CSCI 544: Applied Natural Language Processing

Text Classification

Xuezhe Ma (Max)

USC University of Southern California

Logistical Notes

- HW1 will be released today: start working early
- NLTK

Course Organization

NLP Tasks

Models	Text Classification	Sequence Labeling	Syntactic Parsing	Seq2seq & MT	Language Modeling & Generation
	Naive Bayes Logistic Regression SVM …	HMMs MEMMs CRFs	Tree-structured Models/Algorithms	……	……
	Word Embedding & Neural Networks	RNN LSTM	Neural Structured Networks	Attention & Transformers	Generative Models & LLMs

Classification

- **Categorizing instances of data into “classes”, where class members share some notion of similarity.**

X

{Cat, Dog, Finch, Owl}

Y

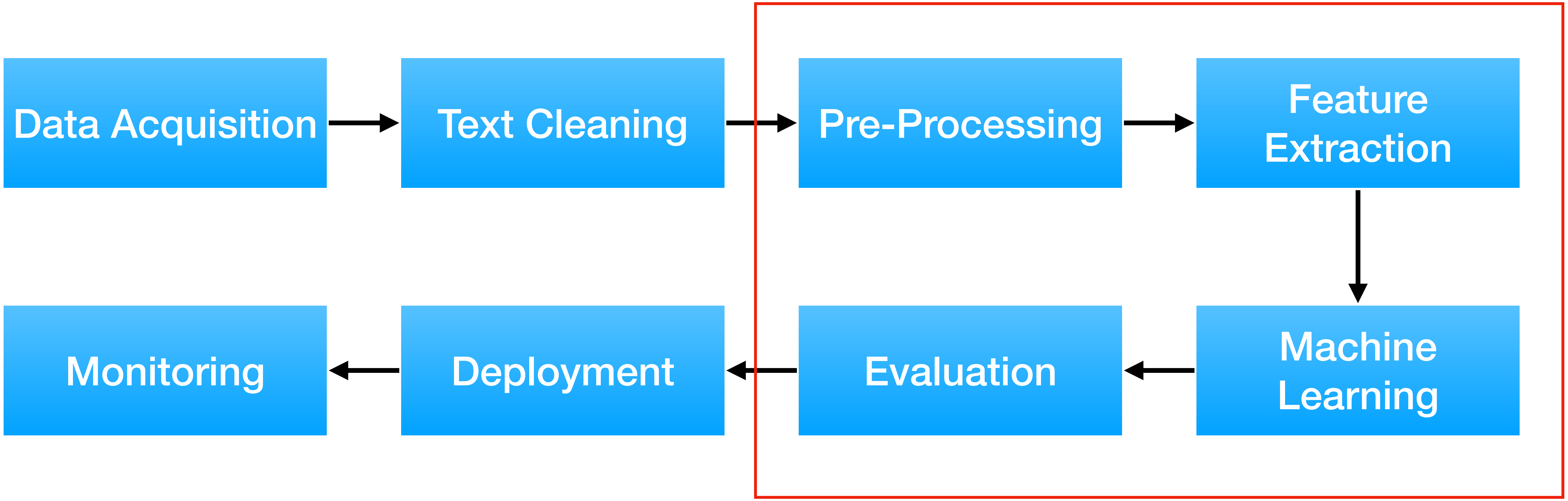
I don’t like scary movies, but this movie is funny at the same time and that is why I liked it.

X

{Positive, Negative}

Y

Recap: NLP System Pipeline



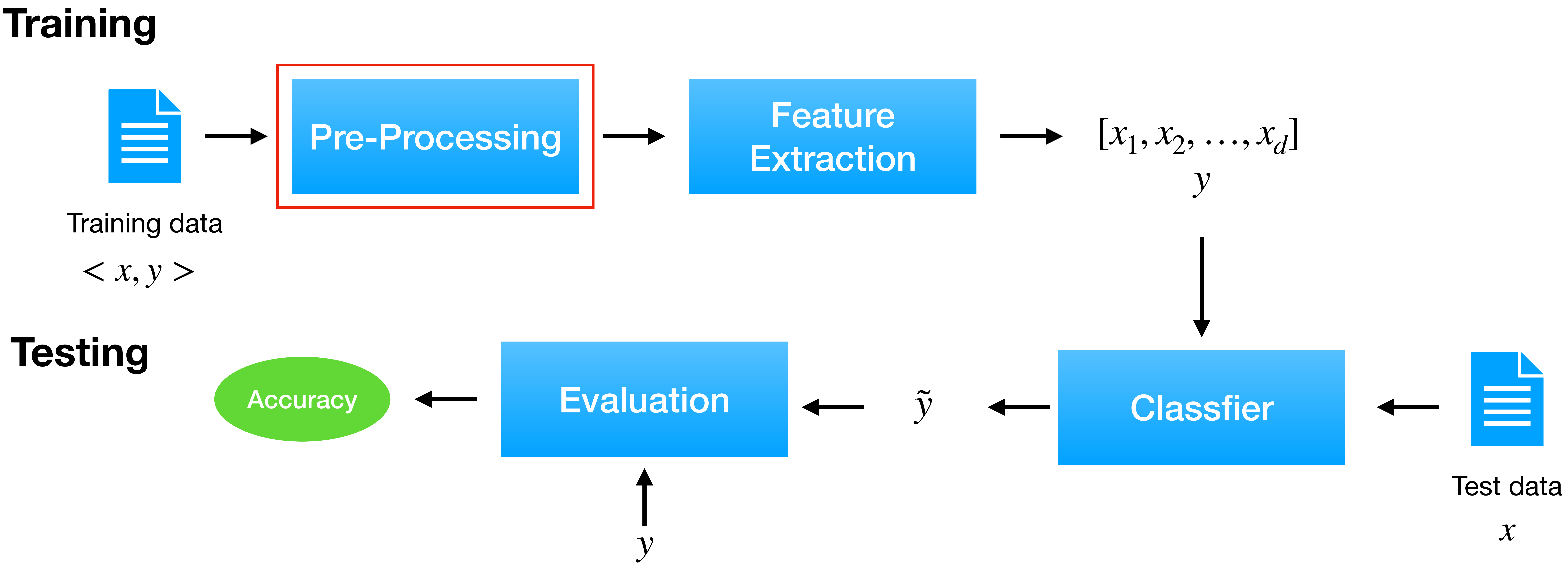
Outline

- **A Concrete Step-by-Step Example**
  - Text Classification, e.g., sentiment analysis
- **Preprocessing**
  - Tokenization
  - Other optional methods
- **Feature Extraction**
  - Vocabulary Creation
  - Feature Representations:
    - Bag of Words (BoW)
    - TF-IDF
- **Classification Algorithms**
  - Naive Bayes Classifier
  - Linear Classifier
- **Evaluation**
  - Metrics

I don't like scary movies, but this movie is funny at the same time and that is why I liked it.

