Data Mining & Machine Learning

CS37300
Purdue University

March 20, 2023

Announcement

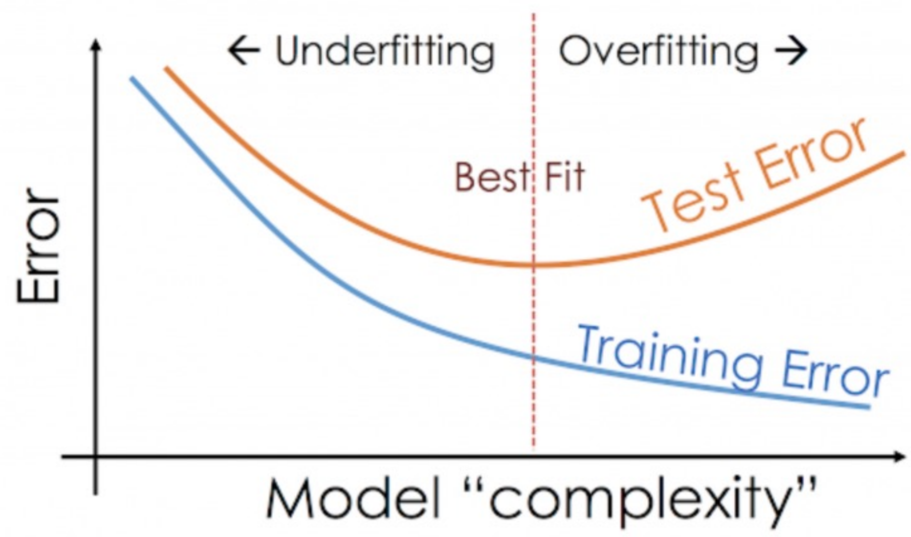
- My office hour this week is moved to **Thursday 5:00-6:00**

- Next week, it’s back to Wednesday 1:00-2:00 as usual

Today's topics

- Intro to learning theory
- Test set bound
- VC dimension
- Structural Risk Minimization

The units for model complexity



- What are the appropriate "units" for the x-axis here?

The main theoretical questions

- If an algorithm outputs a classifier with low test error rate, what can we guarantee about its true error rate?

- For a given model, if a learning algorithm achieves low training error rate, what can we guarantee about its true error rate?
    - Because of overfitting, we know the answer depends on
        - How expressive the model is (how do we measure this?)
        - How much data we have

Binary Classification

- We have a **training data set** **X** = $\{(x_1,y_1),\ldots,(x_n,y_n)\}$ of pairs (x,y)
- x: representation of the **instances** (e.g., feature vectors)
- y: the **labels**
- Let's focus on the case of **binary** labels
- Want to use **X** to **train** a classifier $\hat{h}$: function mapping x to y
- We want $\hat{h}$(x) to predict y correctly for **future** x instances (unknown)
- Called **generalization**
- We don't know the future, so we use **X** as if it is **representative** of what instances we will need to classify in the future
- Typically, we suppose future instances (x,y) have unknown distribution **P** and **X** is a sequence of i.i.d. samples with distribution **P**

Binary Classification

- We have a **training data set** **X** = $\{(x_1,y_1),\ldots,(x_n,y_n)\}$
- Want to use **X** to **train** a classifier $\widehat{h}$
- Future misclassification **error rate**:

$$\mathrm{error}_P(\hat{h}) = \Pr_{(\mathbf{x},\mathbf{y})\sim P}\left(\hat{h}(\mathbf{x}) \neq \mathbf{y}\right)$$

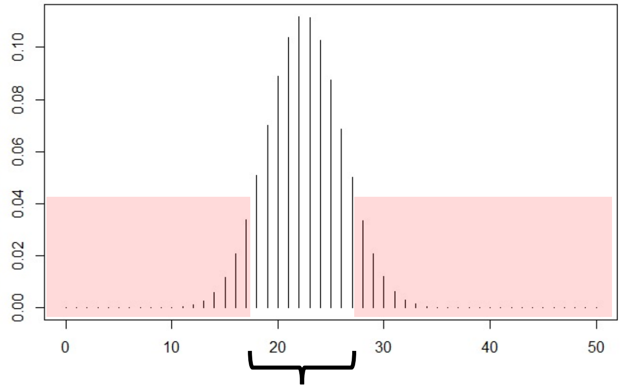
- Want small $\mathrm{error}_{\mathbf{P}}(\widehat{h})$, but don't know **P**
- But we have **X**. Define **training error rate** (a.k.a. empirical error):

$$\mathrm{error}_S(\hat{h}) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}\left[\hat{h}(x_i) \neq y_i\right]$$

