EE 649 Pattern Recognition

Model Selection

Ulisses Braga-Neto

ECE Department

Texas A&M University

Main Ideas

- Obvious questions about classification are: Which classification rule should one choose? How many features should one use?
- A related question is: given a classification rule, how does one pick its free parameters? For example,
  - In k-NN classification, which k to use?
  - In kernel classification, how to choose the kernel bandwidth?
  - With neural networks, which architecture to use? How many epochs to use in training?
- The answer to such questions is not simple. It depends on sample size, the complexity of the classification rule, and on the distribution. In this lecture, we will examine these issues.

Mathematical Formulation

- Given a classification rule $\{\Psi_n\}$, let $\mathcal{C}$ be the collection of classifiers it produces (note that d is fixed, but n varies):

$$\mathcal{C} = \{\psi : R^d \to \{0,1\} \,|\, \psi = \Psi_n(S_n),\ \text{for some } S_n\}$$

- Parameters such as the weights of a fixed-architecture NN or the direction of a LDA hyperplane, are part of the design process, and are thus modeled as part of $\mathcal{C}$.
- *Free* parameters (such as k in k-NN classification, or the number of hidden layers and neurons in an NN) are assumed fixed and thus lead to different families $\mathcal{C}$.
- Therefore, picking free parameters (in fact, picking the classification rule and the dimensionality) corresponds to picking $\mathcal{C}$.

Best Classifier vs. Designed Classifier

- The best classifier in $\mathcal{C}$ according to the true classification error is denoted $\psi_{\mathcal{C}}$:

$$\psi_{\mathcal{C}} = \arg\min_{\psi \in \mathcal{C}} \epsilon[\psi] = \arg\min_{\psi \in \mathcal{C}} P[\psi(X) \neq Y]$$

with error $\epsilon_{\mathcal{C}} = \epsilon[\psi_{\mathcal{C}}]$.

- Given S_n, the designed classifier is denoted $\psi_{n,\mathcal{C}}$:

$$\psi_{n,\mathcal{C}} = \Psi_n(S_n) \in \mathcal{C}$$

with error $\epsilon_{n,\mathcal{C}} = \epsilon[\psi_{n,\mathcal{C}}]$.

Approximation Error vs. Design Error

- The *approximation error* is the difference between the best error in the class and the Bayes error:

  $$\Delta_{\mathcal{C}} = \epsilon_{\mathcal{C}} - \epsilon_d > 0$$

  This reflects how well the classification rule can approximate the Bayes error.

- The *design error* is the difference between the error of the designed classifier and the best error in the class:

  $$\Delta_{n,\mathcal{C}} = \epsilon_{n,\mathcal{C}} - \epsilon_{\mathcal{C}} > 0$$

  This reflects how good a job we can do with the available data to design the best possible classifier in $\mathcal{C}$.

Error of Designed Classifier

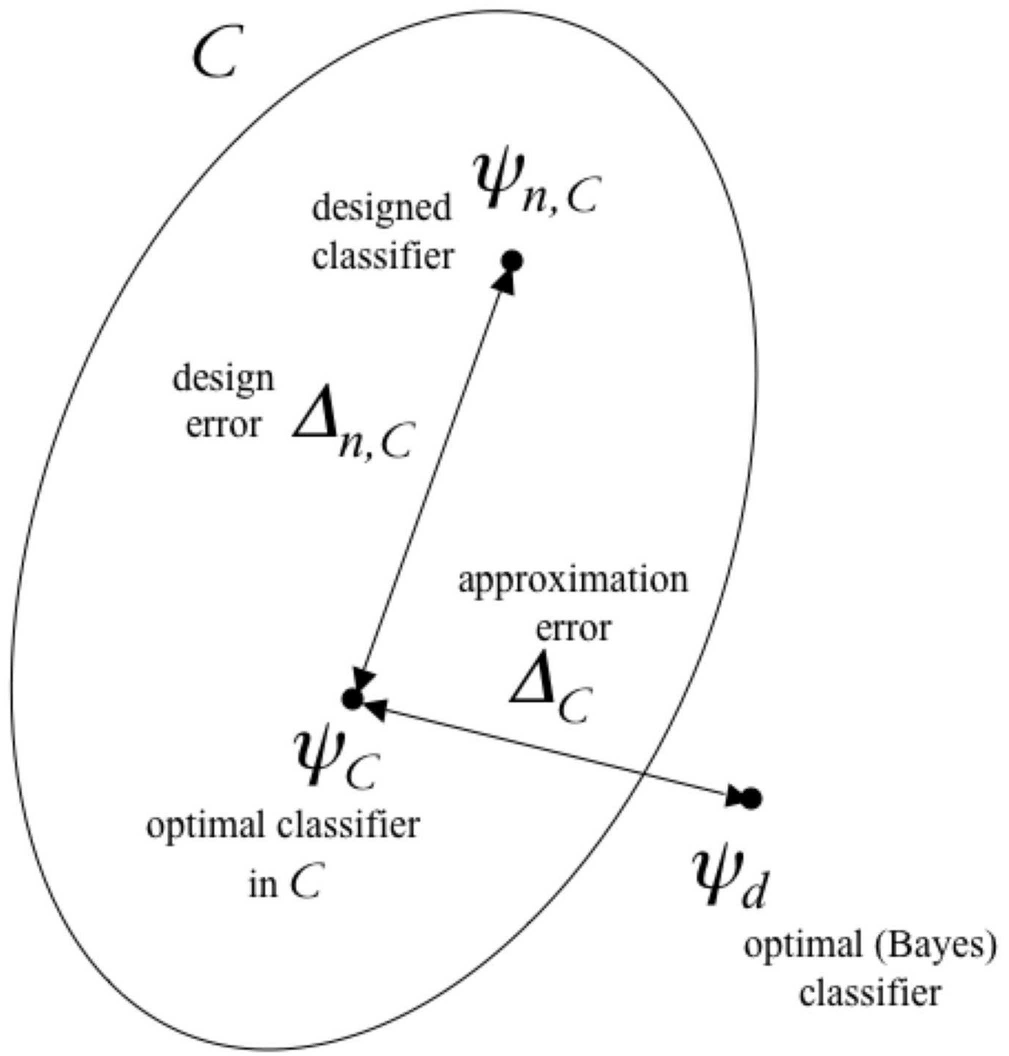
- In practice, what concerns us is the error of the designed classifier from the available data $\epsilon_{n,\mathcal{C}}$.
- This can be decomposed as:

$$\epsilon_{n,\mathcal{C}} = \epsilon_d + \Delta_{\mathcal{C}} + \Delta_{n,\mathcal{C}}$$

- The general performance measure for a classification rule is the expected classification error, which does not depend on the particular particular data:

$$E[\epsilon_{n,\mathcal{C}}] = \epsilon_d + \Delta_{\mathcal{C}} + E[\Delta_{n,\mathcal{C}}]$$

Graphical Representation



Complexity Dilemma

- In order to pick a $\mathcal{C}$, we want the the expected classification error $E[\epsilon_{n,\mathcal{C}}]$ to be as small as possible.
- Therefore, we would like both the approximation error $\Delta_{\mathcal{C}}$ and the expected design error $E[\Delta_{n,\mathcal{C}}]$ to be small.
- Unfortunately, that is not in general possible, due to the *complexity dilemma*.
- The complexity of a classification rule can be defined as its *expressive power*, or how finely it can cut up the feature space. That is, the size of $\mathcal{C}$.

