CSC 580 Principles of Machine Learning

02 Limits of Learning

Jason Pacheco

Department of Computer Science

THE UNIVERSITY OF ARIZONA

*slides credit: built upon CSC 580 Fall 2021 lecture slides by Chicheng Zhang & Kwang-Sung Jun

Motivation

- Machine learning is a general & useful framework…**but it's not "magic"**

- Understand when machine learning will and will not work

Optimal classification with known D

Suppose

- Binary classification: 0-1 loss $\ell(y, \hat{y}) = I(y \neq \hat{y})$
- Data Generating distribution D known for every (x, y)

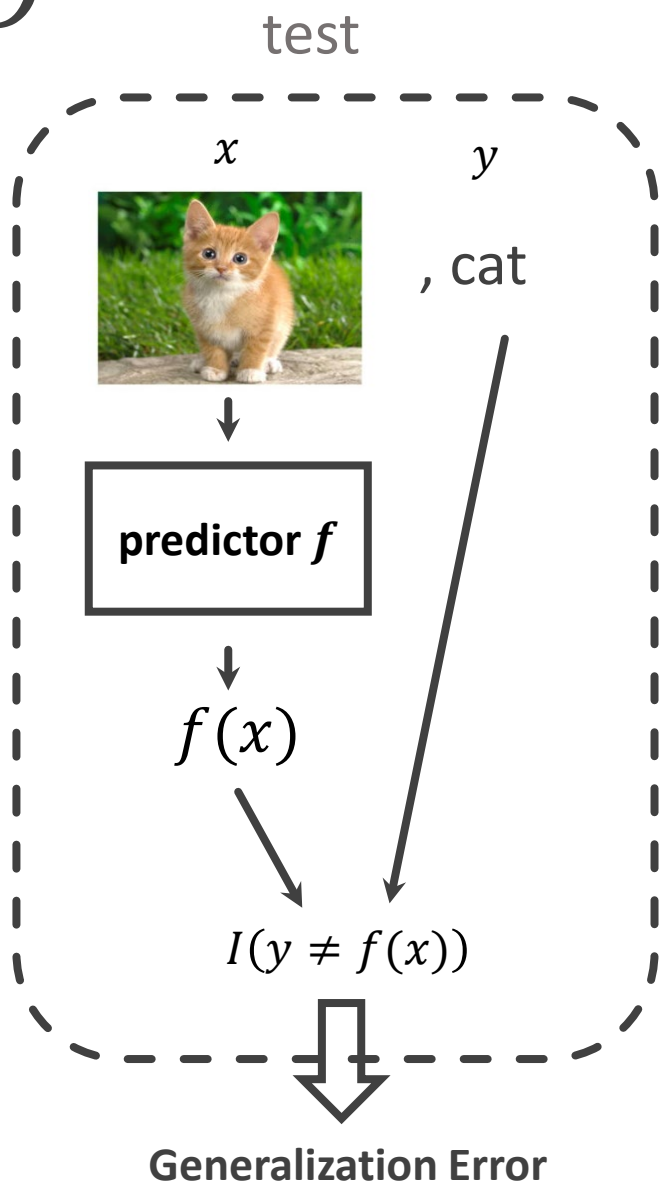
Generalization Error

$$L_D(f) = \mathrm{E}_{(x,y)\sim D} I(y \neq f(x)) = P_{(x,y)\sim D}\,(y \neq f(x))$$

Question

What is the f that minimizes,

$$L_D(f) = P_{(x,y)\sim D}(y \neq f(x))$$



Simple case: discrete domain $\mathcal{X}$

$P_D(x, y)$	$x = 1$	$x = 2$	$x = 3$
$y = -1$	0.2	0.2	0.15
$y = +1$	0.1	0.3	0.05

Which classifier is better?

- $f_1(1) = -1, f_1(2) = -1, f_1(3) = -1 \quad \Rightarrow \quad L_D(f_1) = 0.1 + 0.3 + 0.05$
- $f_2(1) = -1, f_2(2) = +1, f_2(3) = -1 \quad \Rightarrow \quad L_D(f_2) = 0.1 + 0.2 + 0.05$

Is this the best classifier? Why?

- For any x, should choose y that has higher value of $P_D(x, y)$
- $f^*(1) = -1, f^*(2) = +1, f^*(3) = -1$

Bayes optimal classifier

Theorem f_{BO} achieves the smallest 0-1 error among all classifiers.

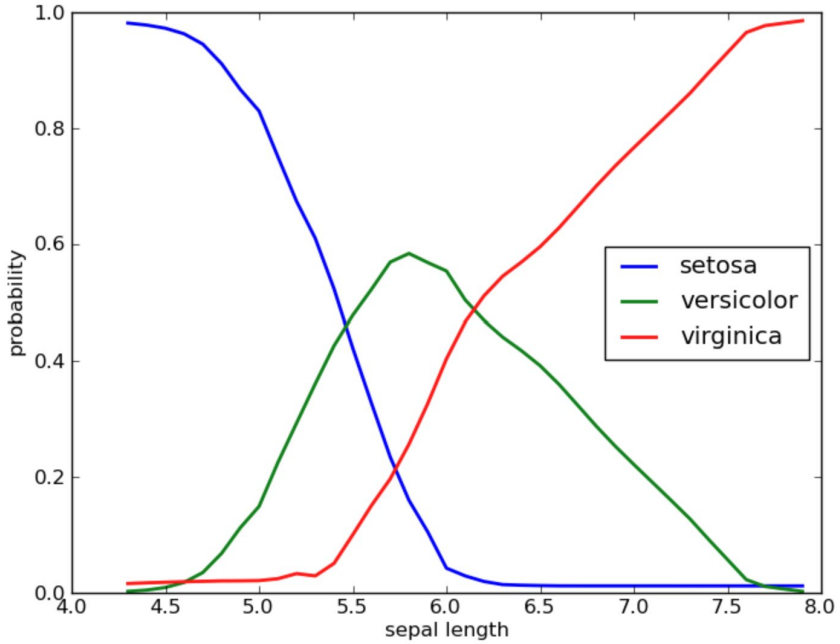
$$f_{BO}(x) = \arg\max_{y \in \mathcal{Y}} P_D(X = x, Y = y) = \arg\max_{y \in \mathcal{Y}} P_D(Y = y \,|X = x)\,, \forall x \in \mathcal{X}$$

Example Iris dataset classification:

