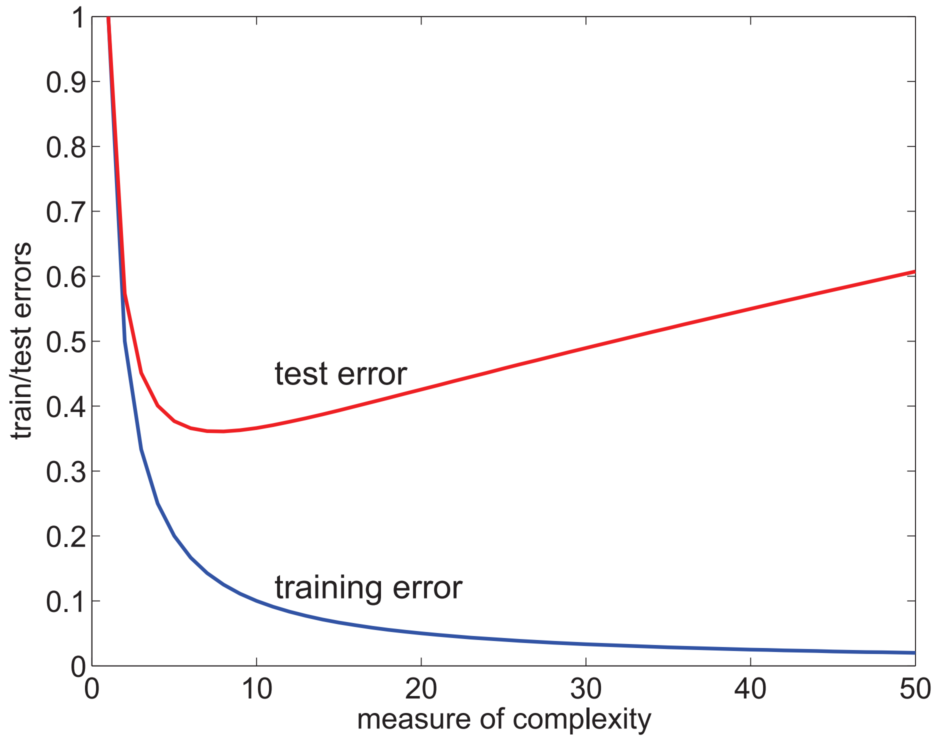
Why care about "complexity"?



- We need a quantitative measure of complexity in order to be able to relate the training error (which we can observe) and the test error (that we'd like to optimize)

Finite case

- We'll start by considering only a finite number of possible classifiers, $h_1(\mathbf{x}), \ldots, h_M(\mathbf{x})$ (e.g., randomly chosen linear classifiers)
- Key questions:

1. Given n training examples and M possible classifiers how far can the training and test errors be?
2. How many training examples do we need so that the errors are close?

The answers will depend on M.

Finite case: definitions

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

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

Finite case: definitions

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

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

- Suppose we choose the classifier that minimizes the training error, $\hat{i}_n = \arg\min_{i=1,\ldots,M} \hat{\mathcal{E}}_n(i)$, then

$$\text{Training error} = \hat{\mathcal{E}}_n(\hat{i}_n)$$

$$\text{Test error} = \mathcal{E}(\hat{i}_n)$$

Finite case: errors

- The training and test errors,

$$\text{Training error} = \hat{\mathcal{E}}_n(\hat{i}_n)$$
$$\text{Test error} = \mathcal{E}(\hat{i}_n)$$

are necessarily close if we can show that the errors are close for all the classifiers in our set:

$$|\hat{\mathcal{E}}_n(i) - \mathcal{E}(i)| \leq \epsilon, \text{ for all } i = 1, \ldots, M$$

- We can now express our key questions more formally in terms of n, M, and ϵ

Finite case: key questions revisited

- Key questions (rewritten):

