Learning on the Web

Tong Zhang

Rutgers University
Machine Learning Problems on Web

- Classification
- Ranking
- User Behavior Modeling
- Recommendation
- Community Analysis
- Quality Assessment
- Exploration Exploitation
- Scalability
- ...
Classification
Ranking
User Behavior Modeling
Recommendation
Community Analysis
Quality Assessment
Exploration Exploitation
Scalability
...
**Classification**

- **Electronic Spam**
  - email spam: unwanted email
  - webpage spam: low-quality pages to be placed high
  - blog spam: random blog pages to promote other pages
  - click spam: misleading clicks of ads or webpages
  - text messaging spam: unwanted text messages
  - usually aim for commercial gains

- **Sentiment analysis**

- **Webpage classification**

- **Query classification**

- ...
Classification

- Electronic Spam
  - email spam: unwanted email
  - webpage spam: low-quality pages to be placed high
  - blog spam: random blog pages to promote other pages
  - click spam: misleading clicks of ads or webpages
  - text messaging spam: unwanted text messages
  - usually aim for commercial gains

- Sentiment analysis

- Webpage classification

- Query classification

- ...

Key issues:
- problem formulation; feature generation; information aggregation; model adaptation
From: 丘先生 <xlma@zjip.com>
Subject: 出售：报销，做帐，票据
Date: July 25, 2012 9:43:08 PM GMT+08:00
Reply-To: <30910@sohu.com>

出售,报销，做帐，票据

餐饮，住宿，咨询，服务，培训，运输，建筑，租赁，广告，商品销售，设计，.......做帐报销;票据。
请加QQ：2297845601 ， 请电丘先生：13715362114
（可加宏）内容（可加宏）
Two ipad reviews: do they like or dislike the product?

<table>
<thead>
<tr>
<th>Review 1</th>
<th>Review 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>This is my first iPad and I just absolutely love it!!! I previously owned a tablet but this by far, beats the tablet I had!! It is so easy to use and the retina is amazing! I now understand why people love their iPad! ...</td>
<td>Poor quality control. Found the corners at the edges where the screen meets the body, to be crimped on each side...</td>
</tr>
</tbody>
</table>

free text is referred to as unstructured data
### Sentiment Analysis

Two ipad reviews: do they **like** or **dislike** the product?

<table>
<thead>
<tr>
<th>Review 1</th>
<th>Review 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>This is my first iPad and I just absolutely <strong>love</strong> it!!! I previously owned a tablet but this by far, beats the tablet I had!! It is so <strong>easy to use</strong> and the retina is amazing! I now understand why people love their iPad! ...</td>
<td><strong>Poor</strong> quality control. Found the corners at the edges where the screen meets the body, to be crimped on each side...</td>
</tr>
</tbody>
</table>

free text is referred to as unstructured data
**Structured Data Example**

A table or relational database

<table>
<thead>
<tr>
<th>Gender</th>
<th>Systolic BP</th>
<th>Weight</th>
<th>Disease Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>175</td>
<td>65</td>
<td>3</td>
</tr>
<tr>
<td>F</td>
<td>141</td>
<td>72</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>F</td>
<td>160</td>
<td>59</td>
<td>2</td>
</tr>
</tbody>
</table>

*Figure: Example of Medical Data Prediction*
Structured versus Unstructured Data

- **Structured data:**
  - table or spreadsheet
  - relational database with well-defined attributes (features)
  - features are usually dense

- **Unstructured data**
  - free format text
  - without well-defined attributes
Structured versus Unstructured Data

- **Structured data:**
  - table or spreadsheet
  - relational database with well-defined attributes (features)
  - features are usually dense

- **Unstructured data**
  - free format text
  - without well-defined attributes

- **Learning:** extract information, find patterns, organize contents
  - encode desired information into unknown labels to predict (output).
  - encode available unstructured data into sparse feature vector
  - combine structured and unstructured data
Goal: represent text by a feature vector
Method: vector space model
- create dictionary of size $m$ consisted of all words
- map each document into an $m$-dimensional vector
  - the $i$-th component is the frequency of word $i$ in the document
  - feature vector is very sparse and high dimensional