{Positive, Negative}

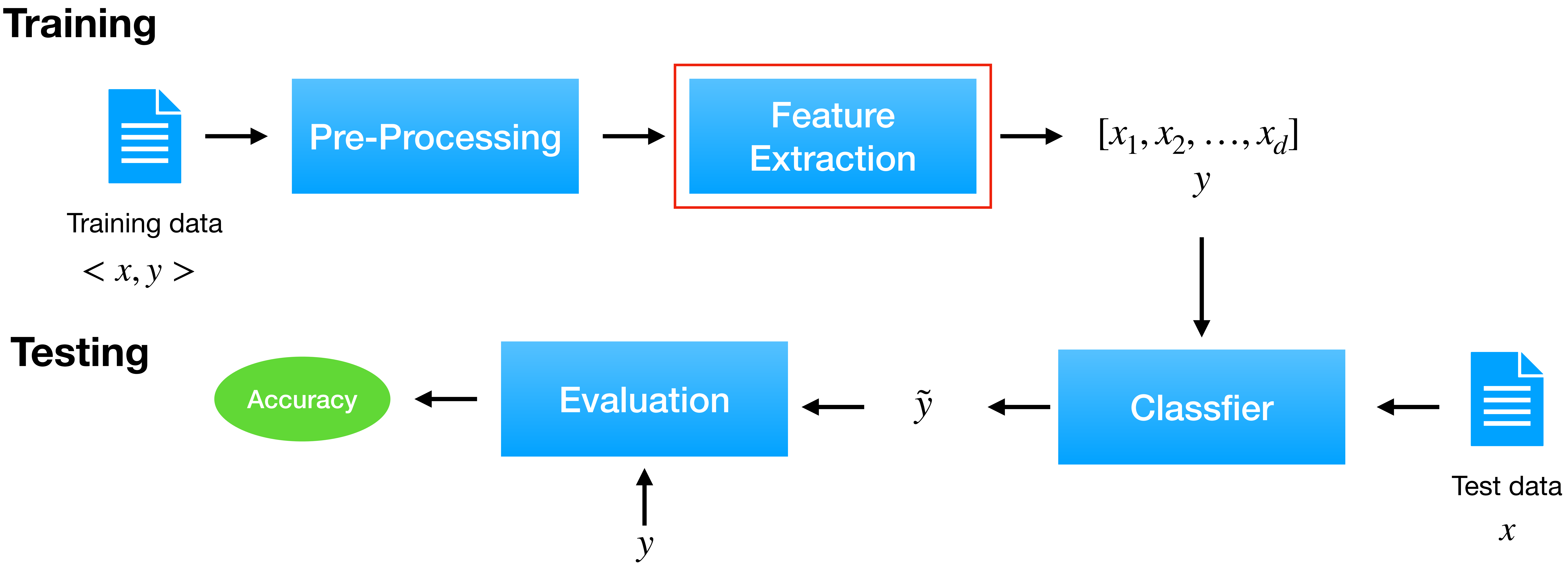
Pipeline


Training
Pre-Processing
Feature Extraction
$[x_1, x_2, \ldots, x_d]$
y
Training data
$< x, y >$
Testing
Accuracy
Evaluation
$\tilde{y}$
Classfier
y
Test data
x

Preprocessing

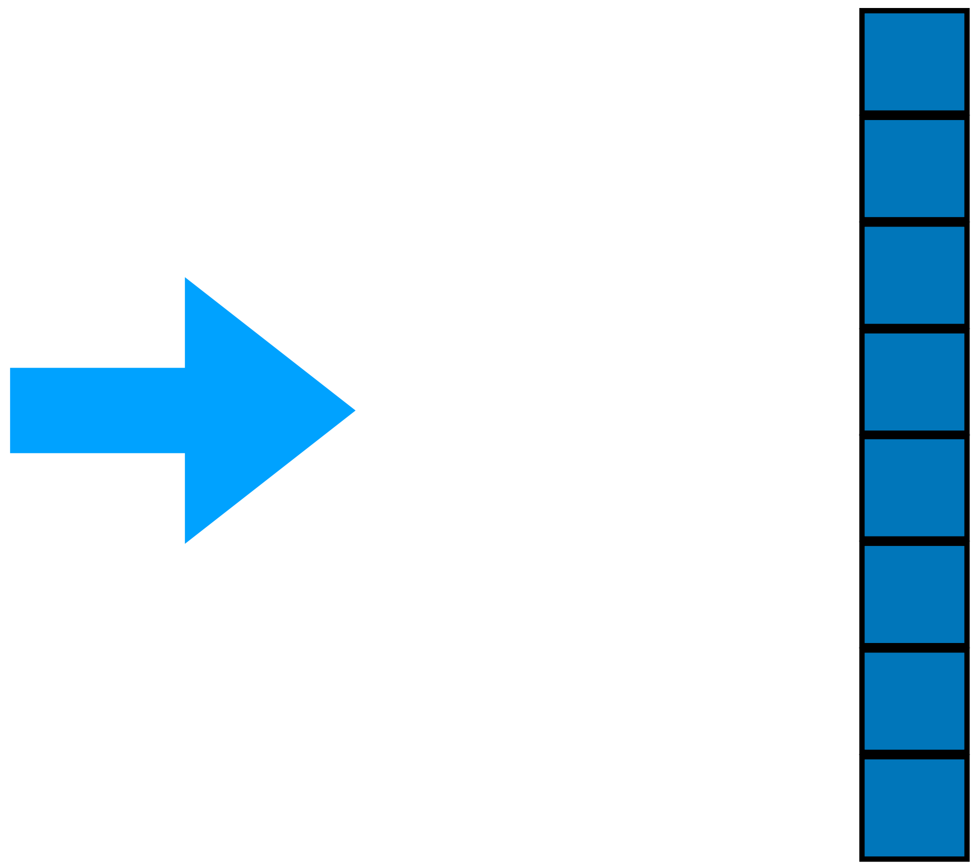
- **Tokenization:** splitting the text into units for processing
  - Removing extra spaces
  - Removing unhelpful tokens, e.g., external URL links
  - Removing unhelpful characters, e.g., non-alphabetical characters
  - Splitting punctuations: Happy New Year! -> Happy New Year !
- **Optional operations**
  - [optional] Contracting and standardizing: won’t -> will not
  - [optional] Converting capital letters to lowercase: New York -> new york
  - [optional] Stemming or Lemmatization: tokenization -> token

Pipeline


Training
Pre-Processing
Feature Extraction
$[x_1, x_2, \ldots, x_d]$
y
Training data
$< x, y >$
Testing
Accuracy
Evaluation
$\tilde{y}$
Classfier
Test data
x
y

Feature Representations

I don't like scary movies, but this movie is funny at the same time and that is why I liked it.



