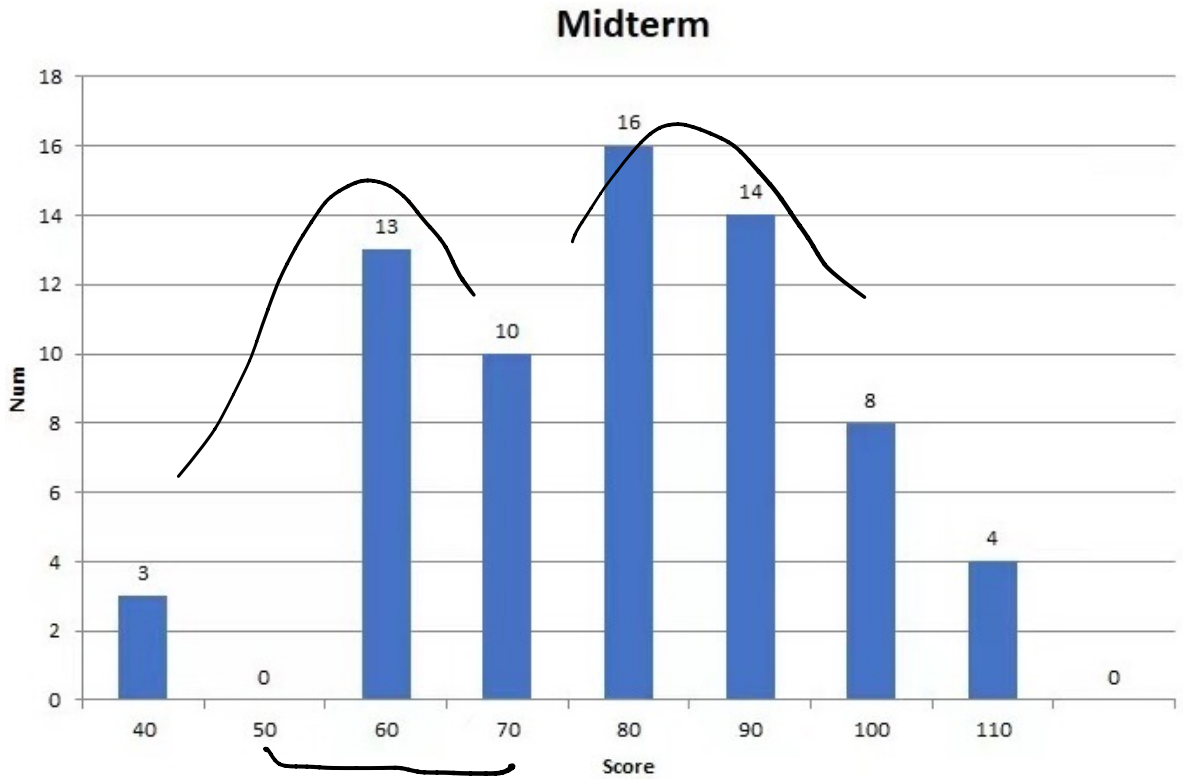
Learning From Data
Lecture 7: Model Selection, Regularization & Learning Theory

Yang Li **yangli@sz.tsinghua.edu.cn**

TBSI

November 12, 2021

Midterm Results



	max	mean	median
raw score	109	57.9	56.5
curved score	110	74.5	74

Introduction

Today's Lecture

Practical tools to improve machine learning performance:

- Model selection: bias and variance trade off, cross-validation
- Regularization

A brief introduction to learning theory

- Empirical risk estimation
- Generalization bound for finite and infinite hypothesis space

Model selection

Empirical error & Generalization error

$1\{h(x^{(i)}) \neq y^{(i)}\} = \begin{cases} 1 & \text{if } \neq \\ 0 & \text{if } = \end{cases}$

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.

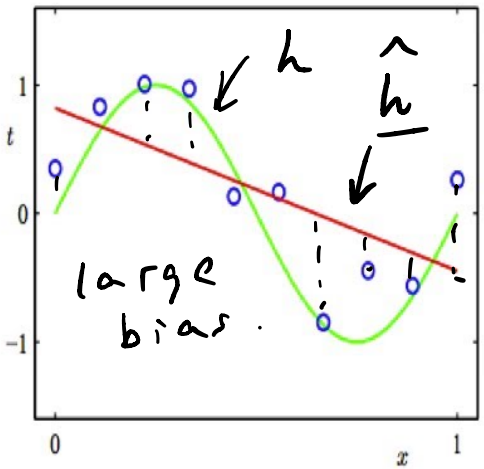
Goal of machine learning

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

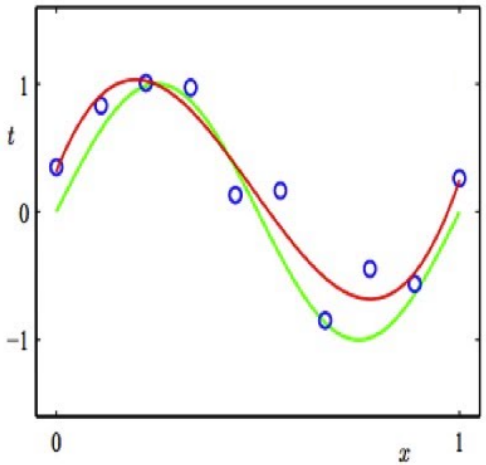
Overfit & Underfit

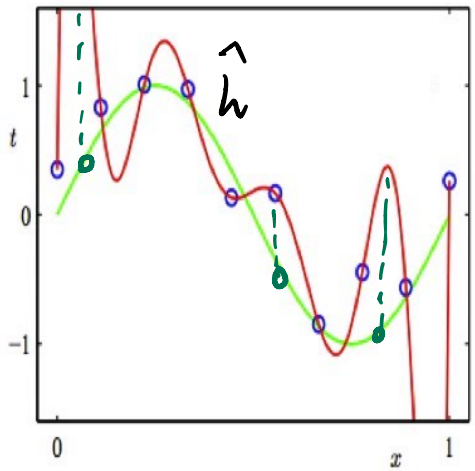
Underfit Both training error and testing error are large