Iris Setosa

Iris Versicolor

Iris Virginica



Proof of theorem

Step 1 consider accuracy,

- $A_D(f) = 1 - L_D(f) = P_D\big(Y = f(X)\big) = \sum_x P_D\big(X = x, Y = f(x)\big)$
- Suffices to show f_{BO} has the highest accuracy

Step 2 comparison,

$$A_D(f_{BO}) - A_D(f) = \sum_x P_D\big(X = x, Y = f_{BO}(x)\big) - P_D\big(X = x, Y = f(x)\big) \geq 0$$

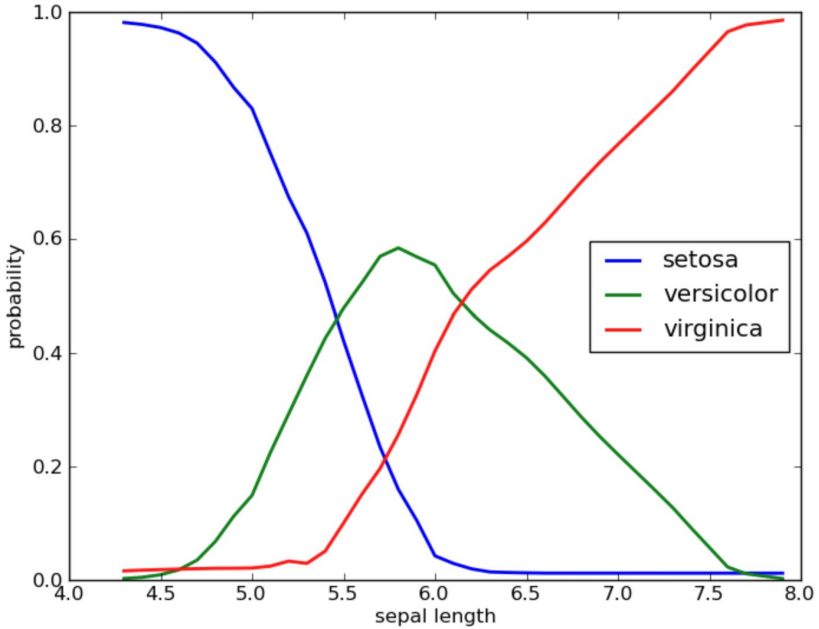
$$f_{BO}(x) = \arg\max_{y \in \mathcal{Y}} P_D(X = x, Y = y)$$

Remarks

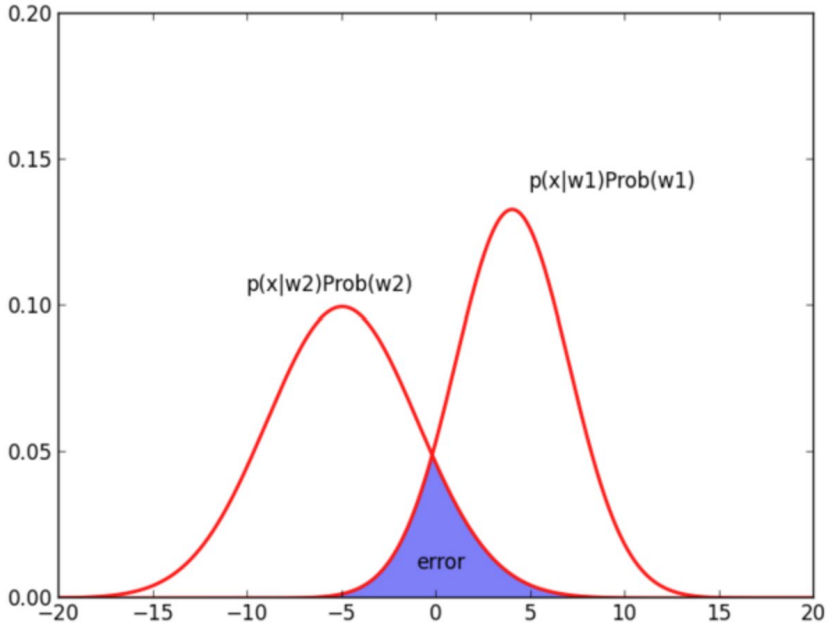
- Similar reasoning can be used to prove the theorem with continuous domain $\mathcal{X}$ (sum -> integral)
- This just shows deterministic classifier, can be extended to show BO is 0-1 optimal for all classifiers

Bayes error rate: alternative form

$$
\begin{aligned}
L_D(f_{BO}) &= P_D\big(Y \neq f_{BO}(X)\big) \\
&= \textstyle\sum_x P_D(Y \neq f_{BO}(x) \mid X = x)\, P_D(X = x) \\
&= \textstyle\sum_x (1 - P_D(Y = f_{BO}(x) \mid X = x))\, P_D(X = x) \\
&= \textstyle\sum_x \left(1 - \max_y P_D(Y = y \mid X = x)\right) P_D(X = x) \\
&= \mathrm{E}\left[1 - \max_y P_D(Y = y \mid X)\right]
\end{aligned}
$$



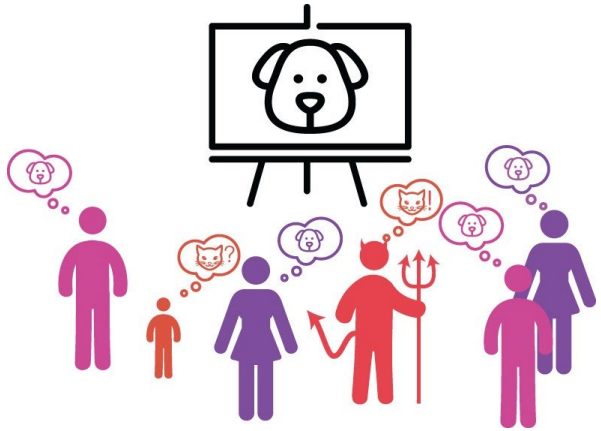
- Special case: binary classification
  - $L_D(f_{BO}) = \sum_x P_D(Y \neq f_{BO}(x), X = x)$
    $= \sum_x \min(\, P_D(Y = +1, X = x)\,, P_D(Y = -1, X = x))$

