Technology and Application of Big Data

Qing LIAO(廖清)

School of Computer Science and Technology

HIT

Course Details

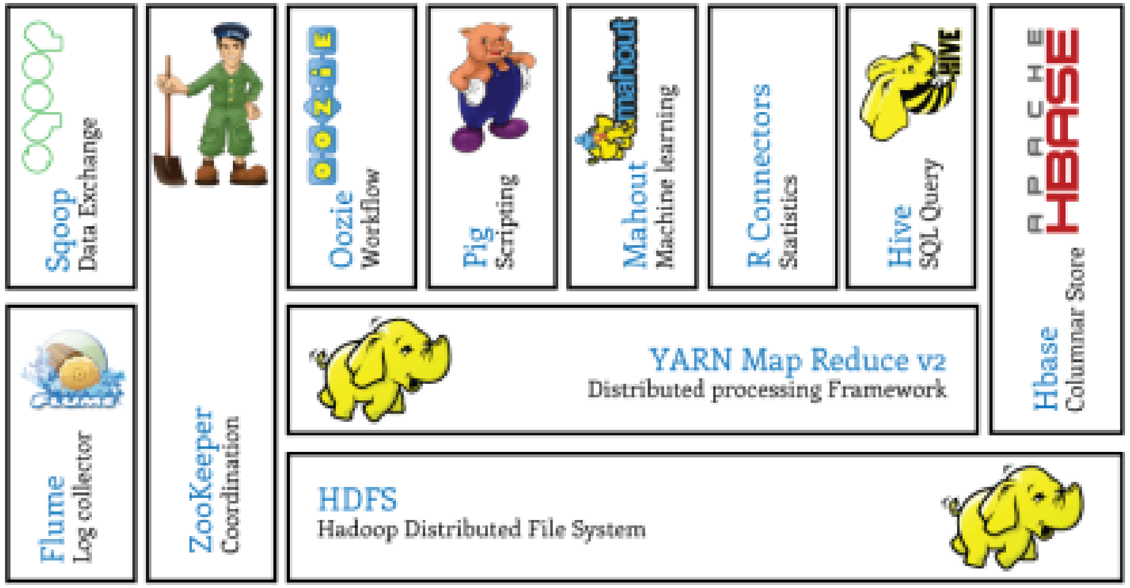
- Instructor:
  - Qing LIAO, liaoqing@hit.edu.cn
  - Rm. 303B, Building C
  - Office hours: by appointment
- Course web site:
  - liaoqing.me
- Reference books/materials:
  - Big data courses from University of California
  - Book: BIG DATA: A Revolution That Will Transform How We Live, Work, and Think
  - Papers
- Grading Scheme:
  - Paper Report 30%
  - Final Exam 70%
- Exam:
  - 21^{st} July(Friday), 14:00-16:00, A502