Overfit Training error is small, testing error is large



underfit



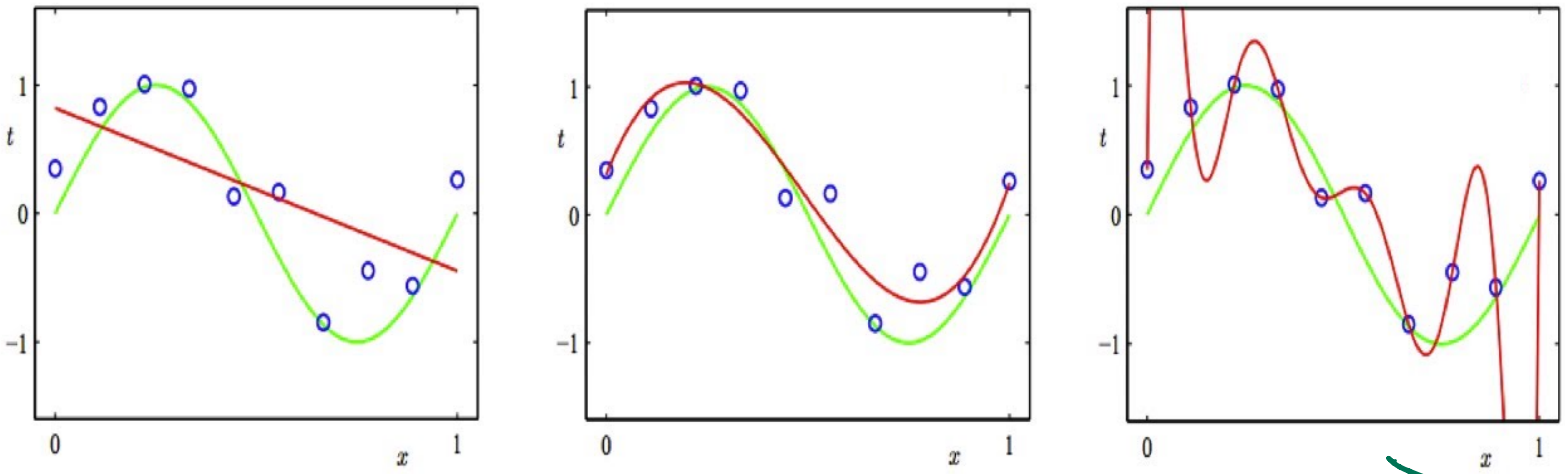


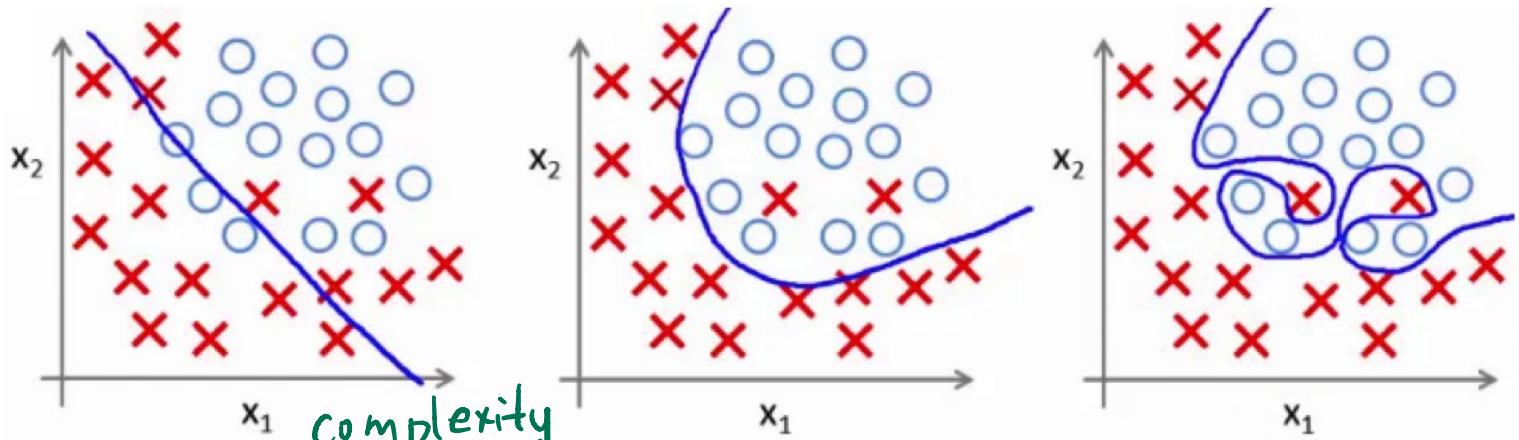
overfit

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

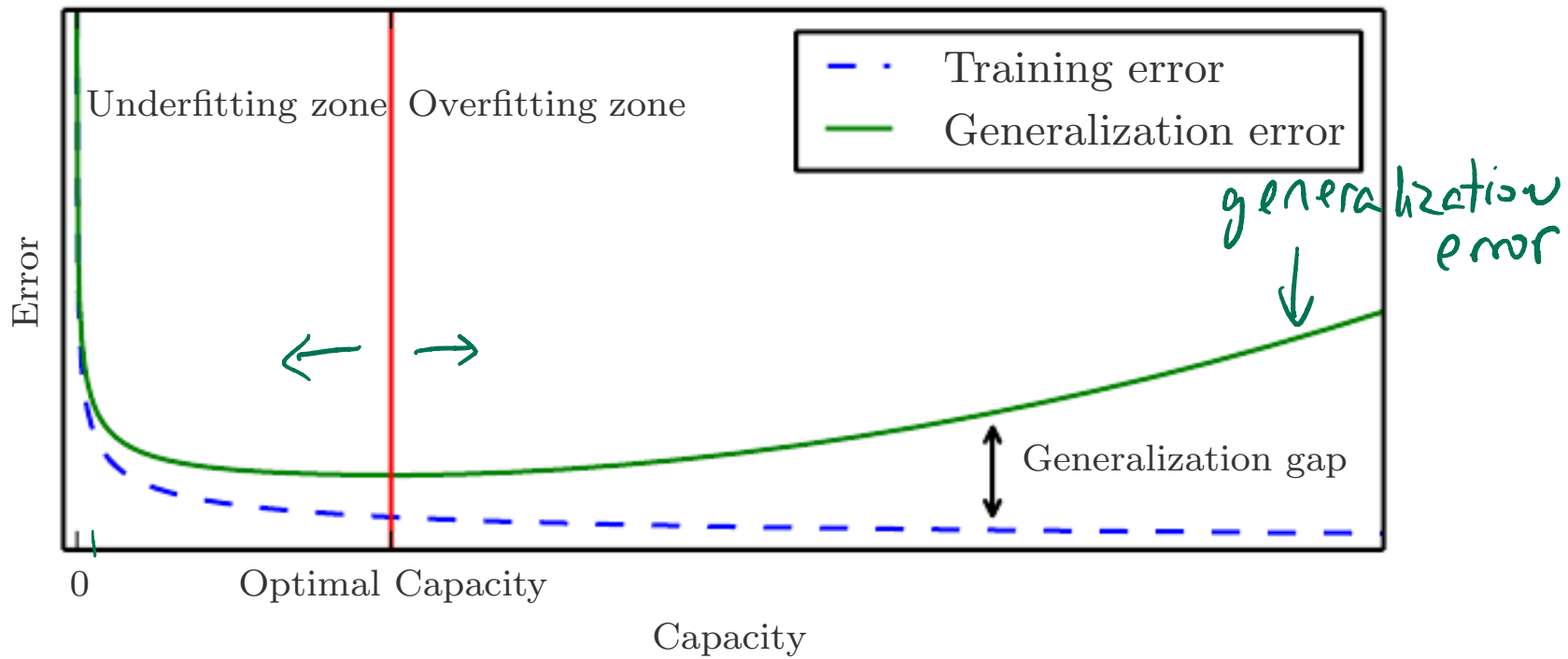
Changing a model's **capacity** controls whether it is more likely to overfit or underfit

- 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 and Variance

Suppose data is generated by the following model:

$$y = h(x) + \epsilon$$

→ P_{xy}: true joint distribution of (x, y)

with $\mathbb{E}[\epsilon] = 0$, $Var(\epsilon) = \sigma^2$

$h(x)$ true hypothesis function, unknown → *fixed value*

$\hat{h}_D(x)$ estimated hypothesis function based on training data

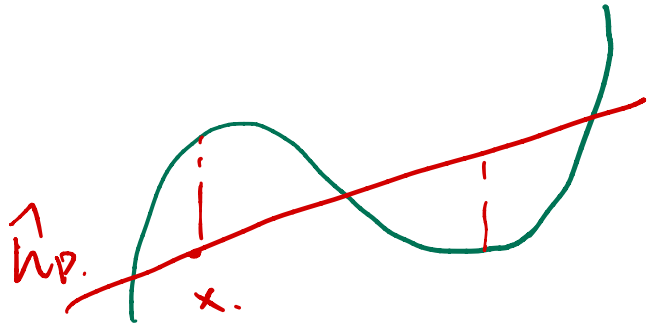
$D = \{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\}$ → *a random variable*

$D \sim P_{xy}$.

Bias and Variance

Suppose data is generated by the following model:

$$y = h(x) + \epsilon$$



with $\mathbb{E}[\epsilon] = 0$, $Var(\epsilon) = \sigma^2$

$h(x)$ true hypothesis function, unknown $\rightarrow$ *fixed value*

$\hat{h}_D(x)$ estimated hypothesis function based on training data $D = \{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\}$ $\rightarrow$ *a random variable*

Bias of a model

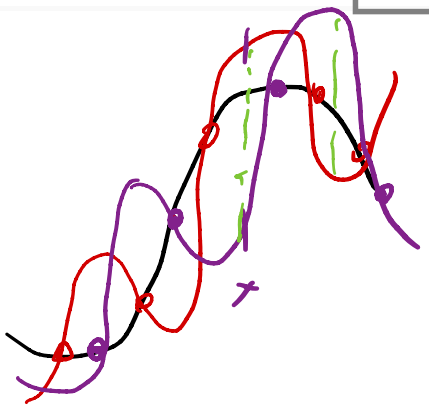
The expected estimation error of $\hat{h}_D$ over all choices of training data D sampled from P_{XY}

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

When we make wrong assumptions about the model, such as too few parameters, $\hat{h}_D$ will have large bias (underfit)

Bias and Variance

$$Var(z) = \mathbb{E}[(z - \mathbb{E}(z))^2]$$
$$= \mathbb{E}[z^2] - (\mathbb{E}[z])^2$$



Variance of a model

The variance of the model learned from different choices of training data

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

- When the model varies a lot with the choice of training data, it has large variance (overfit).

Bias - Variance Tradeoff

If we measure generalization error by MSE

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

(D — error of $\hat{h}_D$.)

- σ^2 represents irreducible error *(caused by noisy data)*
- in practice, increasing capacity tends to increase variance and decrease bias.

[Fact 1] $Var(y) = \mathbb{E}[(y - \mathbb{E}(y))^2]$
$= \mathbb{E}[(h(x)+\varepsilon - \mathbb{E}[h(x)+\varepsilon])^2]$
$= \mathbb{E}[(h(x)+\varepsilon - \mathbb{E}[h(x)] - \mathbb{E}[\varepsilon])^2]$
($\mathbb{E}[h(x)] = h(x)$, $\mathbb{E}[\varepsilon] = 0$)
$= \mathbb{E}[\varepsilon^2] = \sigma^2$

$y = h(x) + \varepsilon$

$Var(z) = \mathbb{E}(z^2) - \mathbb{E}(z)^2$

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

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

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

$= Var[\hat{h}_D(x)] + Var(y) + \underbrace{\left(\mathbb{E}[\hat{h}_D(y)]^2 + \mathbb{E}[y]^2 - 2y\mathbb{E}[\hat{h}_D(x)]\right)}_{\mathbb{E}[(\hat{h}_D(x) - y)]^2}$

(by Fact 1) $Var(y) = \sigma^2$

[Fact 2] $\mathbb{E}[(\hat{h}_D(x) - y)]$
$= \mathbb{E}[(\hat{h}_D(x) - h(x) - \varepsilon)]$
$= \mathbb{E}[\hat{h}_D(x) - h(x)] - \mathbb{E}[\varepsilon]$
($\mathbb{E}[\varepsilon] = 0$)
$= \mathbb{E}[\hat{h}_D(x) - h(x)]$