- Simple idea: try to get small $\mathrm{error}_{\mathbf{X}}(\widehat{h})$
- Under some conditions, this implies small $\mathrm{error}_{\mathbf{P}}(\widehat{h})$

Test Error Concentration: Hoeffding's inequality

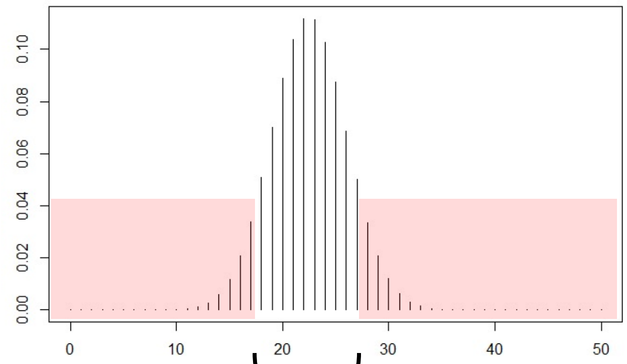
- Suppose we have a **test set** T = $\{(x_1,y_1),\ldots,(x_n,y_n)\}$ iid with distribution P
- For any particular classifier $\hat{h}$, how far is $\text{error}_T(\hat{h})$ from $\text{error}_P(\hat{h})$?
- Notice $$n \cdot \text{error}_T(\hat{h}) = \sum_{i=1}^{n} \mathbb{I}[\hat{h}(x_i) \neq y_i]$$
- Each $\mathbb{I}[\hat{h}(x_i) \neq y_i]$ is an independent {0,1}-valued random variable
- with mean equal $\Pr(\hat{h}(x_i) \neq y_i) = \text{error}_P(\hat{h})$
- So $n \cdot \text{error}_T(\hat{h})$ is a Binomial(n,p) random variable, with $p = \text{error}_P(\hat{h})$



Very low probability that the random variable is very far from its mean

Very high probability that the random variable is close to its mean

Test Error Concentration: Hoeffding's inequality



Very low probability that the random variable is very far from its mean

Very high probability that the random variable is close to its mean

- Mathematically, this is described by **Hoeffding's inequality**:
- For any $\varepsilon > 0$, $\Pr\left(\left|\text{error}_T(\hat{h}) - \text{error}_P(\hat{h})\right| > \varepsilon\right) \leq 2e^{-2\varepsilon^2 n}$
- Equivalently: For any δ with $0 < \delta \leq 1$, with probability at least 1-δ,

$$\left|\text{error}_T(\hat{h}) - \text{error}_P(\hat{h})\right| \leq \sqrt{\frac{\ln(2/\delta)}{2n}}$$

- Why? set $2e^{-2\varepsilon^2 n} = \delta$ and solve for ε

This explains why having more data gives a better estimate of $\text{error}_P(h)$

The main theoretical questions

- If an algorithm outputs a classifier with low test error rate, what can we guarantee about its true error rate?

- For a given model, if a learning algorithm achieves low training error rate, what can we guarantee about its true error rate?
  - Because of overfitting, we know the answer depends on
    - How expressive the model is (how do we measure this?)
    - How much data we have

Training Error Concentration: Uniform Convergence

- We saw how error rate on a **test set** is close to the true error rate
- But what can the **training error rate** tell us about $\text{error}_P(\hat{h})$?
- Now suppose S = $\{(x_1,y_1),\ldots,(x_n,y_n)\}$ is a **training** set $\hat{h}$ is trained on
- Can we claim

$$\left|\text{error}_S(\hat{h}) - \text{error}_P(\hat{h})\right| \leq \sqrt{\frac{\ln(2/\delta)}{2n}} \ ?$$

