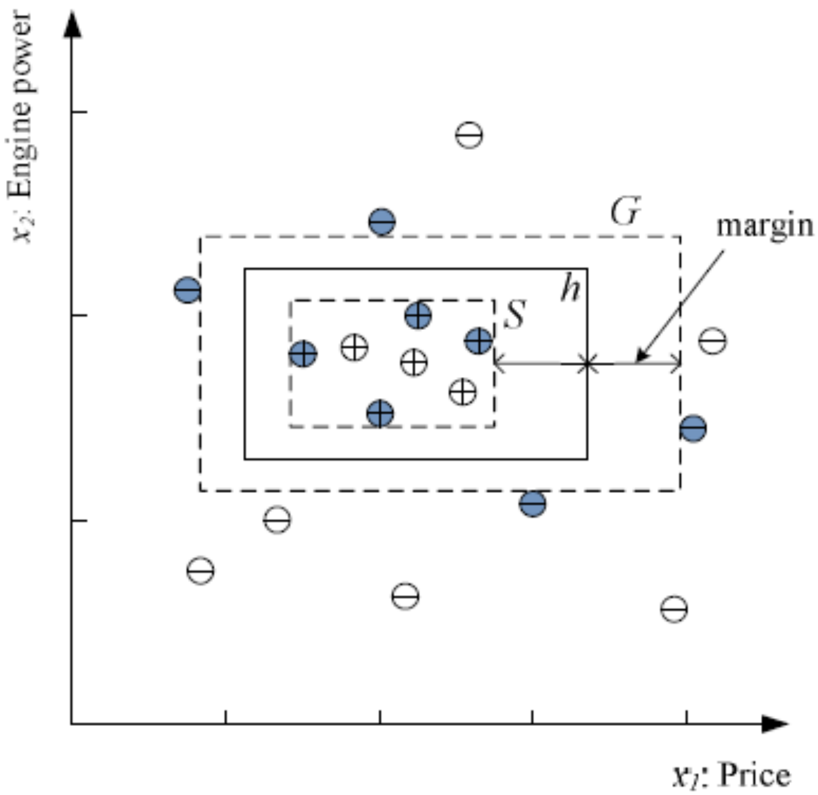
most general hypothesis, G:
largest rectangle with all "+" and none "–".

h ∈ H, between *S* and *G* is consistent and make up the version space (Mitchell, 1997)

For this example, you can come up with an algorithm to update *S* and *G* as you read in more input $\boldsymbol{x}$

Margin

- Choose *h* with largest ***margin***, which is the distance between the boundary and the instances closest to it



Probably Approximately Correct (PAC) Learning

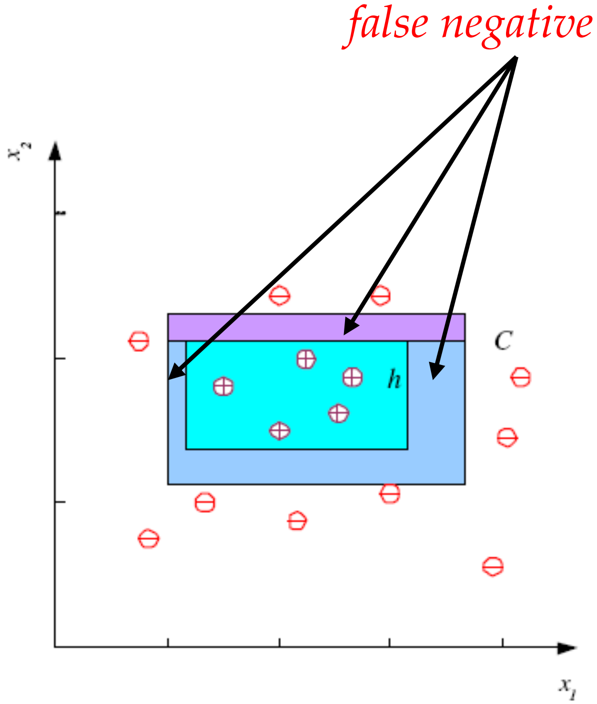
- How many training examples N should we have, such that with probability at least $1-\delta$, h **has error at most** ϵ ? (Blumer et al., 1989) *What is the physical meaning?*

  $P\{C\Delta h \leq \epsilon\} \geq 1-\delta$

  $C\Delta h$ is the region of difference between C and h.

- If all N learning sample points are in h, but the truth is C, then we have learning errors since there are some **unsampled points** in $C\Delta h$
- The above inequality means $P\{C\Delta h > \epsilon\} \leq \delta$
- $P\{C\Delta h > \epsilon\} \leq \sum_{i=1}^{4} P\{\text{error in strip}_i > \epsilon/4\}$
- Pr{one sample misses strip 1} $= 1-\epsilon/4$
- Pr{N samples miss strip 1} $= (1-\epsilon/4)^N$
- $P\{C\Delta h > \epsilon\} \leq \sum_{i=1}^{4}(1-\epsilon/4)^N = 4(1-\epsilon/4)^N$
- $P\{C\Delta h > \epsilon\} \leq \delta \Rightarrow 4\,(1-\epsilon/4)^N \leq \delta$
- Using the identity of $(1-x) \leq e^{-x}$, we have:

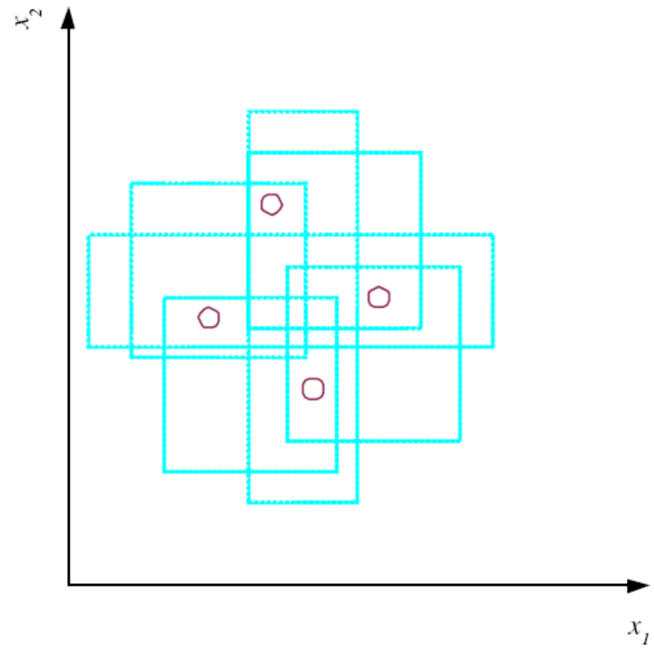
$$4\,e^{-\epsilon N/4} \leq \delta \Leftrightarrow N^* \geq \frac{4}{\epsilon}\log\frac{4}{\delta}$$



* We need at least N^* points to learn C with a high guarantee !!!
* How about for higher dimension or non-regular shape C?