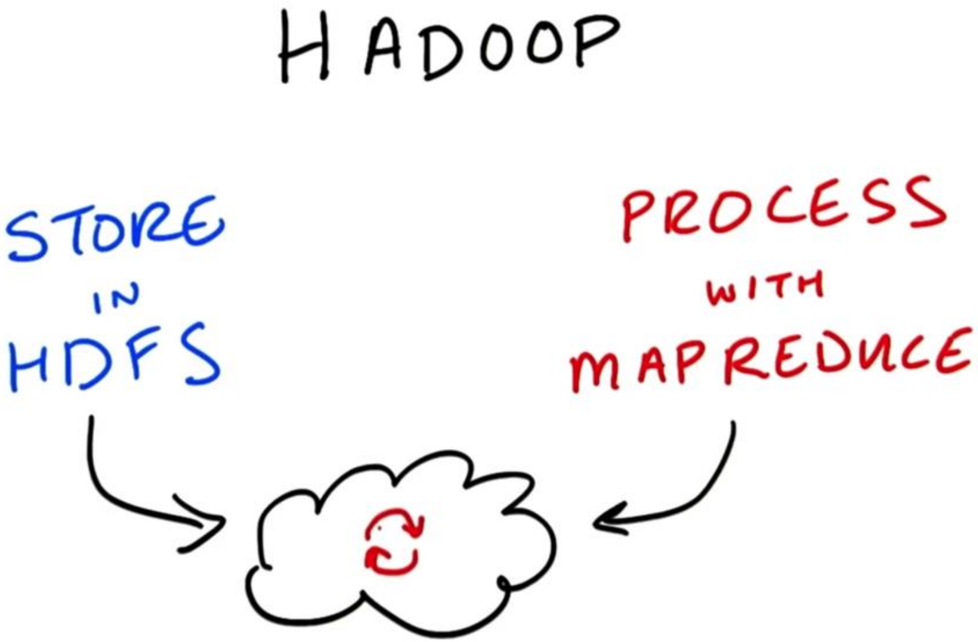
What You Learnt: Overview

- Topics:
  1) Introduction of Big Data
  2) Characterizes of Big Data
  3) How to Get Value from Big Data
  4) Technologies of Big Data
  5) Applications of Big Data
- Prerequisites
  - Statistics and Probability would help
    - But not necessary
  - Machine Learning would help
    - But not necessary

Previous Section

Hadoop Eco-System



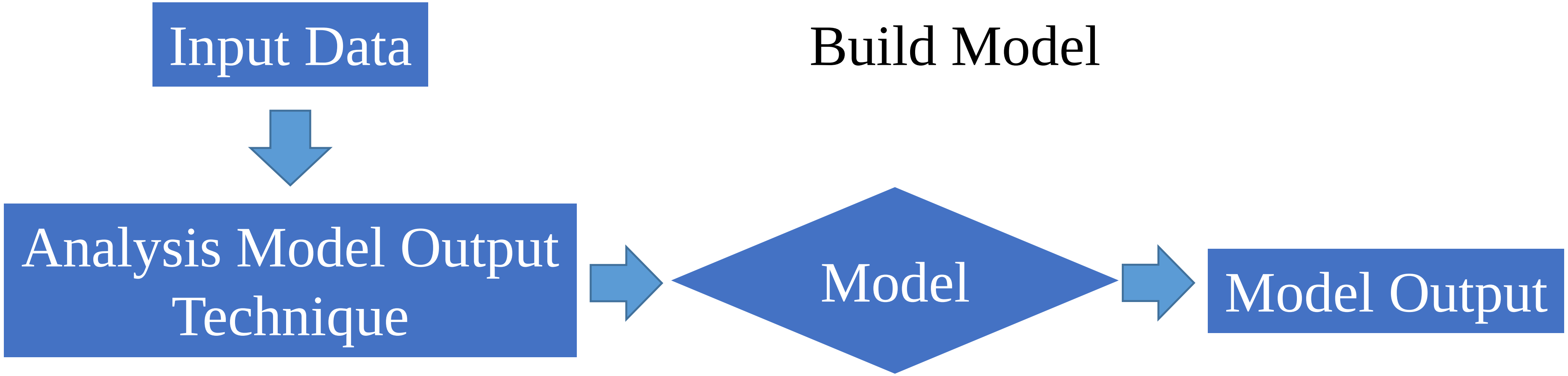


Previous Section

- TianHe(Milk Way) Supercomputer
- NO.1 in “The International Conference for High Performance Computing, Networking, Storage and Analysis(SC10)”
- 2010.11.18， USA

