Learning From Data
Lecture 11: Model Selection & Learning Theory

Yang Li yangli@sz.tsinghua.edu.cn

TBSI

December 17, 2020

Today's Lecture

Practical tools to improve machine learning performance:

- Bias and variance trade off
- Model selection
- A Brief Introduction to learning theory

Start on your project early!

Empirical error & Generalization error

Consider a learning task, the **empirical (training) error** of hypothesis h is the expected loss over m training samples

$$\hat{\epsilon}(h) = \frac{1}{m}\sum_{i=1}^{m} 1\{h(x^{(i)}) \neq y^{(i)}\} \quad \text{(classification, 0-1 loss)}$$

$$\hat{\epsilon}(h) = \frac{1}{m}\sum_{i=1}^{m} ||h(x^{(i)}) - y^{(i)}||_2^2 \quad \text{(regression, least-square loss)}$$

Empirical error & Generalization error

Consider a learning task, the **empirical (training) error** of hypothesis h is the expected loss over m training samples

$$\hat{\epsilon}(h) = \frac{1}{m}\sum_{i=1}^{m} 1\{h(x^{(i)}) \neq y^{(i)}\} \quad \text{(classification, 0-1 loss)}$$

$$\hat{\epsilon}(h) = \frac{1}{m}\sum_{i=1}^{m} ||h(x^{(i)}) - y^{(i)}||_2^2 \quad \text{(regression, least-square loss)}$$

The **generalization (testing) error** of h is the expected error on examples not necessarily in the training set.

Empirical error & Generalization error

Consider a learning task, the **empirical (training) error** of hypothesis h is the expected loss over m training samples

$$\hat{\epsilon}(h) = \frac{1}{m}\sum_{i=1}^{m} 1\{h(x^{(i)}) \neq y^{(i)}\} \quad \text{(classification, 0-1 loss)}$$

$$\hat{\epsilon}(h) = \frac{1}{m}\sum_{i=1}^{m} ||h(x^{(i)}) - y^{(i)}||_2^2 \quad \text{(regression, least-square loss)}$$

The **generalization (testing) error** of h is the expected error on examples not necessarily in the training set.

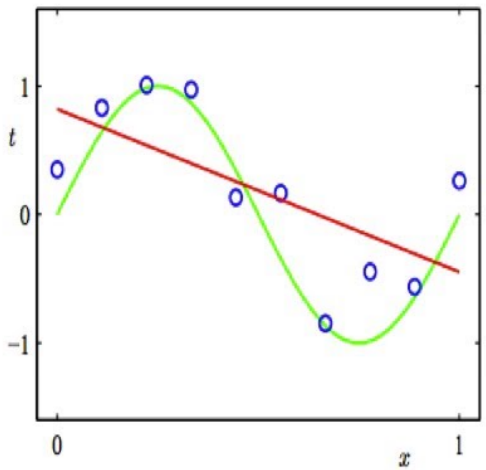
Goal of machine learning

- make training error small (optimization)
- make the gap between empirical and generalization error small

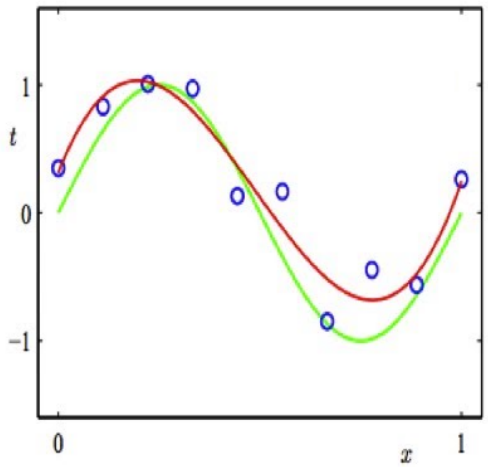
Overfit & Underfit

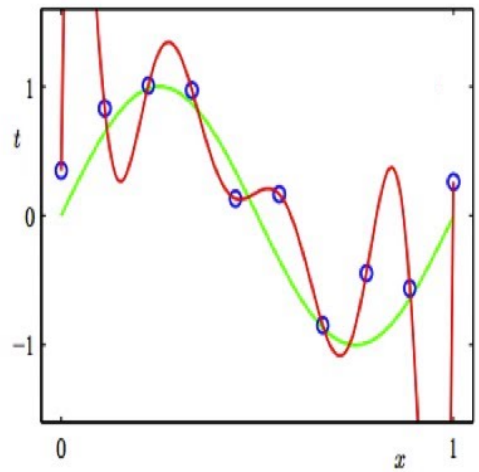
Underfit Both training error and testing error are large

