Text Mining
Statistical Modeling of Textual Data
Lecture 2

Mattias Villani

Division of Statistics
Dept. of Computer and Information Science
Linköping University

Overview

- Text classification
- Regularization
- R's tm package (demo) [TMPackageDemo.R]

Supervised classification

- Predict the **class label** $s \in S$ using a set of **features**.
- Feature = Explanatory variable = Predictor = Covariate
- Binary classification: $s \in \{0, 1\}$
  - Movie reviews: $S = \{\text{pos,neg}\}$
  - E-mail spam: $S = \{\text{Spam,Ham}\}$
  - Bankruptcy: $S = \{\text{Not bankrupt, Bankrupt}\}$
- Multi-class classification: $s \in \{1, 2, ..., K\}$
  - Topic categorization of web pages: $S = \{'\text{News}', '\text{Sports}', '\text{Entertainment}'\}$
  - POS-tagging: $S = \{\text{VB,JJ,NN,...,DT}\}$

Supervised classification, cont.

- Example data:
  - Larry Wall, born in British Columbia, Canada, is the original creator of the programming language Perl. Born in 1956, Larry went to ...
  - Bjarne Stroustrup is a 62-years old computer scientist ...

Person	Income	Age	Single	Payment remarks	Bankrupt
Larry	10	58	Yes	Yes	Yes
Bjarne	15	62	No	Yes	No
⋮	⋮	⋮	⋮	⋮	⋮
Guido	27	56	No	No	No

- Classification: construct prediction machine

$$\text{Features} \rightarrow \text{Class label}$$

- More generally:

$$\text{Features} \rightarrow \Pr(\text{Class label}|\text{Features})$$

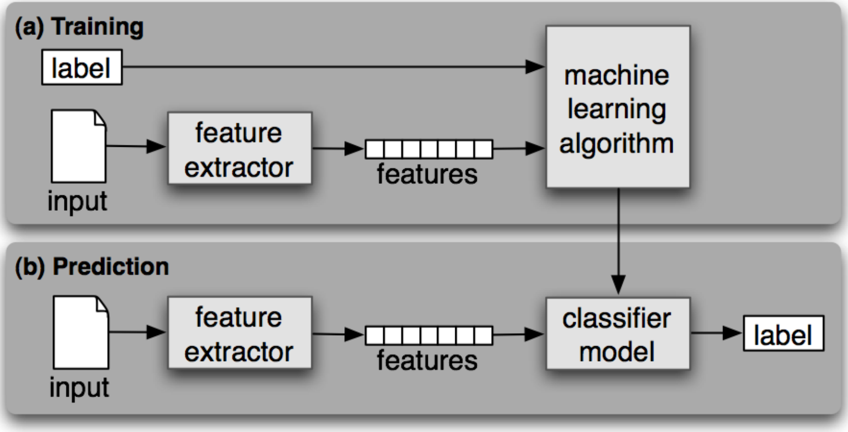
Features from text documents

- Any quantity computed from a document can used as a **feature**:
  - Presence/absence of individual words
  - Number of times an individual word is used
  - Presence/absence of pairs of words
  - Presence/absence of individual bigrams
  - Lexical diversity
  - Word counts
  - Number of web links from document, possibly weighted by Page Rank.
  - etc etc

Document	has('ball')	has('EU')	has('political_arena')	wordlen	Lex. Div.	Topic
Article1	Yes	No	No	4.1	5.4	Sports
Article2	No	No	No	6.5	13.4	Sports
⋮	⋮	⋮		⋮	⋮	⋮
ArticleN	No	No	Yes	7.4	11.1	News

- Constructing clever **discriminating features** is the name of the game!

Supervised learning for classification



The Bayesian classifier

- Bayesian classification

$$\underset{s \in S}{\operatorname{argmax}}\, p(s|\mathbf{x})$$

where $\mathbf{x} = (x_1, ..., x_n)$ is a feature vector.

- By Bayes' theorem

$$p(s|\mathbf{x}) = \frac{p(\mathbf{x}|s)p(s)}{p(\mathbf{x})} \propto p(\mathbf{x}|s)p(s)$$

- Bayesian classification

$$\underset{s \in S}{\operatorname{argmax}}\, p(\mathbf{x}|s)p(s)$$

- $p(s)$ can be easily estimated from training data by relative frequencies.
- Even with binary features [*has(word)*] the outcome space of $p(\mathbf{x}|s)$ is huge (=data are sparse).