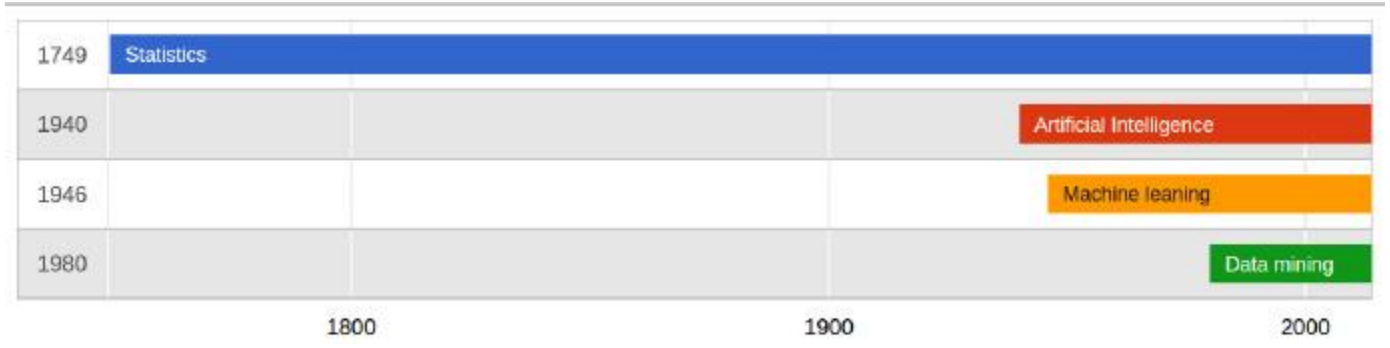
How to Get Value from Big Data

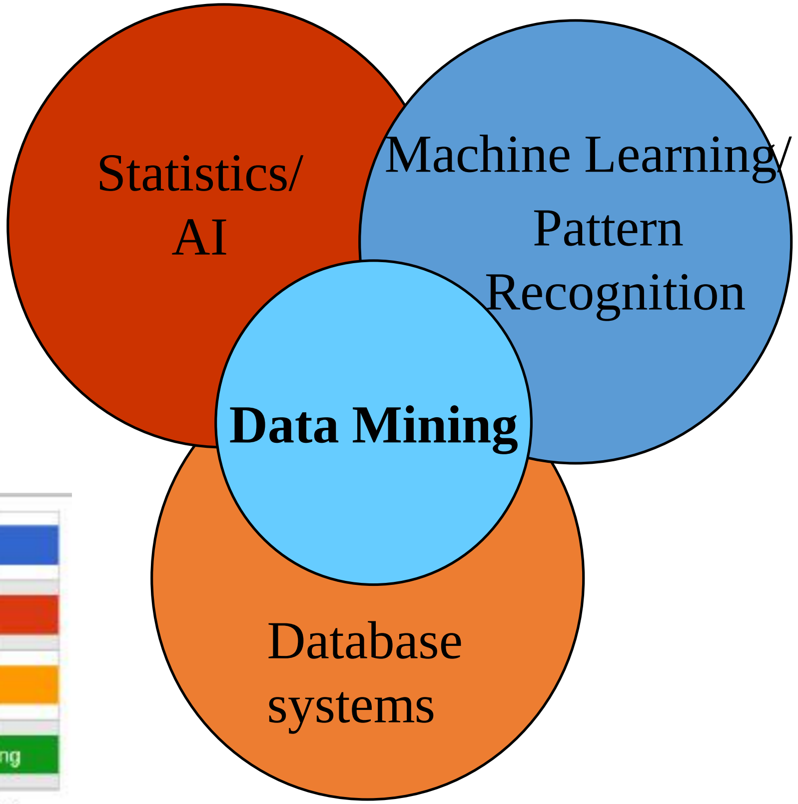
- Step 3: Analyze Data



Technologies of Big Data Analysis

- Artificial Intelligence/ Machine Learning
  - Neural Network
  - Deep Learning
- Data Mining
  - Classification
  - Clustering





