MSML610: Advanced Machine Learning

Machine Learning Theories

Instructor: GP Saggese, PhD - gsaggese@umd.edu

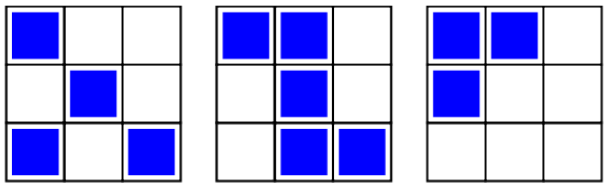
References:

Is machine learning possible?

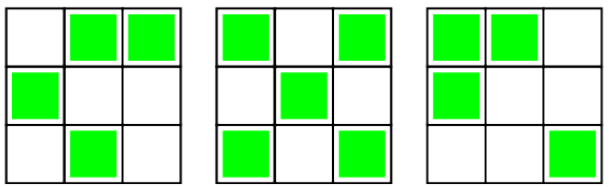
- **Is machine learning possible?**
- Growth function
- The VC dimension
- Overfitting
- Bias Variance Analysis
- Learning curves
- Learn-validation approach

A simple visual ML experiment

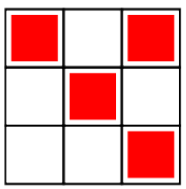
- Consider the supervised classification problem
- **Input**
  - A 9 bit vector represented as a 3x3 black and white array
- **Training set**
  - The blue row $\underline{\boldsymbol{x}}_1, \underline{\boldsymbol{x}}_2, \underline{\boldsymbol{x}}_3$ for $f(\underline{\boldsymbol{x}}) = -1$
  - The green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



$f = -1$



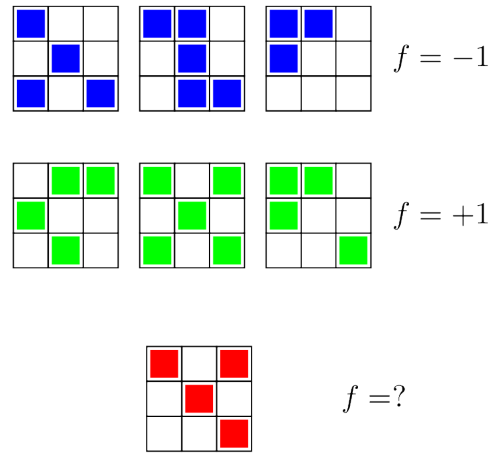
$f = +1$



$f = ?$

A simple visual ML experiment (2/2)

- **Option 1**
  - $f(\underline{\boldsymbol{x}}) = +1$ when $\underline{\boldsymbol{x}}$ has an axis of symmetry
  - $f(\underline{\boldsymbol{x}}) = -1$ when $\underline{\boldsymbol{x}}$ is not symmetric
  - The test set is symmetrical $\implies f(\underline{\boldsymbol{x}}_0) = +1$
- **Option 2**
  - $f(\underline{\boldsymbol{x}}) = +1$ when the top left square $\underline{\boldsymbol{x}}$ is empty
  - $f(\underline{\boldsymbol{x}}) = -1$ when the top left square $\underline{\boldsymbol{x}}$ is full
  - The test set has top left square full $\implies f(\underline{\boldsymbol{x}}_0) = -1$
- 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?**

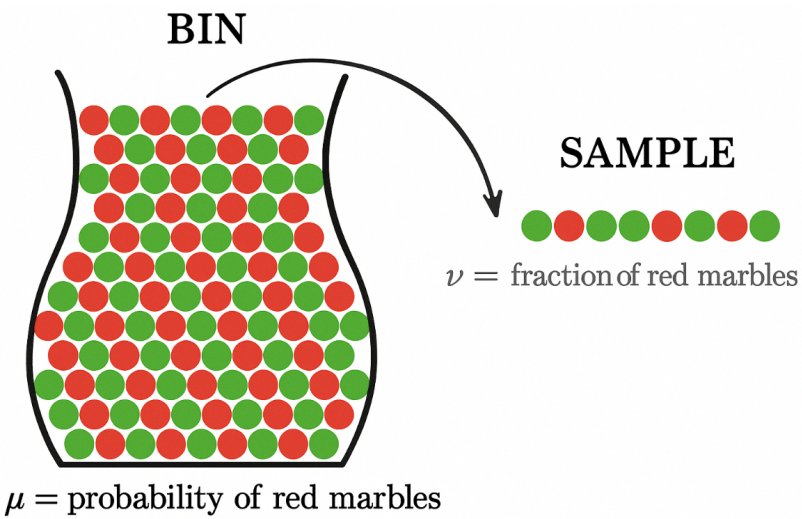


Is machine learning possible?

- The function can assume **any value outside the data** we have
  - E.g., if you have temperature data only for summer, the function could assume a very different value for winter
- How to learn an unknown function?
  - Estimating the function at unseen points seems impossible
  - Requires assumptions or models about the function's behavior
- The difference is between possible and probable
  - What is **possible**
    - We don't know anything about the unknown function
    - E.g., the function could be linear, quadratic, or a sine wave outside our known data
  - What is **probable**
    - We know something about the unknown function
    - This could come from domain knowledge or historical data patterns
    - E.g., if historical weather data usually forms a sinusoidal pattern, it is probable that unknown points 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 $\nu = \frac{1}{N}\sum X_i$
- 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
  - Works for finite or infinite distributions
  - Holds because we picked marbles at random and only if we sample is ν and μ 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 $\mu \in [\mu - \varepsilon, \mu + \varepsilon]$, we want small ε with a large probability
  - The Hoeffding inequality is a statement about ν and not μ (like for a confidence interval), although we use it to state something about ν

Supervised learning: bin analogy (2/2)

- Let's connect the bin analogy, Hoeffding inequality, and feasibility of machine learning
  - We know $f(\underline{\boldsymbol{x}})$ at points $\underline{\boldsymbol{x}} \in \mathcal{X}$
  - We choose hypothesis $h : \mathcal{X} \to \mathcal{Y} = \{0, 1\}$
  - Each point $\underline{\boldsymbol{x}} \in \mathcal{X}$ is a marble
  - We color red if the hypothesis is correct $h(\underline{\boldsymbol{x}}) = f(\underline{\boldsymbol{x}})$, green otherwise
  - The in-sample error $E_{in}(h)$ corresponds to ν
  - There are finite marbles of unknown color, corresponding to $E_{out}(h) = \mu$
  - $\underline{\boldsymbol{x}}_1, ..., \underline{\boldsymbol{x}}_n$ 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 set-up: bin analogy

