An Introduction to Machine Learning and Natural Language Processing Tools

Presented by:
Mark Sammons, Vivek Srikumar
(Many slides courtesy of Nick Rizzolo)

8/24/2010 - 8/26/2010

Some reasonably reliable facts…*

- 1.7 ZB (= 10^21 bytes) of new digital information added worldwide in 2010
- 95% of this is unstructured (e.g. not database entries)
- 25% is images
- 6EB is email (1,000 EB = 1 ZB)
- How to manage/access the tiny relevant fraction of this data?

*Source: Dr. Joseph Kielman, DHS: projected figures taken from presentation in Summer 2010

Keyword search is NOT the answer to every problem…

- Abstract/Aggregative queries:
  - E.g. find news reports about visits by heads of state to other countries
  - E.g. find reviews for movies that some person might like, given a couple of examples
- Enterprise Search:
  - Private collections of documents, e.g. all the documents published by a corporation
  - Much lower redundancy than web documents
  - Need to search for *concepts*, not *words*
  - e.g. looking for proposals with similar research goals: wording may be very different (different scientific disciplines, different emphasis, different methodology)

Even when keyword search is a good start…

- Waaaaay too many documents that match the key words
- Solution: Filter data that is irrelevant (for task in hand)
- Some examples…
  - Different languages
  - Spam vs. non-spam
  - Forum/blog post topic
- How to solve the Spam/non-Spam problem? Suggestions?

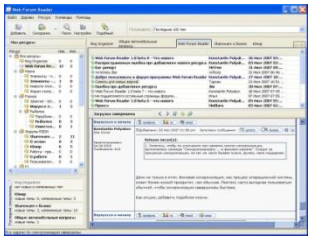
Machine Learning could help…

- Instead of writing/coding rules ("expert systems"), use statistical methods to "learn" rules that perform a classification task, e.g.
  - given a blog post, which of N different topics is it most relevant to?
  - Given an email, is it spam or not?
  - Given a document, which of K different languages is it in?

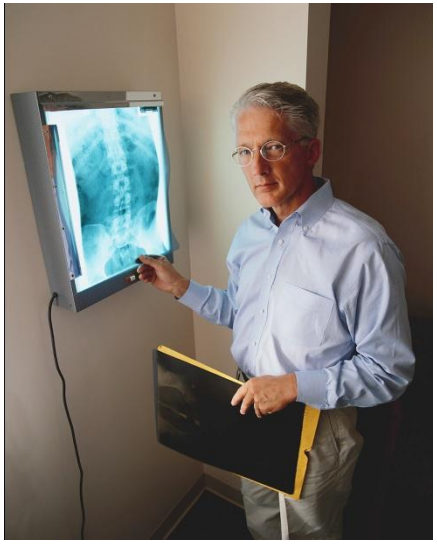
- … and now, a demonstration…

- The demonstration shows what a well-designed classifier can achieve. Here's a very high-level view of how classifiers work in the context of NLP.

