



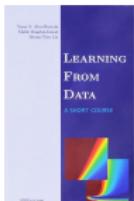
MSML610: Advanced Machine Learning

Machine Learning Theories

Instructor: Dr. GP Saggese - gsaggese@umd.edu

References:

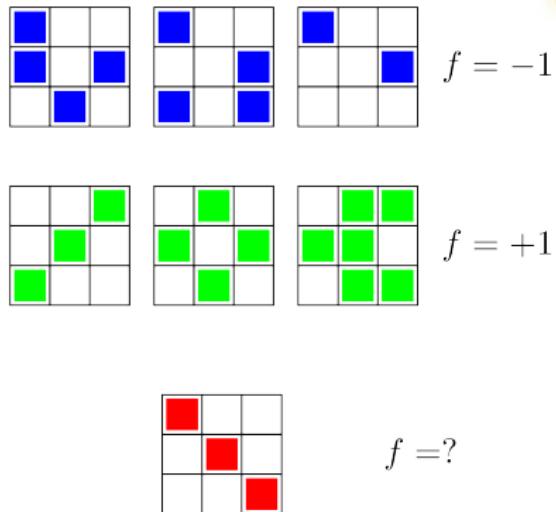
- Abu-Mostafa et al.: “*Learning From Data*” (2012)



- ***Is Machine Learning Even Possible?***
- Growth Function
- The VC Dimension
- Overfitting
- Bias Variance Analysis
- Learning Curves
- Learn-Validation Approach

A Simple Visual ML Experiment (1/2)

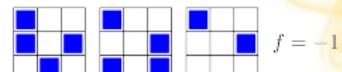
- Consider the supervised classification problem
- Input**
 - A 9 bit vector represented as a 3x3 array
- Training set**
 - The blue row $\underline{x}_1, \underline{x}_2, \underline{x}_3$ for $f(\underline{x}) = -1$
 - The green row $\underline{x}_4, \underline{x}_5, \underline{x}_6$ for $f(\underline{x}) = +1$
- Test set**
 - For the red pattern \underline{x}_0 , is $f(\underline{x}_0) = -1$ or $+1$?



A Simple Visual ML Experiment (2/2)

- **Model 1**

- $f(\underline{x}) = +1$ when \underline{x} has an axis of symmetry
- $f(\underline{x}) = -1$ when \underline{x} is not symmetric
- The test set is symmetrical $\Rightarrow f(\underline{x}_0) = +1$



- **Model 2**

- $f(\underline{x}) = +1$ when the top left square \underline{x} is empty
- $f(\underline{x}) = -1$ when the top left square \underline{x} is full
- The test set has top left square full
 $\Rightarrow f(\underline{x}_0) = -1$



- Many functions fit the 6 training examples

- Some have a value of -1 on the test point, others +1
- Which one is it?



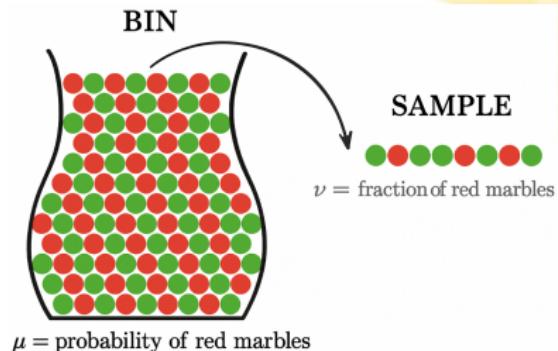
- How can a limited data set reveal enough information to define the entire target function?
 - **Is machine learning possible?**

Is Machine Learning Possible?

- The function can assume **any value outside data**
 - E.g., with summer temperature data, the function could assume a different value for winter
- **How to learn an unknown function?**
 - Estimating at unseen points seems impossible in general
 - Requires assumptions or models about behavior
- Difference between:
 - **Possible**
 - No knowledge of the unknown function
 - E.g., could be linear, quadratic, or sine wave outside known data
 - **Probable**
 - Some knowledge of the unknown function from domain knowledge or historical data patterns
 - E.g., if historical weather data forms a sinusoidal pattern, unknown points likely follow that pattern

Supervised Learning: Bin Analogy (1/2)

- Consider a bin with **red** and **green** marbles
 - We want to estimate
 $\Pr(\text{pick a red marble}) = \mu$
where the value of μ is unknown
 - We pick N marbles independently with replacement
 - The fraction of **red** marbles is ν



- Does ν say anything about μ ?
 - "**No**"
 - In strict terms, we don't know anything about the marbles we didn't pick
 - The sample can be mostly **green**, while the bin is mostly **red**
 - This is *possible*, but *not probable*
 - "**Yes**"
 - Under certain conditions, the sample frequency is close to the real frequency
- **Possible vs probable**
 - It is **possible** that we don't know anything about the marbles in the bin
 - It is **probable** that we know something
 - Hoeffding inequality makes this intuition formal

Hoeffding Inequality

- Consider a Bernoulli random variable X with probability of success μ
- Estimate the mean μ using N samples with $\nu = \frac{1}{N} \sum_i X_i$
- The **probably approximately correct** (PAC) statement holds:

$$\Pr(|\nu - \mu| > \varepsilon) \leq \frac{2}{e^{-2\varepsilon^2 N}}$$

- **Remarks:**
 - Valid for all N and ε , not an asymptotic result
 - Holds only if you sample ν and μ at random and in the same way
 - If N increases, it is exponentially small that ν will deviate from μ by more than ε
 - The bound does not depend on μ
 - Trade-off between N , ε , and the bound:
 - Smaller ε requires larger N for the same probability bound
 - Since $\nu \in [\mu - \varepsilon, \mu + \varepsilon]$, you want small ε with a large probability
 - It is a statement about ν and not μ although you use it to state something about ν (like for a confidence interval)

Supervised Learning: Bin Analogy (2/2)

- Let's connect the bin analogy, Hoeffding inequality, and feasibility of machine learning
 - You know $f(\underline{x})$ at points $\underline{x} \in \mathcal{X}$
 - You choose an hypothesis $h : \mathcal{X} \rightarrow \mathcal{Y} = \{0, 1\}$
 - Each point $\underline{x} \in \mathcal{X}$ is a marble
 - You color **red** if the hypothesis is correct $h(\underline{x}) = f(\underline{x})$, **green** otherwise
 - The in-sample error $E_{in}(h)$ corresponds to ν
 - The marbles of unknown color corresponds to $E_{out}(h) = \mu$
 - $\underline{x}_1, \dots, \underline{x}_n$ are picked randomly and independently from a distribution over \mathcal{X} which is the same as for E_{out}
- Hoeffding inequality holds and bounds the error going from in-sample to out-of-sample

$$\Pr(|E_{in} - E_{out}| > \varepsilon) \leq c$$

- Generalization over unknown points (i.e., marbles) is possible
- **Machine learning is possible!**

Validation vs Learning: Bin Analogy

- You have learned that for a given h , in-sample performance $E_{in}(h) = \nu$ needs to be close to out-of-sample performance $E_{out}(h) = \mu$
 - This is the **validation setup**, after you have already learned a model
- In a **learning setup** you have h to choose from M hypotheses
 - You need a bound on the out-of-sample performance of the chosen hypothesis $h \in \mathcal{H}$, regardless of which hypothesis you choose
 - You need a Hoeffding counterpart for the case of choosing from multiple hypotheses

$$\begin{aligned} & \forall g \in \mathcal{H} = \{h_1, \dots, h_M\} \quad \Pr(|E_{in}(g) - E_{out}(g)| > \varepsilon) \\ & \leq \Pr\left(\bigcup_{i=1}^M (|E_{in}(h_i) - E_{out}(h_i)| > \varepsilon)\right) \\ & \leq \sum_{i=1}^M \Pr(|E_{in}(h_i) - E_{out}(h_i)| > \varepsilon) \quad (\text{by the union bound}) \\ & \leq 2M \exp(-2\varepsilon^2 N) \quad (\text{by Hoeffding}) \end{aligned}$$

Validation vs Learning: Coin Analogy

- In a **validation set-up**, you have a coin and want to determine if it is fair
- Assume the coin is unbiased: $\mu = 0.5$
- Toss the coin 10 times
- How likely is that you get 10 heads (i.e., the coin looks biased $\nu = 0$)?

$$\Pr(\text{coin shows } \nu = 0) = 1/2^{10} = 1/1024 \approx 0.1\%$$

- **Conclusion:** the probability that the out-of-sample performance ($\nu = 0.0$) is completely different from the in-sample perf ($\mu = 0.5$) is very low

