Predictive Methods for Text Mining

Tong Zhang

Yahoo Inc.! New York City

Motivation for Text Mining

- Electronic text documents are abundant:
  - on the internet, inside an organization, personal files, etc.
  - emails, web pages, word documents, pdf files, etc.
- Efficiently utilize resources in electronic text documents.
  - organize documents to facilitate information filtering.
  - discover and find information and patterns in text.
  - create knowledge database from text.
- Focus on the use of machine learning and prediction methods.

Structured Data-mining

- Input: spread-sheet format
  - each column is a well-defined attribute
  - each row is a data point
- Prediction output (mining):
  - patterns and relationships among attributes and data points
  - predict missing values in certain columns

Structured Data Example

Gender	Systolic BP	Weight	Disease Code
M	175	65	3
F	141	72	1
...	...	...	...
F	160	59	2

Figure 1: A Spreadsheet Example of Medical Data

- Prediction problem: disease code for a patient given other columns

Unstructured Text-mining

- Input: free format text without well-defined attributes
- Mining: extract information, find patterns, and organize contents
- Predictive method: extract (predict) unknown information from available text using machine learning
  - the same spreadsheet model for structured data
  - encode available text into numerical vector (input)
  - encode desired information into unknown labels to predict (output).

Some Problems in Predictive Text-mining

- Text categorization: assign text documents to a set of topics
  - prediction problem: given text, predict label (belongs to a topic or not)
- Information extraction: extract and label text chunks from document
  - prediction problem: predict chunk boundaries and the associated labels
- Summarization: extract segments from document to represent its content
  - prediction problem: the representativeness of text segments.
- Search: extract documents relevant to a query.
  - prediction problem: predict the relevance of documents for a query.

The Machine Learning Approach

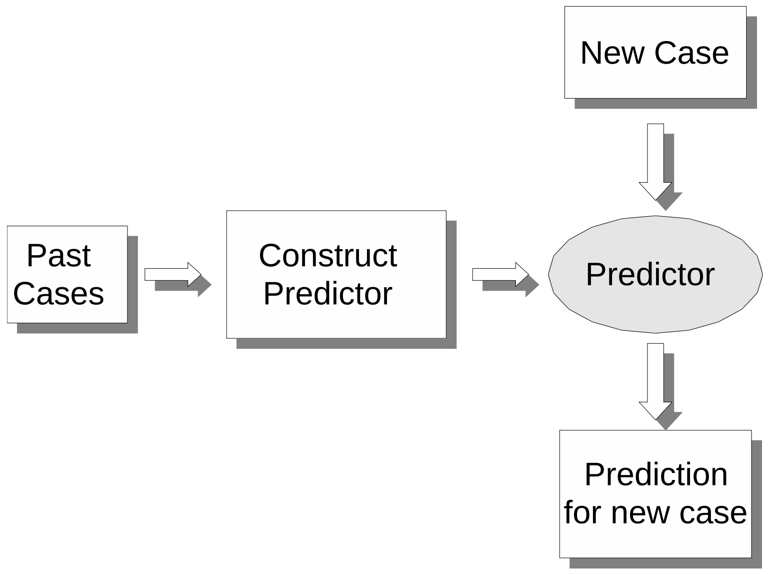


Figure 2: Predicting the Future Based on the Past

- past cases: training data (known input and known output)
- new cases: test data (known input and unknown output)

Supervised learning

- Prediction problem:
  - x: input vector, y: unknown output.
  - want to find function $q(x)$: $y \approx q(x)$
- Quality of prediction: measured by a loss function $L(q(x), y)$
- Predictor $q(x)$ learned from training examples $(x_1, y_1), \ldots, (x_n, y_n)$.
- Some learning problems:
  - classification (discrete output, 0-1 loss)
  - regression (continuous output, squared loss)
  - conditional density est. (probability output, $-\log$ loss)

Outline of the Tutorial

- Basic document processing and representation
- Text categorization
- Information extraction
- Some Final Remarks

Electronic Text

- Standard exchange format: XML (eXtensible Markup Language)
- Embed tree structured annotation into text
  - may encoding sections such as title, body, etc
  - may encoding label information as annotations
- May contain multiple documents in one file

An Example of XML Document

```
<DOC>
<TEXT>
<TITLE>
Solving Regression Problems with Rule-based Classifiers
</TITLE>
<AUTHORS>
<AUTHOR>
Nitin Indurkhya
</AUTHOR>
<AUTHOR>
Sholom M. Weiss
</AUTHOR>
</AUTHORS>
<ABSTRACT>
We describe a lightweight learning method that induces an ensemble
of decision-rule solutions for regression problems.  Instead of
direct prediction of a continuous output variable, the method
discretizes the variable by k-means clustering and solves the
resultant classification problem...
</ABSTRACT>
</TEXT>
</DOC>
```

Text Processing for Predictive Modeling

- How to encode free text into feature vectors so that machine learning methods can utilize

- Basic model of free text:
  - documents are sequences of basic-unit called *tokens*
  - in English: tokens are words
  - in Chinese: tokens are characters (also phrase or character segments).