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

$= Bias(\hat{h}_D(x))$

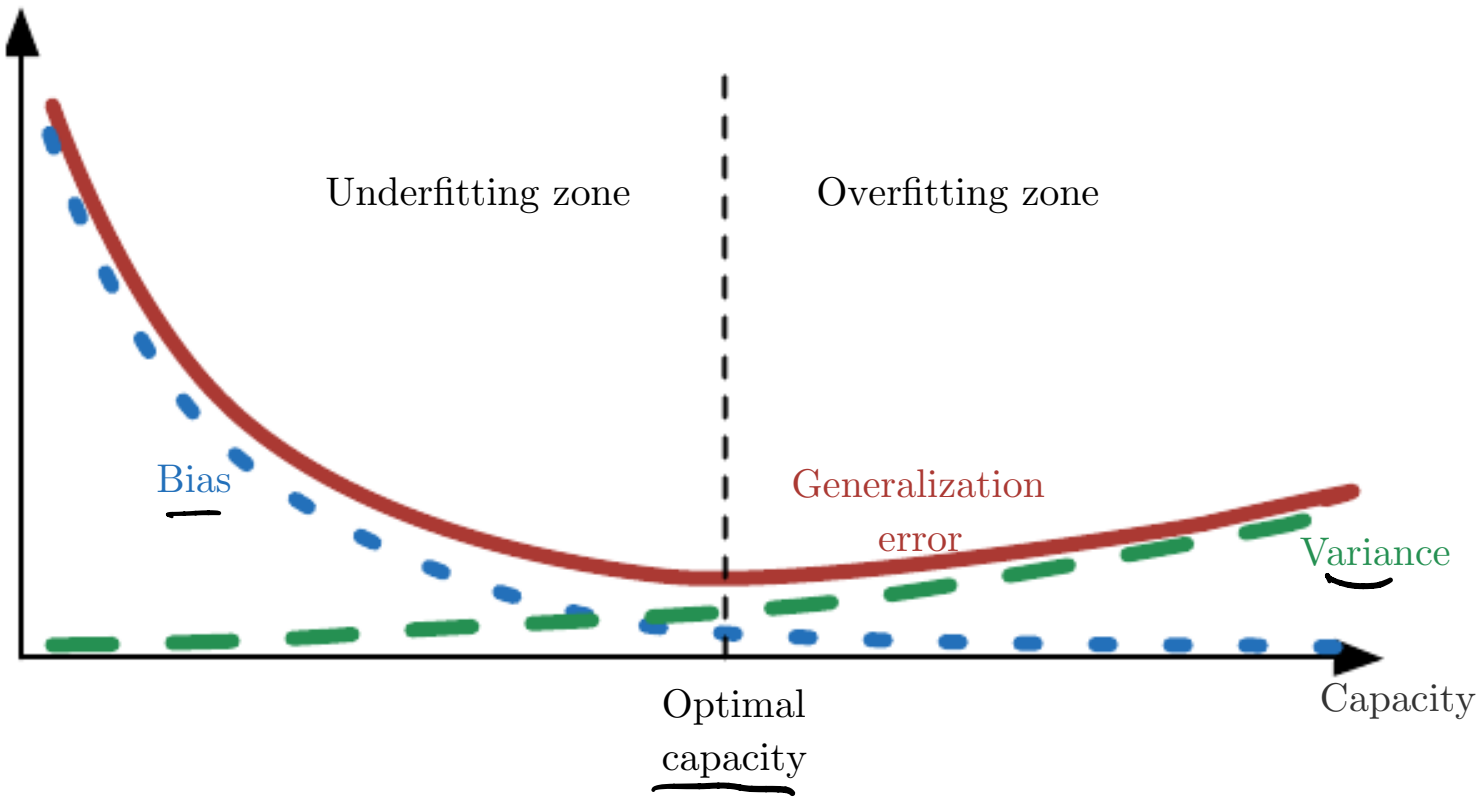
Bias - Variance Tradeoff

$y = h(x) + \varepsilon$. model noise.

If we measure generalization error by MSE

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

- σ^2 represents irreducible error *(caused by noisy data)*
- in practice, increasing capacity tends to increase variance and decrease bias.



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

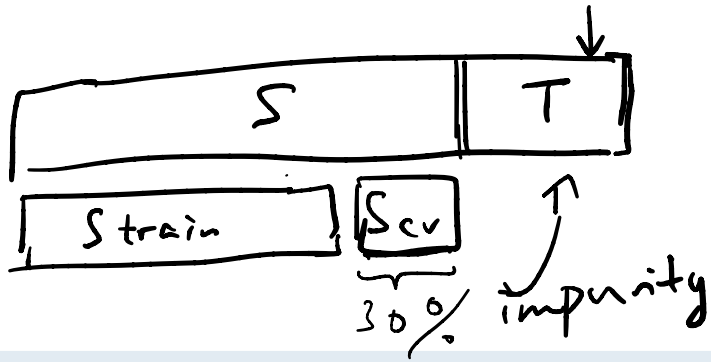
Model Selection

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

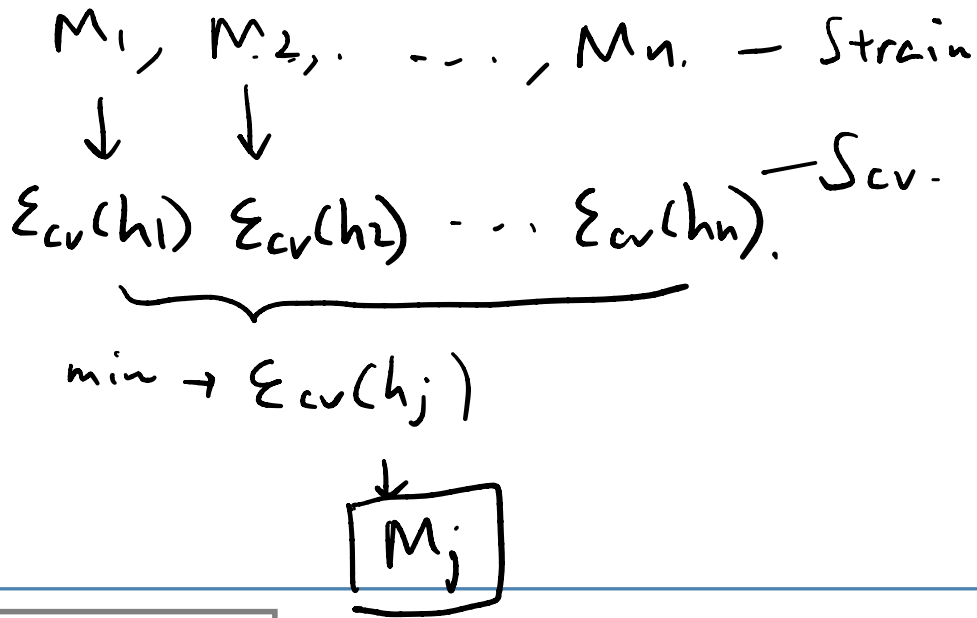
Cross validation is a class of methods for selecting models using a *validation set*.

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

S_{cv}

K-Fold Cross Validation

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

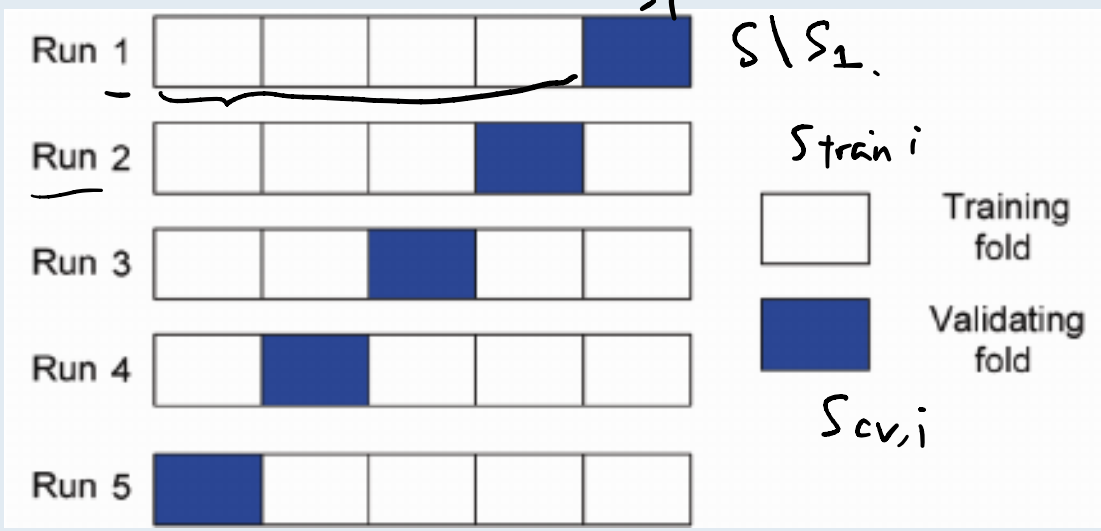
1. Randomly split S into k disjoint subsets $S_1, \ldots, S_k$ of m/k training examples (usually $k = 10$)

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,

$K=5$.