Feature Vector

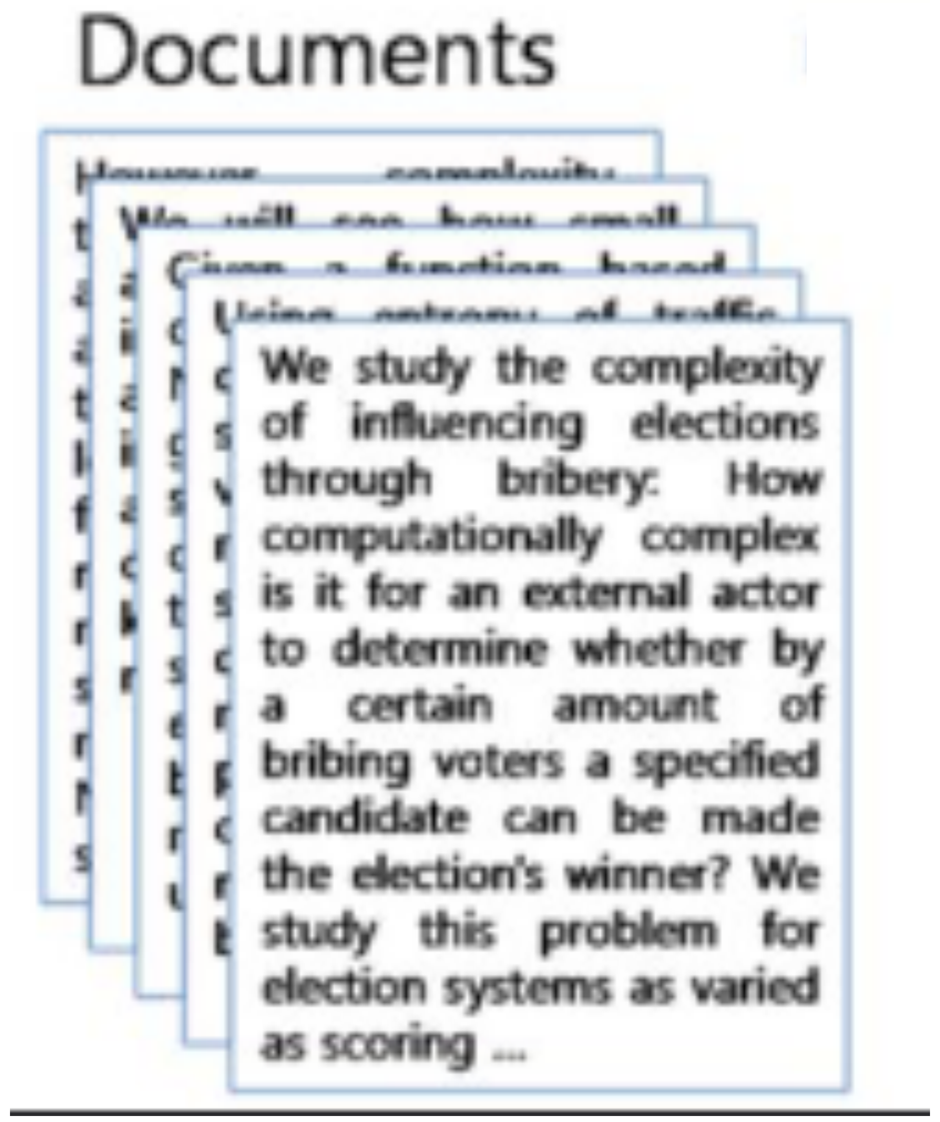
Representing semantic meaning of the data

Feature Extraction

- **Create a word vocabulary**
  - A dictionary of all the words we care about
    - Excluding stop words from dictionary as they are useless for sentiment analysis
  - Mapping each word to a word id: are -> 2
  - Discarding words not included in the vocabulary
- **Representing each data instance into a feature vector**
  - Bag of Words (BoW)
  - Term Frequency - Inverse Document Frequency (TF-IDF)

Feature Extraction

- **Create a word vocabulary**
  - A dictionary of all the words we care about
  - Mapping each word to a word index: are -> 2
  - Discarding words not included in the vocabulary



- Go through all the instances, calculating the counts of each word
- Select the top K (non-stop) words with the most counts
- Assign each word a unique ID

Bag of Words

- **With a word vocabulary of K words, BoW represents each doc/review X into a vector of integers**
- $X = [x_1, \dots, x_k], x_i \in \{0,1,2,\dots,\}$
- $x_i = j$ **indicates that word i appears j times in the doc/review X**

Vocab = [good, bad, nice, expensive, love]

" I love this shirt because it is nice and worm. The color also is nice and matches my skin tone" → **[0, 0, 2, 0, 1]**

Bag of Words

- **Limitations**
  - Insensitive to language structure: all word-order information has been discarded
  - Information in word dependencies is overlooked: new york vs new book
  - The resulting vectors are just word counts and are highly sparse
  - Dominant by common words
- **Why using BoW?**
  - Simple
  - Leads to acceptable performance in some applications

TF-IDF

- **Core Idea:** reflect how important a word is to a document in the dataset
- **Term Frequency (TF):** frequency of a term/word appears in an instance (document)

$$tf_{i,d} = \frac{\#\textbf{word i in document } d}{\#\textbf{all words in document } d}$$

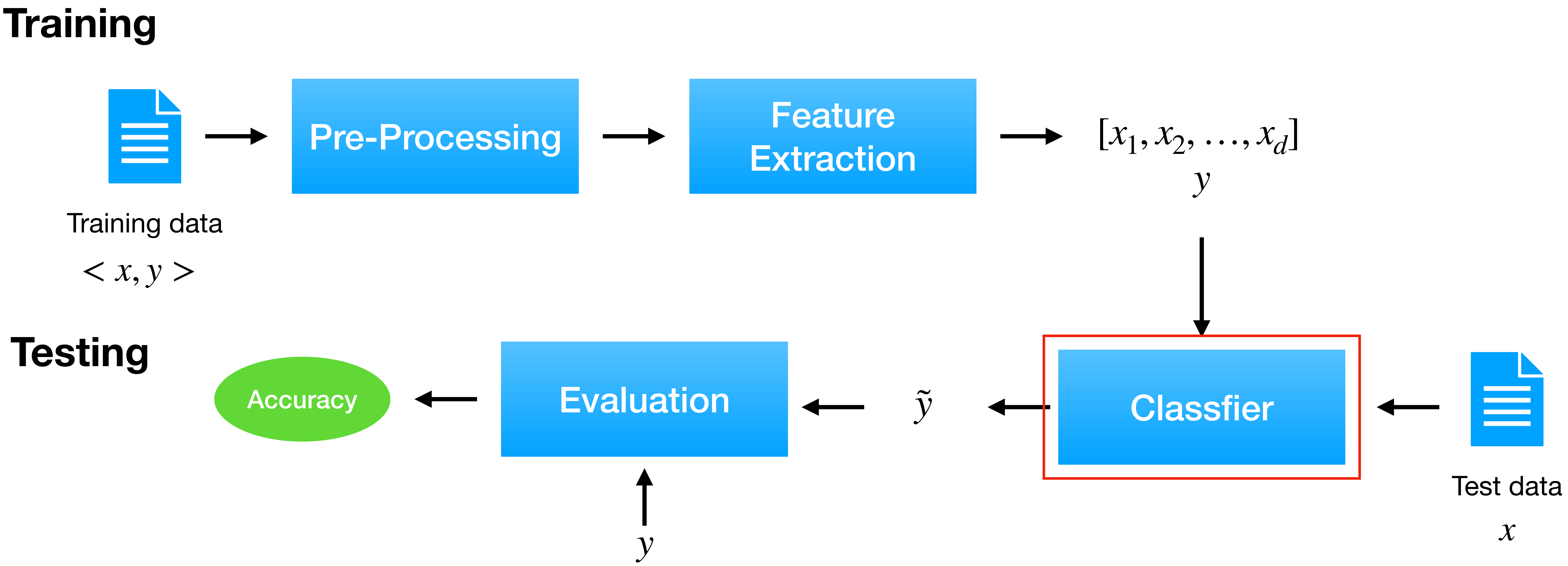
- **Inverse Document Frequency (IDF):** measuring how informative a word is

$$idf_i = \log \frac{\#\textbf{documents}}{\#\textbf{documents with word i}}$$

- **TF-IDF** $tf_{i,d} \times idf_i$

What are the TF-IDF scores for stop words?