Machine Learning & Data Mining

Computer Algorithm

Process of Converting
Data & Experience
Into Knowledge

Computer Model

Machine Learning & Data Mining

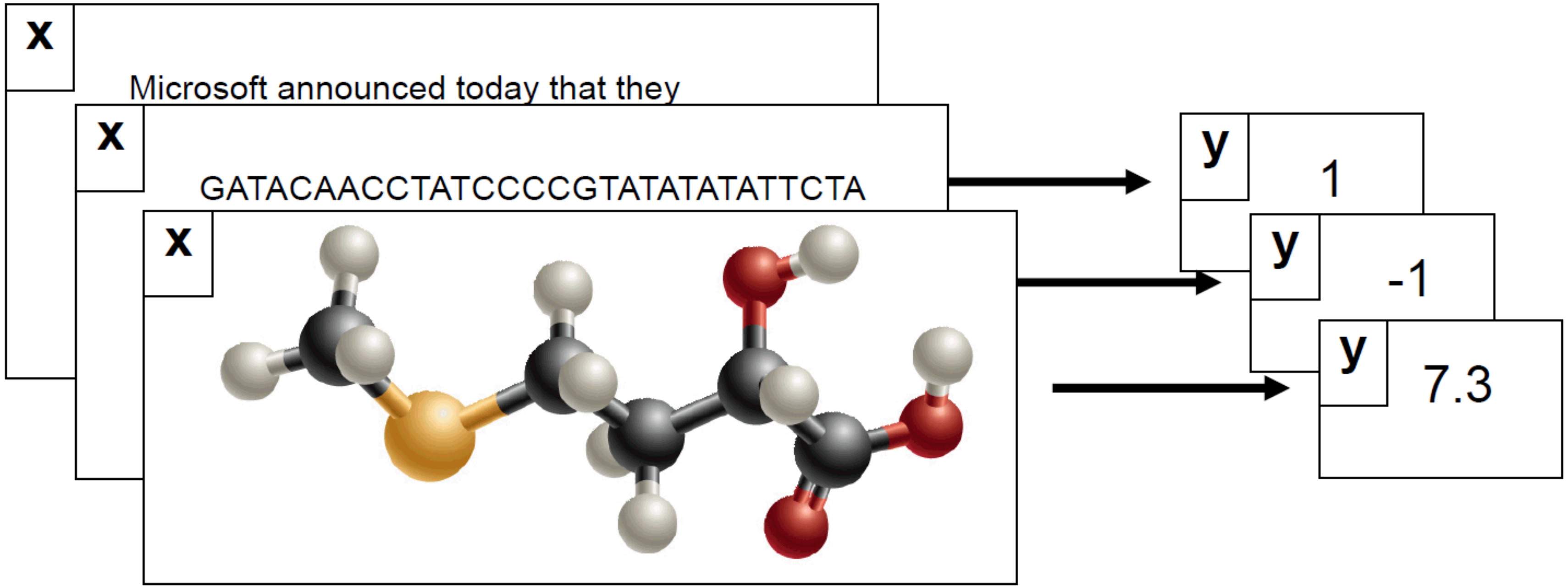
- ML focuses more on algorithms
  - Typically more rigorous
  - Also on analysis (learning theory)
- DM focuses more on knowledge extraction
  - Typically uses ML algorithms
  - Knowledge should be human-understandable

Supervised Learning

- Find function from input space X to output space Y

$$f : X \to Y$$

such that the prediction error is low.



Supervised Learning

Data: X

Target Signal: Y

Logistic Regression
Artificial Neural Nets
Random Forests
Etc…

f(x)≈y

Aside: Unsupervised Learning

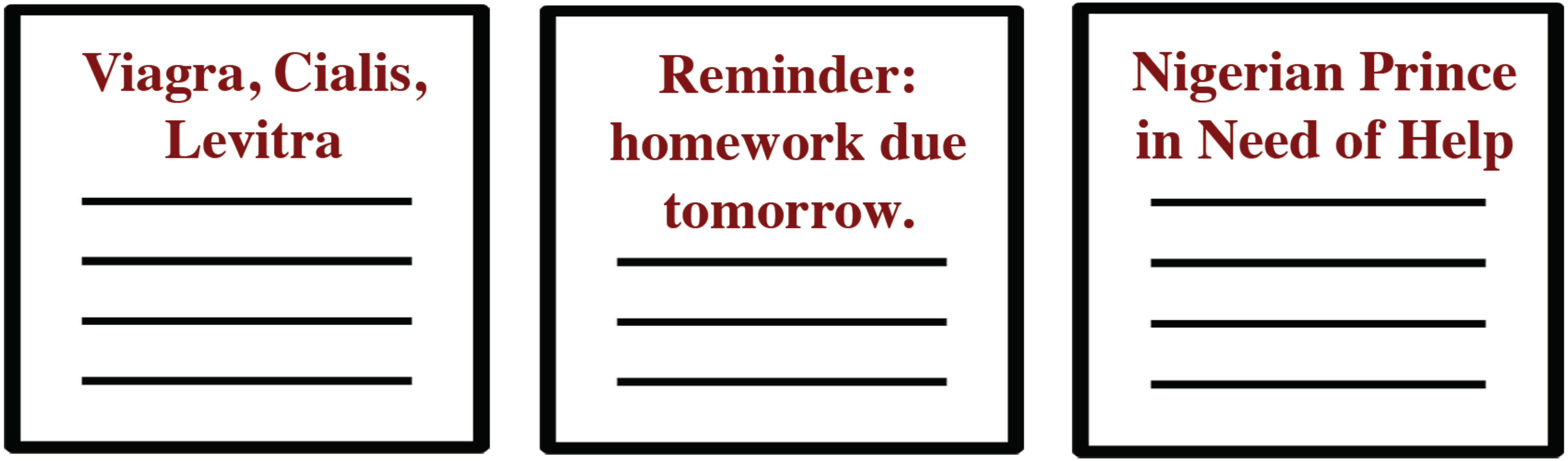
No supervised target!