Tokenization

- Break text into tokens
  - input: Susan’s solution to this problem.
  - output (tokens):
    - Susan
    - ’s
    - solution
    - to
    - this
    - problem
    - .
  - map each token to an integer (index into token dictionary)

Issues in Tokenization

- How to handle style, ambiguity, special symbols, punctuations, and numbers
  - ” versus ’ ’ &amp; versus &
  - 5224-2582: how many tokens (telephone number or subtraction)?
  - U.S. Department: does . indicate end-of-sentence?
- Normalization:
  - U.S. or US or U.S or u.s.
  - May 4 or 05/04 or 5/4 (US format) or 4/5 (European format)
  - Case information: all-cap text (usually in title) versus mixed case text
  - Acronym: LDA (Linear Discriminative Analysis? Latent Dirichlet Allocation?)
- Language issues:
  - Chinese: no space between multi-character words

Simple English Tokenization Procedure

- Separate tokens by delimiters including punctuations and white spaces
- Use a consistent set of rules to handle ambiguities
  - always separate -
  - resolve ending . (e.g. always separate from the preceding word)
  - possible additional processing:
    - case information, end-of-sentence detection, normalization, etc
- Error allowed but should be consistent between training and testing.

Lemmatization and Stemming

- Convert Linguistic variants of a token into a standard base form.
  - goal: reduce number of tokens (tokens with same meaning will match)
  - example: typed → type, am are is → be
- Stemming: reduce tokens into their roots
  - use dictionary: is → be
  - Porter stemmer:
    - a set of rules such as:
      - IES → I
      - S →
    - software by author: http://www.tartarus.org/∼martin/PorterStemmer/
- Effectiveness: improve chance of match but reduce quality of match
  - related processing: map synonyms to a base-form

Document Level Feature Representation

- Word features: token as unit possibly with different type such as stemming.
- Multi-word features:
  - consecutive tokens as a unit
  - two tokens co-occur within a window of certain size
- Partial position information:
  - section
  - position relative to paragraph or document
- Nontext information and features (important for many real applications)
  - e.g. webpages: font size, html tag, domain, url string, anchor text, link information, etc

Vector Space Document Model

- Procedure
  - create Dictionary of size m consisted of all tokens (without punctuations)
  - map each tokenized document into an m-dimensional vector
    - the i-th component is the frequency of token i in the document
    - feature vector is very sparse and high dimensional
- Bag-of-words: representation of text without ordering information
  - can preserve section or partial position information.
  - can combine multiple dictionaries with appropriate weighting (token before/after stemming, using synonyms).
  - can use multi-word features (n-gram, collocation, etc)
- Usually not used for nontext features

Removal of Stopwords

- Remove functional or frequent words without predictive capability
  - examples: a the it they
- Purpose:
  - reduce dictionary size and quality
  - improve computational efficiency
- Can improve prediction accuracy but have potential side-effect
  - removing meaningful words: IT $\rightarrow$ it (information technology, information theory)
- Related to: feature selection

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:

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

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

  - $\text{tf}(j,d)$: term frequency of token j in document d
    - $*$ modification: truncated term frequency such as
      - · binary (presence or absence of words)
      - · 0-1-2 representation (zero, one or multiple occurrence).
  - $\text{df}(j)$: frequency of documents containing term j

Term Weighting in Document Retrieval

- Query q, document d, with term counts tf(j, q) and tf(j, d).
- $r(q, s)$: matching score — how relevant d is for query q
- TFIDF matching score: $r(q, s) = \sum_j$ tf$(j, q) *$ tf(j,d) $*$ idf(j).
- TFIDF with truncated TF (BM25 [Robertson et al]):

$$r(q, s) = \sum_j \frac{\text{tf}(j, q) \cdot (k_3 + 1)}{\text{tf}(j, q) + k_3} * \frac{\text{tf}(j, d) k_1}{\text{tf}(j, d) + k_1 \left((1 - b) + b \cdot l / \bar{l}\right)} * \text{idf}(j)$$

  - k_1, k_3, and b: empirical constants (e.g., $k_1 = 1.2, k_3 = 7, b = 0.75$)
  - $l / \bar{l}$: document length over average document length.

Token Statistics: Zipf's Law

- Consider a text corpus,
  - m: the size of the word dictionary
  - j: the j-th ranked word based on frequency
  - $f(j)$: the frequency of token j
- Zipf's law: $f(j) \propto j^{-\alpha}$ where $\alpha \approx 1$.
  - $f(j) \approx j^{-1} / \ln(m)$.
- Named after Harvard linguistic professor George Kingsley Zipf
- Potentially useful as prior information for Bayesian modeling and for understanding behaviors of learning algorithms (effective token dimension is small).

Summary of Document Level Feature Generation

- Tokenize document, with basic processing
  - stemming, synonym resolution, stop-word removal, etc
- Generate dictionaries
- Generate bag-of-word features for each dictionary and combine
  - use appropriate term weighting (uniform or idf)
  - use (truncated) term frequency count
  - use sparse vector representation.
- Post processing: normalize according to document length
  - e.g. normalize so that the vector lies in the sphere.
- Incorporating nontext features (typically dense vector)

Example Feature Vector for Email Spam Detection

text:title			text:body			nontext	prediction target
...	cheap	...	enlargement	...	ink	from known spam host	spam
...	yes	...	yes	...	yes	yes	true
...	no	...	yes	...	no	yes	true
...	no	...	no	...	no	no	false
...	...	...	...	...	...	...	...