Bag-of-words (BoW): represent text without word ordering info
Improvements
- can preserve section or partial position information.
- can combine multiple dictionaries and use phrases
### Bag of Word Document Representation

<table>
<thead>
<tr>
<th>term</th>
<th>word1</th>
<th>word2</th>
<th>word3</th>
<th>word4</th>
<th>word5</th>
<th>...</th>
<th>wordN</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td>...</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>...</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>5</td>
<td>1</td>
<td>...</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>...</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>...</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>...</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>...</td>
<td>2</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>...</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

**Figure:** Document BoW Representation
Term Weighting

- Modify each word count by the perceived importance of the word

$$\text{tf-idf}(j, d) = \text{tf}(j, d) \times \text{idf}(j)$$

$$\text{idf}(j) = \log \left( \frac{\text{number of documents}}{\text{df}(j)} \right)$$

$$\text{tf}(j, d)$$: term frequency of token $j$ in document $d$
Term Weighting

- Modify each word count by the perceived importance of the word
- Rare words carry more information than common words
- TFIDF weighting of token $j$ in document $d$:

$$\text{tf-idf}(j, d) = \text{tf}(j, d) \times \text{idf}(j)$$

$$\text{idf}(j) = \log \left( \frac{\text{number of documents}}{\text{df}(j)} \right)$$

- $\text{tf}(j, d)$: term frequency of token $j$ in document $d$
- $\text{df}(j)$: frequency of documents containing term $j$
### Feature representation
- bag-of-word binary feature representation of email text without TFIDF
- using known spam host as nontext features

### Text feature versus nontext feature
- text feature: sparse BoW representation, linear classifier works well
- nontext feature: dense and heterogeneous, often needs non-linear interaction

<table>
<thead>
<tr>
<th>text:title</th>
<th>text:body</th>
<th>nontext</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>cheap</td>
<td>enlargement</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>yes</td>
<td>yes</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>no</td>
<td>yes</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>no</td>
<td>no</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Webpage Classification

Problem: determine the topics of a webpage
- does it talks about arts, finance, sports?
- is it a personal homepage, university department page, etc?

Features:
- text (BoW)
- HTML tag
- url
- page layout and images
- links

How to combine features
- integrate different information source into a unified feature representation
- propagate features or class labels through links

Modify standard algorithms
Classify each query into a tree-structured taxonomy
- Apparel and Jewelry/Shoes/Womens Shoes
- Mass Merchants/Baby Products
- ...
Classify each query into a tree-structured taxonomy

- Apparel and Jewelry/Shoes/Womens Shoes
- Mass Merchants/Baby Products
- ...

Challenges

- Large scale: approximately 6000 nodes
- Difficulty:
  - queries are brief: average 2.4 to 2.7 words per query
  - query words alone don’t provide sufficient information for good query classification

- Solution: employ auxiliary knowledge to augment the queries
Classify each query into a tree-structured taxonomy
- Apparel and Jewelry/Shoes/Womens Shoes
- Mass Merchants/Baby Products
- ...

Challenges
- Large scale: approximately 6000 nodes
- Difficulty:
  - queries are brief: average 2.4 to 2.7 words per query
  - query words alone don’t provide sufficient information for good query classification
- Solution: employ auxiliary knowledge to augment the queries

Auxiliary Knowledge
- Send query to a major search engine
- Augment the query using top pages returned by the search engine.
- Remedy the problem of query brevity:
  - words contained in top results pages reveal the category
Query: nikon
Top search result pages contain: camera, photography, lens, ...
These augmented words imply “Digital Camera” as a category.
Can provide matching ads about digital cameras.
Search based Query Classification

- Notations:
  - \( q \): query, \( p \): web-page, \( C = \{ C_j \} \): set of categories