Complexity Dilemma - II

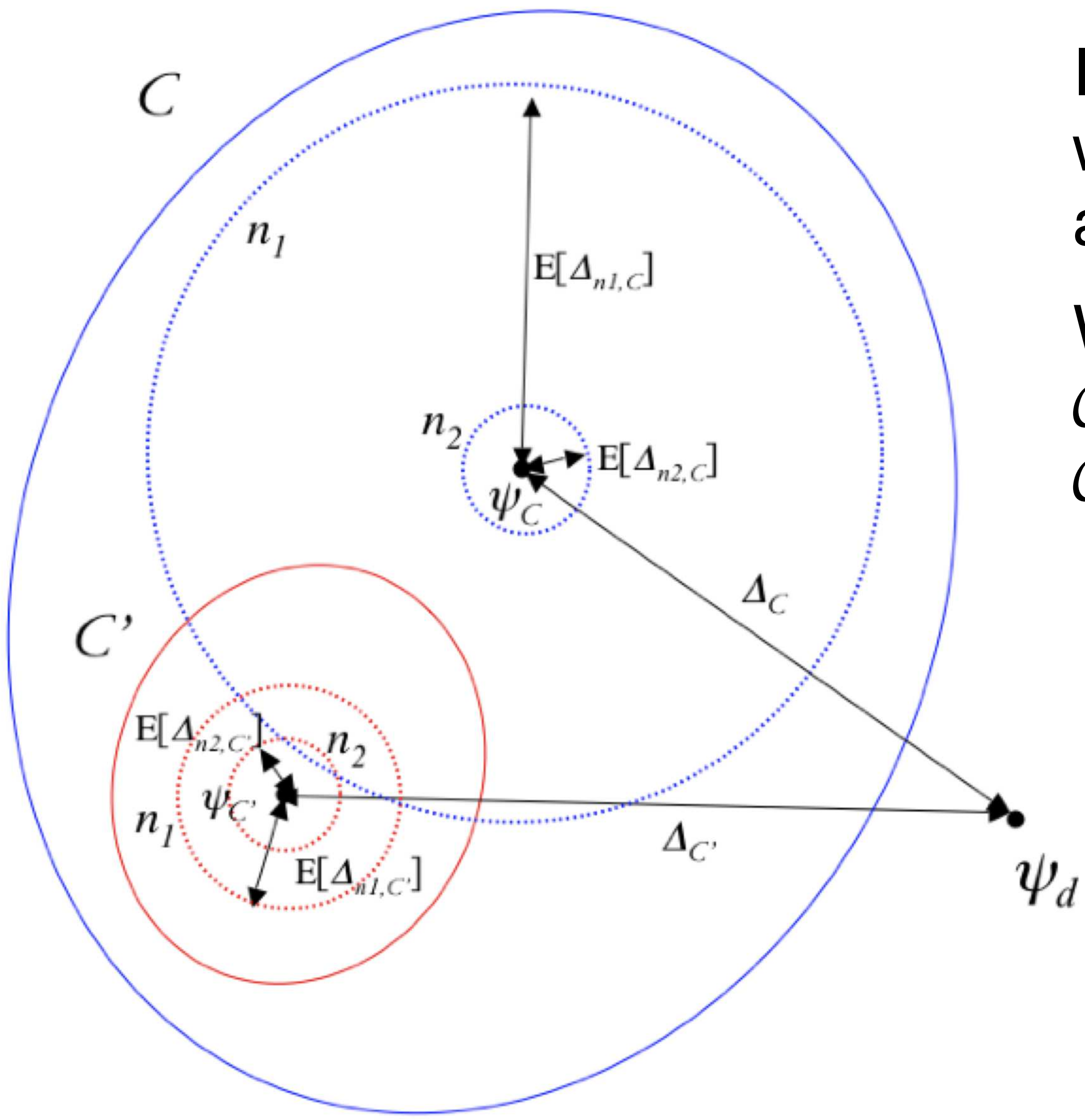
- A larger $\mathcal{C}$ (i.e., a more complex classification rule) is guaranteed to yield a smaller approximation error $\Delta_{\mathcal{C}}$. If we make $\mathcal{C}$ big enough, eventually $\psi_d \in \mathcal{C}$ and we will obtain $\Delta_{\mathcal{C}} = 0$.
- On the other hand, a smaller $\mathcal{C}$ (i.e., a simpler classification rule) will generally produce a smaller expected design error $E[\Delta_{n,\mathcal{C}}]$.
- So what do you do?
- This is a manifestation of the general bias-variance dilemma in statistics: one can generally control the bias or control the variance, but not both.
- Here, the "bias" is $\Delta_{\mathcal{C}}$ and the "variance" is $E[\Delta_{n,\mathcal{C}}]$.

Small Samples vs. Large Samples

- For large samples, the design problem is minimized ($E[\Delta_{n,\mathcal{C}}]$ tends to be small), and the most important objective is to have a small approximation error $\Delta_{\mathcal{C}}$, so in this case the use of complex classification rules is warranted.

- In small-sample settings, however, which are prevalent in many applications, the trade-off is tilted in the other direction. The design problem dominates ($E[\Delta_{n,\mathcal{C}}]$ tends to become large), and having a small approximation error $\Delta_{\mathcal{C}}$ becomes secondary. In this case the use of complex classification rules should be avoided!

Graphical Interpretation

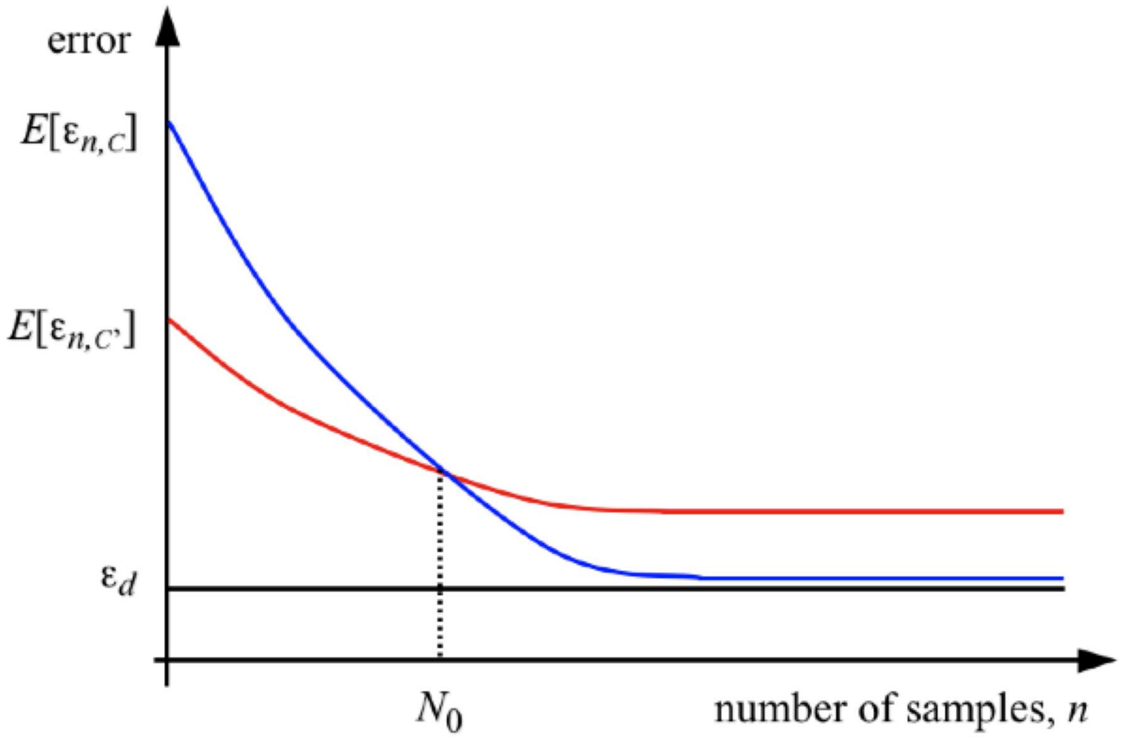


In this example, we assume $\mathcal{C}' \subset \mathcal{C}$ and $n_2 >> n_1$.

