Unsupervised Learning
Lecture 3: Naïve Bayes Classifier and Evaluation Scheme

Tho Quan
qttho@hcmut.edu.vn

Agenda

- Discriminative vs. Generative Classifiers
- Naïve Bayes Classifier
- Evaluation scheme: which method is better?
- NB for Text Classification and IR Metrics

Classification Methodologies

- There are three methodologies:

a) Model a classification rule directly

Examples: k-NN, linear classifier, SVM, neural nets, ..

b) Model the probability of class memberships given input data

Examples: logistic regression, probabilistic neural nets (softmax),…

c) Make a probabilistic model of data within each class

Examples: Naive Bayes….

Classification of Classifications

	Probabilistic	Non-Probabilistic
Discriminative	• Logistic Regression • Probabilistic neural nets • ……..	• K-nn • Linear classifier • SVM • Neural networks • ……
Generative	• Naïve Bayes • ……	**N.A. (?)**

Probability Basics

- Prior, conditional and joint probability for random variables
  - Prior probability: $P(x)$
  - Conditional probability: $P(x_1 \mid x_2), P(x_2 \mid x_1)$
  - Joint probability: $\mathbf{x} = (x_1, x_2), P(\mathbf{x}) = P(x_1, x_2)$
  - Relationship: $P(x_1, x_2) = P(x_2 \mid x_1)P(x_1) = P(x_1 \mid x_2)P(x_2)$
  - Independence:

    $$P(x_2 \mid x_1) = P(x_2), P(x_1 \mid x_2) = P(x_1), P(x_1, x_2) = P(x_1)P(x_2)$$
- Bayesian Rule

$$P(c \mid \mathbf{x}) = \frac{P(\mathbf{x} \mid c)P(c)}{P(\mathbf{x})}$$

$$Posterior = \frac{Likelihood \times Prior}{Evidence}$$

Discriminative

Generative

Probabilistic Classification Principle

- **M**aximum **A** **P**osterior (**MAP**) classification rule
  - For an input $\boldsymbol{x}$, find the largest one from L probabilities output by a discriminative probabilistic classifier $P(c_1 \mid \mathbf{x}), \ldots, P(c_L \mid \mathbf{x})$.
  - Assign $\boldsymbol{x}$ to label c^* if $P(c^* \mid \mathbf{x})$ is the largest.
- Generative classification with the MAP rule
  - Apply Bayesian rule to convert them into posterior probabilities

$$P(c_i \mid \mathbf{x}) = \frac{P(\mathbf{x} \mid c_i)P(c_i)}{P(\mathbf{x})} \propto P(\mathbf{x} \mid c_i)P(c_i)$$

$$\text{for } i = 1, 2, \cdots, L$$

Common factor for all L probabilities

  - Then apply the MAP rule to assign a label

Naïve Bayes

- Bayes classification

$$P(c\,|\,\mathbf{x}) \propto P(\mathbf{x}\,|\,c)P(c) = P(x_1,\cdots,x_n\,|\,c)P(c) \quad \text{for } c = c_1,\ldots,c_L.$$

Difficulty: learning the joint probability $P(x_1,\cdots,x_n\,|\,c)$ is infeasible!

- Naïve Bayes classification
  - Assume all input features are class conditionally independent!

$$\begin{aligned} P(x_1,x_2,\cdots,x_n\,|\,c) &= \underline{P(x_1\,|\,x_2,\cdots,x_n,c)}P(x_2,\cdots,x_n\,|\,c) \\ &= \underline{P(x_1\,|\,c)}P(x_2,\cdots,x_n\,|\,c) \\ &= P(x_1\,|\,c)P(x_2\,|\,c)\cdots P(x_n\,|\,c) \end{aligned}$$

Applying the independence assumption

Summarization for Example

- Bayes' Theorem

$$P(H \mid X) = \frac{P(X \mid H)P(H)}{P(X)}$$

Training tuples

Class-labeled training tuples from the *AllElectronics* customer database.

RID	*age*	*income*	*student*	*credit_rating*	*Class: buys_computer*
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

Naïve Bayesian Classification

1. D is a training set.

$$X = (age = youth, income = medium, student = yes, credit\_rating = fair)$$

2. m classes: C_i, i=1..m

$$P(C_i \mid X) = \frac{P(X \mid C_i)P(C_i)}{P(X)}$$

- P(buys_computer=yes | age=youth, income=medium student=yes, credit_rating=fair)

→ We maximize $P(C_i \mid X)$.