Learning goal is usually to find low-dimensional “summary” or reconstruction.

More on this later in course.

Example: Spam Filtering

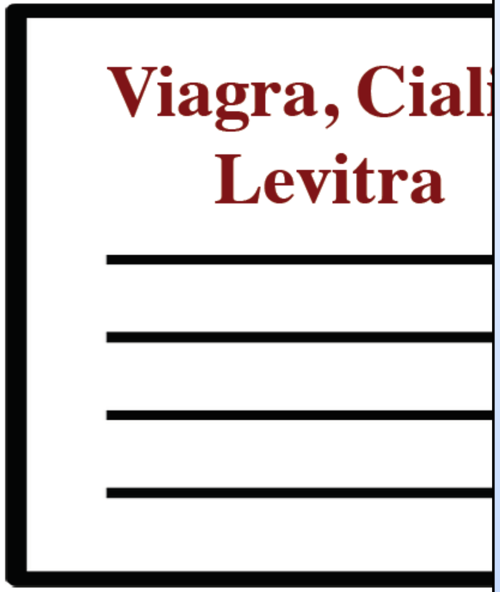
- Goal: write a program to filter spam.



SPAM! **NOT SPAM** **SPAM!**

Example: Spam Filtering

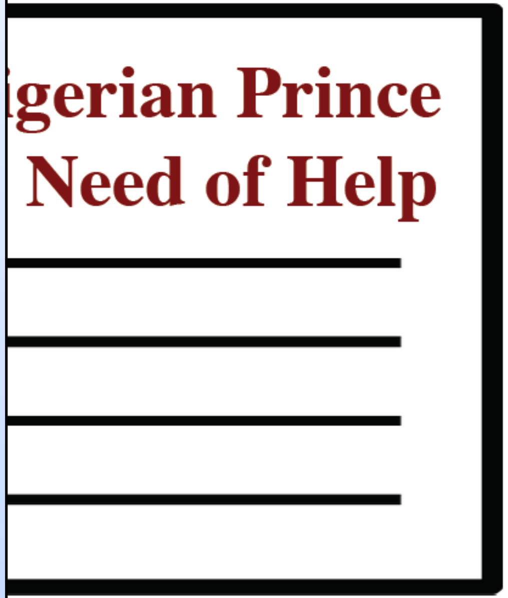
- Goal: write a program to filter spam.



SPAM!

```
FUNCTION SpamFilter(string document)
{
    IF("Viagra" in document)
        RETURN TRUE
    ELSE IF("NIGERIAN PRINCE" in document)
        RETURN TRUE
    ELSE IF("Homework" in document)
        RETURN FALSE
    ELSE
        RETURN FALSE
    END IF

}
```

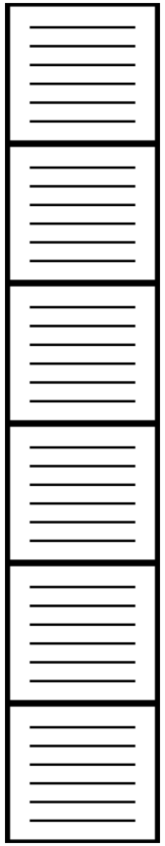


SPAM!

Why is Spam Filtering Hard?

- Easy for humans to recognize
- Hard for humans to write down algorithm
- Lots of IF statements!

Why is Spam Filtering Hard?



Training Set	Bag of Words
SPAM!	(0,0,0,1,1,1)
SPAM!	(1,0,0,1,0,0)
NOT SPAM	(1,0,1,0,1,0)
NOT SPAM	(0,1,1,0,1,0)
SPAM!	(1,0,1,1,0,1)
SPAM!	(1,0,0,0,0,1)

“Feature Vector”

One feature for each word in the Vocabulary

In practice 10k-1M

Linear Models

- Let x denote the bag-of-words for an email

  E.g., x = (1,1,0,0,1,1)