- Problem: given query \( q \), want to find \( s(q, C_j) \)
  - \( s(q, C_j) \) is the quality score of query \( q \) belonging to category \( C_j \).
Search based Query Classification

- Notations:
  - \( q \): query, \( p \): web-page, \( C = \{ C_j \} \): set of categories

- Problem: given query \( q \), want to find \( s(q, C_j) \)

- Information source:
  - top-search results containing pages \( p_1, \ldots, p_k \) with high relevance to \( q \)
  - \( s(p, C_j) \): quality score of page \( p \) belonging to category \( C_j \)

\( s(q, C_j) \) is the quality score of query \( q \) belonging to category \( C_j \).
Search based Query Classification

Notations:
- \( q \): query,
- \( p \): web-page,
- \( C = \{ C_j \} \): set of categories

Problem: given query \( q \), want to find \( s(q, C_j) \)
- \( s(q, C_j) \) is the quality score of query \( q \) belonging to category \( C_j \).

Information source:
- top-search results containing pages \( p_1, \ldots, p_k \) with high relevance to \( q \)
- \( s(p, C_j) \): quality score of page \( p \) belonging to category \( C_j \)
  known through a separate web-page classifier.

Information aggregation:
- voting:

\[
 s(q, C_j) = \frac{\sum_{i=1}^{k} s(p_i, C_j)}{k},
\]

where \( p_i \) is the \( i \)-th ranked page for query \( q \).
- several other methods
Performance Evaluation

- small number of positive examples, most data are negative
- precision, recall, and F-measure

\[
\text{precision} = \frac{\text{number of correct positive predictions}}{\text{number of positive predictions}},
\]

\[
\text{recall} = \frac{\text{number of correct positive predictions}}{\text{number of positive class documents}},
\]

\[
F\text{-measure} = \frac{2}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}}
\]
Effect of using Search Results

![Graph showing precision and recall for different search engines: Baseline, Engine A full-page, Engine A summary, Engine B full-page, Engine B summary. The graph indicates that Engine B summary has the highest precision and recall compared to other options.]
Ranking

- Rank a set of items and display to users in corresponding order.
- Important in web-search:
  - web-page ranking
    - display ranked pages for a query
  - query-refinement and spelling correction
    - display ranked suggestions and candidate corrections
  - web-page summary
    - display ranked sentence segments
- related: crawling/indexing:
  - which page to crawl first
  - pages to keep in the index: priority/quality
Web-Search Problem

- User types a query, search engine returns a result page:
  - select pages from billions of pages.

- Method: given a query
  - search engine assign a relevance score for each page
  - return pages ranked by the scores.

- Quality of search engine:
  - relevance (whether returned pages are on topic and authoritative)
  - presentation issues (diversity, perceived relevance, etc)
  - personalization (predict user specific intention)
  - coverage (size and quality of index).
  - freshness (whether contents are timely).
  - responsiveness (how quickly search engine responds to the query).
  - ...

T. Zhang (Rutgers)
Web-Search Problem

- User types a query, search engine returns a result page:
  - select pages from billions of pages.
- Method: given a query
  - search engine assign a relevance score for each page
  - return pages ranked by the scores.
- Quality of search engine:
  - relevance (whether returned pages are on topic and authoritative)
  - presentation issues (diversity, perceived relevance, etc)
  - personalization (predict user specific intention)
  - coverage (size and quality of index).
  - freshness (whether contents are timely).
  - responsiveness (how quickly search engine responds to the query).
  - ...
Web-Search Ranking: Notations

Notation:
- $q$: query
- $p$: webpage
- $y(p, q)$: true relevance of page $p$ to query $q$ rated by human
- $x(p, q)$: search engine creates a feature for page $p$ and query $q$
- $f(x(p, q))$: search engine assigns a quality score $f(x(p, q))$

Web-search process:
- user input query $q$
- search engine returns page $p$ ordered by highest scores $f(x(p, q))$
Training:
- randomly select queries $q$, and web-pages $p$ for each query.
- use editorial judgment to assign relevance grade $y(p, q)$.
- construct a feature $x(p, q)$ for each query/page pair.
- learn scoring function $\hat{f}(x(p, q))$ to preserve the order of $y(p, q)$ for each $q$.

