Classification

Vikram Pudi
vikram@iiit.ac.in
IIIT Hyderabad

Talk Outline

- Introduction
  - Classification Problem
  - Applications
  - Metrics
  - Combining classifiers
- Classification Techniques

The Classification Problem

Outlook	Temp (°F)	Humidity (%)	Windy?	Class
sunny	75	70	true	*play*
sunny	80	90	true	*don't play*
sunny	85	85	false	*don't play*
sunny	72	95	false	*don't play*
sunny	69	70	false	*play*
overcast	72	90	true	*play*
overcast	83	78	false	*play*
overcast	64	65	true	*play*
overcast	81	75	false	*play*
rain	71	80	true	*don't play*
rain	65	70	true	*don't play*
rain	75	80	false	*play*
rain	68	80	false	*play*
rain	70	96	false	*play*

sunny	77	69	true	?
rain	73	76	false	?

Play Outside?

Model relationship between class labels and attributes

e.g. outlook = overcast ⇒ class = *play*

⇒ Assign class labels to new data with *unknown* labels

Applications

- Text classification
  - Classify emails into spam / non-spam
  - Classify web-pages into yahoo-type hierarchy
  - NLP Problems
    - Tagging: Classify words into verbs, nouns, etc.
- Risk management, Fraud detection, Computer intrusion detection
  - Given the properties of a transaction (items purchased, amount, location, customer profile, etc.)
  - Determine if it is a fraud
- Machine learning / pattern recognition applications
  - Vision
  - Speech recognition
  - etc.
- All of science & knowledge is about predicting future in terms of past
  - So classification is a very fundamental problem with ultra-wide scope of applications

Metrics

1. accuracy
2. classification time per new record
3. training time
4. main memory usage (during classification)
5. model size

Accuracy Measure

- Prediction is just like tossing a coin (random variable X)
  - "Head" is "success" in classification; X = 1
  - "tail" is "error"; X = 0
  - X is actually a mapping: {"success": 1, "error" : 0}
- In statistics, a succession of independent events like this is called a *bernoulli process*
  - Accuracy = P(X = 1) = p
  - mean value = $\mu = E[X] = p\times1 + (1-p)\times0 = p$
  - variance = $\sigma^2 = E[(X-\mu)^2] = p\,(1-p)$
- Confidence intervals: Instead of saying accuracy = 85%, we want to say: accuracy ∈ [83, 87] with a confidence of 95%

Binomial Distribution

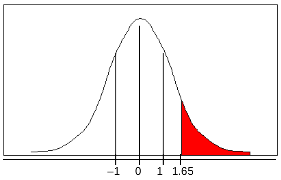
- Treat each classified record as a bernoulli trial
- If there are *n* records, there are *n* independent and identically distributed (iid) bernoulli trials, X_i, i = 1,…,*n*
- Then, the random variable $X = \sum_{i=1,\ldots,n} X_i$ is said to follow a *binomial distribution*
  - $P(X = k) = {}^{n}C_k\, p^k (1-p)^{n-k}$
- Problem: Difficult to compute for large *n*

Normal Distribution

- Continuous distribution with parameters μ (mean), σ^2(variance)
- Probability density:
  $f(x) = (1/\sqrt{(2\pi\sigma^2)}) \exp(-(x-\mu)^2 / (2\sigma^2))$
- Central limit theorem:
  - Under certain conditions, the distribution of the sum of a *large number* of iid random variables is approximately normal
  - A *binomial distribution* with parameters *n* and *p* is approximately normal for large *n* and *p* not too close to 1 or 0
  - The approximating normal distribution has mean $\mu = np$ and standard deviation $\sigma^2 = (n\, p\, (1 - p))$

Confidence Intervals

Normal distribution with mean = 0 and variance = 1



Pr[$X \ge z$]	*z*
0.1%	3.09
0.5%	2.58
1%	2.33
5%	1.65
10%	1.28
20%	0.84
40%	0.25

- E.g. $P[-1.65 \le X \le 1.65] = 1 - 2 \times P[X \ge 1.65] = 90\%$
- To use this we have to transform our random variable to have mean = 0 and variance = 1
- Subtract mean from *X* and divide by standard deviation