Naive Bayes

- **Naive Bayes (NB): features are assumed independent**

$$p(\mathbf{x}|s) = \prod_{j=1}^{n} p(x_j|s)$$

- Naive Bayes solution

$$\operatorname*{argmax}_{s \in S} \left[\prod_{j=1}^{n} p(x_j|s) \right] p(s)$$

- With binary features, $p(x_j|s)$ can be easily estimated by

$$\hat{p}(x_j|s) = \frac{C(x_j, s)}{C(s)}$$

- Example: $s =$news, $x_j =$ has('ball')

$$\hat{p}\left(\text{has(ball)}|\text{news}\right) = \frac{\text{Number of news articles containing the word 'ball'}}{\text{Number of news articles}}$$

Naive Bayes

- **Continuous features** (e.g. lexical diversity) can be handled by:
  - Replacing continous feature with several binary features ($1 \leq$lexDiv< 2, $2 \leq$lexDiv≤ 10 and lexDiv> 10)
  - Estimating $p(x_j|s)$ by a density estimator (e.g. kernel estimator)
- Finding the **most discriminatory features**. Sort from largest to smallest

$$\frac{p(x_j|s = pos)}{p(x_j|s = neg)} \text{ for } j = 1, ..., n.$$

- **Problem with NB**: features are seldom independent $\Rightarrow$ double-counting the evidence of individual features.
- Extreme example: what happens with naive Bayes if you duplicate a feature?
- **Advantages of NB**: simple and fast, yet often surprising accurate classifications.

Multinomial regression

- Logistic regression (Maximum Entropy/**MaxEnt**):

$$p(s=1|\mathbf{x}) = \frac{\exp(\mathbf{x}'\beta)}{1+\exp(\mathbf{x}'\beta)}$$

- Classification rule: Choose $s=0$ if $p(s|\mathbf{x}) < 0.5$ otherwise choose $s=1$.
- ... at least when consequences of different choices of s are the same. Loss/Utility function.
- Multinomial regression for multi-class data with K classes

$$p(s=s_j|\mathbf{x}) = \frac{\exp(\mathbf{x}'\beta_j)}{\sum_{k=1}^{K}\exp(\mathbf{x}'\beta_k)}$$

- Classification

$$\underset{s\in\{s_1,\ldots s_K\}}{\operatorname{argmax}}\; p(s|\mathbf{x})$$

- Classification with text data is like any multi-class regression problem ... but with hundred or thousand of covariates! **Wide data**.

Regularization - Variable selection

- Select a subset of the covariates.
- Old school: **Forward** and **backward selection**.
- New school: **Bayesian variable selection**.
- For eachβ_i introduce binary indicator I_i such that

$$\begin{array}{ll} I_i = 1 & \text{if covariate is in the model, that is } \beta_i \neq 0 \\ I_i = 0 & \text{if covariate is in the model, that is } \beta_i = 0 \end{array}$$

- Use Markov Chain Monte Carlo (MCMC) simulation to approximate $\Pr(I_i|Data)$ for each i.
- Example $S = \{\text{News}, \text{Sports}\}$. $\Pr(\text{News}|\text{x})$.

	has('ball')	has('EU')	has('political_arena')	wordlen	Lex. Div.
$\Pr(I_i\|Data)$	0.2	0.90	0.99	0.01	0.85

Regularization - Shrinkage

- Keep all covariates, but **shrink** their β-coefficient to zero.
- **Penalized likelihood**

$$L_{Ridge}(\beta) = LogLik(\beta) - \lambda\beta'\beta$$

  where λ is the **penalty parameter**.
- Maximize $L_{Ridge}(\beta)$ with respect to β. Trade-off of fit ($LogLik(\beta)$) against complexity penalty $\beta'\beta$.
- **Ridge regression** if regression is linear.
- The penalty can be motivated as a **Bayesian prior** $\beta_i \overset{iid}{\sim} N(0, \lambda^{-1})$.
- λ can be estimated by cross-validation or Bayesian methods.

