NỀN TẢNG AI TẠO SINH

(IT5410 – Foundation of Generative AI)

MỘT SỐ VẤN ĐỀ CỦA LÝ THUYẾT HỌC MÁY VÀ HỌC SÂU

Thân Quang Khoát

Trường CNTT&TT, ĐHBKHN

2025

Nội dung

- Mở đầu
- **Một số vấn đề của Học sâu**
- Một số kiến trúc mạng nơron
- Mô hình sinh sâu
- Đánh giá chất lượng
- Học tăng cường

Learning theory

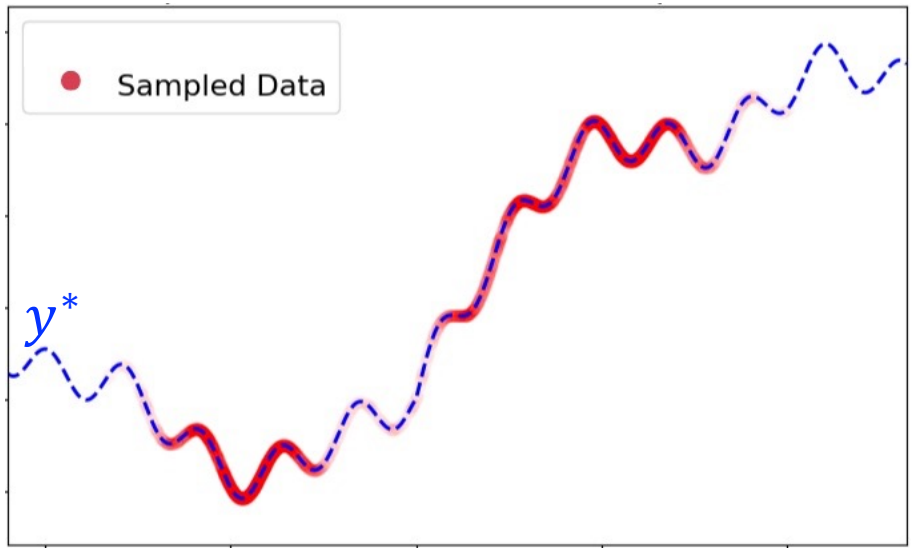
Basic concepts

The supervised learning problem

- There is an **unknown** (measurable) function
  (Có một hàm gán nhãn nào đó)

$$y^*: \boldsymbol{x} \mapsto y$$

  - It maps each input $\boldsymbol{x} \in \mathcal{X}$ to a label (output) $y \in \mathcal{Y}$
  - Sometimes known as the *labeling function*
  - Spaces: *input space* $\mathcal{X}$, *output space* $\mathcal{Y}$



- We can collect a dataset **D** = {($\mathbf{x}_1$, y_1), ($\mathbf{x}_2$, y_2), …, ($\mathbf{x}_m$, y_m)}
  - $y_i = y^*(\boldsymbol{x}_i)$ for any $i \in \{1, \ldots, m\}$
  - $\boldsymbol{x}_i$ is an observation about input $\boldsymbol{x}$ (một quan sát về $\boldsymbol{x}$)
- We need to learn y^* from **D**

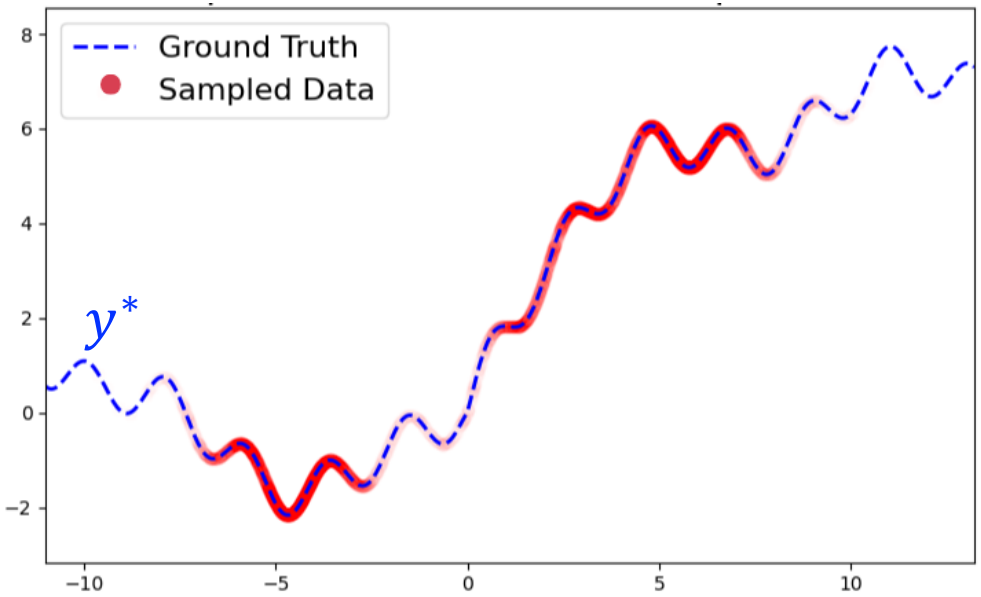
Chúng ta chưa quan tâm đến trường hợp y^* bị **nhiễu** hoặc **ngẫu nhiên** (ở đầu vào hoặc đầu ra).

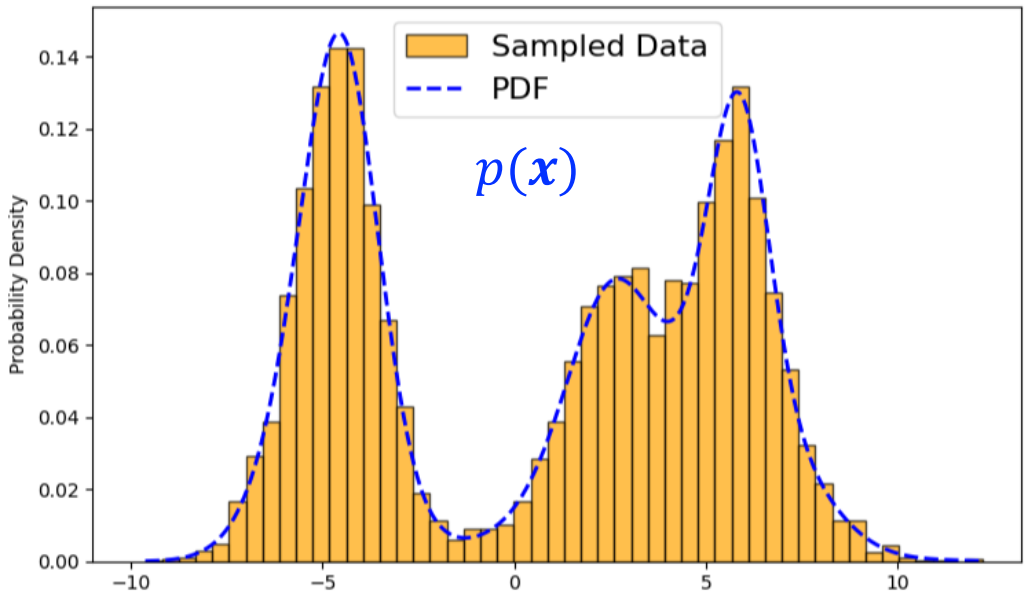
The sampling distribution

- Dataset $\mathbf{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_m, y_m)\}$
  - Each $\mathbf{x}_i$ is collected/generated according to a distribution P, i.e., $\boldsymbol{x}_i$ *is a sample from distribution P*
  - P is often **unknown** in practice.