Estimating Accuracy

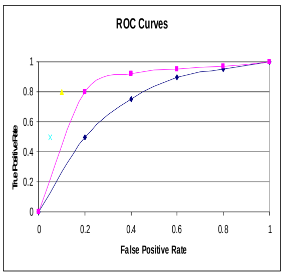
- Holdout method
  - Randomly partition data: training set + test set
  - accuracy = |correctly classified points| / |test data points|
- Stratification
  - Ensure each class has approximately equal proportions in both partitions
- Random subsampling
  - Repeat holdout *k* times. Output average accuracy.
- *k*-fold cross-validation
  - Randomly partition data: $S_1, S_2, \ldots, S_k$
  - First, keep S_1 as test set, remaining as training set
  - Next, keep S_2 as test set, remaining as training set, etc.
  - accuracy = |*total* correctly classified points| / |total data points|
- Recommendation:
  - Stratified 10-fold cross-validation. If possible, repeat 10 times and average results. (reduces variance)

Is Accuracy Enough?

- If only 1% population has cancer, then a test for cancer that classifies *all* people as *non-cancer* will have 99% accuracy.
- Instead output a confusion matrix:

Actual/ Estimate	Class 1	Class 2	Class 3
Class 1	90%	5%	5%
Class 2	2%	91%	7%
Class 3	8%	3%	89%

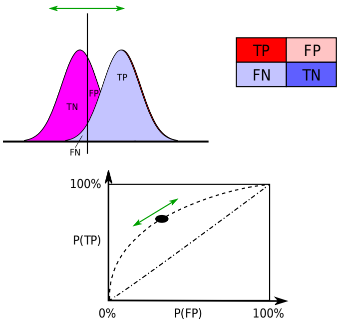
Receiver Operating Characteristic



- Useful in visually comparing classifiers.
- Top-left is best.
- Bottom-right is worst.
- Area under curve is a measure of accuracy.

ROC Interpretation Example

- Blood protein levels in healthy and diseased people are normally distributed with means of 1 g/dL and 2 g/dL.
- Experimenter can adjust threshold (to design a medical test; black vertical line in figure).
- Increasing threshold = fewer false positives



Source: wikipedia. Image by Sharpr - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=44059691

Combining Classifiers

- Get *k* random samples with replacement as training sets (like in random subsampling).
  - ⇒ We get *k* classifiers
- Bagging: Take a *majority vote* for the best class for each new record
- Boosting: Each classifier's vote has a *weight* proportional to its accuracy on training data
  - ⇒ Like a patient taking multiple opinions from several doctors

Talk Outline

- Introduction
- Classification Techniques
  1. Nearest Neighbour Methods
  2. Decision Trees
     - ID3, CART, C4.5, C5.0, SLIQ, SPRINT
  3. Bayesian Methods
     - Naïve Bayes, Bayesian Belief Networks
     - Maximum Entropy Based Approaches
  4. Association Rule Based Approaches
  5. Soft-computing Methods:
     - Genetic Algorithms, Rough Sets, Fuzzy Sets, Neural Networks
  6. Support Vector Machines

Nearest Neighbour Methods

k-NN, Reverse Nearest Neighbours

k-Nearest Neighbours

- Model = Training data
- Classify record R using the *k* nearest neighbours of R in the training data.
- Most frequent class among *k* NNs
- Distance function could be euclidean
- Use an index structure (e.g. R* tree) to find the *k* NNs efficiently

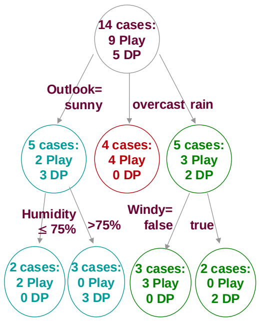
Reverse Nearest Neighbours

- Records which consider *R* as a *k*-NN
- Output most frequent class among RNNs.
- More resilient to outliers.