1. Given n training examples and M possible classifiers, what is the smallest ϵ such that

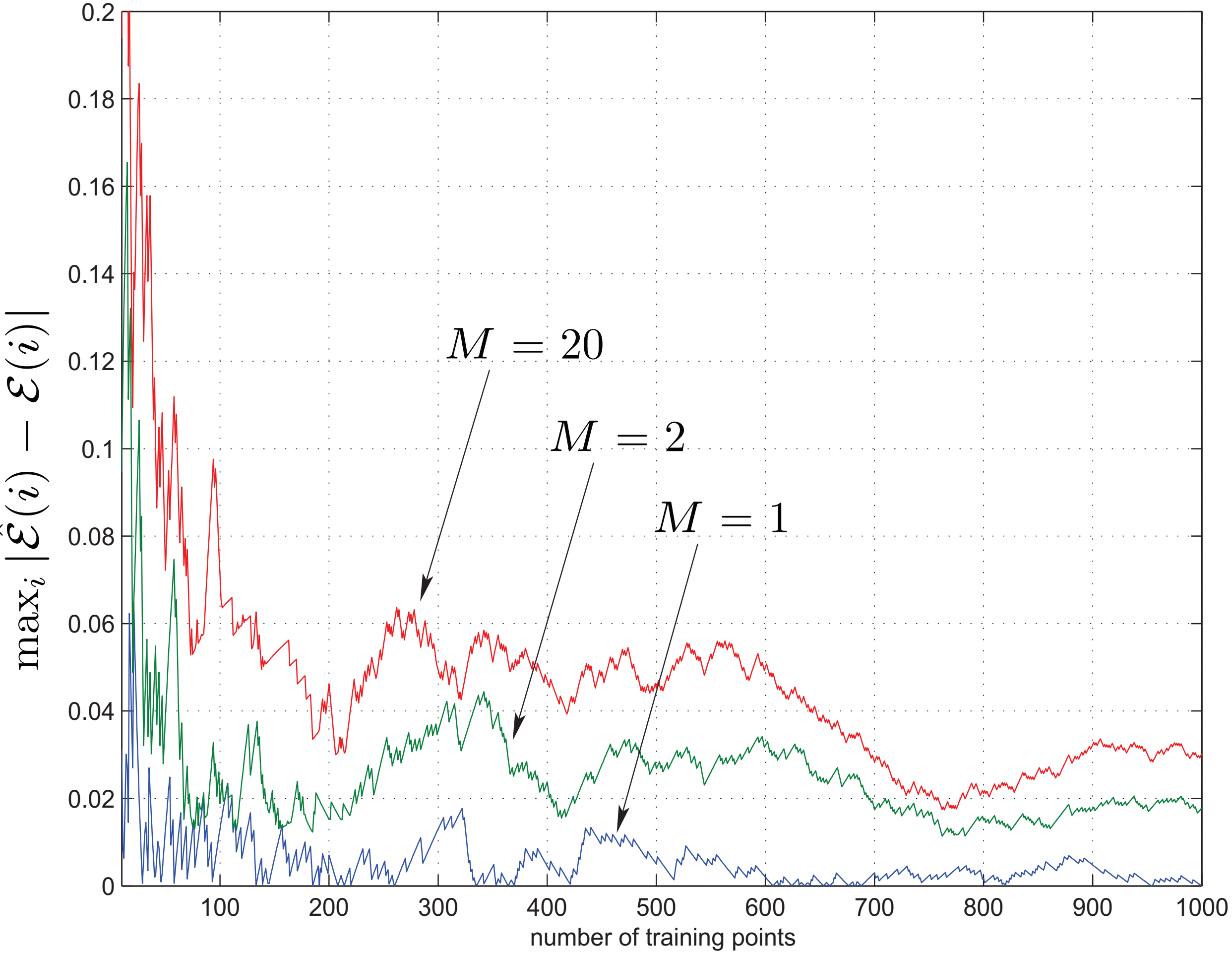
$$\max_{i=1,\ldots,M} |\hat{\mathcal{E}}_n(i) - \mathcal{E}(i)| \leq \epsilon$$

2. For a given ϵ how many training examples do we need so that

$$\max_{i=1,\ldots,M} |\hat{\mathcal{E}}_n(i) - \mathcal{E}(i)| \leq \epsilon$$

Since training examples are sampled at random from some underlying distribution, we can only answer these questions probabilistically.

