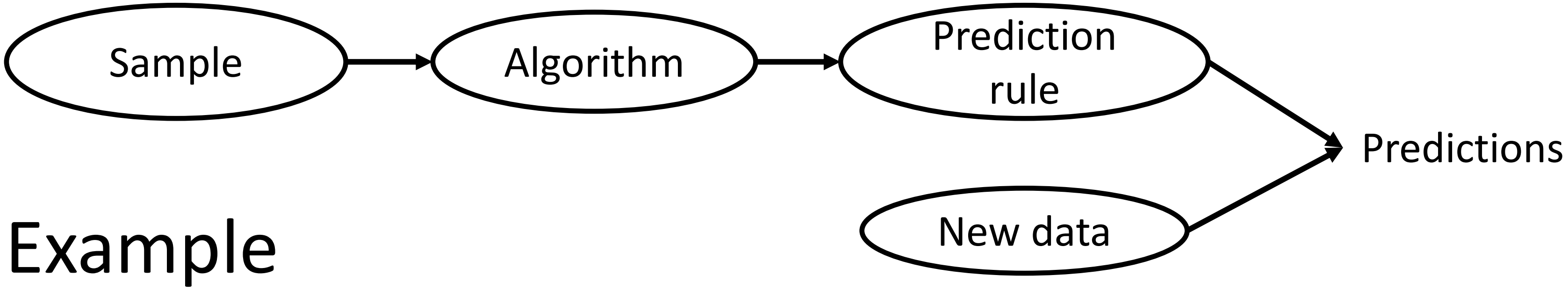
Supervised Learning
K Nearest Neighbors
Validation

Yevgeny Seldin

Supervised Learning

- Protocol



Sample

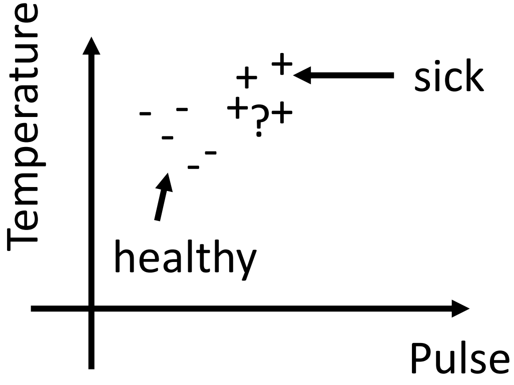
Algorithm

Prediction rule

New data

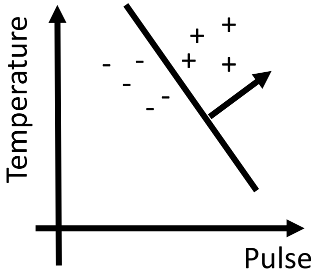
Predictions

- Example



Supervised Learning

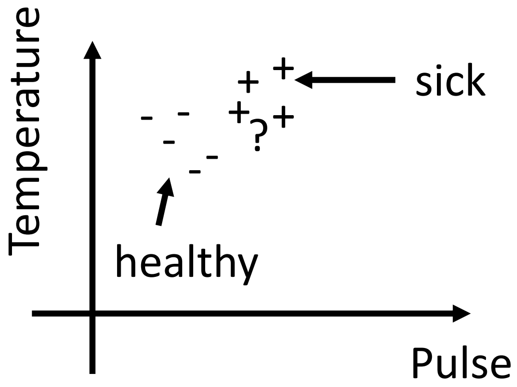
- More examples
  - (age, gender, weight) → height
  - (age, weight, height) → gender
  - (height(1), height(2), height(3)) → height(4)
- Notations
  - $\mathcal{X}$ – sample space (e.g., $\mathcal{X} = \mathbb{R}^d$)
  - $\mathcal{Y}$ – label space (e.g., Classification: $\mathcal{Y} = \{\pm 1\}$; Regression: $\mathcal{Y} = \mathbb{R}$)
  - $S = \{(X_1, Y_1), \dots, (X_n, Y_n)\}$ – training sample (where $X_i \in \mathcal{X}, Y_i \in \mathcal{Y}$)
  - $h: \mathcal{X} \to \mathcal{Y}$ – a prediction rule / hypothesis
  - $\mathcal{H}$ - a set of prediction rules / a hypothesis set



K-Nearest Neighbors (K-NN)

- Algorithm: Predict Y based on K nearest neighbors of X in S.