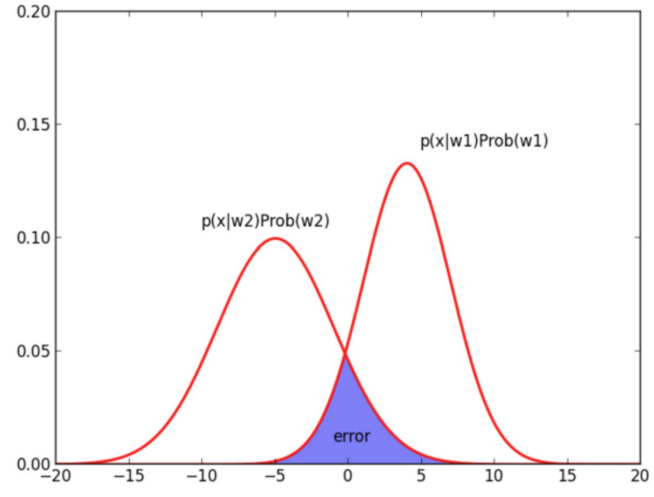


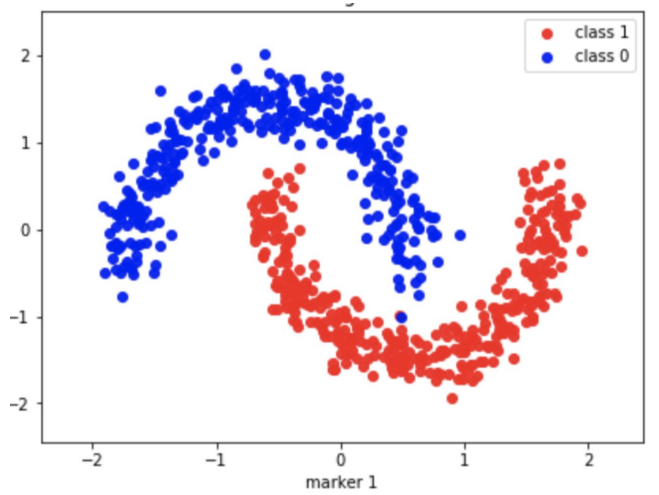
When is the Bayes error rate nonzero?

$$L_D(f_{BO}) = \sum_x \min(\, P_D(Y = +1, X = x)\,, P_D(Y = -1, X = x))$$

- Limited feature representation
- Noise in the training data
  - Feature noise
  - Label noise – e.g. typo transcribing reviews
  - Sensor failure
  - Typo in reviews for sentiment classification
- May not be a single “correct” answer
- Inductive bias of the model / learning algorithm







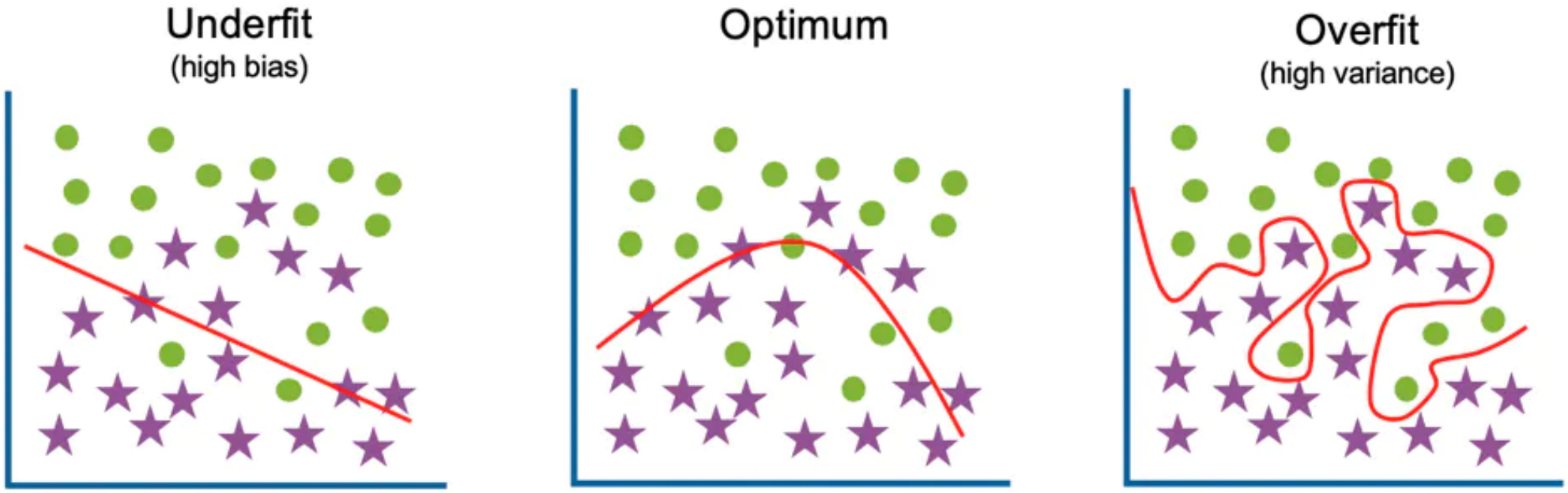
Inductive Bias

Training

Test

How would you label the test examples?

Overfitting vs Underfitting

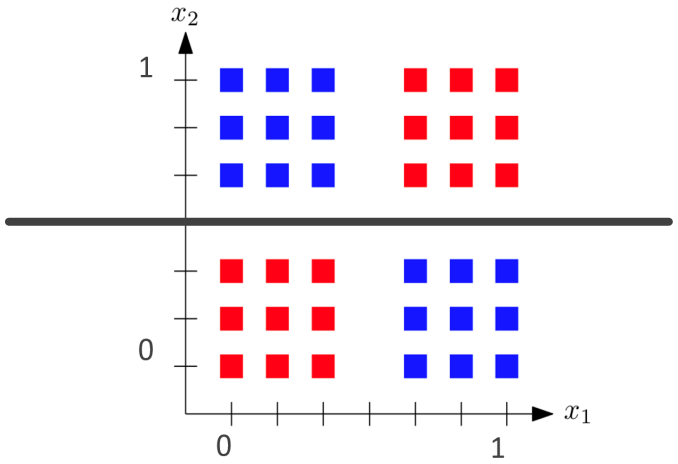


Source: ibm.com

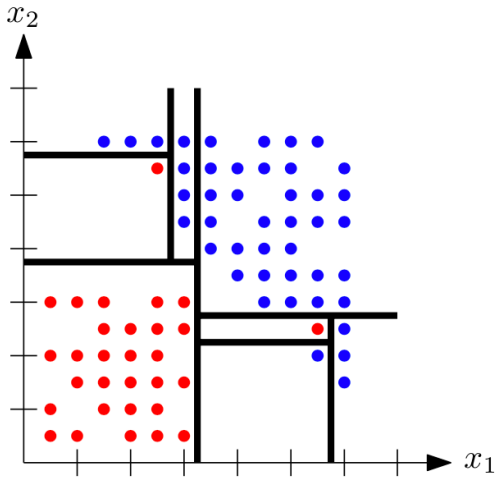
{Over,Under}-fitting

What is the inductive bias of a shallow decision tree?