The data-generation model

- Those samples are often assumed to be i.i.d.
  (Independent and identically distributed – Độc lập với nhau và được sinh ra từ một phân bố)
  - Easy for analysis, each example may contain maximal information





The sampling distribution

- Dataset **D** = {($\mathbf{x}_1$, y_1), ($\mathbf{x}_2$, y_2), …, ($\mathbf{x}_m$, y_m)}
- **Hardness** of the learning problem: The two unknown
  - y^* in the space of all measurable functions
  - P(**x**) in the space of all probability distributions

- In practice, we often find a function *h* to **approximate** y^*
  - h is often called **predictor** or *prediction rule*

Learning algorithm

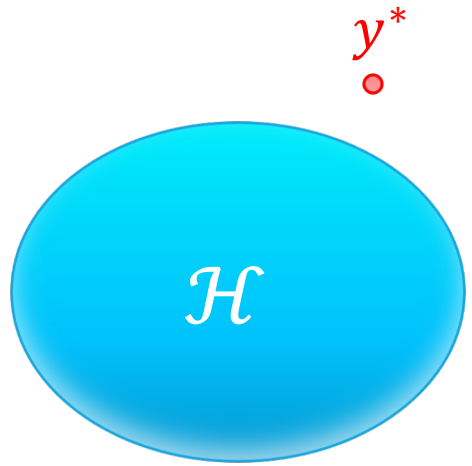
- **Learning algorithm (Learner):** an algorithm that can return one predictor h, based on a given training set **D**

$$\mathcal{A}: \boldsymbol{D} \mapsto h$$

  - Receive a training set **D**,
  - Can access the input space $\mathcal{X}$ and label space $\mathcal{Y}$
  - Return one predictor $h: \mathcal{X} \to \mathcal{Y}$
  - What space does a learning algorithm map from?
- E.g., K-nearest neighbors, ordinary least square
- The learner may not know any thing about the true y^* and P(**x**)

(Thuật toán học thường không biết gì thêm về hàm y^* và phân bố P)

Hypothesis space

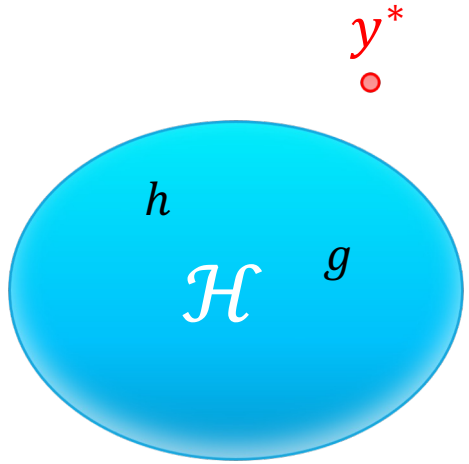


- Ussually, we can choose a restricted set $\mathcal{H}$ of functions
  - A learner must find one $h \in \mathcal{H}$
  - Our hope: $h(\boldsymbol{x}) \approx y^*(\boldsymbol{x}), \quad \forall \boldsymbol{x} \in \mathcal{X}$
- Hypothesis space *(model space)*:

  a set $\mathcal{H}$ of functions, providing candidates h for a learning algorithm
  - Each candidate depends on input features: h: $\mathcal{X} \to \mathcal{Y}$
  - Represents prior knowledge about a task
  - Represents our ***inductive bias*** or *preference*
- Each h is often called a "**model**" or "**predictor**"
- E.g., Linear model: $\mathcal{H} = \{h(\boldsymbol{x}, \boldsymbol{w}) = w_0 + w_1 x_1 + \cdots + w_n x_n \colon w_0, w_1, \ldots, w_n \in \mathbb{R}\}$

Loss function

- How to see the goodness of each $h \in \mathcal{H}$?
  - How well does h predict each input **x**?
  - How to compare two predictors h and g?
- Loss (cost) function: $f: \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$
  - $f(y, \hat{y})$: the cost/loss of prediction $\hat{y}$ about y
  - Tells the quality of each prediction
- Examples:
  - 0-1 loss: $f(y, \hat{y}) = \mathbf{1}_{y \neq \hat{y}}$
  - Square loss: $f(y, \hat{y}) = (y - \hat{y})^2$
  - Absolute loss: $f(y, \hat{y}) = |y - \hat{y}|$
  - Hinge loss: $f(y, \hat{y}) = \max(0, 1 - y\hat{y})$



For vectors:

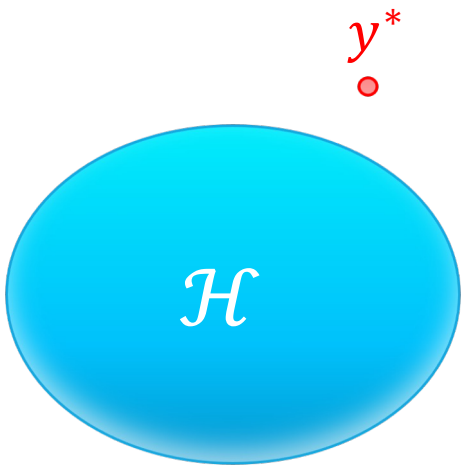
- Square (ℓ_2) loss: $f(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \|\boldsymbol{y} - \hat{\boldsymbol{y}}\|_2^2$
- Absolute (ℓ_1) loss: $f(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \|\boldsymbol{y} - \hat{\boldsymbol{y}}\|_1$

Loss or Risk

- When h makes a prediction about **x**, its loss is $f(y^*(\boldsymbol{x}), h(\boldsymbol{x}))$
- Empirical loss (lỗi thực nghiệm): the loss of a predictor h on the training set **D**

$$F(\boldsymbol{D}, h) = \frac{1}{M}\sum_{i=1}^{M} f(y_i, h(\boldsymbol{x}_i))$$

  - Sometimes known as **Empirical risk**



- Expected loss: the loss over the whole space

$$F(P, h) = \mathbb{E}_{\boldsymbol{x}\sim P}[f(y^*(\boldsymbol{x}), h(\boldsymbol{x}))]$$

  - Sometimes known as **Expected risk**

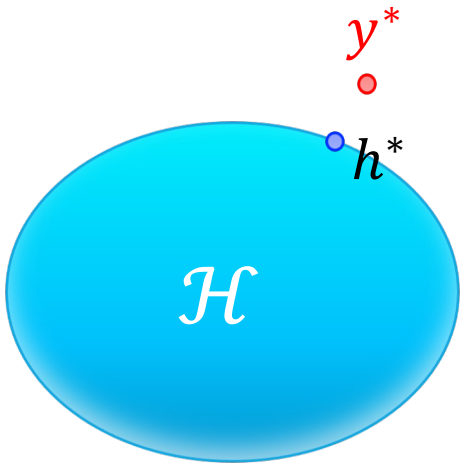
- For binary classification, where $\mathcal{Y} = \{0,1\}$, and absolute loss *f*:

$$\Pr\left(y^*(\boldsymbol{x}) \neq h(\boldsymbol{x})\right) = F(P, h)$$

- For $\mathcal{Y} = \{-1, 1\}$ and absolute loss *f*: $\Pr\left(y^*(\boldsymbol{x}) \neq h(\boldsymbol{x})\right) = \frac{1}{2}F(P, h)$

Learning goal

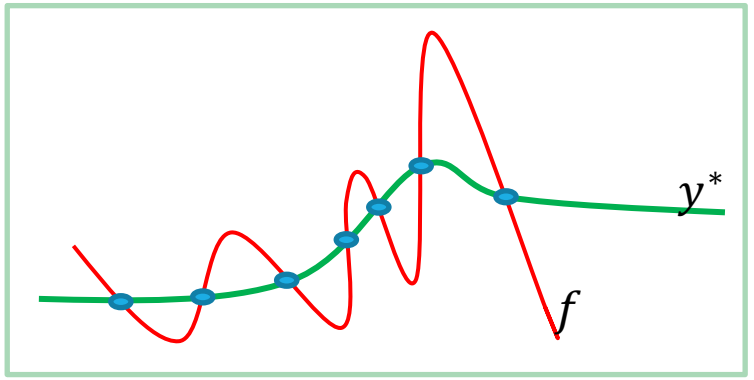
- Learner's task: find a good $h \in \mathcal{H}$, based on a training set **D**
- *Learning goal:* find one $h \in \mathcal{H}$ with *small expected loss*
  - It should **generalize** well on future data instances
  - Small training loss/error is not enough



- Ultimately, we want to find the best one in $\mathcal{H}$

$$h^* = \arg\min_{h \in \mathcal{H}} F(P, h)$$

- Learning ≠ Fitting
  - Fitting focuses on minimizing the training loss $F(\boldsymbol{D}, h)$