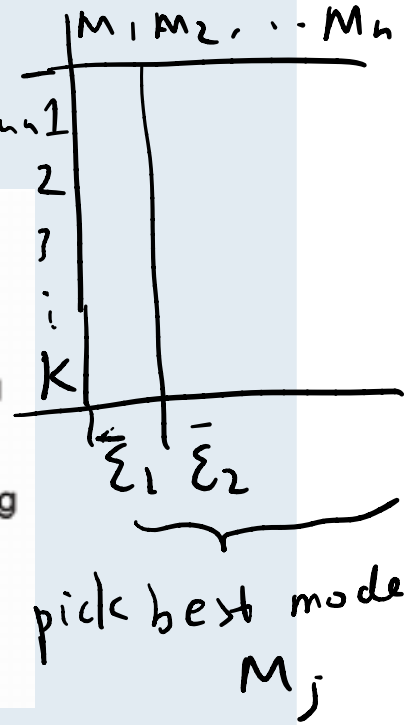


K-Fold Cross Validation

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

evaluate n models.



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 $k = m$. sample size

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 – without replacement.
- 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.

Regularization

- Parameter Norm Penalty
- MAP estimation
- Regularization for neural networks

Regularization

Regularization is any modification we make to a learning algorithm to reduce its generalization error, but not the training error

Regularization

Regularization is any modification we make to a learning algorithm to reduce its generalization error, but not the training error

Common regularization techniques:

- Penalize parameter size
  e.g. linear regression with norm penalty (see PA1& WA1)

$$J(\theta) = \sum_{i=1}^{m} \log p(y^{(i)}|x^{(i)};\theta) + \lambda||\theta||_2^2$$

Regularization

Regularization is any modification we make to a learning algorithm to reduce its generalization error, but not the training error

Common regularization techniques:

- Penalize parameter size
  e.g. linear regression with norm penalty (see PA1& WA1)

$$J(\theta) = \sum_{i=1}^{m} \log p(y^{(i)}|x^{(i)};\theta) + \lambda||\theta||_2^2$$

- Use prior probability: max-a-posteriori estimation

MAP.

Parameter Norm Penalty

Adding a regularization term to the loss (error) function: $\lambda > 0$

$$\tilde{J}(X, Y; \theta) = \underbrace{J(X, Y; \theta)}_{\text{data-dependent loss}} + \lambda \cdot \underbrace{\Omega(\theta)}_{\text{regularizer}}$$

where

$$\Omega(\theta) = \frac{1}{2} \sum_{j=1}^{n} |\theta_j|^q = \frac{1}{2} ||\theta||_q^q$$

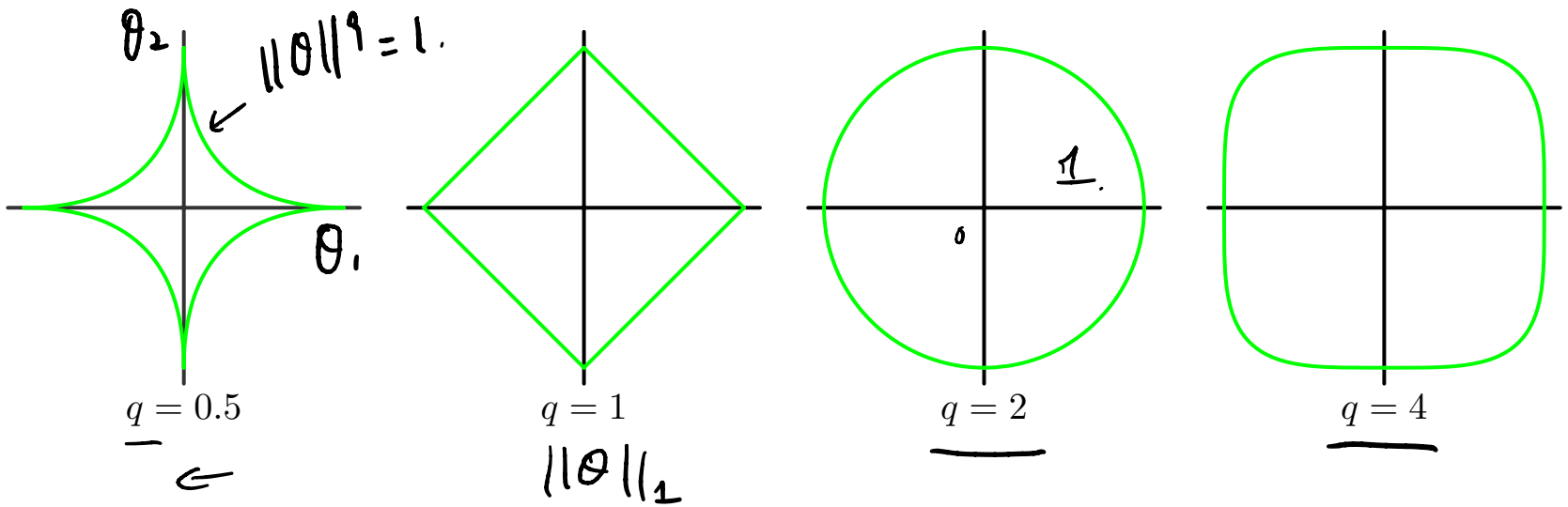
Parameter Norm Penalty

Adding a regularization term to the loss (error) function:

$$\tilde{J}(X, Y; \theta) = \underbrace{J(X, Y; \theta)}_{\text{data-dependent loss}} + \lambda \underbrace{\Omega(\theta)}_{\text{regularizer}}$$

where

$$\Omega(\theta) = \frac{1}{2} \sum_{j=1}^{n} |\theta_j|^q = \frac{1}{2} ||\theta||_q^q$$



Contours of the regularizer ($||\theta||^q = 1$) for different q

L2 parameter penalty

When $q = 2$, it's also known as **Tokhonov regularization** or **ridge regression**

$$\tilde{J}(X, Y; \theta) = J(X, Y; \theta) + \frac{\lambda}{2}\theta^T\theta$$

L2 parameter penalty

When $q = 2$, it's also known as **Tokhonov regularization** or **ridge regression**

$$\tilde{J}(X, Y; \theta) = J(X, Y; \theta) + \frac{\lambda}{2}\theta^T\theta$$

Gradient descent update:

$$\nabla_\theta \left(\frac{\lambda}{2}\theta^T\theta\right) = \lambda\theta$$

$$\begin{aligned}\theta &\leftarrow \theta - \alpha\nabla_\theta \tilde{J}(X, Y; \theta) \\ &= \theta - \alpha(\nabla_\theta J(X, Y; \theta) + \lambda\theta) \\ &= (1 - \alpha\lambda)\theta - \alpha\nabla_\theta J(X, Y; \theta)\end{aligned}$$

L2 penalty multiplicatively shrinks parameter θ by a constant

Also known as **weight decay** *in gradient descent (neural network).*

L2 parameter penalty

When $q = 2$, it's also known as **Tokhonov regularization** or **ridge regression**

$$\tilde{J}(X, Y; \theta) = J(X, Y; \theta) + \frac{\lambda}{2}\theta^T\theta$$

Gradient descent update:

$$\begin{aligned}\theta &\leftarrow \theta - \alpha\nabla_\theta \tilde{J}(X, Y; \theta) \\ &= \theta - \alpha(\nabla_\theta J(X, Y; \theta) + \lambda\theta) \\ &= (1 - \alpha\lambda)\theta - \alpha\nabla_\theta J(X, Y; \theta)\end{aligned}$$

L2 penalty multiplicatively shrinks parameter θ by a constant
Also known as **weight decay** *in gradient descent (neural network).*

Example: regularized least square (WA1)

When $J(X, Y; \theta) = \frac{1}{2}\sum_{i=1}^{m}(y^{(i)} - \theta^T x^{(i)})^2$ (ordinary least squares),
$\tilde{\theta}_{OLS} = (X^TX + \lambda I)^{-1}(X^TY)$

$(X^TX)^{-1}$ underdetermine problem.
decrease generalization error.

L1 parameter penalty

When $q = 1$, $\Omega(\theta) = \frac{1}{2}\sum_{j=1}^{n} |\theta_j|$ is also known as **LASSO regression**.

- If λ is sufficiently large, some coefficients θ_j are driven to 0.
- It will lead to a *sparse* model

L1 parameter penalty

When $q = 1$, $\Omega(\theta) = \frac{1}{2}\sum_{j=1}^{n} |\theta_j|$ is also known as **LASSO regression**.

- If λ is sufficiently large, some coefficients θ_j are driven to 0.
- It will lead to a *sparse* model

Proposition 1

$\lambda \to 0$. $\theta \in D$, D is a convex set.