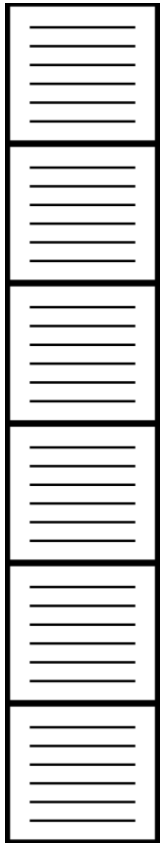
- Linear Classifier:

$$f(x|w,b) \; = \; sign(w^T x - \; b)$$
$$= \; sign(w_1 * x_1 + \; \ldots \; w_6 * x_6 - \; b)$$

Why is Spam Filtering Hard?

$$w = (1,0,0,1,0,1)$$
$$b = 1.5$$

Training Set

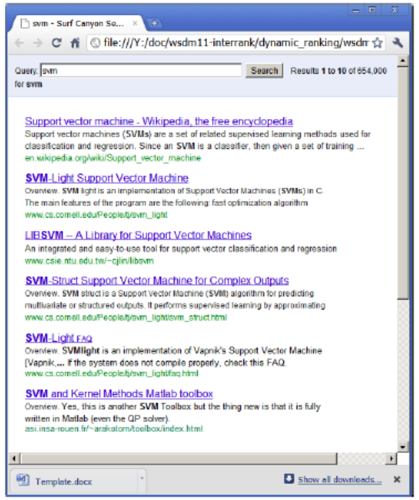


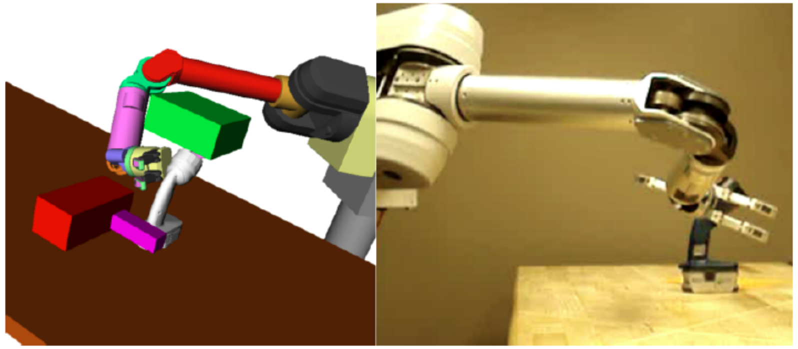
Training Set	Bag of Words	
SPAM!	(0,0,0,1,1,1)	$f(x \mid w, b) = +1$
SPAM!	(1,0,0,1,0,0)	$f(x \mid w, b) = +1$
NOT SPAM	(1,0,1,0,1,0)	$f(x \mid w, b) = -1$
NOT SPAM	(0,1,1,0,1,0)	$f(x \mid w, b) = -1$
SPAM!	(1,0,1,1,0,1)	$f(x \mid w, b) = +1$
SPAM!	(1,0,0,0,0,1)	$f(x \mid w, b) = +1$

$$f(x \mid w, b) = sign(w^T x - b) = sign(w_1 * x_1 + \ldots w_6 * x_6 - b)$$

Linear Models

- Workhorse of Machine Learning





- By end of this lecture, you'll learn 75% how to build basic linear model.

Why Does Machine Learning Work?

- Repeated patterns in the data
  - Typically in the features
  - E.g., “Nigerian Prince” is indicative of spam

- Machine learning will find those patterns
  - Linear model over features
  - E.g., high weight on the words “Nigerian Prince”

Supervised ML Problem & Classification

- **Regression**
  - Predict a real value or a probability
  - E.g., probability of being spam

$$f\ (x \mid w, b) = w^T x - b$$

- **Classification**
  - Predict which class an example belongs to
  - E.g., spam filtering example

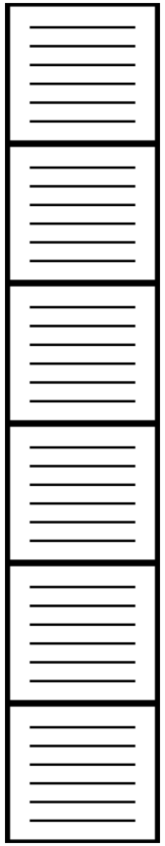
$$f\ (x \mid w, b) = sign(w^T x - b)$$

- **Highly inter-related**
  - Train on Regression => Use for Classification

Why is Spam Filtering Hard?