- We know 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 a **validation setup**, after we have already learned a model
- In a **learning setup** we have h to choose from M hypotheses
  - We need a bound on the out-of-sample performance of the chosen hypothesis $h \in \mathcal{H}$, regardless of which hypothesis we choose
  - We need a Hoeffding counterpart for the case of choosing from multiple hypotheses
  - Pick g from $\mathcal{H} = \{h_1, ..., h_M\}$, so

$$\begin{aligned}
&\Pr(|E_{in}(g) - E_{out}(g)| > \varepsilon) \\
&\quad \leq \Pr(\bigcup_{i=1}^{M}(|E_{in}(h_i) - E_{out}(h_i)| > \varepsilon)) \\
&\quad \leq \sum_{i=1}^{M} \Pr(|E_{in}(h_i) - E_{out}(h_i)| > \varepsilon) \quad \text{(by the union bound)} \\
&\quad \leq 2M \exp(-2\varepsilon^2 N) \qquad\qquad\qquad\qquad \text{(by Hoeffding)}
\end{aligned}$$

  - **Intuition**: if we have 1000 bins with mostly red marbles, it is possible that our sample ends up being 100% green, so the bound is weak

Validation vs learning set-up: Coin Analogy

- In a **validation set-up**, we 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 we get 10 heads (i.e., the coin looks biased $\nu = 0$)?

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

- In other terms the probability that the out-of-sample performance ($\nu = 0.0$) is very different from the in-sample perf ($\mu = 0.5$) is very low

Validation vs learning set-up: Coin Analogy

- In a **learning set-up**, we have many coins and we need to choose one and determine if it's fair
- If we 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

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

- It is probable, more than 50%

Hoeffding inequality: validation vs learning

- In **validation / testing**
  - We can use Hoeffding to assess how well our g (the chosen hypothesis) approximates f (unknown hypothesis):

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

    where:

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

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

  - Since the hypothesis g is final and fixed, Hoeffding inequality guarantees that we can learn since it gives a bound for E_{out} to track E_{in}
- In **learning / training**
  - We need to account that our hypothesis is the best of M hypotheses, so the union bound gives:

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

  - The bound for E_{out} from Hoeffding is weak
- Is the bound weak because it needs to be or because the Hoeffding inequality is not good enough?

Why union bound for Hoeffding is loose: Intuition

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

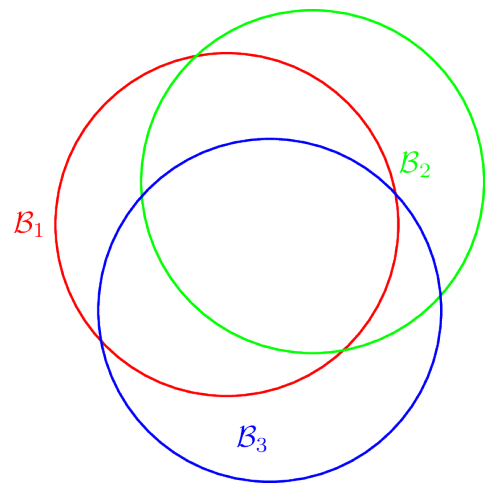
$$\Pr(|E_{in} - E_{out}| > \varepsilon) \leq 2M \exp(-2\varepsilon^2 N)$$

  is artificially too loose

- Is it possible to replace it with a stricter bound?
- M was coming from the bad event:

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

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



Why union bound for Hoeffding is loose: Intuition

- Consider two linearly separable classes on a plane, in terms of the ground truth and a training set
- Consider two 2D perceptrons with similar weights: g_1, g_2
- E_{out} is the area where each hypothesis g_i and the ground truth disagree
- ΔE_{out} is the differential area between the two E_{out}
- E_{in} corresponds to the points in the training set falling in the area corresponding to E_{out}
- ΔE_{in} is the number of points falling in ΔE_{out}, i.e., changing classification going from one hypothesis to the other
- Thus the two "bad events" $\mathcal{B}_1$ and $\mathcal{B}_2$ are related to ΔE_{in} and ΔE_{out}

Training vs testing: college course analogy

- Before the final exam, students receive practice problems and solutions
  - These problems won't appear on the exam
  - Studying the problems helps students improve performance
  - They serve as a "training set" in our learning setup
- Why not give out the exam problems if the goal is to improve exam performance?
  - Doing well in the exam isn't the goal
  - The goal is for students to learn the course material
- The final exam isn't strictly necessary
  - It gauges how well you've learned the material
  - (and motivates you to study)
  - Knowing the exam problems in advance wouldn't gauge learning effectively
- This is similar to training and testing in machine learning

Growth function

- Is machine learning possible?
- **Growth function**
- The VC dimension
- Overfitting
- Bias Variance Analysis
- Learning curves
- Learn-validation approach