Motivating example: blog topics



Blog crawl



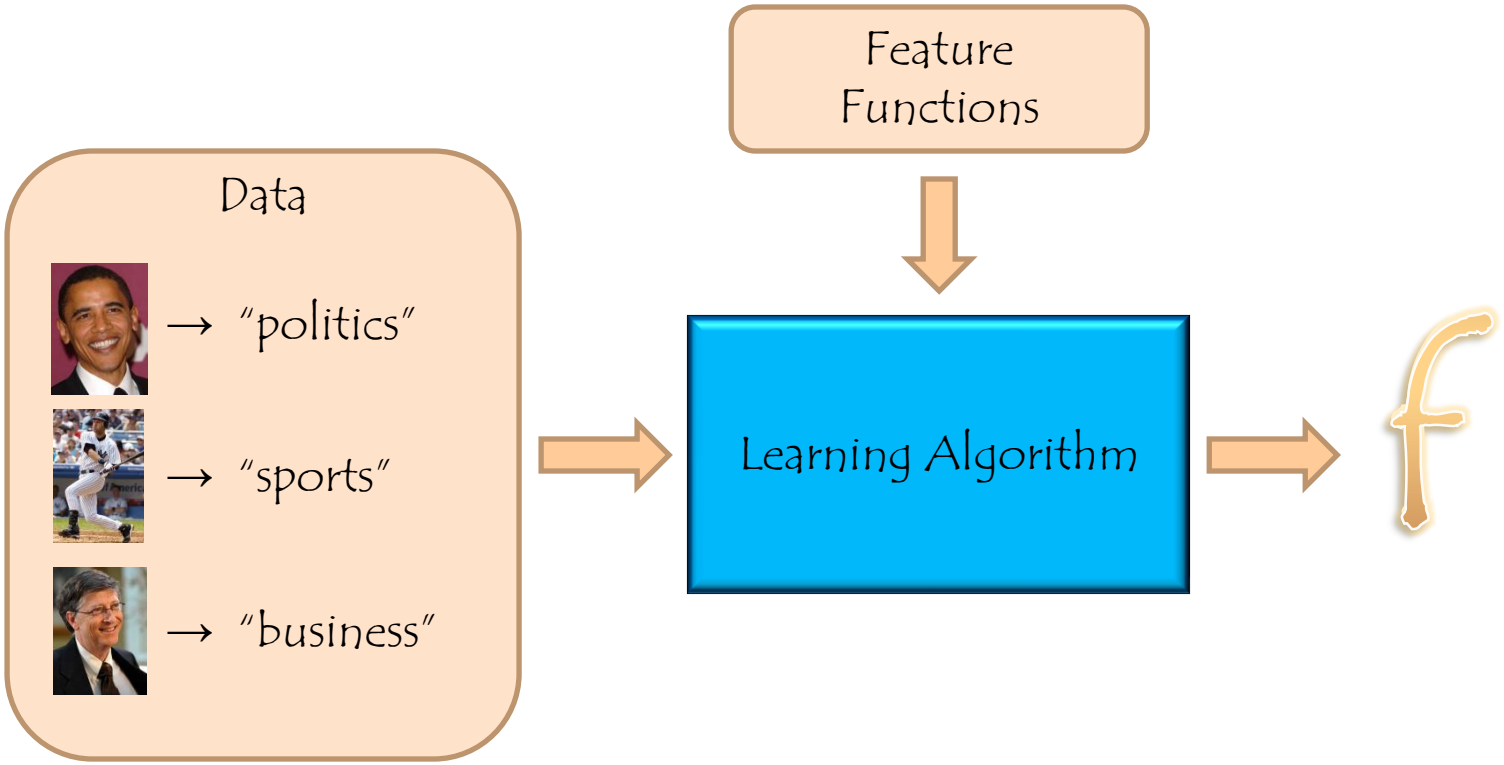
What we need:

f() = "politics"

f() = "sports"

f() = "business"

Where to get it: Machine Learning



So, what are “feature functions”?

- Take same input as
- Indicate some property of the input *a.k.a., a feature*

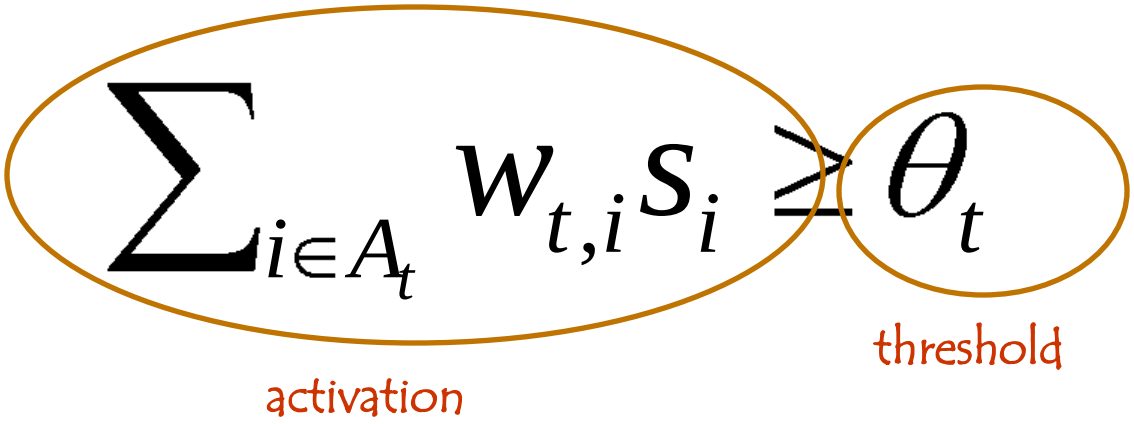
- Typical NLP feature functions
  - Binary
    - Appearance of a given word
    - Appearance of two words consecutively *a.k.a., a bigram*
    - Appearance of a word with a given part of speech
    - Appearance of a named entity (e.g. *“Barack Obama”*)
  - Real
    - Counts of binary features
    - TFIDF (a statistical measure of a document)

What does the Learning Algorithm do?