Training Error Concentration: Uniform Convergence

- We saw how error rate on a **test set** is close to the true error rate
- But what can the **training error rate** tell us about $\text{error}_P(\hat{h})$?
- Now suppose $S = \{(x_1,y_1),\ldots,(x_n,y_n)\}$ is a **training** set $\hat{h}$ is trained on
- Can we claim

$$\cancel{\left|\text{error}_S(\hat{h}) - \text{error}_P(\hat{h})\right| \leq \sqrt{\frac{\ln(2/\delta)}{2n}}} \ ?$$

- $\hat{h}$ is dependent on S, so the variables $\mathbb{I}[\hat{h}(x_i) \neq y_i]$ are no longer (conditionally) iid Bernoulli's with mean $\text{error}_P(\hat{h})$.
- What we need instead is called a **uniform convergence** bound
- Uniform convergence bounds account for the **model complexity**

Model Complexity

- Let $\mathcal{H}$ be the set of all classifiers the model can represent

- Called the **model space** or **hypothesis class**
  - Example: $\mathcal{H}$ might be the set of all linear separators
  - Example: $\mathcal{H}$ might be the set of all decision tree classifiers of depth ≤ 5

Uniform Convergence Bound

Theorem: For any δ with $0 < \delta \leq 1$, with probability at least 1-δ, **every h ∈ $\mathcal{H}$** has

$$\left|\text{error}_S(h) - \text{error}_P(h)\right| \leq c\sqrt{\frac{1}{n}\left(\text{VC}(\mathcal{H}) + \ln\left(\frac{1}{\delta}\right)\right)}$$

(c is a constant)

VC($\mathcal{H}$) is called the Vapnik-Chervonenkis (VC) dimension of $\mathcal{H}$

Since $\hat{h}$ is a classifier in the model class $\mathcal{H}$, we know

$$\left|\text{error}_S(\hat{h}) - \text{error}_P(\hat{h})\right| \leq c\sqrt{\frac{1}{n}\left(\text{VC}(\mathcal{H}) + \ln\left(\frac{1}{\delta}\right)\right)}$$

Compare this with the bound for test error rate:

$$\left|\text{error}_T(\hat{h}) - \text{error}_P(\hat{h})\right| \leq \sqrt{\frac{\ln(2/\delta)}{2n}}$$

VC($\mathcal{H}$) adjusts the bound to account for model complexity

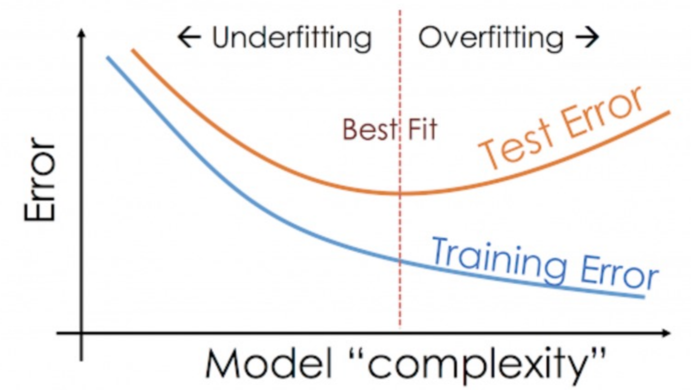
Uniform Convergence Bound

Theorem: For any δ with $0 < \delta \leq 1$, with probability at least 1-δ, **every h $\in \mathcal{H}$** has

$$\left|\text{error}_S(h) - \text{error}_P(h)\right| \leq c\sqrt{\frac{1}{n}\left(\text{VC}(\mathcal{H}) + \ln\left(\frac{1}{\delta}\right)\right)}$$

(c is a constant)

VC($\mathcal{H}$) is called the Vapnik-Chervonenkis (VC) dimension of $\mathcal{H}$



- Low VC($\mathcal{H}$): $\text{error}_S(\hat{h})$ is close to $\text{error}_P(\hat{h})$
- High VC($\mathcal{H}$): can have big gap between $\text{error}_S(\hat{h})$ and $\text{error}_P(\hat{h})$

The Vapnik-Chervonenkis (VC) dimension

- The **VC dimension** of $\mathcal{H}$, denoted VC($\mathcal{H}$), is defined as the largest number n such that there exist points $x_1,\ldots,x_n$ that can be classified in all 2^n possible ways by classifiers in $\mathcal{H}$.