Naïve Bayesian Classification

3. $P(C_i) = |C_{i,D}|/|D|$
   - Or: $P(C_1) = P(C_2) = .. = P(C_m)$

Naïve Bayesian Classification

4. Compute $P(X \mid C_i)$:
   - Assume: Class conditional independence

$$P(X \mid C_i) = \prod_{k=1}^{n} P(x_k \mid C_i) = P(x_1 \mid C_i) * P(x_2 \mid C_i) * .. * P(x_n \mid C_i)$$

5. Class label of *X is* C_i
   - $P(C_i|X) > P(C_j|X)$ với $1<=j<=m$, $j<>i$

Example

$C_1 = \{X'|X'.buys\_computer = yes\}$

$C_2 = \{X''|X''.buys\_computer = no\}$

$$\boldsymbol{X} = (age = youth, income = medium, student = yes, credit\_rating = fair)$$

$$P(buys\_computer = yes) = 9/14 = 0.643$$
$$P(buys\_computer = no) = 5/14 = 0.357$$

$$\begin{aligned}
P(age = youth \mid buys_computer = yes) &= 2/9 = 0.222 \\
P(age = youth \mid buys_computer = no) &= 3/5 = 0.600 \\
P(income = medium \mid buys_computer = yes) &= 4/9 = 0.444 \\
P(income = medium \mid buys_computer = no) &= 2/5 = 0.400 \\
P(student = yes \mid buys_computer = yes) &= 6/9 = 0.667 \\
P(student = yes \mid buys_computer = no) &= 1/5 = 0.200 \\
P(credit_rating = fair \mid buys_computer = yes) &= 6/9 = 0.667 \\
P(credit_rating = fair \mid buys_computer = no) &= 2/5 = 0.400
\end{aligned}$$

$$\begin{aligned}
P(\boldsymbol{X}|buys_computer = yes) = {} & P(age = youth \mid buys_computer = yes) \times \\
& P(income = medium \mid buys_computer = yes) \times \\
& P(student = yes \mid buys_computer = yes) \times \\
& P(credit_rating = fair \mid buys_computer = yes) \\
= {} & 0.222 \times 0.444 \times 0.667 \times 0.667 = 0.044.
\end{aligned}$$

$$P(\boldsymbol{X}|buys\_computer = no) = 0.600 \times 0.400 \times 0.200 \times 0.400 = 0.019.$$

$$P(\boldsymbol{X}|buys\_computer = yes)P(buys\_computer = yes) = 0.044 \times 0.643 = 0.028$$
$$P(\boldsymbol{X}|buys\_computer = no)P(buys\_computer = no) = 0.019 \times 0.357 = 0.007$$

$\rightarrow X \in C_1$

Zero probability

- Zero probability elimination
    - 1,000 tuples: 0 tuples with *income = low, 990 tuples with income = medium, and 10* tuples with *income = high.*
  - Using the Laplacian correction

    → Assume that our training database, D, is **so large** that **adding one to each count** that we need would only make a negligible difference in the estimated probability value, yet would conveniently avoid the case of probability values of zero.

    - we pretend that we have 1 more tuple for each income-value pair.

Zero probability

Suppose a dataset with 1000 tuples, income=low (0), income= medium (990), and income = high (10)

Use **Laplacian correction** (or Laplacian estimator)

Adding 1 to each case

Prob(income = low) = 1/1003

Prob(income = medium) = 991/1003

Prob(income = high) = 11/1003

Model Evaluation and Selection

- Evaluation metrics: How can we measure accuracy? Other metrics to consider?
- Use **validation test set** of class-labeled tuples instead of training set when assessing accuracy
- Methods for estimating a classifier's accuracy:
  - Holdout method, random subsampling
  - Cross-validation

Classifier Evaluation Metrics: Confusion Matrix

Confusion Matrix:

Actual class\Predicted class	C_1	$\neg C_1$
C_1	**True Positives (TP)**	**False Negatives (FN)**
$\neg C_1$	**False Positives (FP)**	**True Negatives (TN)**

Actual class\Predicted class	buy_computer = yes	buy_computer = no	Total
buy_computer = yes	**6954**	**46**	7000
buy_computer = no	**412**	**2588**	3000
Total	7366	2634	10000

- Given *m* classes, an entry, $CM_{i,j}$ in a **confusion matrix** indicates # of tuples in class *i* that were labeled by the classifier as class *j*
- May have extra rows/columns to provide totals