Finite case: errors



Finite case: probabilistic statement

- We can relate n, M, and ϵ by requiring that with high probability, the empirical errors of all the classifiers in our set are ϵ-close to their expected errors:

$$P\left(\max_{i=1,\ldots,M} |\hat{\mathcal{E}}_n(i) - \mathcal{E}(i)| \leq \epsilon \right) \geq 1 - \delta$$

  The probability is taken over the choice of the training set and $1 - \delta$ specifies our confidence in the probabilistic statement.

Finite case: probabilistic statement

- We can relate n, M, and ϵ by requiring that with high probability, the empirical errors of all the classifiers in our set are ϵ-close to their expected errors:

$$P\left(\max_{i=1,\ldots,M} |\hat{\mathcal{E}}_n(i) - \mathcal{E}(i)| \leq \epsilon \right) \geq 1 - \delta$$

  The probability is taken over the choice of the training set and $1 - \delta$ specifies our confidence in the probabilistic statement.

- Equivalently, we can bound the probability that the empirical error of some classifier in our set deviates more than ϵ from the expected error:

$$P\left(\max_{i=1,\ldots,M} |\hat{\mathcal{E}}_n(i) - \mathcal{E}(i)| > \epsilon \right) \leq \delta$$

Finite case cont'd

- Let's fix n, M, and ϵ and try to find δ so that

$$P\Big(\max_{i=1,\ldots,M} |\hat{\mathcal{E}}_n(i) - \mathcal{E}(i)| > \epsilon \Big) \leq \delta$$

still holds. The probability is take over the choice of the training set.

By using the fact that $P(A \text{ or } B) \leq P(A) + P(B)$ we get

$$\begin{aligned} P\left(\max_i |\hat{\mathcal{E}}_n(i) - \mathcal{E}(i)| > \epsilon\right) &\leq \sum_{i=1}^{M} P\left(|\hat{\mathcal{E}}_n(i) - \mathcal{E}(i)| > \epsilon \right) \\ &\leq \sum_{i=1}^{M} 2\exp(-2n\epsilon^2) \quad \text{(Chernoff)} \\ &= M \cdot 2\exp(-2n\epsilon^2) = \delta \end{aligned}$$

Finite case cont'd

- We are now able to relate n, M, ϵ, and δ:

$$M \cdot 2\exp(-2n\epsilon^2) = \delta, \quad \text{or} \quad \epsilon = \sqrt{\frac{\log(M) + \log(2/\delta)}{2n}}$$

- We can restate our result in terms of a bound on the expected error of any classifier in our set.

  Theorem: With probability at least $1-\delta$ over the choice of the training set, for all $i = 1, \ldots, M$

$$\mathcal{E}(i) \leq \hat{\mathcal{E}}_n(i) + \epsilon(n, M, \delta)$$

  where $\epsilon = \epsilon(n, M, \delta)$ is a "complexity penalty".

Measures of complexity

- Typically the set of classifiers is not a finite nor a countable set (e.g., the set of linear classifiers)
- There are still many ways of trying to capture the "effective" number of classifiers in such a set:
  - degrees of freedom (number of parameters)
  - Vapnik-Chervonenkis (VC) dimension
  - description length

  etc.

VC-dimension: preliminaries

- **A set of classifiers $\mathbf{F}$:** For example, this could be the set of all possible linear classifiers, where $h \in F$ means that

$$h(\mathbf{x}) = \text{sign}\left(w_0 + \mathbf{w}_1^T \mathbf{x} \right)$$

for some values of the parameters $w_0, \mathbf{w}_1$.