Shallow tree:



Deep tree:

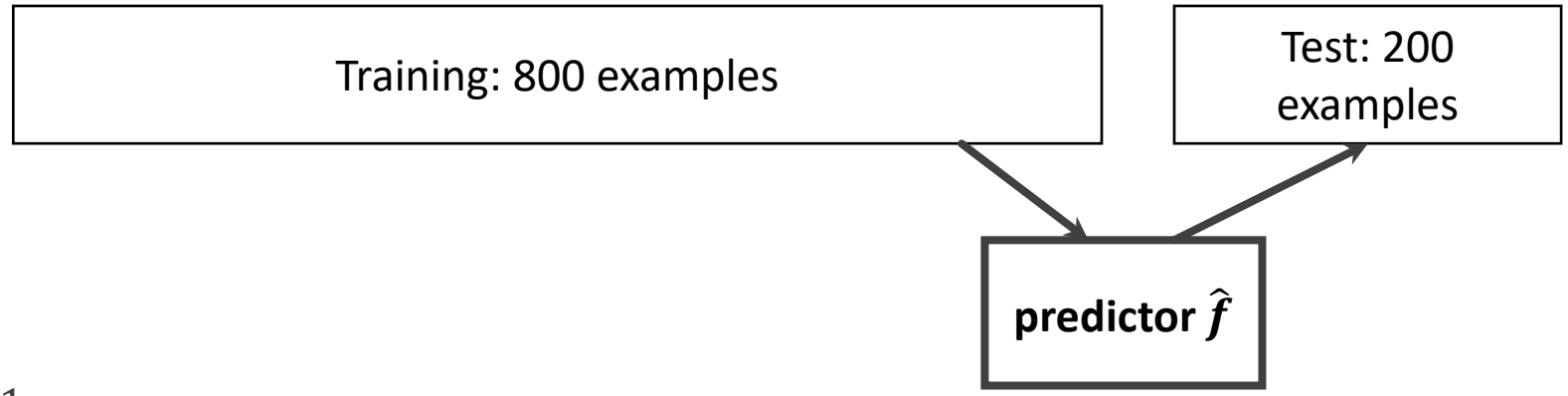


- Underfitting: Can learn something but didn't
- Overfitting: Pay too much attention to idiosyncrasies to training data, and do not generalize well
- A model that neither overfits nor underfits is expected to do best

Unbiased model evaluation using test data

Your Boss says: You may run your recommendation system to on our website only if error <= 10%!

- How can we prove that this is satisfied?
- Idea: reserve some data as test data for evaluating predictors

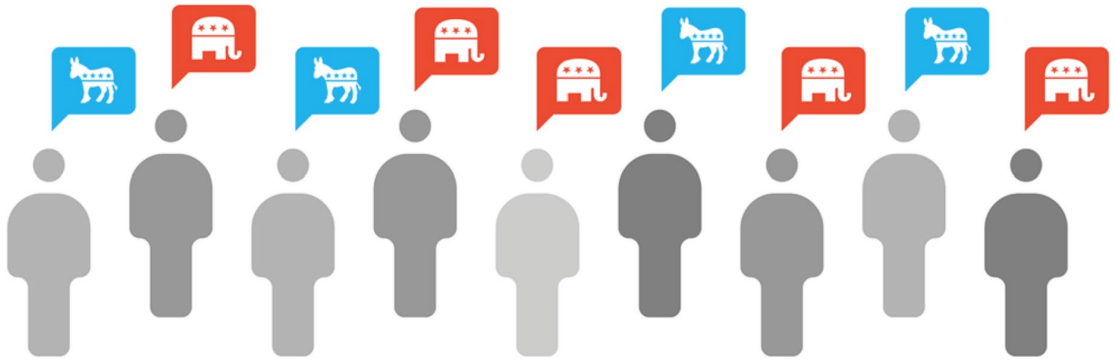


- $L_{\text{test}}(\hat{f}) = \frac{1}{|S_{\text{test}}|} \sum_{(x,y) \in S_{\text{test}}} I(y \neq \hat{f}(x))$
- Law of large numbers $\Rightarrow L_{\text{test}}(\hat{f}) \rightarrow L_D(\hat{f})$

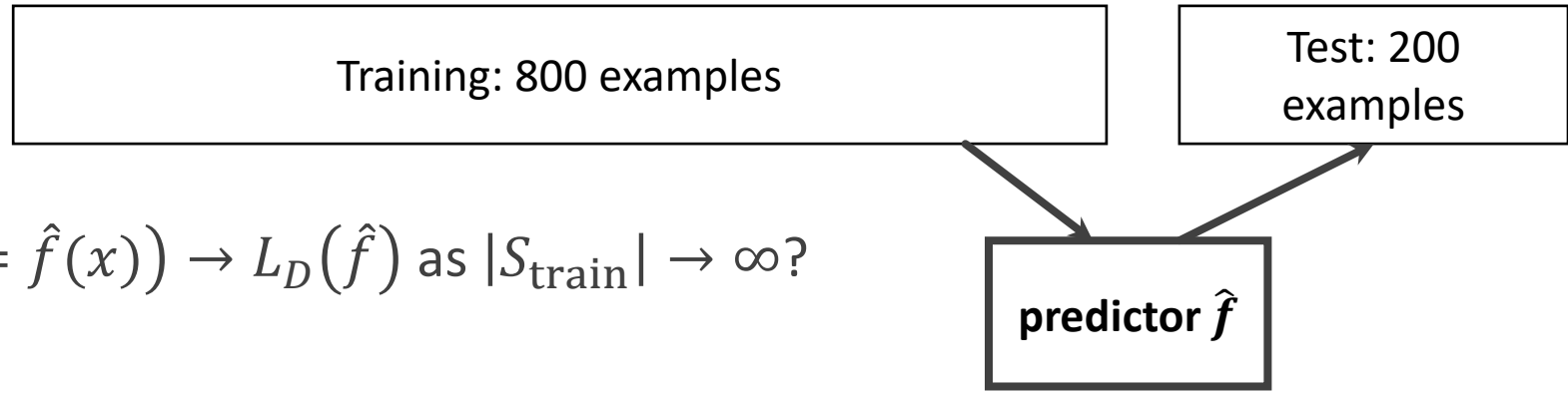
Law of large numbers (LLN)

- Suppose $v_1, \ldots, v_n$ are independent random variables that are identically distributed, the sample average $\bar{v} = \frac{1}{n}\sum_{i=1}^{n} v_i$ converges to $\mathrm{E}[v_1]$ as $n \to \infty$