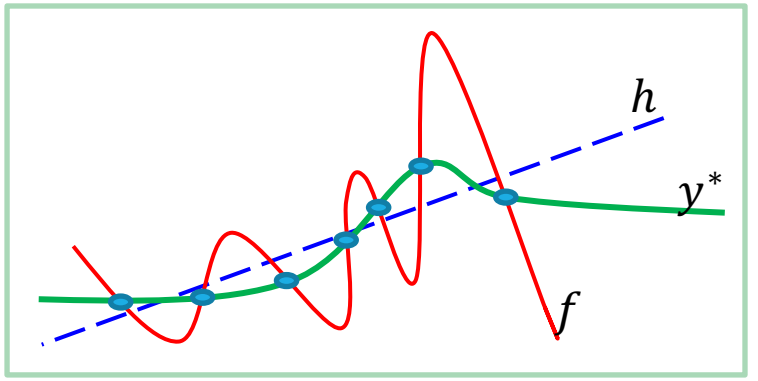
Empirical Risk Minimization (ERM)

- In practice, a learner cannot access $F(P, h)$
- It should rely on $F(\boldsymbol{D}, h)$
  - Can be computed from a training set **D**
- ERM will find

$$h_{erm} = \arg\min_{h \in \mathcal{H}} F(\boldsymbol{D}, h)$$

  - Focuses on fitting (minimizing the training error)
  - However, h_{erm} may get overfitting
- ERM is good in the cases of large training sets, due to

$$F(\boldsymbol{D}, h) \xrightarrow{m \to \infty} F(P, h)$$



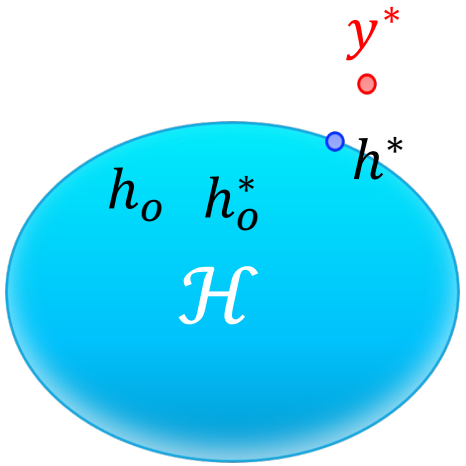
- Regularization-based algorithms:

$$h_{reg} = \arg\min_{h \in \mathcal{H}} F(\boldsymbol{D}, h) + \lambda . g(h)$$

  - $g(h)$ is a penalty on h, λ is a regularization constant
  - This penalty helps us avoid too complex candidates and hence reduce overfitting

Error of a model

- After training, the learning algorithm will return a model $h_o \in \mathcal{H}$
- How well does it work with future data? ⇒ Generalization!
- Maybe: $h_o \neq h_o^*$ and $h_o \neq h^*$
  - $h_o^* = \arg\min_{h \in \mathcal{H}} F(\boldsymbol{D}, h)$ is the *minimizer* of the empirical loss
  - $h^* = \arg\min_{h \in \mathcal{H}} F(P, h)$ is the *best member* of family $\mathcal{H}$



- Note:

$$F(P, h_o) - F(P, y^*) = F(P, h_o) - F(P, h^*) + F(P, h^*) - F(P, y^*)$$

- Estimation error: $F(P, h_o) - F(P, h^*)$
  - How good is the training algorithm?
- Approximation error: $F(P, h^*) - F(P, y^*)$
  - Capacity (representational power) of family $\mathcal{H}$

Bousquet et al. Introduction to statistical learning theory. In Machine Learning, LNAI, volume 3176. Springer, 2004.

Error decomposition

- Estimation error

$$F(P, h_o) - F(P, h^*)$$
$$= F(P, h_o) - F(\boldsymbol{D}, h_o) + F(\boldsymbol{D}, h_o) - F(\boldsymbol{D}, h_o^*) + F(\boldsymbol{D}, h_o^*) - F(P, h^*)$$

  - It can be decomposed into two types of error: Optimization + Generalization
  - Note that

$$|F(P, h_o) - F(\boldsymbol{D}, h_o)| \leq \sup_{h \in \mathcal{H}} |F(P, h) - F(\boldsymbol{D}, h)|$$

$$|F(\boldsymbol{D}, h_o^*) - F(P, h^*)| = \left| \min_{h \in \mathcal{H}} F(\boldsymbol{D}, h) - \min_{h \in \mathcal{H}} F(P, h) \right| \leq \sup_{h \in \mathcal{H}} |F(P, h) - F(\boldsymbol{D}, h)|$$

  - Therefore

$$|F(P, h_o) - F(P, h^*)| \leq |F(\boldsymbol{D}, h_o) - F(\boldsymbol{D}, h_o^*)| + 2 \sup_{h \in \mathcal{H}} |F(P, h) - F(\boldsymbol{D}, h)|$$

(Optimization error) (Generalization error)

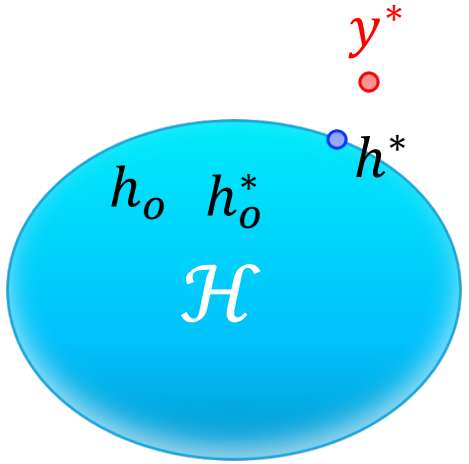
Error decomposition

$$|F(P,h_o) - F(P,h^*)| \leq |F(\boldsymbol{D},h_o) - F(\boldsymbol{D},h_o^*)| + 2\sup_{h\in\mathcal{H}} |F(P,h) - F(\boldsymbol{D},h)|$$

- Estimation error can be decomposed into two types of error

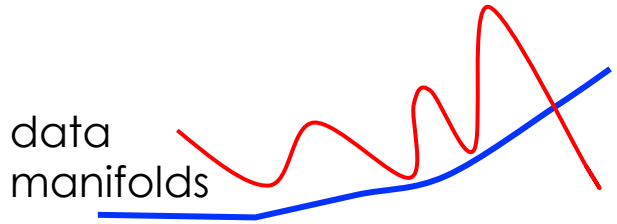
- Optimization error: $F(\boldsymbol{D},h_o) - F(\boldsymbol{D},h_o^*)$
  - How close to optimality is h_o?
  - h_o may not be the global minimum of the training loss
- Generalization error: $F(P,h_o) - F(\boldsymbol{D},h_o)$
  - How far is the training loss from expected loss?
  - Smaller should be better
- In summary:

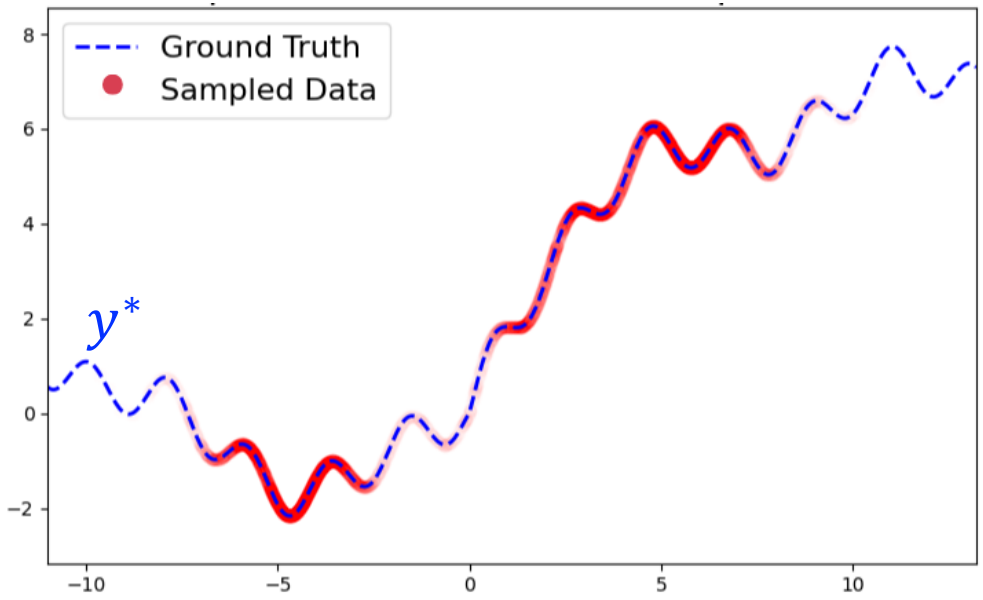
$Error(h_o) :=$ Optimization error +Generalization error +Approximation error