We see that $\mathcal{C}$ is better for n_2, but $\mathcal{C}'$ is better for n_1.

Expected Designed Error Plot

The dilemma is shown clearly by the plotting the expected classification error.



For $n > N_0$, the more complex class $\mathcal{C}$ is better, but for $n < N_0$, the simpler class $\mathcal{C}'$ is better.

Curse of Dimensionality

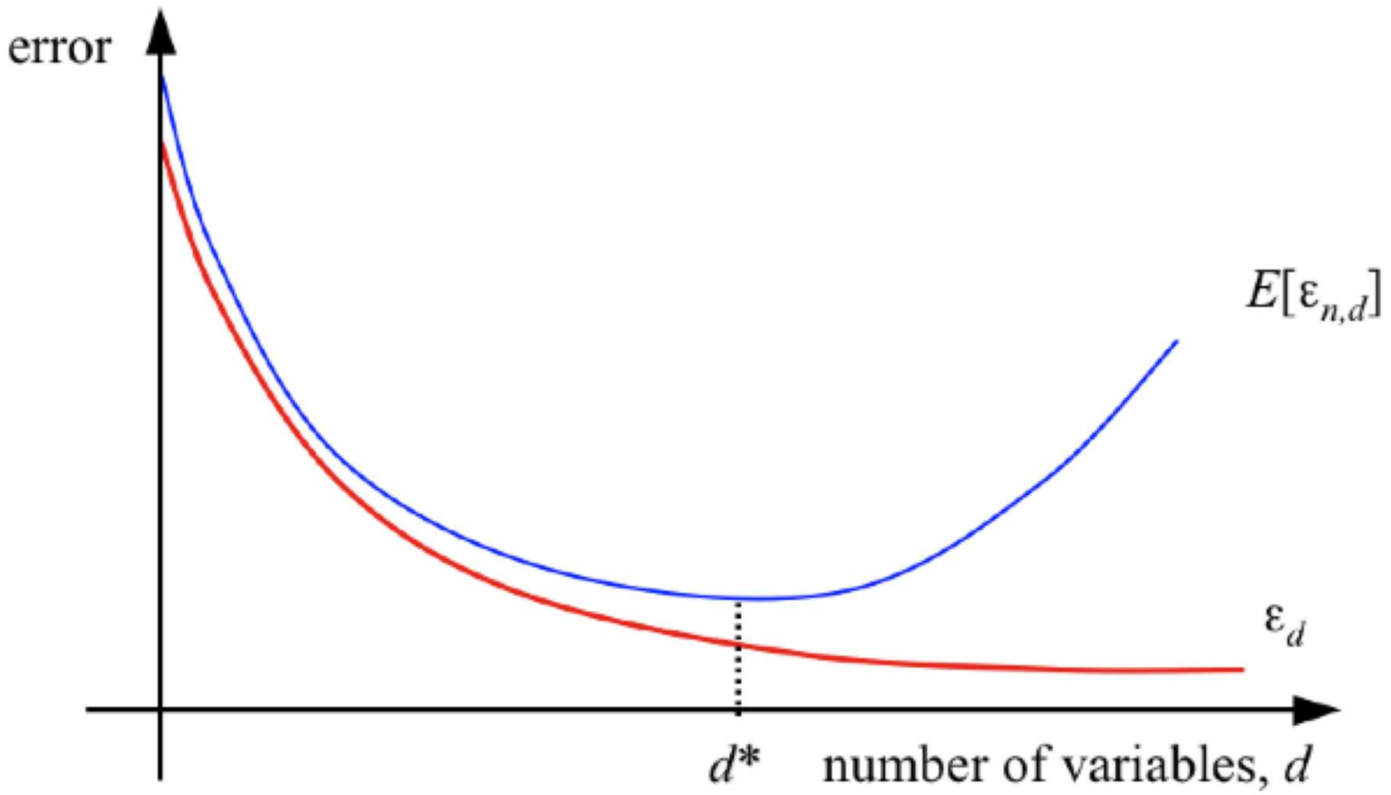
- Increasing the dimensionality d has the effect of increasing the size of $\mathcal{C}$, that is, it increases the complexity.
- For fixed n, the difference

$$E[\epsilon_{n,d}] - \epsilon_d = \Delta_d + E[\Delta_{n,d}]$$

  will generally increase as d increases, as the expected design error $E[\Delta_{n,d}]$ cannot be controlled.
- The net result is that $E[\epsilon_{n,d}]$ initially decreases (following the decrease in ϵ_d) but eventually increases.
- This phenomenon has been variously called the "curse of dimensionality," the "peaking phenomenon," and the "Hughes Phenomenon."

Curse of Dimensionality - II

This defines an *optimal* number of features d^* for a given classification rule and sample size n.



Vapnik-Chervonenkis Dimension

- The Vapnik–Chervonenkis (VC) dimension is a measure of the size, i.e., the complexity, of a class of classifiers $\mathcal{C}$.
- It agrees very naturally with our intuition of complexity as the ability of a classifier to cut up the space finely.
- Furthermore, we will see it can be used to bound, in a distribution-free manner:
  - The difference between the apparent error and the true error of a classifier in $\mathcal{C}$ (we will see that the simpler $\mathcal{C}$ is, the tighter we can bind this, which again agrees with intuition).
  - The expected design error $E[\Delta_{\mathcal{C},n}]$ if the design method is based on minimization of the apparent error.

Shatter Coefficients

- Intuitively, the complexity of a classification rule must have to do with its ability to “pick out” subsets of a given set of points.
- For a given n, consider a set of points $x_1, \ldots, x_n$ in R^d. Given a set $A \subseteq R^d$, then

$$A \cap \{x_1, \ldots, x_n\} \subseteq \{x_1, \ldots, x_n\}$$

is the subset of $\{x_1, \ldots, x_n\}$ “picked out” by A.

- Now consider a family $\mathcal{A}$ of (measurable) sets in R^d, and let

$$N_{\mathcal{A}}(x_1, \ldots, x_n) = |\{A \cap \{x_1, \ldots, x_n\} \,|\, A \in \mathcal{A}\}|$$

i.e., the total number of subsets of $\{x_1, \ldots, x_n\}$ that can be picked out by sets in $\mathcal{A}$.

Shatter Coefficients - II

- The n-th *shatter coefficient* of the family $\mathcal{A}$ is defined as

$$s(\mathcal{A}, n) = \max_{\{x_1,\ldots,x_n\}} N_{\mathcal{A}}(x_1, \ldots, x_n)$$

- The shatter coefficients $s(\mathcal{A}, n)$ measure the richness (the size, the complexity) of $\mathcal{A}$.

- Note that $s(\mathcal{A}, n) = k$ if $N_{\mathcal{A}}(x_1, \ldots, x_n) \leq k$ for all sets of n points, *and* at least one instance $\{z_1, \ldots, z_n\}$ can be found such that $N_{\mathcal{A}}(z_1, \ldots, z_n) = k$.

- Note also that $s(\mathcal{A}, n) \leq 2^n$ for all n. (why?)