Deployment:
- query $q$ comes in.
- return pages $p_1, \ldots, p_m$ in descending order of $\hat{f}(x(p, q))$. 
Measuring Ranking Quality

- Given scoring function $\hat{f}$, return ordered page-list $p_1, \ldots, p_m$ for a query $q$.
- only the order information is important.
- should focus on the relevance of returned pages near the top.
- DCG (discounted cumulative gain) with decreasing weight $c_i$

\[
\text{DCG}(\hat{f}, q) = \sum_{j=1}^{m} c_i r(p_j, q).
\]

- $c_i$: reflects effort (or likelihood) of user clicking on the $i$-th position.
The quality of ranking only depends on the relative order of \( \{ f(x(p_i, q)) : i \} \) for each query \( q \)

Preference relationship: if \( y(p_i, q) < y(p_j, q) \)

\[ x(p_i, q) \prec x(p_j, q) \]

\( p_j \) is more relevant than \( p_i \) for query \( q \)

Pairwise preference learning

- learn a scoring function \( f \) for items to preserve preference \( \prec \).
- two items \( x \) and \( x' \): \( f(x) < f(x') \) when \( x \prec x' \).
- ordering inputs according to \( f(x) \).
Example Loss Function for Preference Learning

Training data: query-url features $x_i$ for $i = 1, \ldots, n$

- $i \prec j$ if url of $x_j$ is more relevant than url of $x_i$ for a certain query $q$.
- Let $S$ be the indices of preference relationships $i \prec j$
- Let $f(X) = [f(x_1), \ldots, f(x_n)]$

Example loss:

$$
\mathcal{R}(f(X)) = \sum_{\{i \prec j\} \in S} \max(0, 1 + f(x_i) - f(x_j))^2
$$
Let $f(x) = 0$
Iterate $t = 1, 2, \ldots$
  - For each $i = 1, \ldots, n$, compute
    \[
    r_i = \frac{\partial}{\partial f_i} R(f) = 2 \sum_{\{i < k\} \in S} \max(0, 1 + f(x_i) - f(x_k)) - 2 \sum_{\{k < i\} \in S} \max(0, 1 + f(x_k) - f(x_i))
    \]
  - Find decision tree $g_t$ that approximately minimizes
    \[
    \min \sum_{i=1}^{n} \|g(x_i) - r_i\|_2^2
    \]
    using a regression tree algorithm.
  - Pick $\eta_t$ and let
    \[
    f(x) \leftarrow f(x) - \eta_t g_t(x)
    \]
An Application of Preference Learning in Web-search

- A Web-search dataset: determine the relevancy of (query, url) pair
- GBrank: boosted tree based on preference learning
- GBDT: boosted tree based on regression
- RankSVM: SVM based on preference learning

**Table:** Precision at $K\%$ for GBrank, GBT, and RankSVM

<table>
<thead>
<tr>
<th>%K</th>
<th>GBrank</th>
<th>GBDT</th>
<th>RankSVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>10%</td>
<td>0.9867</td>
<td>0.9243</td>
<td>0.8524</td>
</tr>
<tr>
<td>20%</td>
<td>0.9722</td>
<td>0.8833</td>
<td>0.8152</td>
</tr>
<tr>
<td>50%</td>
<td>0.8638</td>
<td>0.7814</td>
<td>0.7357</td>
</tr>
<tr>
<td>100%</td>
<td>0.7225</td>
<td>0.6742</td>
<td>0.6465</td>
</tr>
</tbody>
</table>
Summary

- Many machine learning problems on the web
- Many information sources
- Challenges:
  - how to formulate the problems
  - how to generate features
  - how to aggregate information
  - how to adapt learning models
  - how to control data quality
  - how to evaluate performance
  - how to handle large scale computing
  - ...