Pipeline



Training
Training data
$< x, y >$
Pre-Processing
Feature Extraction
$[x_1, x_2, \ldots, x_d]$
y
Testing
Accuracy
Evaluation
y
$\tilde{y}$
Classfier
Test data
x

Classifier

- **Learning/Training: learning a predictive function from an training dataset**
  - $\tilde{Y} = f(X)$
- **Predicting/Testing: given the value of an input feature vector X, predicting the output class Y**
- **Statistical Classifier**
  - $P(Y|X)$
  - $\tilde{Y} = f(X) = \arg\max_{Y} P(Y|X)$
- **Non-Statistical Classifier**
  - Directly modeling the prediction function $f(X)$

Generative vs Discriminative

Generative

- Modeling the joint distribution: $P(X, Y)$
- Using Bayes' Rule to compute
  $$P(Y|X) = \frac{P(X, Y)}{P(X)}$$
- Returning the class that most likely to generate that instance
- Example: Naive Bayes

Discriminative

- Modeling the posterior distribution: $P(Y|X)$ or the prediction function $f(X)$
- Finding the exact function that minimizes classification errors on the training dataset
- Example: Linear classifier, Logistic Regression, Support Vector Machine (SVM), and Neural Networks (NNs)

Generative vs Discriminative

- **Discriminative classifiers are generally more effective, since they directly optimize the classification accuracy. But**
  - They are all sensitive to the choice of features, and in traditional ML these features are heuristically designed/extracted
  - Overfitting can happen if datasets are small
- **Generative classifiers directly model the joint probability, which is helpful when generating text is necessary. But**
  - Modeling the joint probability is a harder problem than classification if classification is our goal
  - They are more vulnerable w.r.t outliers/noises

Naive Bayes Classifier

- **Generative Classifier**

$$P(Y|X) = \frac{P(X,Y)}{P(X)} = \frac{P(X|Y)P(Y)}{P(X)}$$

$$\tilde{Y} = \arg\max P(Y|X)$$

$$= \arg\max_{Y} \frac{P(X|Y)P(Y)}{P(X)}$$

$$= \arg\max_{Y} P(X|Y)P(Y)$$

Naive Bayes Classifier

- **Challenges:**
  - How to compute the prior distribution $P(Y)$
  - How to compute the joint distribution $P(X_1, X_2, \ldots, X_k | Y)$

$$\tilde{Y} = \arg\max_{Y} P(X|Y)P(Y)$$
$$= \arg\max_{Y} P(X_1, X_2, \ldots, X_k | Y)P(Y)$$

Prior

$$P(Y=1) = \frac{\text{\#positive reviews}}{\text{\#all reviews}} \qquad P(Y=0) = \frac{\text{\#negative reviews}}{\text{\#all reviews}}$$

Independent Assumption

$$P(X_1, X_2, \ldots, X_k | Y) = \prod_{i=1}^{k} P(X_i | Y)$$

Naive Bayes Classifier

- **How to compute $P(X_i \mid Y)$?**

$$P(X_1, X_2, \ldots, X_k \mid Y) = \prod_{i=1}^{k} P(X_i \mid Y)$$

Bag of Words

$$P(X_i = m \mid Y = 1) = \frac{\text{\#positive documents with word i appears m times}}{\text{\#positive documents}}$$

TF-IDF?

Naive Bayes Classifier

- **Statistical Bias in Prior**
  - Balance the Prior
- **Sparsity**
  - $P(X_i = m \mid Y) = 0$, then the whole probability $P(X \mid Y) = 0$, no matter the other helpful evidence!
  - Smoothing: adding small non-zero probabilities to each term

Prior $$P(Y=1) = \frac{\textbf{\#positive reviews}}{\textbf{\#all reviews}} \qquad P(Y=0) = \frac{\textbf{\#negative reviews}}{\textbf{\#all reviews}}$$

Bag of Words

$$P(X_i = m \mid Y = 1) = \frac{\textbf{\#positive documents with word i appears m times}}{\textbf{\#positive documents}}$$

Discriminative Classifier

- **Statistical Classifier**
  - $P(Y|X)$
  - $\tilde{Y} = f(X) = \arg\max_{Y} P(Y|X)$
- **Non-Statistical Classifier**
  - Directly modeling the prediction function $f(X)$