Decision Trees

Decision Trees

Outlook	Temp (°F)	Humidity (%)	Windy?	Class
sunny	75	70	true	*play*
sunny	80	90	true	*don't play*
sunny	85	85	false	*don't play*
sunny	72	95	false	*don't play*
sunny	69	70	false	*play*
overcast	72	90	true	*play*
overcast	83	78	false	*play*
overcast	64	65	true	*play*
overcast	81	75	false	*play*
rain	71	80	true	*don't play*
rain	65	70	true	*don't play*
rain	75	80	false	*play*
rain	68	80	false	*play*
rain	70	96	false	*play*



Basic Tree Building Algorithm

MakeTree (Training Data D):
  Partition(D)

Partition (Data D):
  if all points in D are in same class: return
  Evaluate splits for each attribute A
  Use best split found to partition D into $D_1, D_2, \ldots, D_n$
  for each D_i:
    Partition (D_i)

ID3, CART

ID3

- Use *information gain* to determine best split
- *gain* = $H(D) - \Sigma_{i=1\ldots n} P(D_i) H(D_i)$
- $H(p_1, p_2, \ldots, p_m) = -\Sigma_{i=1\ldots m} p_i \log p_i$
- like 20-question game
  - Which attribute is better to look for first: "Is it a living thing?" or "Is it a duster?"

CART

- Only create two children for each node
- Goodness of a split (Φ)
  $\Phi = 2 P(D_1) P(D_2) \Sigma_{i=1\ldots m} | P(C_j / D_1) - P(C_j / D_2) |$

Shannon's Entropy

- An expt has several possible outcomes
- In N (e.g. 12) expts, suppose each outcome occurs M (e.g. 3) times
- This means there are N/M (e.g. 4) possible outcomes
- To represent each outcome, we need log N/M (e.g. 2) bits.
  - This generalizes even when all outcomes are not equally frequent.
  - Reason: For an outcome j that occurs M times, there are N/M equi-probable events among which only one cp to j
- Since $p_i = M / N$ (e.g. 25%), information content of an outcome is $-\log p_i$ (e.g. 2)
- So, expected info content: $H = -\Sigma\, p_i \log p_i$ (e.g. 0.25*2 * 4 = 2)

C4.5, C5.0

- Handle missing data
  - During tree building, ignore missing data
  - During classification, predict value of missing data based on attribute values of other records
- Continuous data
  - Divide continuous data into ranges based on attribute values found in training data
- Pruning
  - Prepruning
  - Postpruning – replace subtree with leaf if error-rate doesn't change much
- Rules
- Goodness of split: Gain-ratio
  - gain favours attributes with many values (leads to over-fitting)
  - gain-ratio = gain / $H(P(D_1), P(D_2), \ldots, P(D_n))$
- C5.0 – Commercial version of C4.5
  - Incorporated boosting; other secret techniques

SLIQ

- Motivations:
  - Previous algorithms consider only memory-resident data
  - In determining the entropy of a split on a non-categorical attribute, the attribute values have to be sorted

SLIQ

- **Data structure: class list and attribute lists**

Outlook	Temp (ºF)	Humidity (%)	Windy?	Class
sunny	75	70	true	Play
sunny	80	90	true	Don't Play
sunny	85	85	false	Don't Play
sunny	72	95	false	Don't Play
sunny	69	70	false	Play
overcast	72	90	true	Play
overcast	83	78	false	Play
overcast	64	65	true	Play
overcast	81	75	false	Play
rain	71	80	true	Don't Play
rain	65	70	true	Don't Play
rain	75	80	false	Play
rain	68	80	false	Play
rain	70	96	false	Play

Index	Humidity (%)	Class	Class List Index	Leaf
1	65	Play	8	N1
2	70	Don't Play	1	N1
3	70	Don't Play	5	N1
4	70	Don't Play	11	N1
5	75	Play	9	N1
6	78	Play	7	N1
7	80	Play	10	N1
8	80	Play	12	N1
9	80	Play	13	N1
10	85	Don't Play	3	N1
11	90	Don't Play	2	N1
12	90	Play	6	N1
13	95	Play	4	N1
14	96	Play	14	N1

The attribute list for Humidity
The class list

SLIQ

- **Data structure: class list and attribute lists**

Humidity (%)	Class List Index
65	8
70	1
70	5
70	11
75	9
78	7
80	10
80	12
80	13
85	3
90	2
90	6
95	4
96	14

The attribute list for Humidity

Index	Class	Leaf
1	Play	N1
2	Don't Play	N1
3	Don't Play	N1
4	Don't Play	N1
5	Play	N1
6	Play	N1
7	Play	N1
8	Play	N1
9	Play	N1
10	Don't Play	N1
11	Don't Play	N1
12	Play	N1
13	Play	N1
14	Play	N1