Validation vs Learning: Coin Analogy

- In a **learning set-up**, you have many coins and you need to choose one and determine if it's fair
- If you have 1000 fair coins, how likely is it that at least one appears totally biased using 10 experiments?
 - I.e., out-of-sample performance is completely different from in-sample performance

$$\begin{aligned}\Pr(\text{at least one coin has } \nu = 0) &= 1 - \Pr(\text{all coins have } \nu \neq 0) \\ &= 1 - (\Pr(\text{a coin has } \nu \neq 0))^{10} \\ &= 1 - (1 - \Pr(\text{a coin has } \nu = 0))^{10} \\ &= 1 - (1 - 1/2^{10})^{1000} \\ &\approx 0.63\%\end{aligned}$$

- **Conclusion:** It is probable, more than 50%

Validation vs Learning: Hoeffding Inequality

- In **validation / testing**

- Use Hoeffding to assess how well our g (the *chosen hypothesis*) approximates f (the *unknown hypothesis*):

$$\Pr(|E_{in} - E_{out}| > \varepsilon) \leq 2 \exp(-2\varepsilon^2 N)$$

where:

$$E_{in}(g) = \frac{1}{N} \sum_i e(g(\underline{x}_i), f(\underline{x}_i))$$

$$E_{out}(g) = \mathbb{E}_{\underline{x}}[e(g(\underline{x}), f(\underline{x}))]$$

- Since the hypothesis g is final and fixed, Hoeffding inequality guarantees that you can learn since it gives a bound for E_{out} to track E_{in}

- In **learning**

- Need to account that our hypothesis is the best of M hypotheses, so:

$$\Pr(|E_{in} - E_{out}| > \varepsilon) \leq 2M \exp(-2\varepsilon^2 N)$$

- The bound for E_{out} from Hoeffding is weak

- **Questions:**

- Is the bound weak because it needs to be?

- Is it possible to replace it with a stricter bound?



Intuition Why Bound for Hoeffding Is Weak

- The Hoeffding inequality and the union bound applied to training set

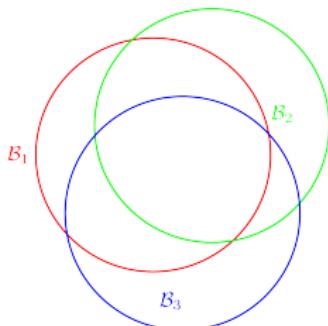
$$\Pr(|E_{in} - E_{out}| > \varepsilon) \leq 2M \exp(-2\varepsilon^2 N)$$

is **artificially** too loose

- M was coming from the bad event:

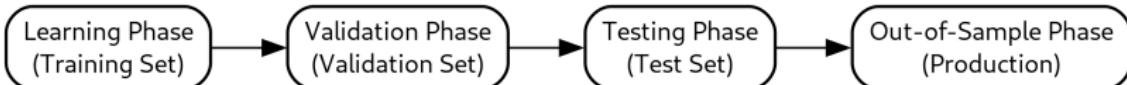
$$\begin{aligned}\mathcal{B}_i &= \text{"hypothesis } h_i \text{ does not generalize out-of-sample"} \\ &= "|E_{in}(h_i) - E_{out}(h_i)| > \varepsilon"\end{aligned}$$

- Since $g \in \{h_1, h_2, \dots, h_M\}$ then
 $\Pr(\mathcal{B}) \leq \Pr(\bigcup_i \mathcal{B}_i) \leq \sum_i \Pr(\mathcal{B}_i)$
- The union bound assumes the events are disjoint, leading to a conservative estimate if events overlap
- In reality**, bad events are extremely overlapping because bad hypotheses are extremely similar



Training vs Testing: College Course Analogy (1/2)

- In machine learning there are several phases



- This set-up is very similar to studying and exams in a college course
 - Students study the material
 - This is the **learning phase**
 - Before the final exam, students receive practice problems and solutions
 - Studying the problems improves performance by understanding what they need to improve
 - This corresponds to the **validation set**

Training vs Testing: College Course Analogy (2/2)

- The final exam corresponds to the **testing phase**
 - These problems are different than the problems in the validation set
 - Why not give out exam problems to improve performance?
 - Doing well in the exam isn't the goal
 - The goal is to learn the course material
 - The final exam isn't strictly necessary
 - Gauges how well you've learned
 - Motivates you to study
 - Knowing exam problems in advance wouldn't gauge learning effectively
- What matters is how students do once they graduate and find a job
 - This is the **out-of-sample phase**

- Is Machine Learning Even Possible?
- ***Growth Function***
- The VC Dimension
- Overfitting
- Bias Variance Analysis
- Learning Curves
- Learn-Validation Approach

Dichotomy: Definition

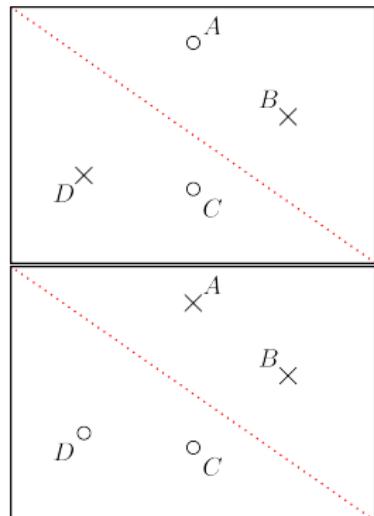
- **Problem:** classify N (fixed) points $\underline{x}_1, \dots, \underline{x}_N$ with an hypothesis set \mathcal{H} of multi-class classifiers
- Consider an assignment D of the points to certain class $\underline{d}_1, \dots, \underline{d}_N$
- D is a **dichotomy** for hypothesis set $\mathcal{H} \iff$ there exists $h \in \mathcal{H}$ that gets the desired classification D

Example

- 4 points in a plane A, B, C, D
- Binary classification
- $\mathcal{H} = \{ \text{ bidimensional perceptrons} \}$
- Moving the separating hyperplane, you get different classifications for the points (i.e., dichotomies)

	D1	D2	D3	D4	D...
A	o	x		...	
B	x	x			
C	o	o			
D	x	o		...	

- There are at most 2^N dichotomies
- Certain classifications are not possible (e.g.,



Dichotomies vs Hypotheses

- An **hypothesis** classifies each point of \mathcal{X} : $\mathcal{X} \rightarrow \{-1, +1\}$
- A **dichotomy** classifies each point of a fixed set:
 $\{\underline{x}_1, \dots, \underline{x}_N\} \rightarrow \{-1, +1\}$
 - Dichotomies are “mini-hypotheses”, i.e., hypotheses restricted to given points
 - A dichotomy depends on:
 - The number of points N
 - Hypothesis set \mathcal{H} (i.e., the possible models)
 - Where the points are placed
 - How the points are assigned
- The **number of different dichotomies** is indicated by $|\mathcal{H}(\underline{x}_1, \dots, \underline{x}_N)|$
 - The number of dichotomies is always finite, since $|\mathcal{H}(\underline{x}_1, \dots, \underline{x}_N)| \leq N^K$
 - The number of hypotheses is usually infinite, i.e., $|\mathcal{H}| = \infty$
- The “complexity” of \mathcal{H} is related to the number of hypothesis
- From the training set point of view what matters are dichotomies and not hypotheses

SCIENCE
ACADEMY Many (infinite) hypotheses can correspond to the same dichotomy



Growth Function

- The **growth function** counts the maximum number of possible dichotomies on N points for a hypothesis set \mathcal{H} :

$$m_{\mathcal{H}}(N) = \max_{\underline{x}_1, \dots, \underline{x}_N \in \mathcal{X}} |\mathcal{H}(\underline{x}_1, \dots, \underline{x}_N)|$$

- Why growth function?**

- The dichotomies depend on point distribution and assignment
- The growth function considers the maximum by placing points in the most "favorable way" for the hypothesis set
- To compute $m_{\mathcal{H}}(N)$ by **brute force**:
 - Consider all possible placements of N points $\underline{x}_1, \dots, \underline{x}_N$
 - Consider all possible assignments of the points to the classes
 - Consider all possible hypotheses $h \in \mathcal{H}$
 - Compute the corresponding dichotomy for h on $\underline{x}_1, \dots, \underline{x}_N$
 - Count the number of different dichotomies