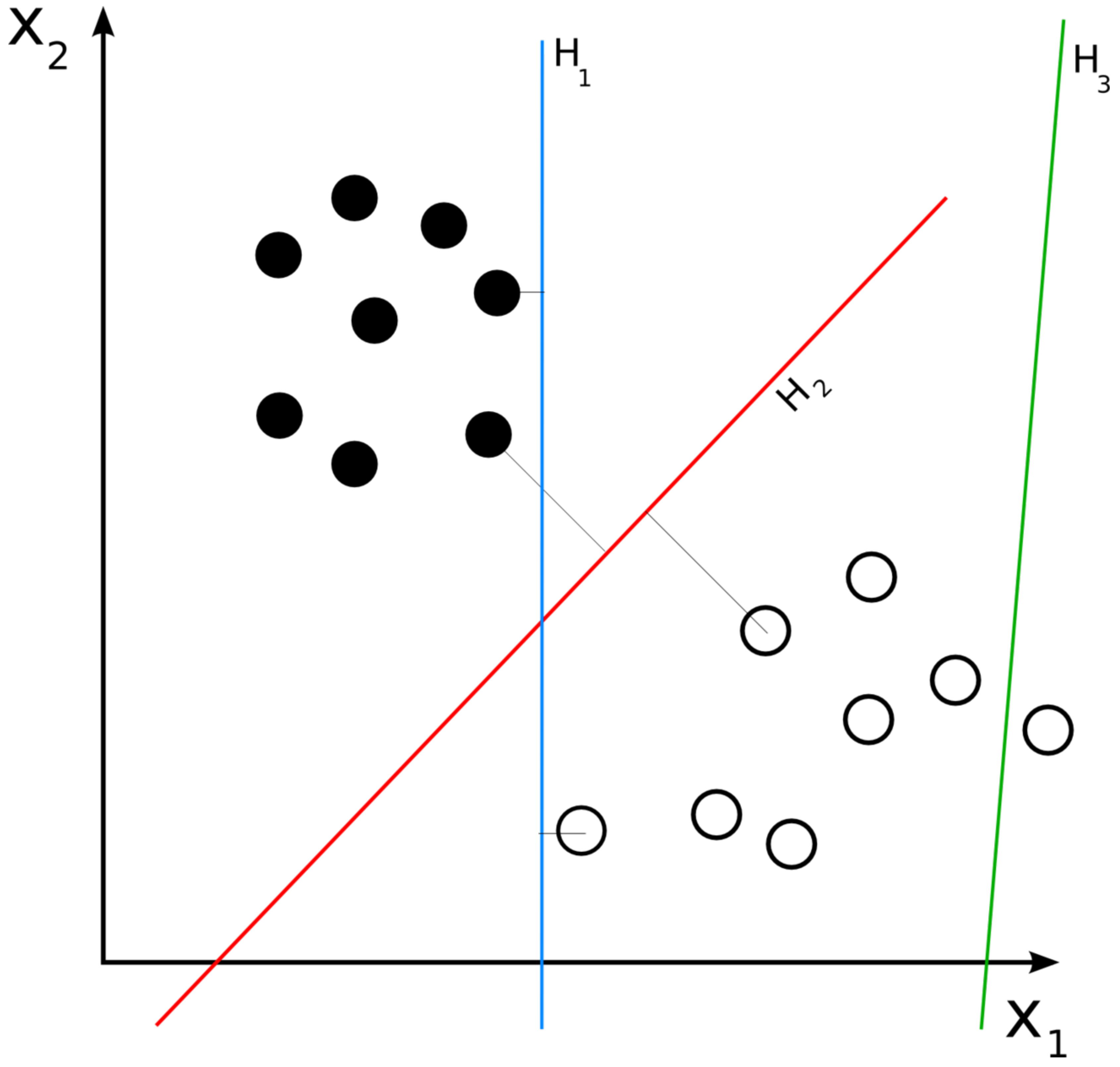
Linear Classifier

- **Discriminative Model with learnable parameters**
  - Assuming the data points are linearly separable in the feature space
  - The goal is to find a boundary

$$f(X) = W^T X = \sum_{i=1}^{k} w_i \cdot x_i$$

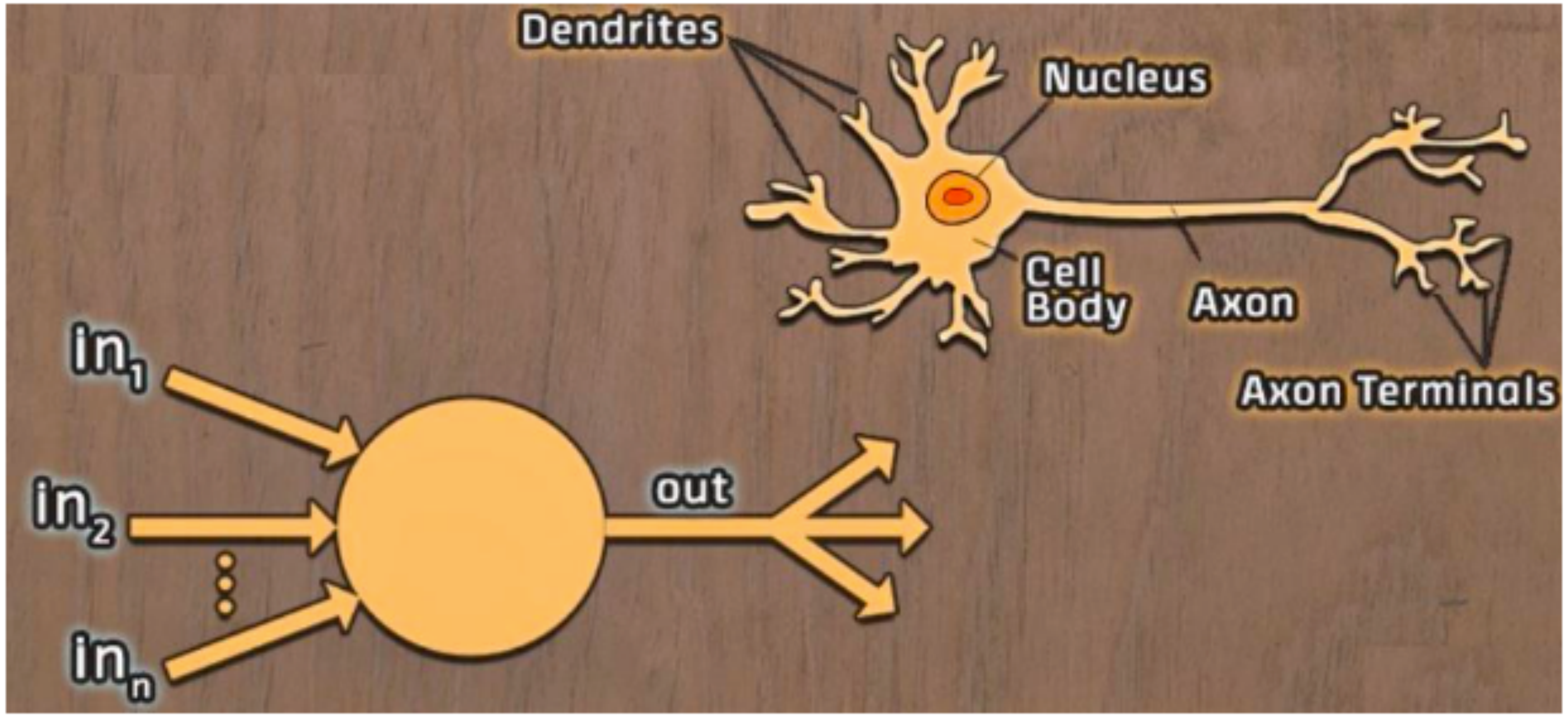
$$\hat{Y} = \begin{cases} 1, & W^T X > 0 \\ 0, & \text{Otherwise} \end{cases}$$

How to learn the parameter W?