- Spreadsheet representation (encode unstructured text into structured data)
  - 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: bag-of-word representation, sparse, linear classifier works well
  - nontext feature: heterogeneous, dense, often contains non-linear interaction

Text Categorization

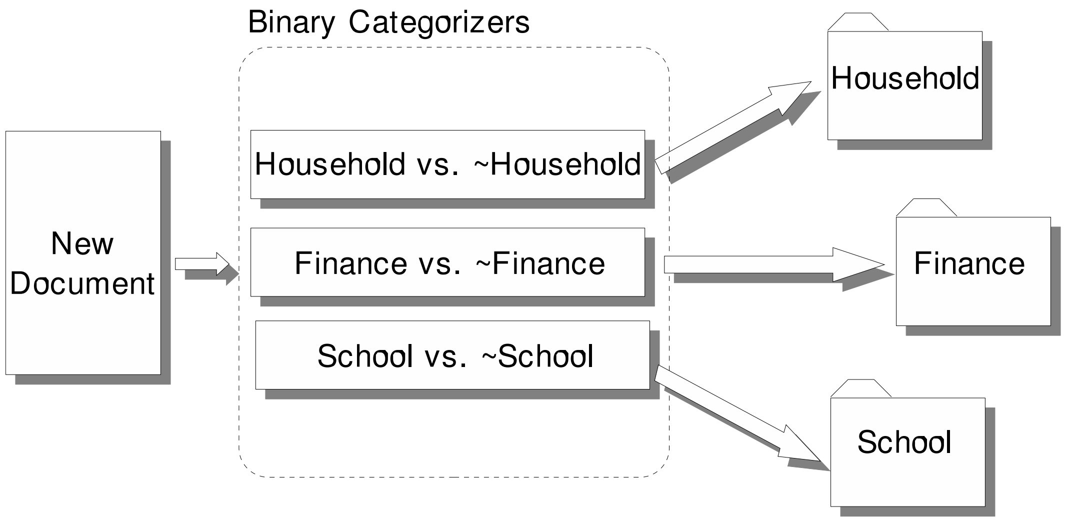


Figure 3: The problem: assign a set of topics to a text document

Text Categorization Applications

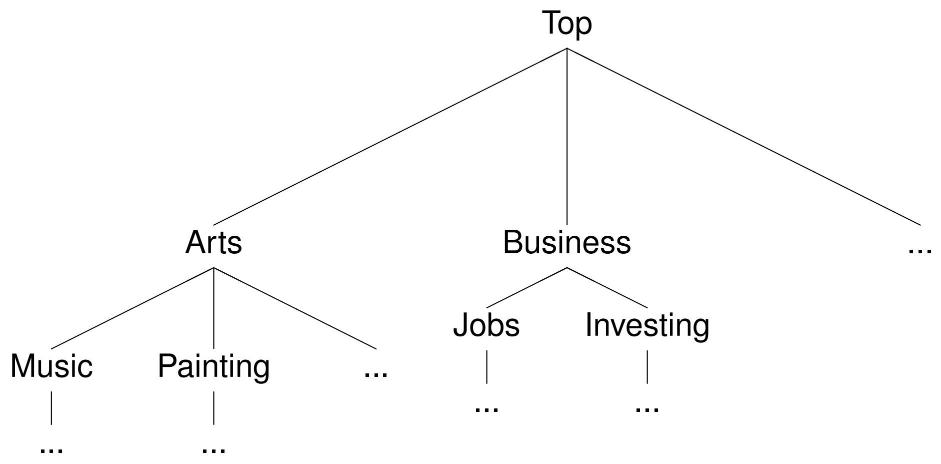
- Electronic spam detection.
- Place text document into a taxonomy.
  - Text document routing (e.g. placing text into folders).
  - Guided navigation (e.g. Yahoo web directory).
  - Concise representation of document to help search and document content matching.
- Filling document-level missing information (e.g. language id, author id, etc).

Electronic Spam Detection

- email spam: unwanted email sent to multiple people
  - typically containing commercial advertising for dubious products, get-rich-quick schemes, etc.
- webpage spam: low-quality pages with the intention to be placed high, or help other (low-quality) pages to be placed high in search engine results.
- blog spam: random blog pages with links to promote other pages or paid ads.
- phone text messaging spam, instant messaging spam, usenet newsgroup spam, etc...

Taxonomy Classification

- Tree structured organization of topics



- Can contain a large number of categories (e.g. hundreds of thousands)
  - how to measure error (what if predict the path partially correct)
  - can we achieve accurate results
  - computational efficiency

Basic Text Categorization Framework

- For each category: determine whether a document belongs to it
  - input: vector representation of document
  - output: binary prediction: belong or not
  - confidence of prediction desirable
- General approach: return a real-valued score
  - text belongs to the category if score larger than a threshold
  - calibrate score to probability
    - map score into $[0, 1]$: the probability of document belong to the category
    - prediction more confident if probability close to zero or one
- Multiple categories: return top scored categories

Probability Calibration

- Given document d and category t: classifier returns a score $s(d,t) \in R$
  - find calibration function f and map $s(d,t)$ to $[0,1]$
  - $P(d$ belongs to topic $t|s(d,t)) = f(s(d,t))$
- Calibration through binning:
  - let $a_0 = -\infty < a_1 < \cdots < a_K = \infty$
  - divide R into K bins: $B_k = [a_{k-1}, a_k)$.
  - Define m_k: number of documents d such that $s(d,t) \in B_k$
  - Define n_k: number of positive documents d such that $s(d,t) \in B_k$
  - f: map $s \in B_k$ to $(n_k + 0.5)/(m_k + 1.0)$.
- More general: use one dimensional density estimation methods

Comments on Probability Calibration

- Goal of calibration: to make score more interpretable.
- Calibration function is often (near) monotonic.
- Calibration generally does not improve classification accuracy.
- Even useful for probability classifiers such as logistic regression
  - discriminative training of classifiers tends to overfit probability estimation in order to achieve optimal classification accuracy
  - tends to produce more confident (towards 0, 1) probability estimate than they should

Common Classification Methods

- Nearest Neighbor and Centroid
- Rule Induction
- Naive Bayes
- Discriminative Linear Classification

Document Similarity in Vector Space Model

- Documents d_1 and d_2.