What Can Vary in a Dichotomy

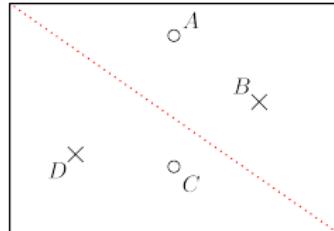
- Given:
 - An hypothesis set \mathcal{H} (e.g., bidimensional perceptrons)
 - N (fixed) points $\underline{x}_1, \dots, \underline{x}_N$
 - An assignment D of the points to certain class $\underline{d}_1, \dots, \underline{d}_N$
- D is a **dichotomy** for hypothesis set $\mathcal{H} \iff$ there exists $h \in \mathcal{H}$ that gets the desired classification D
- There are various quantities in the definition of dichotomy
 - The hypothesis set \mathcal{H}
 - It is fixed
 - The number of dimensions of the input space
 - It is fixed through the hypothesis set \mathcal{H}
 - The number of points N
 - Input to the growth function $m_{\mathcal{H}}(N)$
 - How the points are assigned to the classes $\underline{d}_1, \dots, \underline{d}_N$
 - It is a free parameter, removed by how each hypothesis in \mathcal{H} “splits” the space
 - Where the points are positioned $\underline{x}_1, \dots, \underline{x}_N$
 - It is a free parameter, removed by the growth function through max

Growth Function Is Increasing

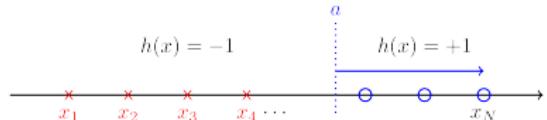
- $m_{\mathcal{H}}(N)$ increases (although not monotonically) with N
- E.g.,
 - The number of dichotomies on $N = 3$ points $m_{\mathcal{H}}(3)$ is smaller or equal than the number of dichotomies on $N = 4$ points
 - In fact we can ignore a new point and get the same classification
- $m_{\mathcal{H}}(N)$ increases with the complexity of \mathcal{H}
- $m_{\mathcal{H}}(N)$ increases with the number of dimensions in the input space (i.e., feature space)

Growth Function: Examples

- Consider the growth function $m_{\mathcal{H}}$ for different hypothesis sets \mathcal{H}
- Perceptron on a plane**
 - $m_{\mathcal{H}}(3) = 8$
 - $m_{\mathcal{H}}(4) = 14$ (2 XOR classifications not possible)

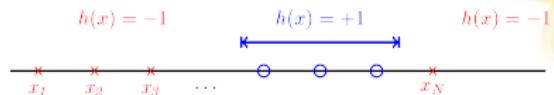


- Positive rays** $\text{sign}(x - a)$ on \mathbb{R}
 - $m_{\mathcal{H}}(N) = N + 1$
 - Origin of rays a can be placed in $N + 1$ intervals

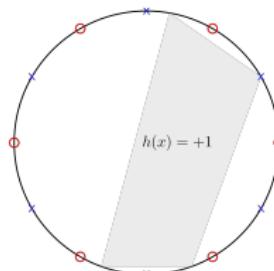


Growth Function: Examples

- **Positive intervals** on \mathbb{R} $x \in [a, b]$
 - $m_{\mathcal{H}}(N) = \binom{N+1}{2} + 1 \sim N^2$
 - Pick 2 distinct intervals out of $N+1$, and there is a dichotomy with 2 points in the same interval



- **Convex sets on a plane**
 - $m_{\mathcal{H}}(N) = 2^N$
 - Place points in a circle and can classify N points in any way



Break Point of an Hypothesis Set

- Given an hypothesis set \mathcal{H}
- A hypothesis set \mathcal{H} **shatters N points** $\iff m_{\mathcal{H}}(N) = 2^N$
 - There is a position of N points and a class assignment that you can classify using $h \in \mathcal{H}$
 - It does not mean all sets of N points can be classified in any way
- k is a **break point** for \mathcal{H} $\iff m_{\mathcal{H}}(k) < 2^k$
 - I.e., no data set of size k can be shattered by \mathcal{H}
 - E.g.,
 - For 2D perceptron: a break point is 4
 - For positive rays: a break point is 2
 - For positive intervals: a break point is 3
 - For convex set on a plane: there is no break point

Break Point for an Hypothesis Set and Learning

- If there is a break point for a hypothesis set \mathcal{H} , it can be shown that:
 - $m_{\mathcal{H}}(N)$ is polynomial in N
 - Instead of Hoeffding's inequality for learning

$$\Pr(|E_{in}(g) - E_{out}(g)| > \varepsilon) \leq 2M e^{-2\varepsilon^2 N}$$

you can use the Vapnik-Chervonenkis inequality:

$$\Pr(\text{bad generalization}) \leq 4m_{\mathcal{H}}(2N) e^{-\frac{1}{8}\varepsilon^2 N}$$

- Since $m_{\mathcal{H}}(N)$ is polynomial in N , it will be dominated by the negative exponential, given enough examples
- You can have a generalization bound: machine learning works!
- A hypothesis set can be characterized from the learning point of view by the **existence and value of a break point**

- Is Machine Learning Even Possible?
- Growth Function
- ***The VC Dimension***
- Overfitting
- Bias Variance Analysis
- Learning Curves
- Learn-Validation Approach

VC Dimension of an Hypothesis Set

- The **VC dimension of a hypothesis set** \mathcal{H} , denoted as $d_{VC}(\mathcal{H})$, is defined as the largest value of N for which $m_{\mathcal{H}}(N) = 2^N$
 - I.e., the VC dimension is the most points \mathcal{H} can shatter
- **Properties** of the VC dimension: if $d_{VC}(\mathcal{H}) = N$ then
 - Exists a constellation of N points that can be shattered by \mathcal{H}
 - Not all sets of N points can be shattered
 - If N points were placed randomly, they could not be necessarily shattered
 - \mathcal{H} can *shatter* N points for any $N \leq d_{VC}(\mathcal{H})$
 - The *smallest break point* is $d_{VC} - 1$
 - The *growth function* in terms of the VC dimension is $m_{\mathcal{H}} \leq \sum_{i=0}^{d_{VC}} \binom{N}{i}$
 - The VC dimension is the *order of the polynomial bounding* $m_{\mathcal{H}}$

VC Dimension: Interpretation

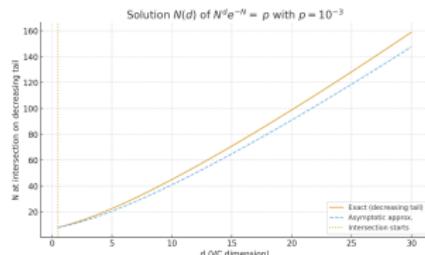
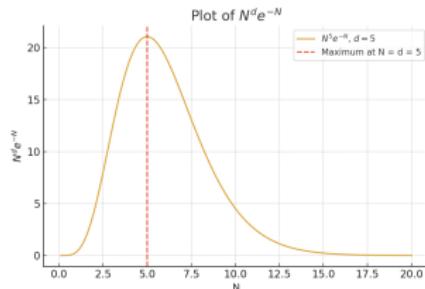
- The VC dimension **measures the complexity** of a hypothesis set in terms of **effective parameters**
- E.g.,
 - A perceptron in a d -dimensional space has $d_{VC} = d + 1$
 - In fact d_{VC} is the number of perceptron parameters!
 - E.g., for a 2D perceptron ($d = 2$), the break point is 2, so $d_{VC} = 3$
- The VC dimension considers the model as a black box in order to estimate effective parameters
 - How many points N a model can shatter, not the number of parameters
- Not all parameters contribute to degrees of freedom
 - E.g., combining N 1D perceptrons gives $2N$ parameters, but the effective degrees of freedom remain 2
- A complex hypothesis \mathcal{H} :
 - Has more parameters (higher VC dimension d_{VC})
 - Requires more examples for training

VC Generalization Bounds

- How many data points are needed to obtain $\Pr(|E_{in} - E_{out}| > \varepsilon) \leq \delta?$
- The VC inequality states

$$\Pr(\text{bad generalization}) \leq 4m_{\mathcal{H}}(2N)e^{-\frac{1}{8}\varepsilon^2 N}$$

- $N^d e^{-N}$ abstracts the upper bound term
 - Plot $N^d e^{-N}$ vs. N : Power dominates for small N , exponential for large N and brings it to 0
 - Vary d (VC dimension) function peaks for larger N , then approaches the region of interest $\leq N^1$
- Plot intersection of $N^d e^{-N}$ with a probability as a function of d
 - Examples N needed are proportional to d
 - Rule of thumb: $N \geq 10d_{VC}$ for generalization



VC Generalization Bounds

- The VC inequality

$$\Pr(|E_{in} - E_{out}| > \varepsilon) \leq 4m_{\mathcal{H}}(2N)e^{-\frac{1}{8}\varepsilon^2 N}$$

can be used in several ways to relate ε , δ , and N , e.g.,

- Examples
 - "Given $\varepsilon = 1\%$ error, how many examples N are needed to get $\delta = 0.05$?"
 - "Given N examples, what's the probability of an error larger than ε ?"
- You can equate δ to $4m_{\mathcal{H}}(2N)e^{\frac{1}{8}\varepsilon^2 N}$ and solve for ε , getting

$$\Omega(N, \mathcal{H}, \delta) = \sqrt{\frac{8}{N} \ln \frac{4m_{\mathcal{H}}(2N)}{\delta}}$$