Perceptron

- First neural network learning model in 1960's, Inspired by the nervous system
- Simple and Limited (single layer models)
- Basic concepts are similar for multi-layer neural networks so it is a good learning tool



Perceptron Learning Algorithm

- **First initialize the model parameters W with random values**
- **Go through the training data one by one and update the weights**
  - For $X, Y \in \{ < X^1, Y^1 > , < X^2, Y^2 > , \ldots, < X^n, Y^n > \}$:
    - $f(X) = W^T X$, $\hat{Y} = 1$ if $f(X) > 0$ else 0
    - $W \leftarrow W + \gamma(Y - \hat{Y})X$

Y	$\hat{Y}$	$Y - \hat{Y}$
0	0	0
0	1	-1
1	0	1
1	1	0

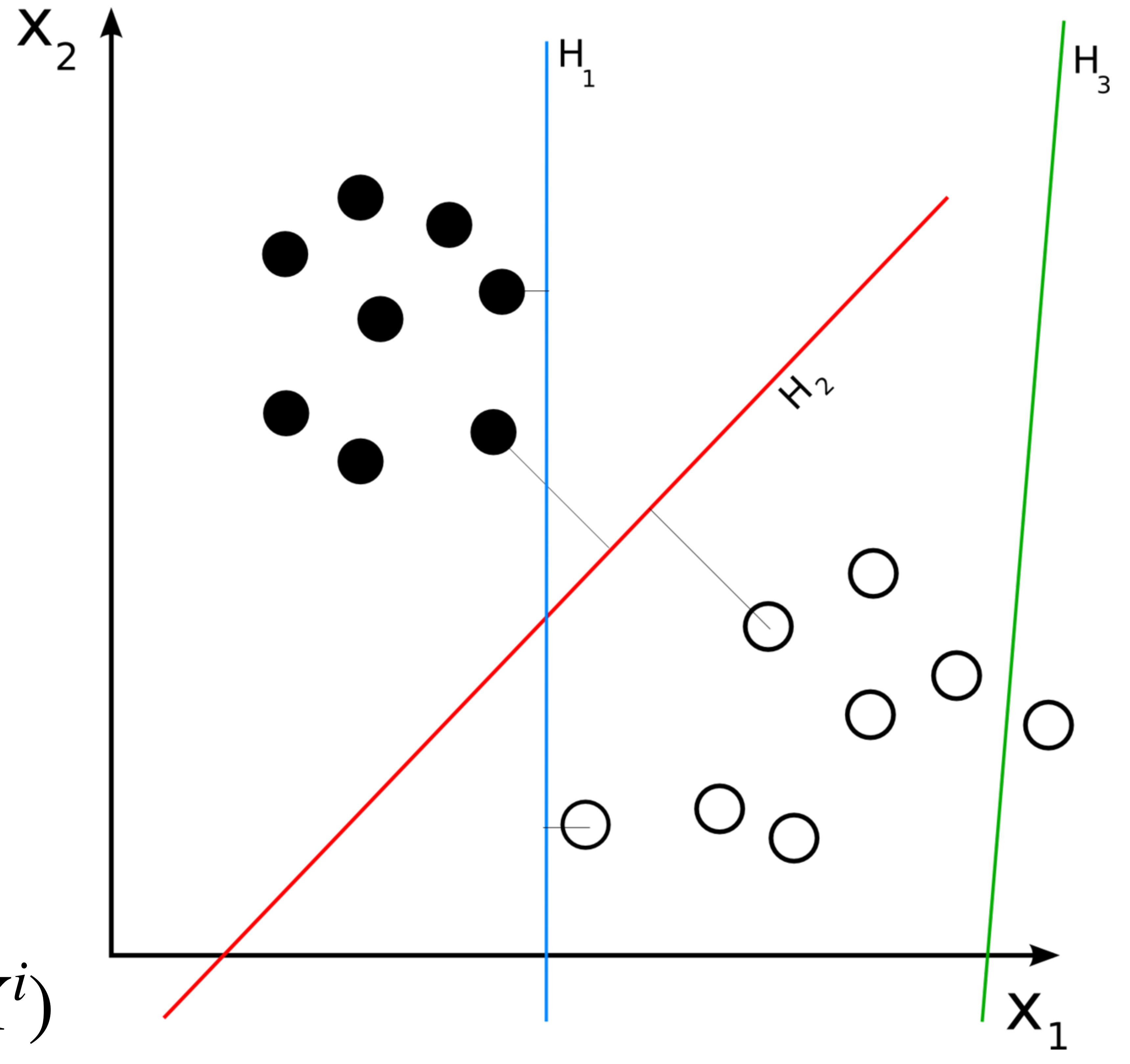
Only change parameters W if an error is there

Continue training until total training error ceases to improve

Perceptron Convergence Theorem:
guaranteed to find a solution in finite time if a solution exists

Optimal Boundary

- **Issues of Perceptron:**
  - What if the data points are NOT linearly separable?
  - Are the perceptron solutions optimal?



- **Model Training**
  - Given the training data

$$< X^1, Y^1 > , < X^2, Y^2 > , \ldots, < X^n, Y^n > ,$$

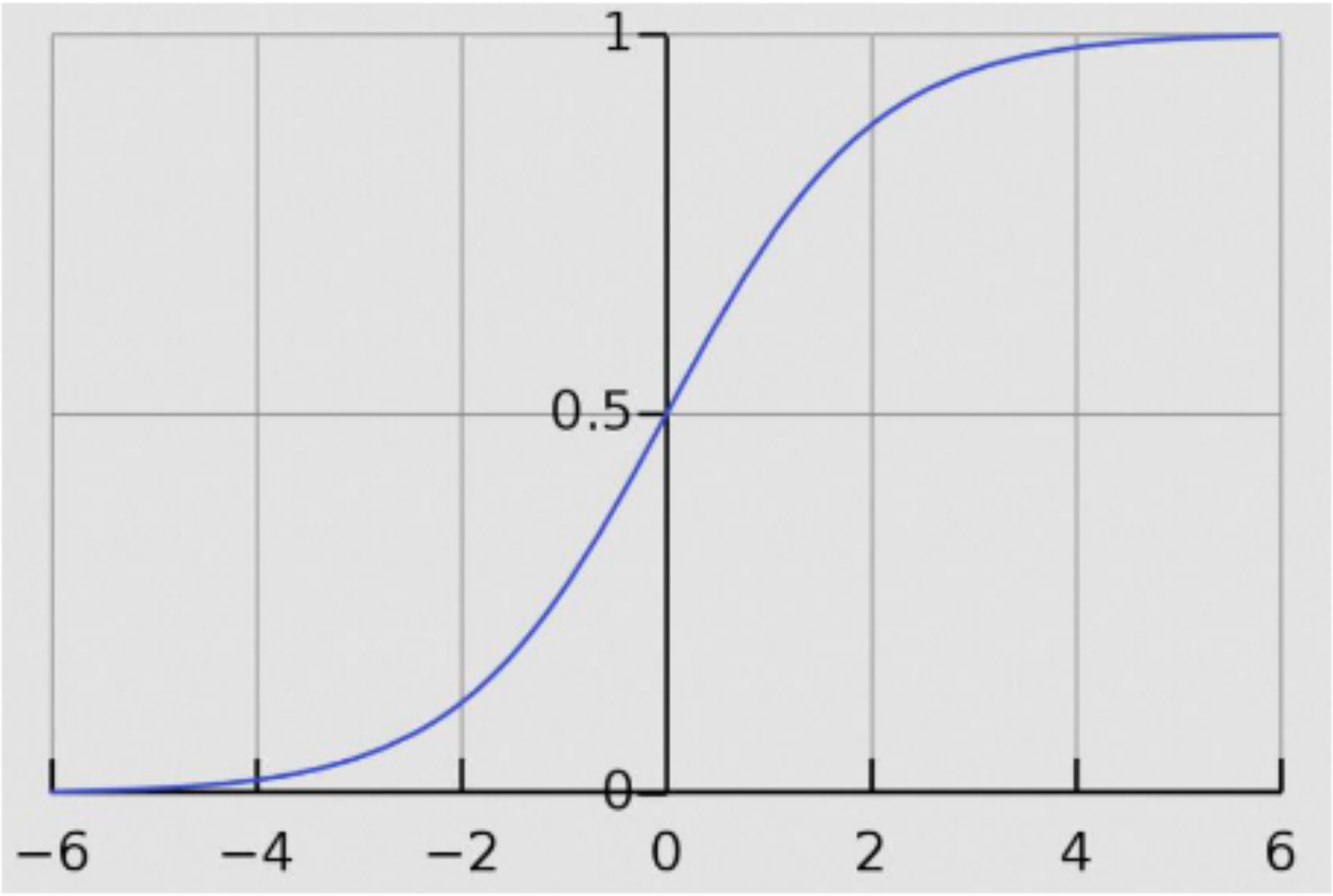
$$W^* = \arg\min_W \sum_{j=1}^{n} L(Y^i, W^T X^i)$$