- Un-normalized TFIDF weighted score:
  - similarity score: $\sum_j \mathrm{idf}(j) * \mathrm{tf}(j, d_1) * \mathrm{tf}(j, d_2)$
  - used in document retrieval.

- Normalized cosine measure:
  - map d_1 and d_2 into feature vectors u_1 and u_2 (with or without term weighting)
  - normalize $v_1 = u_1/\|u_1\|_2$ and $v_2 = u_2/\|u_2\|_2$
  - similarity is $v_1^T v_2 \in [0, 1]$
    - $*$ similar documents have score ≈ 1, different documents have score ≈ 0

- Other methods: cosine in low-dimensional space (SVD/LSI), similarity along document manifold, etc

Nearest Neighbor Method

- Given a document similarity measure and training data
  1. compute the similarity of new-Doc to all documents in training data
  2. select the k documents that are most similar to new-Doc.
  3. select label that occurs most frequently in the k selected documents.
- Advantage:
  - require few positive examples and no negative examples
  - easy to understand and to update when new data come in
- Disadvantage:
  - can be memory and computationally inefficient with many training data
  - relies on similarity measure (what about nontext features)

Centroid Method

- To reduce computational inefficiency of nearest neighbor method
- Consider similarity measure: $s(d_1, d_2) = \sum_j w_j * \mathrm{tf}(j, d_1) * \mathrm{tf}(j, d_2)$.
  - let T be the document collection belonging to category t
  - let centroid term frequency $\mathrm{tf}(j, t) = \sum_{d \in T} \mathrm{tf}(j, d)/|T|$.
  - score of document d and category t: $s(d, t) = \sum_j w_j * \mathrm{tf}(j, d) * \mathrm{tf}(j, t)$.
- Advantages: Simple, Understandable, and Efficient
  - useful for taxonomy with large number of coherent categories with small number of examples per category
  - tend to be less accurate than other methods when category isn't coherent and contain many training examples
- Improvements: mixture of Gaussian per category, similarity function learning

Rule Based Classification

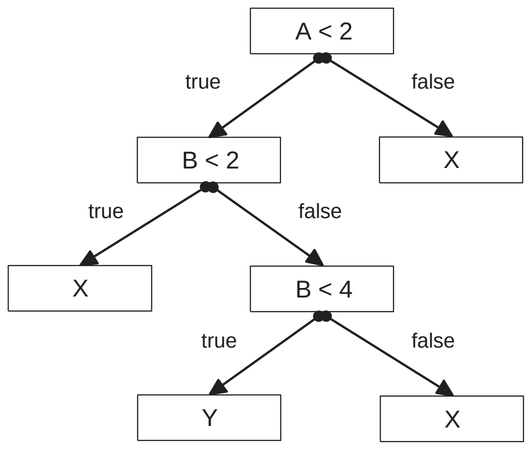
shr $\longrightarrow$ earn
div $\longrightarrow$ earn
dividend $\longrightarrow$ earn
payout $\longrightarrow$ earn
qtr $\longrightarrow$ earn
earnings & sees $\longrightarrow$ earn
quarter & cts $\longrightarrow$ earn
split $\longrightarrow$ earn
profit $\longrightarrow$ earn
OTHERWISE $\longrightarrow$ $\sim$earn

Figure 4: Example Decision Rules for Reuters “earn” category

- Easy to understand and possibly modify by a human
- Can incorporate nontext features more easily than other methods

Rule Learning through Decision Trees

Figure 5: Example Decision Tree



Equivalent rules (read along tree path):

$A < 2$ & $B < 2 \longrightarrow$ category-X $\qquad$ $A < 2$ & $B \geq 2$ & $B < 4 \longrightarrow$ category-Y $\qquad$ ...

Decision Trees

- Decision Tree partition the data into segments along paths to leaf-nodes
- Each node consists a test to partition data: is an attribute value $<$ a threshold
- Constant prediction at each node:

$$\text{probability score} = \frac{\text{number of in-class documents reaching the node}}{\text{number of documents reaching the node}}$$

- Two-stage learning:
  - Tree growing: recursively search (attribute,threshold) pair to reduce error
  - Tree pruning: remove deep tree nodes to avoid overfitting

Tree Growing

- Given smooth (convex) loss function $L(f, y)$ (such as $(f - y)^2$) and n training data (X_i, Y_i) ($i = 1, \ldots, n$)
- Recursively do the following until reaching a certain depth or no improvement can be obtained by splitting
  - at each leaf-node, let S be the training data reaching it
  - the optimal loss at the node is: $\min_f \sum_{i \in S} L(f, Y_i)$.
  - for each partition (attribute,threshold) pair (j, θ),
    - $*$ partition S into $S_1(j, \theta)$ and $S_2(j, \theta)$ using the test
    - $*$ the optimal loss with this partition $\min_{f_1, f_2} [\sum_{i \in S_1} L(f_1, Y_i) + \sum_{i \in S_2} L(f_2, Y_i)]$
  - for each leaf node: grow the tree by using (j, θ) that reduces the loss most

Tree Pruning

- Fully grown tree tends to overfit the data
  - data are partitioned into very small segments
  - insufficient data at each leaf node to reliably estimate probability
- Pruning: removing deep tree nodes so that leaf nodes in the resulting tree contain sufficient data for reliable probability estimate
  - many different methods
- Prune to a fixed tree size:
  - given a loss function $L'(f, y)$ (may differ from training loss: e.g. non-smooth classification loss)
  - recursively removing leaf-nodes with least reduction of loss L' until number of leaf-nodes reach a fixed size