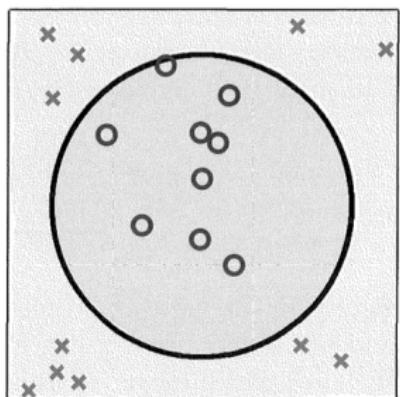
- Then you can say $|E_{out} - E_{in}| \leq \Omega(N, \mathcal{H}, \delta)$ with probability $\geq 1 - \delta$
 - The generalization bounds are then: $\Pr(E_{out} \leq E_{in} + \Omega) \geq 1 - \delta$

How to Void the VC Analysis Guarantee

- Consider the case where data is genuinely non-linear
 - E.g., “o” points in the center and “x” in the corners
- Transform to high-dimensional \mathcal{Z} with:

$$\Phi : \underline{x} = (x_0, \dots, x_d) \rightarrow \underline{z} = (z_0, \dots, z_{\tilde{d}})$$

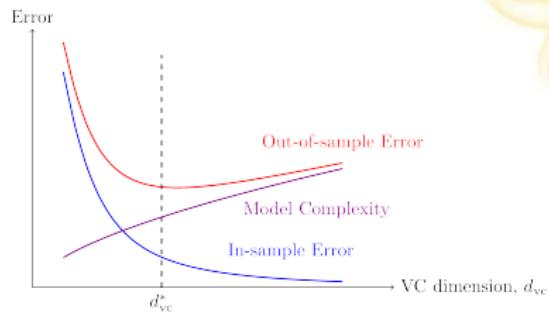
- $d_{VC} \leq \tilde{d} + 1$; smaller \tilde{d} improves generalization
 - Use $\underline{z} = (1, x_1, x_2, x_1 x_2, x_1^2, x_2^2)$
 - Why not $\underline{z} = (1, x_1^2, x_2^2)$?
 - Why not $\underline{z} = (1, x_1^2 + x_2^2)$?
 - Why not $\underline{z} = (x_1^2 + x_2^2 - 0.6)$?
- Some model coefficients were zero and discarded, leaving machine learning the rest
 - VC analysis is a warranty, forfeited if data is examined before model selection (data snooping)
 - From VC analysis, complexity is that of the initial hypothesis set



- Is Machine Learning Even Possible?
- Growth Function
- The VC Dimension
- ***Overfitting***
- Bias Variance Analysis
- Learning Curves
- Learn-Validation Approach

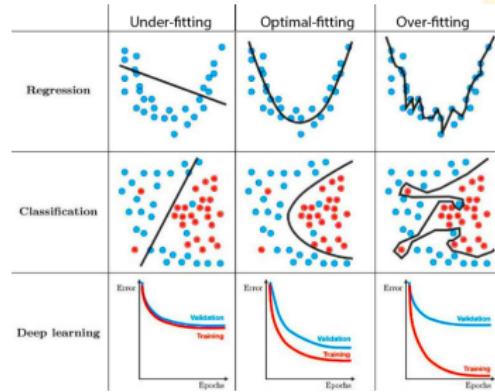
Overfitting: Definition

- **Overfitting** occurs when the model fits the data more than what is warranted
- Surpass point where E_{out} is minimal (optimal fit)
 - Model complexity too high for data/noise
 - Noise in training set mistaken for signal
- **Fitting noise instead of signal** is not useless but harmful
 - Model infers in-sample pattern that, when extrapolated out-of-sample, deviates from target function \implies poor generalization



Optimal Fit

- The opposite of overfitting is **optimal fit**
 - Train a model with proper complexity for the data
- The optimal fit:
 - Implies E_{out} is minimal
 - Does not imply generalization error $E_{out} - E_{in}$ is minimal
 - E.g., no training implies generalization error equal to 0
- The **generalization error** is the additional error $E_{out} - E_{in}$ when going from in-sample to out-of-sample



Overfitting: Diamond Price Example

- Predict diamond price as a function of carat size (regression problem)
- **True relationship**

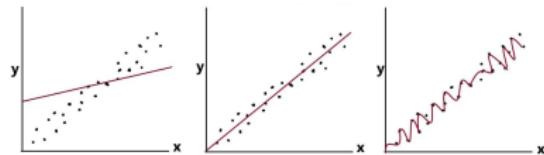
$$\text{price} \sim (\text{carat size})^2 + \varepsilon$$

where:

- Square function: price increases more with rarity
- Noise ε : e.g., market noise, missing features

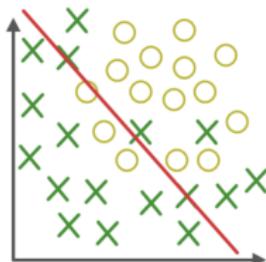
- **Fit with:**

- *Line*
 - Underfit
 - High bias (large error)
 - Low variance (stable model)
- *Polynomial of degree 2*
 - right fit
- *Polynomial of degree 10*
 - Overfit (wiggly curve)
 - Low bias
 - High variance (many degrees of freedom)

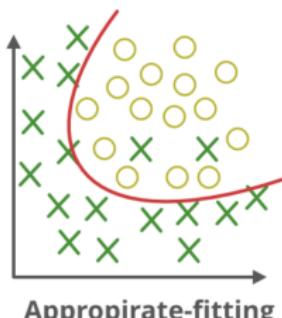


Overfitting: Classification Example

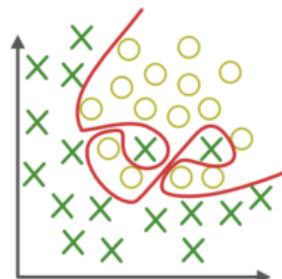
- Assume:
 - You want to separate 2 classes using 2 features x_1, x_2
 - The true class boundary has a parabola shape
- You can use logistic regression and a decision boundary equal to:
 - A line $\text{logit}(w_0 + w_1x + w_2y)$
 - Underfit
 - High bias, low variance
 - A parabola $\text{logit}(w_0 + w_1x + w_2x^2 + w_3xy + w_4y^2)$
 - Right fit
 - A wiggly decision boundary $\text{logit}(w_0 + \text{high powers of } x_1, x_2)$
 - Overfit
 - Low bias, high variance



Under-fitting



Appropriate-fitting



Over-fitting

- Is Machine Learning Even Possible?
- Growth Function
- The VC Dimension
- Overfitting
- ***Bias Variance Analysis***
- Learning Curves
- Learn-Validation Approach

VC Analysis vs Bias-Variance Analysis

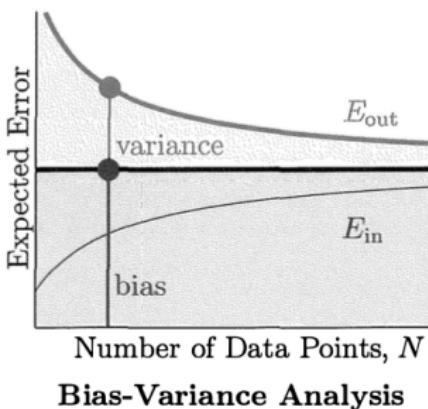
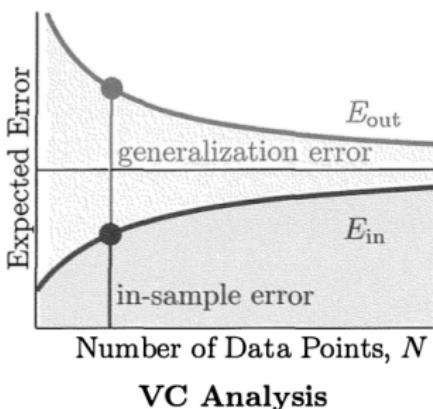
- Both VC analysis and bias-variance analysis are concerned with the hypothesis set \mathcal{H}

- VC analysis:**

$$E_{out} \leq E_{in} + \Omega(\mathcal{H})$$

- Bias-variance analysis**

$$E_{out} = \text{bias} + \text{variance}$$



Hypothesis Set and Bias-Variance Analysis

- **Learning** consists in finding $g \in \mathcal{H}$ such that $g \approx f$ where f is an unknown function
- The **tradeoff in learning** is between:
 - Bias vs variance
 - Overfitting vs underfitting
 - More complex vs less complex \mathcal{H} / h
 - Approximation (in-sample) vs generalization (out-of-sample)

Decomposing Error in Bias-Variance (1/4)

- **Problem**

- Regression set-up: target is a real-valued function
- Hypothesis set $\mathcal{H} = \{h_1(\underline{x}), h_2(\underline{x}), \dots, h_n(\underline{x})\}$
- Training data \mathcal{D} with N examples
- Squared error $E_{out} = \mathbb{E}[(g(\underline{x}) - f(\underline{x}))^2]$
- Choose the best function $g \in \mathcal{H}$ that approximates unknown f

- **Question**

- What is the out-of-sample error $E_{out}(g)$ as function of \mathcal{H} for a training set of N examples?