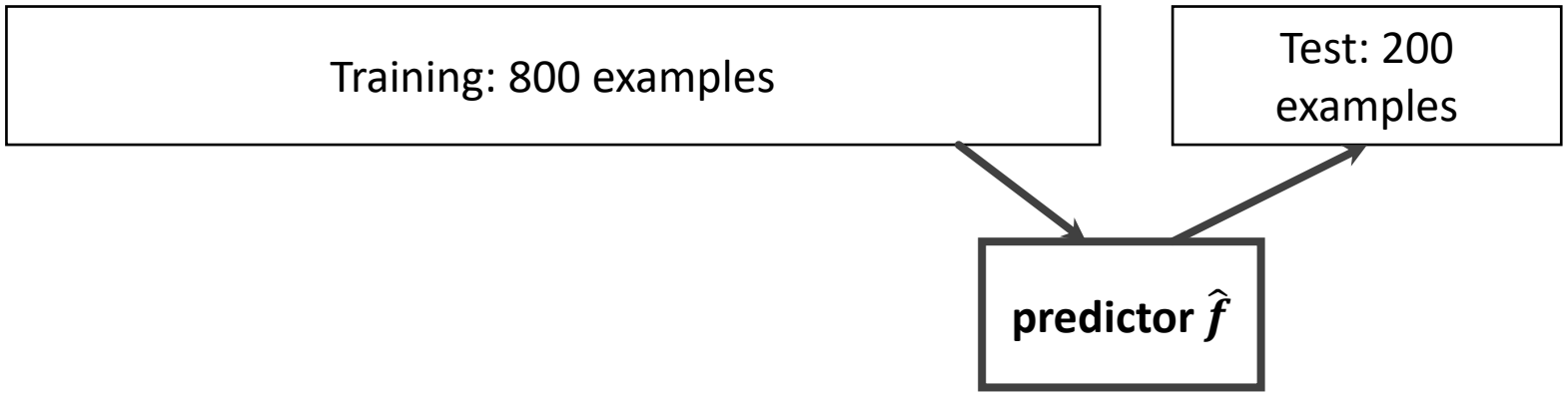
- Useful in e.g. election poll
- Foundations of statistics



- Can we apply LLN to conclude that
- $L_{\text{train}}(\hat{f}) = \frac{1}{|S_{\text{train}}|}\sum_{(x,y)\in S_{\text{train}}} I\left(y \neq \hat{f}(x)\right) \to L_D(\hat{f})$ as $|S_{\text{train}}| \to \infty$?



Never touch your test data!



- If $\hat{f}$ depends on test examples, $L_{\text{test}}(\hat{f})$ may no longer estimate $L_D(\hat{f})$ accurately
- E.g. indirect dependence:
  - adaptive data analysis – choose a new learning algorithm based on seeing that the previous algorithm produces a high-test-error model

Case Study: MNIST Dataset

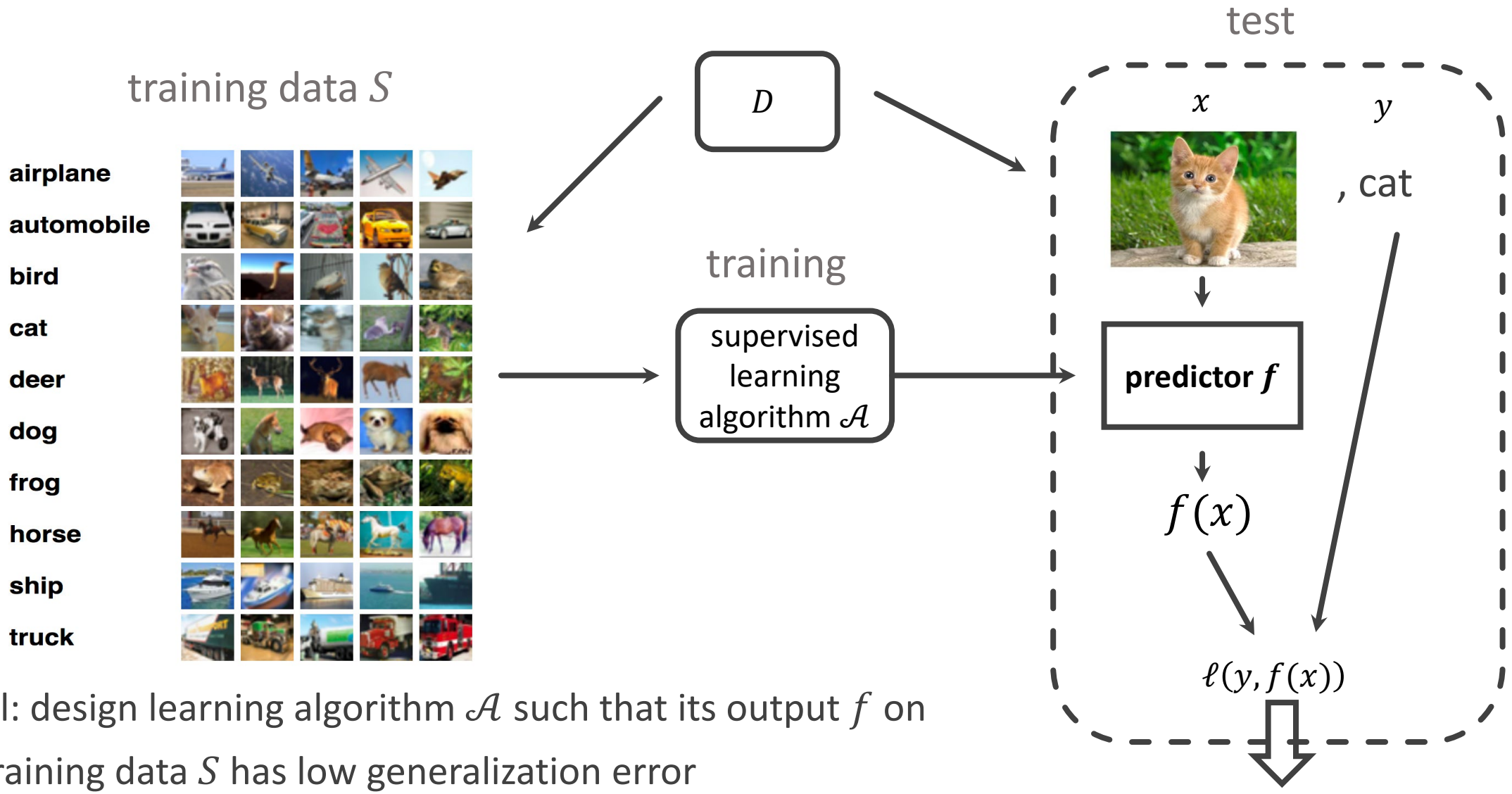
All publications use standard train/test split