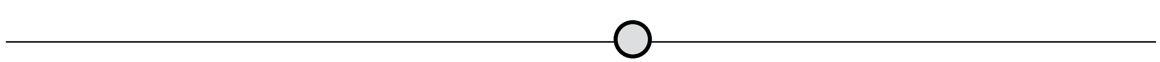
- If no such largest n exists, define VC($\mathcal{H}$)=∞.

- Any points $x_1,\ldots,x_n$ are said to be **shattered** by $\mathcal{H}$ if they can be classified in all 2^n possible ways by classifiers in $\mathcal{H}$.

- So equivalently: VC($\mathcal{H}$)= the largest size of a set shattered by $\mathcal{H}$

The Vapnik-Chervonenkis (VC) dimension

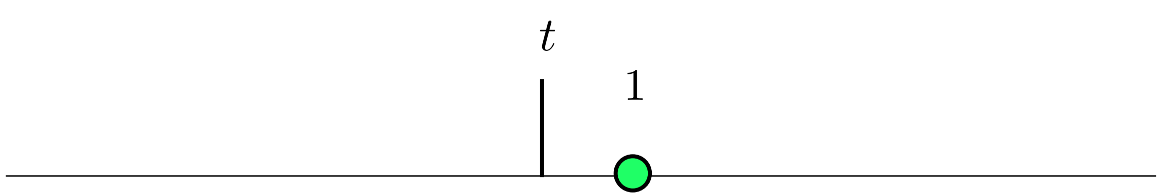
- The **VC dimension** of $\mathcal{H}$, denoted VC($\mathcal{H}$), is defined as the largest number n such that there exist points $x_1,\ldots,x_n$ that can be classified in all 2^n possible ways by classifiers in $\mathcal{H}$.
- If no such largest n exists, define VC($\mathcal{H}$)=∞.
- Any points $x_1,\ldots,x_n$ are said to be **shattered** by $\mathcal{H}$ if they can be classified in all 2^n possible ways by classifiers in $\mathcal{H}$.
- So equivalently: VC($\mathcal{H}$)= the largest size of a set shattered by $\mathcal{H}$
- Example: one-dimensional thresholds $h_t(x) = \begin{cases} 1 & \text{if } x \geq t \\ 0 & \text{if } x < t \end{cases}$



The Vapnik-Chervonenkis (VC) dimension

- The **VC dimension** of $\mathcal{H}$, denoted VC($\mathcal{H}$), is defined as the largest number n such that there exist points $x_1,\ldots,x_n$ that can be classified in all 2^n possible ways by classifiers in $\mathcal{H}$.
- If no such largest n exists, define VC($\mathcal{H}$)=∞.
- Any points $x_1,\ldots,x_n$ are said to be **shattered** by $\mathcal{H}$ if they can be classified in all 2^n possible ways by classifiers in $\mathcal{H}$.
- So equivalently: VC($\mathcal{H}$)= the largest size of a set shattered by $\mathcal{H}$
- Example: one-dimensional thresholds

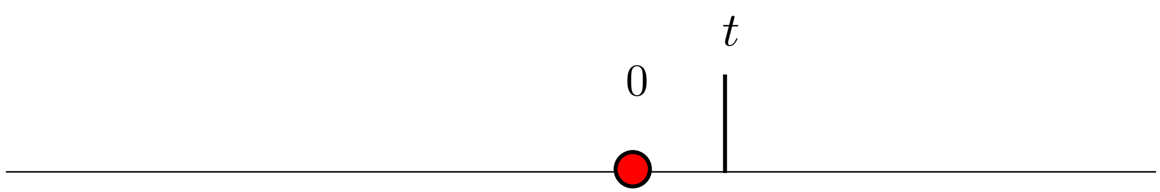
$$h_t(x) = \begin{cases} 1 & \text{if } x \geq t \\ 0 & \text{if } x < t \end{cases}$$



The Vapnik-Chervonenkis (VC) dimension

- The **VC dimension** of $\mathcal{H}$, denoted VC($\mathcal{H}$), is defined as the largest number n such that there exist points $x_1,\ldots,x_n$ that can be classified in all 2^n possible ways by classifiers in $\mathcal{H}$.
- If no such largest n exists, define VC($\mathcal{H}$)=∞.
- Any points $x_1,\ldots,x_n$ are said to be **shattered** by $\mathcal{H}$ if they can be classified in all 2^n possible ways by classifiers in $\mathcal{H}$.
- So equivalently: VC($\mathcal{H}$)= the largest size of a set shattered by $\mathcal{H}$
- Example: one-dimensional thresholds