Decomposing Error in Bias-Variance (2/4)

- The final hypothesis g depends on training set D , so make the dependency explicit $g^{(D)}$:

$$E_{out}(g^{(D)}) \triangleq \mathbb{E}_{\underline{x}}[(g^{(D)}(\underline{x}) - f(\underline{x}))^2]$$

- Interested in:
 - Hypothesis set \mathcal{H} rather than specific h
 - Training set D of N examples, rather than a specific D
- Remove dependency from D by averaging over all possible training sets D with N examples:

$$E_{out}(\mathcal{H}) \triangleq \mathbb{E}_D[E_{out}(g^{(D)})] = \mathbb{E}_D[\mathbb{E}_{\underline{x}}[(g^{(D)}(\underline{x}) - f(\underline{x}))^2]]$$

Decomposing Error in Bias-Variance (3/4)

- Switch the order of the expectations since the quantity is non-negative:

$$E_{out}(\mathcal{H}) = \mathbb{E}_{\underline{x}}[\mathbb{E}_D[(g^{(D)}(\underline{x}) - f(\underline{x}))^2]]$$

- Focus on $\mathbb{E}_D[(g^{(D)}(\underline{x}) - f(\underline{x}))^2]$ which is a function of \underline{x}
- Define the *average hypothesis* over all training sets as:

$$\bar{g}(\underline{x}) \triangleq \mathbb{E}_D[g^{(D)}(\underline{x})]$$

- Add and subtract it inside the \mathbb{E}_D expression:

$$\begin{aligned} E_{out}(\mathcal{H}) &= \mathbb{E}_{\underline{x}} \left[\mathbb{E}_D \left[(g^{(D)}(\underline{x}) - f(\underline{x}))^2 \right] \right] \\ &= \mathbb{E}_{\underline{x}} \mathbb{E}_D[(g^{(D)} - \bar{g} + \bar{g} - f)^2] \\ &= \mathbb{E}_{\underline{x}} \mathbb{E}_D[(g^{(D)} - \bar{g})^2 + (\bar{g} - f)^2 + 2(g^{(D)} - \bar{g})(\bar{g} - f)] \\ &\quad (\mathbb{E}_D \text{ is linear and } (\bar{g} - f) \text{ doesn't depend on } D) \\ &= \mathbb{E}_{\underline{x}} [\mathbb{E}_D[(g^{(D)} - \bar{g})^2] + (\bar{g} - f)^2 + 2\mathbb{E}_D[(g^{(D)} - \bar{g})](\bar{g} - f)] \end{aligned}$$

Decomposing Error in Bias-Variance (4/4)

- The cross term:

$$\mathbb{E}_D[(g^{(D)} - \bar{g})(\bar{g} - f)]$$

disappears since applying the expectation on D , it is equal to:

$$(g^{(D)} - \mathbb{E}_D[\bar{g}])(\bar{g} - f) = 0 \cdot (\bar{g} - f) = 0 \cdot \text{constant}$$

- Finally:

$$\begin{aligned} E_{out}(\mathcal{H}) &= \mathbb{E}_{\underline{x}}[\mathbb{E}_D[(g^{(D)} - \bar{g})^2] + (\bar{g}(\underline{x}) - f(\underline{x}))^2]] \\ &= \mathbb{E}_{\underline{x}}[\mathbb{E}_D[(g^{(D)} - \bar{g})^2]] + \mathbb{E}_{\underline{x}}[(\bar{g} - f)^2] \quad (\mathbb{E}_{\underline{x}} \text{ is linear}) \\ &= \mathbb{E}_{\underline{x}}[\text{var}(\underline{x})] + \mathbb{E}_{\underline{x}}[\text{bias}(\underline{x})^2] \\ &= \text{variance} + \text{bias} \end{aligned}$$

Interpretation of Average Hypothesis

- The **average hypothesis** over all training sets

$$\bar{g}(\underline{x}) \triangleq \mathbb{E}_D[g^{(D)}(\underline{x})]$$

can be interpreted as the “best” hypothesis from \mathcal{H} training on N samples

- Note: \bar{g} is not necessarily $\in \mathcal{H}$
- In fact it’s like **ensemble learning**:
 - Consider all the possible data sets D with N samples
 - Learn g from each D
 - Average all the hypotheses

Interpretation of Variance and Bias Terms

- The out-of-sample error can be decomposed as:

$$E_{out}(\mathcal{H}) = \text{bias}^2 + \text{variance}$$

- Bias term**

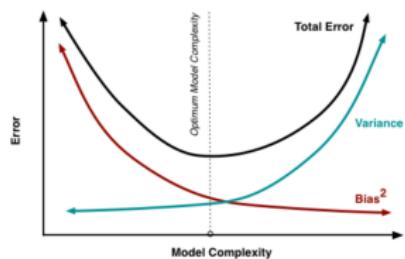
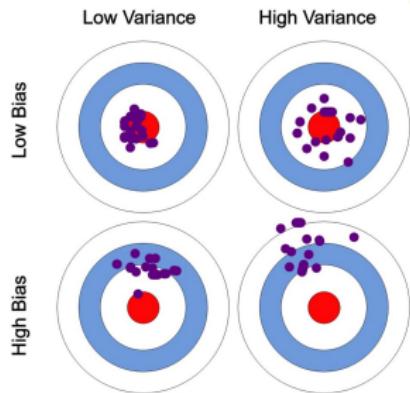
$$\text{bias}^2 = \mathbb{E}_{\underline{x}}[(\bar{g}(\underline{x}) - f(\underline{x}))^2]$$

- Does not depend on learning as it is not a function of the data set D
- Measures how limited \mathcal{H} is
 - i.e., the ability of \mathcal{H} to approximate the target with infinite training sets

- Variance term**

$$\text{variance} = \mathbb{E}_{\underline{x}} \mathbb{E}_D[(g^{(D)}(\underline{x}) - \bar{g}(\underline{x}))^2]$$

- Measures variability of the learned hypothesis from D for any \underline{x}
 - With infinite training sets, we could focus on the “best” g , which is \bar{g}
 - But we have only one data set D at a time, incurring a cost

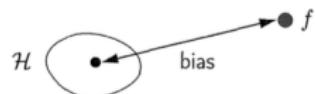


Variance and Bias Term Varying Cardinality of \mathcal{H}

- If hypothesis set **has a single function**:

$$\mathcal{H} = \{h \neq f\}$$

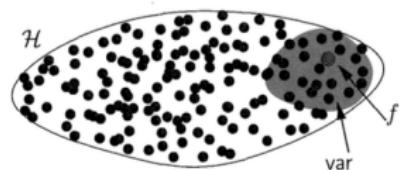
- Large bias
 - h might be far from f
- Variance = 0
 - No cost in choosing hypothesis



- If hypothesis set **has many functions**:

$$\mathcal{H} = \{\text{many hypotheses } h\}$$

- Bias can be 0
 - E.g., if $f \in \mathcal{H}$
- Large variance
 - Depending on data set D , end up far from f
 - Larger \mathcal{H} , farther g from f



Bias-Variance Trade-Off: Numerical Example

- **Machine learning problem:**
 - Target function $f(x) = \sin(\pi x)$, $x \in [-1, 1]$
 - Noiseless target
 - You have $f(\underline{x})$ for $N = 2$ points
- **Two hypotheses sets \mathcal{H} :**
 - Constant model: $\mathcal{H}_0 : h(x) = b$
 - Linear model: $\mathcal{H}_1 : h(x) = ax + b$
- **Which model is best?**
 - Depends on the perspective!
 - Best for *approximation*: minimal error approximating the sinusoid
 - Best for *learning*: learn the unknown function with minimal error from 2 points

Bias-Variance Trade-Off: Numerical Example

- **Approximation**

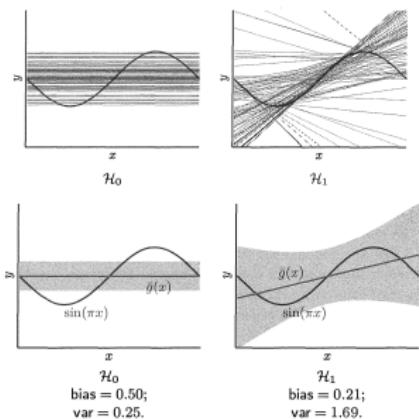
- $E_{out}(g_0) = 0.5$
 - g_0 is a constant and approximates the sinusoid poorly (higher bias)
- $E_{out}(g_1) = 0.2$
 - g_1 is a line and has more degrees of freedom (lower bias)
- The line model *approximates better* than the constant model