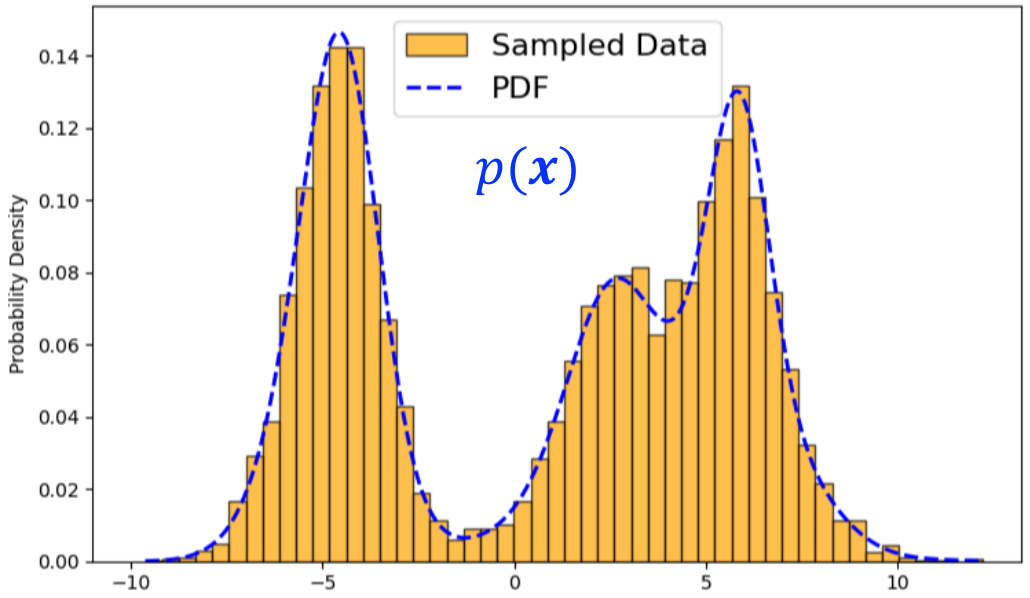


Errors by different factors

- **Data:** manifold
  - Complexity of the data space (độ phức tạp của không gian dữ liệu)
- **Data:** distribution
  - Complexity of the sampling distribution (độ phức tạp của phân bố lấy mẫu)
  - Representativeness of the training set, …

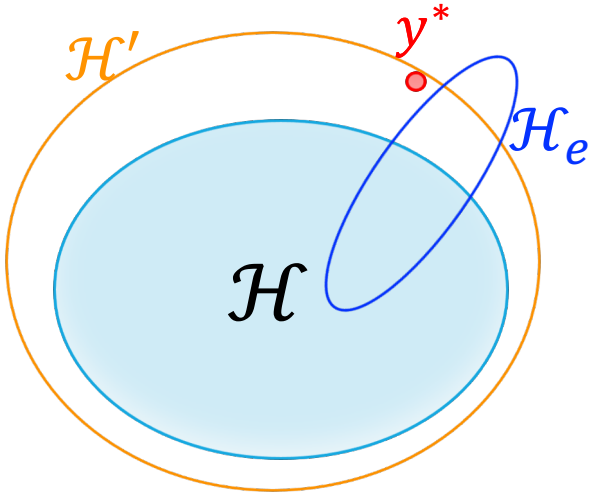






Errors by different factors (cont.)

- **Hypothesis space** $\mathcal{H}$
  - A bigger space ($\mathcal{H}'$), the (probably) smaller approximation error ➔Why?
  - More complex members, the (probably) smaller approximation error ➔ larger capacity
  - An *effective* space ($\mathcal{H}_e$) is enough ➔ not too big/complex
  - **Realizable** case: $F(P, h^*) = 0$
- **Training algorithm** $\mathcal{A}$
  - A better $\mathcal{A}$ implies smaller estimation error
  - A **bad** $\mathcal{A}$ can provide small optimization error, but large generalization error ➔ overfitting
  - A good $\mathcal{A}$ can localize an *effective* subset $\mathcal{H}^* \subset \mathcal{H}$



Errors by different factors (cont.)

- Each learner $\mathcal{A}$ can have its own properties and biases
  - Some focus on *sparse solutions*, e.g., LASSO
  - Some can find simple models with *low complexity* (e.g., most weights ≈ 0)
  - Some often return models which are *robust w.r.t small changes in the input*
  - Some often return models which are *robust w.r.t weight noises*
  - Some are stable, …
- Those biases should be used to analyze generalization
  - Generalization ability of a learning algorithm $\mathcal{A}$

Hypothesis space vs. Learner

- Consider the (unknown) regression function $y^*(\boldsymbol{x})$
- Let $\mathcal{A}_D$ be the regressor, learned by method $\mathcal{A}$ from a training set **D**
- Note: We want that $\mathcal{A}_D$ well approximates the truth y^*.
  - $\mathcal{A}_D$ is random, according to the randomness when collecting **D**.
- For any instance **x**, the error made by $\mathcal{A}_D$ is $(y^*(\boldsymbol{x}) - \mathcal{A}_D(\boldsymbol{x}))^2$
- ***The error made by the learner $\mathcal{A}$:***

  (Lỗi của thuật toán $\mathcal{A}$ khi phán đoán **x**)

$$err_A(\boldsymbol{x}) = \mathbb{E}_{\boldsymbol{D}}(y^*(\boldsymbol{x}) - \mathcal{A}_D(\boldsymbol{x}))^2$$

  - Why expectation? a different training set $\boldsymbol{D}$' will make $\mathcal{A}$ to return a different function $\mathcal{A}_D$,