The VC Dimension

- If $s(\mathcal{A}, n) = 2^n$, then there is a set of points $\{z_1, \ldots, z_n\}$ such that $N_{\mathcal{A}}(z_1, \ldots, z_n) = 2^n$, and we say that $\mathcal{A}$ *shatters* $\{z_1, \ldots, z_n\}$.
- On the other hand, if $s(\mathcal{A}, n) < 2^n$, then any set of points $\{x_1, \ldots, x_n\}$ contains at least one subset that cannot be picked out by any member of $\mathcal{A}$. In addition, we must have $s(\mathcal{A}, m) < 2^m$, for all $m > n$.
- The VC dimension $V_{\mathcal{A}}$ of $\mathcal{A}$ (assuming $|\mathcal{A}| \geq 2$) is the largest integer $k \geq 1$ such that $s(\mathcal{A}, k) = 2^k$. If $s(\mathcal{A}, n) = 2^n$ for all n, then $V_{\mathcal{A}} = \infty$.
- The VC dimension of $\mathcal{A}$ is the *maximal number of points in R^d that can be shattered by $\mathcal{A}$*.
- Clearly, $V_{\mathcal{A}}$ measures the complexity of $\mathcal{A}$!

Some Simple Examples

1. Let $\mathcal{A}$ be the class of half-lines: $\mathcal{A} = \{(-\infty, a] \mid a \in R\}$, then
$$s(\mathcal{A}, n) = n + 1 \text{ and } V_{\mathcal{A}} = 1$$
2. Let $\mathcal{A}$ be the class of intervals: $\mathcal{A} = \{[a, b] \mid a, b \in R\}$, then
$$s(\mathcal{A}, n) = \frac{n(n+1)}{2} + 1 \text{ and } V_{\mathcal{A}} = 2$$
3. Let $\mathcal{A}_d$ be the class of “half-rectangles” in R^d:
$\mathcal{A}_d = \{(-\infty, a_1] \times \cdots \times (-\infty, a_d] \mid (a_1, \ldots, a_d) \in R^d\}$,
then $V_{\mathcal{A}_d} = d$.
4. Let $\mathcal{A}_d$ be the class of rectangles in R^d:
$\mathcal{A}_d = \{[a_1, b_1] \times \cdots \times [a_d, b_d] \mid (a_1, \ldots, a_d, b_1, \ldots, b_d]) \in R^{2d}\}$,
then $V_{\mathcal{A}_d} = 2d$.

Some Observations

- In all the examples, the VC dimension is equal to the number of parameters. While this is intuitive, it is *not* true in general. In fact, one can find a one-parameter family $\mathcal{A}$ for which $V_{\mathcal{A}} = \infty$. So be careful about naively attributing complexity to the number of parameters!
- Examples 3 and 4 generalize examples 1 and 2, respectively, by means of cartesian products. It can be shown that in such cases:
$$s(\mathcal{A}_d, n) \leq s(\mathcal{A}, n)^d, \quad \text{for all } n$$
- A general bound for shatter coefficients is
$$s(\mathcal{A}, n) \leq \sum_{i=0}^{V_{\mathcal{A}}} \binom{n}{i}, \quad \text{for all } n$$

Examples 1 and 2 achieve this bound, so it is tight.

Oriented Hyperplanes

- An *oriented hyperplane* is a hyperplane plus a direction that picks one of the two half-spaces created by it.
- Any oriented hyperplane in R^d can be uniquely specified by the associated half-space $\{x \in R^d \,|\, ax \geq b\}$ for some $a \in R^d, b \in R$.
- Oriented hyperplanes correspond to linear classifiers and thus the following result is very important.
- (Corollary 13.1 DGL) Let $\mathcal{A}_d$ be the class of half-spaces in R^d: $\mathcal{A}_d = \{x \in R^d \,|\, ax \geq b, a \in R^d, b \in R\}$. Then

$$s(\mathcal{A}, n) \;=\; 2\sum_{i=0}^{d} \binom{n-1}{i}$$

and $V_{\mathcal{A}_d} = d + 1$.

Example

- For $d = 2$,

$$s(\mathcal{A}_2, 1) = 2\binom{0}{0} = 2 = 2^1$$

$$s(\mathcal{A}_2, 2) = 2\left[\binom{1}{0} + \binom{1}{1}\right] = 4 = 2^2$$

$$s(\mathcal{A}_2, 3) = 2\left[\binom{2}{0} + \binom{2}{1} + \binom{2}{2}\right] = 8 = 2^3$$

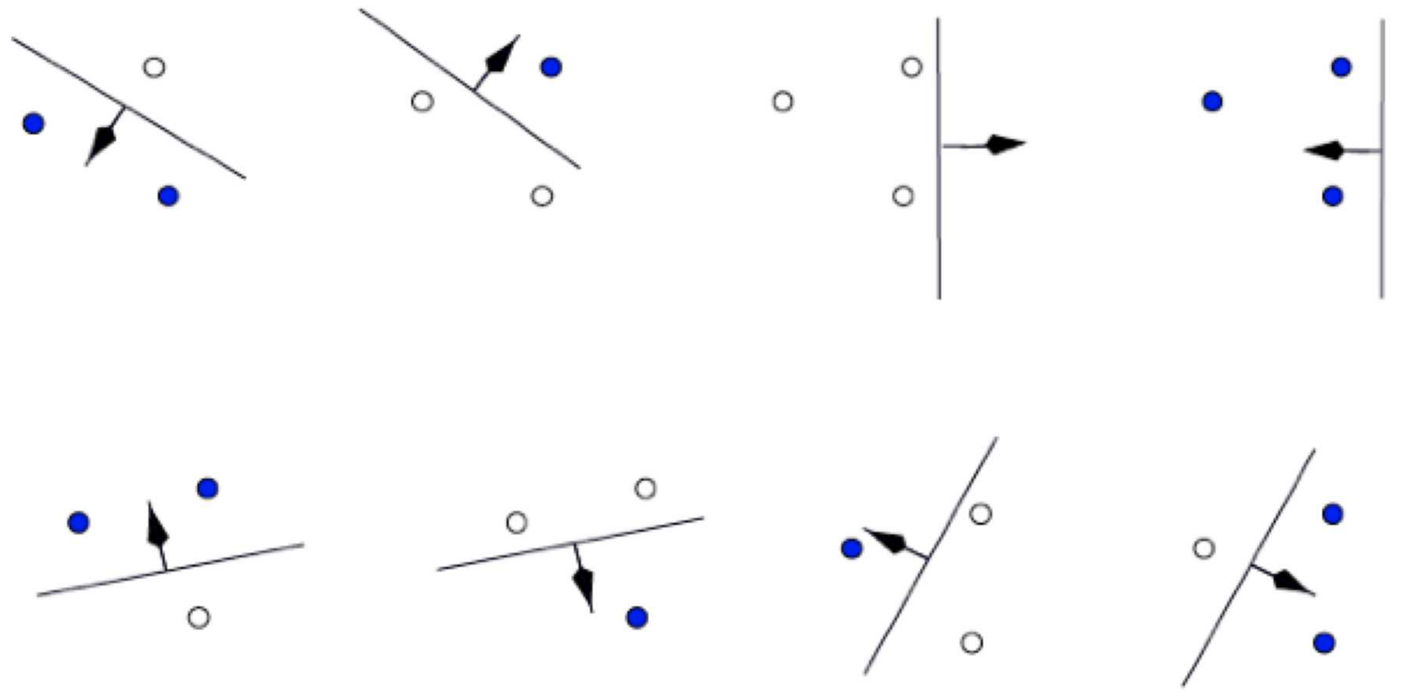
$$s(\mathcal{A}_2, 4) = 2\left[\binom{3}{0} + \binom{3}{1} + \binom{3}{2}\right] = 14 < 16 = 2^4$$

  Therefore, $V_{\mathcal{A}_2} = 3$.

- The above results imply that there is a set of 3 points that can be shattered by oriented hyperplanes in R^2, but no set of 4 points can be shattered — there are at least 2 subsets out of the $2^4 = 16$ that cannot be picked out.