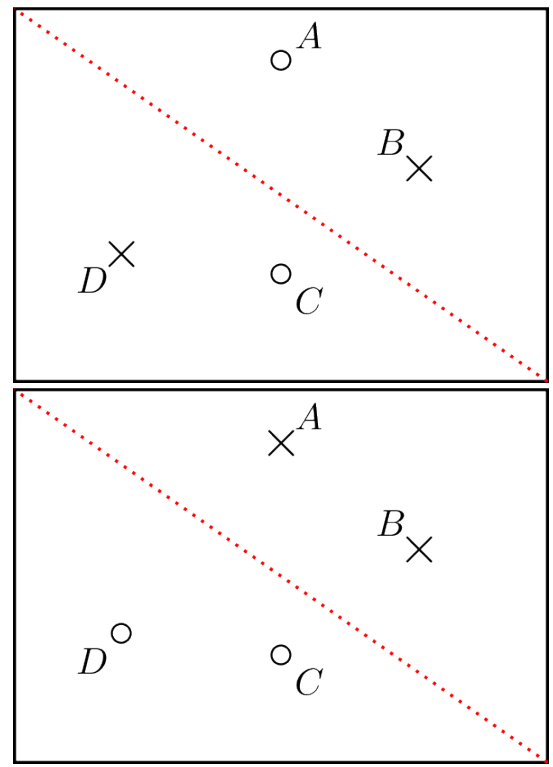
Dichotomy: definition

- Consider the problem of classifying N (fixed) points $\underline{\boldsymbol{x}}_1, ..., \underline{\boldsymbol{x}}_N$ with an hypothesis set $\mathcal{H}$
- Consider an assignment D of the points to certain class $\underline{\boldsymbol{d}}_1, ..., \underline{\boldsymbol{d}}_N$
- 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
- $\mathcal{H} = \{$ bidimensional perceptrons (binary classifier) $\}$
- Moving the separating hyperplane, one gets different classifications for the 4 points (i.e., dichotomies)

	d1	d2	d3
A	0	0	0
B	0	0	
C	0	0	
D	0	1	

- Certain classifications are not possible (e.g., XOR assignment)



Training set, dichotomies, hypotheses

- An hypothesis classifies each point of $\mathcal{X}$: $\mathcal{X} \to \{-1, +1\}$
- A dichotomy classifies each point of a set: $\{\underline{\boldsymbol{x}}_1, ..., \underline{\boldsymbol{x}}_N\} \to \{-1, +1\}$
  - Dichotomies are "mini-hypotheses", i.e., hypotheses restricted to given points
  - A dichotomy depends on $\mathcal{H}$ and on where the points are placed
  - Certain dichotomies can exist for a certain position of the N points, but not for all the sets of N points
- The number of different dichotomies is indicated by $|\mathcal{H}(\underline{\boldsymbol{x}}_1, ..., \underline{\boldsymbol{x}}_N)|$
  - This notation is like applying each hypothesis to the points and see how points are classified
- From the training set point of view, what matters are dichotomies and not hypotheses
  - The number of dichotomies is always finite, since $|\mathcal{H}(\underline{\boldsymbol{x}}_1, ..., \underline{\boldsymbol{x}}_N)| \leq N^K$
  - The number of hypotheses is usually infinite, i.e., $|\mathcal{H}| = \infty$
  - 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 $\mathcal{H}$:

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

- The dichotomies depend on point distribution
  - 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 $\underline{\boldsymbol{x}}_1, ..., \underline{\boldsymbol{x}}_N$
  - Consider all possible hypotheses $h \in \mathcal{H}$
  - Compute the corresponding dichotomy for h on $\underline{\boldsymbol{x}}_1, ..., \underline{\boldsymbol{x}}_N$
  - Count the number of different dichotomies

What can vary in a dichotomy

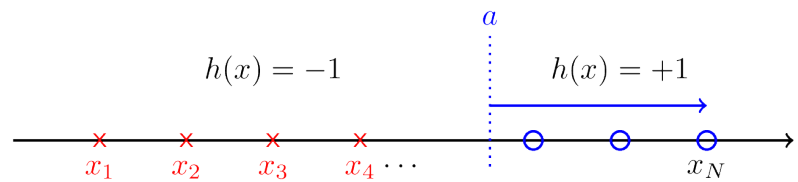
- Given:
  - An hypothesis set $\mathcal{H}$
  - N (fixed) points $\underline{\boldsymbol{x}}_1, ..., \underline{\boldsymbol{x}}_N$
  - An assignment D of the points to certain class $\underline{\boldsymbol{d}}_1, ..., \underline{\boldsymbol{d}}_N$
- D is a dichotomy for hypothesis set $\mathcal{H}$ $\iff$ there exists $h \in \mathcal{H}$ that gets the desired classification D
- There are various quantities in play 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
    - It is determined by each hypothesis in $\mathcal{H}$
  - Where the points are positioned
    - It is a free parameter, removed by the 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 $\mathcal{H}$

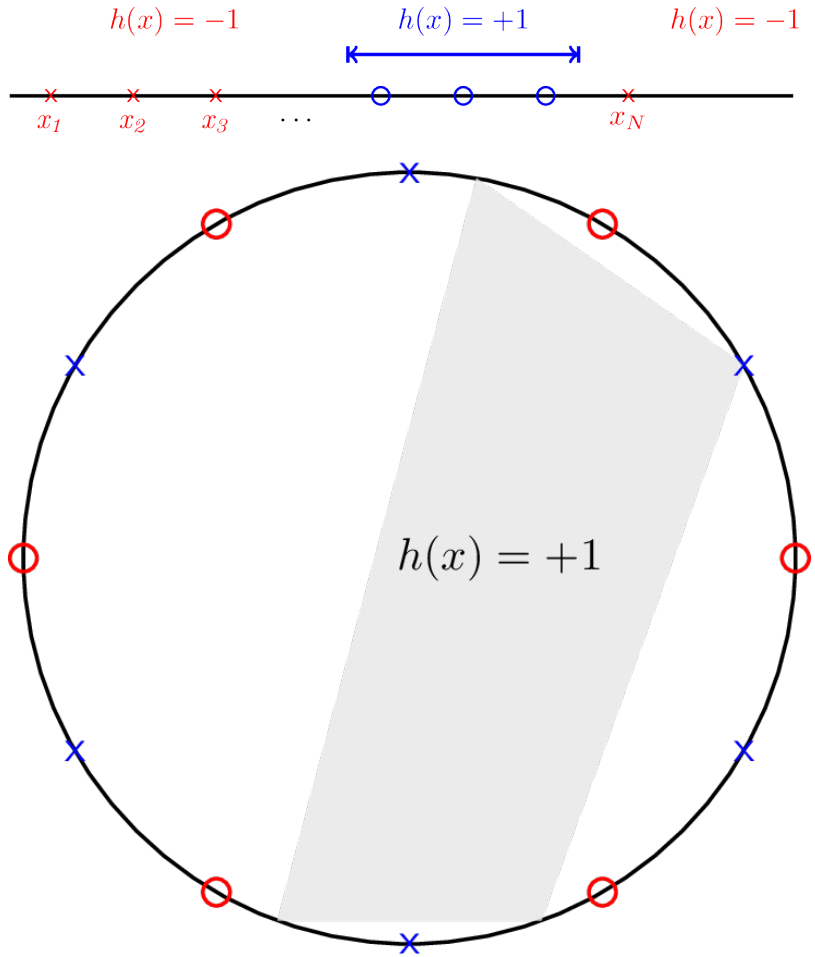
Growth function: Examples

- 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 $\mathbb{R}$ $x \in [a, b]$
  - $m_{\mathcal{H}}(N) = \binom{N+1}{2} + 1 \sim \mathrm{N}^2$
  - 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 that we can classify in any way 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:

1. $m_{\mathcal{H}}(N)$ is polynomial in N
2. In Hoeffding's inequality for learning, replace M with $m_{\mathcal{H}}(N)$:
   - Instead of:

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

     we 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
     - We can have a generalization bound (i.e., we can learn)

- A hypothesis set can be characterized for learning by the existence and value of a break point

$m_{\mathcal{H}}(N)$ can replace M in Hoeffding inequality

- How does $m_{\mathcal{H}}(N)$ relate to overlaps?
  - Given an hypothesis g, the "bad event" is a function of which data set D is used for training
  - Consider the space of data sets as an area of the plane: for some data sets $|E_{in} - E_{out}| > \varepsilon$, and we color the area representing the data set as bad
  - Hoeffding tells us that the area representing the bad event for hypothesis g is small
  - The union bound tells us that the areas representing the bad events for the various hypothesis are not overlapping (even if small) and thus the space is quickly filled, since there are many hypothesis (often infinity)
  - The VC bound tells us that there is a lot of overlap between the bad events.
  - The intuition is that if one point is colored by an hypothesis as bad, we know that many others hypothesis, say 100, will color the same paint as bad events, so that the area is 100 smaller than what would have been without overlap
  - The growth function is a measure of how many hypothesis correspond to the same dichotomy
- What to do about E_{out}?
  - The problem is that the bad event not only is function of E_{in} (which is function of the data set) but also of E_{out} which is function of the entire space

The VC dimension

- Is machine learning 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
- 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 (as before)
  - 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
  - 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 $d_{VC} = 3$
- 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
- The VC dimension considers:
  - How many points N perceptrons can shatter, not the number of parameters
  - The model as a black box to estimate effective parameters
- 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 $\Pr(|E_{in} - E_{out}| > \varepsilon) \leq \delta$?
- $N^d e^{-N}$ abstracts the term $\delta = 4 m_{\mathcal{H}}(2N) e^{\frac{1}{8}\varepsilon^2 N}$
  - If we plot $N^d e^{-N}$ as a function of N, the power wins for small N, then the exponential dominates, bringing the function to 0
  - Varying d (the VC dimension), the function peaks for larger N and then goes in the region of interest $< 1, 0.1, 0.01, ...$
- We can plot the intersection N of $N^d e^{-N}$ with a certain probability (e.g., 10^{-3}) as a function of d
  - The number of examples N needed to get to a certain level of performance is proportional to d
  - As a rule of thumb: $N \geq 10 d_{VC}$ to get reasonable generalization

VC Generalization Bounds

- One can use the VC inequality in several ways, relating ε, δ, and N, e.g.,
  - "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?"
- We 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 we 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:

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

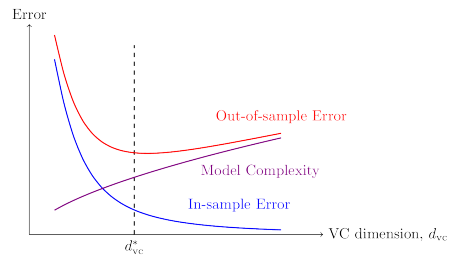
- $d_{VC} \leq \tilde{d} + 1$ and smaller $\tilde{d}$ improves generalization bounds
  - We can use $\underline{\boldsymbol{z}} = (1, x_1, x_2, x_1 x_2, x_1^2, x_2^2)$
  - Why not $\underline{\boldsymbol{z}} = (1, x_1^2, x_2^2)$?
  - Why not $\underline{\boldsymbol{z}} = (1, x_1^2 + x_2^2)$?
  - Why not $\underline{\boldsymbol{z}} = (x_1^2 + x_2^2 - 0.6)$?
- 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 (i.e., data snooping)
  - From VC analysis, complexity is that of the initial hypothesis set

Overfitting

- Is machine learning 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
- In practice, we surpass the point where E_{out} is minimal (optimal fit)
  - Model complexity is too high for the data/noise
  - Noise in the training set is mistaken for signal
- Fitting the noise instead than signal is worse than useless, it is harmful
  - The model infers a pattern in-sample that, when extrapolated out-of-sample, deviates from the target function $\Longrightarrow$ poor generalization



Optimal fit

- "Optimal fit" is the opposite of overfitting
  - Train a model with the proper complexity for the data
- The optimal fit:
  - Implies that E_{out} is minimal
  - Does not imply that generalization error $E_{out} - E_{in}$ is minimal (e.g., no training at all implies generalization error equal to 0)
- The generalization error is the additional error $E_{out} - E_{in}$ we see when we go from in-sample to out-of-sample

Overfitting: diamond price example

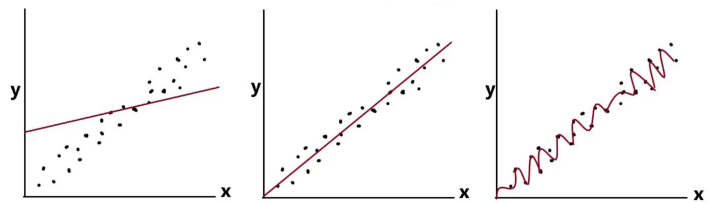
- Predict diamond price as a function of carat size (regression problem)
- True relationship:

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

  where:

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

- **Fit with:**
  - Line $\rightarrow$ underfit
    - High bias (large error)
    - Low variance (stable model)
  - Polynomial of degree 2 $\rightarrow$ right fit
  - Polynomial of degree 10 $\rightarrow$ overfit (wiggly curve)
    - Low bias
    - High variance (many degrees of freedom)



