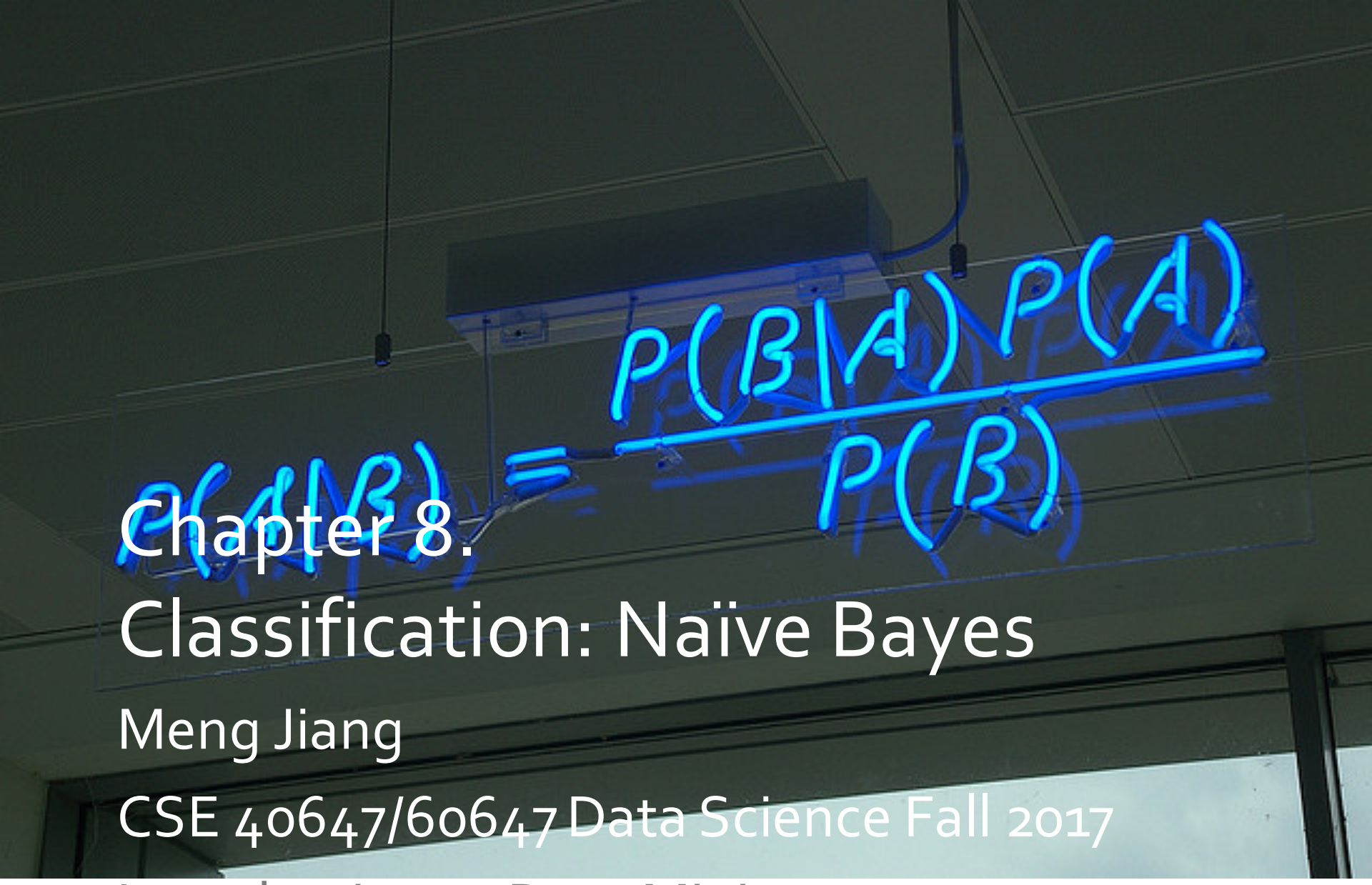


Chapter 8. Classification: Naïve Bayes

Meng Jiang

CSE 40647/60647 Data Science Fall 2017

Introduction to Data Mining

Classification: Basic Concepts

- Classification: Basic Concepts
- Decision Tree Induction
- **Bayes Classification Methods**
- Model Evaluation and Selection
- Techniques to Improve Classification Accuracy: Ensemble Methods