- Input: distance measure $d(x, x')$
- Examples:
  - Euclidian distance
  - Manhattan distance
  - Travel distance
  - Edit distance



- **The choice of d determines the success or failure of K-NN!**

Evaluation

- $\ell(y', y)$ – loss/error function

  Loss for predicting y' when the reality is y

- Examples:
  - Zero-one loss

    $$\ell(y', y) = \mathbb{I}(y' \neq y)$$

    $$= \begin{cases} 0, & if\ y' = y \\ 1, & if\ y' \neq y \end{cases}$$

  - Squared loss

    $$\ell(y', y) = (y' - y)^2$$

  - Absolute loss

    $$\ell(y', y) = |\mathrm{y}' - \mathrm{y}|$$

- **The loss function determines the cost of different mistakes!!!**

- Example: Fire alarm

Depends on the house

y y'	no fire	fire
no fire	0	5.000.000
fire	2.000	0

“constant”

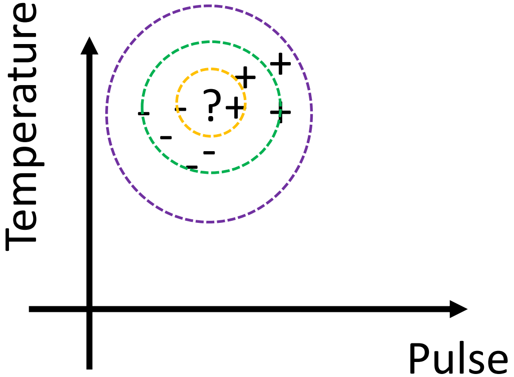
So far

- KNN – predict based on K nearest neighbors

- Input:
  - Distance measure $d(x, x')$ – domain knowledge

- Evaluation:
  - Loss function $\ell(y', y)$ – domain knowledge

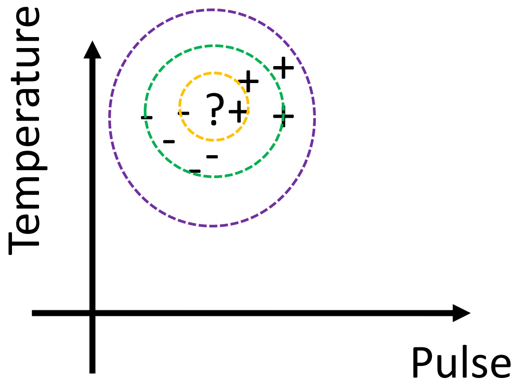
How to pick K?



- What is good/bad about small *K*?
  - Say, *K*=1?

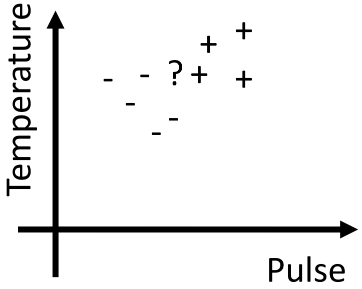
- What is good/bad about large *K*?
  - Say, *K*=n?

How to pick K?



- Target: minimize the expected loss
  - $L(h_{KNN}) = \mathbb{E}[\ell(h_{KNN}(X), Y)]$
- Assumption
  - (X, Y) are sampled from a fixed (unknown) distribution $p(X, Y)$
  - The expectation is with respect to $p(X, Y)$
- Challenge: $p(X, Y)$ is unknown, and so is $L(h_{KNN})$
- How to estimate $L(h_{KNN})$?
  - Use the empirical loss $\hat{L}(h_{KNN}, S) = \frac{1}{n}\sum_{i=1}^{n} \ell(h_{KNN}(X_i), Y_i)$
  - What is $\hat{L}(h_{1NN}, S)$?
  - In general, $\hat{L}(h_{KNN}, S)$ is an underestimate of $L(h_{KNN})$.