Vapnik-Chervonenkis (VC) Dimension

- N points can be labeled in 2^N configurations as +/–
- If for *any* of these configurations, a hypothesis $h \in \mathcal{H}$ separates positives from negatives, then $\mathcal{H}$ **shatters** N points
- Maximum number of points that can be shattered is VC of $\mathcal{H}$
- $\mathcal{H}$ shatters N if there exists $h \in \mathcal{H}$ consistent for any of these: $\mathrm{VC}(\mathcal{H}) = N$



FUNDAMENTALS OF MACHINE LEARNING

VAPNIK-CHERVONENKIS (VC) DIMENSION

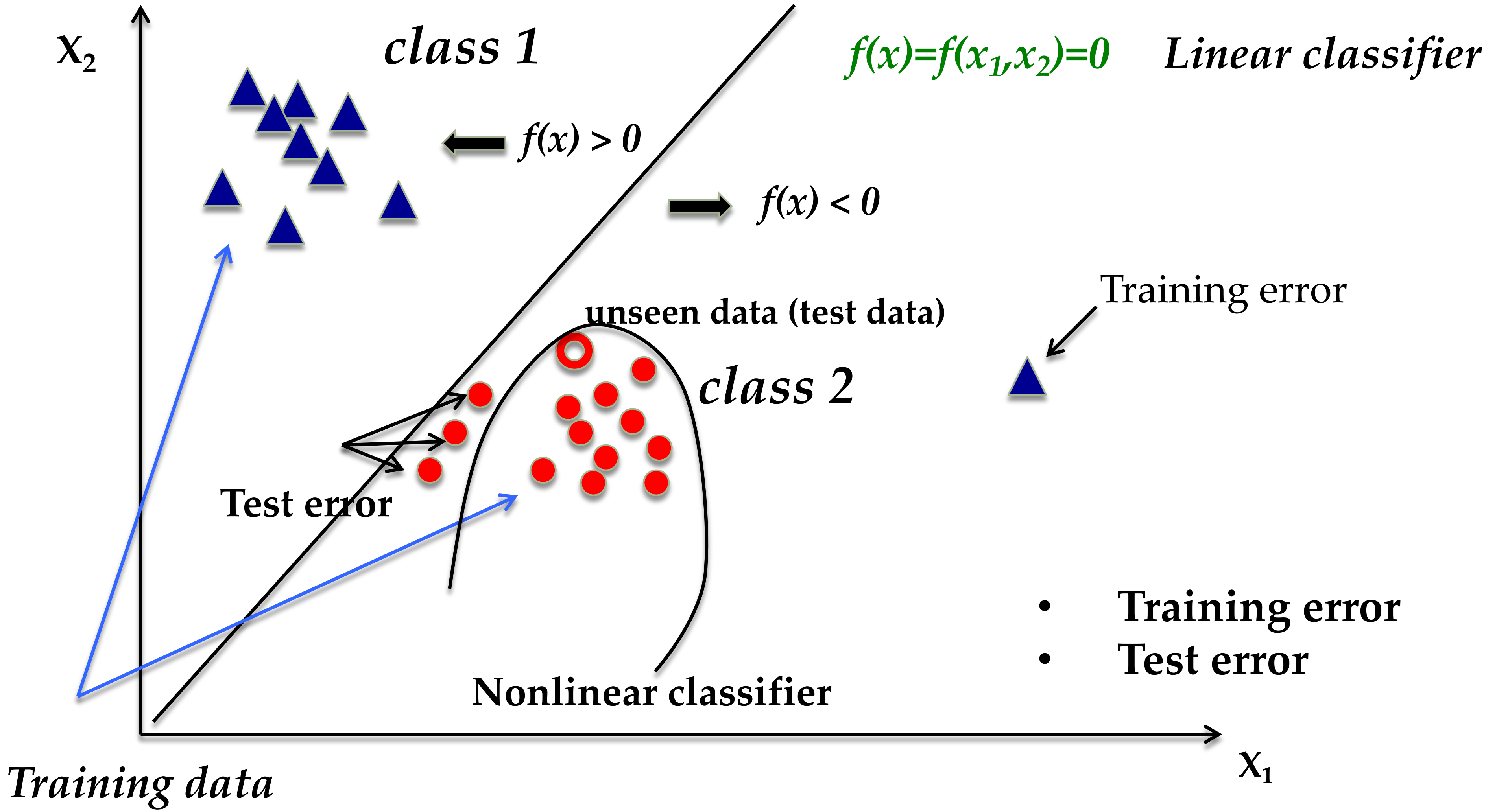
CSCI3320

Prof. John C.S. Lui, CSE Department, CUHK
Introduction to Machine Learning

VC Dimension

- VC dimension is an important concept in statistical learning theory and machine learning
- Provides performance of learning machine in terms of "***capacity***"
- **Motivation:** classification problem

Classification Problem



- Training error
- Test error

- Improve training error, not necessary improves test error
- Increase learning capacity may increase test error

Points to Remember

- Always look for **test errors** along with **training error**
- Improving on training error not always improve test error
- Increase in machine capacity may result in poor test performance

- The 2nd and 3rd point are related.
  - As we increase machine capacity, test error will first reduce, then it will increase. This is known as **over-fitting** *(good for training data but bad for testing data)*

The BIG Question

- Is there any equation that relates "**training error**" with "**test error**"?
- **Equation**: gives upper bound of test error with probability $1-\eta$