Bayes' Theorem: Basics

Proof of Bayes Theorem

The probability of two events A and B happening, P(A∩B), is the probability of A, P(A), times the probability of B given that A has occurred, P(B|A).

$$P(A \cap B) = P(A)P(B|A) \tag{1}$$

On the other hand, the probability of A and B is also equal to the probability of B times the probability of A given B.

$$P(A \cap B) = P(B)P(A|B) \tag{2}$$

Equating the two yields:

$$P(B)P(A|B) = P(A)P(B|A) \tag{3}$$

and thus

$$P(A|B) = P(A)\frac{P(B|A)}{P(B)} \tag{4}$$

This equation, known as Bayes Theorem is the basis of statistical inference.

Bayesian Classification: Why?

- A statistical classifier: performs *probabilistic prediction*, *i.e.*, predicts class membership probabilities
- Foundation: Based on Bayes' Theorem.
- Performance: A simple Bayesian classifier, *naïve Bayesian classifier*, has comparable performance with decision tree and selected neural network classifiers
- Incremental: Each training example can incrementally increase/decrease the probability that a hypothesis is correct — prior knowledge can be combined with observed data

Bayes' Theorem: Basics

- Bayes' Theorem: $$P(H|\mathbf{X}) = \frac{P(\mathbf{X}|H)P(H)}{P(\mathbf{X})} = P(\mathbf{X}|H) \times P(H) / P(\mathbf{X})$$
  - Let **X** be a data sample ("*evidence*"): class label is unknown
  - Let H be a *hypothesis* that X belongs to class C
  - Classification is to determine P(H|**X**), (i.e., *posteriori probability):* the probability that the hypothesis holds given the observed data sample **X**
  - P(H) (*prior probability*): the initial probability
    - E.g., **X** will buy computer, regardless of age, income, …
  - P(**X**): probability that sample data is observed
  - P(**X**|H) (likelihood): the probability of observing the sample **X**, given that the hypothesis holds
    - E.g., Given that **X** will buy computer, the prob. that X is 31..40, medium income

Prediction Based on Bayes' Theorem

- Given training data **X**, *posteriori probability of a hypothesis* H, P(H|**X**), follows the Bayes' theorem

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

- Informally, this can be viewed as

  posteriori = likelihood x prior/evidence

- Predicts **X** belongs to C_i iff the probability $P(C_i|\mathbf{X})$ is the highest among all the $P(C_k|X)$ for all the *k* classes
- Practical difficulty: It requires initial knowledge of many probabilities, involving significant computational cost

Classification is to Derive the Maximum Posteriori

- Let D be a training set of tuples and their associated class labels, and each tuple is represented by an n-D attribute vector $\mathbf{X} = (x_1, x_2, \ldots, x_n)$
- Suppose there are *m* classes $C_1, C_2, \ldots, C_m$.
- Classification is to derive the maximum posteriori, i.e., the maximal $P(C_i|\mathbf{X})$
- This can be derived from Bayes' theorem

$$P(C_i|\mathbf{X}) = \frac{P(\mathbf{X}|C_i)P(C_i)}{P(\mathbf{X})}$$

- Since P(X) is constant for all classes, only

$$P(C_i|\mathbf{X}) = P(\mathbf{X}|C_i)P(C_i)$$

needs to be maximized

Naïve Bayes Classifier

- A simplified assumption: attributes are conditionally independent (i.e., no dependence relation between attributes):