The class list

Node: N1

Value	Class	Frequency
<=65	Play	1

SLIQ

- **Data structure: class list and attribute lists**

Humidity (%)	Class List Index
65	8
70	1
70	5
70	11
75	9
78	7
80	10
80	12
80	13
85	3
90	2
90	6
95	4
96	14

The attribute list for Humidity

Index	Class	Leaf
1	Play	N1
2	Don't Play	N1
3	Don't Play	N1
4	Don't Play	N1
5	Play	N1
6	Play	N1
7	Play	N1
8	Play	N1
9	Play	N1
10	Don't Play	N1
11	Don't Play	N1
12	Play	N1
13	Play	N1
14	Play	N1

The class list

Node: N1

Value	Class	Frequency
<=65	Play	1
<=70	Play	2

SLIQ

- **Data structure: class list and attribute lists**

Humidity (%)	Class List Index
65	8
70	1
70	5
70	11
75	9
78	7
80	10
80	12
80	13
85	3
90	2
90	6
95	4
96	14

The attribute list for Humidity

Index	Class	Leaf
1	Play	N1
2	Don't Play	N1
3	Don't Play	N1
4	Don't Play	N1
5	Play	N1
6	Play	N1
7	Play	N1
8	Play	N1
9	Play	N1
10	Don't Play	N1
11	Don't Play	N1
12	Play	N1
13	Play	N1
14	Play	N1

The class list

Node: N1

Value	Class	Frequency
<=65	Play	1
<=70	Play	3

SLIQ

- **Data structure: class list and attribute lists**

Humidity (%)	Class List Index
65	8
70	1
70	5
70	11
75	9
78	7
80	10
80	12
80	13
85	3
90	2
90	6
95	4
96	14

The attribute list for Humidity

Index	Class	Leaf
1	Play	N1
2	Don't Play	N1
3	Don't Play	N1
4	Don't Play	N1
5	Play	N1
6	Play	N1
7	Play	N1
8	Play	N1
9	Play	N1
10	Don't Play	N1
11	Don't Play	N1
12	Play	N1
13	Play	N1
14	Play	N1

The class list

Node: N1

Value	Class	Frequency
<=65	Play	1
<=70	Play	3
<=70	DP	1
…	…	…
<=96	Play	9
<=96	DP	5

The entropies of various split points can be calculated from these figures. The next attribute list is then scanned

SLIQ

- **Data structure: class list and attribute lists**

Humidity (%)	Class List Index
65	8
70	1
70	5
70	11
75	9
78	7
80	10
80	12
80	13
85	3
90	2
90	6
95	4
96	14

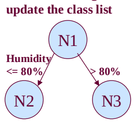
The attribute list for Humidity

Index	Class	Leaf
1	Play	N1
2	Don't Play	N1
3	Don't Play	N1
4	Don't Play	N1
5	Play	N1
6	Play	N1
7	Play	N1
8	Play	N1
9	Play	N1
10	Don't Play	N1
11	Don't Play	N1
12	Play	N1
13	Play	N1
14	Play	N1

The class list

Assume the split point "Humidity <= 80%" is the best one among all possible splits of all attributes

The Humidity attribute list is scanned again to update the class list



SLIQ

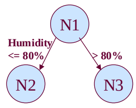
- **Data structure: class list and attribute lists**

Humidity (%)	Class List Index
65	8
70	1
70	5
70	11
75	9
78	7
80	10
80	12
80	13
85	3
90	2
90	6
95	4
96	14

The attribute list for Humidity

Index	Class	Leaf
1	Play	N1
2	Don't Play	N1
3	Don't Play	N1
4	Don't Play	N1
5	Play	N1
6	Play	N1
7	Play	N1
8	Play	N2
9	Play	N1
10	Don't Play	N1
11	Don't Play	N1
12	Play	N1
13	Play	N1
14	Play	N1

The class list



SLIQ

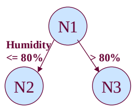
- **Data structure: class list and attribute lists**

Humidity (%)	Class List Index
65	8
70	1
70	5
70	11
75	9
78	7
80	10
80	12
80	13
85	3
90	2
90	6
95	4
96	14