VC-dimension: preliminaries

- **Complexity:** how many different ways can we label n training points $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ with classifiers $h \in F$?

  In other words, how many distinct binary vectors

  $$[h(\mathbf{x}_1)\; h(\mathbf{x}_2)\; \ldots\; h(\mathbf{x}_n)]$$

  do we get by trying out each $h \in F$ in turn?

  $$\begin{array}{ll} [\; -1 \quad 1 \quad \ldots \quad 1 \;] & h_1 \\ [\; 1 \quad -1 \quad \ldots \quad 1 \;] & h_2 \\ & \ldots \end{array}$$

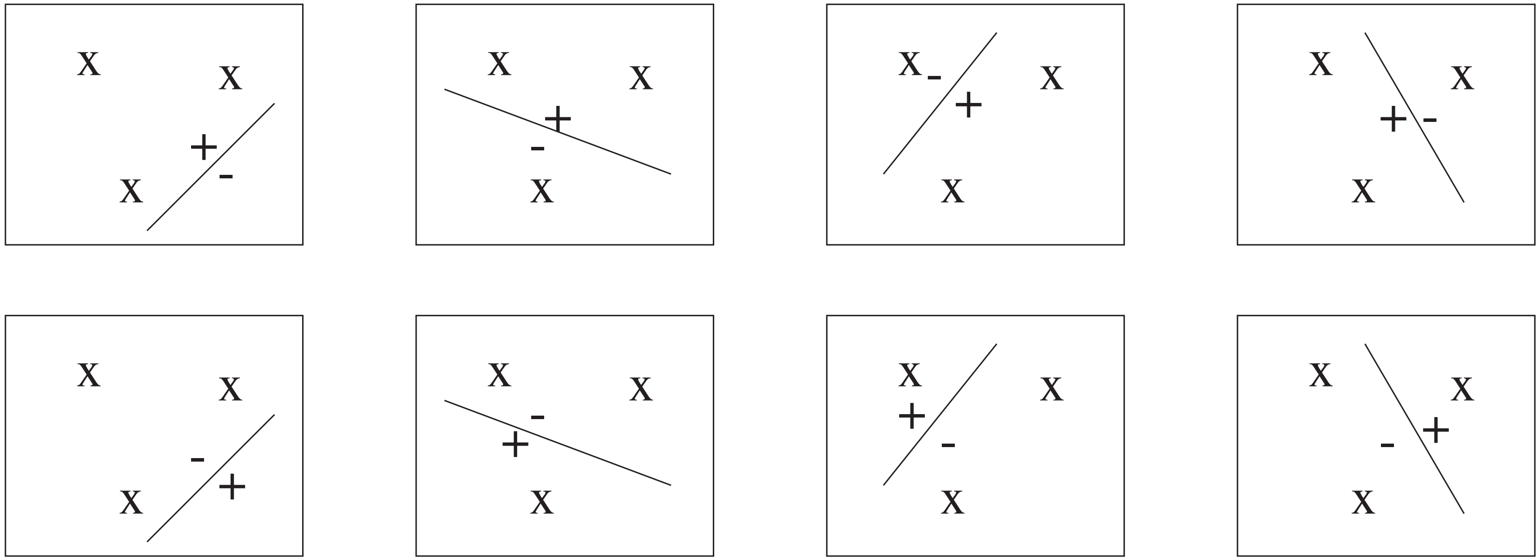
VC-dimension: shattering

- A set of classifiers F *shatters* n points $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ if

$$[h(\mathbf{x}_1)\ h(\mathbf{x}_2)\ \ldots\ h(\mathbf{x}_n)],\quad h \in F$$

generates all 2^n distinct labelings.