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

- This greatly reduces the computation cost: Only counts the class distribution
- If A_k is categorical, $P(x_k|C_i)$ is the # of tuples in C_i having value x_k for A_k divided by $|C_{i,D}|$ (# of tuples of C_i in D)
- If A_k is continous-valued, $P(x_k|C_i)$ is usually computed based on Gaussian distribution with a mean μ and standard deviation σ

$$g(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

and $P(x_k|C_i)$ is $P(\mathbf{X} \mid C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i})$

Naïve Bayes Classifier: Training Dataset

- Class:
  - C1: buys_computer = 'yes'
  - C2: buys_computer = 'no'
- Data to be classified:
  - X = (age <=30, Income = medium, Student = yes, Credit_rating = Fair)

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Naïve Bayes Classifier: An Example

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

- $P(C_i)$: P(buys_computer = "yes") = 9/14 = 0.643
  P(buys_computer = "no") = 5/14= 0.357
- Compute $P(X|C_i)$ for each class

P(age = "<=30"|buys_computer = "yes") = 2/9 = 0.222
P(age = "<= 30"|buys_computer = "no") = 3/5 = 0.6
P(income = "medium" | buys_computer = "yes") = 4/9 = 0.444
P(income = "medium" | buys_computer = "no") = 2/5 = 0.4
P(student = "yes" | buys_computer = "yes) = 6/9 = 0.667
P(student = "yes" | buys_computer = "no") = 1/5 = 0.2
P(credit_rating = "fair" | buys_computer = "yes") = 6/9 = 0.667
P(credit_rating = "fair" | buys_computer = "no") = 2/5 = 0.4

- **X = (age <= 30 , income = medium, student = yes, credit_rating = fair)**

$P(X|C_i)$: P(X|buys_computer = "yes") = 0.222 x 0.444 x 0.667 x 0.667 = 0.044
P(X|buys_computer = "no") = 0.6 x 0.4 x 0.2 x 0.4 = 0.019

$P(X|C_i)*P(C_i)$: P(X|buys_computer = "yes") * P(buys_computer = "yes") = 0.028
P(X|buys_computer = "no") * P(buys_computer = "no") = 0.007

Therefore, X belongs to class ("buys_computer = yes")

Avoiding the Zero-Probability Problem

- Naïve Bayesian prediction requires each conditional prob. be **non-zero**. Otherwise, the predicted prob. will be zero

$$P(X \mid C_i) = \prod_{k=1}^{n} P(x_k \mid C_i)$$

- Ex. 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
  - The "corrected" prob. estimates are close to their "uncorrected" counterparts

Naïve Bayes Classifier: Comments

- Advantages
  - Easy to implement
  - Good results obtained in most of the cases
- Disadvantages
  - Assumption: class conditional independence, therefore loss of accuracy
  - Practically, dependencies exist among variables
    - E.g.,
      - hospitals: patients: Profile: age, family history, etc.
      - Symptoms: fever, cough etc.
      - Disease: lung cancer, diabetes, etc.
    - Dependencies among these cannot be modeled by Naïve Bayes Classifier