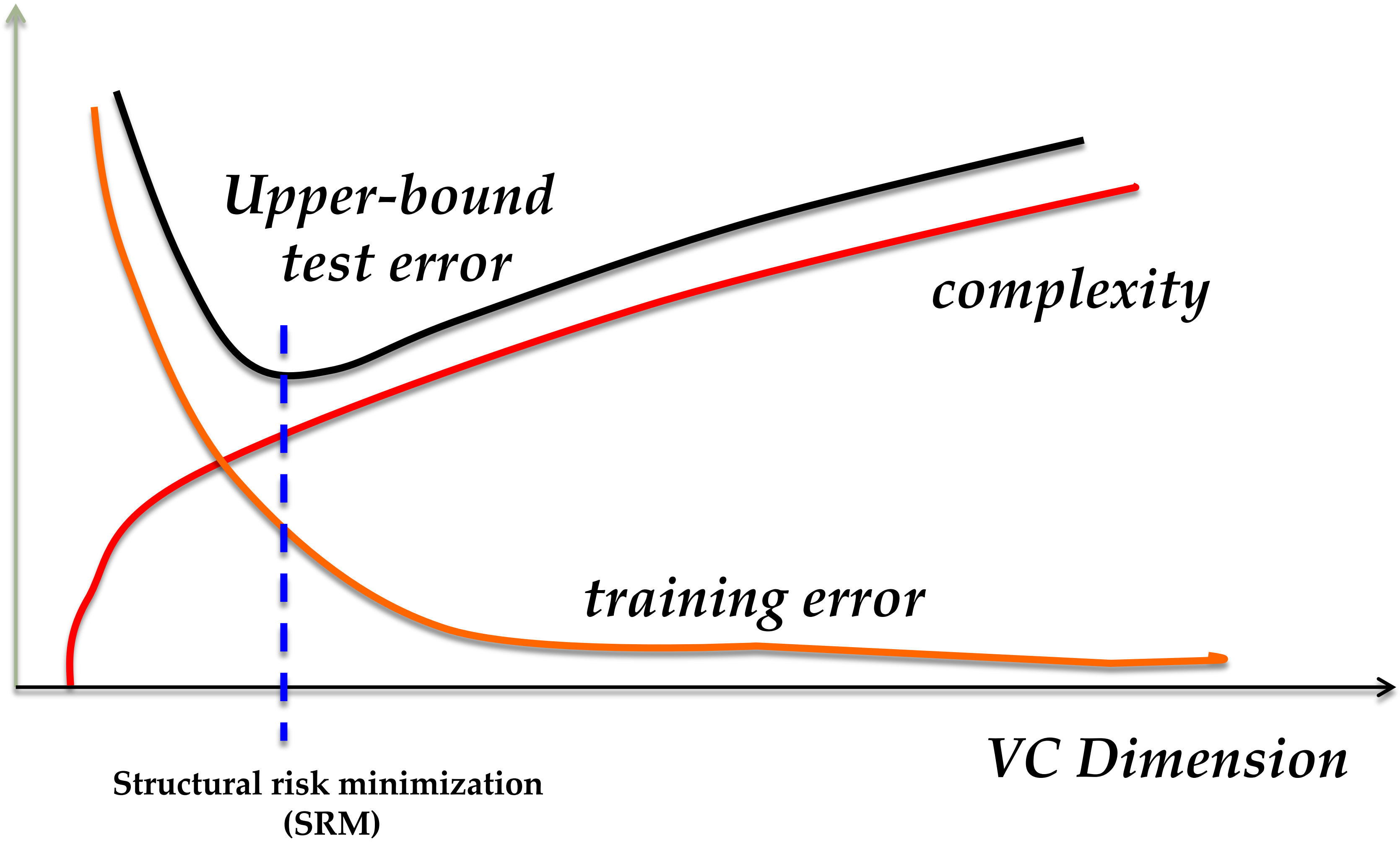
$$\text{test error} \leq \text{training error} + \sqrt{\frac{C\left(\log\left(\frac{2N}{C}\right)+1\right)-\log\left(\frac{\eta}{4}\right)}{N}}$$

$$N = \text{number of training samples}$$
$$C = \text{machine capacity, or VC dimension}$$

$$\text{test error} \leq \text{training error} + \text{penalty(complexity)}$$

- As machine capacity C (or VC) increases, penalty (and the upper bound) increases
- As we fix N and the training error, C (or VC) **positively affects** the training error

Pictorial View (for fixed N)



VC Dimension

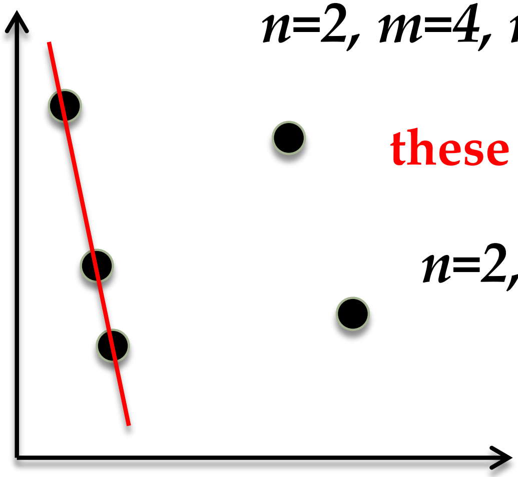
How to determine VC Dimension for a given classifier or hypothesis?

VC Dimension

- VC Dimension for a non-linear classifier is still an open research problem
- We focus on VC Dimension for linear classifier
- We need to learn two important concepts
  - **Points in General Position**
  - **Shattering**

Concept 1: Points in General Position

- **Statement:** In a *n*-dimensional feature space, a set of *m* points ($m > n$) is in "general position" if and only if no subset of ($n + 1$) points lie on ($n - 1$) dimensional hyperplane



$n=2, m=4, m>n$

these 4 points are in general position

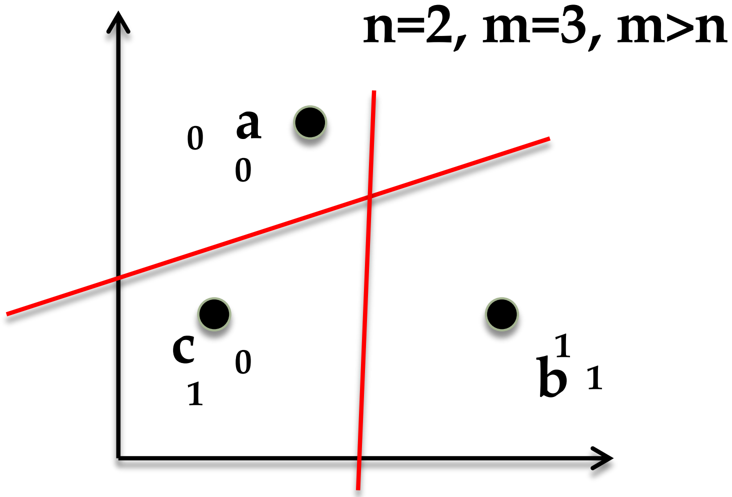
$n=2, m=5, m>n$

these five points are NOT in general position

No 3 points lie on a straight line

Concept 2: Shattering

- **Statement:** A hypothesis (H) shatters *m* points in *n*-dimensional space if **all possible combinations** of *m* points in *n*-dimensional space are correctly classified by H



a	b	c
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

A straight line shatters 3 points in 2 dimensions provided they are in general position

We can add the 4th point, now we have 16 combinations, and we can't find a straight line to separate them all. 4 points in 2 dimension space cannot be shattered by a straight line.

VC Dimension

- **VC Dimension** is the cardinality of the largest set of points that the hypothesis can shatter
- **VC Dimension of the linear classifier:** (n+1) (points should be in **general position**)
  - For 2 dimensions, VC dimension of a line is 3
- **VC Dimension of nonlinear classifier:** very difficult to compute

Points to Remember

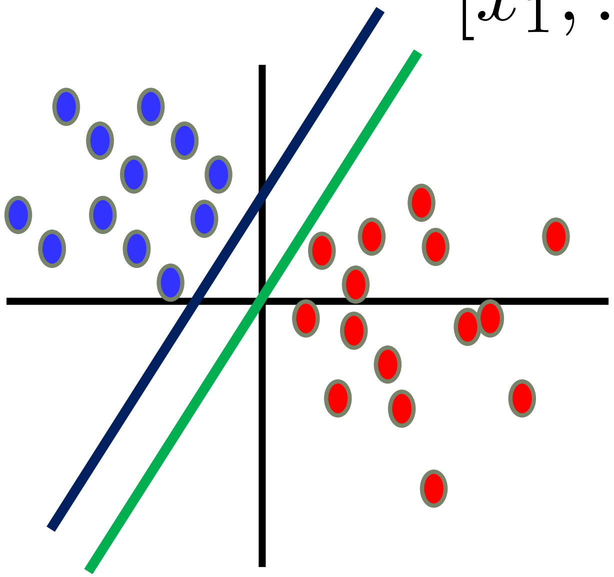
- **VC Dimension** is directly related to machine/hypothesis capacity
- For a given training set size and training error, VC dimension gives probabilistic upper bound of test error
- VC Dimension is a cardinality of the largest set of points that the machine/hypothesis can shatter
- For a **good generalization** (less test error), VC dimension of a machine/hypothesis should be finite. High value of VC dimension gives good generalization for asymptotical solutions.

