Internet Measurement and Data Analysis (10)

Kenjiro Cho

2012-12-05

review of previous class

Class 9 Topology and graph (11/28)

- Routing protocols
- Graph theory
- exercise: shortest-path algorithm

today's topics

Class 10 Anomaly detection and machine learning

- Anomaly detection
- Machine Learning
- SPAM filtering and Bayes theorem
- exercise: naive Bayesian filter

anomalies

- traffic problems
- routing problems, reachability problems
- DNS problems
- attacks, intrusions
- CPU load problems

causes of anomalies

- access concentration, congestion
- attacks: DoS, viruses/worms
- outages: equipment failures, circuit failures, accidents, power outages
- maintenance

anomaly detection

- avoid or reduce losses caused by service degradation or disruption
- monitoring individual items: post an alert when the monitored value exceeds the predefined threshold
  - passive monitoring
  - active monitoring
- signature based anomaly detection:
  - pattern matching with known anomalies
  - IDS: Intrusion Detection System
  - cannot detect unknown anomalies
  - need to keep the pattern database up-to-date
- anomaly detection by statistical methods:
  - detect discrepancies from normal states
  - in general, need to learn "normal" states

responses to anomalies

- report to system administrators
  - posting alert messages
- identifying types of anomalies
  - provide information to help operators to understand the cause of the problem
  - difficult to find causes, especially for statistical methods
- automated responses
  - automatically generating filtering rules, failover, etc

anomaly examples

- Flash Crowd
  - access concentration to specific services (news, events, etc)
- DoS/DDoS
  - send a large volume of traffic to a specific host
  - zombie PCs are often used as attackers
- scanning
  - for most cases, to find hosts having known security holes
- worms/viruses
  - many incidents (SQL Slammer, Code Red, etc)
- route hijacking
  - announcing someone else's prefixes (mostly by mis-configuration)