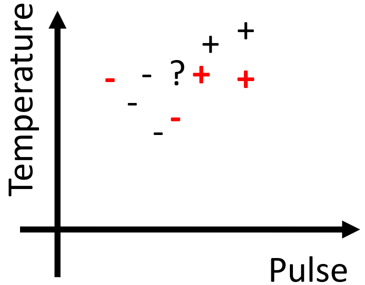
Validation



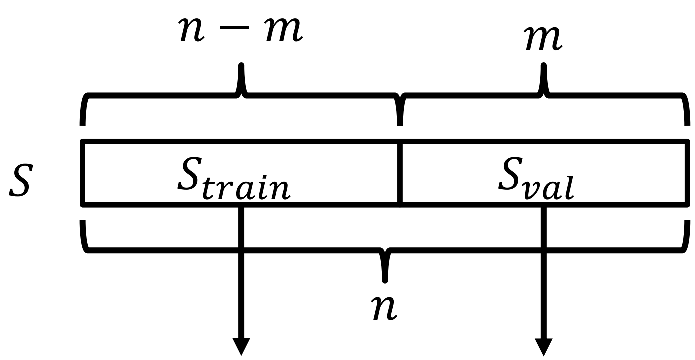
S

(X, Y)

Validation



$\boxed{(X, Y)}$



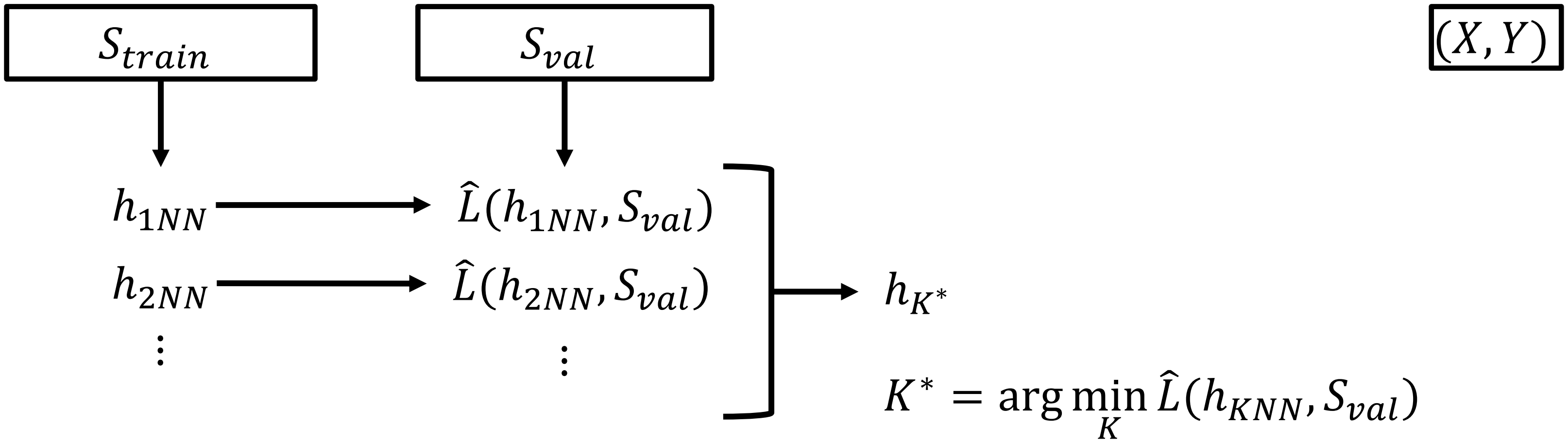
h_{KNNS_t}

$$\hat{L}(h_{KNNS_t}, S_{val}) = \frac{1}{m}\sum_{i=1}^{m} \ell(h_{KNNS_t}(X_i), Y_i)$$

- **Assumptions**
  - $\{(\boldsymbol{X_1}, \boldsymbol{Y_1}), \ldots, (\boldsymbol{X_m}, \boldsymbol{Y_m})\}$ **are independent identically distributed (i.i.d.)**
  - **And come from the same distribution as new samples** $(\boldsymbol{X}, \boldsymbol{Y})$

- $\hat{L}(h_{KNNS_t}, S_{val})$ is an **unbiased** estimate of $L(h_{KNNS_t})$
  - $\mathbb{E}[\hat{L}(h_{KNNS_t}, S_{val})] = \mathbb{E}\left[\frac{1}{m}\sum_{i=1}^{m} \ell(h_{KNNS_t}(X_i), Y_i)\right] = \frac{1}{m}\sum_{i=1}^{m} \mathbb{E}[\ell(h_{KNNS_t}(X_i), Y_i)] = L(h_{KNNS_t})$
  - From the perspective of h_{KNNS_t} the samples in S_{val} are indistinguishable from new samples (X, Y)