Learners & Complexity

- We have seen the tradeoffs of underfitting/overfitting
  - Complexity of the learner
  - Representation Power
- Different learners have different power

parameter θ

$[x_1, \ldots, x_d]$ → **classifier** → ***predicted class***

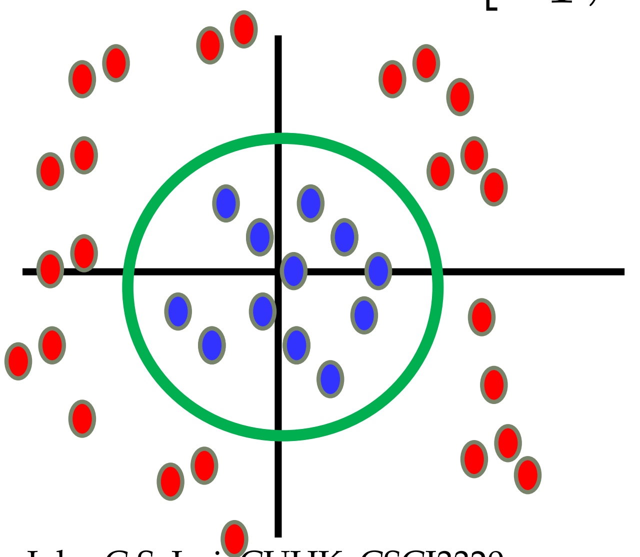


$$c(x) = \text{sign}(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$

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

Learners & Complexity

- We have seen the tradeoffs of underfit/overfit
  - Complexity of the learner
  - Representation Power
- Different learners have different power



$$c(x) = \mathrm{sign}(\boldsymbol{x}^T \boldsymbol{x} - \theta)$$

Learners & Complexity

- We have seen the tradeoffs of underfit/overfit
  - Complexity of the learner
  - Representation Power
- Different learners have different power

- Trade-off:
  - More power: represent more complex system, may overfit
  - Less power: won't overfit, but may not find "best" solution

- We use VC dimension to ***quantify representation power***

The BIG Question

- Relates "**training error**" with "**test error**"?
- **Equation**: gives upper bound of test error with probability $1-\eta$

$$\text{test error} \leq \text{training error} + \sqrt{\frac{C\left(\log\left(\frac{2N}{C}\right)+1\right)-\log\left(\frac{\eta}{4}\right)}{N}}$$

$$N = \text{number of training samples}$$
$$C = \text{machine capacity, or VC dimension}$$

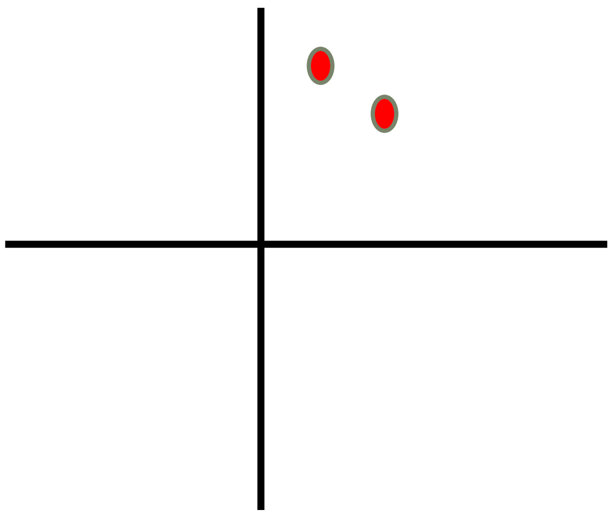
$$\text{test error} \leq \text{training error} + \text{penalty}(\text{complexity})$$

- As machine capacity C (or VC) increases, penalty (and the upper bound) increases
- As we fix N and the training error, C (or VC) **positively affects** the training error

Shattering

- We say a classifier f(x) can shatter points $x^{(1)} \dots x^{(n)}$ iff for ***ALL*** $y^{(1)} \dots y^{(n)}$, f(x) can achieve **ZERO** error on the training data $(x^{(1)},y^{(1)}), (x^{(2)},y^{(2)}), \dots, (x^{(n)},y^{(n)})$.
- In other words, there exits some θ that gets **zero error**
- Can we shatter these points using:

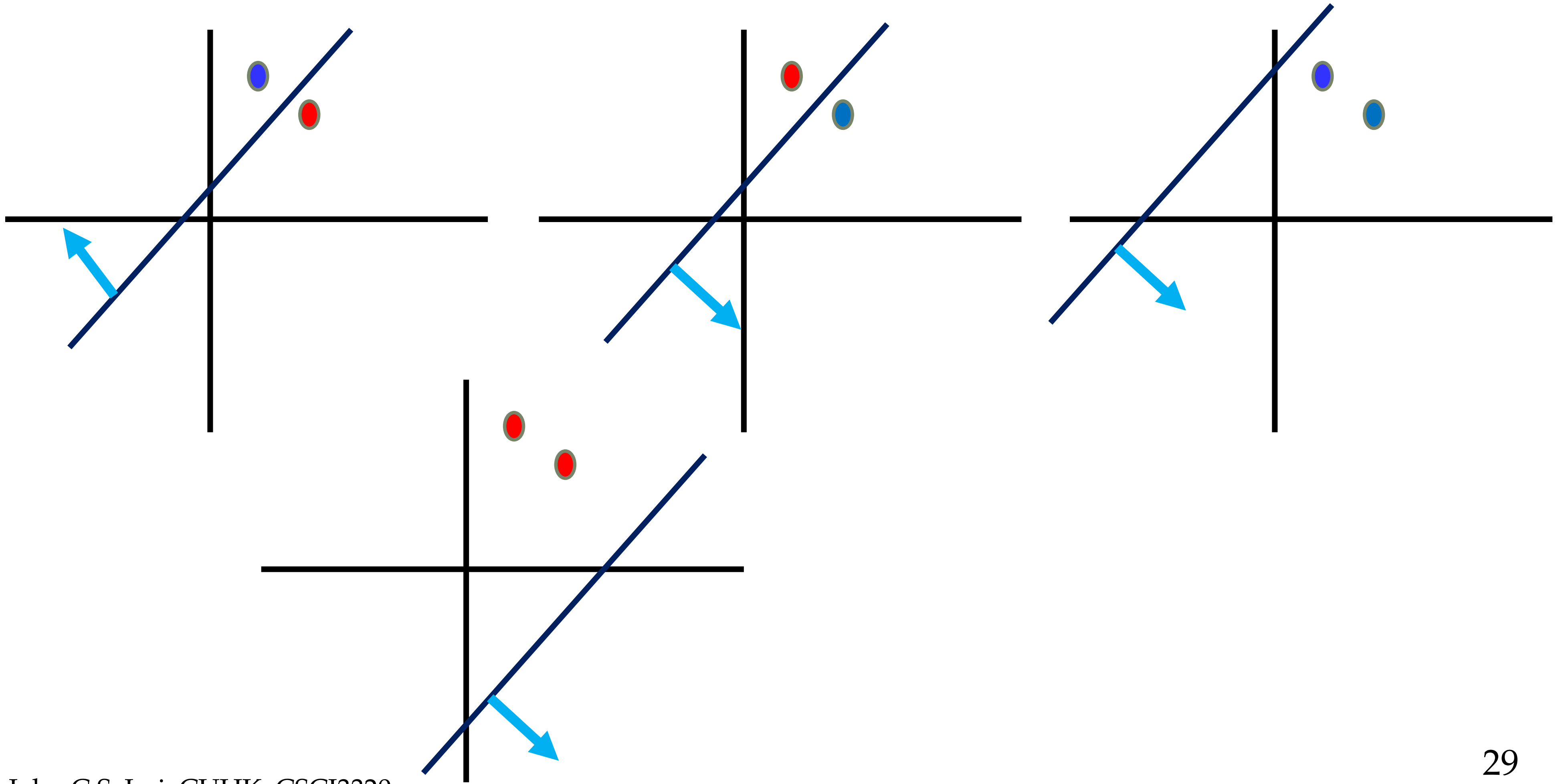
$$c(x) = \mathrm{sign}(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$