L is the loss function

Different L results in different algorithms: logistic regression, SVM, …

Logistic Regression

- **Statistical Classifier**
  - $P(Y|X)$
- **Assuming that the likelihood function is a logistic function of the linear score**
  - $P(Y=1|X) = \dfrac{1}{1+e^{-W^T X}}$



- **Predictive Function**
  - $\tilde{Y} = f(X) = \arg\max_{Y} P(Y|X)$

$$= \begin{cases} 1, & W^T X > 0 \\ 0, & \text{Otherwise} \end{cases}$$

Logistic Regression: Learning Algorithm

- **Maximum Log-Likelihood**
  - Given the training data $< X^1, Y^1 >, < X^2, Y^2 >, \ldots, < X^n, Y^n >$,
  - Find W that maximizes the likelihood on training data
  - $$W^* = \arg\max_W \sum_{i=1}^{n} \log P(Y^i | X^i)$$
  - where $P(Y = 1 | X) = \dfrac{1}{1 + e^{-W^T X}}$
- **No closed-form solution**
  - Using numerical optimization methods, e.g, gradient descent or Newton method
  - Convex function: unique global optimal solution of W^*

Support Vector Machines (SVMs)

- **The margin of a linear classifier as the width that the boundary could be increased by before hitting a datapoint**
- **SVMs: maximize the margin to find the optimal boundary**
  - Note: $Y \in \{1, -1\}$

$$\min W^T W,\ s.t.$$

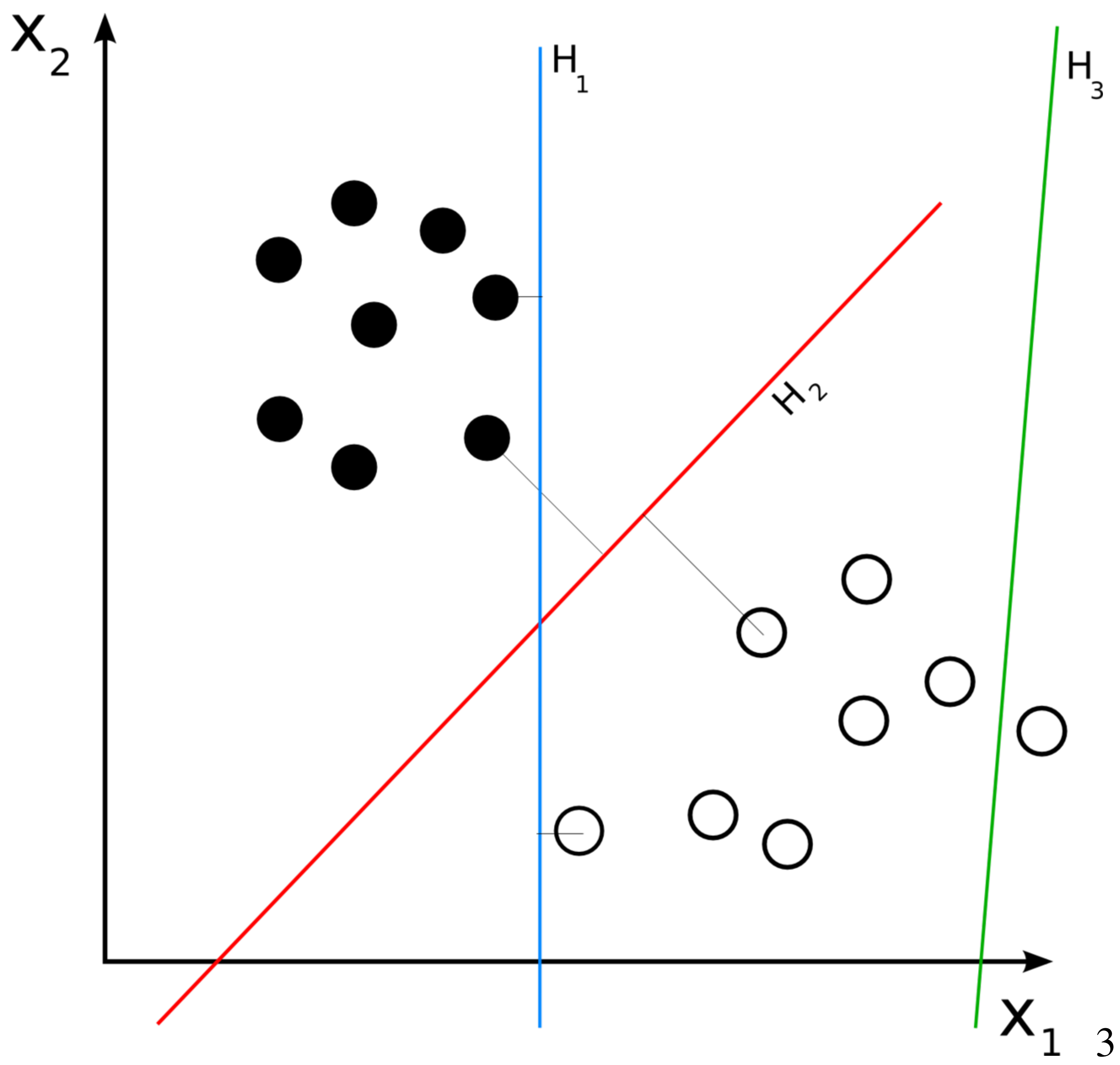
$$W^T X^i \cdot Y^i \geq 1, \forall \{ < X^i, Y^i > \}$$

- **How to manage the case where the training data are not linearly separable?**
  - Incorporating soft constraints