Example - II

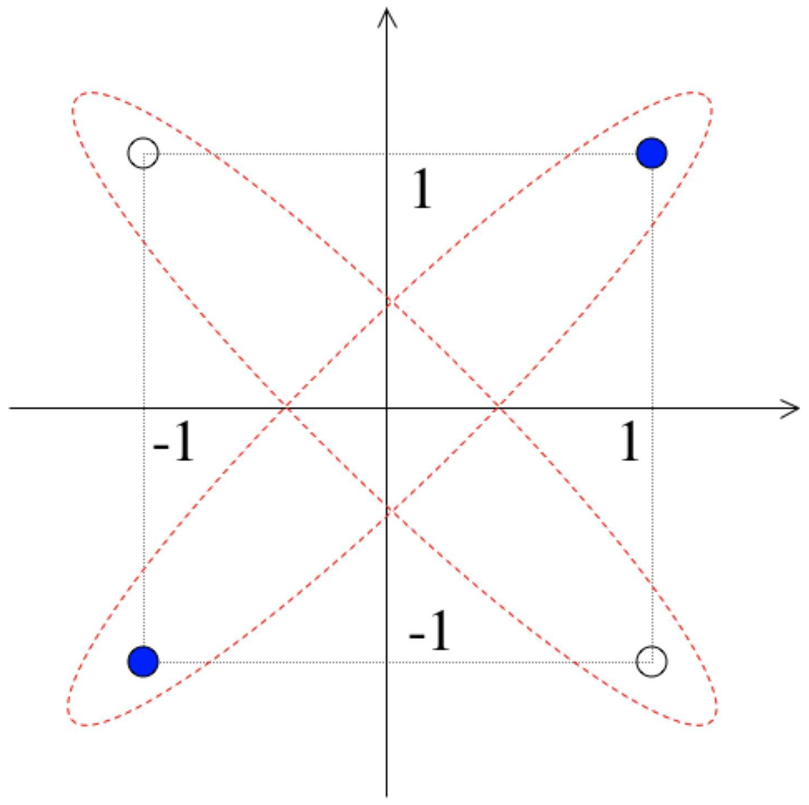
In fact, any set of 3 points (in general position) can be shattered by oriented hyperplanes in R^2.



(figure from Burges' tutorial paper)

Example - III

However, for the familiar XOR problem, each of the 2 indicated subsets of points cannot be picked out by any oriented hyperplane.



VC Theory of Classification

- The preceding concepts can be applied to define the complexity of a class $\mathcal{C}$ of classifiers (i.e., a classification rule).
- Given a classifier $\psi \in \mathcal{C}$, let us define the set $A_\psi = \{x \in R^d \mid \psi(x) = 1\}$, that is, the 1-decision region (this specifies the classifier completely, since the 0-decision region is simply A_ψ^c).
- Let $\mathcal{A}_\mathcal{C} = \{A_\psi \mid \psi \in \mathcal{C}\}$, that is, the family of all 1-decision regions produced by $\mathcal{C}$. We define the shatter coefficients $\mathcal{S}(\mathcal{C}, n)$ and VC dimension $V_\mathcal{C}$ for $\mathcal{C}$ as

$$\mathcal{S}(\mathcal{C}, n) = s(\mathcal{A}_\mathcal{C}, n)$$

$$V_\mathcal{C} = V_{\mathcal{A}_\mathcal{C}}$$

VC Parameters for Linear Classifiers

- This includes NMC, LDA, Perceptrons, and Linear SVMs.
- The results for oriented hyperplanes apply, therefore

$$\mathcal{S}(\mathcal{C}, n) = 2 \sum_{i=0}^{d} \binom{n-1}{i}$$

  with $V_{\mathcal{C}} = d + 1$.
- The VC dimension thus increases linearly with the number of variables.
- Note: the fact that all linear classification rules have the same VC dimension does not mean they will necessarily perform the same, particularly in the small-sample case.

VC Parameters for k-NN Classifiers

- For $k = 1$, clearly any set of points can be shattered (just use the points as training data).
- This is also true for any $k > 1$. Therefore, $V_{\mathcal{C}} = \infty$.
- Classes $\mathcal{C}$ with finite VC dimension are called *VC classes*. Thus, the class $\mathcal{C}_k$ of k-NN classifiers is not a VC class, for each $k > 1$.
- Classification rules that have infinite VC dimension are not necessarily awful. For example, there is evidence that 3-NN is a very decent rule, even for small-samples (We also know that the asymptotic k-NN error rate is near the Bayes error).
- However, the worst-case scenario if $V_{\mathcal{C}} = \infty$ is indeed awful, as we shall see.

VC Parameters for Decision Trees

- This applies to decision trees with data-independent splits (that is, fixed-partition tree classifiers).
- A binary tree with a depth of k levels of splitting nodes has at most $2^k - 1$ splitting nodes and at most 2^k leaves.
- Thus, for a fixed-partition tree, we have that

$$\mathcal{S}(\mathcal{C}, n) = \begin{cases} 2^n, & n \leq 2^k \\ 2^{2^k}, & n > 2^k \end{cases}$$

  and it follows that $V_\mathcal{C} = 2^k$.
- The shatter coefficients and VC dimension thus grow very fast (exponentially) with the number of levels.
- The case is different for data-dependent decision trees (e.g., CART and BSP). If stopping or pruning criteria are not strict enough, one may have $V_\mathcal{C} = \infty$ in those cases.

VC Parameters for Non-Linear SVMs

- It is easy to see that the shatter coefficients and VC dimension correspond to those of linear classification in the transformed high-dimensional space.
- More precisely, if the *minimal* space where the kernel can be written as a dot product is m, then $V_{\mathcal{C}} = m + 1$.
- For, the polynomial kernel

$$K(x, y) = (x^T y)^p = (x_1 y_1 + \cdots + x_d y_d)^p$$

  we have $m = \binom{d+p-1}{p}$, i.e., the number of distinct powers of $x_i y_i$ in the expansion of $K(x, y)$, so $V_{\mathcal{C}} = \binom{d+p-1}{p} + 1$.
- For certain kernels, such as the Gaussian kernel,

$$K(x, y) = \exp(-|x - y|^2 / \sigma^2)$$

  the minimal space is infinitely dimensional, so $V_{\mathcal{C}} = \infty$.

VC Parameters for Neural Networks

- A basic result is (DGL Thm 30.5): For the class $\mathcal{C}_k$ of neural networks with k neurons in one hidden layer, and arbitrary sigmoids,
$$V_{\mathcal{C}_k} \geq 2 \left\lfloor \frac{k}{2} \right\rfloor d$$
where $\lfloor x \rfloor$ is the largest integer $\leq x$. If k is even, this simplifies to $V_{\mathcal{C}} \geq kd$.
- For threshold sigmoids, $V_{\mathcal{C}_k} < \infty$. In fact,
$$\mathcal{S}(\mathcal{C}_k, n) \leq (ne)^{\gamma} \text{ and } V_{\mathcal{C}_k} \leq 2\,\gamma \log_2(e\gamma)$$
where $\gamma = kd + 2k + 1$ is the number of weights.
- The threshold sigmoid achieves the smallest possible VC dimension among all sigmoids.
- In fact, there are sigmoids for which $V_{\mathcal{C}_k} = \infty$ for $k \geq 2$!