Discussion

- https://www.springboard.com/blog/machine-learning-interview-questions/

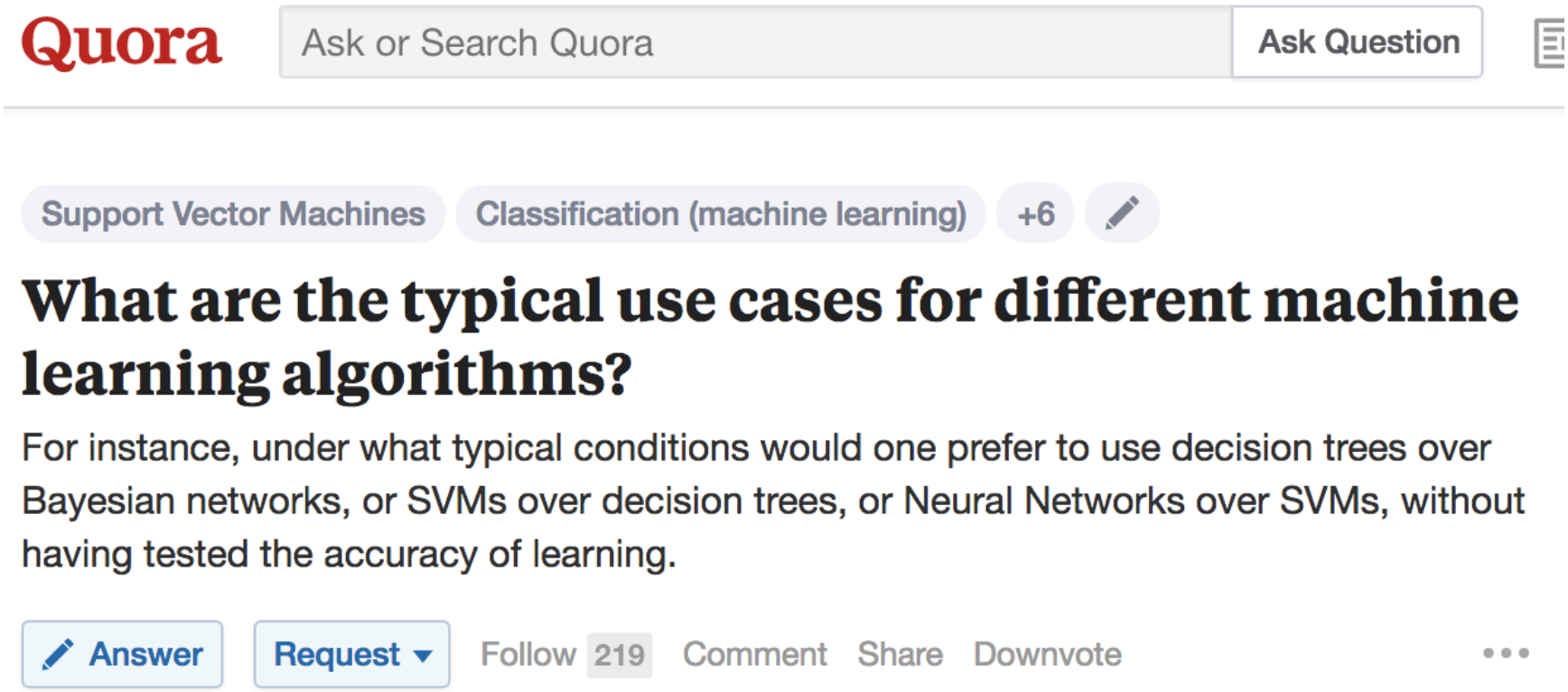
Q7- Why is "Naive" Bayes naive?

More reading: Why is "naive Bayes" naive? (Quora)

Despite its practical applications, especially in text mining, Naive Bayes is considered "Naive" because it makes an assumption that is virtually impossible to see in real-life data: the conditional probability is calculated as the pure product of the individual probabilities of components. This implies the absolute independence of features — a condition probably never met in real life.

As a Quora commenter put it whimsically, a Naive Bayes classifier that figured out that you liked pickles and ice cream would probably naively recommend you a pickle ice cream.

Discussion


Quora
Ask or Search Quora
Ask Question
Support Vector Machines
Classification (machine learning)
+6
What are the typical use cases for different machine learning algorithms?
For instance, under what typical conditions would one prefer to use decision trees over Bayesian networks, or SVMs over decision trees, or Neural Networks over SVMs, without having tested the accuracy of learning.
Answer
Request
Follow 219
Comment
Share
Downvote

Discussion

Yuval Feinstein, Algorithmic Software Engineer in NLP,IR and Machine Learning

Answered Apr 5, 2013

Some more emphasis on things Ganesh Parameswaran and Amir Masoud Abdol mentioned in passing:

1. Start with your goal - the list of algorithms in the question seems to point towards classification (supervised learning). Do you just need the best classification in terms of accuracy? Or do you also need meaningful features? In the latter case, decision trees and Naive Bayes might be better than other, more "black box", algorithms, like SVM or ANN.
2. First of all, look at the data. Clean it. Choose appropriate features. The choice of algorithm is secondary to having lots of quality data.
3. Set up in advance criteria for success - what is your error measure? Is 75% precision good enough?
4. Resources - some algorithms need more memory or more runtime than others - e.g. random forests need more memory.