Classifier Evaluation Metrics: Accuracy, Error Rate, Sensitivity and Specificity

A\P	C	¬C	
C	**TP**	**FN**	**P**
¬C	**FP**	**TN**	**N**
	P'	**N'**	**All**

- **Classifier Accuracy,** or recognition rate: percentage of test set tuples that are correctly classified

  Accuracy = (TP + TN)/All

- **Error rate:** *1 – accuracy*, or

  Error rate = (FP + FN)/All

- **Class Imbalance Problem**:
  - One class may be *rare*, e.g. fraud, or HIV-positive
  - Significant *majority of the negative class* and minority of the positive class
  - **Sensitivity**: True Positive recognition rate
    - **Sensitivity = TP/P**
  - **Specificity**: True Negative recognition rate
    - **Specificity = TN/N**

Classifier Evaluation Metrics: Precision and Recall, and F-measures

- **Precision**: exactness – what % of tuples that the classifier labeled as positive are actually positive

$$precision = \frac{TP}{TP + FP}$$

- **Recall:** completeness – what % of positive tuples did the classifier label as positive?

$$recall = \frac{TP}{TP + FN}$$

- Perfect score is 1.0
- Inverse relationship between precision & recall
- ***F* measure (*F*$_1$ or *F*-score)**: harmonic mean of precision and recall,

$$F = \frac{2 \times precision \times recall}{precision + recall}$$

- ***F*$_\beta$:** weighted measure of precision and recall
  - assigns ß times as much weight to recall as to precision

$$F_\beta = \frac{(1 + \beta^2) \times precision \times recall}{\beta^2 \times precision + recall}$$

Classifier Evaluation Metrics: Example

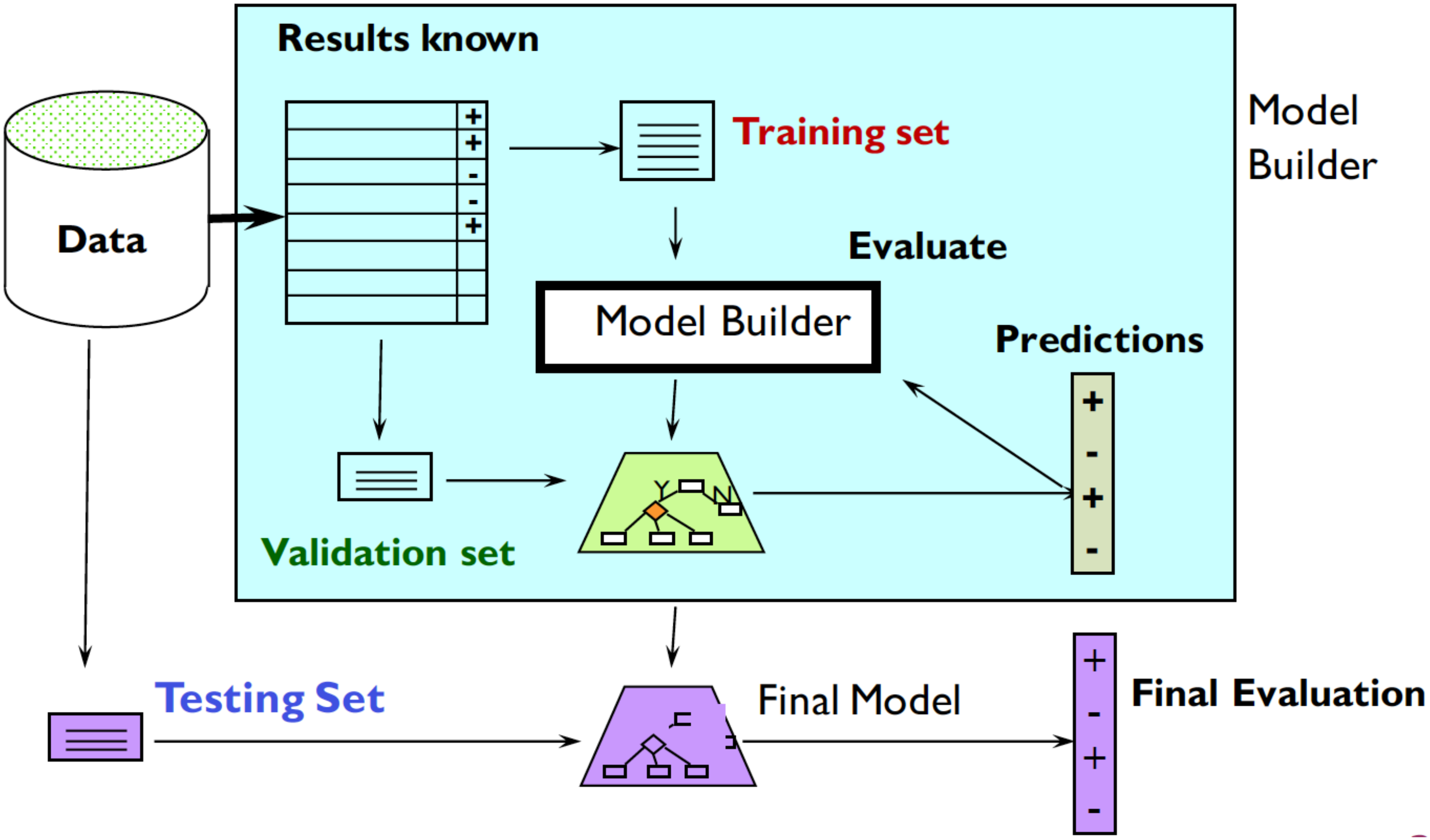
Actual Class\Predicted class	cancer = yes	cancer = no	Total	Recognition(%)
cancer = yes	**90**	**210**	300	30.00 (*sensitivity*
cancer = no	**140**	**9560**	9700	98.56 (*specificity)*
Total	230	9770	10000	96.40 (*accuracy*)