Overfitting: 2-features classification example

- Assume:
  - We want to separate 2 classes using 2 features x_1, x_2
  - The class boundary of sample points has a parabola shape
- We can use logistic regression and a decision boundary equal to:
  - A line $\text{logit}(w_0 + w_1 x + w_2 y) \rightarrow$ underfit
    - High bias, low variance
  - A parabola $\text{logit}(w_0 + w_1 x + w_2 x^2 + w_3 xy + w_4 y^2) \rightarrow$ right fit
  - A wiggly decision boundary $\text{logit}(w_0 + \text{high powers of } x_1, x_2) \rightarrow$ overfit
    - Low bias, high variance

Margin in classification

- Classification margin is the difference between the chosen class and the next predicted class
- Even if the error on training data gets to 0, one can improve out-of-sample performance by increasing the margin
    - More robust to noise

Bias Variance Analysis

- Is machine learning possible?
- Growth function
- The VC dimension
- Overfitting
- **Bias Variance Analysis**
- Learning curves
- Learn-validation approach

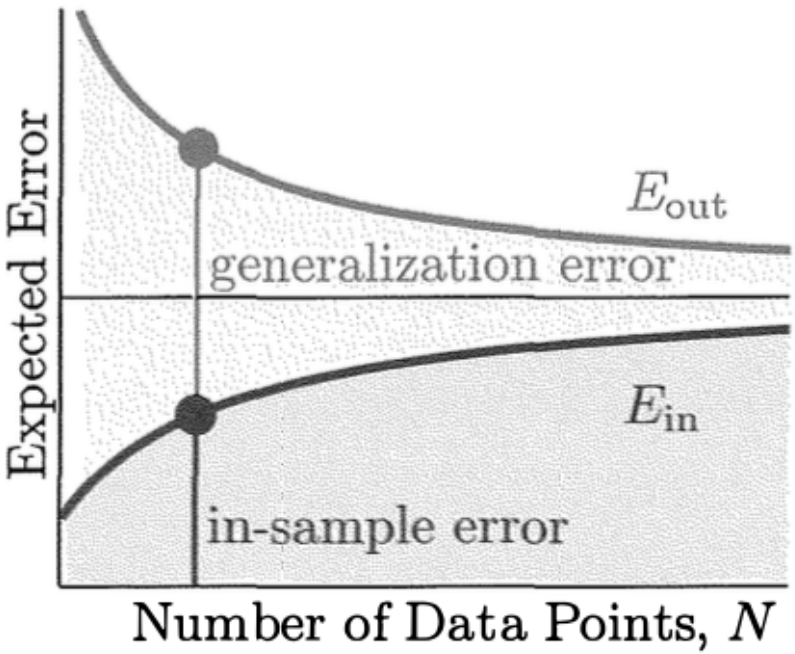
VC analysis vs bias-variance analysis

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

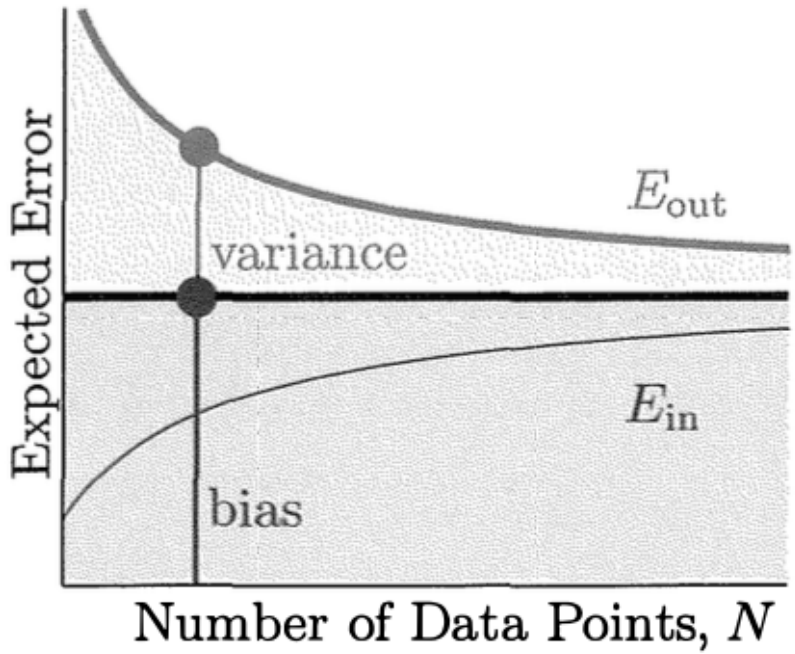
$$E_{out} \leq E_{in} + \Omega(\mathcal{H})$$

  - Bias-variance analysis:

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



VC Analysis



Bias-Variance Analysis

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

- Consider machine learning problem
  - Regression set-up: target is a real-valued function
  - Hypothesis set $\mathcal{H} = \{h_1(\underline{\boldsymbol{x}}), h_2(\underline{\boldsymbol{x}}), ...h_n(\underline{\boldsymbol{x}})\}$
  - Training data D with N examples
  - Error is squared error $E_out = \mathbb{E}[(g(\underline{\boldsymbol{x}}) - f(\underline{\boldsymbol{x}}))^2]$
  - Choose the best function g from $\mathcal{H}$ that approximates f
- 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

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

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

- We are interested in:
  - The hypothesis set $\mathcal{H}$ rather than the specific h; and
  - In a training set D of N examples, rather than the specific D
- Therefore we Remove the dependency from D by averaging over all the possible training sets D with N examples:

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

Decomposing error in bias-variance

- 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:

$$\overline{g}(\underline{x}) \stackrel{def}{=} \mathbb{E}_D[g^{(D)}(\underline{x})]$$

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

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

Decomposing error in bias-variance

- The cross term:

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

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

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

- Finally:

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

Interpretation of average hypothesis

- The average hypothesis over all training sets

$$\overline{g}(\underline{\boldsymbol{x}}) \stackrel{def}{=} \mathbb{E}_D[g^{(D)}(\underline{\boldsymbol{x}})]$$

  can be interpreted as the "best" hypothesis from $\mathcal{H}$ training on N samples
  - Note: $\overline{g}$ 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 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**