Discussion

5. Online/batch - How soon do you need to classify a new instance? Most algorithms require intensive training, but allow for a relatively quick labeling of a new instance.

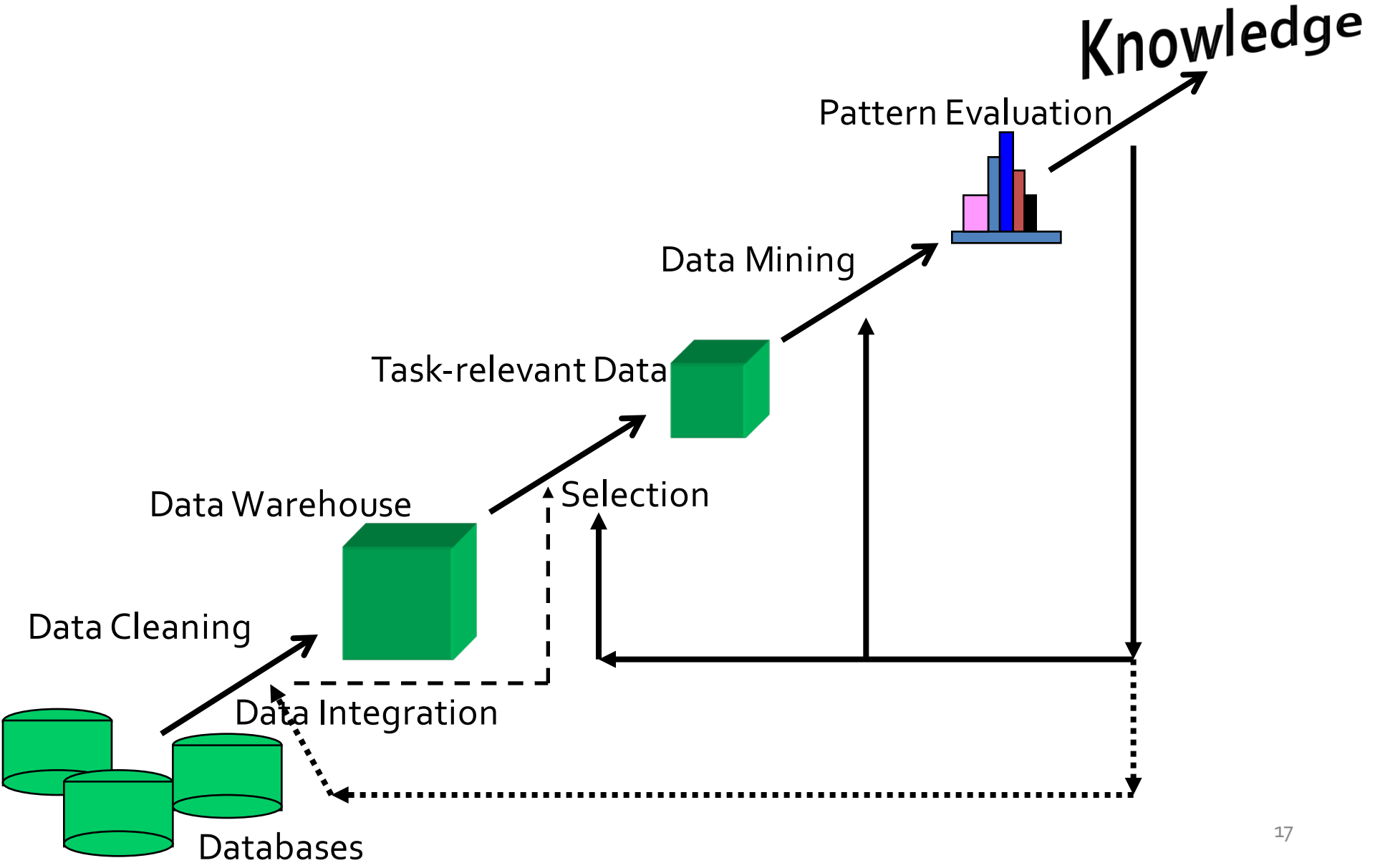
6. Scaling - for millions of instances, it is probably better to use an algorithm that scales well (e.g. Naive Bayes).