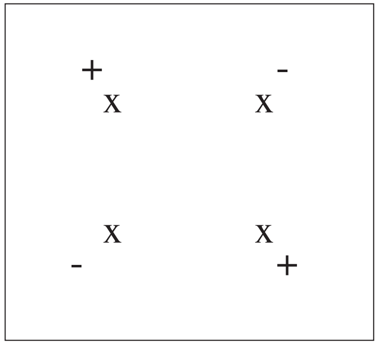
- Example: linear decision boundaries shatter (any) 3 points in 2D



but not any 4 points...

VC-dimension: shattering cont'd

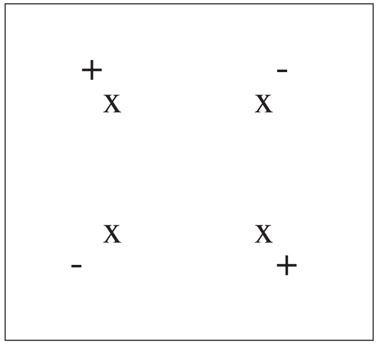
- We cannot shatter any set of 4 points in 2D with linear classifiers. For example, we cannot generate the following XOR-labeling:



- More generally: the set of all d-dimensional linear classifiers can shatter exactly $d+1$ points

VC-dimension: shattering cont'd

- We cannot shatter any set of 4 points in 2D with linear classifiers. For example, we cannot generate the following XOR-labeling:



- More generally: the set of all d-dimensional linear classifiers can shatter exactly $d+1$ points
- **Definition:** The VC-dimension d_{VC} of a set of classifiers F is the number of points F can shatter

Learning and VC-dimension

- We learn something only after we no longer can shatter the training points (have more than d_{VC} training examples)

  Rationale: suppose we have n training examples and labels $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$ and $n < d_{VC}$. Does the training set constrain our prediction for $\mathbf{x}_{n+1}$?

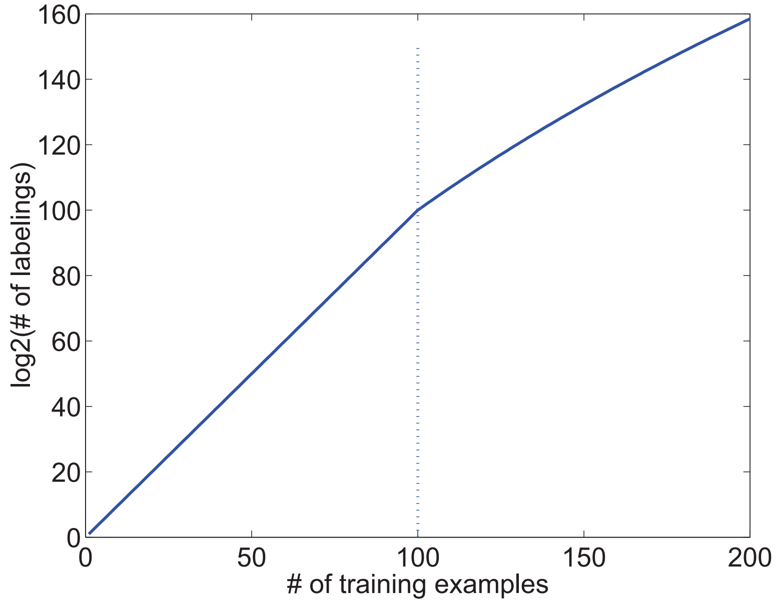
  Because we expect to be able to shatter $n+1$ points ($\leq d_{VC}$) it follows that we can find $h_1, h_2 \in F$, both consistent with training labels, but

  $$h_1(\mathbf{x}_{n+1}) = 1, \quad h_2(\mathbf{x}_{n+1}) = -1$$

  We therefore cannot determine which label to predict for $\mathbf{x}_{n+1}$.