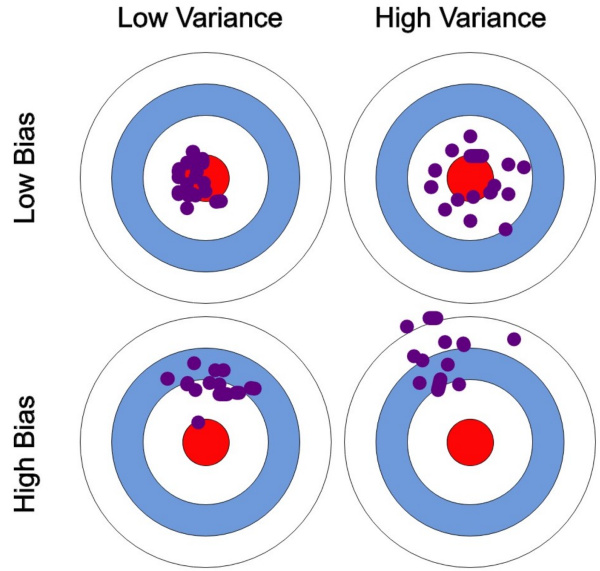
$$\text{bias}^2 = \mathbb{E}_{\underline{x}}[(\overline{g}(\underline{x}) - f(\underline{x}))^2]$$

  - 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**

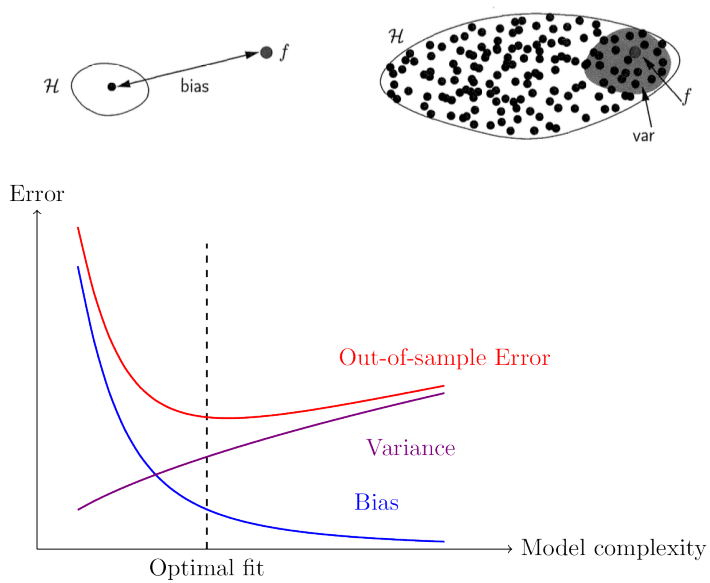
$$\text{variance} = \mathbb{E}_{\underline{x}}\mathbb{E}_D[(g^{(D)}(\underline{x}) - \overline{g}(\underline{x}))^2]$$

  - 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 $\overline{g}$
    - But we have only one data set D at a time, incurring a cost



Variance and bias term varying cardinality of $\mathcal{H}$

- If we have a hypothesis set with a single function: $\mathcal{H} = \{h \neq f\}$
  - Bias can be large
    - Since h might be far from f
  - Variance $= 0$
    - There is no cost of choosing an hypothesis
- If we have $\mathcal{H} = \{\text{many hypotheses } h\}$
  - Bias can be 0
    - E.g., if $f \in \mathcal{H}$
  - Variance can be large
    - Since depending on which data set D we get, we end up far from f
    - We can assume that the larger $\mathcal{H}$ is, the farther apart g from f will be



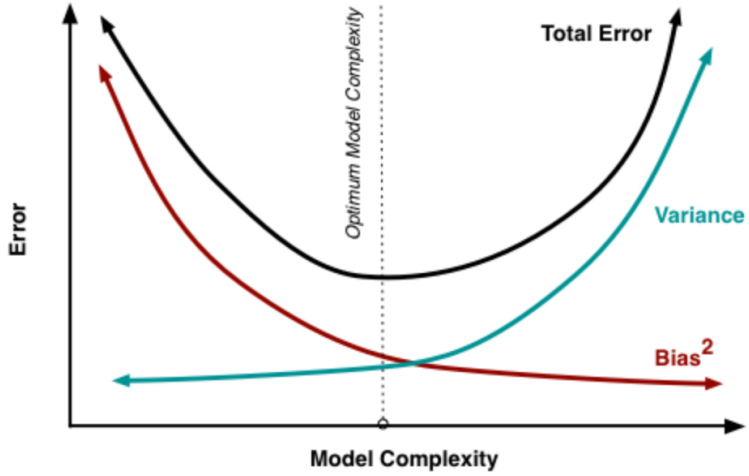
Bias-variance trade-off: numerical trade-off

- Assume:
  - Target function $f(x) = \sin(\pi x), x \in [-1, 1]$
  - Noiseless target
  - Value of $f(\underline{\boldsymbol{x}})$ for $N = 2$ points
- Hypotheses sets:
  - $\mathcal{H}_0 : h(x) = b$ constant
  - $\mathcal{H}_1 : h(x) = ax + b$ linear
- Which one 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
- Approximation:
  - $E_{out}(g_0) = 0.5$
  - $E_{out}(g_1) = 0.2$; g_1 has more degrees of freedom
- Learning:
  - Pick 2 points as training set, learn g, compute $\mathbb{E}_D[E_{out}(g)]$
  - Different D gives different g
  - Average over all data sets D gives $\overline{g}$:

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

Bias-variance curves

- Bias-variance curve are plots of E_{out} increasing the complexity of the model
  - Can diagnose bias-variance problem
- Typical form 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
  - Is shaped like an hyperbole
- 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



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 curves and regularization

- We can use a complex model together with regularization to learn at the same time:
  - The model coefficients $\underline{\boldsymbol{w}}$
  - The model "complexity" (e.g., VC dimension), which is related to the regularization parameter λ
- For each different values of $\lambda = \{10^{-1}, 1.0, 10\}$ we optimize:

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

  - $\underline{\boldsymbol{w}}(\lambda)$ is the optimal model as function of λ
- Then estimate E_{out} using $\underline{\boldsymbol{w}}(\lambda)$ and λ
  - Small λ means
    - Complex model (with respect to data)
    - Low bias
    - High variance
  - Large λ means
    - Simple model
    - High bias
    - Low variance
- There will be an intermediate value of λ that optimizes the trade-off between bias and variance

Bias-variance decomposition with a noisy target

- We can extend the bias-variance decomposition to the noisy target

$$y = \underline{\boldsymbol{w}}^T \underline{\boldsymbol{x}} + \varepsilon$$

- With similar hypothesis and a similar analysis we conclude that:

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

- **Interpretation**:
  - The error is the sum of 3 contributions
    1. Variance: from the set of hypotheses to the centroid of the hypothesis set
    2. Bias: from the centroid of the hypothesis set to the noiseless function
    3. 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 our hypothesis set cannot capture:

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

    where
    - $h^*()$ is the best approximation of $f(\underline{\boldsymbol{x}})$ in the hypothesis set $\mathcal{H}$
    - E.g., $\overline{g}(x)$
- 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

- In bias-variance analysis, the error for a noisy target is decomposed into:
  - Bias (deterministic noise)
  - Variance
  - Stochastic noise
- **Deterministic noise:**
  - Fixed for a particular $\underline{\boldsymbol{x}}$
  - Depends on $\mathcal{H}$
  - Independent of ε or D
- **Stochastic noise:**
  - Not fixed for $\underline{\boldsymbol{x}}$
  - Independent of D or $\mathcal{H}$
- In an actual machine learning problem, there's no difference between stochastic and deterministic noise, since $\mathcal{H}$ and D are fixed
  - E.g., from the training set alone, we cannot tell if the data is from a *noiseless complex* target or a *noisy simple* target

Deterministic vs stochastic noise example

- 2 targets:
  - Noisy low-order target (5-th order polynomial)
  - Noiseless high-order target (50-th order polynomial)
  - Generate $N = 15$ data points from them
- 2 models:
  - $\mathcal{H}_2$ low-order hypothesis (2nd order polynomial)
  - $\mathcal{H}_{10}$ high-order hypothesis (10-th order polynomial)
- When learning a model there is no difference between deterministic and stochastic noise
- In fact the learning algorithm only sees the samples in the training set and one cannot distinguish the two different sources
- For noisy low-order target: going from fitting the 2nd order to the 10-th order polynomial we see that $\downarrow E_{in}$ (we have more degrees of freedoms) and $\uparrow\uparrow E_{out}$ (since the 10-th polynomial fits the noise)
- For noiseless high-order target: exactly the same phenomenon!
- Knowing that the target is a 10-th order polynomial, one can think that

Amount of data and model complexity

- The lesson learned from bias-variance analysis is that one must match the *model complexity*:
  - To the *data resources*
  - To the *signal to noise ratio*
  - **Not** to the *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

Bias-variance curves for neural networks

- For neural networks one can plot E_{in} and E_{out} as function of the training epochs
- One starts with random weights
- Both E_{in} and E_{out}:
  - Are large
  - Start exactly from the same value
    - No generalization error
    - Since the model is random, it has no optimistic bias on the training set
- As learning proceeds:
  - E_{in} and E_{out} decrease
  - The generalization error $E_{out} - E_{in}$ increases
- It is like the VC dimension is increasing from 0 towards all the available degrees of freedom while exploring the weight space

Overfitting as a function of data resources, model complexity, noise

- We can measure overfitting as $\frac{E_{out} - E_{in}}{E_{out}}$
  - $\uparrow$ data resources $(N) \implies \downarrow$ overfitting
  - $\uparrow$ model complexity $(d_{VC}) \implies \uparrow$ overfitting
  - $\uparrow$ stochastic noise (σ^2) / deterministic noise (target complexity) $\implies \uparrow$ overfitting
- There is an error to which both E_{in} and E_{out} converge for $N \to \infty$
  - Irreducible error
  - This error depends on stochastic and deterministic noise
  - There is no variance (since $N = \infty$)

Learning curves

- Is machine learning 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 training / test 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

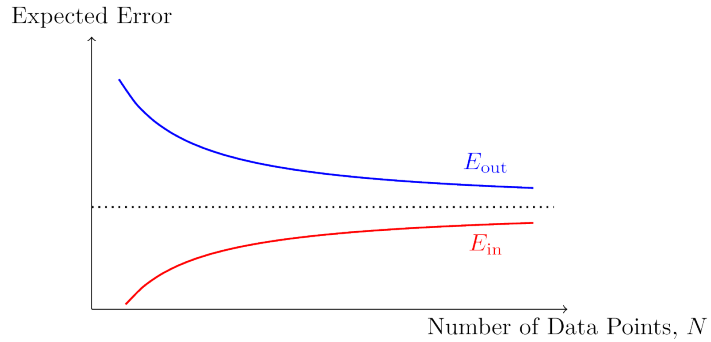
$$E_{in}, E_{out} = f(N|d_{VC})$$

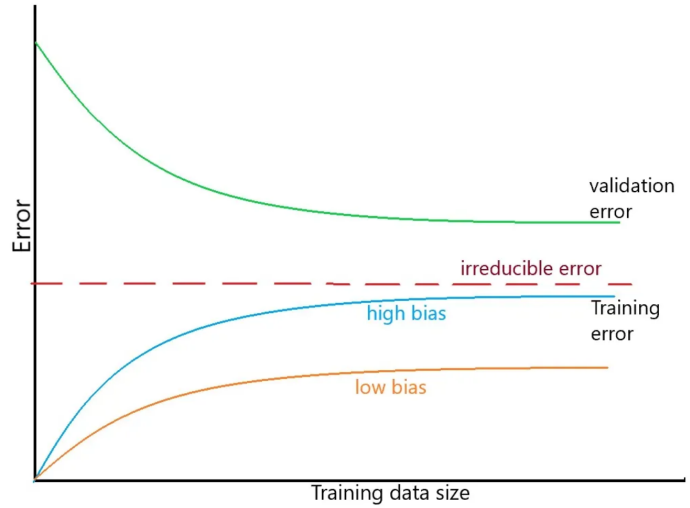
  - Fix the model
  - Vary the size N of training set