Overfit Training error is small, testing error is large



underfit



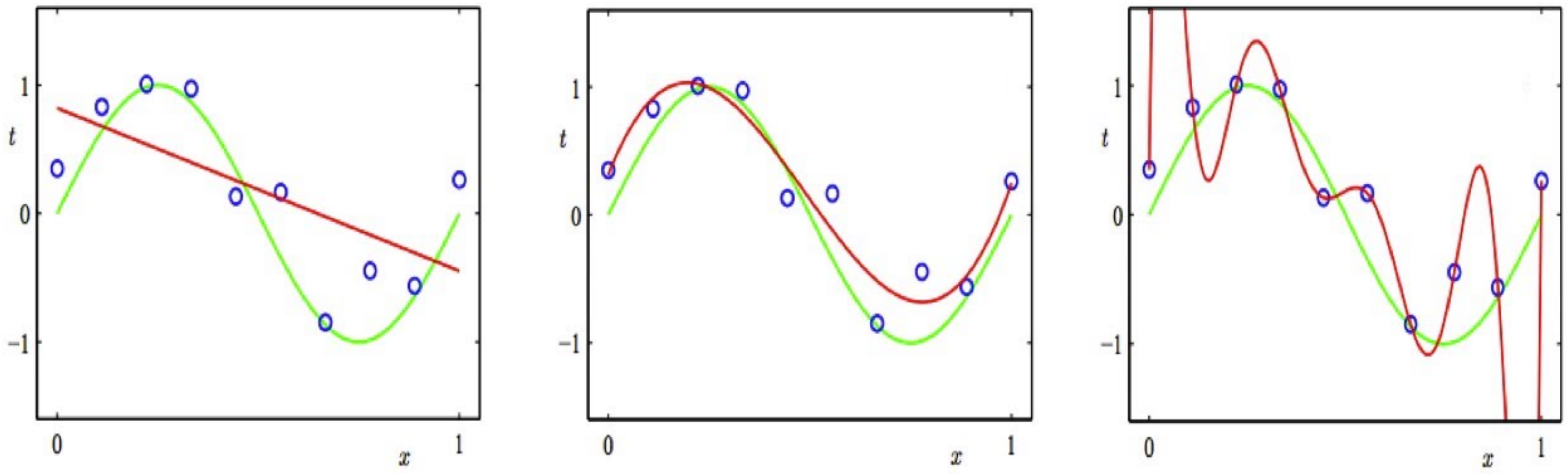


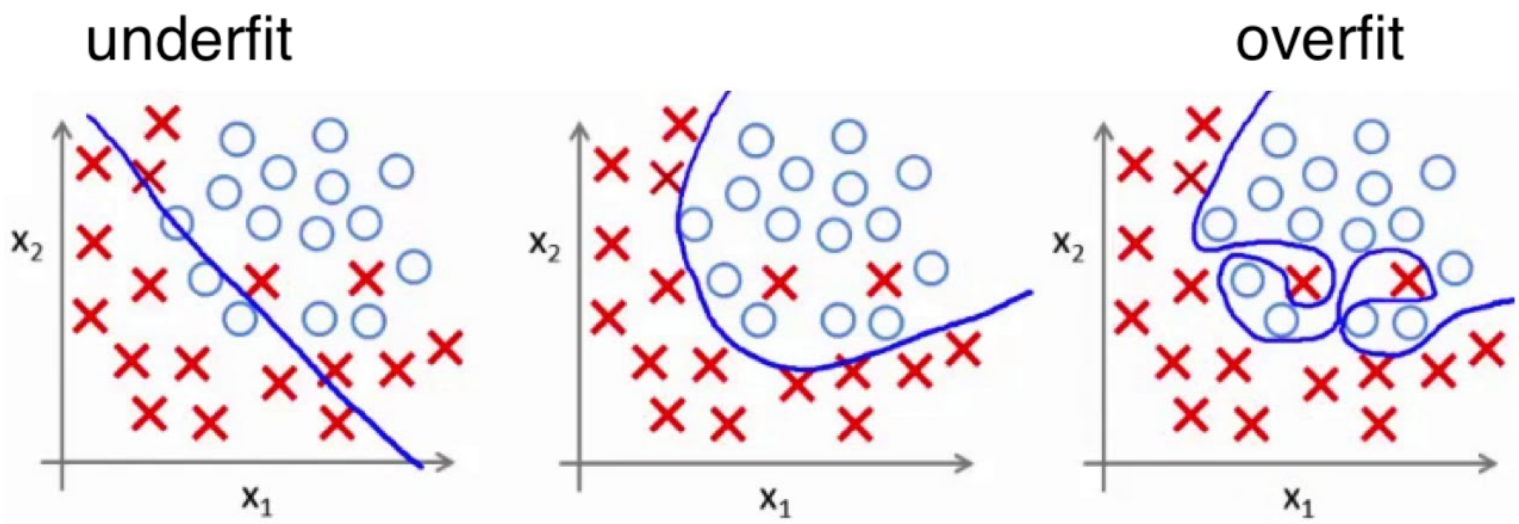
overfit

Overfit & Underfit

Underfit Both training error and testing error are large

Overfit Training error is small, testing error is large





Model capacity: the ability to fit a wide variety of functions

Model Capacity

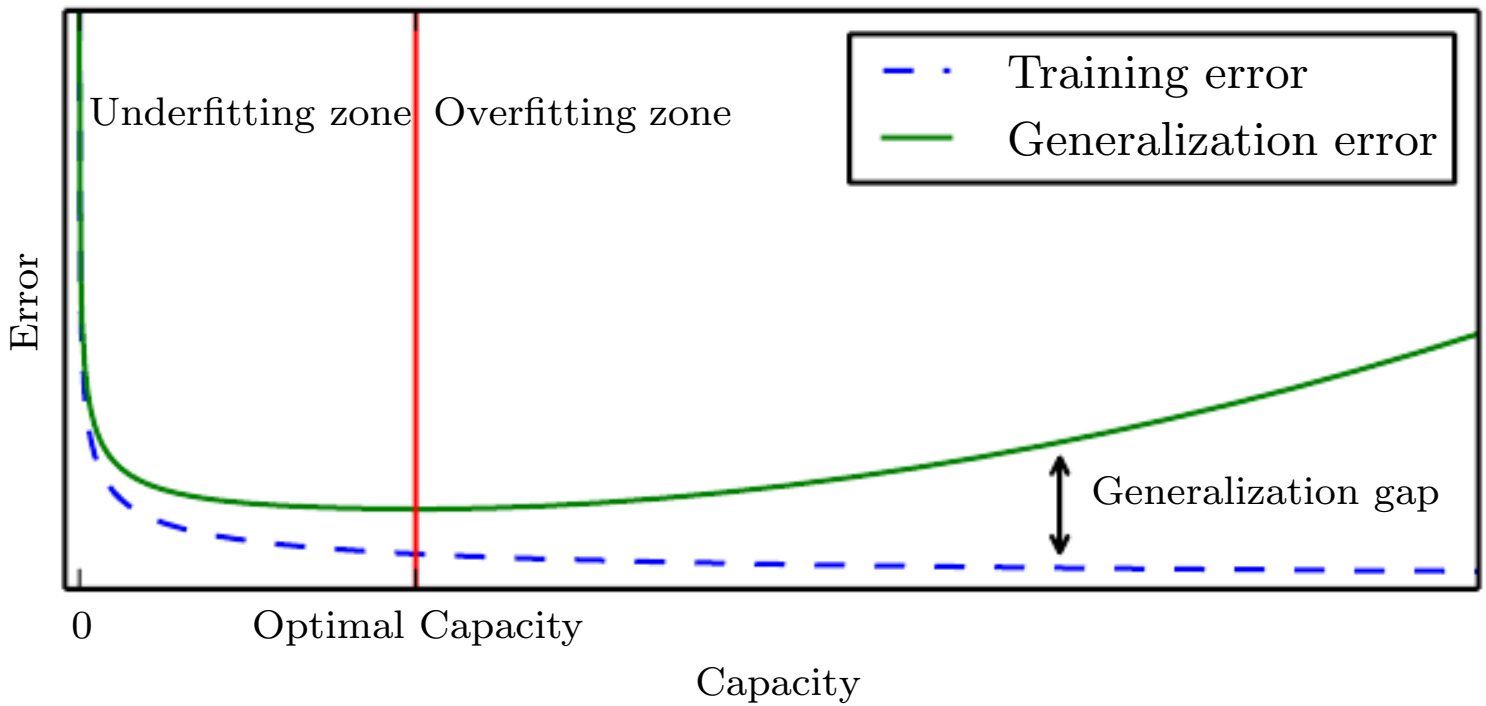
Changing a model's **capacity** controls whether it is more likely to overfit or underfit

- Choose a model's hypothesis space: e.g. increase # of features (adding parameters)
- Find the best among a family of hypothesis functions

Model Capacity

Changing a model's **capacity** controls whether it is more likely to overfit or underfit

- Choose a model's hypothesis space: e.g. increase # of features (adding parameters)
- Find the best among a family of hypothesis functions



How to formalize this idea?