The attribute list for Humidity

Index	Class	Leaf
1	Play	N2
2	Don't Play	N1
3	Don't Play	N1
4	Don't Play	N1
5	Play	N1
6	Play	N1
7	Play	N1
8	Play	N2
9	Play	N1
10	Don't Play	N1
11	Don't Play	N1
12	Play	N1
13	Play	N1
14	Play	N1

The class list



SLIQ

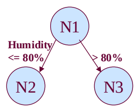
- **Data structure: class list and attribute lists**

Humidity (%)	Class List Index
65	8
70	1
70	5
70	11
75	9
78	7
80	10
80	12
80	13
85	3
90	2
90	6
95	4
96	14

The attribute list for Humidity

Index	Class	Leaf
1	Play	N2
2	Don't Play	N3
3	Don't Play	N3
4	Don't Play	N3
5	Play	N2
6	Play	N3
7	Play	N2
8	Play	N2
9	Play	N2
10	Don't Play	N2
11	Don't Play	N2
12	Play	N2
13	Play	N2
14	Play	N3

The class list



SLIQ

- **Data structure: class list and attribute lists**

Humidity (%)	Class List Index
65	8
70	1
70	5
70	11
75	9
78	7
80	10
80	12
80	13
85	3
90	2
90	6
95	4
96	14

The attribute list for Humidity

Index	Class	Leaf
1	Play	N2
2	Don't Play	N3
3	Don't Play	N3
4	Don't Play	N3
5	Play	N2
6	Play	N3
7	Play	N2
8	Play	N2
9	Play	N2
10	Don't Play	N2
11	Don't Play	N2
12	Play	N2
13	Play	N2
14	Play	N3

The class list

Node: N2

Value	Class	Frequency
<=65	Play	1
…	…	…
<=80	Play	7
<=80	DP	2

Node: N3

Value	Class	Frequency
<=85	DP	1
…	…	…
<=96	Play	2
<=96	DP	3

SLIQ

- Motivations (review):
  - Previous algorithms consider only memory-resident data
    - At any time, only the class list and 1 attribute list in memory
    - A new layer (vs. the child nodes of a single node) is created by at most 2 scans of each attribute list
  - In determining the entropy of a split on a non-categorical attribute, the attribute values have to be sorted
    - Presorting: each attribute list is sorted only once

SPRINT

- Motivations:
  - The class list in SLIQ has to reside in memory, which is still a bottleneck for scalability
  - Can the decision tree building process be carried out by multiple machines in parallel?
  - Frequent lookup of the central class list produces a lot of network communication in the parallel case

SPRINT

- Proposed Solutions
  - Eliminate the class list
    1. Class labels distributed to each attribute list
       => Redundant data, but the memory-resident and network communication bottlenecks are removed
    2. Each node keeps its own set of attribute lists
       => No need to lookup the node information
  - Each node is assigned a partition of each attribute list. The nodes are ordered so that the combined lists of non-categorical attributes remain sorted
  - Each node produces its local histograms in parallel, the combined histograms can be used to find the best splits

Bayesian Methods

Naïve Bayes

- New data point to classify: $X=(x_1,x_2,\ldots x_m)$
- Strategy:
  - Calculate $P(C_i/X)$ for each class C_i.
  - Select C_i for which $P(C_i/X)$ is maximum

$$P(C_i/X) = P(X/C_i)\,P(C_i) / P(X)$$
$$\propto P(X/C_i)\,P(C_i)$$
$$\propto P(x_1/C_i)\,P(x_2/C_i)\ldots P(x_m/C_i)\,P(C_i)$$

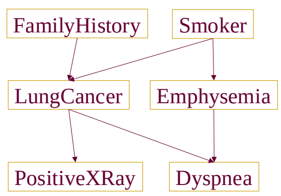
- Naïvely assumes that each x_i is independent
- We represent $P(X/C_i)$ by $P(X)$, etc. when unambiguous

Bayesian Belief Networks

- Naïve Bayes assumes independence between attributes – Not always correct!
- If we don't assume independence, the problem becomes *exponential* – every attribute can be dependent on every other attribute.
- Luckily, in real life most attributes don't depend (directly) on other attributes.
- A Bayesian network explicitly encodes dependencies between attributes.