Shattering

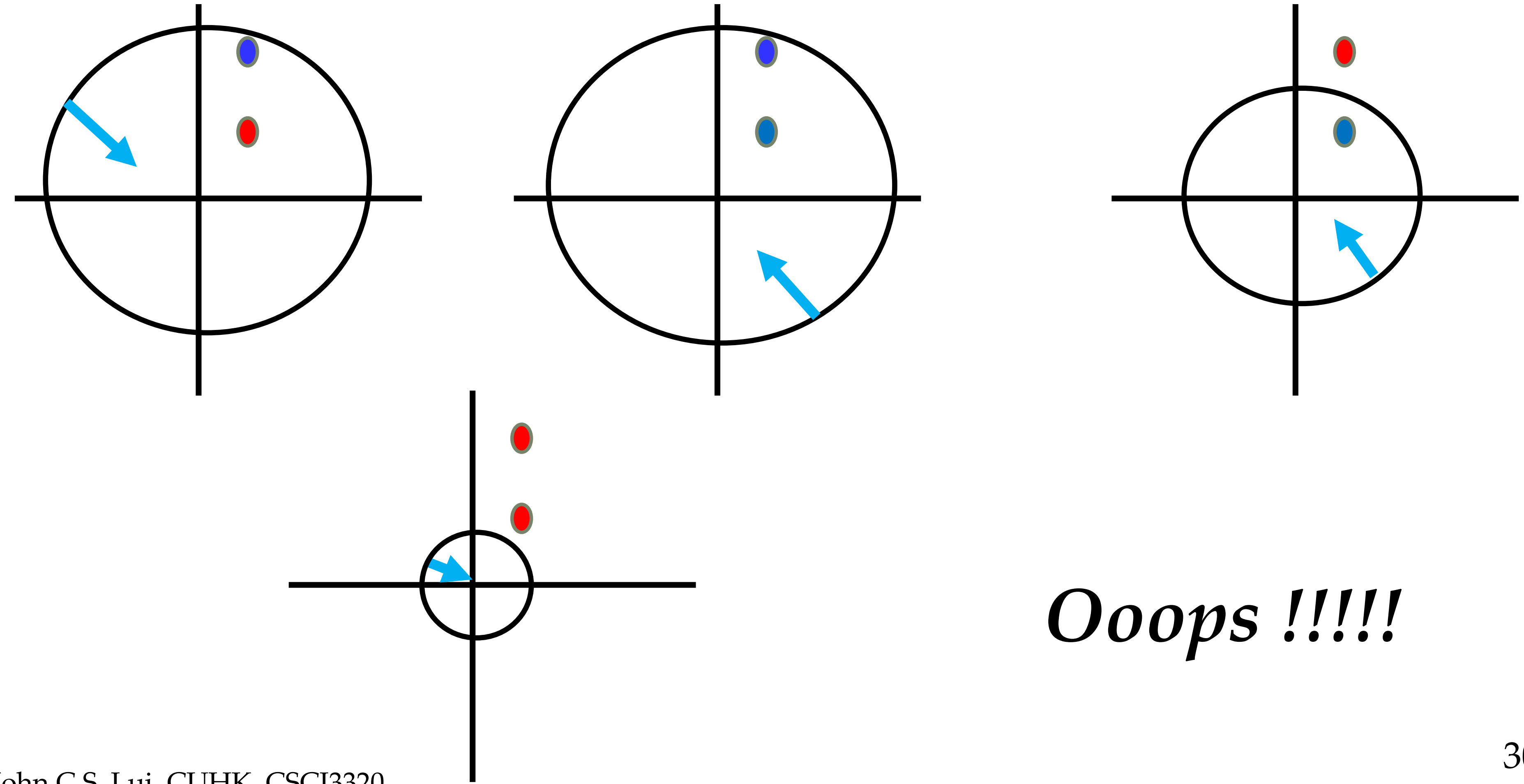
- Can we shatter these points using:

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



Shattering

- Can we shatter these points using: $c(x) = \mathrm{sign}(\boldsymbol{x}^T \boldsymbol{x} - \theta)$



Ooops !!!!!

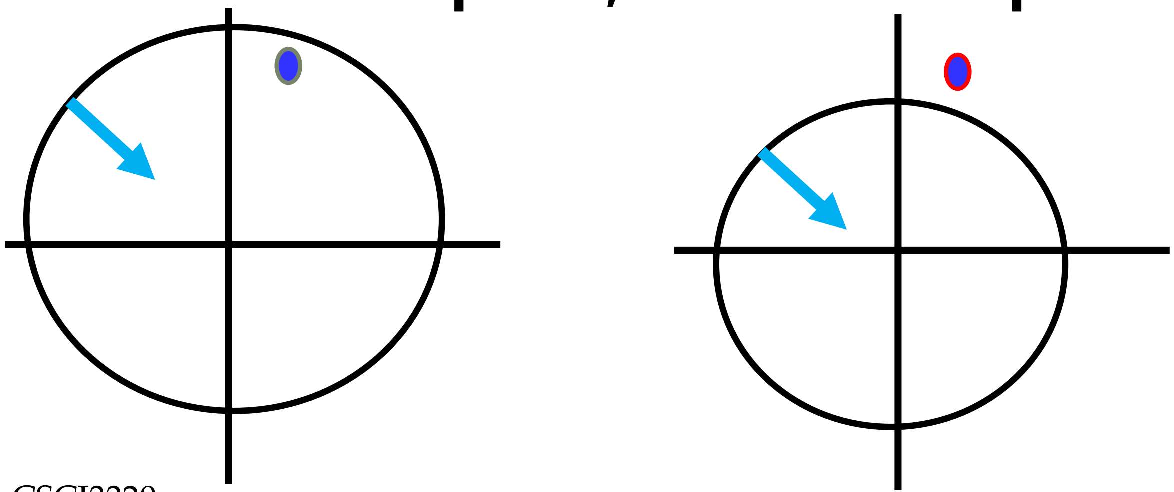
VC Dimension

- The VC dimension C is defined as the *maximum number of points* h that can be arranged so that $f(x)$ can shatter them
- Think of it as a GAME:
  - Choose the definition of $f(x;\theta)$
  - **Player 1:** choose positions for $x^{(1)} \ldots x^{(n)}$
  - **Player 2:** choose target labels $y^{(1)} \ldots y^{(n)}$
  - If $f(x;\theta)$ **can reproduce the target labels, P1 wins**

$$\ni (x^{(1)}, \ldots, x^{(h)}) \text{ s.t. } \forall (y^{(1)}, \ldots, y^{(h)}) \ \ni \theta \text{ s.t. } \forall i \, f(x^{(i)}; \theta) = y^{(i)}$$