Improving Decision Tree Performance

- Advantage of decision tree: interpretable, handle nontext features easily
- Disadvantage: usually not the most accurate classifier by itself
- Improve accuracy through tree ensemble:
  - bagging and random forest (Breiman):
    - majority voting of random trees, each learned with a random sample of features and random subset of training data
  - boosting (small stepsize version):
    - pick a small step size $\eta < 1$ and initial predictor $f = 0$
    - iterate for a fixed number of iterations:
      - build tree predictor f_T approximately $\min \sum_i L(f(X_i) + \eta f_T(X_i), Y_i)$
      - update predictor: $f \leftarrow f + \eta f_T$.
      - gradient version (Friedman): $L(f(X_i) + \eta f_T(X_i), Y_i) \approx L(f(X_i)) + \eta L'(f(X_i), Y_i) f_T(X_i) + 0.5\eta^2 L''(f(X_i)) f_T(X_i)^2$.

Naive Bayes Method (multinomial model)

- Each category t determines a probability distribution over the dictionary: probability for token j is $P_t(j) = \mu_t(j)$
- Given class t and a document d in the class: first generate document length $\ell(d)$, then each word in d is randomly drawn from the dictionary according to P_t
- Probability of (fixed length) document d with word counts $c(j, d)$:

$$\boldsymbol{P(d|t) = P(\ell(d)|t) \prod_j \mu_t(j)^{c(j,d)}}$$

- Model nontext features with mixture of Gaussian.

Naive Bayes Method (parameter inference)

- Requires only positive data.
- Dirichlet prior: $P(\mu) \propto \prod_j \mu_j^\lambda$
- MAP estimate (λ: small smoothing parameter):

$$\boldsymbol{\mu_t = \max_{\mu} \sum_{d=1}^{N} \sum_j (c(j,d) + \lambda/N) \ln \mu_j, \quad \mu_j \geq 0, \sum_j \mu_j = 1.}$$

- Solution: $\boldsymbol{\mu_t(j) = (\lambda + \sum_d c(j,d)) / \sum_{j'} (\lambda + \sum_d c(j',d))}$
- Score of document d for category t (topic indep length):

$$s(d,t) = \ln P(t) + \sum_j c(j,d) \ln \mu_t(j) \quad P(t) = \frac{\text{number of docs with category } t}{\text{number of docs}}$$

Linear Classification

- Given input feature vector x, and label $y \in \{\pm 1\}$ for category t
- Linear classifier:
  - weight vector w_t for each category t
  - scoring function for each category: $w_t^T x - \theta_t$ (θ_t: a threshold)
  - decision rule: in class if score > 0, and out-of-class otherwise
- Training methods: naive Bayes, online methods, regularized empirical risk minimization, etc
- Good model for text features, but less so for nontext features.

Online Training (Perceptron)

- Given (X_i, Y_i): $Y_i \in \{\pm 1\}$
- Initialize weight $w = 0$
- Iterate over i
  - if $\boldsymbol{w^T X_i Y_i \leq 0}$, then update $\boldsymbol{w \leftarrow w + X_i Y_i}$
- Advantage: simple and efficient, performs quite well
  - similar to stochastic gradient descent, which can solve formulations such as SVM and logistic regression

Regularization: Fitting versus Generalization

- A good predictor $f(x)$: generalizes well on unseen data.
  - fits well on the training data.
    - requires a more expressive model.
  - performance on the unseen data matches that on the training data.
    - requires a less expressive (more stable) model.

- Over-fitting of data:
  - $f(x) = Y_i$ if $x = X_i$ at a training point
  - $f(x) = 0$ otherwise.

Linear Regularization

- Regularization: restrict the model expressiveness.
- Linear predictor: $\boldsymbol{f(w, x) = w^T x}$.
- Restrict the linear family size: $\boldsymbol{g(w) \leq R}$.
  - example: $g(w) = w^2$.
  - R: tuning parameter (estimated through cross-validation).
- Benefit of regularization:
  - statistical: robust to large number of features.
  - numerical: stabilize solution.

Linear Regularized Empirical Risk Minimization

- Goal: minimize average loss $\boldsymbol{L(w^T x, y)}$ on unseen data.
- A practical method: minimize observed loss:

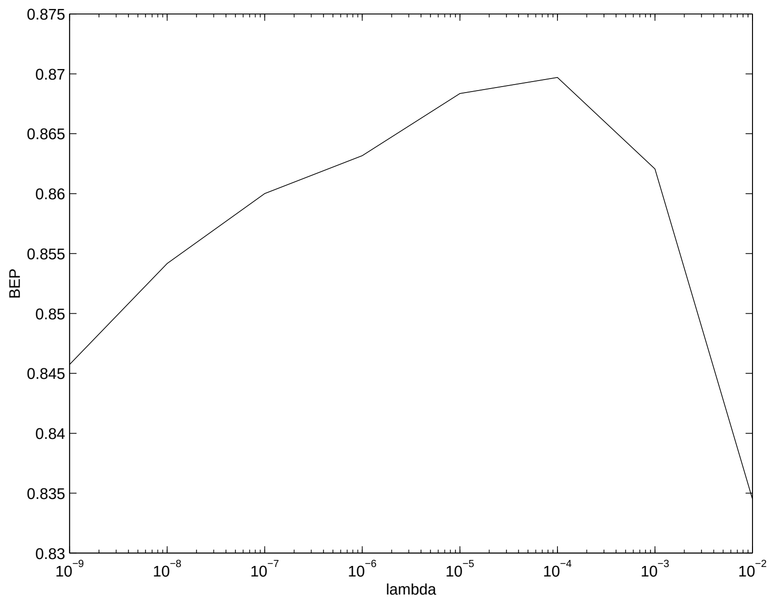
$$\hat{w} = \arg\min_w \frac{1}{n} \sum_i L(w^T X_i, Y_i),$$

$$\text{such that} \quad g(w) \leq R.$$

- Equivalent formulation ($\lambda \geq 0$):

$$\boldsymbol{\hat{w} = \arg\min_w \left[\frac{1}{n} \sum_i L(w^T X_i, Y_i) + \lambda g(w) \right]}$$