Distribution-Free VC Bound

- We will see how to use $\mathcal{S}(\mathcal{C}, n)$ and $V_{\mathcal{C}}$ to bound

$$P(|\hat{\epsilon}[\psi] - \epsilon[\psi]| > \tau), \quad \text{for all } \tau > 0$$

for any classifier $\psi \in \mathcal{C}$, and *any* distribution of (X, Y), where $\hat{\epsilon}_n[\psi]$ is the *empirical error*, given data S_n:

$$\hat{\epsilon}_n[\psi] = \frac{1}{n} \sum_{i=1}^{n} |y_i - \psi(x_i)|$$

- If S_n could be assumed independent of every $\psi \in \mathcal{C}$, then $\hat{\epsilon}_n[\psi]$ would be an independent test-set error, and we could use Hoeffding's Inequality (see HW 4) to get

$$P(|\hat{\epsilon}[\psi] - \epsilon[\psi]| > \tau) \leq 2e^{-2n\tau^2}, \quad \text{for all } \tau > 0$$

Distribution-Free VC Bound - II

- *That is not sufficient.* If we want to study $|\hat{\epsilon}[\psi] - \epsilon[\psi]|$ for any distribution, and any classifier $\psi \in \mathcal{C}$, in particular a designed classifier ψ_n, we cannot assume independence from S_n.
- The solution is to bound $P(|\hat{\epsilon}[\psi] - \epsilon[\psi]| > \tau)$ *uniformly* for all possible $\psi \in \mathcal{C}$, that is, to find a (distribution-free) bound for the probability of the worst-case scenario:

$$P\left(\sup_{\psi \in \mathcal{C}} |\hat{\epsilon}[\psi] - \epsilon[\psi]| > \tau\right), \quad \text{for all } \tau > 0$$

Vapnik-Chervonenkis Theorem

- (DGL Thm 12.6) Regardless of the distribution of (X, Y),

$$P\left(\sup_{\psi\in\mathcal{C}} |\hat{\epsilon}[\psi] - \epsilon[\psi]| > \tau\right) \leq 8\mathcal{S}(\mathcal{C}, n)e^{-n\tau^2/32}, \quad \text{for all } \tau > 0$$

- If $V_{\mathcal{C}}$ is finite, we can use the inequality $\mathcal{S}(\mathcal{C}, n) \leq (n+1)^{V_{\mathcal{C}}}$ to write the bound in terms of $V_{\mathcal{C}}$:

$$P\left(\sup_{\psi\in\mathcal{C}} |\hat{\epsilon}[\psi] - \epsilon[\psi]| > \tau\right) \leq 8(n+1)^{V_{\mathcal{C}}} e^{-n\tau^2/32}, \quad \text{for all } \tau > 0$$

- Therefore, if $V_{\mathcal{C}}$ is finite, the term $e^{-n\tau^2/32}$ dominates, and the bound decreases *exponentially* fast as $n \to \infty$.

Empirical Risk Minimization

- If the classification rule consists of picking the classifier in $\mathcal{C}$ that minimizes the apparent error, then a distribution-free bound can be derived for the design error $\Delta_{n,\mathcal{C}}$, in terms of the VC dimension.
- Such a method of designing classifiers is called *Empirical Risk Minimization* (ERM).
- Some of the classification rules we have considered use the ERM principle. For example: the perceptron, the (discrete and continuous) histogram classifier, and neural networks with weights picked to minimize the empirical error.

Empirical Risk Minimization - II

- (DGL Lemma 8.2) First, let us derive an important inequality for $\Delta_{n,\mathcal{C}}$:

$$\begin{aligned}\Delta_{n,\mathcal{C}} &= \epsilon_{n,\mathcal{C}} - \epsilon_{\mathcal{C}} = \epsilon_{n,\mathcal{C}} - \hat{\epsilon}_{n,\mathcal{C}} + \hat{\epsilon}_{n,\mathcal{C}} - \inf_{\psi\in\mathcal{C}} \epsilon[\psi] \\ &= \epsilon[\psi_{n,\mathcal{C}}] - \hat{\epsilon}[\psi_{n,\mathcal{C}}] + \inf_{\psi\in\mathcal{C}} \hat{\epsilon}[\psi] - \inf_{\psi\in\mathcal{C}} \epsilon[\psi] \\ &\leq \epsilon[\psi_{n,\mathcal{C}}] - \hat{\epsilon}[\psi_{n,\mathcal{C}}] + \sup_{\psi\in\mathcal{C}} |\hat{\epsilon}[\psi] - \epsilon[\psi]| \\ &\leq 2\sup_{\psi\in\mathcal{C}} |\hat{\epsilon}[\psi] - \epsilon[\psi]|\end{aligned}$$

where we used $\hat{\epsilon}_{n,\mathcal{C}} = \inf_{\psi\in\mathcal{C}} \hat{\epsilon}[\psi]$ (because of ERM) and

$$\inf f(x) - \inf g(x) \leq \sup(f(x) - g(x)) \leq \sup |f(x) - g(x)|$$

Empirical Risk Minimization - III

- Therefore, we have

$$P(\epsilon_{n,\mathcal{C}} - \epsilon_{\mathcal{C}} > \tau) \;\leq\; P\left(\sup_{\psi \in \mathcal{C}} |\hat{\epsilon}[\psi] - \epsilon[\psi]| > \frac{\tau}{2}\right)$$

- We can now use the VC Theorem to get:

$$P(\epsilon_{n,\mathcal{C}} - \epsilon_{\mathcal{C}} > \tau) \;\leq\; 8(n+1)^{V_{\mathcal{C}}} e^{-n\tau^2/128}, \quad \text{for all } \tau > 0$$

- This bounds, in a distribution-free manner, the design error for a classification rule based on ERM. If $V_{\mathcal{C}} < \infty$ (which we are assuming above), the design error converges to zero exponentially fast as $n \to \infty$.

Empirical Risk Minimization - IV

- We can also bound the expected design error $E[\Delta_{n,\mathcal{C}}]$, by using the following fact (easy to show): if Z is a non-negative r.v. such that $P(Z > t) \leq ce^{-2nt^2}$, for all $t > 0$ and a given $c > 0$, we have

$$E[Z] \leq \sqrt{\frac{1 + \log c}{2n}}$$

- Letting $Z = (\epsilon_{n,\mathcal{C}} - \epsilon_{\mathcal{C}})/16$, $c = 8(n+1)^{V_{\mathcal{C}}}$, and using the result from the previous slide, we obtain

$$E[\Delta_{n,\mathcal{C}}] = E[\epsilon_{n,\mathcal{C}} - \epsilon_{\mathcal{C}}] \;\leq\; 16\sqrt{\frac{V_{\mathcal{C}} \log(n+1) + 4}{2n}}$$

Lower Bound for the Design Error

- One may think that by not using the ERM principle in classifier design, one is not impacted by the VC dimension of the classification rule.
- The following negative result shows that, *independently* of how to pick ψ_n from $\mathcal{C}$, the worst-case scenario still demands one to have $n \gg V_\mathcal{C}$.
- (DGL Thm 14.5) Let $2 < V_\mathcal{C} < \infty$, and let Ω be the set of all r.v.'s (X, Y) corresponding to a given $\epsilon_\mathcal{C} = \inf_{\psi \in \mathcal{C}} \epsilon[\psi]$. Then for every classification rule associated with $\mathcal{C}$,

$$\sup_{(X,Y)\in\Omega} E[\epsilon_{n,\mathcal{C}} - \epsilon_\mathcal{C}] \geq e^{-8}\sqrt{\frac{\epsilon_\mathcal{C}(V_\mathcal{C}-1)}{24n}}$$

for $n \geq \frac{V_\mathcal{C}-1}{2\epsilon_\mathcal{C}} \max\{9, 1/(1-2\epsilon_\mathcal{C})^2\}$.