Selection of K



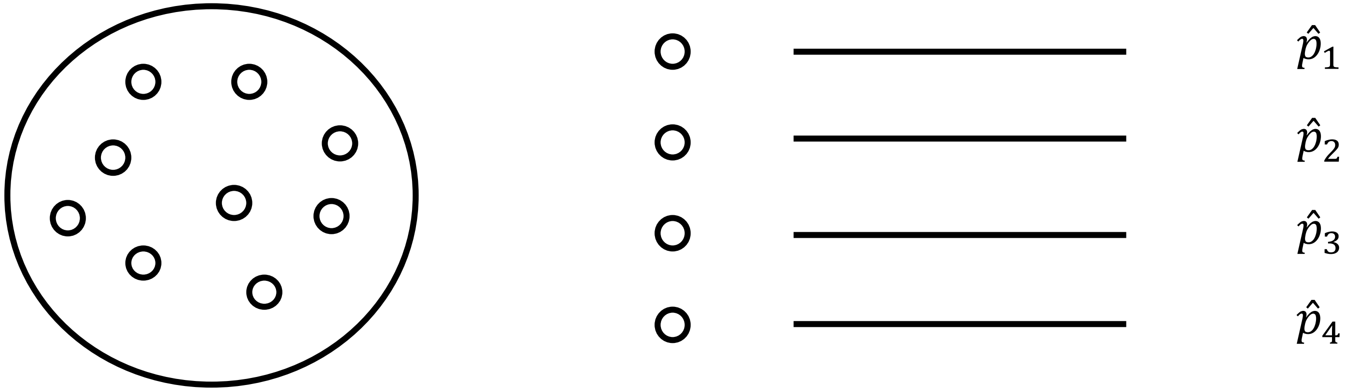
$$K^* = \arg\min_K \hat{L}(h_{KNN}, S_{val})$$

- **Selection introduces bias!**
  - Each $\hat{L}(h_{KNN}, S_{val})$ is an unbiased estimate of $L(h_{KNN})$
  - But $\hat{L}(h_{K^*NN}, S_{val})$ is a **biased** estimate of $L(h_{K^*NN})$!!!
    - From the perspective of h_{K^*NN} the samples in S_{val} are distinguishable from new samples (X,Y)
    - $\mathbb{E}[\ell(h_{K^*NN}(X_i), Y_i)] \neq \mathbb{E}[\ell(h_{K^*NN}(X), Y)]$

dependent!