7. Cost of labels - you need to get the labels for the training set from somewhere. Getting labels is usually costly in time, effort and money. Some algorithms can do with less labeled data. I second Ganesh Parameswaran's recommendation for semi-supervised models, as they can do more with the same amount of labels.

8. Ensembles - In many cases, using several algorithms and combining the results (say using a majority vote) works better than individual algorithms.

9. Iterate - If you have reached your criteria for success - stop. Otherwise, try a different feature/algorithm combination.

Reminder: KDD Process



References

- C. Apte and S. Weiss. Data mining with decision trees and decision rules. Future Generation Computer Systems, 13, 1997
- P. K. Chan and S. J. Stolfo. Learning arbiter and combiner trees from partitioned data for scaling machine learning. KDD'95
- A. J. Dobson. An Introduction to Generalized Linear Models. Chapman & Hall, 1990.
- R. O. Duda, P. E. Hart, and D. G. Stork. Pattern Classification, 2ed. John Wiley, 2001
- U. M. Fayyad. Branching on attribute values in decision tree generation. AAAI'94.
- Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. J. Computer and System Sciences, 1997.
- J. Gehrke, R. Ramakrishnan, and V. Ganti. Rainforest: A framework for fast decision tree construction of large datasets. VLDB'98.
- J. Gehrke, V. Gant, R. Ramakrishnan, and W.-Y. Loh, BOAT -- Optimistic Decision Tree Construction. SIGMOD'99.
- T. Hastie, R. Tibshirani, and J. Friedman. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer-Verlag, 2001.
- T.-S. Lim, W.-Y. Loh, and Y.-S. Shih. A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. Machine Learning, 2000

References (cont.)

- J. Magidson. The Chaid approach to segmentation modeling: Chi-squared automatic interaction detection. In R. P. Bagozzi, editor, Advanced Methods of Marketing Research, Blackwell Business, 1994
- M. Mehta, R. Agrawal, and J. Rissanen. SLIQ : A fast scalable classifier for data mining. EDBT'96
- T. M. Mitchell. Machine Learning. McGraw Hill, 1997
- S. K. Murthy, Automatic Construction of Decision Trees from Data: A Multi-Disciplinary Survey, Data Mining and Knowledge Discovery 2(4): 345-389, 1998
- J. R. Quinlan. Induction of decision trees. Machine Learning, 1:81-106, 1986.
- J. R. Quinlan. C4.5: Programs for Machine Learning. Morgan Kaufmann, 1993.
- J. R. Quinlan. Bagging, boosting, and c4.5. AAAI'96.
- R. Rastogi and K. Shim. **Public: A decision tree classifier that integrates building and pruning**. VLDB'98
- J. Shafer, R. Agrawal, and M. Mehta. **SPRINT : A scalable parallel classifier for data mining**. VLDB'96
- J. W. Shavlik and T. G. Dietterich. **Readings in Machine Learning**. Morgan Kaufmann, 1990
- P. Tan, M. Steinbach, and V. Kumar. **Introduction to Data Mining**. Addison Wesley, 2005
- S. M. Weiss and C. A. Kulikowski. **Computer Systems that Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems**. Morgan Kaufman, 1991
- S. M. Weiss and N. Indurkhya. **Predictive Data Mining**. Morgan Kaufmann, 1997
- I. H. Witten and E. Frank. **Data Mining: Practical Machine Learning Tools and Techniques**, 2ed. Morgan Kaufmann, 2005