Assuming $J(\theta)$ is a convex function over some convex set and $\lambda > 0$, solving $\min_\theta \tilde{J}(X, Y; \theta) = J(X, Y; \theta) + \frac{\lambda}{2}\sum_{j=1}^{n} |\theta_j|^q$ is equivalent to

(1) $\Leftrightarrow$ (2)

$$\begin{cases} \min_\theta J(X, Y; \theta) \\ s.t. \sum_{j=1}^{n} |\theta|^q \le \eta \end{cases}$$

regional constraint on $|\theta|^q$

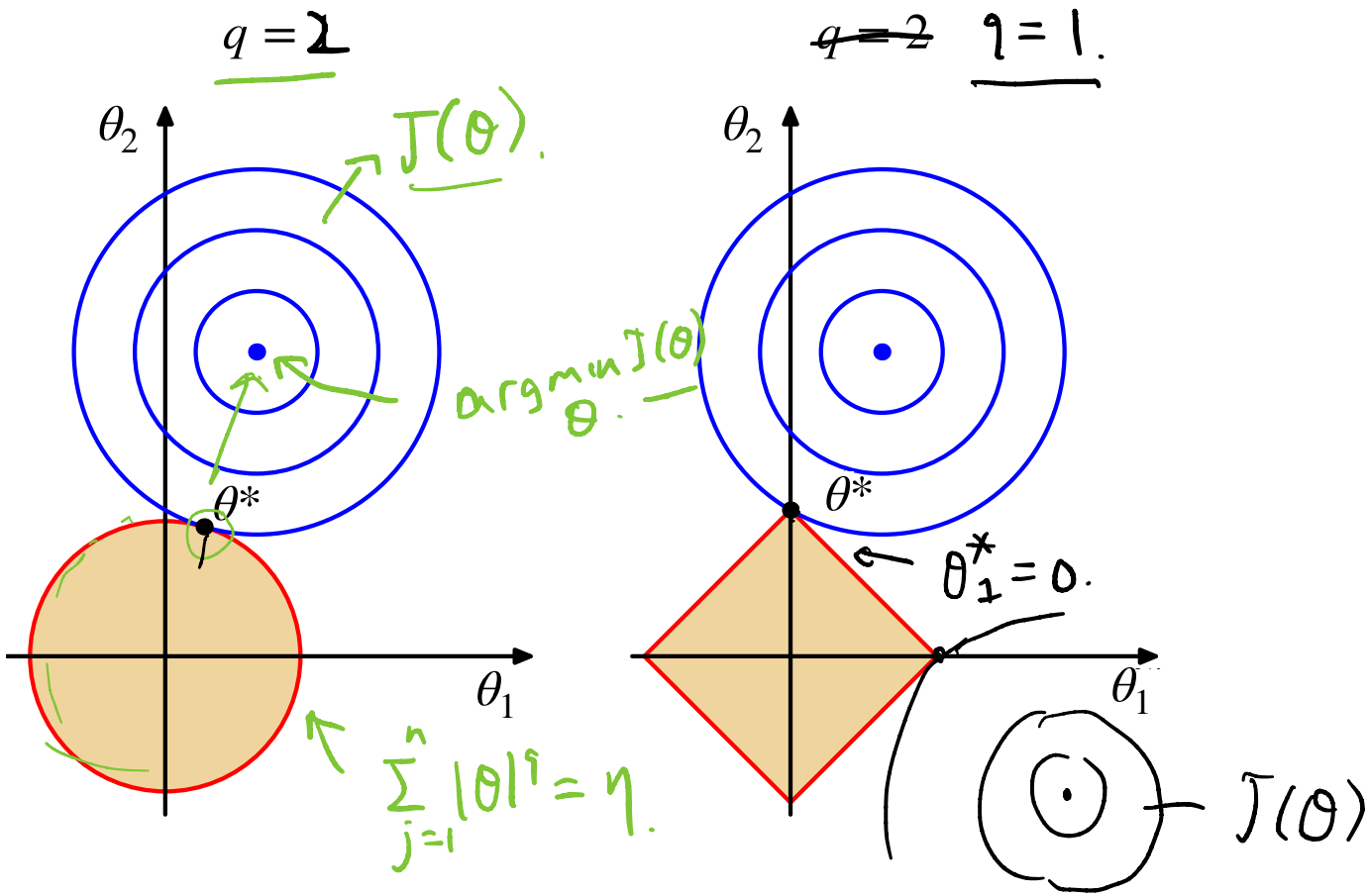
for some constant $\eta > 0$ (*). Furthermore, $\eta = \sum_{j=1}^{n} |\theta_j^*(\lambda)|^q$ where $\theta^*(\lambda) = \text{argmin}_\theta \tilde{J}(X, Y; \theta, \lambda)$

$\lambda > 0$, θ^* optimal for (1) $\Leftrightarrow \exists\, \eta > 0$, θ^* optimal for (2)

- (*) assume constraint is satisfiable (e.g. with slater's condition)
- Choosing λ is equivalent to choosing η and vice versa
- Smaller $\lambda \to$ larger constraint region

L1 vs L2 parameter penalty

contour plot of unregularized error $J(X, Y; \theta)$ and the constraint region $\sum_{j=1}^{n} |\theta|^q \leq \eta$



The lasso (l1 regularizer) gives a sparse solution with $\theta_1^* = 0$.

Bayesian Statistics

Maximum likelihood estimation: θ is an unknown constant

$$\theta_{MLE} = \arg\max_{\theta} \prod_{i=1}^{m} p(y^{(i)}|x^{(i)};\theta)$$

$y^{(i)}|x^{(i)} \sim P_{y|x}$

Bayesian view: θ is a random variable

$$\theta \sim p(\theta)$$

Given training set $S = \{x^{(i)}, y^{(i)}\}$, posterior distribution of θ

$$p(\theta|S) = \frac{p(S|\theta)p(\theta)}{p(S)}$$

evidence/data

Fully Bayesian statistics

$$p(\theta|S) = \frac{p(S|\theta)p(\theta)}{p(S)}$$

$$p(\theta|S) = \frac{\prod_{i=1}^{m} p(y^{(i)}|x^{(i)},\theta)p(\theta)}{\int_\theta (\prod_{i=1}^{m} p(y^{(i)}|x^{(i)},\theta)p(\theta))d\theta}$$

To predict the label for new sample x, compute the posterior distribution over training set S:

$$p(y|x,S) = \int_\theta p(y|x,\theta)p(\theta|S)d\theta$$

The label is

$$\mathbb{E}[y|x,S] = \int_y y\ p(y|x,S)dy$$

Fully bayesian estimate of θ is difficult to compute, has no close-form solution.

Bayesian Statistics

Posterior distribution on class label y using $p(\theta|S)$

$$p(y|x,S) = \int_{\theta} p(y|x,\theta)p(\theta|S)d\theta$$

Bayesian Statistics

Posterior distribution on class label y using $p(\theta|S)$

$$p(y|x,S) = \int_\theta p(y|x,\theta)p(\theta|S)d\theta$$

We can approximate $p(y|x,\theta)$ as follows:

MAP approximation

The **MAP (maximum a posteriori) estimate** of θ is

$$\theta_{MAP} = \underset{\theta}{\text{argmax}} \prod_{i=1}^{m} p(y^{(i)}|x^{(i)},\theta)p(\theta)$$

Bayesian Statistics

Posterior distribution on class label y using $p(\theta|S)$

$$p(y|x,S) = \int_\theta p(y|x,\theta)p(\theta|S)d\theta$$

We can approximate $p(y|x,\theta)$ as follows:

MAP approximation

The **MAP (maximum a posteriori) estimate** of θ is

$$\theta_{MAP} = \arg\max_\theta \prod_{i=1}^m p(y^{(i)}|x^{(i)},\theta)p(\theta)$$

$p(y^{(i)}|x^{(i)},\theta)$ is not the same as $p(y^{(i)}|x^{(i)};\theta)$

↑ RV.

↑ constant parameter

MAP estimation and regularized least square

Recall ordinary least square is equivalent to maximum likelihood estimation when $p(y^{(i)}|x^{(i)}) \sim \mathcal{N}(\theta^T x^{(i)}, \sigma^2)$:

$y = \theta^T x^i + \varepsilon$

$\varepsilon \sim N(0, \sigma^2)$

$$\theta_{MLE} = \arg\max_{\theta} \prod_{i=1}^{m} p(y^i|x^i;\theta)$$

$$= (X^T X)^{-1} X^T Y = \theta_{OLS}$$

MAP estimation and regularized least square

Recall ordinary least square is equivalent to maximum likelihood estimation when $p(y^{(i)}|x^{(i)}) \sim \mathcal{N}(\theta^T x^{(i)}, \sigma^2)$:

$$\begin{aligned}\theta_{MLE} &= \underset{\theta}{\text{argmax}} \prod_{i=1}^{m} p(y^i|x^i;\theta) \\ &= (X^T X)^{-1} X^T Y = \theta_{OLS}\end{aligned}$$

The MAP estimation when $\theta \sim N(0, \tau^2 I)$ is

$$\begin{aligned}\theta_{MAP} &= \underset{\theta}{\text{argmax}} \left(\prod_{i=1}^{m} p(y^i|x^i;\theta) \right) p(\theta) \\ &= \underset{\theta}{\text{argmin}} \left(\frac{\sigma^2}{\tau^2}\theta^T\theta + (Y - X\theta)^T(Y - X\theta) \right) \\ &= (X^T X + \frac{\sigma^2}{\tau} I)^{-1} X^T Y = \tilde{\theta}_{OLS} \text{ when } \lambda = \frac{\sigma^2}{\tau}\end{aligned}$$

Discussion on MAP Estimation

General remarks on MAP:

- When θ is uniform, $\theta_{MAP} = \theta_{MLE}$
- A common choice for $p(\theta)$ is $\theta \sim \mathcal{N}(0, \tau^2 I)$, and θ_{MAP} corresponds to weight decay (L2-regularization)
- When θ is an isotropic Laplace distribution, θ_{MAP} corresponds to LASSO (L1-regularization). *See WA3*
- θ_{MAP} often have smaller norm than θ_{MLE}