Learning and VC-dimension

- We learn something only after we no longer can shatter the training points (have more than d_{VC} training examples)



$$n \le d_{VC}: \qquad \#\text{ of labelings } = 2^n$$

$$n > d_{VC}: \qquad \#\text{ of labelings } \le \left(\frac{en}{d_{VC}}\right)^{d_{VC}}$$

Learning and VC-dimension

- By essentially replacing log M in the finite case with the log of the number of possible labelings by the set of classifiers over n (really 2n) points, we get an analogous result:

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

$$\varepsilon(h) \leq \hat{\varepsilon}_n(h) + \xi(n, d_{VC}, \delta)$$

$$\xi(n, d_{VC}, \delta) = \sqrt{\frac{d_{VC}\left(\log\frac{2n}{d_{VC}} + 1\right) + \log\frac{4}{\delta}}{n}}$$

Unfortunately, a loose bound

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

  Model 1 d_1
  Model 2 d_2
  Model 3 d_3

  where $d_1 \leq d_2 \leq d_3 \leq \ldots$

- Basic model selection criterion:

$$\text{Criterion} = \text{(empirical) score} + \text{Complexity penalty}$$

Structural risk minimization

- In structural risk minimization we define the models in terms of VC-dimension (or refinements)

Model 1	$d_{VC} = d_1$
Model 2	$d_{VC} = d_2$
Model 3	$d_{VC} = d_3$

  where $d_1 \leq d_2 \leq d_3 \leq \ldots$

- The selection criterion: lowest upper *bound* on the expected loss

$$\text{Expected loss} \leq \text{Empirical loss} + \text{Complexity penalty}$$

Example

- Models of increasing complexity

  Model 1 $K(\mathbf{x}_1, \mathbf{x}_2) = (1 + (\mathbf{x}_1^T \mathbf{x}_2))$
  Model 2 $K(\mathbf{x}_1, \mathbf{x}_2) = (1 + (\mathbf{x}_1^T \mathbf{x}_2))^2$
  Model 3 $K(\mathbf{x}_1, \mathbf{x}_2) = (1 + (\mathbf{x}_1^T \mathbf{x}_2))^3$
  $\ldots$ $\ldots$

- These are nested, i.e.,

$$F_1 \subseteq F_2 \subseteq F_3 \subseteq \ldots$$

  where F_k refers to the set of possible decision boundaries that the model k can represent.

- Still need to derive the criterion...

Structural risk minimization cont'd

- For our zero-one loss (classification error), we can derive the following complexity penalty (Vapnik 1995):

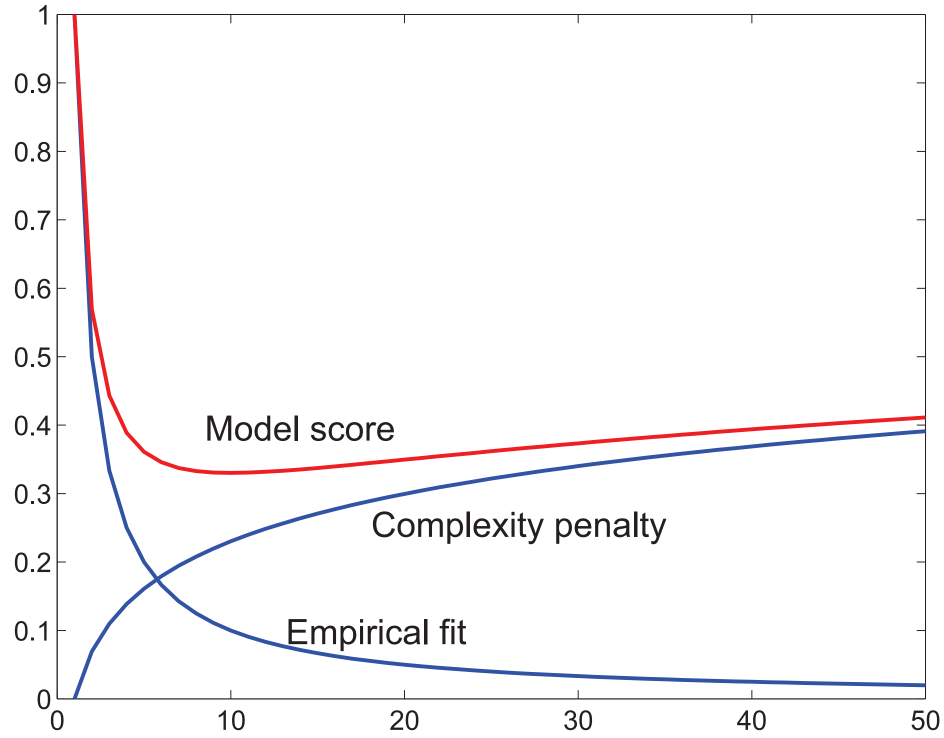
$$\epsilon(n, \delta, d) = \sqrt{\frac{d_{VC}(\log(2n/d_{VC}) + 1) + \log(1/(4\delta))}{n}}$$