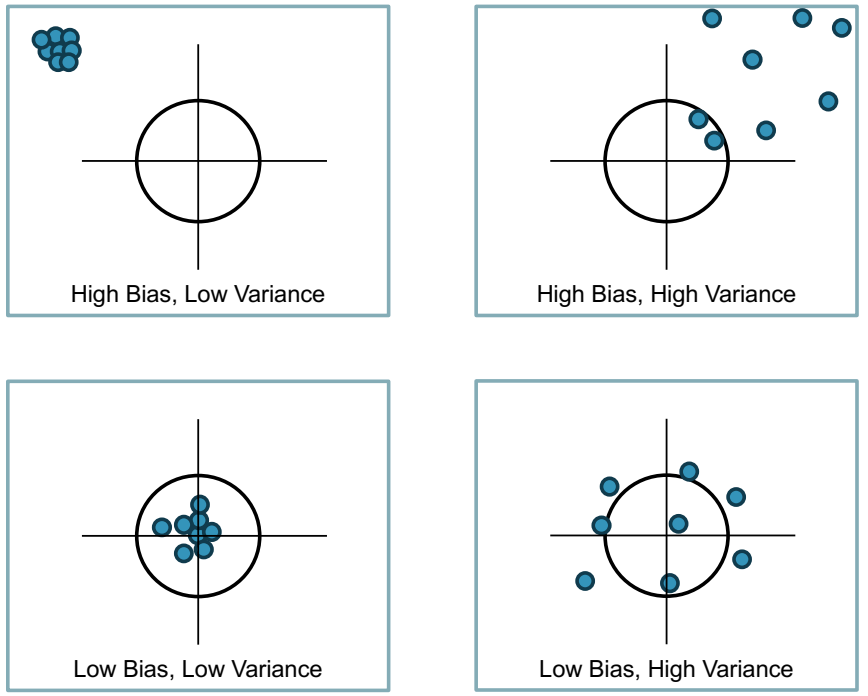
The Bias-Variance Decomposition

$$err_A(\boldsymbol{x}) = [Bias]^2 + Variance$$

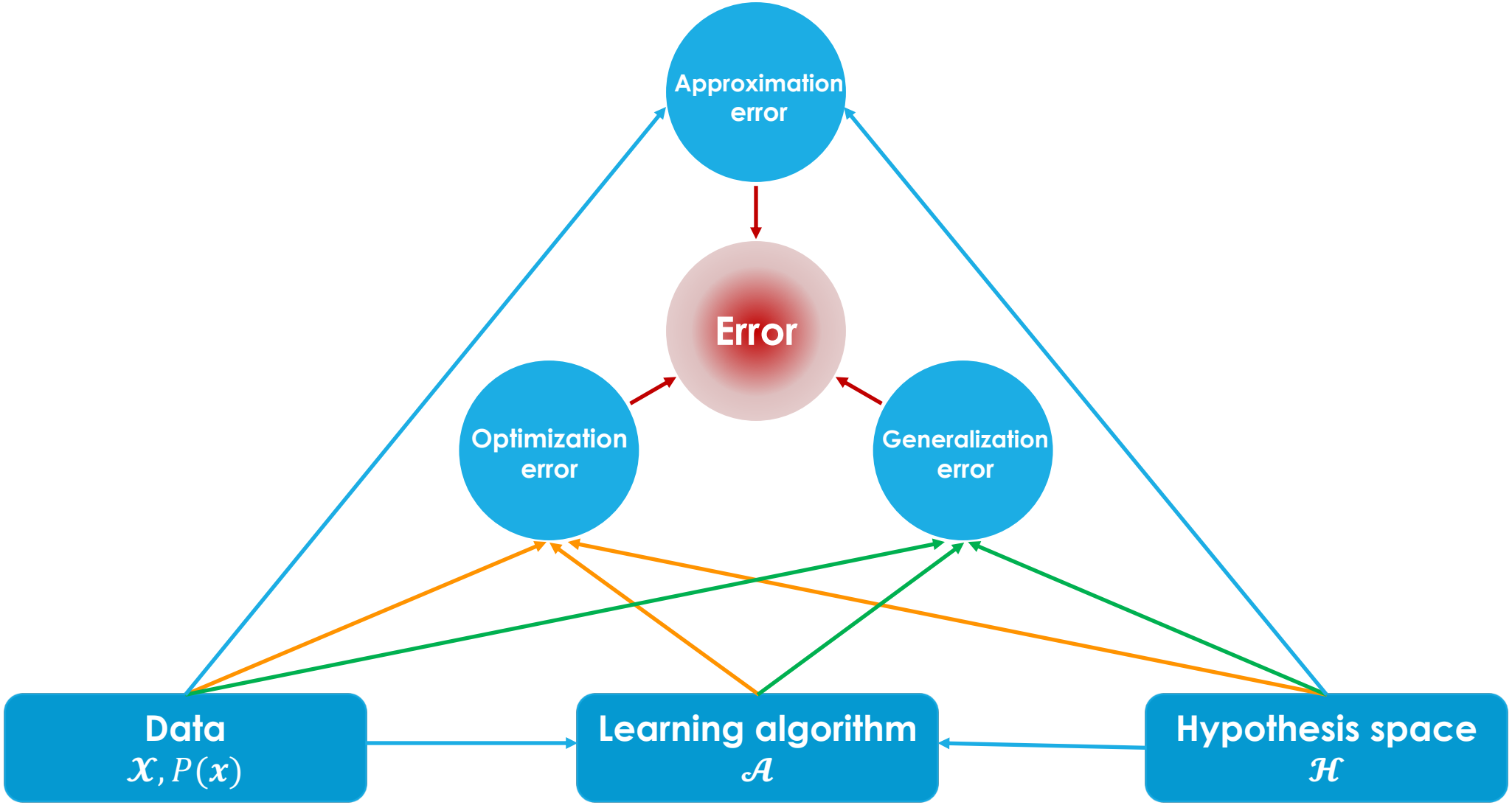
- $Bias = y^*(\boldsymbol{x}) - \mathbb{E}_{\boldsymbol{D}}[\mathcal{A}_D(\boldsymbol{x})]$; $\quad Variance = \mathbb{E}_{\boldsymbol{D}}\left[\left(\mathcal{A}_D(\boldsymbol{x}) - \mathbb{E}_{\boldsymbol{D}'}\mathcal{A}_{D'}(\boldsymbol{x})\right)^2\right]$

- This is known as **Bias-Variance Decomposition**
  - *Bias:* how far is the true value from the mean of predictions by method $\mathcal{A}$?
  - *Variance:* how much does each prediction by $\mathcal{A}$ vary around its mean?
  - Small bias? Increase model complexity
    ➔ Variance tends to increase
  - Small variance? Decrease model complexity
    ➔ Bias tends to increase

Trade-off



A unified view



Error bounds

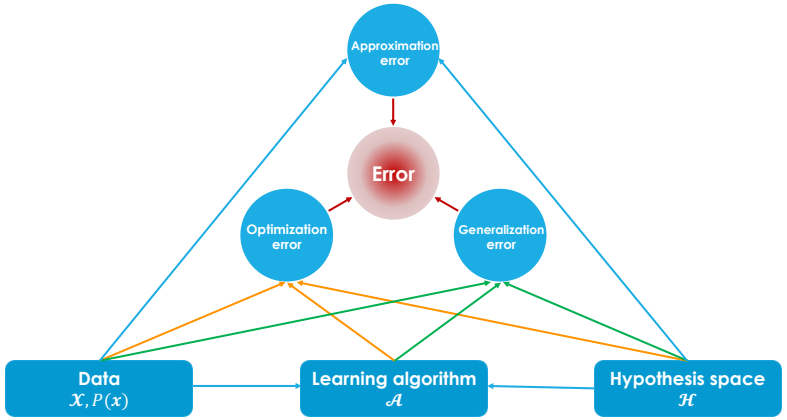
- Study theoretical guarantees by estimating the errors
  (tạo ra các đảm bảo lý thuyết bằng việc ước lượng lỗi)
- Approximation error:

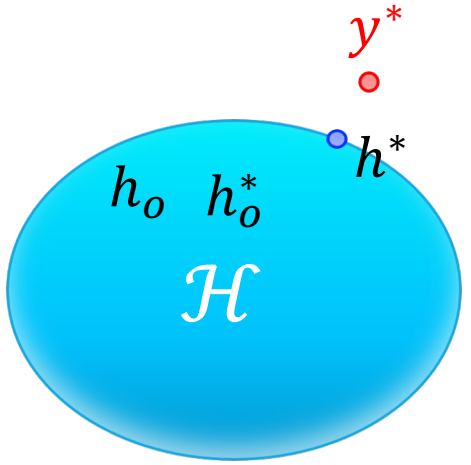
$$|F(P, y^*) - F(P, h^*)| \leq \epsilon_a$$

  - Capacity of family $\mathcal{H}$
- *Optimization error:*

$$|F(\boldsymbol{D}, h_o^*) - F(\boldsymbol{D}, h_o)| \leq \epsilon_o$$

  - Depending on the capacity of learning algorithm $\mathcal{A}$
  - and on the used optimizer





Bounds on Generalization Error

$$|F(P, h_o) - F(\boldsymbol{D}, h_o)| \leq \epsilon_g$$

- Generalizability of a learned function h_o

- Uniform bounds:

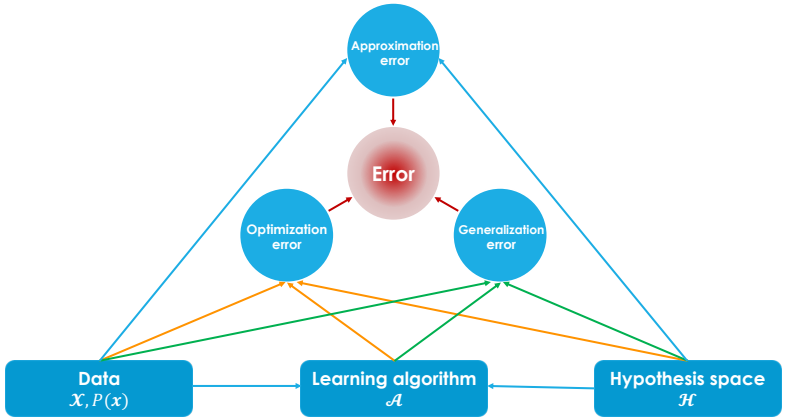
$$\sup_{h \in \mathcal{H}} |F(P, h) - F(\boldsymbol{D}, h)| \leq \epsilon_g$$