Hundreds of publications compare to each other

Type	Classifier	Distortion	Preprocessing	Error rate (%)
Linear classifier	Pairwise linear classifier	None	Deskewing	7.6[10]
Decision stream with Extremely randomized trees	Single model (depth > 400 levels)	None	None	2.7[28]
K-Nearest Neighbors	K-NN with rigid transformations	None	None	0.96[29]
K-Nearest Neighbors	K-NN with non-linear deformation (P2DHMDM)	None	Shiftable edges	0.52[30]
Boosted Stumps	Product of stumps on Haar features	None	Haar features	0.87[31]
Non-linear classifier	40 PCA + quadratic classifier	None	None	3.3[10]
Random Forest	Fast Unified Random Forests for Survival, Regression, and Classification (RF-SRC)[32]	None	Simple statistical pixel importance	2.8[33]
Support-vector machine (SVM)	Virtual SVM, deg-9 poly, 2-pixel jittered	None	Deskewing	0.56[34]
Deep neural network (DNN)	2-layer 784-800-10	None	None	1.6[35]
Deep neural network	2-layer 784-800-10	Elastic distortions	None	0.7[35]
Deep neural network	6-layer 784-2500-2000-1500-1000-500-10	Elastic distortions	None	0.35[36]
Convolutional neural network (CNN)	6-layer 784-40-80-500-1000-2000-10	None	Expansion of the training data	0.31[37]
Convolutional neural network	6-layer 784-50-100-500-1000-10-10	None	Expansion of the training data	0.27[38]
Convolutional neural network (CNN)	13-layer 64-128(5x)-256(3x)-512-2048-256-256-10	None	None	0.25[22]
Convolutional neural network	Committee of 35 CNNs, 1-20-P-40-P-150-10	Elastic distortions	Width normalizations	0.23[17]
Convolutional neural network	Committee of 5 CNNs, 6-layer 784-50-100-500-1000-10-10	None	Expansion of the training data	0.21[24][25]
Random Multimodel Deep Learning (RMDL)	10 NN-10 RNN - 10 CNN	None	None	0.18[27]
Convolutional neural network	Committee of 20 CNNS with Squeeze-and-Excitation Networks[39]	None	Data augmentation	0.17[40]
Convolutional neural network	Ensemble of 3 CNNs with varying kernel sizes	None	Data augmentation consisting of rotation and translation	0.09[41]

What's the problem with this?

Supervised learning setup

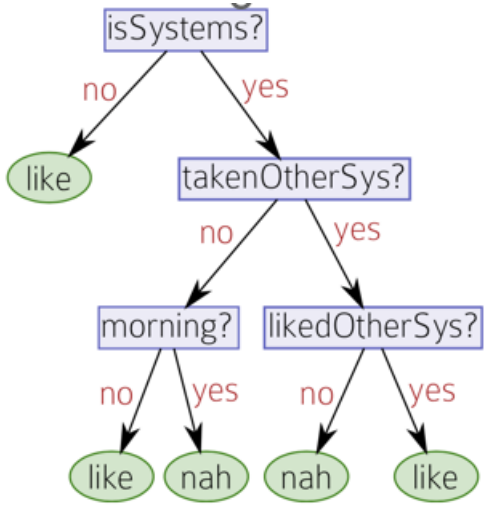


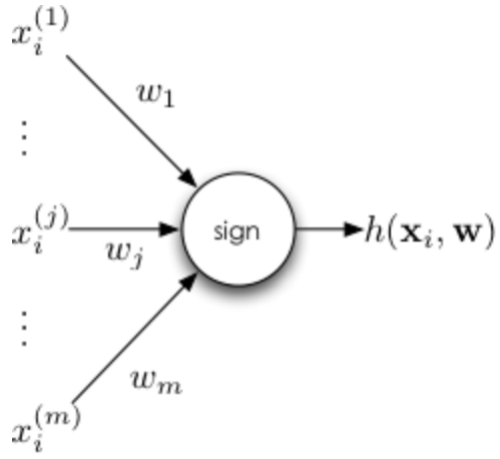
- Goal: design learning algorithm $\mathcal{A}$ such that its output f on iid training data S has low generalization error

Generalization error: $L_D(f) = \mathrm{E}_{(x,y)\sim D}\, \ell(y, f(x))$

Terminologies

- Model: the predictor $\hat{f}$
  - Often from a model class $\mathcal{F}$,
  - e.g. $\mathcal{F} = \{\text{decision trees}\}, \{\text{linear classifiers}\}$





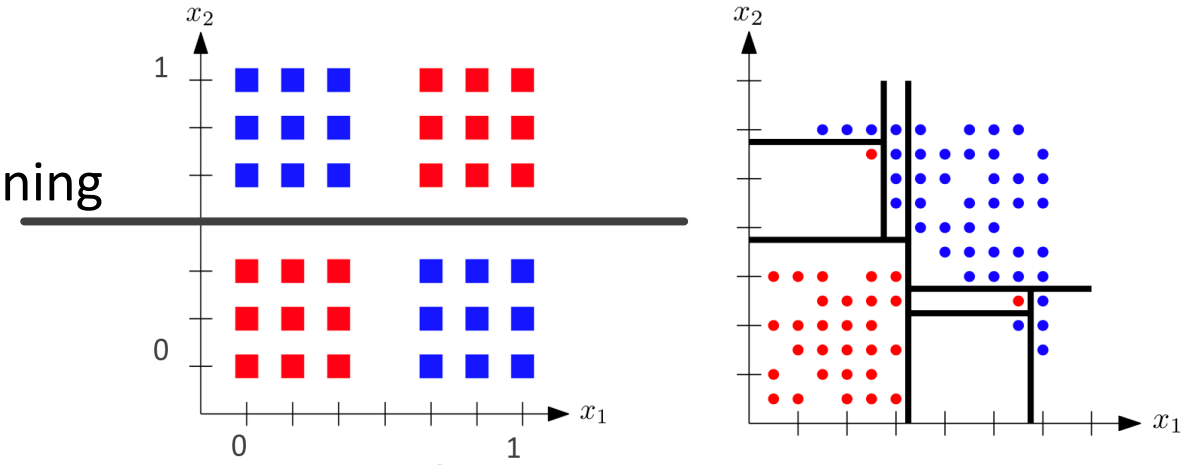
- Parameter: specifics of $\hat{f}$
  - E.g. for decision tree $\hat{f}$: tree structure, questions in nodes, labels in leaves
  - For linear classifier: linear coefficients

- Hyperparameter: specifics of learning algorithm $\mathcal{A}$
  - E.g. in DecisionTreeTrain, constrain to output tree of depth $\leq h$
  - Tuning hyperparameters often results in {over, under}-fitting

Hyperparameter tuning using validation set

- E.g. in decision tree training, how to choose tree depth $h \in \{1, \dots, H\}$?