Bias & Variance

$\hat{h}(x)$: estimated hypothesis function of a model. $h(x)$: true hypothesis function

Bias & Variance

$\hat{h}(x)$: estimated hypothesis function of a model. $h(x)$: true hypothesis function

Bias of a model: the expected generalization error if we were to fit it to an infinitely large training set.

$$Bias(\hat{h}) = \mathbb{E}[\hat{h}(x) - h(x)] = \mathbb{E}[\hat{h}(x) - y]$$

- When we make wrong assumptions in the model, such as too few paramters, it has large bias (underfit)

Bias & Variance

$\hat{h}(x)$: estimated hypothesis function of a model. $h(x)$: true hypothesis function

Bias of a model: the expected generalization error if we were to fit it to an infinitely large training set.

$$Bias(\hat{h}) = \mathbb{E}[\hat{h}(x) - h(x)] = \mathbb{E}[\hat{h}(x) - y]$$

- When we make wrong assumptions in the model, such as too few paramters, it has large bias (underfit)

Variance of a model:

$$Var(\hat{h}) = \mathbb{E}[\hat{h}(x)^2] - \mathbb{E}[\hat{h}(x)]^2$$

- When the model overfits "spurious" patterns, it has large variance (overfit).

Bias - Variance Tradeoff

If we measure generalization error by MSE

$$MSE = \mathbb{E}[(\hat{h}(x) - h(x))^2] = Bias(\hat{h})^2 + Var(\hat{h}) + \sigma^2,$$

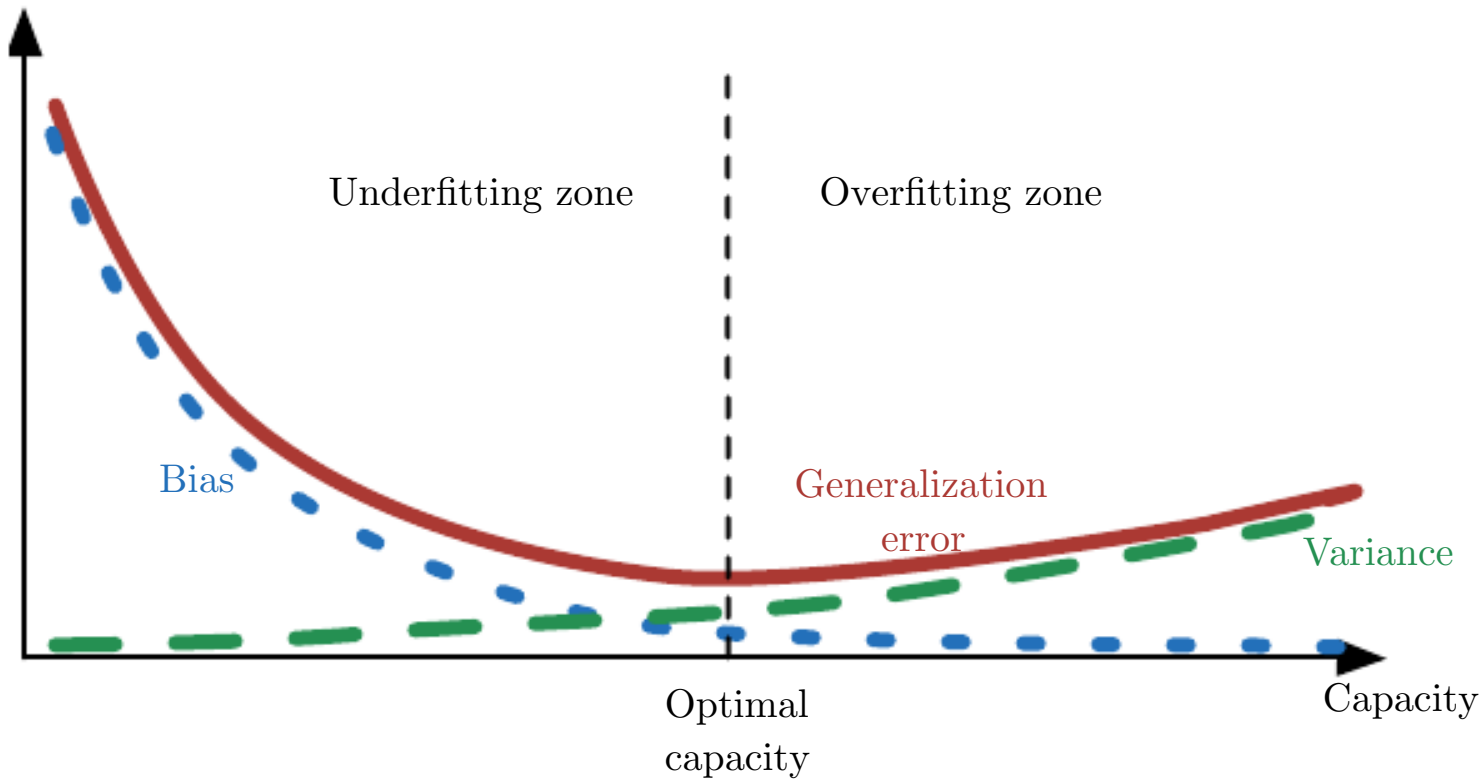
- σ^2 represents irreducible error
- in practice, increasing capacity tends to increase variance and decrease bias.

Bias - Variance Tradeoff

If we measure generalization error by MSE

$$MSE = \mathbb{E}[(\hat{h}(x) - h(x))^2] = Bias(\hat{h})^2 + Var(\hat{h}) + \sigma^2,$$

- σ^2 represents irreducible error
- in practice, increasing capacity tends to increase variance and decrease bias.



Regularization

How to reduce generalization error?

- L-p norm penalty : $\Omega(\theta) = \frac{1}{2}||\theta||_q^q$ $\leftarrow$ *reduces model complexity*
- Bayesian estimation of parameters, MAP estimation $\leftarrow$ *model prior distribution of parameters*
- Neural network regularization methods: e.g.
  - drop out
  - batch normalization

  special layers that reduce model complexity

Model Selection

For a given task, how do we select which model to use?

- Different learning models
  - e.g. SVM vs. logistic regression for binary classification
- Same learning models with different **hyperparameters**
  - e.g. # of clusters in k-means clustering

Model Selection

For a given task, how do we select which model to use?

- Different learning models
  - e.g. SVM vs. logistic regression for binary classification
- Same learning models with different **hyperparameters**
  - e.g. # of clusters in k-means clustering

Cross validation is a class of methods for selecting models using a *validation set*.