$\Updownarrow$

Lp- regularization

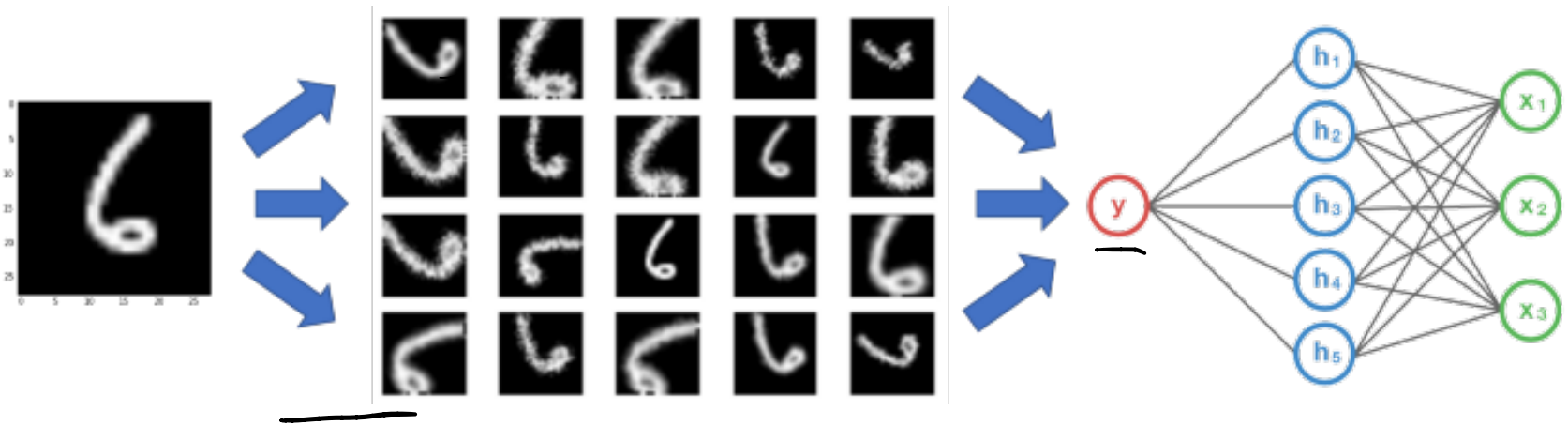
Regularization for neural networks

Common regularization techniques:

- Data augmentation
- Parameter sharing
- Drop out
- ...

Data augmentation

Create fake data and add it to the training set. (Useful in certain tasks such as object classification.)



Generate fake digits via geometric transformation, e.g. scale, rotation etc

Generate images of different styles using GAN

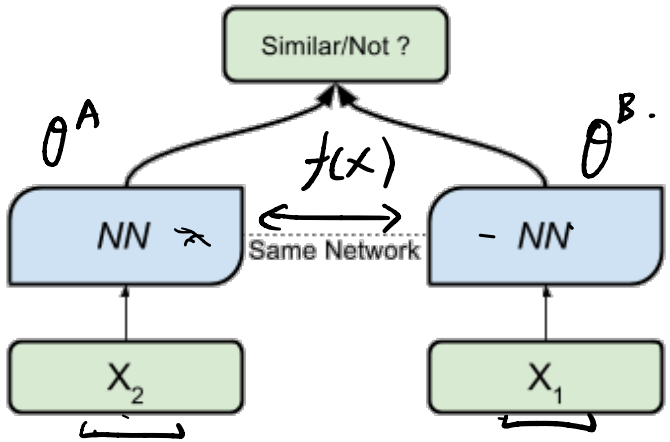
Shorten et. al. A survey on Image Data Augmentation for Deep Learning, 2019

Parameter Sharing

Force sets of parameters to be equal based on prior knowledge.

Siamese Network

- Given input X, learns a discriminative feature $f(X)$
- For every pair of samples (X_1, X_2) in the same class, minimize their distance in feature space $||f(X_1) - f(X_2)||^2$



Convolutional Neural Network (CNN)

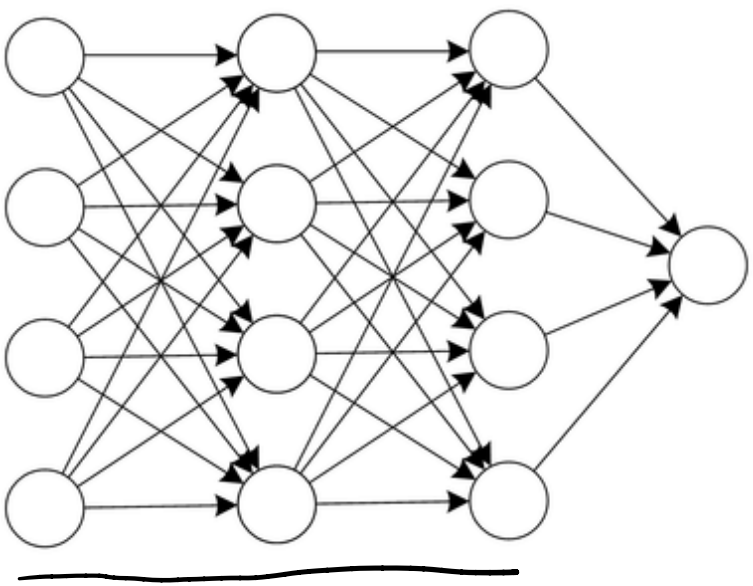
- Image features should be invariant to translation
- CNN shares parameters across multiple image locations.

Soft parameter sharing: add a norm penalty between sets of parameters:

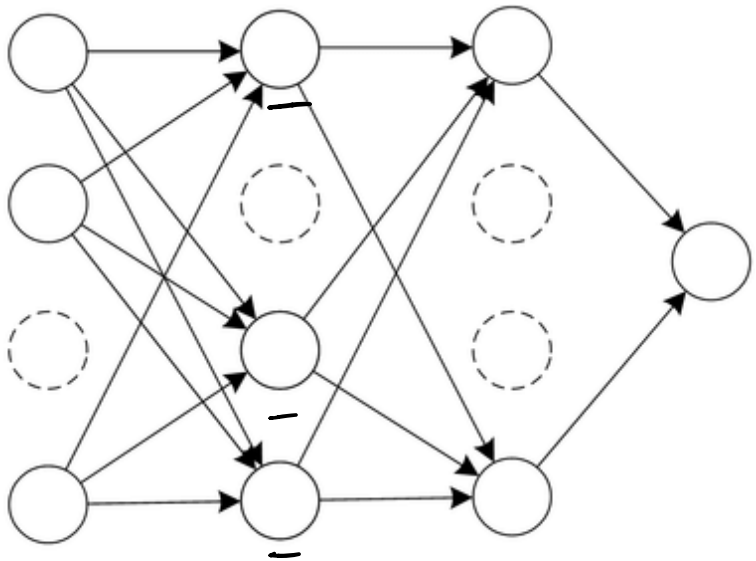
$$\Omega(\theta^A, \theta^B) = ||\theta^A - \theta^B||_2^2$$

Drop Out

- Randomly remove a non-output unit from network by multiplying its output by zero (with probability p)
- In each mini-batch, randomly sample binary masks to apply to all inputs and hidden units
- Dropout trains an ensemble of different sub-networks to prevent "co-adaptation" of neurons (i.e. highly correlated hidden units)



(a) Standard Neural Network



(b) Network after Dropout

Learning Theory

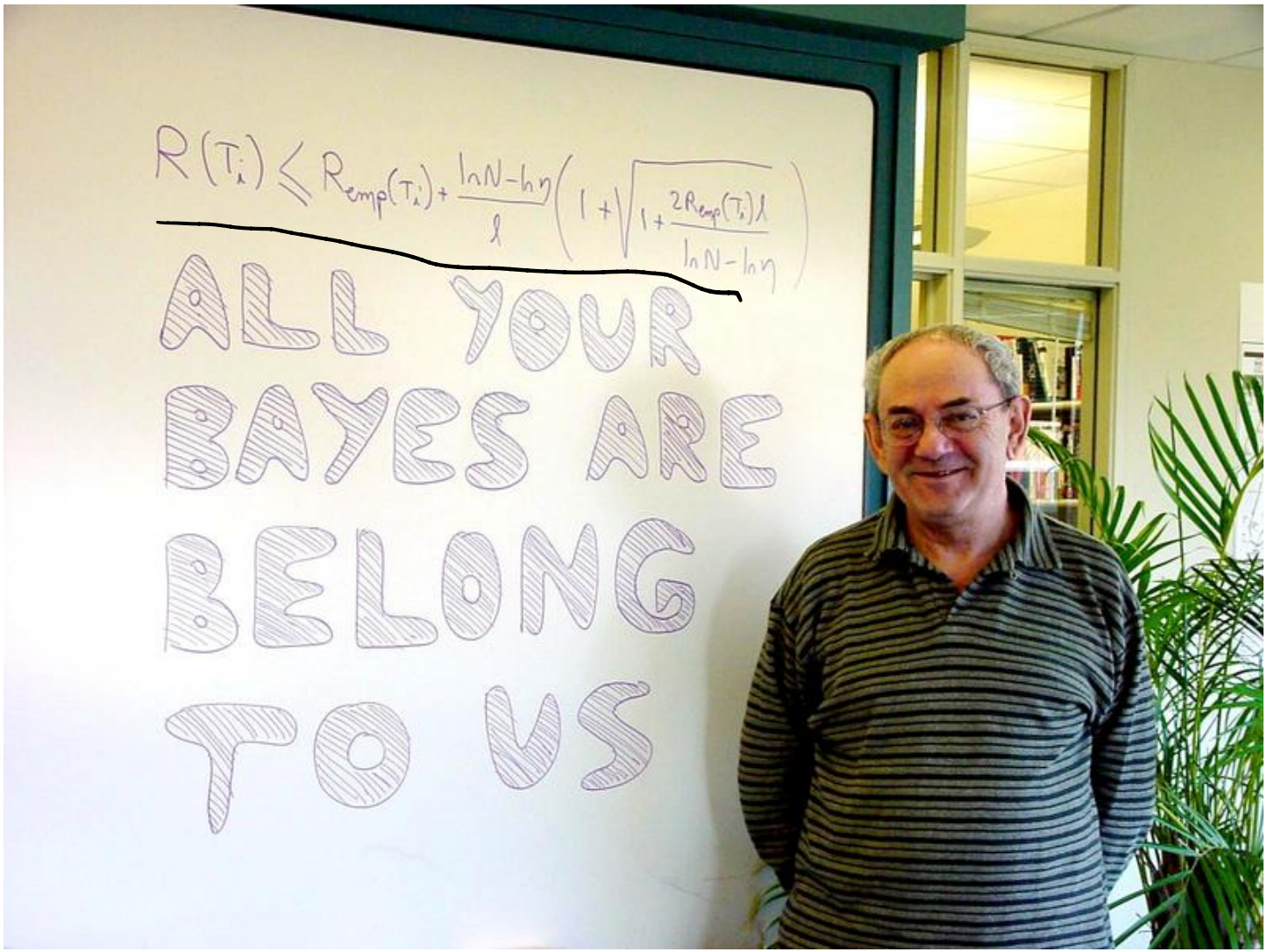
- Empirical Risk Estimation
- Uniform Convergence and Sample Complexity
- Infinite H