- For each hyperparameter $h \in \{1, \dots, H\}$:
  - Train Tree_h using DecisionTreeTrain by constraining the tree depth to be h
- Choose one from $\mathrm{Tree}_1, \dots, \mathrm{Tree}_H$

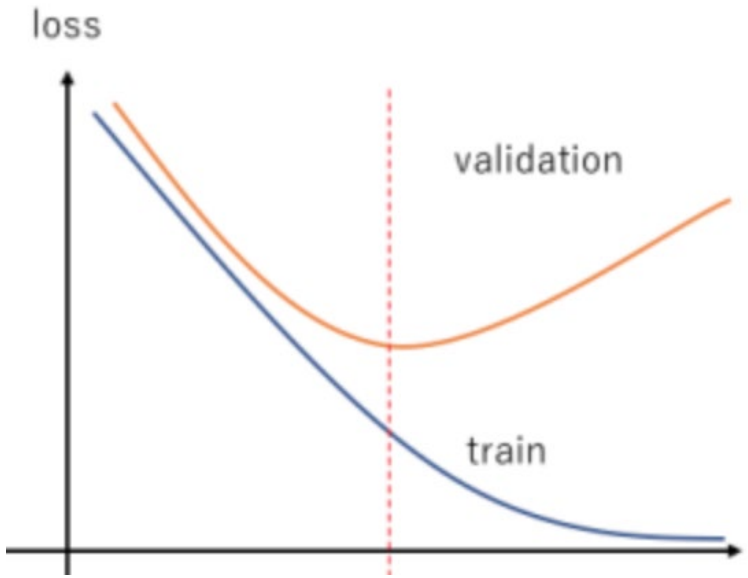


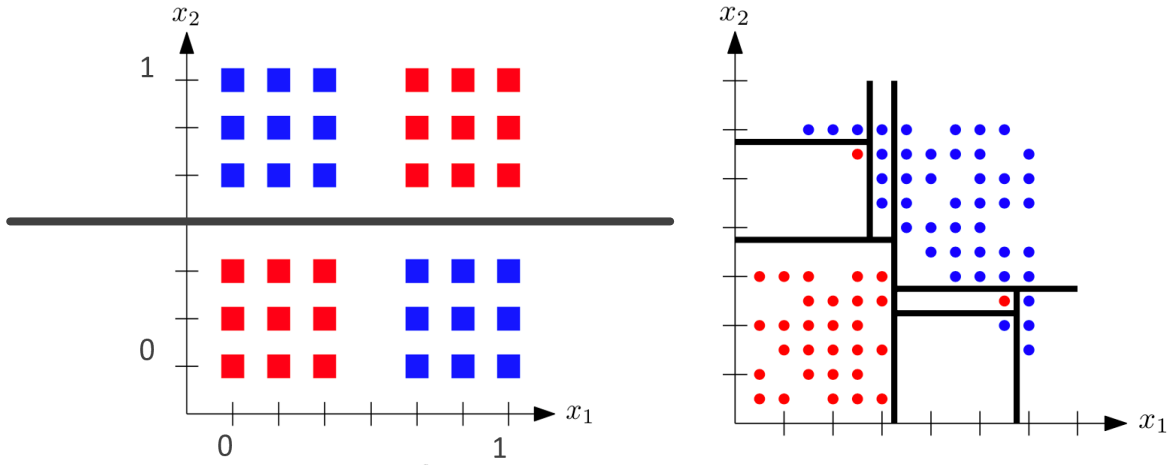
- Idea 1: choose Tree_h that minimizes training error
- Idea 2: choose Tree_h that minimizes test error
- Idea 3: further split training set to training set and validation set (development/hold-out set), (1) train Tree_h's using the (new) training set; (2) choose Tree_h that minimizes validation error

Training: 700 examples	Dev:100 examples	Test: 200 examples

Hyperparameter tuning using validation set

- E.g. in decision tree training, how to choose tree depth $h \in \{1, \dots, H\}$?





- Law of large numbers => Validation error closely approximates test error & generalization error