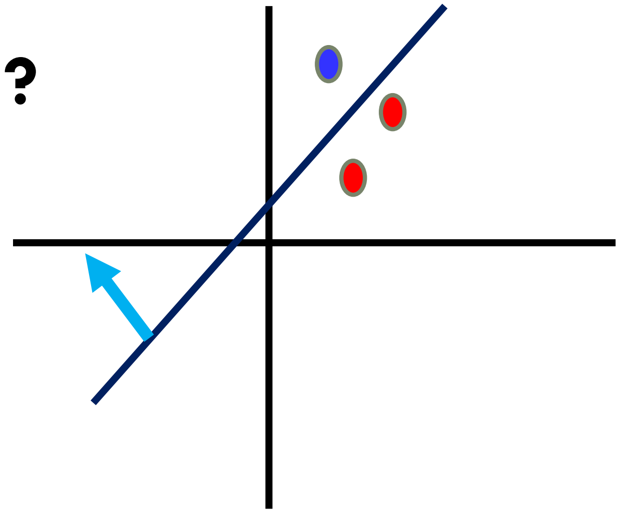
VC Dimension

- The VC dimension C is defined as the ***maximum number of points h*** that an be arranged so that *f(x)* can shatter them
- **Example:** What is the VC dimension of the (zero-centered) circle, $f(x;\theta) = sign\,(\boldsymbol{x}'\boldsymbol{x} - \theta)$ **?**
- **VC Dimension = 1**
  - **We can shatter one point, but not two points**

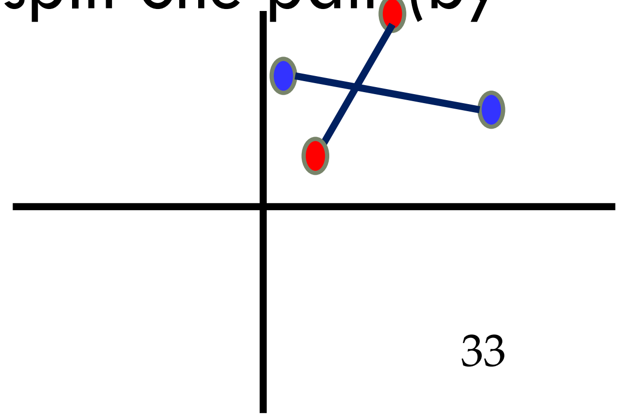


VC Dimension

- **Example:** What is the VC dimension of the two-dimensional line:
  - $f(x;\theta) = sign\,(\theta_0 + \theta_1\, x_1 + \theta_2\, x_2)$ **?**



- **VC Dimension = 3 ?**
  - **Yes**
- **VC Dimension = 4?**
  - **No**
  - Any line pass through these points must split one pair (by crossing one of the lines)



A linear classifier in *d* dimension with constant Term: VC Dimension = *d+1*

VC Dimension

- VC dimension measures the "power" (or capacity) of the learner
- In general, it does NOT necessary equal to the # of parameters !!!!

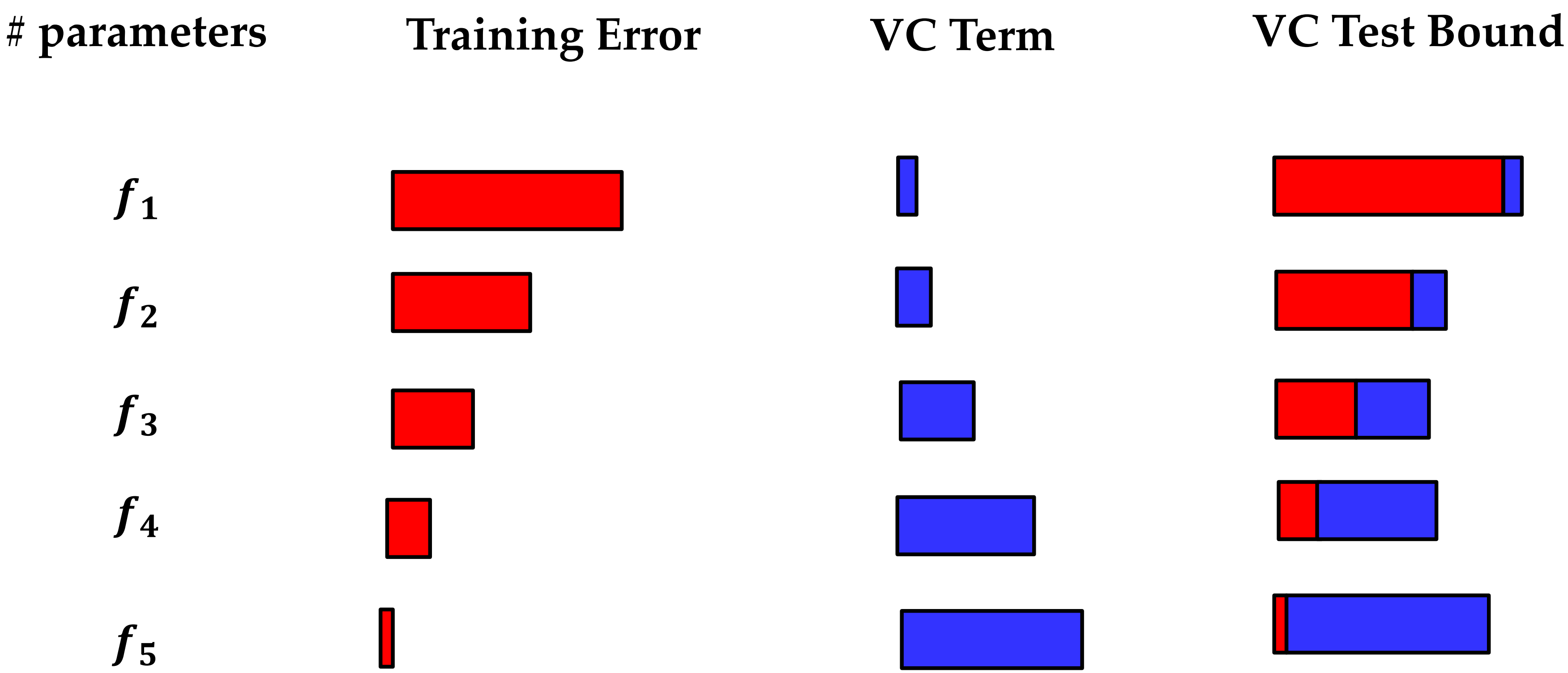
- Number of parameters (or features) does not necessary equal to complexity
  - Give an example of a classifier with lots of parameters but not much power ……
  - Give an example of a classifier with one parameter but lots of power ……..
- Still on going work to determine the VC dimension of different learners…..