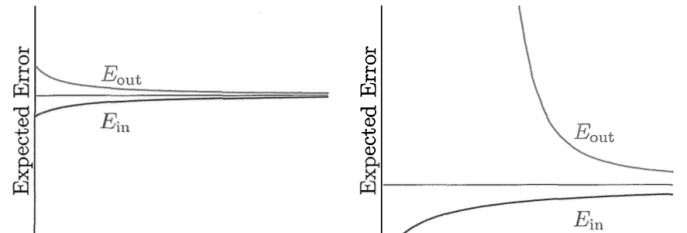
Typical form of learning curves

... tmp.prettier_on_str._4r6shu5.md 25ms - Learning curves plot E_{in} and E_{out} error as a function of data amount

- **Small** N
  - With small data N, E_{in} might be small (even 0) depending on model capacity (VC dimension)
  - The model is likely overfitted, memorizing examples, generalizing poorly, and E_{out} is large
- **Increasing** N
  - E_{in} increases as the model cannot fit all data
  - E_{out} decreases as the model fits better and generalizes better ($E_{out} - E_{in}$ decreases)
  - Asymptotically, E_{in} reaches a minimum (not 0 if noise), while E_{out} starts increasing, entering overfitting regime
  - $E_{out} \geq E_{in}$ for any N







High-bias vs high-variance regime

- From the learning curve we can see two regimes:
  - High-variance regime for small N
    - E_{in} is small
    - $E_{out} > E_{in}$
    - More data helps; gap between E_{in} and E_{out} decreases
    - Small data set D; high dependency on D
  - **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

Learn-validation approach

- Is machine learning possible?
- Growth function
- The VC dimension
- Overfitting
- Bias Variance Analysis
- Learning curves
- **Learn-validation approach**
  - Train / test
  - Cross-validation

Train / test

- Is machine learning 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 an out-of-sample point $(\underline{\boldsymbol{x}}', y)$
- The error of the model h is:

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

where the error can be:

  - Squared error $(h(\underline{\boldsymbol{x}}) - y)^2$
  - Binary error $I[h(\underline{\boldsymbol{x}}) - y]$
  - ...

- The error on an out-of-sample point is an unbiased estimate of E_{out}, since

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

by definition

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

Estimating out-of-sample error with K points

- To improve the estimate, use a validation set, i.e., K points instead of one point $(\underline{\boldsymbol{x}}_1, y_1), ..., (\underline{\boldsymbol{x}}_K, y_K)$ drawn IID
- Compute the error on the validation set as:

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

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

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

- The error is:

$$\begin{aligned}
\mathbb{V}[E_{val}(h)] &= \frac{1}{K^2} \sum_i \mathbb{V}[e(h(\underline{\boldsymbol{x}}_i), y_i)] + \text{covariances} \\
&= \frac{1}{K^2} \sum_i \mathbb{V}[e(h(\underline{\boldsymbol{x}}_i), y_i)] \qquad \text{(covariances are 0 because } \underline{\boldsymbol{x}} \\
&= \frac{K\sigma^2}{K^2} = \frac{\sigma^2}{K}
\end{aligned}$$

Trade-off between training and validation set size

- **Problem**: to better estimate E_{val}, we need points from the training set:

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

- We know:

$$\uparrow K \implies \mathbb{V}[E_{val}] \implies |E_{val} - E_{out}| \downarrow \text{ (most reliable estimate)}$$

but

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

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

- **Solution**:
  - Rule of thumb: 70-30 or 80-20 split between train and validation
  - 20%-30% of data for validation

E_{out} 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: e.g., validation / test sets

Reusing validation / test set for training

- Never use D_{val} for training, at least during research
  - If D_{val} affects learning (e.g., model selection), this data set is biased and optimistic and cannot assess the model
- **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

Cross-validation

- Is machine learning possible?
- Growth function
- The VC dimension
- Overfitting
- Bias Variance Analysis
- Learning curves
- Learn-validation approach
  - Train / test
  - **Cross-validation**

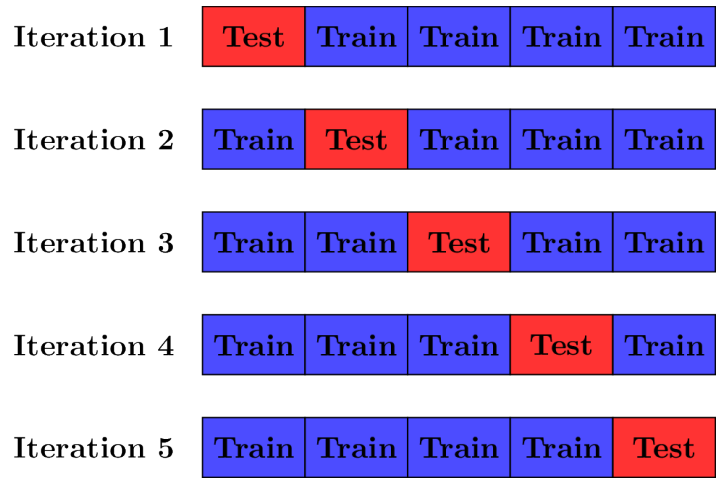
Cross-validation

- Divide the dataset into K folds, each with $\frac{N}{K}$ samples
- Each fold should reflect the full dataset's statistics
  - E.g., stratified sampling
- There are K iterations $i = 1, ..., K$:
  - In the i-th iteration, train on all folds except i (use $\frac{K-1}{K}N$ points) and get $g^{(-i)}(\underline{\boldsymbol{x}})$
  - Validate on the i-th fold (use $\frac{N}{K}$ points) to compute

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

- Average the K error rates and compute bounds:

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



5x cross-validation

Train / test validation

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

- Cross-validation results depend on fold selection
  - To remove this dependency, repeat cross-validation multiple times (e.g., 10) and average results
- Note: "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
  - Like "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:
  - Estimate the validation error on a single point (bad):

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

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

$$E_{val} = \frac{1}{N}\sum_i E_{val}[g_i^-]$$

Leave-one-out cross-validation: pros and cons

- **Pros**
  - Max data used for training
  - Deterministic procedure (no fold selection dependency)
- **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