Hold-out cross validation

Given training set S and candidate models $M_1,\ldots,M_n$:

1. Randomly split S into S_{train} and S_{cv} (e.g. 70% S_{train})
2. Training each M_i on S_{train},
3. Select the model with smallest empirical error on S_{cv}

Hold-out cross validation

Given training set S and candidate models $M_1,\ldots,M_n$:

1. Randomly split S into S_{train} and S_{cv} (e.g. 70% S_{train})
2. Training each M_i on S_{train},
3. Select the model with smallest empirical error on S_{cv}

Disavantages of hold-out cross validation

- "wastes" about 30% data
- chances of an unfortunate split

K-Fold Cross Validation

Goal: ensure each sample is equally likely to be selected for validation.

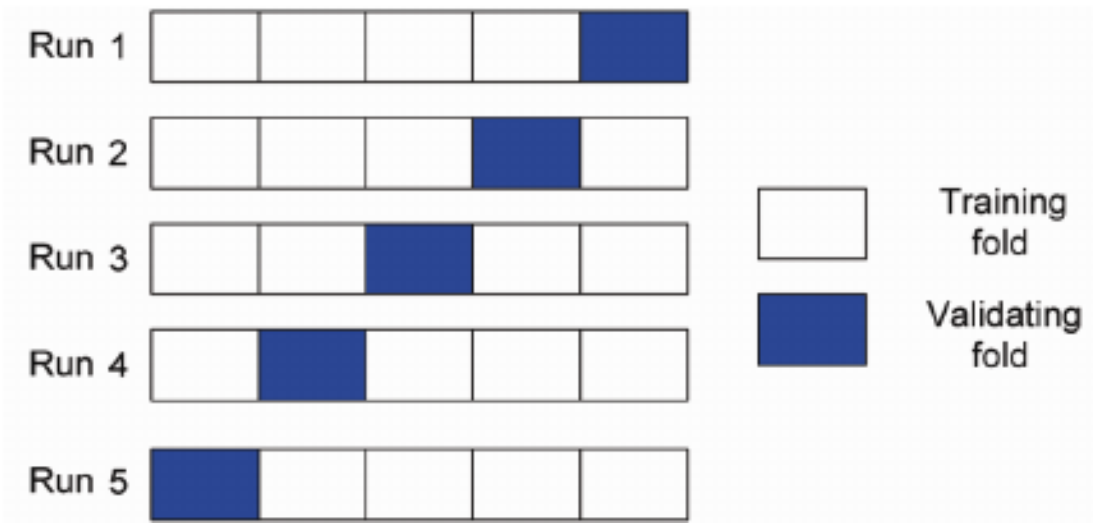
1. Randomly split S into k disjoint subsets $S_1, \ldots, S_k$ of m/k training examples (usually $k = 10$)

K-Fold Cross Validation

Goal: ensure each sample is equally likely to be selected for validation.

1. Randomly split S into k disjoint subsets $S_1, \ldots, S_k$ of m/k training examples (usually $k = 10$)
2. For $j = 1 \ldots k$:

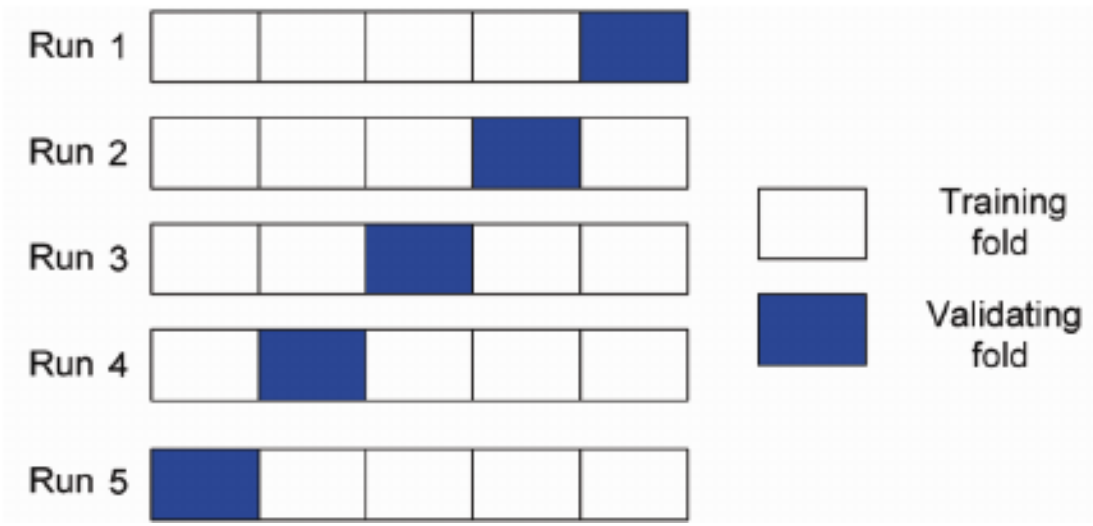
   Train each model on $S \backslash S_j$, then validate on S_j,



K-Fold Cross Validation

Goal: ensure each sample is equally likely to be selected for validation.

1. Randomly split S into k disjoint subsets $S_1, \ldots, S_k$ of m/k training examples (usually $k = 10$)
2. For $j = 1 \ldots k$:
   Train each model on $S \backslash S_j$, then validate on S_j,



3. Select the model with the smallest **average** empirical error among all k trails.

Leave-One-Out Cross Validation

A special case of k-fold cross validation, when $k = m$.

1. For each training example x_i
   Train each model on $S \backslash \{x_i\}$, then evaluate on x_i,
2. Select the model with the smallest average empirical error among all m trails.

Often used when training data is scarce.

Other Cross Validation Methods

- Random subsampling
- Bootstrapping: sample with replacement from training examples (used for small training set)
- Information criteria based methods: e.g. Bayesian information criterion (BIC), Akaike information criterion (AIC)

Other Cross Validation Methods

- Random subsampling
- Bootstrapping: sample with replacement from training examples (used for small training set)
- Information criteria based methods: e.g. Bayesian information criterion (BIC), Akaike information criterion (AIC)

Cross validation can also be used to evaluate a single model.