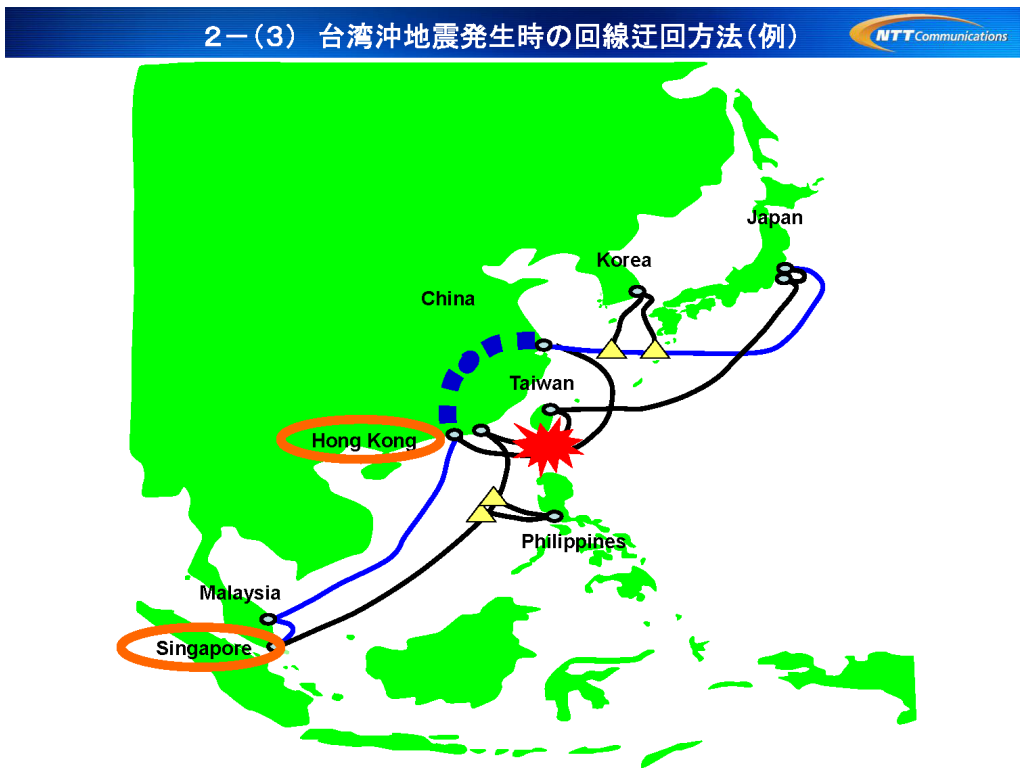
YouTube hijacked

- 2008-02-24: worldwide traffic to YouTube was redirected to Pakistan
- cause
  - by the order of Pakistan government, Pakistan Telecom announced a false prefix on BGP in order to block domestic access to YouTube
  - a large ISP, PCCW, leaked the announce to the global Internet
  - as a result, worldwide traffic to YouTube was redirected to Pakistan by the false route announcement

reference:
`http://www.renesys.com/blog/2008/02/pakistan_hijacks_youtube_1.shtmly`

communication service disruption by Taiwan earthquake

- 2006-12-26: M7.1 earthquake occurred off the coast of Taiwan
- submarine cables were damaged, communication services to/from Asia were affected
- Indonesia's international link capacity became less than 20%
- ISPs restored services by rerouting



source: JANOG26

`http://www.janog.gr.jp/meeting/janog26/doc/post-cable.pdf`

disconnection between ISPs

- a case of a dispute of 2 Tier-1 ISPs over connection fees
- in 2005, Level 3 asked Cogent to switch from non-paid peering to paid connection because of the increase in traffic
- other cases
  - in 2008, Cogent and Telia stopped peering
  - in 2008, Level 3 and Cogent stopped peering
  - in 2010, Level 3 and Comcast dispute

references:
`http://www.renesys.com/blog/2006/11/sprint-and-cogent-peer.shtml`
`http://wirelesswire.jp/Watching_World/201012011624.html`

anomaly detection by statistical methods

- time-series
- correlation
- PCA
- clustering
- entropy

machine learning

- supervised learning
  - requires training beforehand using test data
- unsupervised learning
  - automatically performs classification or pattern extraction
  - no training required
  - cluster analysis, PCA, etc

identifying and filtering SPAM email

SPAM: unsolicited bulk messages

SPAM test methods

- tests by senders
  - white lists
  - black lists
  - gray listing
- tests by content
  - bayesian spam filter: widely used
  - learns frequencies of words from SPAM and HAM email, calculate a probability for an email to be SPAM
  - the accuracy improves as it is used

conditional probability

Question:

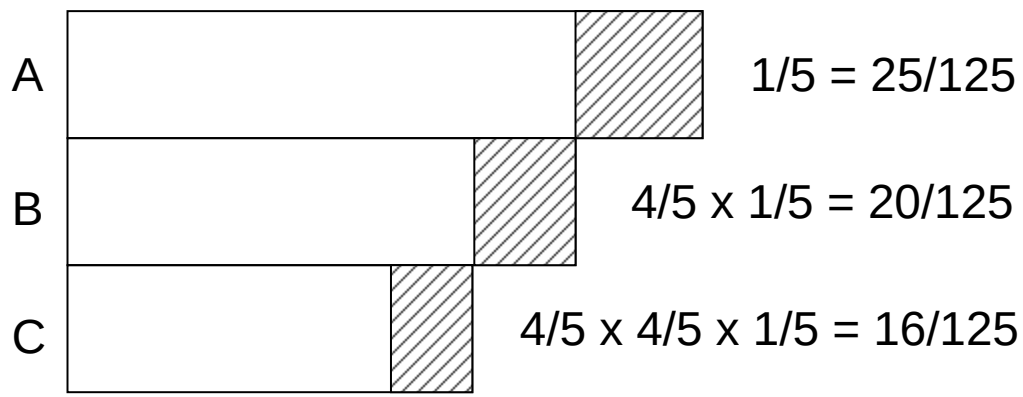
- Student K leaves behind his cap once every 5 times. He visited 3 friends, A, B and C in this order and when he came home he found his cap was left behind. What is the probability that K left his cap at B's house? (1976, Waseda University, entrance exam)

conditional probability

Question:

- Student K leaves behind his cap once every 5 times. He visited 3 friends, A, B and C in this order and when he came home he found his cap was left behind. What is the probability that K left his cap at B's house? (1976, Waseda University, entrance exam)

Answer:



the prob. of the cap left at B / the prob. of the cap left at either house $= 20/61$

Bayes' theorem

conditional probability

- the probability of B when A is known to occur: $P(B|A)$
  - the sample space is restricted to event A, within which the area $(A \cap B)$ is of interest

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

Bayes' theorem

- posterior probability: when A causes B, the probability of event A occurring given that event B has occurred: $P(A|B)$
  - $P(A)$: the probability of A to occur (prior probability)
  - $P(A|B)$: the probability of A occurring given that B has occurred (posterior probability)

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

applications of bayes' theorem

based on the observations, inferring the probability of a cause: many engineering applications

- communications: based on received signal with noise, extract original signal
- medical tests: based on a medical test result, find the probability of a person actually having the disease
- spam tests: based on the content of email, find the probability of an email being spam

example: disease test

Question:

- the population ratio having a certain disease is 50/1000. a test for the disease is known to have positive for 90% of people having the disease but also have positive for 10% of people not having the disease.
  when a person get positive by this test, what is the probability of the person actually having the disease?

example: disease test

Question:

- the population ratio having a certain disease is 50/1000. a test for the disease is known to have positive for 90% of people having the disease but also have positive for 10% of people not having the disease.
  when a person get positive by this test, what is the probability of the person actually having the disease?

Answer: the probability of the person having the disease:
$P(D) = 50/1000 = 0.05$
the probability of a result to be positive: $P(R) = P(D \cap R) + P(\bar{D} \cap R)$
when the result is positive, the posterior probability that the person has the disease

$$\begin{aligned} P(D|R) &= \frac{P(D \cap R)}{P(R)} \\ &= (0.05 \times 0.9)/(0.05 \times 0.9 + 0.95 \times 0.1) = 0.321 \end{aligned}$$

spam email tests

- for training, prepare spam messages (SPAM) and non-spam messages (HAM)
- for words often included in SPAM, compute
  - the conditional probability that SPAM include a word
  - the conditional probability that HAM include a word
- then, compute the posterior probability of an unknown message being SPAM

example: for word A, assume $P(A|S) = 0.3$, $P(A|H) = 0.01$, $\frac{P(H)}{P(S)} = 2$. then, compute $P(S|A)$.

$$\begin{aligned} P(S|A) &= \frac{P(S)P(A|S)}{P(S)P(A|S) + P(H)P(A|H)} \\ &= \frac{P(A|S)}{P(A|S) + P(A|H)P(H)/P(S)} \\ &= \frac{0.3}{0.3 + 0.01 \times 2} = 0.94 \end{aligned}$$

naive Bayesian classifier

- in practice, multiple tokens are used
  - combinations of tokens require huge data
- naive Bayesian classifier: assumes tokens are independent
  - tokens are not independent, but it works most of the cases
  - training step:
    - using classified training samples, compute the conditional probabilities of tokens being included in SPAM
  - prediction step:
    - for unknown messages, compute the posterior probabilities of tokens included in a message to decide whether the message is SPAM or HAM
- in the training step, the conditional probability of each token can be independently computed
- use Bayesian joint probability to compute the joint probability for SPAM testing from individual token's SPAM probability

naive Bayesian classifier (details)

let tokens be $x_1, x_2, \ldots, x_n$. when these tokens are observed, the posterior probability of a message being SPAM is:

$$P(S|x_1, \ldots, x_n) = \frac{P(S)P(x_1, \ldots, x_n|S)}{P(x_1, \ldots, x_n)}$$

the numerator shows the joint probability of the token to be observed and the message is SPAM, and thus, can be written as follows. by applying the definition of conditional probability:

$$\begin{aligned} P(S, x_1, \ldots, x_n) &= P(S)P(x_1, \ldots, x_n|S) \\ &= P(S)P(x_1|S)P(x_2, \ldots, x_n|S, x_1) \\ &= P(S)P(x_1|S)P(x_2|S, x_1)P(x_3, \ldots, x_n|S, x_1, x_2) \end{aligned}$$

assume each token is conditionally independent from other tokens

$$P(x_i|S, x_j) = P(x_i|S)$$

then, the above joint probability becomes

$$P(S, x_1, \ldots, x_n) = P(S)P(x_1|S)P(x_2|S)\cdots P(x_n|S) = P(S)\prod_{i=1}^{n} P(x_i|S)$$

thus, assuming tokens are independent, the posterior probability of the message being SPAM is

$$P(S|x_1, \ldots, x_n) = \frac{P(S)\prod_{i=1}^{n} P(x_i|S)}{P(S)\prod_{i=1}^{n} P(x_i|S) + P(H)\prod_{i=1}^{n} P(x_i|H)}$$

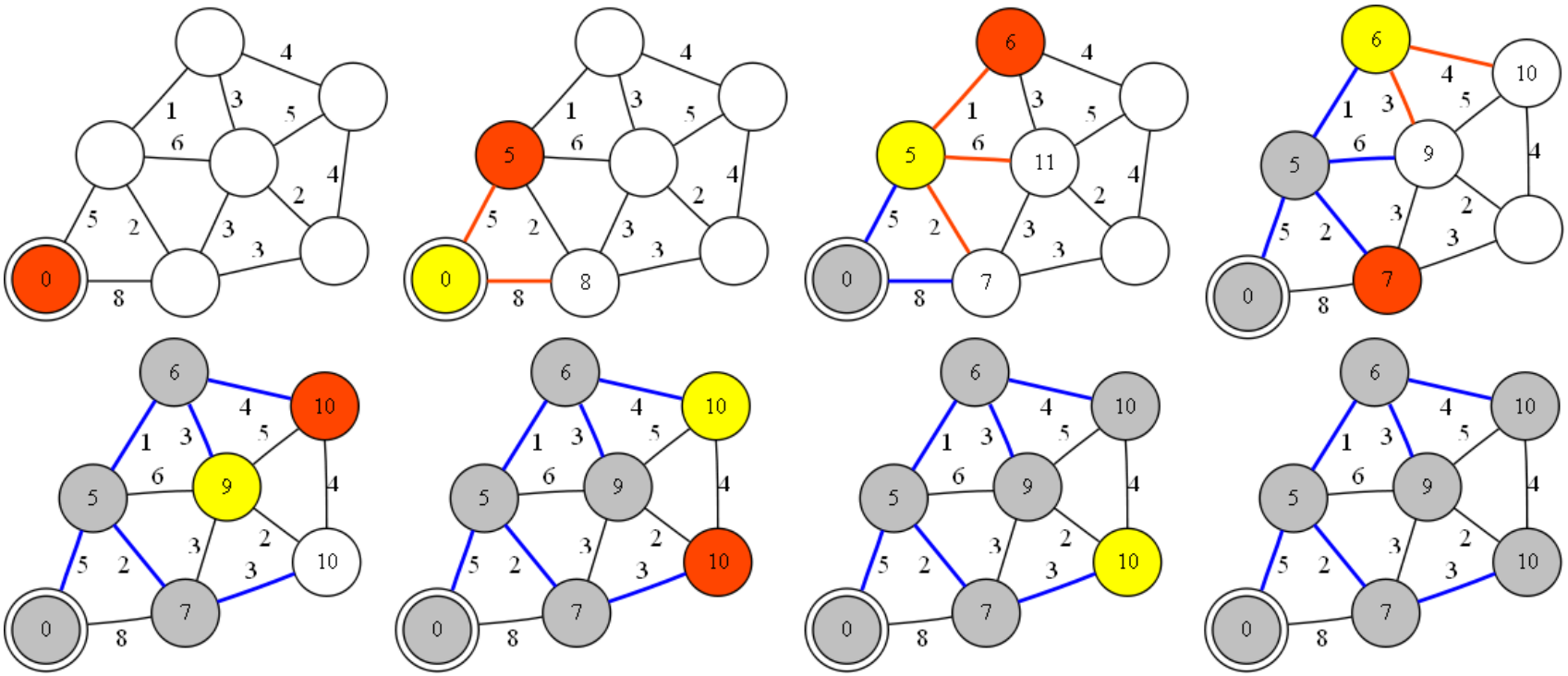
previous exercise: Dijkstra algorithm

- read a topology file, and compute shortest paths

```
% cat topology.txt
a - b 5
a - c 8
b - c 2
b - d 1
b - e 6
c - e 3
d - e 3
c - f 3
e - f 2
d - g 4
e - g 5
f - g 4
% ruby dijkstra.rb -s a topology.txt
a: (0) a
b: (5) a b
c: (7) a b c
d: (6) a b d
e: (9) a b d e
f: (10) a b c f
g: (10) a b d g
%
```

Dijkstra algorithm

1. cost initialization: start_node = 0, other_nodes = infinity
2. loop:
   (1) find the node with the lowest cost among the unfinished nodes, and fix its cost
   (2) update the cost of its neighbors



dijkstra algorithm

sample code (1/4)

```
# dijkstra's algorithm based on the pseudo code in the wikipedia
# http://en.wikipedia.org/wiki/Dijkstra%27s_algorithm
#
require 'optparse'

source = nil # source of spanning-tree

OptionParser.new {|opt|
  opt.on('-s VAL') {|v| source = v}
  opt.parse!(ARGV)
}

INFINITY = 0x7fffffff # constant to represent a large number
```

sample code (2/4)

```
# read topology file and initialize nodes and edges
# each line of topology file should be "node1 (-|->) node2 weight_val"
nodes = Array.new # all nodes in graph
edges = Hash.new # all edges in graph
ARGF.each_line do |line|
  s, op, t, w = line.split
  next if line[0] == ?# || w == nil
  unless op == "-" || op == "->"