in S_{val}

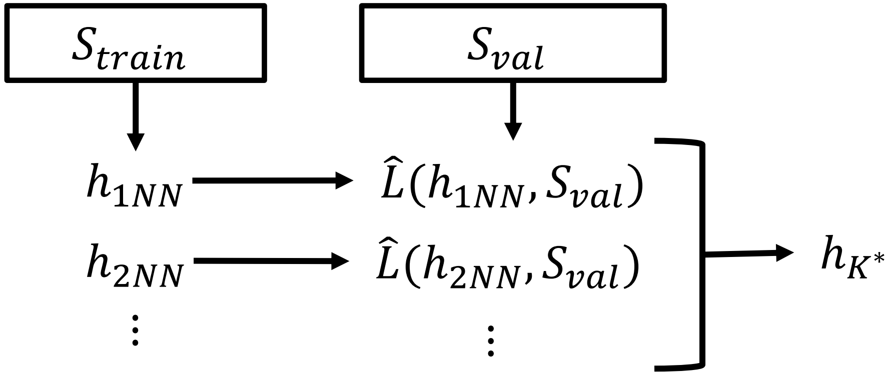
Illustration of Selection Bias



A bag of coins with bias p

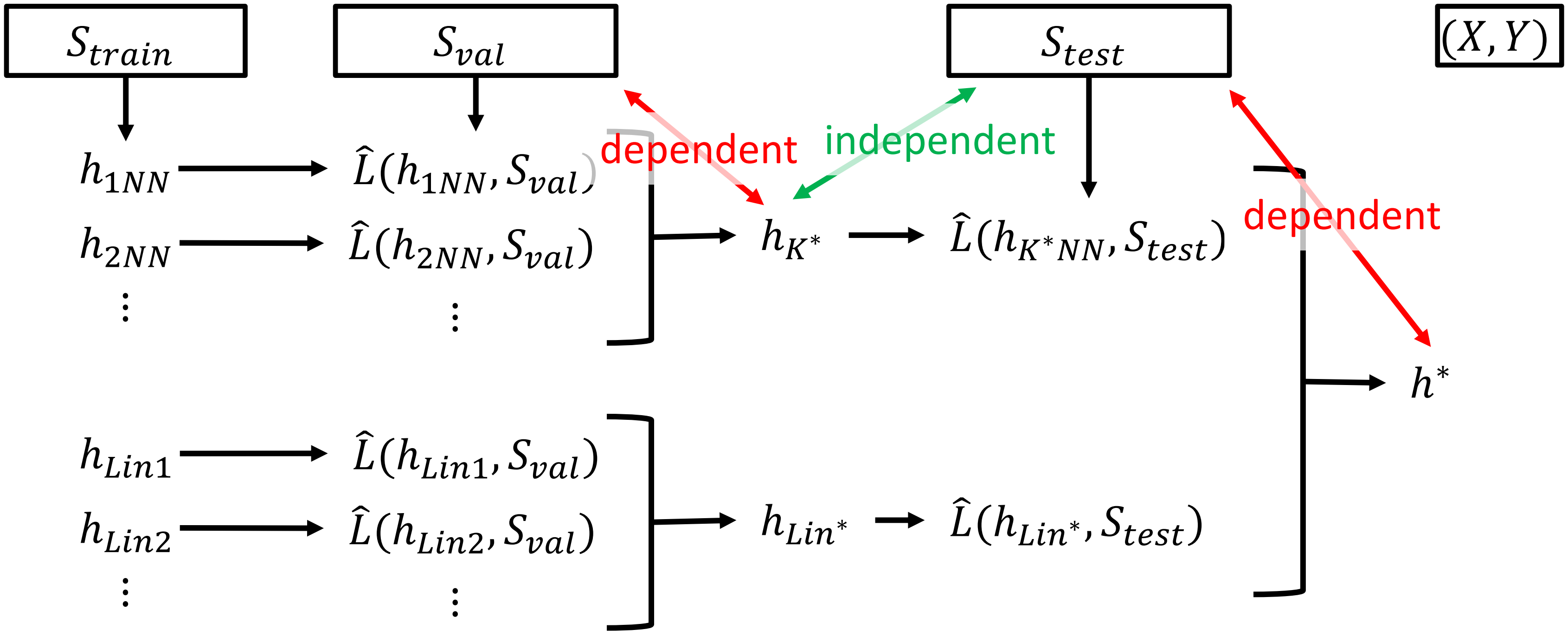
- $i^* = \arg\min_i \hat{p}_i$
- While each $\hat{p}_i$ is an unbiased estimate of p: $\mathbb{E}[\hat{p}_i] = p$
- $\hat{p}_{i^*}$ is a **biased** estimate of p: $\mathbb{E}[\hat{p}_{i^*}] \neq p$
- Outcome-based selection introduces bias!

So how can we estimate $L(h_{K^*NN})$?