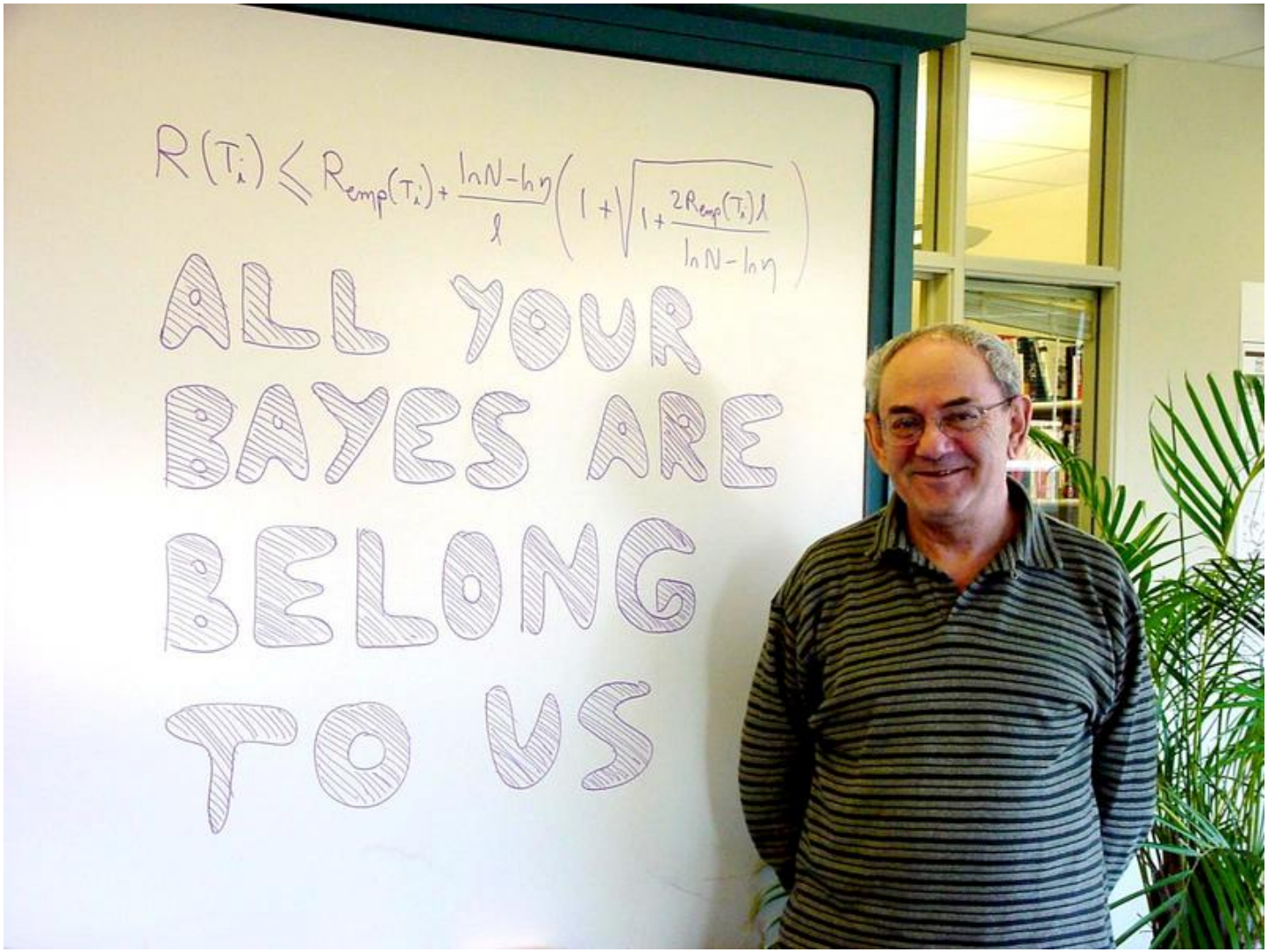
Learning Theory Introduction

Introduction to Learning Theory

- Empirical risk estimation
- Learning bounds
  - Finite Hypothesis Class
  - Infinite Hypothesis Class

Learning theory

How to quantify generalization error?



Prof. Vladimir Vapnik in front of his famous theorem

Empirical Risk Estimation

Preliminaries

Lemma 1 (Union Bound)

Let $A_1, A_2, \ldots, A_k$ be k different events, then

$$P(A_1 \cup \ldots \cup A_k) \leq P(A_1) + \ldots + P(A_k)$$

Probability of any one of k events happening is less the sums of their probabilities.

Preliminaries

Lemma 2 (Hoeffding Inequality, Chernoff bound)

Let $Z_1, \ldots, Z_m$ be m i.i.d. random variables drawn from a Bernoulli(ϕ) distribution. i.e. $P(Z_i = 1) = \phi$, $P(Z_i = 0) = 1 - \phi$.
Let $\hat{\phi} = \frac{1}{m} \sum_{i=1}^{m} Z_i$ be the sample mean of RVs.
For any $\gamma > 0$,

$$P(|\phi - \hat{\phi}| > \gamma) \leq 2 \exp(-2\gamma^2 m)$$

The probability of $\hat{\phi}$ having large estimation error is small when m is large!

Empirical risk

Simplified assumption: $y \in (0, 1)$

- Training set: $S = (x^{(i)}, y^{(i)}); i = 1, \ldots, m$ with $(x^{(i)}, y^{(i)}) \sim \mathcal{D}$
- For hypothesis h, the **training error** or **empirical risk/error** in learning theory is defined as

$$\hat{\epsilon}(h) = \frac{1}{m} \sum_{i=1}^{m} 1\{h(x^{(i)}) \neq y^{(i)}\}$$

- The **generalization error** is

$$\epsilon(h) = P_{(x,y)\sim\mathcal{D}}(h(x) \neq y)$$

- **PAC assumption**: assume that training data and test data (for evaluating generalization error) were drawn from the same distribution $\mathcal{D}$

Hypothesis Class and ERM

Hypothesis class

The **hypothesis class $\mathcal{H}$ used by a learning algorithm** is the set of all classsifiers considered by it.
e.g. Linear classification considers $h_\theta(x) = 1\{\theta^T x \geq 0\}$

Empirical Risk Minimization (ERM): the "simplest" learning algorithm: pick the best hypothesis h from hypothesis class $\mathcal{H}$

$$\hat{h} = \underset{h \in \mathcal{H}}{\operatorname{argmin}}\, \hat{\epsilon}(h)$$

How to measure the generalization error of empirical risk minimization over $\mathcal{H}$?