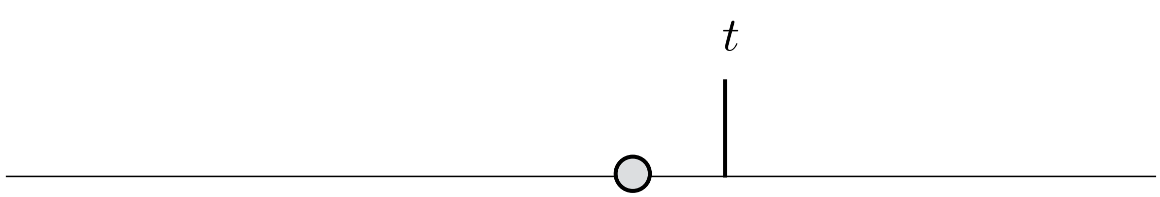
$$h_t(x) = \begin{cases} 1 & \text{if } x \geq t \\ 0 & \text{if } x < t \end{cases}$$



The Vapnik-Chervonenkis (VC) dimension

- The **VC dimension** of $\mathcal{H}$, denoted VC($\mathcal{H}$), is defined as the largest number n such that there exist points $x_1,\ldots,x_n$ that can be classified in all 2^n possible ways by classifiers in $\mathcal{H}$.
- If no such largest n exists, define VC($\mathcal{H}$)=∞.
- Any points $x_1,\ldots,x_n$ are said to be **shattered** by $\mathcal{H}$ if they can be classified in all 2^n possible ways by classifiers in $\mathcal{H}$.
- So equivalently: VC($\mathcal{H}$)= the largest size of a set shattered by $\mathcal{H}$
- Example: one-dimensional thresholds

$$h_t(x) = \begin{cases} 1 & \text{if } x \geq t \\ 0 & \text{if } x < t \end{cases}$$



Can shatter one point:
$\mathrm{VC}(\mathcal{H}) \geq 1$

The Vapnik-Chervonenkis (VC) dimension

- The **VC dimension** of $\mathcal{H}$, denoted VC($\mathcal{H}$), is defined as the largest number n such that there exist points $x_1,\ldots,x_n$ that can be classified in all 2^n possible ways by classifiers in $\mathcal{H}$.

- If no such largest n exists, define VC($\mathcal{H}$)=∞.

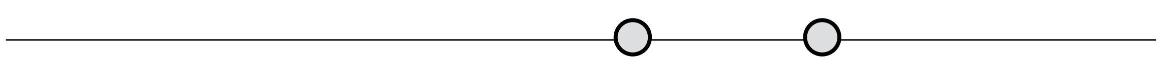
- Any points $x_1,\ldots,x_n$ are said to be **shattered** by $\mathcal{H}$ if they can be classified in all 2^n possible ways by classifiers in $\mathcal{H}$.

- So equivalently: VC($\mathcal{H}$)= the largest size of a set shattered by $\mathcal{H}$

- Example: one-dimensional thresholds

$$h_t(x) = \begin{cases} 1 & \text{if } x \geq t \\ 0 & \text{if } x < t \end{cases}$$

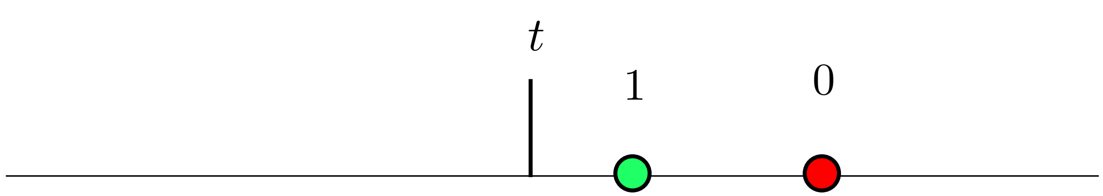
Can we shatter 2 points?