Bayesian Belief Network

DAG



Conditional Probability Table for LungCancer

	FH,S	FH,!S	!FH,S	!FH,!S
LC	0.8	0.5	0.7	0.1
!LC	0.2	0.5	0.3	0.9

$P(X) = P(x_1 | \text{Parents}(x_1))\,P(x_2 | \text{Parents}(x_2))\ldots P(x_m | \text{Parents}(x_m))$

e.g. P(PositiveXRay, Dyspnea)

Maximum Entropy Approach

- Think emails, keywords, spam / non-spam
- Given a new data point $X=\{x_1,x_2,\ldots,x_m\}$ to classify calculate $P(C_i/X)$ for each class C_i.
- Select C_i for which $P(C_i/X)$ is maximum

$$P(C_i/X) = P(X/C_i)\,P(C_i) / P(X) \propto P(X/C_i)\,P(C_i)$$

- Naïve Bayes assumes that each x_i is independent
- Instead estimate $P(X/C_i)$ directly from training data: $support_{C_i}(X)$
- Problem: There may be no instance of X in training data.
  - Training data is usually sparse
- Solution: Estimate $P(X/C_i)$ from available features in training data: $P(Y_j/C_i)$ might be known for several Y_j

Background: Shannon's Entropy

- An expt has several possible outcomes
- In N expts, suppose each outcome occurs M times
- This means there are N/M possible outcomes
- To represent each outcome, we need log N/M bits.
  - This generalizes even when all outcomes are not equally frequent.
  - Reason: For an outcome j that occurs M times, there are N/M equi-probable events among which only one cp to j
- Since $p_i = M / N$, information content of an outcome is $-\log p_i$
- So, expected info content: $H = - \Sigma\, p_i \log p_i$

Maximum Entropy Principle

- Entropy corresponds to the disorder in a system
  - Intuition: A highly ordered system will require less bits to represent it
- If we do not have evidence for any particular order in a system, we should assume that no such order exists
- The order that we know of can be represented in the form of constraints
- Hence, we should maximize the entropy of a system subject to the known constraints
- If the constraints are consistent, there is a unique solution that maximizes entropy.

Max Ent in Classification

- Among the distributions $P(X/C_i)$, choose the one that has maximum entropy.
- Use the selected distribution to classify according to bayesian approach.

Distribution of a Dataset

- Let T_k be a bit-vector (0010110) representing the presence/absence of N items in a record; $k = (1\ldots2^N)$
  - T_k's are the possible *outcomes* of each record
- Our first task is to find $P(T_k)$ for each k
- Then, to estimate P(X), we can just add the $P(T_k)$'s for all T_k's that satisfy X

Constraints on the Distribution

- $P(Y_j) = d_j$ (say) is known for several Y_j
- Let there be M such Y_j's
- For each such Y_j, we have a constraint:

$$\Sigma_{(k:\ T_k \text{ satisfies } Y_j)} P(T_k) = d_j$$

- We also have: $\Sigma_k P(T_k) = 1$

Maximum Entropy

Find $P(T_k)$ that satisfies these constraints, while assuming nothing else about the distribution

- Maximize entropy $(-\Sigma_k P(T_k) \log P(T_k))$ subject to constraints
- Theorem: Resulting distribution can be written as a product:

$$P(T_k) = \mu_0 \prod_{\text{(constraint j satisfied by } T_k)} \mu_j$$

  - where there is a μ_j for each constraint
  - μ_0 is to ensure that $\Sigma_k P(T_k) = 1$
  - If we can compute the μ_j values, we are done.

Generalized Iterative Scaling Algorithm

\# N items, M constraints

$P(X_k) = 1 / 2^N$ // for k = (1…2^N); Uniform distribution

$\mu_j = 1$ # for j = (1…M)

while all constraints *not* satisfied:

for each constraint C_j:

$S_j = \Sigma_{(k:\ T_k \text{ satisfies } Y_j)} P(X_k)$

$\mu_j {*}= d_j / S_j$

$P(X_k) = \mu_0 \prod_{(j \text{ satisfied by } T_k)} \mu_j$

\# μ_0 is to ensure that $\Sigma_k P(X_k) = 1$

Example

- Imagine emails represented as sets of keywords and being classified as spam/non-spam.
- Let us focus on 3 keywords A, B, C.
- From the training data, let us say we get 4 "constraints" for the spam class:
  - P(A) = 0.2 = d_1 (say)
  - P(B) = 0.3 = d_2
  - P(C) = 0.1 = d_3
  - P(AB) = 0.1 = d_4
- Now, our task is to use the GIS algorithm to determine P(ABC).