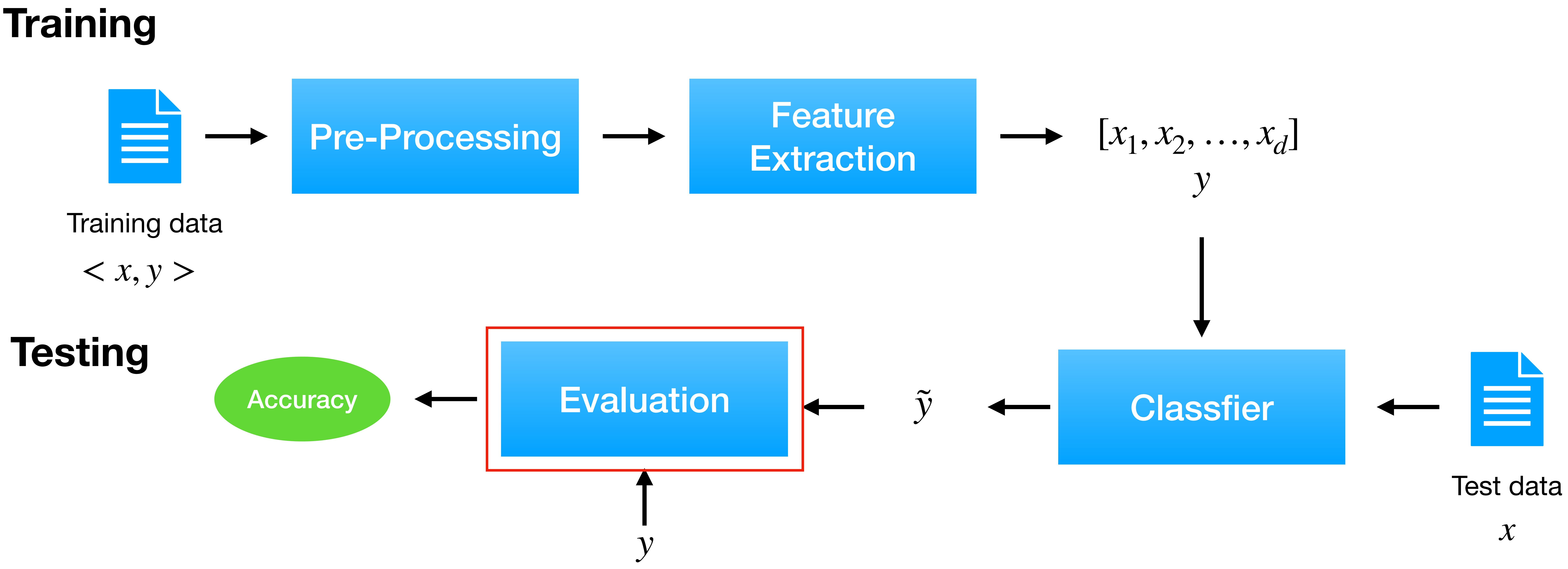
$$\min W^T W + c \sum_{j=1}^{n} \epsilon_j,\ s.t.$$

$$W^T X^i \cdot Y^i >= 1 - \epsilon_i, \forall \{ < X^i, Y^i > \}$$

$$0 < \epsilon_i < 1$$



Pipeline


Training
Pre-Processing
Feature Extraction
$[x_1, x_2, \ldots, x_d]$
y
Training data
$< x, y >$
Testing
Accuracy
Evaluation
$\tilde{y}$
Classfier
y
Test data
x

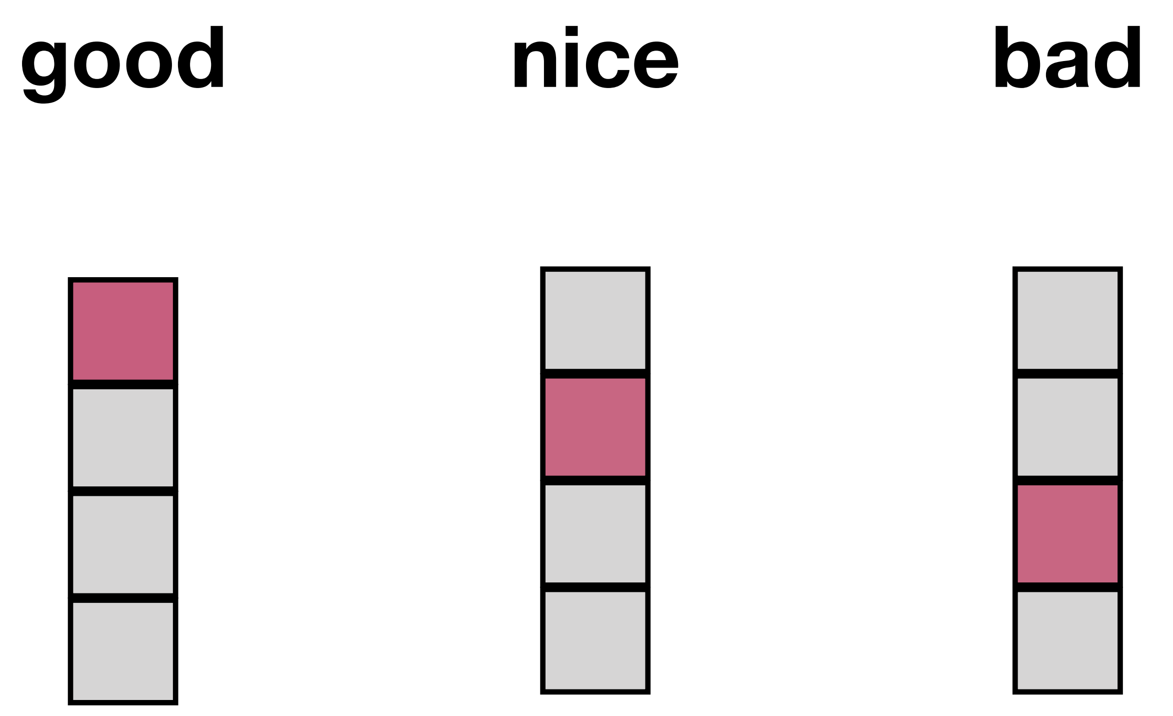
Model Evaluation

- **We randomly split the annotated dataset into three subsets**
  - Training set: learning model parameters
  - Validation set: selecting hyper-parameters
    - e.g. learning rate γ in perceptron; c in SVMs etc.
  - Test set: evaluating how well the learned model generalizes on unseen data
- **Evaluation Metrics**
  - **Accuracy**: proportion of correctly classified items
  - Sensitive w.r.t imbalance (1% positives vs. 99% negatives)
  - **Precision**, **Recall** and **F1-score**
  - **Precision:** $\dfrac{\text{\#True Positives}}{\text{\#True Positives} + \text{\#False Positives}}$
  - **Recall:** $\dfrac{\text{\#True Positives}}{\text{\#True Positives} + \text{\#False Negatives}}$
  - **F1:** $\dfrac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$

		$\hat{Y}$ = 1	$\hat{Y}$ = 0
Y	1	True Positive	False Negative
	0	False Positive	True Negative

Problems of Traditional Text Classification

- **Great efforts on classification algorithms/models**
  - Assuming the features are given
- **Insufficient attention on feature representations**
  - Bag of words & TF-IDF
  - Only frequency information, not semantic meaning of each word
  - No contextual information



$$\mathrm{dist}(\textbf{good}, \textbf{nice}) = \mathrm{dist}(\textbf{good}, \textbf{bad})$$

Bat

hit with bat

Q&A