Infinite VC Dimension Case

- If $V_{\mathcal{C}} = \infty$, the lower bound from the previous slide does not hold. We get instead a worse result.
- Here, we cannot make $n \gg V_{\mathcal{C}}$, and a worst-case bound can be found that is independent of n (this means that there exists a situation where the design error cannot be reduced no matter how large n may be). This is shown by the following result.
- (DGL Thm 14.3) If $V_{\mathcal{C}} = \infty$, then for every $\delta > 0$, and every classification rule associated with $\mathcal{C}$, there is a distribution for (X, Y) with $\epsilon_{\mathcal{C}} = 0$ but

$$E[\epsilon_{n,\mathcal{C}} - \epsilon_{\mathcal{C}}] = E[\epsilon_{n,\mathcal{C}}] > \frac{1}{2e} - \delta, \text{ for all } n > 1$$

Structural Risk Minimization

- The *Structural Risk Minimization* (SRM) principle is a model selection method that balances a small apparent error against the complexity of the classification rule.
- Let us begin by rewriting the bound in the VC Theorem, doing away with the supremum and the absolute value:

$$P\left(\epsilon[\psi] - \hat{\epsilon}[\psi] > \tau\right) \leq 8(n+1)^{V_{\mathcal{C}}} e^{-n\tau^2/32}, \quad \text{for all } \tau > 0$$

  which holds for any $\psi \in \mathcal{C}$.
- Assuming $V_{\mathcal{C}} < \infty$, let the right-hand side be equal to ξ, where $0 \leq \xi \leq 1$. Solving for τ gives:

$$\tau(\xi) = \sqrt{\frac{32}{n}\left[V_{\mathcal{C}} \log(n+1) - \log\left(\frac{\xi}{8}\right)\right]}$$

Structural Risk Minimization - II

- For a given ξ, we thus have

  $$P\left(\epsilon[\psi] - \hat{\epsilon}[\psi] > \tau(\xi)\right) \leq \xi \Rightarrow P\left(\epsilon[\psi] - \hat{\epsilon}[\psi] \leq \tau(\xi)\right) \geq 1 - \xi$$

- This allows us to say that the inequality

  $$\epsilon[\psi] \leq \hat{\epsilon}[\psi] + \tau(\xi)$$

  holds with probability at least $1 - \xi$.

- But this holds for any $\psi \in \mathcal{C}$ (that was the whole point of the VC Theorem), therefore we can in particular let $\psi = \psi_{n,\mathcal{C}}$, a designed classifier from S_n.

Structural Risk Minimization - III

- With $\psi = \psi_{n,\mathcal{C}}$, we have $\epsilon[\psi] = \epsilon_{n,\mathcal{C}}$, the designed classifier error, and $\hat{\epsilon}[\psi] = \hat{\epsilon}_{n,\mathcal{C}}$, the apparent error (i.e., the resubstitution error), and we can write that

$$\epsilon_{n,\mathcal{C}} \leq \hat{\epsilon}_{n,\mathcal{C}} + \sqrt{\frac{32}{n}\left[V_{\mathcal{C}} \log(n+1) - \log\left(\frac{\xi}{8}\right)\right]}$$

  holds with probability at least $1 - \xi$.
- The second term on the right-hand side is the so-called *VC confidence*. The smaller this term is, the better.
- The actual form of the VC confidence may vary according to the derivation, but in any case it
  - depends only on $V_{\mathcal{C}}$ and n, for a given ξ;
  - is small if $n \gg V_{\mathcal{C}}$, and large otherwise.

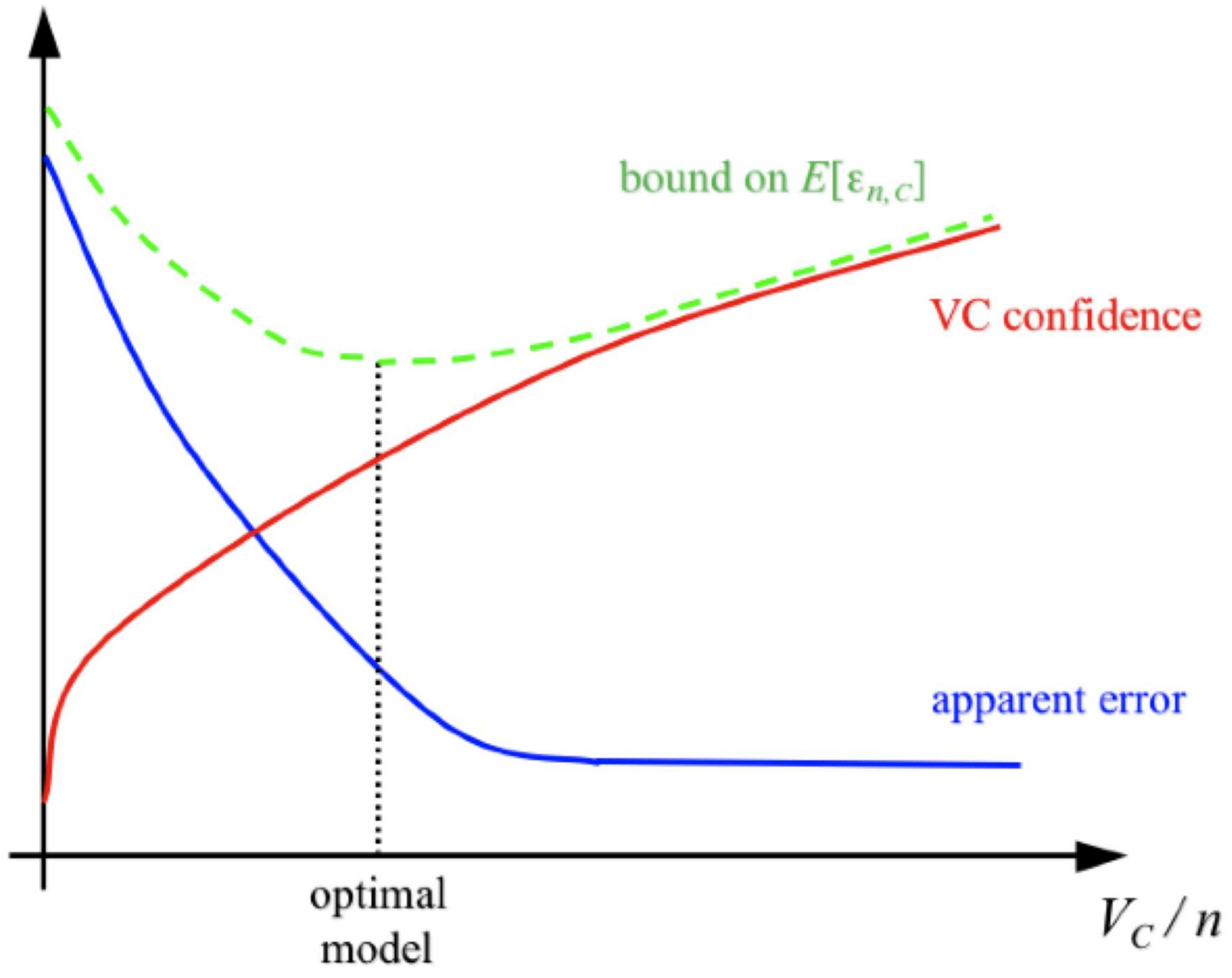
Structural Risk Minimization - IV

- Now let $\mathcal{C}^k$ be a sequence of classes associated with classification rules Ψ_n^k, for $k = 1, 2, \ldots$
- We want to be able to pick a classification rule such that the classification error $\epsilon_{n,\mathcal{C}}$ is minimal.
- From the previous slide, this can be done by picking a ξ sufficiently close to 1, computing

$$\hat{\epsilon}_{n,\mathcal{C}^k} + \sqrt{\frac{32}{n}\left[V_{\mathcal{C}^k} \log(n+1) - \log\left(\frac{\xi}{8}\right)\right]}$$

  for each k, and then picking k such that this is minimal.
- So we will want to minimize $\hat{\epsilon}_{n,\mathcal{C}^k}$, but will penalize large $V_{\mathcal{C}}$ compared to n.