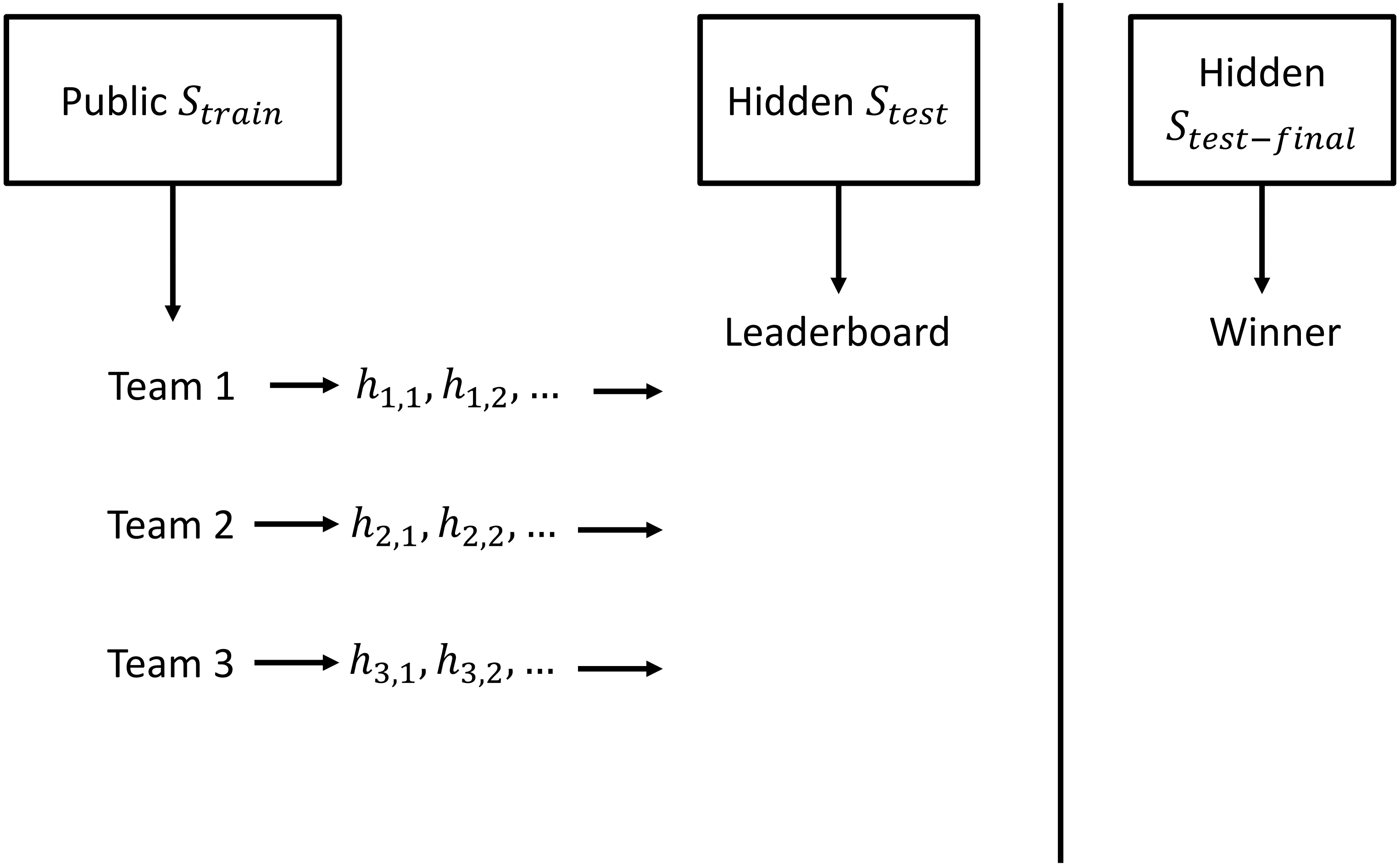
$\boxed{(X, Y)}$

Testing


S_{train}
S_{val}
S_{test}
(X,Y)
h_{1NN}
$\hat{L}(h_{1NN}, S_{val})$
h_{2NN}
$\hat{L}(h_{2NN}, S_{val})$
$\vdots$
dependent
independent
h_{K^*}
$\hat{L}(h_{K^*NN}, S_{test})$
dependent
h^*
h_{Lin1}
$\hat{L}(h_{Lin1}, S_{val})$
h_{Lin2}
$\hat{L}(h_{Lin2}, S_{val})$
h_{Lin^*}
$\hat{L}(h_{Lin^*}, S_{test})$

- **It's not about how you call it; it's about how you use it!!!**
- $\hat{L}(h^*, S_{test})$ is a **biased** estimate of $L(h^*)$!