- Training input: a feature-based representation of examples, together with the labels we want to be able to predict
  - E.g. 1,000 email feature sets labeled either "spam" or "non-spam"
  - Each feature set is extracted from an email using the feature functions
  - Labels are typically assigned by human annotator
- Computes statistics over features, relating features to labels
- Generates a classifier (statistical model) that predicts a label based on the features that are active.
  - E.g. if feature "word-'VIAGRA'-is-present" is active, predict "SPAM"
- Training output: the classifier. Now it will take feature representations of new emails (no label!) and predict a label.
  - Sometimes, it will be wrong!

Update Rules

- Decision rule: Linear Threshold Function



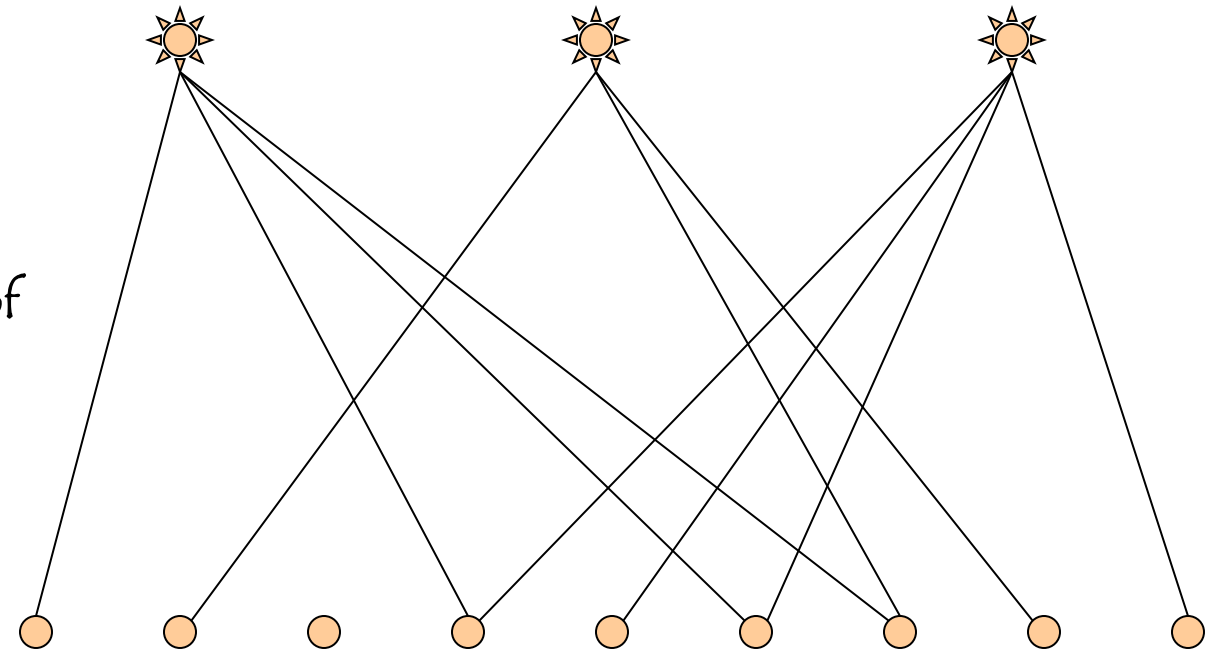
- Winnow – mistake driven update rules:
  - Promotion: if $\sum_{i \in A_t} w_{t,i} s_i < \theta_t$, $\forall i \in A_t, w_{t,i} \leftarrow w_{t,i} \cdot \alpha_t^{s_i}$
  - Demotion: if $\sum_{i \in A_t} w_{t,i} s_i \geq \theta_t$, $\forall i \in A_t, w_{t,i} \leftarrow w_{t,i} \cdot \beta_t^{s_i}$