Predictive Accuracy versus λ



- small λ: more expressive model
- large λ: less expressive (more stable) model

Dimension Independent Generalization Ability

- Least squares with square regularization:
  - $\hat{w} = \arg\min_w \left[\frac{1}{n}\sum_i (w^T X_i - Y_i)^2 + \lambda w^2\right]$.
  - predictive power of $\hat{w}$:

$$\left(1 + \frac{1}{\lambda n}\max_x x^2\right)^2 \underbrace{\min_w E_{x,y}\left((w^T x - y)^2 + \lambda w^2\right)}_{\text{best possible regularized loss}}$$

- Robust to large feature set:
  - square reg $g(w) = \frac{1}{2}w^2$
  - generalization depends on x^2, not on dimension of x.

Some regularization conditions

- Square regularization: $g(w) = w^2/2$.
- Sparse regularization: $g(w) = w^2/2 + \epsilon|w|$.
  - ϵ control sparsity.
- Entropy regularization with positive w:
  $g(w, b) = \sum_i w_i \ln(w_i/\mu_i)$.
  - in theory more robust to irrelevant features.

Loss function

- Regularized Linear classification:

$$\hat{w} = \arg\min_{w} \left[\frac{1}{n} \sum_{i} L(w^T X_i, Y_i) + \lambda g(w) \right]$$

- Natural optimization criterion: empirical classification error minimization

$$L(f, y) = I(fy \leq 0)$$

  - Computationally: hard to minimize.
  - Statistically: regularization has no effect.

Convex Risk Minimization

- Minimize upper bound of classification error.
  - convex loss: easy to optimize.
  - $L(f, y)$ is convex of f.

- Different loss functions.
  - Logistic Regression: $\boldsymbol{L(f, y) = \ln(1 + \exp(-fy))}$.
  - Adaboost: $\boldsymbol{L(f, y) = \exp(-fy)}$
  - SVM: $\boldsymbol{L(f, y) = \max(0, 1 - fy)}$.
  - Modified Huber loss:

$$\boldsymbol{L(f, y) = \begin{cases} -4fy & fy < -1 \\ (fy - 1)^2 & fy \in [-1, 1] \\ 0 & fy > 1. \end{cases}}$$

Loss Function Graphs

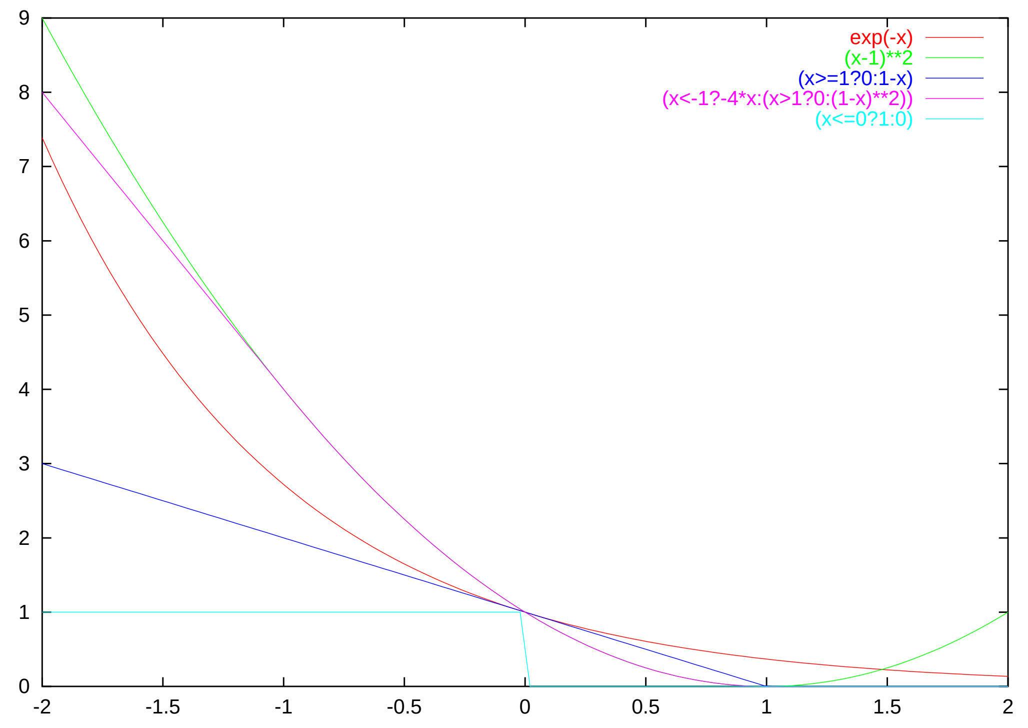


Figure 6: Plots of Least Squares loss, Modified Huber loss, Exponential loss, SVM loss, against Classification Error.