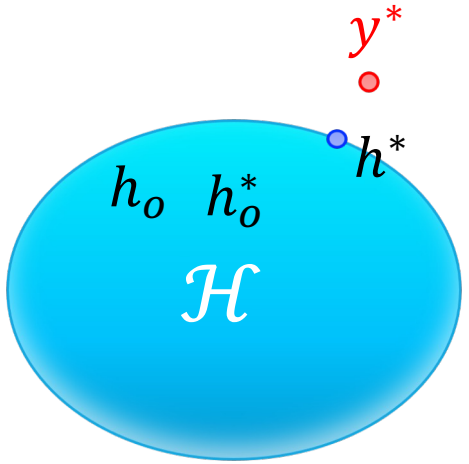
  - Focus on the **worst** member
  - May not be a good way to explain a learned function h_o

- PAC-Bayes bounds:

$$|\mathbb{E}_{h \in \mathcal{H}}[F(P, h) - F(\boldsymbol{D}, h)]| \leq \epsilon_g$$

  - Study the error on average over $\mathcal{H}$
  - May not explain a learned function h_o



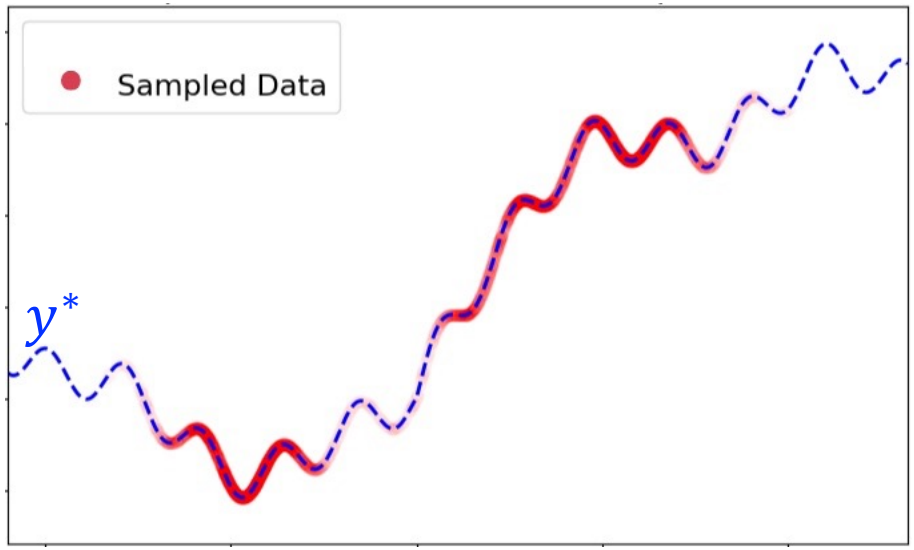


Bousquet et al. Introduction to statistical learning theory. In Machine Learning, LNAI, volume 3176. Springer, 2004.
Nagarajan & Kolter. Uniform convergence may be unable to explain generalization in deep learning. *Advances in Neural Information Processing Systems*. 2019.

Learnability

Generalization ability

- Consider a model $h_o \in \mathcal{H}$
  - Learned from a training set **D** ➔ h_o depends on **D**,
  - h_o is a function of **D**, and we can write $h_o = h_o(\boldsymbol{D})$
- We want to estimate $F(P, h_o)$
  - *How well can h_o generalize on unseen data?*
  - But P is unknown !!!
  - Note that $F(P, h_o)$ depends on **D**
  - When **D** changes, so does $F(P, h_o)$
- Training set **D**
  - Generated by a random process ➔ sometimes good, sometimes bad for training
  - There is always some probability that **D** happens to be very *nonrepresentative* of the underlying distribution P



Probabilistic bounds

- An estimate of $F(P, h_o)$ should be in the form of **probabilistic bounds**

$$\Pr(F(P, h_o) > \epsilon) < \delta$$

- Interpret:
  - The expected loss is greater than ϵ, with probability at most δ
    (xác suất để lỗi trung bình vượt quá ϵ là nhỏ hơn δ)
  - δ is the probability of getting a nonrepresentative sample **D**, so that $F(P, h_o)$ is large
  - ϵ is our prediction about the expected loss (phán đoán của ta về lỗi trung bình)
- Hope: *small ϵ and δ*
- Probability to sample a training set for which $F(P, h_o)$ is small

$$\Pr(F(P, h_o) \leq \epsilon) = \Pr(\{\boldsymbol{D}\colon F(P, h_o) \leq \epsilon\}) \quad \geq \quad 1 - \delta$$

  - $1 - \delta$ represents **confidence** of our prediction ϵ

Approximately correct predictor

- When $F(P, h_o) \leq \epsilon$, we call h_o as *approximately correct predictor*
- The failure of the learner: $F(P, h_o) > \epsilon$
- We want to **upper bound** the probability of this event:

$$\Pr(F(P, h_o) > \epsilon)$$

  - When **D** contains m i.i.d samples from P, we can consider **D** to be generated from distribution $P^m = P \times P \times \cdots \times P$. Then

$$\Pr(F(P, h_o) > \epsilon) = P^m(\{\boldsymbol{D}\colon F(P, h_o) > \epsilon\})$$

  - Tỉ lệ của vùng mà chứa các tập mẫu sẽ tạo ra lỗi lớn
- Sometimes we want to **lower** bound

$$\Pr(F(P, h_o) > \epsilon) > ???$$

Provably approximately correct (PAC)

- **Learnability**: How to define it?
  - PAC learning was introduced by Leslie Valiant (1984), the winner of the 2010 Turing Award for the introduction of the PAC model
- **PAC learnability:** A hypothesis class $\mathcal{H}$ is *PAC learnable* if there exist a learning alg. and a function $\mathrm{m_H}: (0,1)\times(0,1) \to \mathbb{N}$ with the following property:
  For every $\epsilon, \delta \in (0,1)$, for every distribution P over $\mathcal{X}$, and for every labeling function $y^*: \mathcal{X} \to \mathcal{Y}$, if the realizable assumption holds, then when running the learning algorithm on $m \geq m_H(\epsilon, \delta)$ i.i.d. examples generated by P and labeled by y^*, the algorithm returns a hypothesis h_o such that $F(P, h_o) \leq \epsilon$ with probability of at least $1 - \delta$ (over the choice of the examples)

$$\Pr(F(P, h_o) \leq \epsilon) \quad \geq \quad 1 - \delta$$

  - Sau khi học từ một tập có ít nhất $\mathrm{m_H}$ mẫu, một thuật toán học sẽ trả về một hàm h_o thoả mãn $\Pr(F(P, h_o) \leq \epsilon) \geq 1 - \delta$
  - *Realizable assumption:* $F(P, h^*) = 0$ for the best member $h^* = \arg\min_{h \in \mathcal{H}} F(P, h)$

Valiant, L. G. (1984). A theory of the learnable. *Communications of the ACM*, *27*(11), 1134-1142.

Provably approximately correct (PAC)

$$\Pr(F(P, h_o) \leq \epsilon) \quad \geq \quad 1 - \delta$$

- The accuracy parameter ϵ determines how far h_o can be from the optimal one (this corresponds to the “approximately correct”),
- The confidence δ indicates how likely h_o is to meet that accuracy requirement (corresponds to the “probably” part of “PAC”).

Valiant, L. G. (1984). A theory of the learnable. *Communications of the ACM*, *27*(11), 1134-1142.

Agnostic PAC learnability

- Realizable cases are rare in practice
  - They require $\mathcal{H}$ to be too huge to "contain" the truth y^*
- **Non-realizable** cases: $F(P, h^*) \neq 0$ for the best member h^*
  - Hard problems
  - Small family $\mathcal{H}$, or each member is not complex enough
- Agnostic PAC learnability: … *when running the learning algorithm on $m \geq m_H(\epsilon, \delta)$ i.i.d. examples generated by P and labeled by y^*, the algorithm returns a hypothesis h_o such that, with probability at least $1 - \delta$ (over the choice of the examples):*

$$F(P, h_o) \leq F(P, h^*) + \epsilon$$

  - No learner can learn a model with arbitrarily small error
  - A learner can still declare success if its error is not much larger than the best error achievable by a predictor from $\mathcal{H}$

Shalev-Shwartz, Shai, and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge University Press, 2014.