    raise ArgumentError, "edge_type should be either '-' or '->'"
  end
  weight = w.to_i
  nodes << s unless nodes.include?(s) # add s to nodes
  nodes << t unless nodes.include?(t) # add t to nodes
  # add this to edges
  if (edges.has_key?(s))
    edges[s][t] = weight
  else
    edges[s] = {t=>weight}
  end
  if (op == "-")  # if this edge is undirected, add the reverse directed edge
    if (edges.has_key?(t))
      edges[t][s] = weight
    else
      edges[t] = {s=>weight}
    end
  end
end
# sanity check
if source == nil
  raise ArgumentError, "specify source_node by '-s source'"
end
unless nodes.include?(source)
  raise ArgumentError, "source_node(#{source}) is not in the graph"
end
```

sample code (3/4)

```
# create and initialize 2 hashes: distance and previous
dist = Hash.new # distance for destination
prev = Hash.new # previous node in the best path
nodes.each do |i|
  dist[i] = INFINITY # Unknown distance function from source to v
  prev[i] = -1 # Previous node in best path from source
end

# run the dijkstra algorithm
dist[source] = 0 # Distance from source to source
while (nodes.length > 0)
  # u := vertex in Q with smallest dist[]
  u = nil
  nodes.each do |v|
    if (!u) || (dist[v] < dist[u])
      u = v
    end
  end
  if (dist[u] == INFINITY)
    break # all remaining vertices are inaccessible from source
  end
  nodes = nodes - [u] # remove u from Q
  # update dist[] of u's neighbors
  edges[u].keys.each do |v|
    alt = dist[u] + edges[u][v]
    if (alt < dist[v])
      dist[v] = alt
      prev[v]  = u
    end
  end
end
```

sample code (4/4)

```
# print the shortest-path spanning-tree
dist.sort.each do |v, d|
  print "#{v}: " # destination node
  if d != INFINITY
    print "(#{d}) " # distance
    # construct path from dest to source
    i = v
    path = "#{i}"
    while prev[i] != -1 do
      path.insert(0, "#{prev[i]} ") # prepend previous node
      i = prev[i]
    end
    puts "#{path}" # print path from source to dest
  else
    puts "unreachable"
  end
end
```