- Case of finite $\mathcal{H}$
- Case of infinite $\mathcal{H}$

Uniform Convergence and Sample Complexity

- Case of Finite H
- Infinite H

Case of Finite $\mathcal{H}$

Goal: give guarantee on generalization error $\epsilon(h)$

- Show $\hat{\epsilon}(h)$ (training error) is a good estimate of $\epsilon(h)$
- Derive an upper bound on $\epsilon(h)$

For any $h_i \in \mathcal{H}$, the event of h_i miss-classification given sample $(x, y) \sim \mathcal{D}$:

$$Z = 1\{h_i(x) \neq y\}$$

$Z_j = 1\{h_i(x^{(j)}) \neq y^{(j)}\}$: event of h_i miss-classifying sample $x^{(j)}$

Case of Finite $\mathcal{H}$

Goal: give guarantee on generalization error $\epsilon(h)$

- Show $\hat{\epsilon}(h)$ (training error) is a good estimate of $\epsilon(h)$
- Derive an upper bound on $\epsilon(h)$

For any $h_i \in \mathcal{H}$, the event of h_i miss-classification given sample $(x, y) \sim \mathcal{D}$:

$$Z = 1\{h_i(x) \neq y\}$$

$Z_j = 1\{h_i(x^{(j)}) \neq y^{(j)}\}$: event of h_i miss-classifying sample $x^{(j)}$

Training error of $h_i \in \mathcal{H}$ is:

$$\hat{\epsilon}(h_i) = \frac{1}{m} \sum_{j=1}^{m} 1\{h_i(x^{(j)}) \neq y^{(j)}\}$$

$$\hat{\epsilon}(h_i) = \frac{1}{m} \sum_{j=1}^{m} Z_j$$

Case of Finite $\mathcal{H}$

Training error of $h_i \in \mathcal{H}$ is:

$$\hat{\epsilon}(h_i) = \frac{1}{m} \sum_{j=1}^{m} Z_j$$

where $Z_j \sim Bernoulli(\epsilon(h_i))$

By Hoeffding inequality,

$$P(|\epsilon(h_i) - \hat{\epsilon}(h_i)| > \gamma) \leq 2e^{-2\gamma^2 m}$$

By Union bound,

$$P(\forall h \in \mathcal{H}.|\epsilon(h) - \hat{\epsilon}(h)| \leq \gamma) \geq 1 - 2ke^{-2\gamma^2 m}$$

Uniform Convergence Results

Corollary 3

Given γ *and* $\delta > 0$, *If*

$$m \geq \frac{1}{2\gamma^2} \log \frac{2k}{\delta}$$

Then with probability at least $1 - \delta$, *we have* $|\epsilon(h) - \hat{\epsilon}(h)| \leq \gamma$ *for all* $\mathcal{H}$.
m *is called the algorithm's* **sample complexity**.

Remarks

- Lower bound on m tell us how many training examples we need to make generalization guarantee.
- # of training examples needed is logarithm in k

Uniform Convergence Results

Corollary 4

With probability $1-\delta$, *for all* $h \in \mathcal{H}$,

$$|\hat{\epsilon}(h) - \epsilon(h)| \leq \sqrt{\frac{1}{2m} \log \frac{2k}{\delta}}$$

What is the convergence result when we pick $\hat{h} = \text{argmin}_{h \in \mathcal{H}} \hat{\epsilon}(h)$

Theorem 5 (Uniform convergence)

Let $|\mathcal{H}| = k$, *and* m,δ *be fixed. With probability at least* $1-\delta$, *we have*

$$\epsilon(\hat{h}) \leq \left(\min_{h \in \mathcal{H}} \epsilon(h) \right) + 2\sqrt{\frac{1}{2m} \log \frac{2k}{\delta}}$$

Infinite hypothesis class: Challenges

Can we apply the same theorem to infinite $\mathcal{H}$?

Example

- Suppose $\mathcal{H}$ is parameterized by d real numbers. e.g. $\theta = [\theta_1, \theta_2, \ldots, \theta_d] \in \mathbb{R}^d$ in linear regression with $d-1$ unknowns.
- In a 64-bit floating point representation, size of hypothesis class: $|\mathcal{H}| = 2^{64d}$
- How many samples do we need to guarantee $\epsilon(\hat{h}) \leq \epsilon(h^*) + 2\gamma$ to hold with probability at least $1-\delta$?

$$m \geq O\left(\frac{1}{\gamma^2} \log \frac{2^{64d}}{\delta}\right) = O\left(\frac{d}{\gamma^2} \log \frac{1}{\delta}\right) = O_{\gamma,\delta}(d)$$

To learn **well**, *the number of samples has to be linear in d*

Infinite hypothesis class: Challenges

Size of $\mathcal{H}$ depends on the choice of parameterization

Example

$2n + 2$ parameters:

$$h_{u,v} = \mathbf{1}\{(u_0^2 - v_0^2) + (u_1^2 - v_1^2)x_1 + \ldots + (u_n^2 - v_n^2)x_n \geq 0\}$$

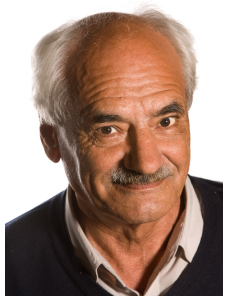
is equivalent the hypothesis with $n + 1$ parameters:

$$h_\theta(x) = \mathbf{1}\{\theta_0 + \theta_1 x_1 + \ldots + \theta_n x_n \geq 0\}$$

We need a complexity measure of a hypothesis class invariant to parameterization choice

Infinite hypothesis class: VapnikChervonenkis theory

A computational learning theory developed during 1960-1990 explaining the learning process from a statistical point of view.



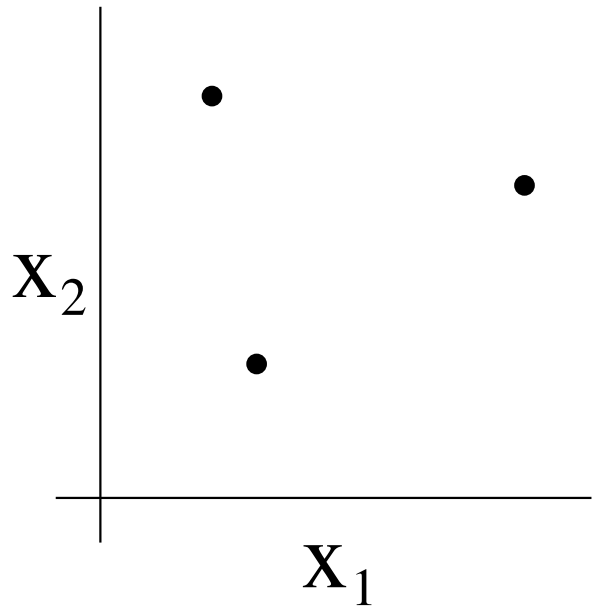
Alexey Chervonenkis (1938-2014), Russian mathematician

Vladimir Vapnik (Facebook AI Research, Vencore Labs)
Most known for his contribution in statistical learning theory

Shattering a point set

- Given d points $x^{(i)} \in \mathcal{X}$, $i = 1, \ldots, d$, $\mathcal{H}$ **shatters** S if $\mathcal{H}$ can realize any labeling on S.

Example: $S = \{x^{(1)}, x^{(2)}, x^{(3)}\}$ where $x^{(i)} \in \mathbb{R}^2$.