Sample complexity

- **Sample complexity**: the minimal number of examples are required to guarantee a probably approximately correct solution
- ***Lemma:*** *if the realizable assumption holds, every finite hypothesis class* $\mathcal{H}$ *is PAC learnable with sample complexity* $m_H(\epsilon, \delta) \leq \left\lceil \frac{\log(|\mathcal{H}|/\delta)}{\epsilon} \right\rceil$

- $m_H(\epsilon, \delta)$ is often required to be *polynomial* in $1/\epsilon$ and $\log(\delta)$
  - to say learnability of $\mathcal{H}$ [Valiant, 1984]
- *The curse of dimensionality:* $m_H(\epsilon, \delta)$ is exponential in the dimensionality n

(lời nguyền của số chiều: nếu lượng mẫu cần thiết có cỡ cấp hàm mũ của n)

  - $\mathcal{H}$ không thực tế!

Shalev-Shwartz, Shai, and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge University Press, 2014.

No-free-lunch theorem

- For one problem, there are many learners
  - Some are really powerful, e.g., using SGD
  - More training samples can help training to be loss closer and closer to expected loss

  ➔ the trained model can be better
- **Is there a universal learner that can learn any functions** just by increasing training size?
- **No-free-lunch theorem:** *Let $\mathcal{A}$ be any learner for the task of binary classification with input space $\mathcal{X}$, the 0-1 loss, and any training size $m \leq |\mathcal{X}|/2$. Then there exists a distribution P over $\mathcal{X} \times \{0,1\}$ satisfying:*
  - *There exists a function h with $F(P,h) = 0$*
  - *$F(P, \mathcal{A}(\boldsymbol{D})) \geq 1/8$, with with probability of at least $1/7$ over the choice of $\boldsymbol{D} \sim P^m$*
- No universal learner !

Assessing the Generalization ability

Basic bounds

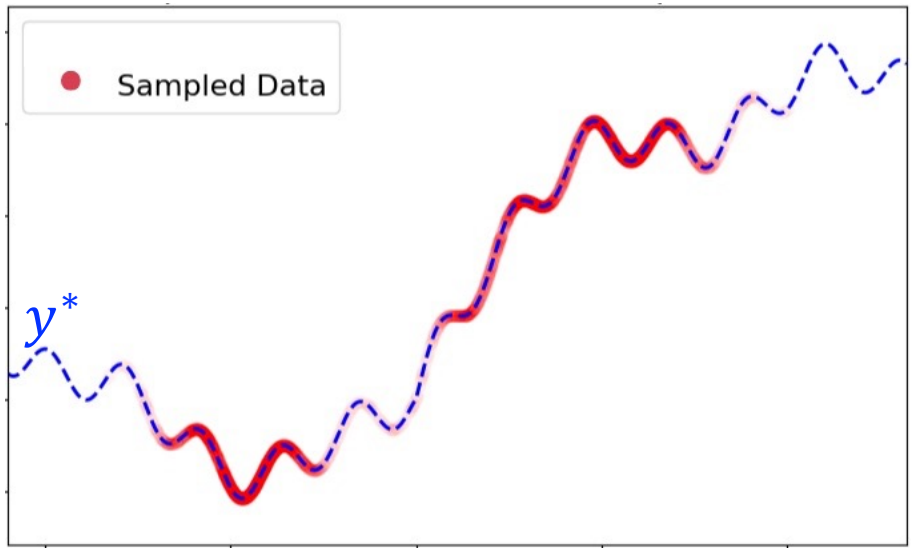
Generalization ability

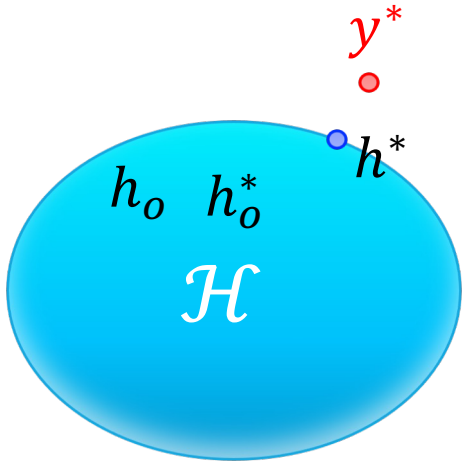
- Consider a model $h_o \in \mathcal{H}$
  - Learned from a training set **D**
  - h_o is a function of **D**, and we can write $h_o = h_o(\boldsymbol{D})$
- We want to estimate $F(P, h_o)$
  - *How well can h_o generalize on unseen data?*
  - But P is unknown !!!
  - Need to assure

$$\Pr(F(P, h_o) \leq F(P, h^*) + \epsilon) \quad \geq \quad 1 - \delta$$

- Another way: estimate Generalization error

$$F(P, h_o) - F(\boldsymbol{D}, h_o)$$





Bounds on Generalization error

$$F(P, h_o) - F(\boldsymbol{D}, h_o)$$

- $F(\boldsymbol{D}, h_o)$ is computed from **D** with *m* samples, and is an empirical version of $F(P, h_o)$

- By the law of large numbers: $\Pr\left(\lim_{m\to\infty} F(\boldsymbol{D}, h_o) = F(P, h_o)\right) = 1$
  - But this law does not provide an explicit form of the *convergence rate*

- **Theorem (Hoeffding):** *Let $z_1, z_2, \ldots, z_m$ be i.i.d. random variables and function $g(z) \in [a, b]$. Then for all $\epsilon > 0$, we have*

$$\Pr\left(\mathbb{E}_z[g(z)] - \frac{1}{m}\sum_{i=1}^{m} g(z_i) > \epsilon\right) \leq \exp\left(-\frac{2m\epsilon^2}{(b-a)^2}\right)$$

  - Denote the right hand side as δ, then

$$\Pr\left(\mathbb{E}_z[g(z)] - \frac{1}{m}\sum_{i=1}^{m} g(z_i) > (b-a)\sqrt{\frac{1}{2m}\log\frac{1}{\delta}}\right) \leq \delta$$

Basic bounds: Hoeffding's inequality

- Or with probability at least $1 - \delta$, we have

$$\mathbb{E}_z[g(z)] - \frac{1}{m}\sum_{i=1}^{m} g(z_i) \;\leq\; (b-a)\sqrt{\frac{1}{2m}\log\frac{1}{\delta}} \qquad (1)$$

  - This is true for any *fixed* function g (đúng đối với một hàm g cố định)
  - However, it does not apply when g depends on the sampling of the data (không đúng nếu g phụ thuộc vào tập $z_1, z_2, \ldots, z_m$)
- **Limitation:** for a given fixed function g
  - There exists a sample **D** of size *m*, so that inequality (1) holds
  - The set S of those samples has measure $\Pr(\boldsymbol{D} \in S) \geq 1 - \delta$
  - However, a different function g' may have a different S'
  - For a given sample **D**, only some members of $\mathcal{H}$ can satisfy (1)
    ➔ a model h_o trained from **D** may not satisfy (1)

Uniform bound

- Before seeing the data, which function will the algorithm choose?
  - We do not know!
- Consider

$$\sup_{h \in \mathcal{H}} [F(P, h) - F(\boldsymbol{D}, h)] \quad (2)$$

  - Note: $F(P, h_o) - F(\boldsymbol{D}, h_o) \leq \sup_{h \in \mathcal{H}} [F(P, h) - F(\boldsymbol{D}, h)]$
- If we can upper bound (2), we are done!
  - This bound on error will hold true for every member of $\mathcal{H}$
    (cận trên này về lỗi sẽ đúng đối với mọi hàm trong $\mathcal{H}$)
  - It tries to estimate the error of the **worst** member !!!
    (tức là ta xem xét lỗi của thành viên tệ nhất)