Graphical Representation of SRM



Some Words of Caution

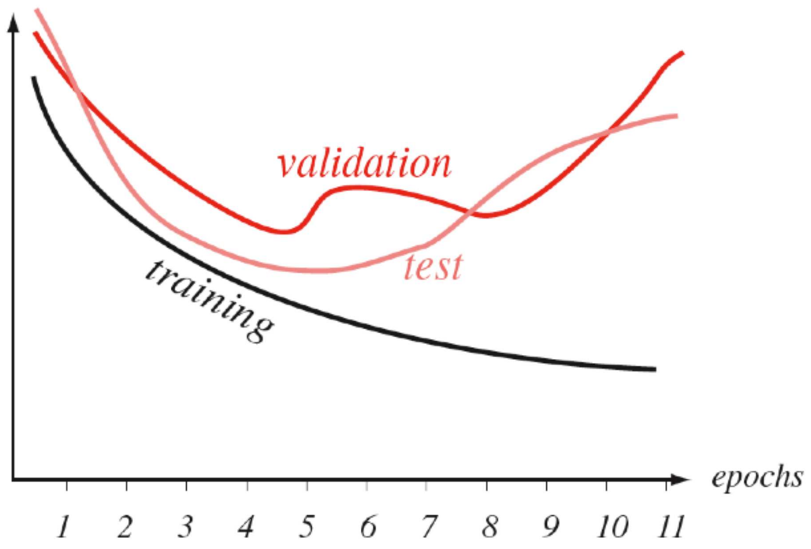
- The VC theory provides a powerful method to obtain distribution-free performance guarantees.
- However, its approach is the minimax, or worst-case, approach, because it has to do with universal performance guarantees, that is, for any distribution of (X, Y).
- Therefore, the bounds derived in the theory, though tight with respect to worst-case scenarios, tend to be slack in practice, particularly with small sample sizes.

Other Model Selection Methods

- SRM works for classification rules with finite VC dimension. We also have the caveats associated with VC bounds.
- We may not want to be restricted by this. We mention alternative approaches below.
- *Minimum-Description-Length* (MDL) methods replace error minimization by minimization of a sum of entropies, one relative to encoding the error and the other relative to encoding the classifier description, in an effort to balance increased error against increased model complexity.

Other Model Selection Methods - II

- In *Validation Set* methods, one designs the classifiers on the training set, and picks the one that produces the smallest error count on a set of independent samples, called the *validation set*. Note that a third independent set of samples, the *test set*, would be needed to get an unbiased assessment of the true error for the selected classifier. For example, this approach is commonly used to decide when to stop training a neural network.



(DHS Figure 6.6)

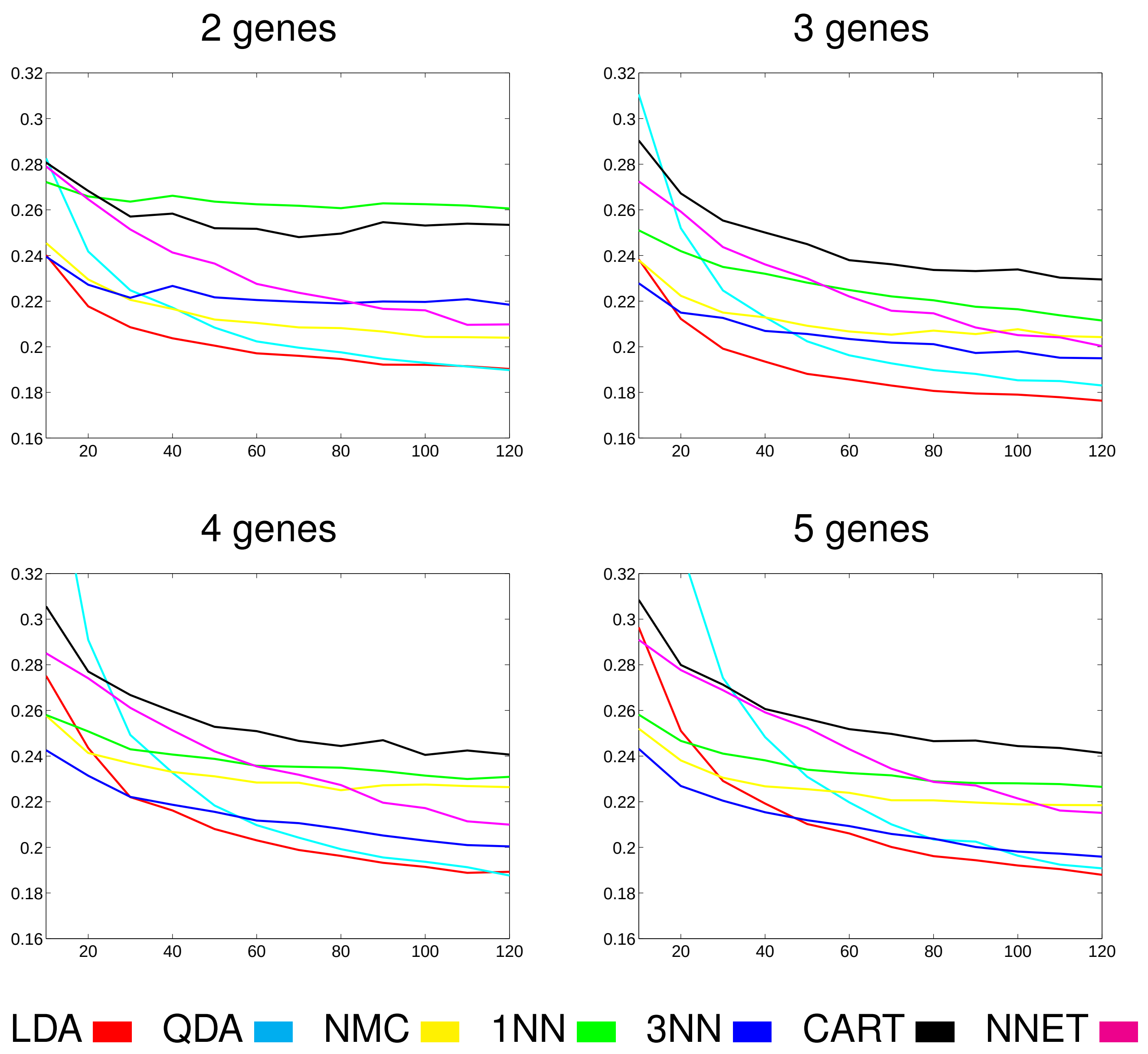
Other Model Selection Methods - III

- If there is not enough data to set aside a validation set, one may simply use a given error estimator on the training data itself to pick the classifier. This approach is only recommended if the error estimator has very good properties for the classification rules and sample size under consideration (e.g., simply using the apparent error will usually not work).
- Cross-validation is a popular choice here, because of its unbiasedness, however one has to keep in mind its large variance, especially in accentuated small-sample situations.

Simulation Example

- The relative merits of complexity and sample size with regards to classification error can be illustrated by means of simulation.
- Using the familiar cancer data from van der Vijvner et al:
  - A 1000 random subsets of size n ranging from 10 to 120 are drawn from the original 295 samples.
  - From 2 through 5 genes among the top 70 reported are used.
  - Classifiers are designed based on several classification rules of different complexity.
  - The true error is estimated by hold-out using the remaining $295 - n$ samples.
- The plots on the next slide display expected classification error versus sample size n.

Simulation Plots



Some Observations

- Simple rules such as NMC and LDA do a good job.
- The more complex rule QDA performs well for the largest sample sizes, but its performance degrades quickly for the smallest sample sizes.
- The 3NN rule does a very good job, despite an infinite VC dimension.
- The neural network performed well for two variables, but its performance quickly degrades as the number of genes increases, due to overfitting.
- CART and 1NN perform poorly, due to severe overfitting (even with the stopping criterion used for CART).
- The performance of some rules, especially QDA, degrades as the number of variables increases.