Using VC Dimension

- Used validation / testing (cross-validation) to select complexity

Using VC Dimension

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



Conclusion

- VC dimension explains the “power” of the learner
- Higher the value, more powerful is the learning in reducing ***training error***
- The testing error is more important (assume we fixed *N*)
  - Reduces when we increase the power of a learner
  - Then it will increase (overfit)
- Slowly increase the power of your learner so to “minimize” (or reduce) the testing error

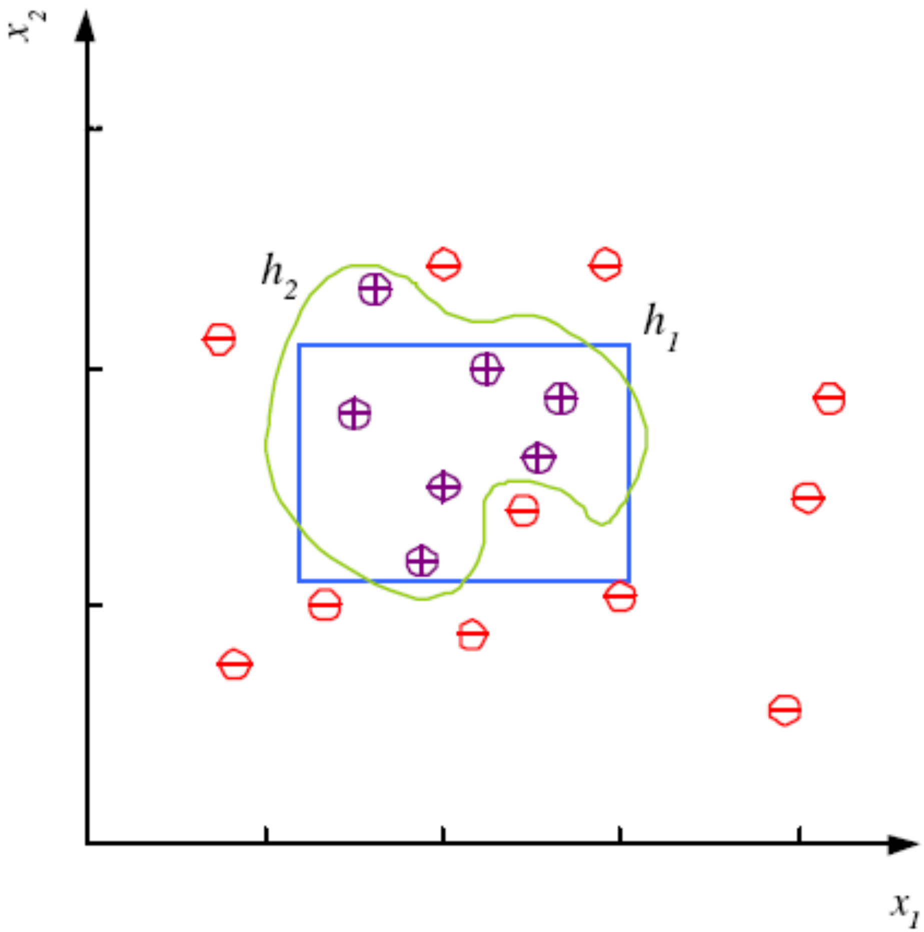
Noise and Model Complexity

Noise: "mis-labeled" data

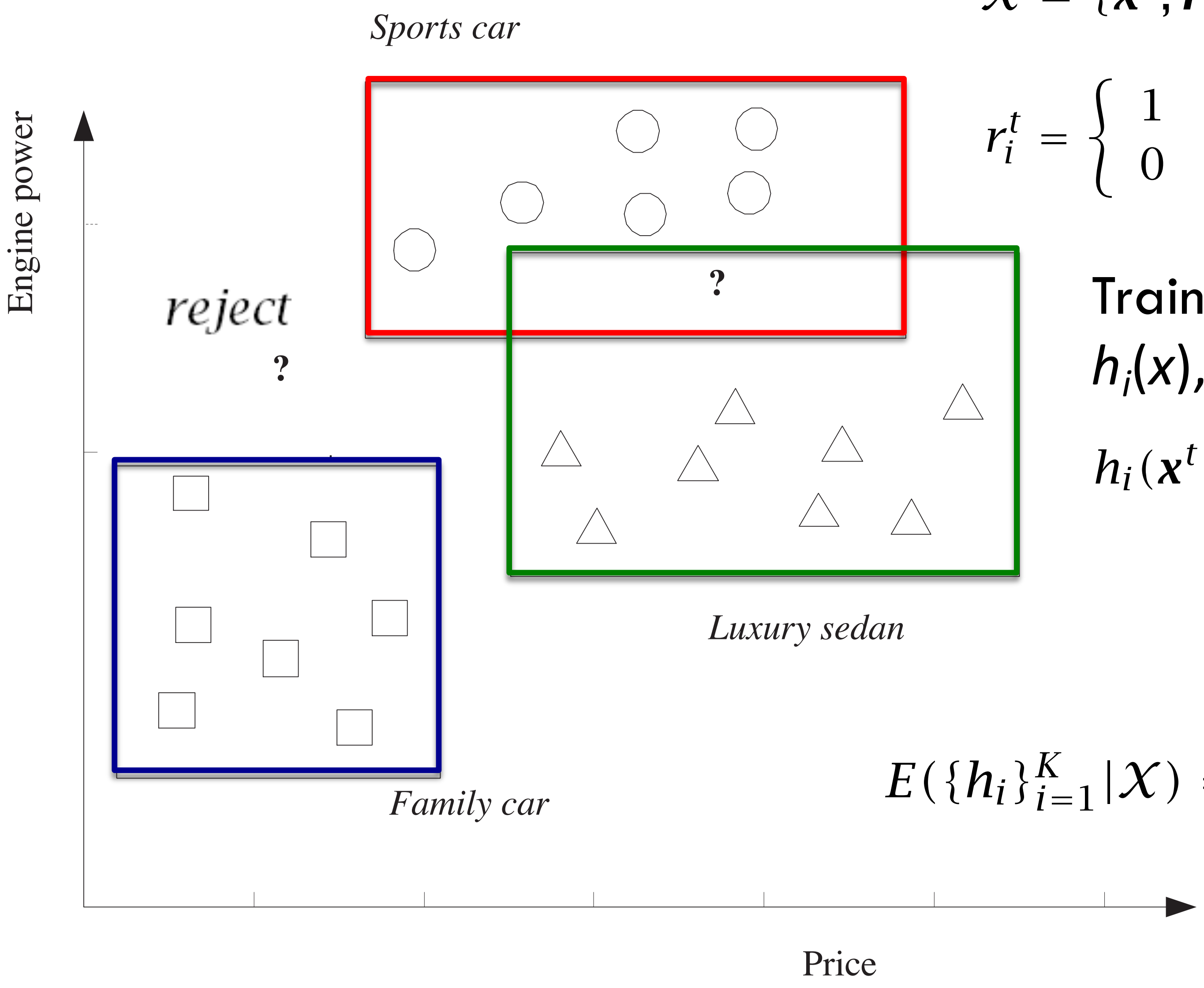
Latent variable: hidden or unobservable attributes **(example)**