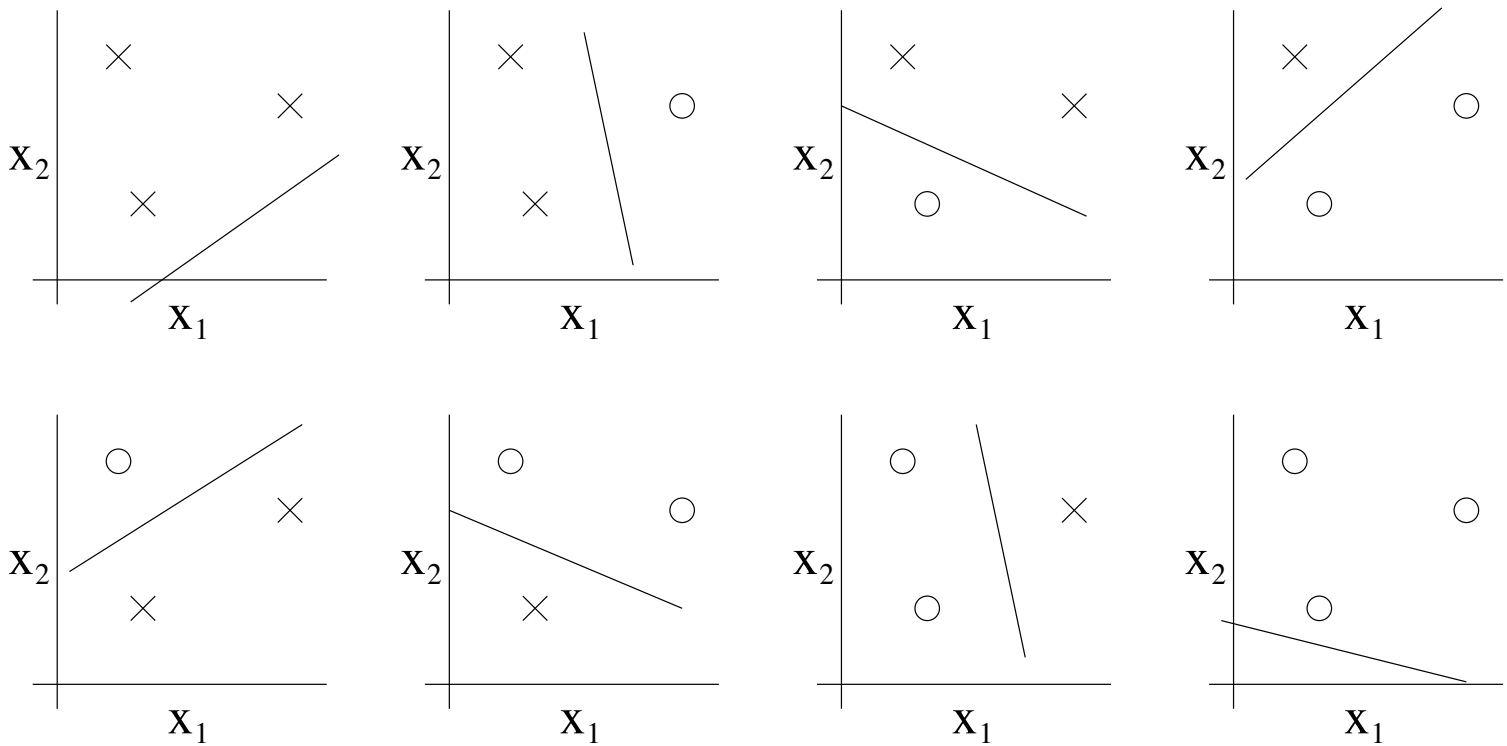


Suppose $y^{(i)} \in \{0, 1\}$, how many possible labelings does S have?

Shattering a point set

- Example: Let $\mathcal{H}_{LTF,2}$ be the linear threshold function in $\mathbb{R}^2$ (e.g. in the perceptron algorithm)

$$h(x) = \begin{cases} 1 & w_1 x_1 + w_2 x_2 \geq b \\ 0 & \text{otherwise} \end{cases}$$

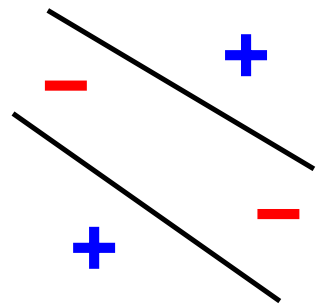


$\mathcal{H}_{LTF,2}$ shatters $S = \{x^{(1)}, x^{(2)}, x^{(3)}\}$

VC Dimension

The **Vapnik-Chervonenkis** dimension of $\mathcal{H}$, or $VC(\mathcal{H})$, is the cardinality of the largest set shattered by $\mathcal{H}$.

- Example: $VC(H_{LTF,2}) = 3$

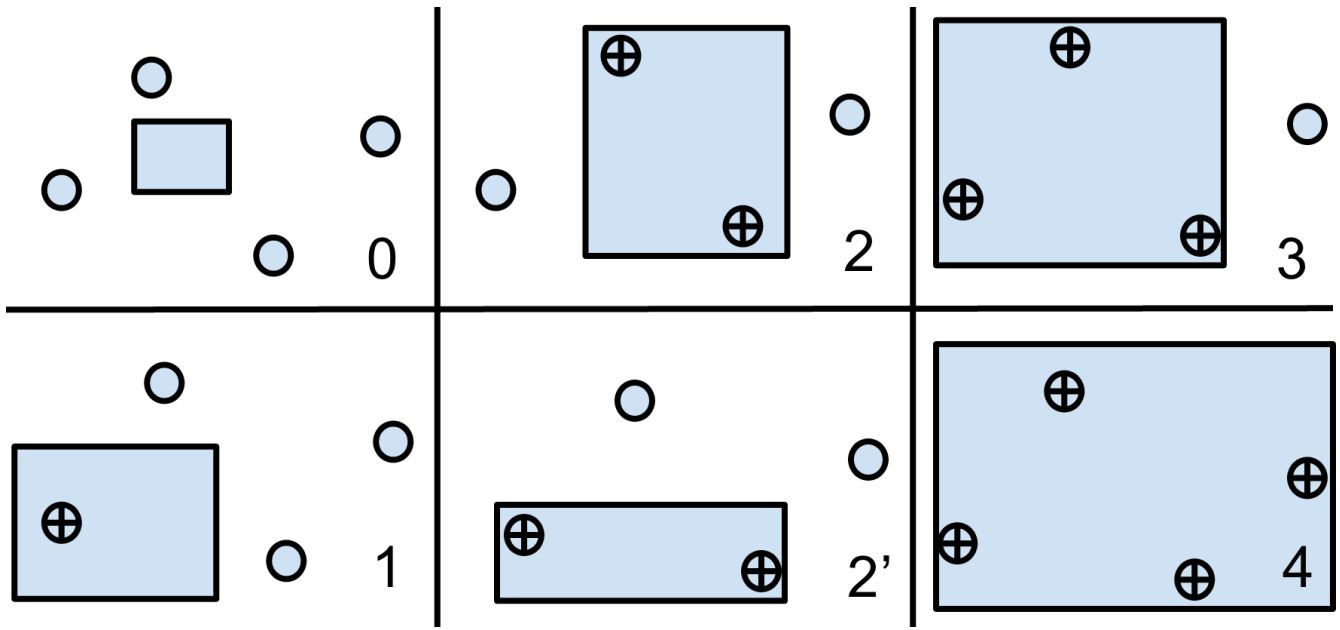


$\mathcal{H}_{LTF}$ can not shatter 4 points: for any 4 points, label points on the diagonal as '+'. (See Radon's theorem)