- Define T_i as a bit-vector representing the presence/absence of keywords.
- E.g. T_3 = 011 represents the presence of B,C and absence of A.
- Note that P(T_3) is "not" the same as P(BC) since the latter doesn't care about the presence/absence of A.

ABC
T_0 = 000
T_1 = 001
T_2 = 010
T_3 = 011
T_4 = 100
T_5 = 101
T_6 = 110
T_7 = 111

The previous 4 constraints can be rewritten in terms of the T_i's:

P(A) = P(T_4) + P(T_5) + P(T_6) + P(T_7) = 0.2
P(B) = ... = 0.3
P(C) = ... = 0.1
P(AB) = P(T_6) + P(T_7) = 0.1
//AB is satisfied by T_6 and T_7

Define:
S_1 = P(T_4) + P(T_5) + P(T_6) + P(T_7) ----(1)
S_2 = ... ---(2)
S_3 = ... ---(3)
S_4 = P(T_6) + P(T_7) -----------------------------(4)

Start GIS with P(T_i) = 1/8 = 0.125 (for all i)
Also, set mu_1 = mu_2 = mu_3 = mu_4 = 1 and mu_0 = 1
There are 4 mu's because there are 4 constraints.

Next, we calculate S_i for each constraint i, as per (1), (2), (3), (4) above.
S_1 = 1/8 + 1/8 + 1/8 + 1/8 = 0.5
similarly, S_2 = S_3 = 0.5, S_4 = 0.25

S_1 is 0.5; but we want it to be 0.2 (since P(A) = 0.2). Since S_1 is the sum of some P(T_i)'s in equation (1), we want to reduce these P(T_i)'s. We want to scale them down by 0.2 / 0.5. So we set: mu_i *= d_i / S_i. Thus, we get:

mu_1 = 1 * 0.2 / 0.5 = 0.4
mu_2 = 1 * 0.3 / 0.5 = 0.6
mu_3 = 1 * 0.1 / 0.5 = 0.2
mu_4 = 1 * 0.1 / 0.25 = 0.4

Using these mu's, we recalculate P(T_i)'s as:
P(T_i) = product of those mu_j's whose cp constraint is satisfied by T_i
Thus:
P(T_0) = 1 // T_0 (000) doesn't satisfy any constraint
P(T_1) = mu_3 // T_1 (001) satisfies constraint 3 only
P(T_2) = mu_2
P(T_3) = mu_2 * mu_3 // T_3 (011) satisfies constraints 2 & 3
P(T_4) = mu_1
P(T_5) = mu_1 * mu_3
P(T_6) = mu_1 * mu_2 * mu_4 // T_6 (110) satisfies constraints 1, 2 & 4
P(T_7) = mu_1 * mu_2 * mu_3 * mu_4 // T_7 (111) satisfies all 4 constraints

But sum of above $P(T_i)$'s might not turn out to be 1. Infact, it turns out to be 2.5152 for this example. So we scale all of them down by 2.5152 to make the sum equal 1. Then, we get:

$P(T_0) = 1 / 2.5152 = 0.4$ (approx)
$P(T_1) = 0.08$
$P(T_2) = 0.24$
$P(T_3) = 0.048$
$P(T_4) = 0.16$
$P(T_5) = 0.032$
$P(T_6) = 0.04$
$P(T_7) = 0.008$

That was the 1st iteration of GIS. These numbers are closer to the actual $P(T_i)$ values. For example, we know that $P(A)=0.2$, $P(B)=0.3$, $P(C)=0.1$. If A,B,C are mutually exclusive, then $P(A \text{ or } B \text{ or } C) = 0.6$ (the sum). Notice that $P(T_0)$ above is 0.4 which is $1 - 0.6$. If we run the algorithm for more iterations we get better results.

Our task was to determine $P(ABC)$. So we just output the value of $P(T_7)$ above.