Model Selection / Assessment

Partition your data into Train-Validation-Test sets

Fit Each Model

Evaluate / Select Model

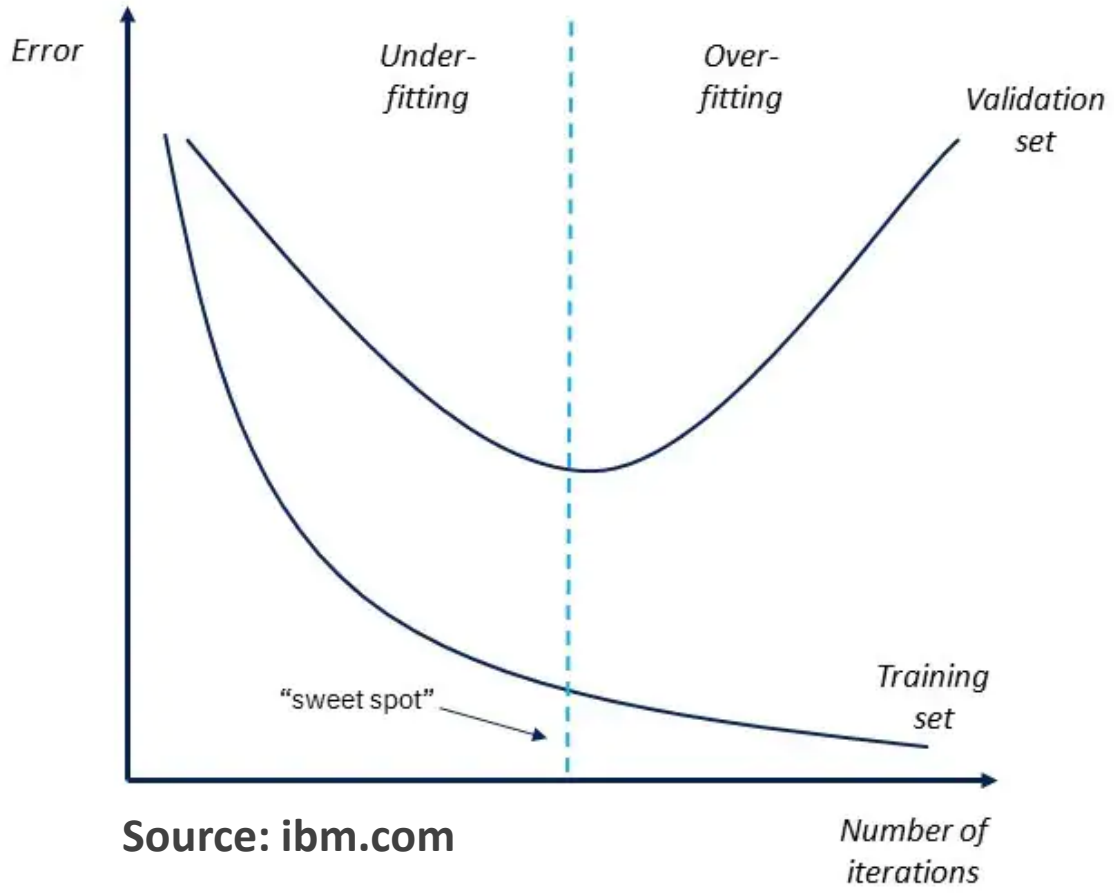
Assess Model

- Ideally, Test set is kept in a “vault” and only peek at it once model is selected
- Training-Validation-Test splits work if you have enough data (“data rich”)
- As a general rule 50% Training, 25% Validation, 25% Test (very loose rule)

Source: Hastie, Tibishrani, Freidman

Overfitting vs Underfitting

Underfitting performs poorly on *both* training and validation…



Source: ibm.com

…overfitting performs well on training but not on validation

KNN Model Selection / Assessment

1. Train a set of models K=1,…,K^{max} on training data:

$$\text{model}_{K=1}(\mathcal{D}^{\text{train}}),\ \ldots,\ \text{model}_{K=K^{\max}}(\mathcal{D}^{\text{train}})$$

2. Evaluate model accuracy on validation data:

$$\text{Error}(\text{model}_{K=1}, \mathcal{D}^{\text{val}}),\ \ldots,\ \text{Error}(\text{model}_{K=K^{\max}}, \mathcal{D}^{\text{val}})$$

3. Select model with lowest validation error:

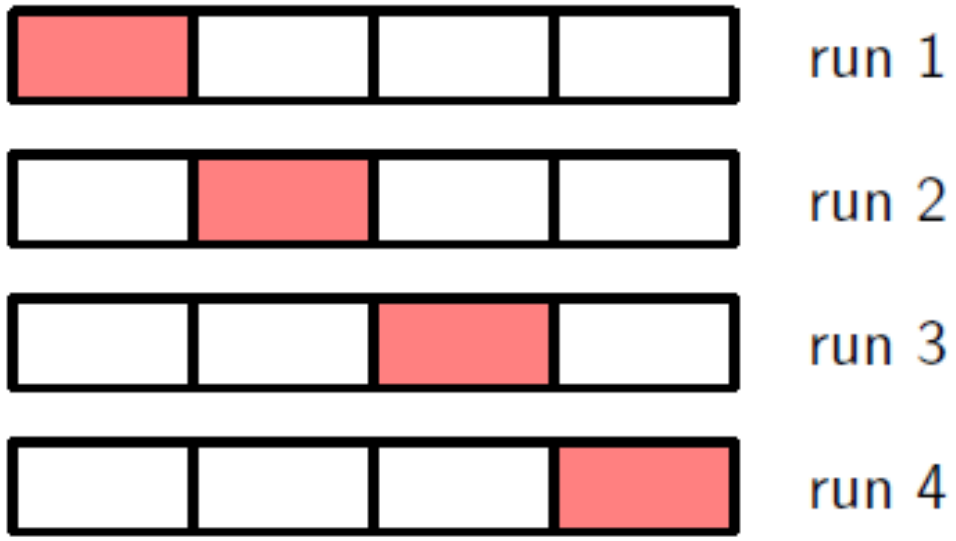
$$K^* = \arg\min_K\ \text{Error}(\text{model}_K, \mathcal{D}^{\text{val}})$$

3. Evaluate model error on test:

$$\text{Error}(\text{model}_{K^*}, \mathcal{D}^{\text{test}})$$

What are some drawbacks of this approach?

Cross-Validation



N-fold Cross Validation Partition training data into N “chunks” and for each run select one chunk to be validation data

For each run, fit to training data (N-1 chunks) and measure accuracy on validation set. Average model error across all runs.

Drawback Need a lot of training data to partition.

Source: Bishop, C. PRML

Hyperparameter tuning: cross-validation

- Main idea: split the training / validation data in multiple ways

- For hyperparameter $h \in \{1, \dots, H\}$
  - For $k \in \{1, \dots, K\}$
    - train $\hat{f}_k^h$ with $S \setminus \text{fold}_k$
    - measure error rate $e_{h,k}$ of $\hat{f}_k^h$ on fold_k
  - Compute the average error of the above: $\widehat{\text{err}}^h = \frac{1}{K}\sum_{k=1}^{K} e_{h,k}$
- Choose $\hat{h} = \arg\min_h \widehat{\text{err}}^h$
- Train $\hat{f}$ using S (all the training points) with hyperparameter $\hat{h}$

- $k = |S|$: leave one out cross validation (LOOCV)

An example real-world machine learning pipeline

- Any step can go wrong
  - E.g. data collection, data representation

- Debugging pipeline: run oracle experiments
  - Assuming the downstream tasks are perfectly done, is this step achieving what we want?

- General suggestions:
  - Build the stupidest thing that could possibly work
  - Decide whether / where to fix it

1	real world goal	increase revenue
2	real world mechanism	better ad display
3	learning problem	classify click-through
4	data collection	interaction w/ current system
5	collected data	query, ad, click
6	data representation	bow^2, $\pm$ click
7	select model family	decision trees, depth 20
8	select training data	subset from april'16
9	train model & hyperparams	final decision tree
10	predict on test data	subset from may'16
11	evaluate error	zero/one loss for $\pm$ click
12	deploy!	(hope we achieve our goal)

Next lecture (8/31)

- Geometric view of machine learning; nearest neighbor methods
- Assigned reading: CIML Chap. 3 (Geometry and Nearest Neighbors)
- HW1 will be assigned