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

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 $\mathcal{H}$

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:

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

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

sample mean of Z_j.

Preliminaries

Here we make use of two famous inequalities:

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

$\phi \sim$ true mean

$\hat{\phi} \sim$ sample mean

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 $\gamma > 0$,

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

sample size

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

Case of Finite $\mathcal{H}$

$Z_j = 1\{h_i(x^{(j)}) \neq y^{(j)}\}$.

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

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

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

Case of Finite $\mathcal{H}$

Hoeffding:

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

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

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

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

$\phi = \epsilon(h_i)$

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

By Hoeffding inequality, for any h_i,

$$P(|\epsilon(h_i) - \hat{\epsilon}(h_i)| > \gamma) \le 2e^{-2\gamma^2 m} \quad (1)$$

A_i : $|\epsilon(h_i) - \hat{\epsilon}(h_i)| > \gamma$.

By union bound,

$$P(\exists h \in \mathcal{H} \mid |\epsilon(h) - \hat{\epsilon}(h)| > \gamma) = P(A_1 \cup \dots \cup A_k) \le \sum_{i=1}^{k} P(A_i)$$

$$\le \sum_{i=1}^{k} P(|\epsilon(h_i) - \hat{\epsilon}(h_i)| > \gamma)$$

By (1), $\le \sum_{i=1}^{k} 2e^{-2\gamma^2 m} = 2ke^{-2\gamma^2 m}$.

$P(\forall h \in \mathcal{H} \mid |\epsilon(h) - \hat{\epsilon}(h)| \le \gamma) \ge 1 - 2ke^{-2\gamma^2 m}$

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,

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

Given γ, with probability at least $1-\delta$, $\forall h \in H. |\epsilon(h) - \hat{\epsilon}(h)| \leq \gamma$.

Uniform Convergence Results

$P(\forall h \in H \mid |\epsilon(h) - \hat{\epsilon}(h)| \le \gamma) \ge 1 - \delta.$

$\delta = 2ke^{-2\gamma^2 m}$

$\log \delta = \log(2k) + (-2\gamma^2 m)$

$m = \frac{\log 2k - \log \delta}{2\gamma^2} = \frac{1}{2\gamma^2} \log\left(\frac{2k}{\delta}\right)$

Corollary 3

Given γ and $\delta > 0$, If

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

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

↑ lower bound of 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

m

$|k = |\mathcal{H}|$.

Uniform Convergence Results

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

$$\gamma = \sqrt{\frac{1}{2m} \log \frac{2k}{\delta}}.$$

Corollary 4

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

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

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 $\hat{h} = \text{argmin}_{h \in \mathcal{H}} \hat{\epsilon}(h)$

Uniform Convergence Theorem for Finite $\mathcal{H}$

$\hat{h} = \underset{h \in \mathcal{H}}{\operatorname{argmin}} \hat{\varepsilon}(h)$ ← empirical estimation

$h^* = \underset{h \in \mathcal{H}}{\operatorname{argmin}} \varepsilon(h)$ ← true hypothesis

Using previous corollaries, we can bound $\epsilon(\hat{h})$:

Theorem 5 (Uniform convergence)

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

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

generalization risk of $\hat{h}$; $\min_{h \in \mathcal{H}} \epsilon(h) = \varepsilon(h^*)$

proof. With prob. at least $1-\delta$, $m \geq \frac{1}{2\gamma^2} \log \frac{2k}{\delta}$, $|\varepsilon(h) - \hat{\varepsilon}(h)| \leq \gamma$ for all h.

$|a| \leq \gamma$. $-\gamma \leq a \leq \gamma$.

Then $|\varepsilon(\hat{h}) - \hat{\varepsilon}(\hat{h})| \leq \gamma$

$\varepsilon(\hat{h}) - \hat{\varepsilon}(\hat{h}) \leq \gamma$.

$\varepsilon(\hat{h}) \leq \gamma + \hat{\varepsilon}(\hat{h})$.

$\hat{\varepsilon}(\hat{h}) \leq \hat{\varepsilon}(h^*)$ by definition.

$\varepsilon(\hat{h}) \leq \gamma + \hat{\varepsilon}(h^*)$. (1).

similarly.

$|\varepsilon(h^*) - \hat{\varepsilon}(h^*)| \leq \gamma$

$\varepsilon(h^*) - \hat{\varepsilon}(h^*) \geq -\gamma$

$\hat{\varepsilon}(h^*) \leq \varepsilon(h^*) + \gamma$. (2).

Combine (1) and (2).

$\varepsilon(\hat{h}) \leq \gamma + \hat{\varepsilon}(h^*) \leq \gamma + \varepsilon(h^*) + \gamma$ ((2))

$\varepsilon(\hat{h}) \leq \varepsilon(h^*) + 2\gamma$.

By corollary 4.

$\varepsilon(\hat{h}) \leq \varepsilon(h^*) + 2\sqrt{\frac{1}{2m} \log \frac{2k}{\delta}}$. □

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.

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: $|\mathcal{H}| = 2^{64d}$

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 $1-\delta$?

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

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 $1-\delta$?

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

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:

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

Infinite hypothesis class: Challenges

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

Example

$2n + 2$ parameters:

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

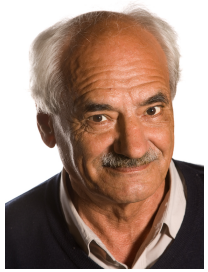
is equivalent the hypothesis with $n + 1$ parameters:

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

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

Infinite hypothesis class: Vapnik-Chervonenkis theory

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



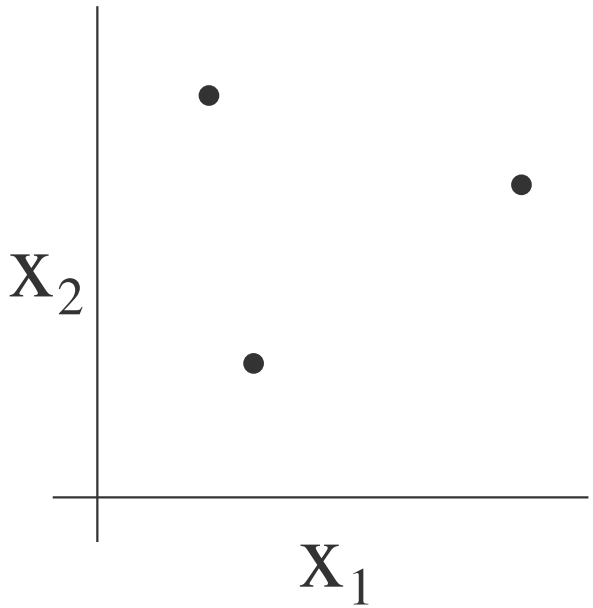
Alexey Chervonenkis (1938-2014), Russian mathematician

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 $x^{(i)} \in \mathbb{R}^2$.