One Basic System: One vs. All Linear Threshold Function

Targets (concepts)

Weighted edges, instead of weight vectors

Features



- Prediction is "one vs. all"

A Training Example – 3 Newsgroups

Health, disease, review:

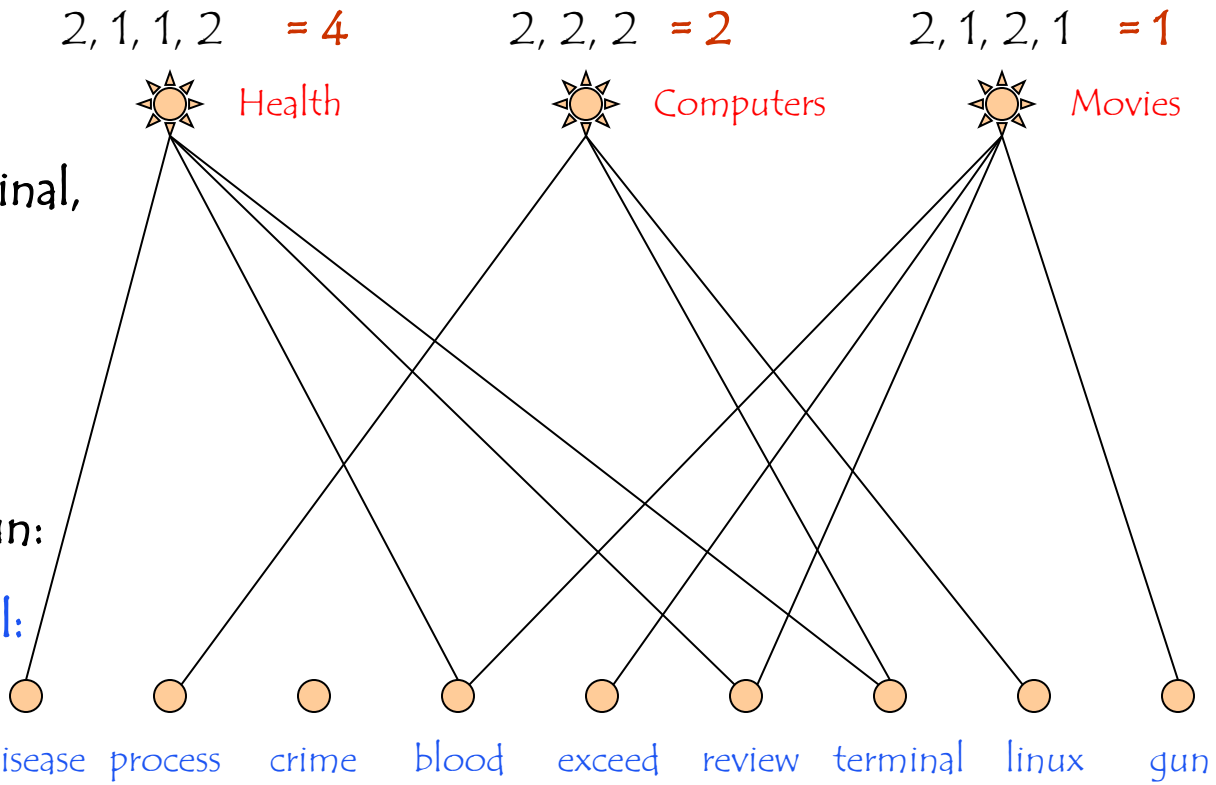
Computers, process, terminal, linux:

Health, blood, terminal:

Movies, blood, review:

Movies, blood, exceed, gun:

?, disease, exceed, terminal:



Update rule: Winnow

$\alpha = 2,\ \beta = 1/2,\ \theta = 3.5$

A Training Example, abstracted…

1, 1001, 1006:

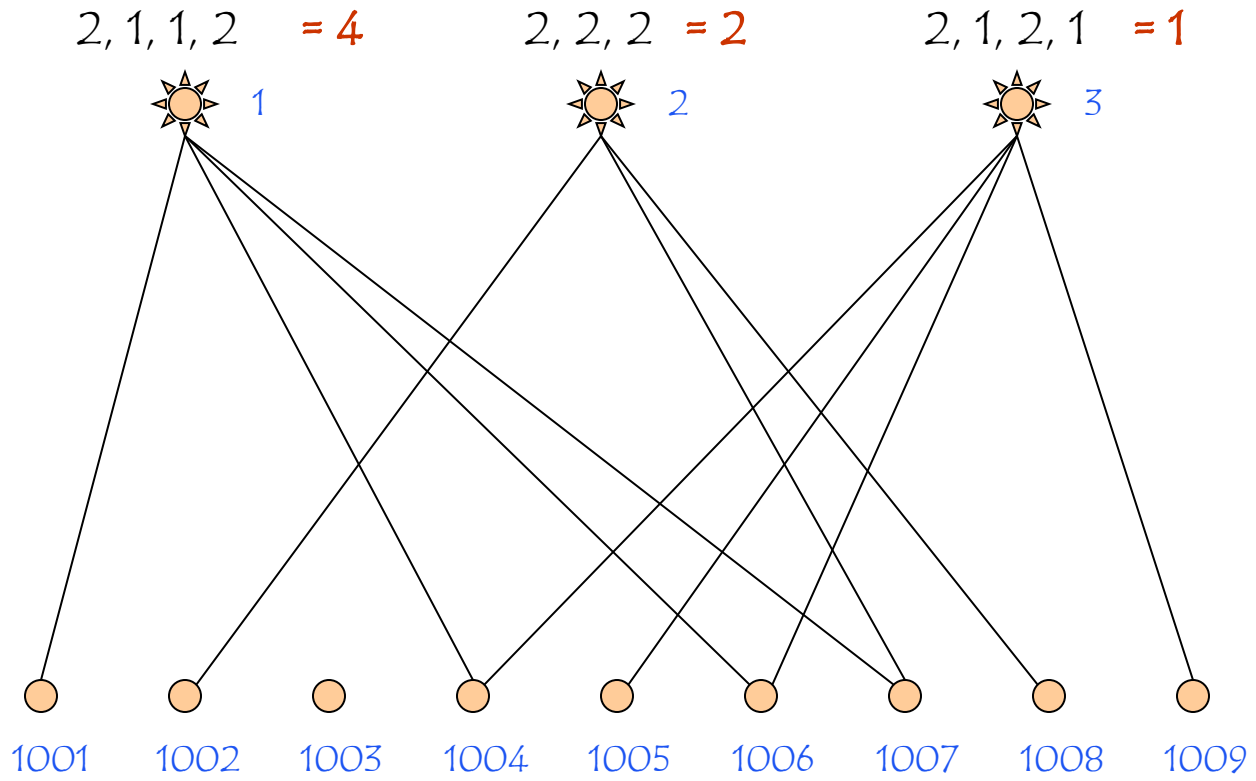
2, 1002, 1007, 1008:

1, 1004, 1007:

3, 1006, 1004:

3, 1004, 1005, 1009:

1001, 1005, 1007:



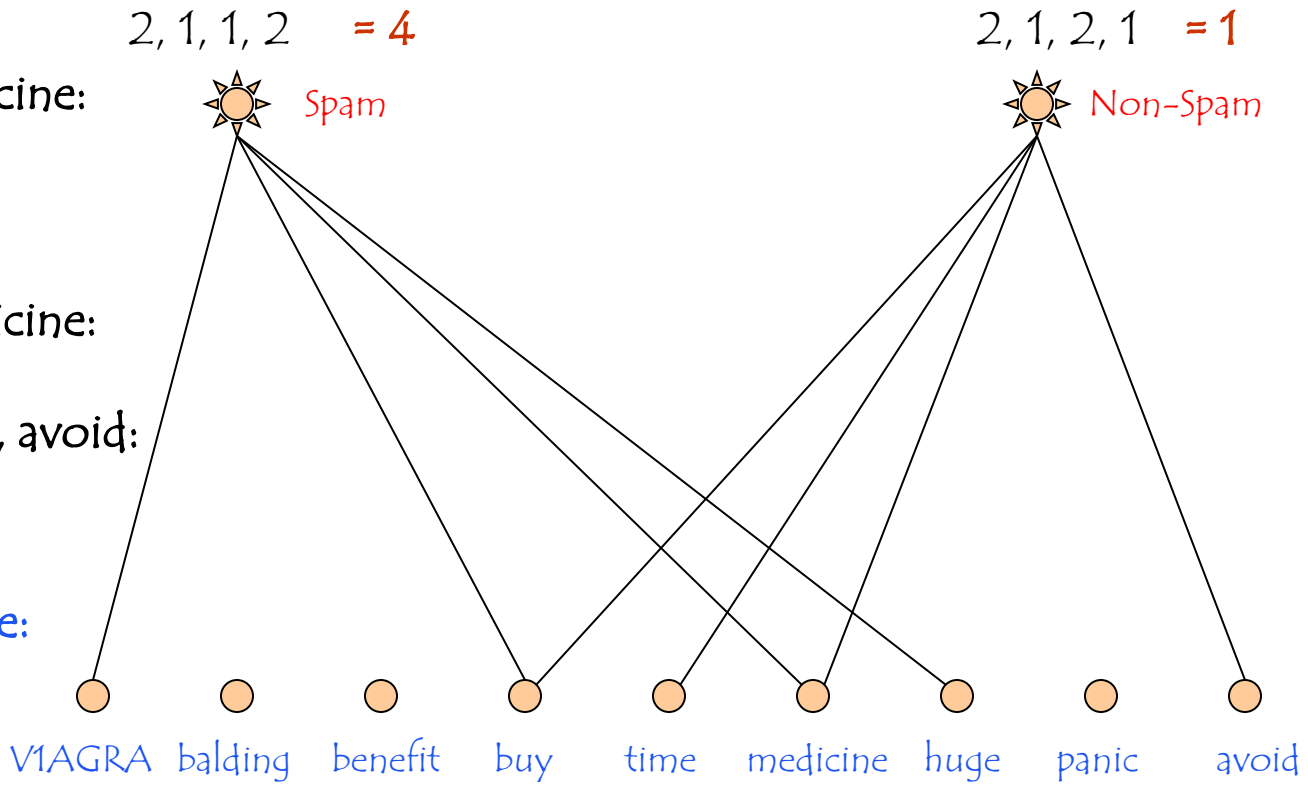
Update rule: Winnow

$\alpha = 2,\ \beta = 1/2,\ \theta = 3.5$

Adapting to New Task: Spam Filtering

- What if we want to learn Spam vs. Non-Spam?
- Demonstration showed that the same black box can be adapted to multiple problems… so what happens internally?
- Feature functions are generic pattern extractors…
  - for a new set of documents, new features will be extracted
  - E.g. for Spam, we'd expect to see some features like "word-'VIAGRA'-is-present"
- New documents come with their own set of labels (again, assigned by human annotators)
- So we reuse the same code, but generate a new classifier…

A Training Example – Spam Filtering



Spam, V1AGRA, medicine:

Spam, buy, huge:

Non-Spam, buy, medicine:

Non-Spam, buy, time, avoid:

?, V1AGRA, time, huge:

Update rule: Winnow

$\alpha = 2,\ \beta = 1/2,\ \theta = 3.5$

Some Analysis…

- We defined a very generic feature set – 'bag-of-words'
- We did reasonably well on three different tasks
- Can we do better on each task?
- …of course. If we add good feature functions, the learning algorithm will find more useful patterns.
- Suggestions for patterns for…
  - Spam filtering?
  - Newsgroup classification?
  - …are the features we add for Spam filtering good for Newsgroups?
  - When we add specialized features, are we "cheating"?
- In fact, a lot of time is usually spent engineering good features for individual tasks. It's one way to add domain knowledge.

A Caveat

- It's often a lot of work to learn to use a new tool set
- It can be tempting to think it would be easier to just implement what you need yourself
  - Sometimes, you'll be right
  - But probably not this time
- Learning a tool set is an investment: payoff comes later
  - It's easy to add new functionality – it may already be a method in some class in a library; if not, there's infrastructure to support it
  - You will avoid certain errors: someone already made them and coded against them
  - Probably, it's a lot more work than you think to DIY

Homework (!)

To prepare for tomorrow's tutorial, you should:

- Log in to the DSSI server via SSH
- Check that you can transfer a file to your home directory from your laptop
- If you have any questions, ask Tim or Yuancheng:
  - Tim: weninge1@illinois.edu
  - Yuancheng: ytu@illinois.edu

- Bring your laptop to the tutorial!!!