- **Learning**

- Algorithm:
 - Pick 2 points as training set D
 - Learn g from D
 - Compute $\mathbb{E}_D[E_{out}(g)]$
- Average over all data sets D :

$$E_{out} = \text{bias}^2 + \text{variance}$$

- $E_{out}(g_0) = 0.5 + 0.25 = 0.75$
 - g_0 is more stable given the data set (lower variance)
- $E_{out}(g_1) = 0.2 + 1.69 = 1.9$
 - g_1 heavily depends on the training set (higher variance)

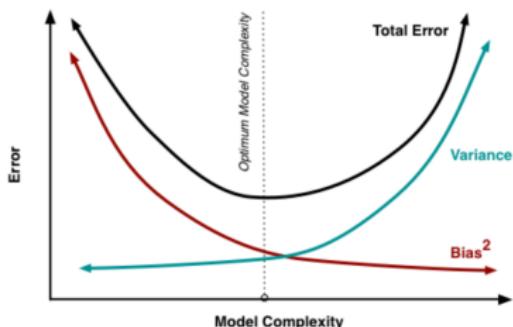
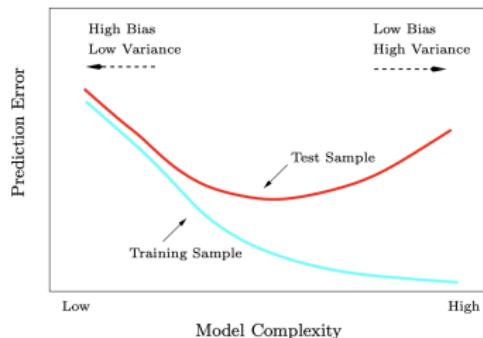


SCIENCE The constant model *learns better* than
ACADEMY the line model



Bias-Variance Curves

- **Bias-variance curves** are plots of E_{out} increasing the complexity of the model
- Typical **shape** of bias-variance curves
 - E_{in} and E_{out} start from the same point
 - E_{in}
 - Is decreasing with increasing model complexity
 - Can even go to 0
 - E_{out}
 - Is always larger than E_{in}
 - Is the sum of bias and variance
 - Has a bowl shape
 - Reaches a minimum for optimal fit
 - Before the minimum there is a “high bias / underfitting” regime
 - After the minimum there is a “high variance / overfitting” regime



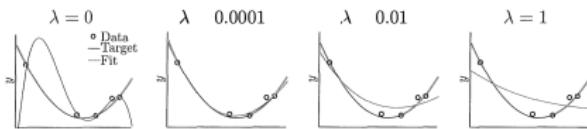
Bias-Variance Curves and Regularization

- Use **model with regularization** to learn at the same time:
 - Model coefficients \underline{w}
 - Model “complexity” (e.g., VC dimension)
- Learn the optimal model $\underline{w}(\lambda)$ as a function of $\lambda = \{\dots, 10^{-1}, 1.0, 10, \dots\}$ by optimizing:

$$\underline{w}(\lambda) = \operatorname{argmin}_{\underline{w}} E_{\text{aug}}(\underline{w}) = E_{\text{in}}(\underline{w}) + \Omega(\lambda)$$

- Interpret regularization parameter:

- Small λ :
 - Complex model
 - Low bias
 - High variance
- Large λ :
 - Simple model
 - High bias
 - Low variance
- An intermediate λ is optimal



How to Measure the Model Complexity

- Number of features
- Parameters for model form / degrees of freedom, e.g.,
 - VC dimension d_{VC}
 - Degree of polynomials
 - k in KNN
 - ν in NuSVM
- Regularization param λ
- Training epochs for neural network

Bias-Variance Decomposition with a Noisy Target

- Extend bias-variance decomposition to **a noisy target**

$$y = f(\underline{x}; \underline{w}) + \varepsilon = \underline{w}^T \underline{x} + \varepsilon$$

- With similar hypothesis and analysis conclude that:

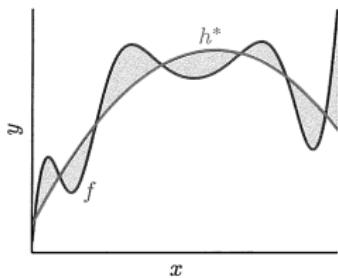
$$\begin{aligned} E_{out}(\mathcal{H}) &= \mathbb{E}_{D, \underline{x}} \left[(g^{(D)} - \bar{g})^2 \right] + \mathbb{E}_{\underline{x}} \left[(\bar{g} - f)^2 \right] + \mathbb{E}_{\varepsilon, \underline{x}} \left[(f - y)^2 \right] \\ &= \text{variance} + \text{bias} + \text{stochastic noise} \end{aligned}$$

- The out-of-sample error is the sum of 3 contributions
 - Variance**: from the set of hypotheses to the centroid of the hypothesis set
 - Bias**: from the centroid of the hypothesis set to the noiseless function
 - Noise**: from the noiseless function to the real function

Bias as Deterministic Noise

- The bias term can be interpreted as **deterministic noise**
- **Bias** is the part of the target function that hypothesis set cannot capture:

$$h^*(\underline{x}) - f(\underline{x})$$



where:

- $h^()$ is the best approximation of $f(\underline{x})$ in the hypothesis set \mathcal{H} (e.g., $\bar{g}(x)$)
- The hypothesis set \mathcal{H} cannot learn the deterministic noise since it is outside of its ability, and thus it behaves like “noise”

Deterministic vs Stochastic Noise in Practice

- **Deterministic noise:**
 - Fixed for a particular \underline{x}
 - Depends on \mathcal{H}
 - Independent of ε or D
- **Stochastic noise:**
 - Not fixed for \underline{x}
 - Independent of D or \mathcal{H}
- In a machine learning problem, no difference exists between stochastic and deterministic noise, as \mathcal{H} and D are fixed
 - From the training set alone, you cannot determine if data is from a *noiseless complex* target or a *noisy simple* target

Deterministic vs Stochastic Noise Example

- **Two targets:**

- Noisy low-order target (10th order polynomial)
- Noiseless high-order target (50th order polynomial)

- **Training set**

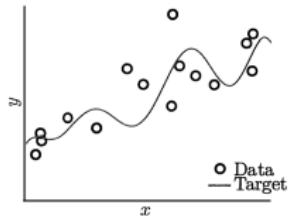
- Generate $N = 15$ data points

- **Two models**

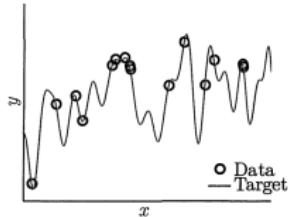
- \mathcal{H}_2 low-order hypothesis (2nd order polynomial)
- \mathcal{H}_{10} high-order hypothesis (10th order polynomial)

- Learning model:

- Sees only training samples
- Can't distinguish noise sources



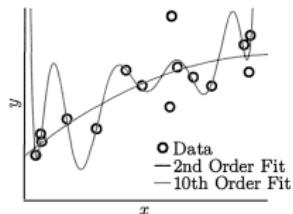
(a) 10th order target function



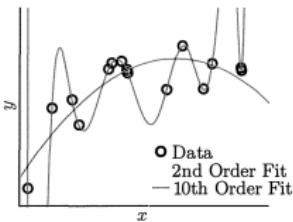
(b) 50th order target function

Deterministic vs Stochastic Noise Example

- Noisy low-order target:
 - Fit 2nd to 10th order polynomial
 - $\downarrow E_{in}$ (more degrees of freedom), $\uparrow\uparrow E_{out}$ (fits noise)
- Noiseless high-order target
 - Fit 2nd to 10th order polynomial
 - Same phenomenon
 - $\downarrow E_{in}$ (more degrees of freedom), $\uparrow\uparrow E_{out}$ (fits noise)



10th order noisy target		
	2nd Order	10th Order
E_{in}	0.050	0.034
E_{out}	0.127	9.00



50th order noiseless target		
	2nd Order	10th Order
E_{in}	0.029	10^{-5}
E_{out}	0.120	7680

degrees of freedom of model = number of data point

- Use 1 or 2 degrees of freedom



Amount of Data and Model Complexity

- One must match the *model complexity* to
 - **Data resources**
 - **Signal to noise ratio**
 - **Not target complexity**
- The rule of thumb is:

$$d_{VC}(\text{degrees of freedom of the model}) = N(\text{number of data points})/10$$