1. This is an increasing function of d_{VC}
2. Increases as δ decreases
3. Decreases as a function of n

(this is not the only choice...)

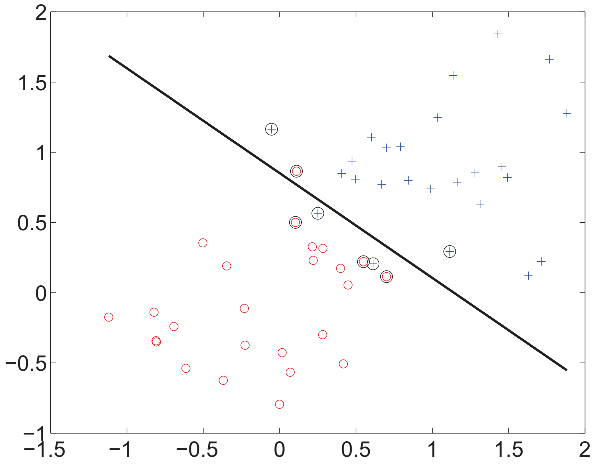
Structural risk minimization cont'd

- Competition of terms...
  1. Empirical loss decreases with increasing d_{VC}
  2. Complexity penalty increases with increasing d_{VC}

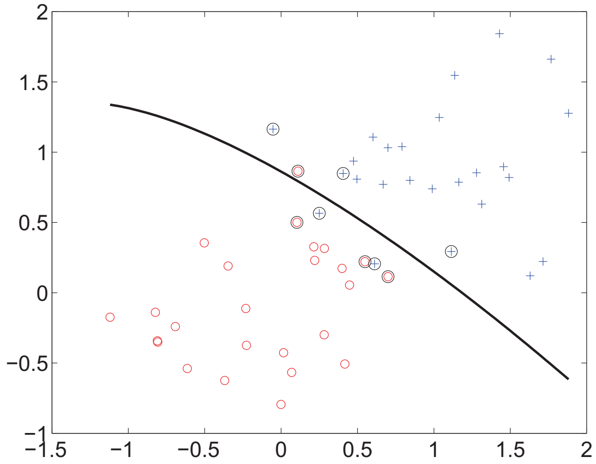


- We find the minimum of the model score (bound).