Uniform bound for finite hypothesis class

- Assume that $\mathcal{H}$ is finite, and $|\mathcal{H}| = N$ (tập $\mathcal{H}$ có N phần tử)
- For each $h_i \in \mathcal{H}$, let $\mathbf{D}_i$ be a '*bad*' sample so that $F(P, h_i) - F(\boldsymbol{D}_i, h_i) > \epsilon$
  - Hoeffding's inequality: $\Pr(\boldsymbol{D}_i) = \Pr(\boldsymbol{D}_i : F(P, h_i) - F(\boldsymbol{D}_i, h_i) > \epsilon) \leq \exp\left(-\frac{m\epsilon^2}{2C^2}\right)$
    (C is the maximum value of the loss function)
- Union bound: $\Pr(\boldsymbol{D}_i \cup \boldsymbol{D}_j) \leq \Pr(\boldsymbol{D}_i) + \Pr(\boldsymbol{D}_j)$
  - Generally: $\Pr(\boldsymbol{D}_1 \cup \boldsymbol{D}_2 \cup \cdots \cup \boldsymbol{D}_N) \leq \sum_{i=1}^{N} \Pr(\boldsymbol{D}_i)$
- As a result: $\Pr(\exists h \in \mathcal{H} : F(P, h) - F(\boldsymbol{D}, h) > \epsilon) \leq \sum_{i=1}^{N} \Pr(\boldsymbol{D}_i) \leq N \exp\left(-\frac{m\epsilon^2}{2C^2}\right)$
- Hence, for all $\delta > 0$, with probability at least $1 - \delta$:

$$\forall h \in \mathcal{H}, \qquad F(P, h) \leq F(\boldsymbol{D}, h) + C\sqrt{\frac{2\log N - 2\log \delta}{m}}$$

PAC learnability: a revisit

- Consider realizable cases and finite hypothesis class $\mathcal{H}$
- Let ERM as the learner that always returns $h^*_{erm} = \arg\min_{h \in \mathcal{H}} F(\boldsymbol{D}, h)$
  - The realizable assumption implies $F(\boldsymbol{D}, h^*_{erm}) = 0$
- Let $\mathcal{H}_{erm}$ contain all functions returns by this learner
  - Then, for all $\delta > 0$, with probability at least $1 - \delta$:

$$\forall h_{erm} \in \mathcal{H}_{erm}, \qquad F(P, h_{erm}) \leq C \sqrt{\frac{2 \log N - 2 \log \delta}{m}}$$

- For all $\epsilon > 0$, letting $m_H = C^2(2 \log N - 2 \log \delta)/\epsilon^2$, for every $m \geq m_H$:

$$\Pr(F(P, h_{erm}) \leq \epsilon) \quad \geq \quad 1 - \delta$$

- $\mathcal{H}$ is PAC learnable!

Infinite hypothesis class

$$\forall h \in \mathcal{H}, \qquad F(P,h) \leq F(\boldsymbol{D},h) + C\sqrt{\frac{2\log N - 2\log\delta}{m}}$$

This bound does not apply to the infinite case

$$N = |\mathcal{H}| = \infty$$

Different approaches

- Performance of a model depends on:
  - The learning algorithm $\mathcal{A}$
  - The model family $\mathcal{H}$
  - Data distribution P
  - Labeling function y^*
- Approaches to bounding generalization ability:
  - Complexity/capacity of family $\mathcal{H}$ ➔ Complexity-based bounds
  - Capacity of the learning algorithm $\mathcal{A}$ ➔ Algorithmic-based bounds
  - Complexity of the sampling distribution P ➔ Data-dependent bounds
  - Capacity of the trained model ➔ Model-dependent bounds

Algorithmic robustness

- Robust learner can receive a training set **D** to
  - produce a model which has similar loss around the training samples
  - return a model which can generalize well to the local areas around **D**
- [Xu & Mannor, 2012] formalize the concept of Robustness
- **Definition:** *A learning algorithm $\mathcal{A}$ is (K, ϵ)-**robust**, for $K \in \mathbb{N}$ and $\epsilon: \mathcal{Z}^m \to \mathbb{R}$, if $\mathcal{Z}$ can be partitioned into K disjoint sets, denoted by $\{\mathcal{Z}_k\}_{k=1}^K$, such that the following holds for all $\boldsymbol{D} \in \mathcal{Z}^m$:*

$$\forall \boldsymbol{s} \in \boldsymbol{D}, \forall \boldsymbol{z} \in \mathcal{Z}, if\ \boldsymbol{s}, \boldsymbol{z} \in \mathcal{Z}_k \text{ for some } k \in [K] \text{ then } |\ell(\mathcal{A}_D, \boldsymbol{s}) - \ell(\mathcal{A}_D, \boldsymbol{z})| \leq \epsilon(\boldsymbol{D})$$

  - Where $\ell(\mathcal{A}_D, \boldsymbol{s})$ is the loss of model $\mathcal{A}_D$ at example **s**, and $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ for supervised problems
  - Every model returned from $\mathcal{A}$ should be robust at some small areas around the training points (mỗi mô hình học được sẽ vững trước sự thay đổi nhỏ xung quanh các mẫu học)
  - A *non-robust* model will be sensitive to some small changes in the input
    ➔ Vulnerable to Adversarial Attacks

Xu H, Mannor S. Robustness and generalization. *Machine learning*. 2012.

Robustness-based bound

- **Theorem:** *Assume that $\mathcal{A}$ is (K, ϵ)-robust. Let **D** contain m i.i.d samples from distribution P, and h is trained from **D** by $\mathcal{A}$, and f be the loss with max value C. For any $\delta > 0$, with probability at least $1 - \delta$,*

$$F(P, h) \leq F(\boldsymbol{D}, h) + \epsilon(\boldsymbol{D}) + C\sqrt{\frac{2K \log 2 + 2\log(1/\delta)}{m}} \quad (9)$$

  - When both ϵ and training loss is small, the expected loss can be small
    ➔h can generalize well and avoid adversarial attacks
  - A robust learner can return models with high generalization ability
    but the reverse is not true!
  - *Tradeoff:* increasing K can reduce ϵ
- Idea to improve generalization
  - Penalize the robustness level ϵ when training h ➔ Robust training, adversarial training

Xu H, Mannor S. Robustness and generalization. *Machine learning*. 2012.

Robust algorithms

- **Learners with Lipschitz loss:** if the instance set $\mathcal{Z}$ is compact and the loss f is L-Lipschitz continuous in the input, then $\mathcal{A}$ is $(\mathcal{N}(\gamma/2, \mathcal{Z}), L\gamma)$-robust for all $\gamma > 0$.
  - $\mathcal{N}(\gamma/2, \mathcal{Z})$ is the covering number, i.e., the minimal number of balls with radius at most $\gamma/2$ that cover $\mathcal{Z}$ completely
    (số lượng ít nhất các hình cầu có bán kính không quá $\gamma/2$ mà có thể bao toàn bộ $\mathcal{Z}$)
- **LASSO:** $(\mathcal{N}(\gamma/2, \mathcal{Z}), \gamma\frac{1}{m}\sum_{i=1}^{m} y_i^2 /\lambda + \gamma)$-robust for all $\gamma > 0$

Minimize
$F(\boldsymbol{D}, h) + \lambda\|\boldsymbol{w}\|_1$

- **PCA:** $(\mathcal{N}(\gamma/2, \mathcal{Z}), 2\gamma dB)$-robust, if we use d components and B is the diameter of $\mathcal{Z}$
- **Max margin learners:** $(\mathcal{N}(\gamma/2, \mathcal{Z}), \gamma)$-robust
  - where γ is the margin of $\mathcal{A}_D$:
    $\mathcal{A}_D(\boldsymbol{x}) = \mathcal{A}_D(\boldsymbol{s})$ for any $\boldsymbol{x} \in \mathcal{X}$ and $\boldsymbol{s} \in \boldsymbol{D}$ in the same ball with diameter γ
- **Learners for neural networks:** $(\mathcal{N}(\gamma/2, \mathcal{Z}), \gamma W^d)$-robust
  - For DNNs with d layers, each weight matrix has norm at most W

Algorithmic robustness: issues

$$F(P,h) \leq F(\boldsymbol{D},h) + \epsilon(\boldsymbol{D}) + C\sqrt{\frac{2K\log 2 + 2\log(1/\delta)}{m}}$$

- K is often large to ensure a small ϵ
  - Previous examples: $\mathrm{K} = \mathcal{N}(\gamma/2, \mathcal{Z})$ ➔ exponential in the dimension
  - ➔ Vacuous/trivial bound
- Vacuousness also come from ϵ
  - By definition, $\epsilon(\boldsymbol{D}) = \sup\{|\ell(h,\boldsymbol{s}) - \ell(h,\boldsymbol{z})| : \boldsymbol{s} \in \boldsymbol{D}, \boldsymbol{z} \in \mathcal{Z}_k \text{ where } \mathcal{Z}_k \text{ contains } \boldsymbol{s}\}$
  - For 0-1 loss, one incorrect prediction may lead to $\epsilon = 1$ ➔ Vacuous