The Vapnik-Chervonenkis (VC) dimension

- The **VC dimension** of $\mathcal{H}$, denoted VC($\mathcal{H}$), is defined as the largest number n such that there exist points $x_1,\ldots,x_n$ that can be classified in all 2^n possible ways by classifiers in $\mathcal{H}$.
- If no such largest n exists, define VC($\mathcal{H}$)=∞.
- Any points $x_1,\ldots,x_n$ are said to be **shattered** by $\mathcal{H}$ if they can be classified in all 2^n possible ways by classifiers in $\mathcal{H}$.
- So equivalently: VC($\mathcal{H}$)= the largest size of a set shattered by $\mathcal{H}$
- Example: one-dimensional thresholds

$$h_t(x) = \begin{cases} 1 & \text{if } x \geq t \\ 0 & \text{if } x < t \end{cases}$$



Can we shatter 2 points?
For any 2 points,
any h_t that labels the leftmost point 1 must label the rightmost point 1 too.
Therefore, cannot realize 1, 0 labels

$\mathrm{VC}(\mathcal{H}) < 2$

The Vapnik-Chervonenkis (VC) dimension

- The **VC dimension** of $\mathcal{H}$, denoted VC($\mathcal{H}$), is defined as the largest number n such that there exist points $x_1,\ldots,x_n$ that can be classified in all 2^n possible ways by classifiers in $\mathcal{H}$.
- If no such largest n exists, define VC($\mathcal{H}$)=∞.
- Any points $x_1,\ldots,x_n$ are said to be **shattered** by $\mathcal{H}$ if they can be classified in all 2^n possible ways by classifiers in $\mathcal{H}$.
- So equivalently: VC($\mathcal{H}$)= the largest size of a set shattered by $\mathcal{H}$
- Example: one-dimensional thresholds $h_t(x) = \begin{cases} 1 & \text{if } x \geq t \\ 0 & \text{if } x < t \end{cases}$

$$\mathrm{VC}(\mathcal{H}) \geq 1 \text{ and } \mathrm{VC}(\mathcal{H}) < 2$$

$$\text{Therefore, } \mathrm{VC}(\mathcal{H}) = 1$$

The Vapnik-Chervonenkis (VC) dimension

- The **VC dimension** of $\mathcal{H}$, denoted VC($\mathcal{H}$), is defined as the largest number n such that there exist points $x_1,\ldots,x_n$ that can be classified in all 2^n possible ways by classifiers in $\mathcal{H}$.
- If no such largest n exists, define VC($\mathcal{H}$)=∞.

Example: Linear separators in $\mathbb{R}^d$ $\quad f_{\mathbf{w},b}(\mathbf{x}) = \mathrm{sign}(\mathbf{w}^\top \mathbf{x} + b)$

$$\mathrm{VC}(\mathcal{H}) = d + 1$$

Example: Decision trees of depth ≤ D

Suppose x has k features, each with values in {0,1}

Let $\mathcal{H}$ be the set of all decision tree classifiers of depth ≤ D

$$\mathrm{VC}(\mathcal{H}) \leq D2^{D+2}\log_2(k+1)$$

Sauer's Lemma

- Why is VC($\mathcal{H}$) relevant to uniform convergence?
- It bounds the **effective** number of classifiers
- Sauer's Lemma:
  For any points $x_1,\ldots,x_n$, the number of distinct ways $x_1,\ldots,x_n$ can be classified by elements of $\mathcal{H}$ is at most $n^{VC(\mathcal{H})}$

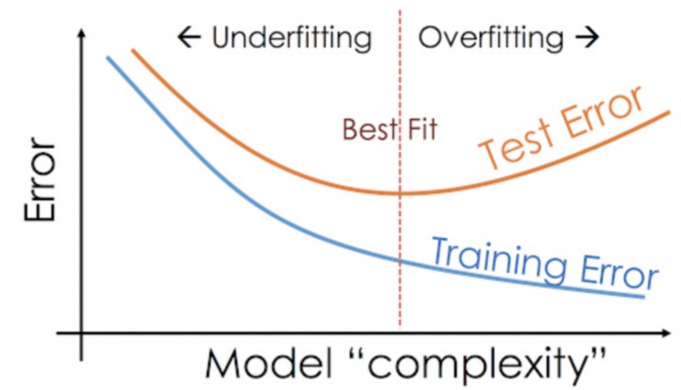
Uniform Convergence Bound

Theorem: For any δ with $0 < \delta \leq 1$, with probability at least 1-δ, **every h $\in \mathcal{H}$** has

$$\left|\mathrm{error}_S(h) - \mathrm{error}_P(h)\right| \leq c\sqrt{\frac{1}{n}\left(\mathrm{VC}(\mathcal{H}) + \ln\left(\frac{1}{\delta}\right)\right)}$$