Lasso - Shrinkage and variable selection

- Replace Ridge penalty

$$L_{Ridge}(\beta) = LogLik(\beta) - \lambda \sum_{j=1}^{n} \beta_j^2$$

by

$$L_{Lasso}(\beta) = LogLik(\beta) - \lambda \sum_{j=1}^{n} |\beta_j|$$

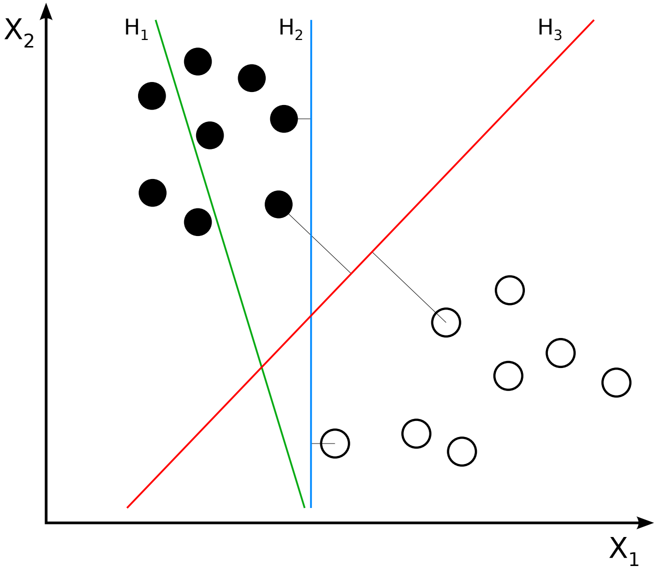
- The β that maximizes $L_{Lasso}(\beta)$ is called the **Lasso estimator**.
- Some parameters are shrunked exactly to zero $\Rightarrow$ Lasso does **both shrinkage AND variable selection**.
- Lasso penalty is equivalent to a double exponential prior

$$p(\beta_i) = \frac{\lambda}{2} \exp\left(\lambda \left|\beta_i - 0\right|\right)$$

Support vector machines

- One of the best classifiers around.
- Finds the line in covariate space that maximally separates the two classes.
- When the points are not linearly separable: add a slack-variable $\xi_i > 0$ for each observation. Allow misclassification, but make it costly.
- Non-linear separing curves can be obtained by basis expansion (think about adding x^2, x^3 and so on)
- The kernel trick makes it possible to handle many covariates.
- Drawback: not so easily extended to multi-class.
- `svm` function in R-package `e1071` [or `nltk.classify.svm`]

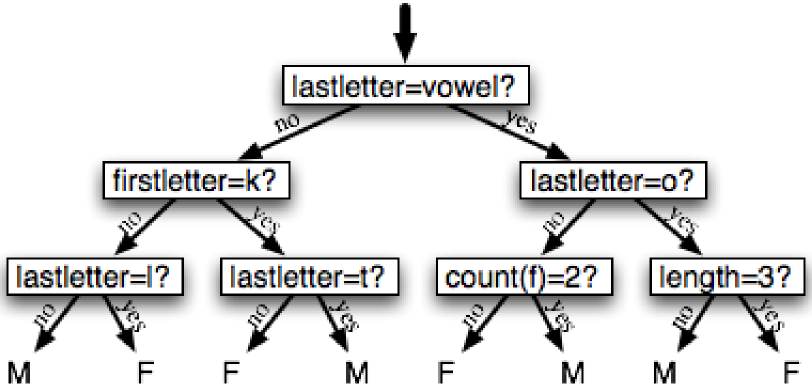
Linear SVMs



Regression trees and random forest

- Binary partioning tree.
- At each internal node decide:
  - Which covariate to split on
  - Where to split the covariate ($X_j < c$. Trivial for binary covariates)
- The optimal splitting variables and split-points are chosen to minimize the mis-classification rate (or other similar measures).
- **Random forest** (**RF**) predicts using an average of many small trees.
- Each tree in RF is grown on a random subset of variables. Makes it possible to handle **many covariates**. Parallel.
- Advantage of RF: better predictions than trees.
- RF harder to interpret, but provide variable importance scores.
- **R packages**: `tree` and `rpart` (trees), `randomForest` (RF).

Regression trees



Evaluating a classifier: Accuracy and Error

- **Confusion** matrix:

		Truth	
		Spam	Not Spam
Decision	Spam	tp	fp
	Not Spam	fn	tn

- tp = true positive, fp = false positive
- fn = false negative, tn = true negative
- **Accuracy** is the proportion of correctly classified items

$$\texttt{Accuracy} = \frac{tp + tn}{tp + tn + fn + fp}$$

- **Error** is the proportion of wrongly classified items

$$\text{Error} = \text{1-Accuracy}$$

Accuracy can be misleading

- Accuracy is problematic when tn is large. High accuracy can then be obtained by not acting at all!

		Truth	
		Spam	Not Spam
Choice	Spam	0	0
	Not Spam	100	900

Evaluating a classifier: the F-measure

- **Confusion matrix**:

		Truth	
		Spam	Good
Choice	Spam	tp	fp
	Good	fn	tn

- **Precision** = proportion of selected items that the system got right

$$\text{Precision} = \frac{\text{tp}}{\text{tp+fp}}$$

- **Recall** = proportion of spam that the system classified as spam

$$\text{Recall} = \frac{\text{tp}}{\text{tp+fn}}$$

- **F-measure** is a trade-off between Precision and Recall (harmonic mean):

$$F = \frac{1}{\alpha \frac{1}{\textit{Precision}} + (1-\alpha)\frac{1}{\textit{Recall}}}$$