$$w = (1,0,0,1,0,1)$$
$$b = 1.5$$

Training Set



	Bag of Words	
SPAM!	(0,0,0,1,1,1)	$f(x\|w,b) = +1$
SPAM!	(1,0,0,1,0,0)	$f(x\|w,b) = +1$
NOT SPAM	(1,0,1,0,1,0)	$f(x\|w,b) = -1$
NOT SPAM	(0,1,1,0,1,0)	$f(x\|w,b) = -1$
SPAM!	(1,0,1,1,0,1)	$f(x\|w,b) = +1$
SPAM!	(1,0,0,0,0,1)	$f(x\|w,b) = +1$

$$f(x|w,b) = sign(w^T x - b) = sign(w_1 * x_1 + \dots w\_6 * x_6 - b)$$

Formal Definitions

- Training set: $S = \{(x_i, y_i)\}_{i=1}^{N}$, $x \in R^D, y \in \{-1, +1\}$

- Model class: $f\ (x \mid w, b)\ =\ w^T\ x\ -\ b$, Linear Models

aka hypothesis class

- Goal: find (w, b) that predicts well on S.
  – How to quantify "well"?

Basic Supervised Learning Recipe

- Training set: $S = \{(x_i, y_i)\}_{i=1}^{N},\ x \in R^D, y \in \{-1, +1\}$

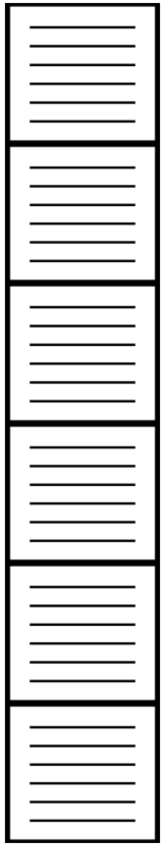
- Model class: $f\ (x \mid w, b) = w^T x - b,$ Linear Models

- Loss Function: $L(target, predict) = (target - predict)^2,$ Square Loss

- Learning Objective: $argmin_{w,b} \sum_{i=1}^{N} L(y_i, f(x_i|w, b)),$ Optimization Problem

Why is Spam Filtering Hard?

$$w = (0.05, 0.05, -0.68, 0.68, -0.63, 0.68)$$
$$b = 0.27$$



Training Set		Bag of Words	
	SPAM!	(0,0,0,1,1,1)	$f(x\|w,b) = +1$
	SPAM!	(1,0,0,1,0,0)	$f(x\|w,b) = +1$
	NOT SPAM	(1,0,1,0,1,0)	$f(x\|w,b) = -1$
	NOT SPAM	(0,1,1,0,1,0)	$f(x\|w,b) = -1$
	SPAM!	(1,0,1,1,0,1)	$f(x\|w,b) = +1$
	SPAM!	(1,0,0,0,0,1)	$f(x\|w,b) = +1$

$$f(x|w,b) = sign(w^T x - b) = sign(w_1 * x_1 + \ldots w\_6 * x_6 - b)$$

Learning Algorithm

$$argmin_{w,b} \sum_{i=1}^{N} L(y_i, f(x_i|w, b))$$

- Typically, requires optimization algorithm.
- Simplest: Gradient Descent

$$w_{t+1} \leftarrow w_t - \partial_w \sum_{i=1}^{N} L\left(y_i, f(x_i \mid w_t, b_t)\right)$$

Loop for T
iterations

$$b_{t+1} \leftarrow b_t - \partial_b \sum_{i=1}^{N} L\left(y_i, f(x_i \mid w_t, b_t)\right)$$

Gradient Review

$$\partial_w \sum_{i=1}^{N} L\big(y_i, f(x_i \mid w,b)\big)$$

$$= \sum_{i=1}^{N} \partial_w L\big(y_i, f(x_i \mid w,b)\big)$$

$$= \sum_{i=1}^{N} -2(y_i - f(x_i \mid w,b))\partial_w f(x_i \mid w,b)$$

$$L(target, predict) = (target - predict)^2$$

Chain Rule

$$= \sum_{i=1}^{N} -2(y_i - w^T x + b)x$$

$$f\ (x \mid w, b)\ =\ w^T\ x\ -\ b$$

Recap: Supervised Learning Recipe

- Training set: $S = \{(x_i, y_i)\}_{i=1}^{N},\ x \in R^D, y \in \{-1, +1\}$
- Model class: $f\ (x \mid w, b)\ =\ w^T\ x\ -\ b,$ Linear Models
- Loss Function: $L(target, predict) = (target - predict)^2$, Square Loss
- Learning Objective: $argmin_{w,b} \sum_{i=1}^{N} L(y_i, f(x_i|w, b))$, Optimization Problem

Recap: Supervised Learning Recipe

- Training set: $S = \{(x_i, y_i)\}_{i=1}^{N},\ x \in R^D, y \in \{-1, +1\}$

- Model class: odels

- Loss Function: s

- Learning Obj on Problem

Congratulations!
You now know the basic steps to training a model!