- To show $VC(\mathcal{H}) \geq d$, it's sufficient to find **one** set of d points shattered by $\mathcal{H}$
- To show $VC(\mathcal{H}) < d$, need to prove $\mathcal{H}$ doesn't shatter any set of d points

VC Dimension

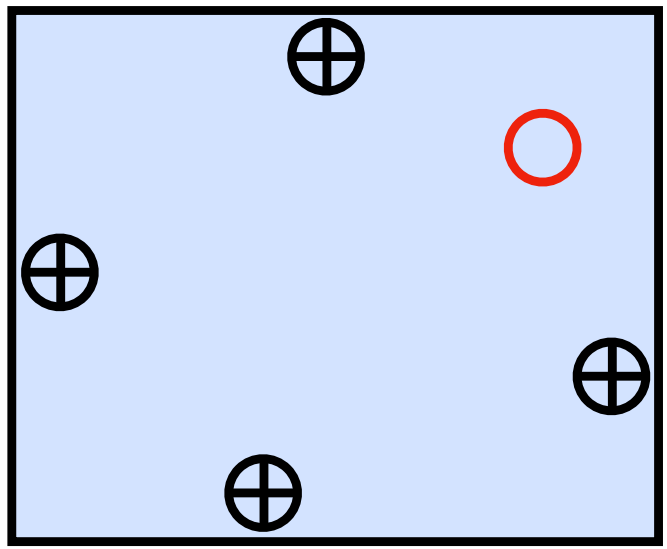
- Example: $VC(AxisAlignedRectangles) = 4$



Axis-aligned rectangles can shatter 4 points.
$VC(AxisAlignedRectangles) \geq 4$

VC Dimension

- Example: $VC(AxisAlignedRectangles) = 4$



For any 5 points, label topmost, bottommost, leftmost and rightmost points as "+".

$VC(AxisAlignedRectangles) < 5$

Discussion on VC Dimension

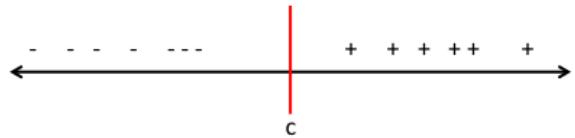
More VC results of common $\mathcal{H}$:

- $VC(ConstantFunctions) =$

Discussion on VC Dimension

More VC results of common $\mathcal{H}$:

- $VC(\textit{ConstantFunctions}) = 0$
- $VC(\textit{PositiveHalf-Lines}) = 1, \mathcal{X} = \mathbb{R}$

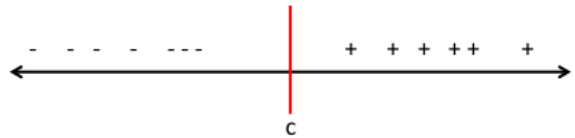


- $VC(\textit{Intervals}) = 2, \mathcal{X} = \mathbb{R}$
- $VC(\textit{LTF in } \mathbb{R}^n) = n + 1, \mathcal{X} = \mathbb{R}^n \leftarrow$ *prove this at home!*

Discussion on VC Dimension

More VC results of common $\mathcal{H}$:

- $VC(ConstantFunctions) = 0$
- $VC(PositiveHalf\text{-}Lines) = 1, \mathcal{X} = \mathbb{R}$



- $VC(Intervals) = 2, \mathcal{X} = \mathbb{R}$
- $VC(LTF\ in\ \mathbb{R}^n) = n + 1, \mathcal{X} = \mathbb{R}^n \leftarrow$ *prove this at home!*

Proposition 1

If $\mathcal{H}$ is finite, VC dimension is related to the cardinality of $\mathcal{H}$:

$$VC(\mathcal{H}) \leq log|\mathcal{H}|$$

Proof. Let $d = VC|\mathcal{H}|$. There must exists a shattered set of size d on which H realizes all possible labelings. Every labeling must have a corresponding hypothesis, then $|\mathcal{H}| \geq 2^d$

Learning bound for infinite $\mathcal{H}$

Theorem 6

Given $\mathcal{H}$, let $d = VC(\mathcal{H})$.

- *With probability at least $1 - \delta$, we have that for all h*

$$|\epsilon(h) - \hat{\epsilon}(h)| \leq O\left(\sqrt{\frac{d}{m}\log\frac{m}{d} + \frac{1}{m}\log\frac{1}{\delta}}\right)$$

- *Thus, with probability at least $1 - \delta$, we also have*

$$\epsilon(\hat{h}) \leq \epsilon(h^*) + O\left(\sqrt{\frac{d}{m}\log\frac{m}{d} + \frac{1}{m}\log\frac{1}{\delta}}\right)$$

Learning bound for infinite $\mathcal{H}$

Corollary 7

For $|\epsilon(h) - \hat{\epsilon}(h)| \leq \gamma$ *to hold for all* $h \in \mathcal{H}$ *with probability at least* $1 - \delta$*, it suffices that* $m = O_{\gamma,\delta}(d)$.

Remarks

- Sample complexity using $\mathcal{H}$ is linear in $VC(\mathcal{H})$
- For "most"[a] hypothesis classes, the VC dimension is linear in terms of parameters
- For algorithms minimizing training error, # training examples needed is roughly linear in number of parameters in $\mathcal{H}$.

[a] Not always true for deep neural networks

VC Dimension of Deep Neural Networks

Theorem 8 (Cover, 1968; Baum and Haussler, 1989)

Let $\mathcal{N}$ be an arbitrary feedforward neural net with w weights that consists of linear threshold activations, then $VC(\mathcal{N}) = O(w \log w)$.

Recent progress

- For feed-forward neural networks with piecewise-linear activation functions (e.g. ReLU), let w be the number of parameters and l be the number of layers, $VC(\mathcal{N}) = O(wl \log(w))$ [Bartlett et. al., 2017]
- *Among all networks with the same size (number of weights), more layers have larger VC dimension* , thus more training samples are needed to learn a deeper network

Bartlett and W. Maass (2003) Vapnik-Chervonenkis Dimension of Neural Nets
Bartlett et. al., (2017) Nearly-tight VC-dimension and pseudodimension bounds for piecewise linear neural networks.