Respectable ML Competitions



Public S_{train}

Hidden S_{test}

Leaderboard

Hidden $S_{test-final}$

Winner

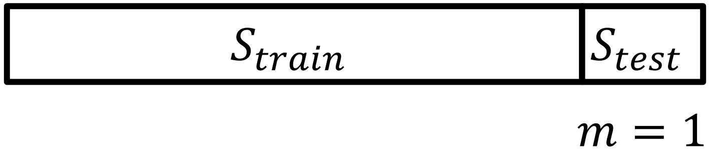
Team 1 → $h_{1,1}, h_{1,2}, \ldots$ →

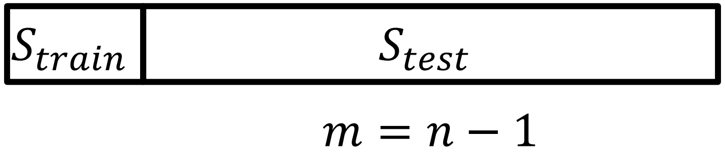
Team 2 → $h_{2,1}, h_{2,2}, \ldots$ →

Team 3 → $h_{3,1}, h_{3,2}, \ldots$ →

How to split the data into train/test/…?

- Consider the following extremes:





  - $m = 1$
    - $\hat{L}(h, S_{test}) \in \{0,1\}$, never approaches $L(h)$
  - $m = n - 1$
    - The training procedure only observes one label

What can be said about $L(h)$ based on $\hat{L}(h, S_{val})$?

- $\hat{L}(h, S_{val})$ is an unbiased estimate of $L(h)$

- But consider the case $m = 1$:
  - $\hat{L}(h, S_{val}) \in \{0,1\}$ – never close to $L(h)$!

- Being unbiased is neither sufficient, nor necessary

- We need concentration!

Relation to “coin flips”

- $Z_i = \ell(h(X_i), Y_i) \in \{0,1\}$
  - Bernoulli random variable, “a coin flip”
- $\mathbb{E}[Z_i] = \mathbb{E}[\ell(h(X_i), Y_i)] = L(h) = p$
  - The bias of the coin
- $\hat{L}(h, S_{val}) = \frac{1}{m}\sum_i^m Z_i = \hat{p}_m$
  - An average of m “coin flips”
- How far can $\hat{p}_m$ be from p?
  - We will study this later in the course

Do we need to reshuffle the data?
(before splitting into train/validation/test)

- Theory:
  - The data are assumed to be i.i.d., so it does not matter
- Practice:
  - Yes, if the data are sorted by irrelevant parameter
  - No, if data order carries information relevant for testing, e.g., ordering by time

Division of responsibilities

Task	User	Algorithm

Division of responsibilities

Task	User	Algorithm
Evaluation/error measure $\ell(y', y)$		

Division of responsibilities

Task	User	Algorithm
Evaluation/error measure $\ell(y', y)$	V	

Division of responsibilities

Task	User	Algorithm
Evaluation/error measure $\ell(y', y)$	V	
Distance measure $d(x, x')$		

Division of responsibilities

Task	User	Algorithm
Evaluation/error measure $\ell(y', y)$	V	
Distance measure $d(x, x')$	V	V

Division of responsibilities

Task	User	Algorithm
Evaluation/error measure $\ell(y', y)$	V	
Distance measure $d(x, x')$	V	V
K		

Division of responsibilities

Task	User	Algorithm
Evaluation/error measure $\ell(y', y)$	V	
Distance measure $d(x, x')$	V	V
K		V

Division of responsibilities

Task	User	Algorithm
Evaluation/error measure $\ell(y', y)$	V	
Distance measure $d(x, x')$	V	V
K		V
Validation size m		

Division of responsibilities

Task	User	Algorithm
Evaluation/error measure $\ell(y', y)$	V	
Distance measure $d(x, x')$	V	V
K		V
Validation size m		V

Division of responsibilities

Task	User	Algorithm
Evaluation/error measure $\ell(y', y)$	V	
Distance measure $d(x, x')$	V	V
K		V
Validation size m		V

- Learning is used for *selection*
  - Selection of d
  - Selection of K
  - Selection of m

- The options for selection are provided by the user, but can be quite broad

- The evaluation measure ℓ is provided solely by the user (cannot be selected by learning)

Summary

- Selection introduces bias
- Data not involved in selection can be used to obtain unbiased loss estimates
- It’s not about how you call it, it’s about how you use it!
- Unbiasedness is neither necessary, nor sufficient, we need concentration!
- In the remainder of the course we will study how to live with the bias and achieve concentration