- In other words, 10 data points needed to fit a degree of freedom
- If the data is noisy, you need even more data

Overfitting as a Function of Data Resources, Model Complexity, Noise

- You can measure overfitting in terms of **generalization error** $\frac{E_{out} - E_{in}}{E_{out}}$
 - \uparrow data resources $N \implies \downarrow$ overfitting
 - \uparrow model complexity $d_{vc} \implies \uparrow$ overfitting
 - \uparrow deterministic noise (target complexity) $\implies \uparrow$ overfitting
 - \uparrow stochastic noise $\sigma^2 \implies \uparrow$ overfitting

- Is Machine Learning Even Possible?
- Growth Function
- The VC Dimension
- Overfitting
- Bias Variance Analysis
- ***Learning Curves***
- Learn-Validation Approach

Learning Curves vs Bias-Variance Curves

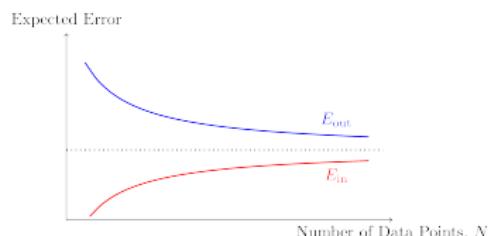
- **Learning curves** are the dual of the bias-variance curves
- For **bias-variance curves**

$$E_{in}, E_{out} = f(d_{VC}|N)$$

- Keep the complexity of training set fixed (number of examples N)
- Vary the model in terms of:
 - Model complexity d
 - Number of features p
 - Regularization amount λ
- For **learning curves**

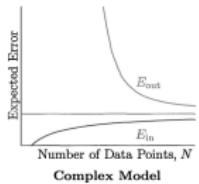
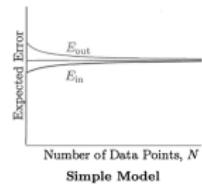
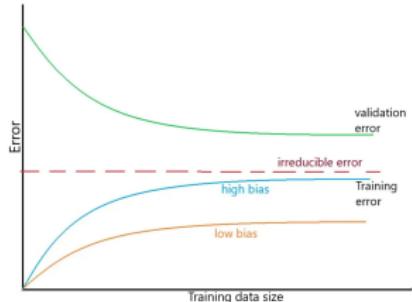
$$E_{in}, E_{out} = f(N|d_{VC})$$

- Fix the model
- Vary the size N of training set



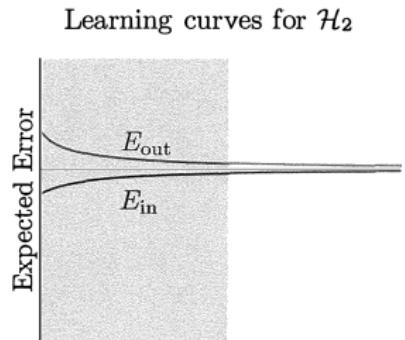
Typical Form of Learning Curves

- Learning curves plot E_{in} and E_{out} as a function of data amount N , given the model $h()$
 - $E_{out} \geq E_{in}$ for any N
- **Small N**
 - E_{in} is small (even 0)
 - The model is likely overfitted, memorizing and generalizing poorly
 - E_{out} is large
- **Increasing N**
 - $E_{in} \uparrow$ as the model cannot fit all data
 - $E_{out} \downarrow$ as the model fits better and generalizes better
 - Generalization error $E_{out} - E_{in} \downarrow$
- $N \rightarrow \infty$
 - E_{in} reaches a maximum (irreducible error)
 - E_{out} reaches a minimum
 - $E_{out} - E_{in}$ depends on the complexity of the model



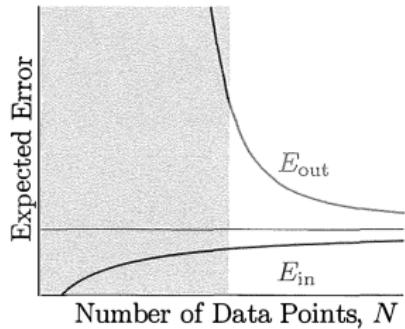
High-Bias vs High-Variance Regime

- From the learning curve you can see the two regimes:
 - High-variance regime** for small N
 - E_{in} is small
 - Small data set $D \implies$ high dependency on D
 - More data helps reduce gap between E_{in} and E_{out}
 - High-bias regime** for large N
 - E_{in} is large; flattens for large N
 - Best model fitted; more data won't help
 - E_{out} can be close to E_{in} (good generalization) or not



Number of Data Points, N

Learning curves for \mathcal{H}_{10}



Number of Data Points, N

- Is Machine Learning Even Possible?
- Growth Function
- The VC Dimension
- Overfitting
- Bias Variance Analysis
- Learning Curves
- ***Learn-Validation Approach***
 - Train / Test
 - Cross-Validation

- Is Machine Learning Even Possible?
- Growth Function
- The VC Dimension
- Overfitting
- Bias Variance Analysis
- Learning Curves
- Learn-Validation Approach
 - *Train / Test*
 - Cross-Validation

Estimating Out-Of-Sample Error with One Point

- Pick a **single out-of-sample point** (\underline{x}', y)
- The error of the model h is:

$$E_{val}(h) = e(h(\underline{x}'), y)$$

where the error can be:

- Squared error $(h(\underline{x}) - y)^2$
- Binary error $I[h(\underline{x}) - y]$
- ...
- The **error on an out-of-sample point** is an **unbiased estimate** of E_{out} , since:

$$\mathbb{E}[E_{val}(h)] = \mathbb{E}[e(h(\underline{x}), y)] = E_{out}$$

- The quality of the estimate depends on the standard error $\mathbb{V}[e(h(\underline{x}), y)]$, which is an unknown value

Estimating Out-Of-Sample Error with K Points

- To improve the estimate, use a validation set with K points $(\underline{x}_1, y_1), \dots, (\underline{x}_K, y_K)$ drawn IID
- Compute the **error on the validation set** as:

$$E_{val}(h) = \frac{1}{K} \sum_{i=1}^K e(h(\underline{x}_i), y_i)$$

- The validation error is an **unbiased estimate** of out-of-sample error since:

$$\mathbb{E}[E_{val}(h)] = E_{out}(h)$$

- The standard error is \sqrt{K} smaller:

$$\mathbb{V}[E_{val}(h)] = \frac{1}{K^2} \sum_i \mathbb{V}[e(h(\underline{x}_i), y_i)] + \text{covariances}$$

(covariances are 0 because \underline{x}_i independent)

$$\begin{aligned} &= \frac{1}{K^2} \sum_i \mathbb{V}[e(h(\underline{x}_i), y_i)] \\ &= \frac{K\sigma^2}{K^2} = \frac{\sigma^2}{K} \end{aligned}$$

Trade-Off Between Training and Validation Set

- **Problem:** you have finite amount of data points N

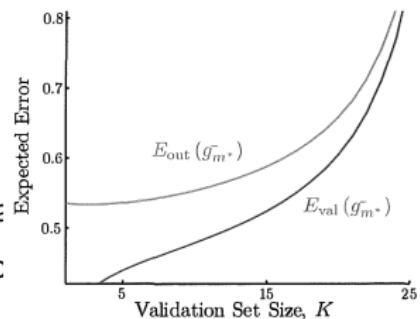
$$D_{val} = \{K \text{ points}\}$$

$$D_{train} = \{N - K \text{ points}\}$$

- You know that:

$\uparrow K \implies |E_{val} - E_{out}| \downarrow$ (most reliable estimate)

$\implies \downarrow N - K \implies E_{in}, E_{out} \uparrow$ (worse mc)



- If K is too big, you get a **reliable estimate of a bad number!**

- **Solution:**

- Rule of thumb: 70-30 or 80-20 split between train and validation

Error From VC Analysis vs Learn-Validation Approach

- In general:

$$E_{out}(h) = E_{in}(h) + \text{generalization error}$$

- **VC analysis**

- Estimates generalization error as “overfit penalty” in terms of hypothesis set complexity

- **Learn-validation**

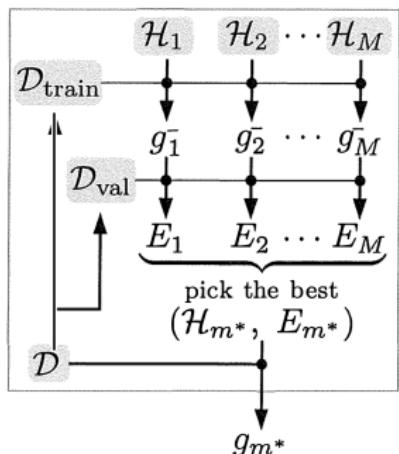
- Estimates E_{out} directly by holding out data as validation set:

$$E_{val} \approx E_{out}$$

- Use learn-validation approach at different points of the modeling flow:
 - For selecting hyperparameters (validation set)
 - To estimate final performance (test set)