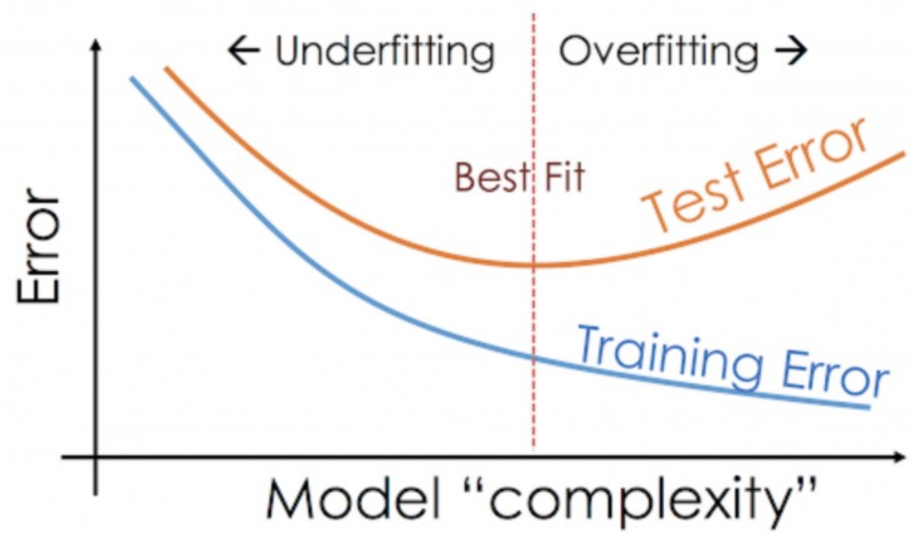
(c is a constant)

VC($\mathcal{H}$) is called the Vapnik-Chervonenkis (VC) dimension of $\mathcal{H}$



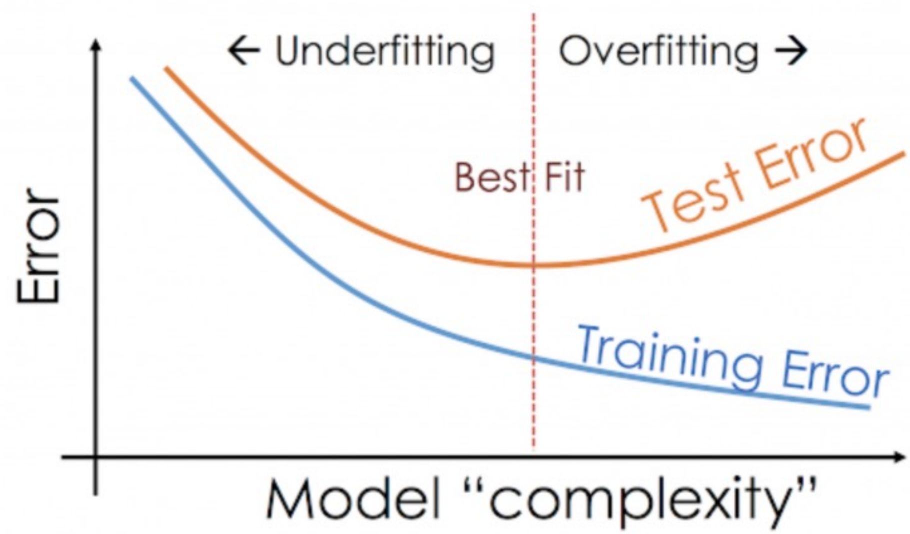
- Low VC($\mathcal{H}$): $\mathrm{error}_S(\hat{h})$ is close to $\mathrm{error}_P(\hat{h})$
- High VC($\mathcal{H}$): can have big gap between $\mathrm{error}_S(\hat{h})$ and $\mathrm{error}_P(\hat{h})$

Optimizing model complexity vs training error



- How can we **optimize** the trade-off between model complexity and training error rate?
- We want to find the "sweet spot" where the training error is small, but there is no overfitting.