- *Precision* = 90/230 = 39.13% *Recall* = 90/300 = 30.00%

Evaluating Classifier Accuracy: Holdout & Cross-Validation Methods

- **Holdout method**
  - Given data is randomly partitioned into two independent sets
    - Training set (e.g., 2/3) for model construction
    - Test set (e.g., 1/3) for accuracy estimation
  - <u>Random sampling</u>: a variation of holdout
    - Repeat holdout k times, accuracy = avg. of the accuracies obtained
- **Cross-validation** (*k*-fold, where k = 10 is most popular)
  - Randomly partition the data into *k mutually exclusive* subsets, each approximately equal size
  - At *i*-th iteration, use D_i as test set and others as training set
  - <u>Leave-one-out</u>: *k* folds where *k* = # of tuples, for small sized data
  - <u>***Stratified cross-validation***</u>: folds are stratified so that class dist. in each fold is approx. the same as that in the initial data

Train, Validation, Test



Uses of NB classification

- Text Classification
- Spam Filtering
- Hybrid Recommender System
  - Recommender Systems apply machine learning and data mining techniques for filtering unseen information and can predict whether a user would like a given resource
- Online Application
  - Simple Emotion Modeling

Examples of Text Classification

- CLASSES=BINARY
  - “spam” / “not spam”
- CLASSES =TOPICS
  - “finance” / “sports” / “politics”
- CLASSES =OPINION
  - “like” / “hate” / “neutral”
- CLASSES =TOPICS
  - “AI” / “Theory” / “Graphics”
- CLASSES =AUTHOR
  - “Shakespeare” / “Marlowe” / “Ben Jonson”

Naive Bayes for Text Categorization

- Attributes are text positions, values are words.

$$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j) \prod_i P(x_i \mid c_j)$$

$$= \underset{c_j \in C}{\operatorname{argmax}} P(c_j) P(x_1 = \text{"our"} \mid c_j) \cdots P(x_n = \text{"text"} \mid c_j)$$

- Still too many possibilities
- Assume that classification is *independent* of the positions of the words
  - Use same parameters for each position
  - Result is bag of words model (over tokens not types)

Naïve Bayes: Learning

- From training corpus, extract *Vocabulary*
- Calculate required $P(c_j)$ and $P(x_k \mid c_j)$ terms
  - For each c_j in C do
    - $docs_j \leftarrow$ subset of documents for which the target class is c_j
    - $$P(c_j) \leftarrow \frac{|docs_j|}{|\text{total \# documents}|}$$
  - $Text_j \leftarrow$ single document containing all $docs_j$
  - for each word x_k in *Vocabulary*
    - $n_k \leftarrow$ number of occurrences of x_k in $Text_j$

$$P(x_k \mid c_j) \leftarrow \frac{n_k + \alpha}{n + \alpha |Vocabulary|}$$

Naïve Bayes: Classifying

- positions ← all word positions in current document which contain tokens found in *Vocabulary*

- Return c_{NB}, where

$$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j) \prod_{i \in positions} P(x_i \mid c_j)$$