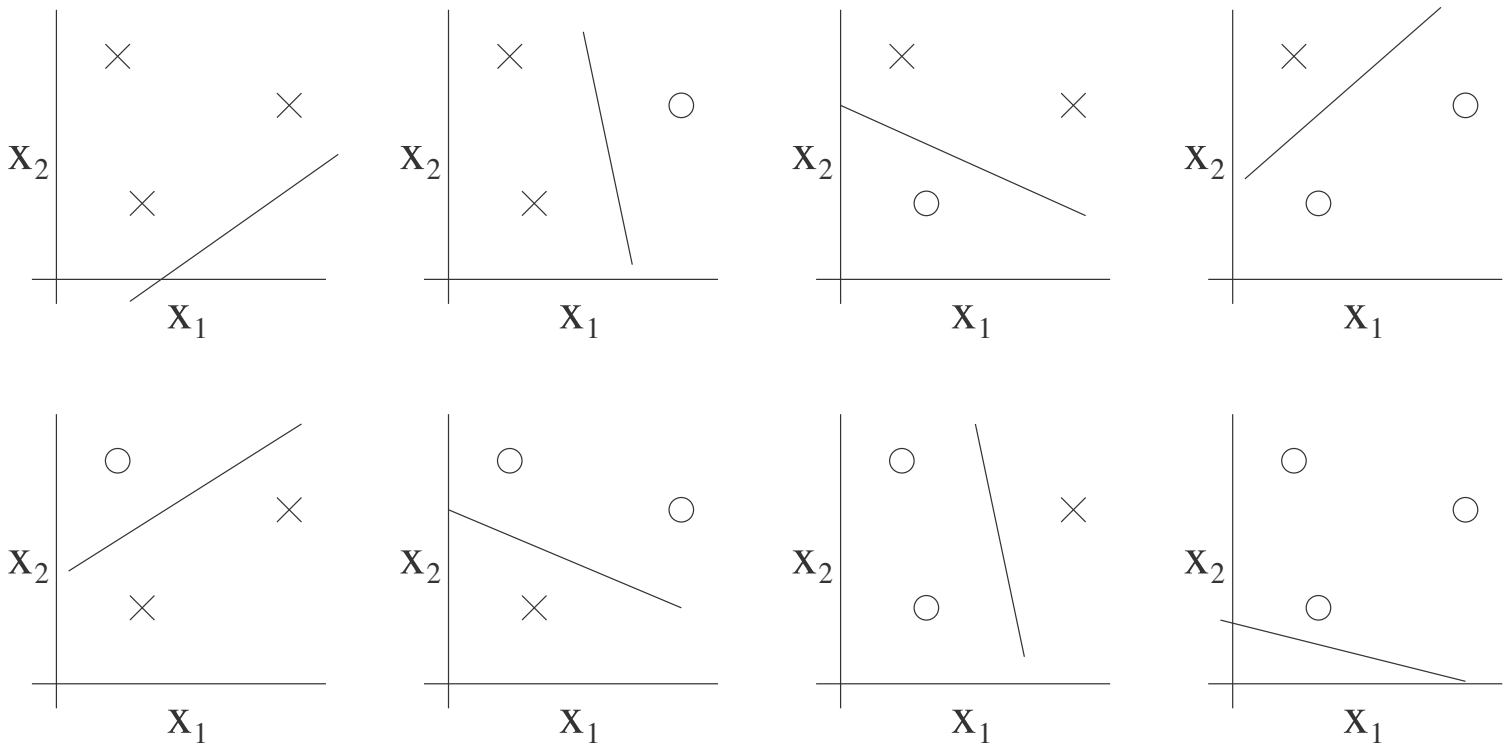


Suppose $y^{(i)} \in \{0, 1\}$, how many possible labelings does S have?

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)

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

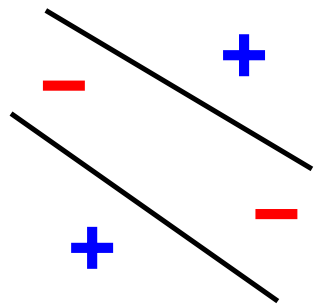


$\mathcal{H}_{LTF,2}$ shatters $S = \{x^{(1)}, x^{(2)}, x^{(3)}\}$

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

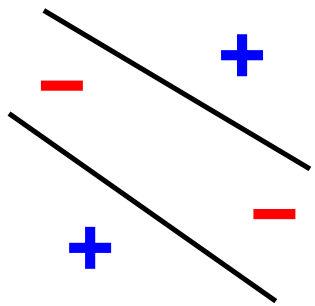


$\mathcal{H}_{LTF}$ can not shatter 4 points: for any 4 points, label points on the diagonal as '+'. (See Radon's theorem)

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

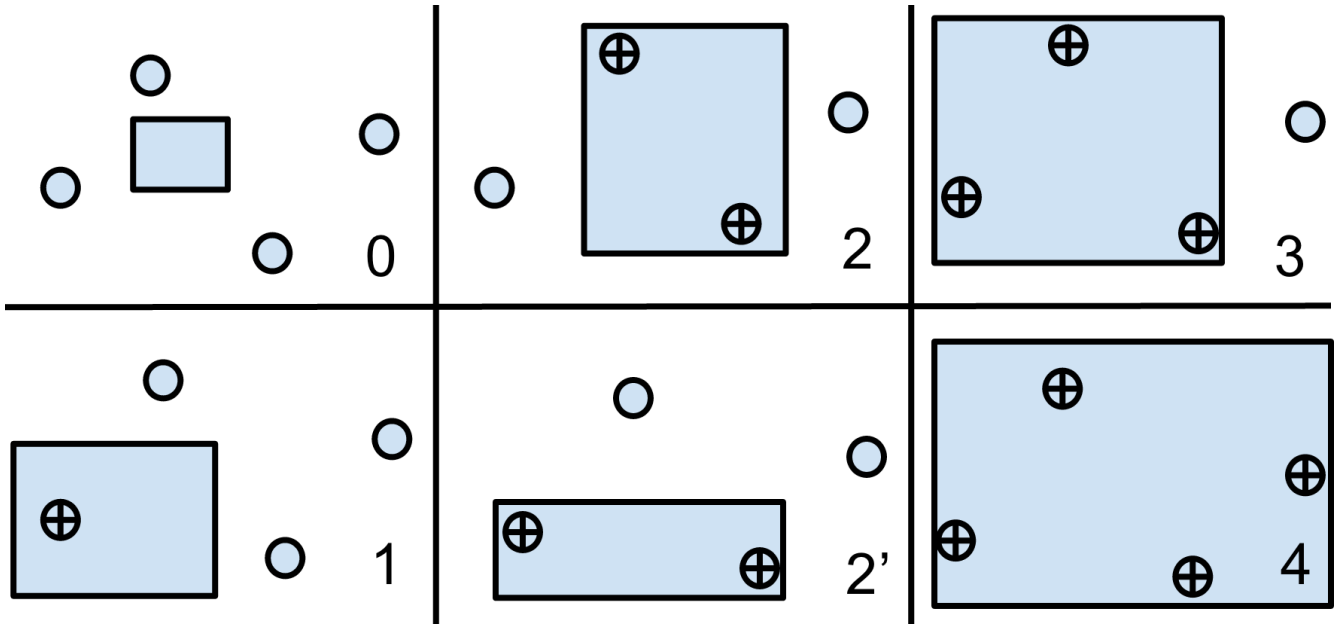


$\mathcal{H}_{LTF}$ can not shatter 4 points: for any 4 points, label points on the diagonal as '+'. (See Radon's theorem)

- 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

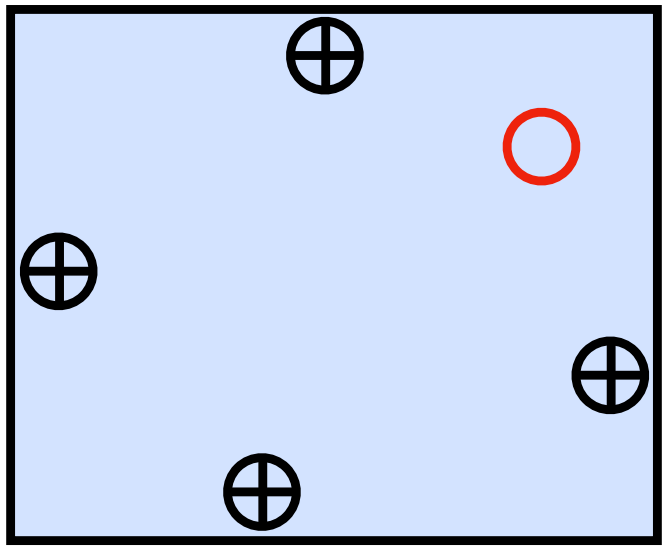
- Example: $VC(AxisAlignedRectangles) = 4$



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

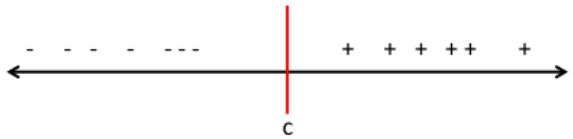
More VC results of common $\mathcal{H}$:

- $VC(ConstantFunctions) =$

Discussion on VC Dimension

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

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

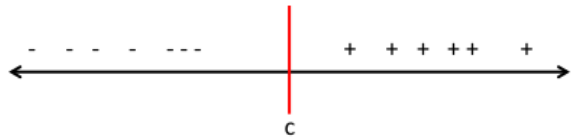


- $VC(Intervals) = 2, \mathcal{X} = \mathbb{R}$
- $VC(LTF\ in\ \mathbb{R}^n) = n + 1, \mathcal{X} = \mathbb{R}^n \leftarrow$ *prove this at home!*

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 2

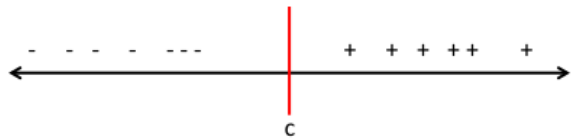
If $\mathcal{H}$ is finite, VC dimension is related to the cardinality of $\mathcal{H}$:

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

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 2

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*

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

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 $m = O_{y,\delta}(d)$.

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 $VC(\mathcal{N}) = O(w \log w)$.

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]

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.

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.