Statistical Model For Convex Risk Minimization

- Regularized convex risk minimization:

$$\hat{w} = \arg\min_{w} \left[\frac{1}{n} \sum_{i} L(w^T x_i, y_i) + \lambda g(w) \right]$$

  - leads to conditional probability estimation.
    - $*$ logistic regression: $p(y|x) = 1/(1 + \exp(-w^T x))$.
      - $\cdot$ cannot model $p(y|x) = 0, 1$.
    - $*$ modified Huber: $p(y|x) = \max(0, \min(1, 2w^T x - 1))$.
    - $*$ SVM (hinge-loss): lose probability information asymptotically.
  - more stable than classification error minimization.

Comments on Regularized Linear Classification

- Computation: convex optimization problem
  - convex duality plays a role
  - optimization can be done for primal or dual formulations
  - stochastic gradient descent with early stopping often works well:
    - iterate over i, and update weight w as
    - $\boldsymbol{\nabla g(w) \leftarrow \nabla g(w) - \eta(L'(w^T X_i, Y_i) X_i + \lambda \nabla g(w))}$

- Advantages:
  - good theoretical understanding, state of the art performance
  - loss function: determine the probability model.
  - regularization: avoid over-fitting, achieve sparse representation.

- Disadvantage: less efficient than naive Bayes, does not automatically generate nonlinear features.

Performance Evaluation for Text Categorization

- Measure classification performance for each category
  - small number of positive examples, most data are negative
  - precision, recall, and F-measure

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

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

$$F\text{-measure} = \frac{2}{1/\text{precision} + 1/\text{recall}}$$

  - precision/recall trade-off: adjust decision threshold
    - $*$ higher threshold usually implies higher precision and lower recall

Comparisons of Some Linear Classifiers

	Naive Bayes	Least Squares	Logistic Reg	SVM
precision	77.0	87.1	88.0	89.2
recall	76.9	84.9	84.9	84.0
F_1	77.0	86.0	86.4	86.5

Table 1: Binary classification performance on Reuters-21578

- Data: http://www.daviddlewis.com/resources/testcollections/reuters21578
- Least squares: nearly as good as other loss functions but more sensitive to regularization parameter tuning

Dealing with Large Taxonomy

- Flat categorization (one-versus-all): train a scoring function for each category separately.
- Hierarchical classification: train a classifier at each node to determine which path to follow.
- Error correcting output code (ECOC): solve a smaller number of classification problems, each corresponding to a group of original categories.
- Design loss function to handle multi-class problems with tree-structured output directly.
  - related to structured output prediction: promising but not yet convincing results, and more difficult to implement

Flat Classification for Large Taxonomy

- Train a scoring function for each category
- Return top scored categories as results
- Training efficiency
  - Naive Bayes or Centroid: linear in sample size
  - Discriminative classification: linear in category size * sample size
    - solution: subsample of negative data.
- Testing efficiency for linear classifier: require highly sparse weight vectors
  - use ideas from search: create reverse index for each weight vector, select top keywords from document and only consider categories with non-zero weights matching the keywords

Hierarchical Classification

- Train a scoring function for each node in the taxonomy
  - determine whether to follow left or right branch.
- Testing: greedily follow the prediction at each node until the leaf
  - refined approach: multiply probability estimate along path.
  - keep track of a few top scoring paths.
- Training efficiency: efficient and generally linear in sample size.
- Testing efficiency: require path pruning.

Error Correcting Output Code

- Randomly (or use a design matrix) group categories into super categories with probability 0.5.

- Train a scoring function for each super category.

- Testing:
  - decoding: score for each category is the sum of probability estimate for each super category it belongs to.
  - return top scored categories

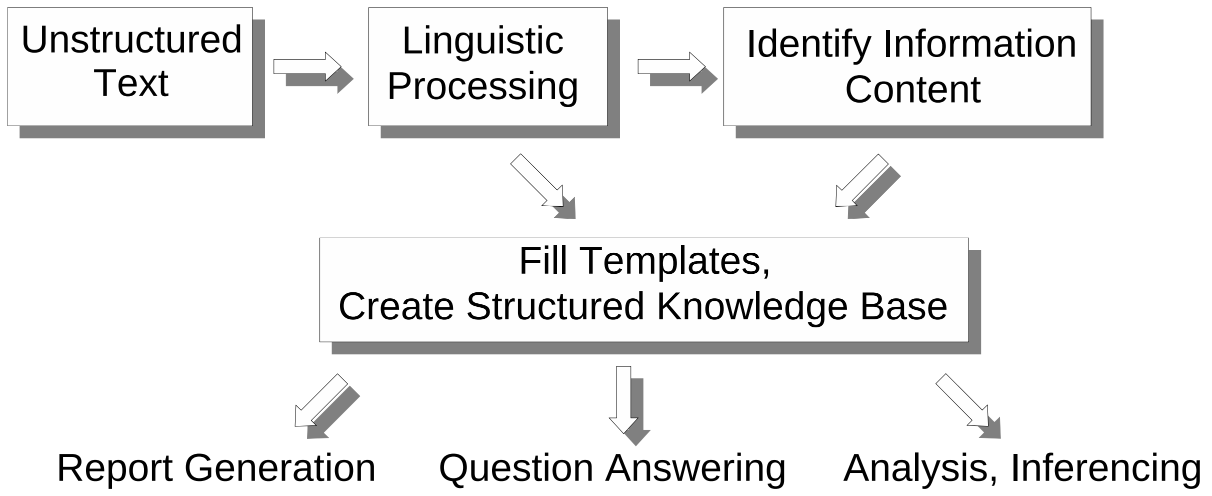
- Theoretical support: under appropriate (and reasonable) assumptions, may require as few as logarithmic number of super categories without losing much accuracy.