But is your model any good?

Example: Self-Driving Cars

Basic Setup

- Mounted cameras
- Use image features

- Human demonstrations

- $f(x|w)$ = steering angle
- Learn on training set

Overfitting

- Very accurate model
- But crashed on live test!

Test Error

- “True” distribution: $P(x, y)$
  - Unknown to us

All possible emails

- Train: $f(x) = y$
  - Using training data: $S = \{(x_i, y_i)\}_{i=1}^{N}$
  - Sampled independently from $P(x, y)$

Prediction Loss on all possible emails

- Test Error: $L_p(f) = E_{(x,y)\sim P(x,y)}[L(y, f(y))]$
- Overfitting: Test Error >> Training Error

Overfitting vs Underfitting

- High variance implies **overfitting**
  - Model class unstable
  - Variance increases with model complexity
  - Variance reduces with more training data.
- High bias implies underfitting
  - Even with no variance, model class has high error
  - Bias decreases with model complexity
  - Independent of training data size

Model Selection

- Finite training data
- Complex model classes overfit
- Simple model classes underfit
- Goal: choose model class with the best generalization error

Model Selection

- Finite training data
- Complex m
- Simple m
- Goal: cho
  generalization error

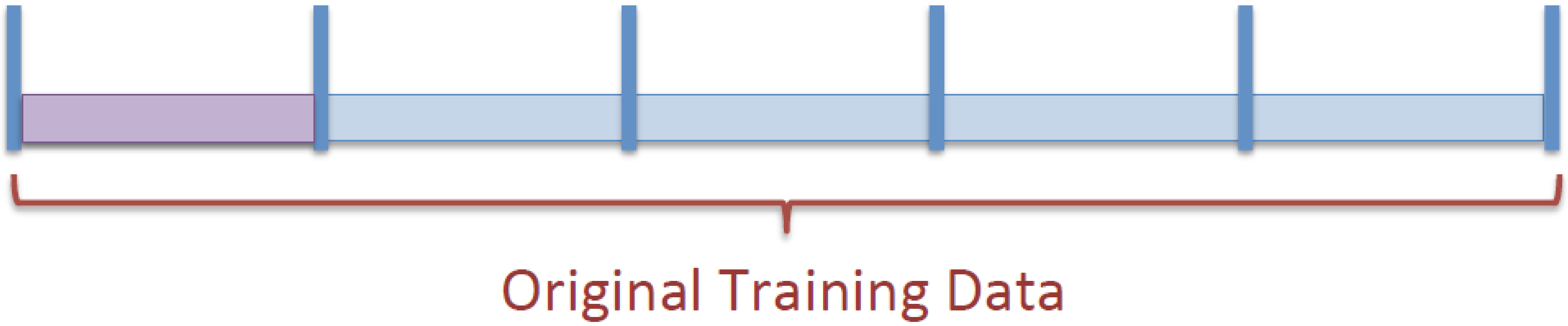
But we can't measure generalization error directly! (We don't have access to the whole distribution.)

Use a Validation Set!

- Split data to Training Set and Validation Set
- Train model on Training Set
- Evaluate on Validation Set
- What’s wrong with this?

Keep training and evaluation separate!

5-Fold Cross Validation



- Split data into 5 equal partitions
- Train on 4 partitions
- Evaluate on 1 partition
- Allows re-using training data as test data

Complete Pipeline (Supervised Learning)

$$S = \left\{(x_i, y_i)\right\}_{i=1}^{N}$$

Training Data

$$f(x \mid w, b) = w^T x - b$$

Model Class(es)

$$L(a,b) = (a-b)^2$$

Loss Function

$$\underset{w,b}{\operatorname{argmin}} \sum_{i=1}^{N} L\big(y_i, f(x_i \mid w, b)\big)$$

Cross Validation & Model Selection

Profit!