today's exercise: SPAM filtering

- SPAM filtering using naive bayesian classifier
  - based on the code from "Programming Collective Intelligence" Chapter 6

```
% ruby naivebayes.rb
classifying "quick rabbit" => good
classifying "quick money" => bad
```

naive bayesian classifier for the exercise

compute the propbability of a document to be classified into a specific category by words appearing in the dicument

$$P(C)\prod_{i=1}^{n} P(x_i|C)$$

- $P(C)$: the probability of the category
- $\prod_{i=1}^{n} P(x_i|C)$: product of the conditional probability of each word in the category

select the category with the highest probability

- threshold： the probability of the best category should be *thresh* times higher than that of the second best category

SPAM classifier script

- training and classifier

```
# create a classifier instance
cl = NaiveBayes.new

# training
cl.train('Nobody owns the water.','good')
cl.train('the quick rabbit jumps fences','good')
cl.train('buy pharmaceuticals now','bad')
cl.train('make quick money at the online casino','bad')
cl.train('the quick brown fox jumps','good')

# classify
sample_data = [ "quick rabbit", "quick money" ]

sample_data.each do |s|
  print "classifying \"#{s}\" => "
  puts cl.classify(s, default="unknown")
end
```

script: Classifier Class (1/2)

```
# feature extraction
def getwords(doc)
  words = doc.split(/\W+/)
  words.map!{|w| w.downcase}
  words.select{|w| w.length < 20 && w.length > 2 }.uniq
end

# base class for classifier
class Classifier
  def initialize
    # initialize arrays for feature counts, category counts
    @fc, @cc = {}, {}
  end

  def getfeatures(doc)
    getwords(doc)
  end

  # increment feature/category count
  def incf(f, cat)
    @fc[f] ||= {}
    @fc[f][cat] ||= 0
    @fc[f][cat] += 1
  end

  # increment category count
  def incc(cat)
    @cc[cat] ||= 0
    @cc[cat] += 1
  end

  ...
```

script: Classifier Class (2/2)

```
  def fprob(f,cat)
    if catcount(cat) == 0
      return 0.0
    end

    # the total number of times this feature appeared in this
    # category divided by the total number of items in this category
    Float(fcount(f, cat)) / catcount(cat)
  end

  def weightedprob(f, cat, weight=1.0, ap=0.5)
    # calculate current probability
    basicprob = fprob(f, cat)
    # count the number of times this feature has appeared in all categories
    totals = 0
    categories.each do |c|
      totals += fcount(f,c)
    end
    # calculate the weighted average
    ((weight * ap) + (totals * basicprob)) / (weight + totals)
  end

  def train(item, cat)
    features = getfeatures(item)
    features.each do |f|
      incf(f, cat)
    end
    incc(cat)
  end
end
```

script: NaiveBayes Class

```
# naive baysian classifier
class NaiveBayes < Classifier
  def initialize
    super
    @thresholds = {}
  end

  def docprob(item, cat)
    features = getfeatures(item)
    # multiply the probabilities of all the features together
    p = 1.0
    features.each do |f|
      p *= weightedprob(f, cat)
    end
    return p
  end

  def prob(item, cat)
    catprob = Float(catcount(cat)) / totalcount
    docprob = docprob(item, cat)
    return docprob * catprob
  end

  def classify(item, default=nil)
    # find the category with the highest probability
    probs, max, best = {}, 0.0, nil
    categories.each do |cat|
      probs[cat] = prob(item, cat)
      if probs[cat] > max
        max = probs[cat]
        best = cat
      end
    end
    # make sure the probability exceeds threshold*next best
```

summary

Class 10 Anomaly detection and machine learning

- Anomaly detection
- Machine Learning
- SPAM filtering and Bayes theorem
- exercise: naive Bayesian filter

next class

Class 11 Data Mining (12/12)

- Pattern extraction
- Classification
- Clustering
- exercise: clustering