Use the simpler model 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) or ***KISS***



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



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

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

K dimension

Train hypotheses

$h_i(x)$, i =1,...,K:

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

$$E(\{h_i\}_{i=1}^K | \mathcal{X}) = \sum_{t=1}^N \sum_{i=1}^K 1(h_i(\boldsymbol{x}^t) \neq r_i^t)$$

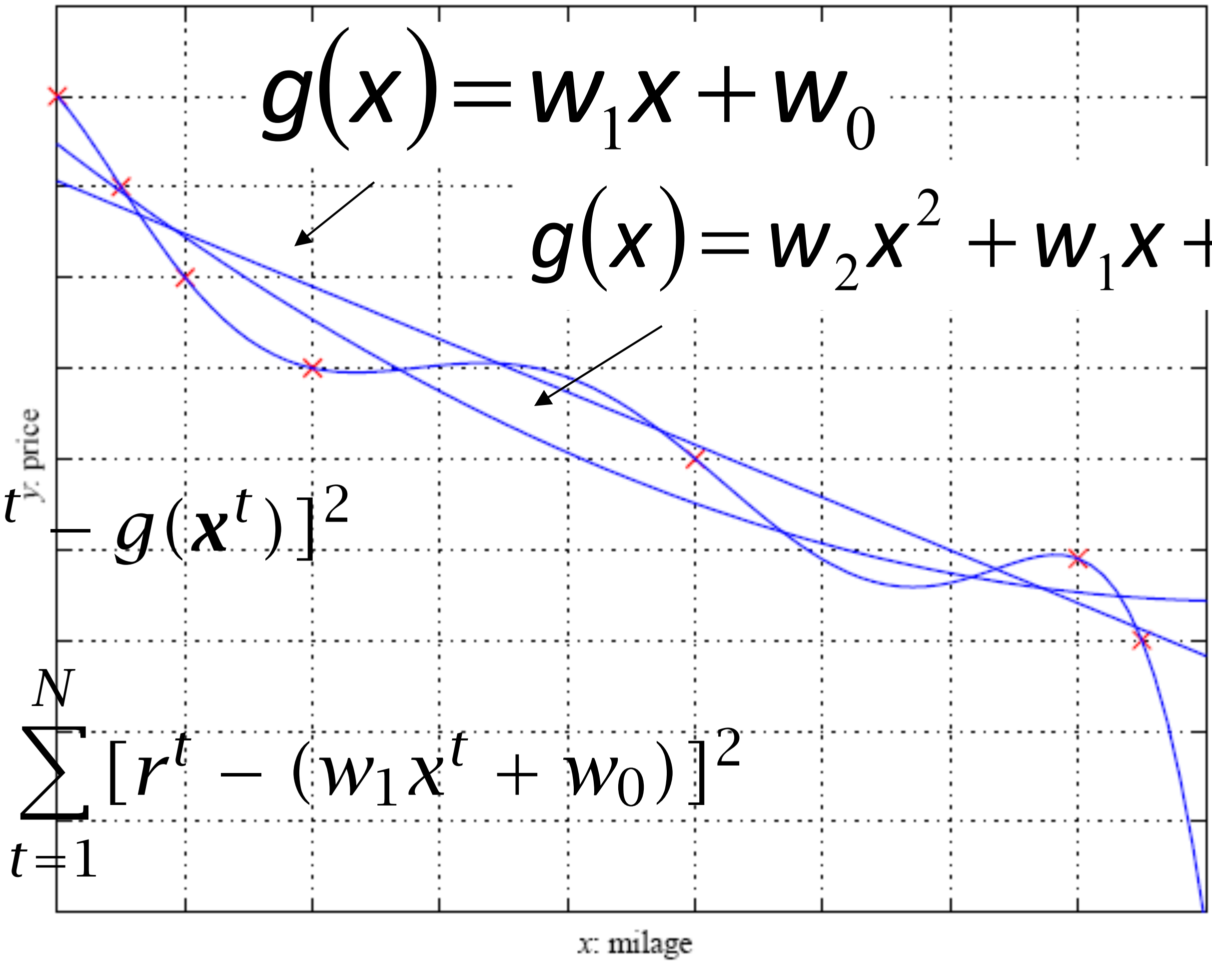
Regression

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

$r^t \in \Re$

$r^t = g(\boldsymbol{x}^t) + \epsilon$

Find *g()* via polynomial interpolation



$$g(x) = w_1 x + w_0$$

$$g(x) = w_2 x^2 + w_1 x + w_0$$

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

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

Regression

- Assume we consider a "*linear function*" *f*
- We consider: $g(x) = w_1 x + w_0$
- Our empirical error on the training set $\mathcal{X}$:

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

- Determine the two parameters via partial derivative and equate them to zero

$$\begin{aligned} w_1 &= \frac{\sum_t x^t r^t - \overline{x}\,\overline{r}N}{\sum_t (x^t)^2 - N\overline{x}^2} \\ w_0 &= \overline{r} - w_1 \overline{x} \end{aligned}$$

where $\overline{x} = \sum_t x^t/N$ and $\overline{r} = \sum_t r^t/N$.

Model Selection & Generalization

- Learning is *an "ill-posed problem"*; where data is not sufficient to find a unique solution
- The need for inductive bias, assumptions about $\mathcal{H}$
- The assumption of "*rectangle*" in family car classification is an inductive bias. Previous example of "linear function *g()*" is another inductive bias.
- Generalization: How well a model performs on **new data** ?
- **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, $E\downarrow$
- As c ($\mathcal{H}$)$\uparrow$, first $E\downarrow$ and then $E\uparrow$

Cross-Validation

- To estimate generalization error, we need data unseen during training. We split the data as
  - Training set (50%)
  - Validation set (25%)
  - Test (publication) set (25%)
  - *Give example to illustrate "training, validation and test".*
- Resampling when there is few data

Dimensions of a Supervised Learner

1. **Model:** $g(\mathbf{x}|\theta)$ **input:** $\mathcal{X} = \{x^t, r^t\}_{t=1}^N$

   where $g(\cdot)$ is the model, x is the input, and θ are the parameters.

2. **Loss function:** $E(\theta|\mathcal{X}) = \sum_t L(r^t, g(x^t|\theta))$

   In class learning where outputs are 0/1, $L(\cdot)$ checks for equality or not; in regression, because the output is a numeric value, we have ordering information for distance and one possibility is to use the square of the difference.

3. **Optimization procedure:** $\theta^* = \arg\min_\theta E(\theta|\mathcal{X})$

   Various optimization algorithms (e.g., convex optimization, linear programming, gradient-based method, simulated annealing,..etc) are used.