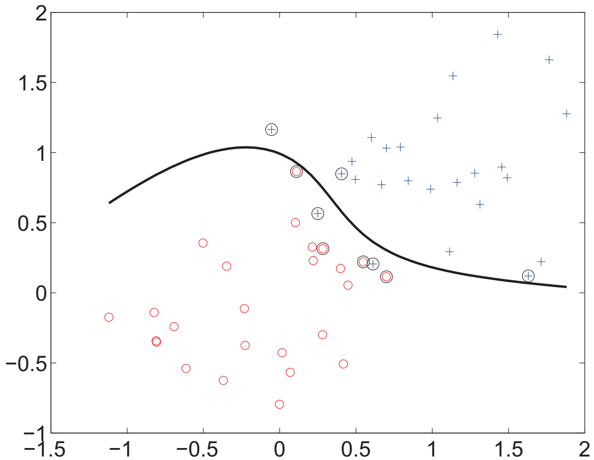
Structural risk minimization: example



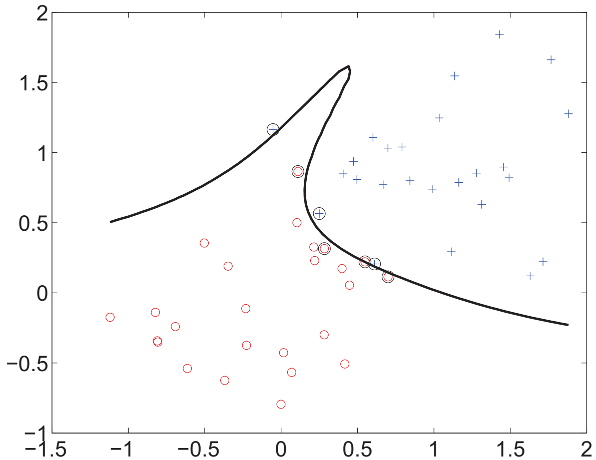
linear



2^{nd} order polynomial



4^{th} order polynomial



8^{th} order polynomial

Structural risk minimization: example cont'd

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

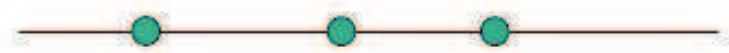
Model	d_{VC}	Empirical fit	Complexity penalty $\epsilon(n, \delta, d_{VC})$
1^{st} order	3	0.06	0.5501
2^{nd} order	6	0.06	0.6999
4^{th} order	15	0.04	0.9494
8^{th} order	45	0.02	1.2849

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

Example: VC dimension of 1-dimensional intervals

- *X* = *R*(e.g., heights of people)
- *H* is the set of hypotheses of the form *a < x <b*
- *Subset containing two instances S ={3.1, 5.7}*

- *Can S be shattered by H?*
- *Yes, e.g., (1<x<2), (1<x<4),(4<x<7),(1<x<7)*
- *Since we have found a set of two that can be shattered, VC(H)is at least two*
- *However, no subset of size three can be shattered*



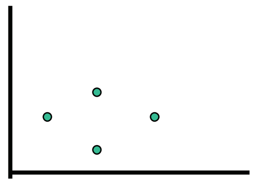
- Therefore VC(H) =2
- Here |H| is infinite but VC(H)is finite

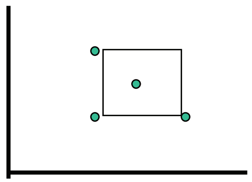
2. (T/F) If there exists a set of k instances that cannot be shattered by H, then VC(H) < k.

3. Give the VC dimension of the class:
H is the set of all perceptrons in 2D plane, i.e.
$H = \{h_{\mathrm{w}} | h_{\mathrm{w}} = \theta(w_0 + w_1 x_1 + w_2 x_2) \text{ where } \theta(z) = 1 \text{ iff } z \geq 0 \text{ otherwise } \theta_z = 0\}$.

4. H = Axis parallel rectangles in R2
What is the VC dimension of H?

Some four instance can be shattered and some cannot

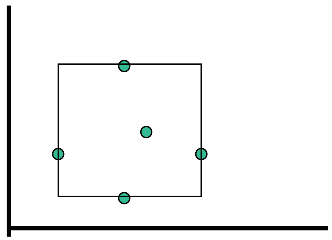




(need to consider here 16 different rectangles)

Shows that VC(H)>=4

- But, no five instances can be shattered



Since, there can be at most 4 distinct extreme points (smallest or largest along some dimension) and these cannot be included (labeled +) without including the 5th point.

Therefore VC(H) = 4