Reusing Validation / Test Set for Training

- Any data used for learning (e.g., model training or model selection)
 - Is biased and optimistic
 - Cannot be used to assess the learned model
- **Never use D_{val} / D_{test} for training or D_{train} for evaluation**
- After research is done, all available data can be used to learn
- **Algorithm**
 1. Train with $N - K$ points to learn g^-
 2. Use K points to compute $E_{val}[g^-]$ estimating $E_{out}[g^-]$
 3. Once the model form is finalized, use all N data points (including validation, test set) to re-train to get g
 - $E_{val}[g] < E_{val}[g^-]$ since g is learned on a larger data set than g^- and is thus better
 4. Deliver to customers:
 - Final hypothesis g
 - Upper bound of out-of-sample performance $E_{val}[g^-]$



Learn-Validation Approach: Pros and Cons

- **Pros**
 - Estimate E_{out} using E_{val}
 - Simple to compute, no complexity from VC analysis
- **Cons**
 - Cannot use all data for learning and validation; need a compromise
 - Learned model and E_{val} depend on the split; different splits can give different results

- Is Machine Learning Even Possible?
- Growth Function
- The VC Dimension
- Overfitting
- Bias Variance Analysis
- Learning Curves
- Learn-Validation Approach
 - Train / Test
 - ***Cross-Validation***

Cross-validation

- **Divide dataset** into K folds, each with $\frac{N}{K}$ samples
 - Each fold should reflect dataset statistics
 - E.g., stratified sampling
- **Do K iterations** $i = 1, \dots, K$
- In i -th iteration
 - Train on all folds except i using $\frac{K-1}{K}N$ points and get $g^{(-i)}(\underline{x})$
 - Validate on i -th fold using $\frac{N}{K}$ points to compute

$$E_{val}^{(i)} = E_{val}[g^{(-i)}(\underline{x})]$$

- **Average** K error rates and compute bounds:

$$E_{val} = \frac{1}{K} \sum_i E_{val}^{(i)}$$

Train / test validation

Train/Test Split

Train	Train	Train	Train	Test	Test
-------	-------	-------	-------	------	------

5x cross-validation

Iteration 1

Test	Train	Train	Train	Train
------	-------	-------	-------	-------

Iteration 2

Train	Test	Train	Train	Train
-------	------	-------	-------	-------

Iteration 3

Train	Train	Test	Train	Train
-------	-------	------	-------	-------

Iteration 4

Train	Train	Train	Test	Train
-------	-------	-------	------	-------

Iteration 5

Train	Train	Train	Train	Test
-------	-------	-------	-------	------

Cross-Validation: Pros and Cons

- **Pros**

- Efficient data usage
 - All data used for learning and validation
- Better estimate of E_{val} than separate training/validation sets
- Folds can be stratified

- **Cons**

- Computationally intensive: K learning phases
- Dependency on fold selection
- Errors E_{val} are not independent
 - Coupling through common training samples
 - Experimentally not completely correlated

Repeated Cross-Validation

- **Repeated Cross-Validation**

- Cross-validation results depend on fold selection
- To remove this dependency, repeat cross-validation multiple times (e.g., 10) and average results
- Note that “*10x 10-fold cross-validation*” is different than “*1x 100-fold cross-validation*”

Leave-One-Out Cross-Validation

- **Leave-one-out** (LOO) cross-validation
 - There are N training sessions
 - Each session trains on $N - 1$ points and validates on 1 point
 - Equivalent to “ N -fold cross-validation” where N is the number of examples in the dataset
- **Train**
 - For i -th fold, $N - 1$ samples for training $\implies g^{(-i)} \approx g$
- **Validate**
 - Estimate the validation error on a single point (bad):

$$E_{val}[g^{(-i)}] = e(g^{(-i)}(\underline{x}_i), y_i) \not\approx E_{out}[g^{(-i)}]$$

- Average E_{val} over the points as estimate of E_{out} (good):

$$E_{val} = \frac{1}{N} \sum_{i=1}^K E_{val}[g^{(-i)}]$$

Leave-One-Out Cross-Validation: Pros and Cons

- **Pros**

- Max data used for training
- Deterministic procedure
 - No dependency on fold selection

- **Cons**

- High computational cost (as many learning phases as data points)
- Cannot be stratified
- Higher correlation between cross-validation estimates

Bootstrap

- **Algorithm**
 - Pick N samples with replacement from a data set with N instances to build training set
 - Pick the elements never chosen to build the test set ("out-of-bag" samples)
 - Training set contains 63.2% of all the samples, 36.8% in the test set
- **Pros**
 - Works for small data sets since it "expands" the data
- **Cons**
 - Not flexible
 - Smaller percentage of instances are used for training set than 10-fold cross validation

Bootstrap: Problem

- **Assumption**
 - Given $X \sim F$
 - Draw n IID samples X_i from F
- **Goal:**
 - Consider a sampling statistic $T = g(X_1, \dots, X_n)$
 - E.g., mean, median, OLS coefficients, ...
 - You want to:
 - Estimate distribution of T , e.g., $F_T(x)$, $f_T(x)$
 - Estimate a statistic of T , e.g., mean, std err
 - Construct confidence intervals, e.g., $\mu \pm \epsilon$
 - Calculate standard errors, e.g., $\sigma(\hat{T}_n)$
 - Hypothesis testing

Bootstrap Procedure: Algorithm

- Use observed data X_1, \dots, X_n to construct estimated population distribution \hat{F}_n
- Repeat B times:
 - Draw n samples with replacement from estimated population distribution \hat{F}_n
 - Sampling with replacement from X_i equals sampling from \hat{F}_n
 - Compute sample statistics from n samples: $T^{(i)} = g(X_1^{(i)}, \dots, X_n^{(i)})$
- Use B samples of $T^{(i)}$ to estimate empirical distribution \hat{T}
- Compute statistics (e.g., confidence interval, standard error) of T from empirical distribution of \hat{T}

Bootstrap: Pros

- **Tremendously useful tool**
 - Fewer assumptions
 - No need for simplifying assumptions for closed formulas
 - Data doesn't need to be Gaussian
 - Generality
 - Applies to any sample statistics, even non-linear ones (e.g., median)
- **Math → simulation**
 - Bootstrap is a non-parametric method
 - Does not rely on large sample sizes, e.g., Central Limit Theorem (CLT) / Law Large Numbers (LLN)
 - Bootstrap frees data scientists from complex math, approximations, and asymptotics

Bootstrap: Example of Die Rolls

- **Problem**

- Compute distribution of sum of rolling a die 50 times

$$Y = \sum_{i=1}^{50} X_i = g(X_1, \dots, X_{50})$$

- Sample statistics similar to sample mean, but can be anything

- **Solution**

1. **By math**

- If you know PMF of the die, compute distribution using mathematics
- Theorem of lazy statistician for mean, variance
- Compute PDF of Y by convolving PDFs

2. **By sampling** (real or simulated)

- Roll die 50 times
- Compute sample statistic
- Repeat procedure
- Plot approximate distribution of Y

3. If only 50 samples of die are known → **bootstrap**

Bootstrap of the Median: Pseudo-Code

```
def bootstrap_median(x, n_boot):
    # Compute n_boot sample statistics.
    median_boot = [0.0] * n_boot
    for i in range(n_boot):
        # Sample with replacement.
        x_star = sample with replacements from x
        # Compute median for bootstrapped samples.
        median_boot[i] = median(x_star)
    # Compute mean and std err from approximation of sample statistics.
    m_median = numpy.mean(median_boot)
    se_median = numpy.std(median_boot)
    return m_median, se_median
```

Bootstrap for variance of sample statistics: explanation

- Under the hypotheses of bootstrap:
 - $X \sim F$
 - n samples X_i IID from F
 - Statistic $T = g(X_1, \dots, X_n)$
 - Compute $\mathbb{V}_F[T]$ (variance of sample statistics), where F is unknown

- **First approximation**

- Only samples from F available
- Approximate F with \hat{F} , as empirical CDF \hat{F} converges to F

$$\mathbb{V}_F[T] \approx \mathbb{V}_{\hat{F}}[T]$$

- Approximation
 - Not small
 - Depends on sample size and shape of F

- **Second approximation**

- Exact \hat{F} may lack closed formula for $\mathbb{V}_{\hat{F}}[T]$
- Use LLN and simulation to approximate variance:

$$\mathbb{V}_{\hat{F}}[T] \approx v_{boot} = \frac{1}{B} \sum_i (T_i - \bar{T})^2$$



SCIENCE
ACADEMY

Approximation size reduced by increasing B