- Efficiency: depending on the number of super categories and decoding step

Information Extraction

- Extract information from unstructured text.
  - entity extraction.
  - co-reference resolution.
  - relationship extraction, semantic annotation.
- Represent in a structured knowledge base.
- Example: named entity recognition
  - find entities such as people, organization, location.

Information Extraction System Diagram



Named entity example

ORGANIZATION PERSON LOCATION COUNTRY

THOUSAND OAKS , Calif. (Reuters) - President Bush on Friday called the worst blackout in North American history a “ wake-up call ” and said he would push to upgrade the nation’s electricity grid to head off future breakdowns.

Bush was briefed by Treasury Secretary John Snow about how financial markets were holding up and the White House said Health and Human Services Secretary Tommy Thompson was contacting hospitals in affected areas to make sure they had all the supplies they needed.

Bush said investigators needed to find out why the outages cascaded so quickly through much of the northeastern United States and the Canadian province of Ontario, knocking New York City, Detroit, Cleveland, Ottawa, Toronto, and a host of smaller cities back into the pre-electric age.

Modeling Entity Extraction as Prediction Problem

- Named entity recognition as a sequential prediction problem.
  - Token-based tagging: assign a tag for each token.
  - Tag representation:
    - B-X (beginning of entity X)
    - I-X (inside entity X)
    - O (not an entity)

- A classifier to predict tag for each token, based on the associated feature vector.

- Linear classification works very well.

Example

```
Bush    was briefed by Treasury Secretary  John   Snow
B-PER    O      O     O  B-ORG      O       B-PER  I-PER

about how financial markets were  holding up and the
  O    O     O          O      O      O      O   O   O

White House  said Health   and    Human  Services Secretary
B-ORG I-ORG   O    B-ORG  I-ORG   I-ORG   I-ORG       O

Tommy Thompson was contacting hospitals ...
B-PER I-ORG     O     O          O
```

Local Modeling Approach

- Base prediction problem: predict tag for each token position.
- Train classifier: at each token position, assign confidence score for each possible tag value.
- Testing: independent decoding for each token, or optimize tag-sequence using viterbi.
- Can use previously decoded tags as features in prediction.
- Advantage: efficient training, and competitive accuracy.

Global Modeling Approach

- Base prediction problem: predict tag-sequence for the entire sentence as a single label (different tag-sequence considered different labels)
- Probability modeling: conditional random field (CRF) [Lafferty, et al])
  - maximum entropy model to estimate sequence conditional probability
  - claimed advantage: solve the label bias problem
- Structured output approach ([Taskar et al], [Tsochantaridis et al] ...)
  - can adapt formulations as upper bounds for arbitrary cost functions
  - can be used for many other problems
- Perceptron learning ([Collins])
  - simple implementation
  - can adapt to cost functions by step size adjustment (not in the literature)

Features (local learning)

- Local features of a certain window size (2):
  - Tokens
  - Previously assigned tags.
  - Word-type: capitalization, is a number, prefix, suffix, etc.
  - Other token features: part-of-speech...
  - Dictionary lookup features: words inside dictionary:
    - “New York” in location dictionary.
  - Linguistic patterns: regular expression, etc.
  - Possible occurrence of local features.
- Long range features: parsing based and multiple-occurrence based.

An Experiment (CONLL03)

- English text
  - Taken from the new Reuters corpus (1996/8).
  - 200,000 training tokens; 50,000 testing tokens.
- Four entity types:
  - LOC, ORG, PER, MISC
- baseline (phrases in test data matching the training data):
  - precision: 78.33; recall: 65.23; FB1: 71.18

Impacts of Various Features

- All-upper-case text with token features only:
  - precision: 91.94; recall: 74.25; FB1: 82.15
- Mixed-case text with token features only:
  - precision: 93.70; recall: 74.89; FB1: 83.25
- With token-level features:
  - precision: 90.11; recall: 88.67; FB1: 89.39
- With additional local features: token features + word-type features + dictionary features + pos/chunking:
  - precision: 92.70; recall: 91.32; FB1: 92.01.

Some Other Applications of Predictive Methods

- Summarization (for documents and search results)
  - predict importance of sentences or text segments containing query words
- Question Answering
  - predict whether a segment of text answers the question
  - may extract the exact answer
- Matching and Co-reference resolution (webpages, product records, queries, different mentions of entities, etc)
  - predict the probability of two inputs to match
- Others natural language applications: Semantic Annotation, Machine Translation, Parsing, etc

Some Issues in Applying Prediction Methods

- Testing data differs from training data
  - data change through time
  - training data available in one domain (news article) but test data in another domain (e.g. web-pages)
  - sampling of training data is biased

- How to learn with small number of data and adapt under domain change
  - construct prior from unlabeled data
  - learn or construct task-invariant representation from related tasks

- Efficiency and performance optimization under resource constraints
  - complicated task such question answering, best systems often too slow
  - complicated feature generation often costly (sparsity)
  - kernel methods often not practical

Final Remarks on using Prediction Methods

- Can solve many problems
  - modeling
    - ∗ formulate as predicting unknown information from known information
    - ∗ incorporate prior information and domain-specific knowledge
  - automatic performance evaluation important
    - ∗ e.g. how to evaluate summarization and machine translation
    - ∗ often need to track multiple performance metrics
    - ∗ system components may not be designed to directly optimize evaluation
- Easier to maintain than hand-tuned rule systems
  - at the feature level rather than at the rule-set level
  - system improvement can benefit from training data, feature engineering, and statistical modeling/learning techniques