Optimizing model complexity vs training error



- How can we **optimize** the trade-off between model complexity and training error rate?
- We want to find the "sweet spot" where the training error is small, but there is no overfitting.
- If there aren't too many models to pick from: cross-validation.
- Otherwise, a theoretical approach: Structural Risk Minimization

Model complexity vs training error rate

- Suppose we have a sequence of model classes

$$\mathcal{H}_1 \subset \mathcal{H}_2 \subset \mathcal{H}_3 \subset \cdots$$

- Example: $\mathcal{H}_1$ is decision trees of depth 1, $\mathcal{H}_2$ trees of depth 2, $\mathcal{H}_3$ depth 3…

- Example: $\mathcal{H}_k$ is 2-layer neural network classifiers with k hidden units

- $\mathcal{H}_k$ with larger k represents a more-complex model
    - Easier to get small training error with more-complex model
    - But also has higher VC dimension, so true error and training error aren't guaranteed to be as close
    - The theory gives us a way to optimize this trade-off

Structural Risk Minimization (SRM)

- Suppose we have a sequence of hypothesis classes

$$\mathcal{H}_1 \subset \mathcal{H}_2 \subset \mathcal{H}_3 \subset \cdots$$

- For each k, let $\hat{h}_k$ be a classifier trained based on model k: i.e., $\hat{h}_k \in \mathcal{H}_k$

Theorem: For any δ with $0 < \delta \leq 1$, with probability at least 1-δ,
for every k,

$$\operatorname{error}_P(\hat{h}_k) \leq \operatorname{error}_S(\hat{h}_k) + c\sqrt{\frac{1}{n}\left(\operatorname{VC}(\mathcal{H}_k) + \ln\left(\frac{2k^2}{\delta}\right)\right)}$$

$\mathcal{H}_k$ with larger k represents a more-complex model: **Higher VC($\mathcal{H}_k$)**

- Easier to get small training error with more-complex model:
  - **Smaller error$_S(\hat{h}_k)$**

- But also has higher VC dimension, so true error and training error aren't guaranteed to be as close
  - **Larger** $c\sqrt{\frac{1}{n}\left(\operatorname{VC}(\mathcal{H}_k) + \ln\left(\frac{2k^2}{\delta}\right)\right)}$

Structural Risk Minimization (SRM)

- Suppose we have a sequence of hypothesis classes

$$\mathcal{H}_1 \subset \mathcal{H}_2 \subset \mathcal{H}_3 \subset \cdots$$

- For each k, let $\hat{h}_k$ be a classifier trained based on model k: i.e., $\hat{h}_k \in \mathcal{H}_k$

Theorem: For any δ with $0 < \delta \leq 1$, with probability at least 1-δ, **for every k**,

$$\mathrm{error}_P(\hat{h}_k) \leq \mathrm{error}_S(\hat{h}_k) + \boxed{F(VC(k), n, \delta)}$$

The principle of **Structural Risk Minimization** (SRM):

Minimize the upper bound on the true error rate

Let $\hat{k}$ be the value of k that **minimizes**

$$\mathrm{error}_S(\hat{h}_k) + \boxed{F(VC(k), n, \delta)}$$

Optimizes the trade-off between training error and model complexity

Output: $\hat{h}_{\hat{k}}$

Structural Risk Minimization (SRM)

SRM outputs h that minimizes

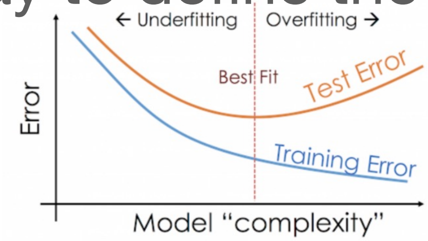
$$\mathrm{error}_S(h) + r(h)$$

This is just **regularization** (e.g., recall SVM was like this too)

This shows us that regularization and model selection are related

Summary

- Hoeffding's inequality relates test error rate to true error rate
- Uniform convergence bounds relate training error rate to true error rate, if the amount of data is large compared to the VC dimension
- VC dimension is a way to define the "units" for model complexity



- Structural Risk Minimization (SRM) is a principle for optimizing the trade-off between model complexity and training error rate, by choosing the model complexity that minimizes an upper bound on the true error rate