Text classification.

Victor Kitov

v.v.kitov@yandex.ru

Document classification

- Major applications:
    - News filtering and organization
    - Document organization and retrieval
    - Opinion Mining (sentiment analysis)
    - E-mail classification and spam filtering
- Document classification vs labelling

Tokenization

1. Split documents into individual tokens.
    - tokens may be words or symbol sequences
    - may or may not include punctuation

Tokenization

1. Split documents into individual tokens.
    - tokens may be words or symbol sequences
    - may or may not include punctuation
2. Form the set of all distinct tokens $\{t_1, t_2, ...\}$.
    - ignore *stop-words* (exact list depends on the application)
    - ignore tokens which are too rare and too frequent
    - account only for particular parts of speech (nouns, adjectives? verbs? ...)

Tokenization

1. Split documents into individual tokens.
   - tokens may be words or symbol sequences
   - may or may not include punctuation
2. Form the set of all distinct tokens $\{t_1, t_2, ...\}$.
   - ignore *stop-words* (exact list depends on the application)
   - ignore tokens which are too rare and too frequent
   - account only for particular parts of speech (nouns, adjectives? verbs? ...)
3. May add *bigram/trigram collocations*

Tokenization

1. Split documents into individual tokens.
    - tokens may be words or symbol sequences
    - may or may not include punctuation
2. Form the set of all distinct tokens $\{t_1, t_2, ...\}$.
    - ignore *stop-words* (exact list depends on the application)
    - ignore tokens which are too rare and too frequent
    - account only for particular parts of speech (nouns, adjectives? verbs? ...)
3. May add *bigram/trigram collocations*
4. May normalize words:
    - *stemming*
        - fast, does not need dictionary
    - *lemmatization*
        - more accurate, needs dictionary

Table of Contents

Documents representation

- Typical representation of text for classification:
  - we evaluate only the presence of each distinct word in document d
  - order of words does not matter («*bag-of-words*» assumption)
- To account for word order - extract collocations as tokens

Term frequency

- Term-frequency model: $TF(i) = n_i$ or $TF(i) = \frac{n_i}{n}$
  - n_i is the number of times t_i appeared in d
  - n total number of tokens in d
  - second definition gives invariance to document length
- TF(i) measures how common is token t_i in the document.
- To make distribution of $TF(i) = n_i$ less skewed it is usually calculated as $TF(i) = ln(1 + n_i)$

Inverted document frequency

- Inverted document frequency: $IDF(i) = \frac{N}{N_i}$
    - N - total number of documents in the collection
    - N_i - number of documents, containing token t_i.
- $IDF(i)$ measures how specific is token i.
- To avoid skewness IDF is more frequently used as

$$IDF(i) = \ln\left(1 + \frac{N}{N_i}\right)$$

Vector representation of documents

- Consider document d and its feature representation x.

- Indicator model: $x^i = \mathbb{I}[t_i \in d]$.
- TF model: $x^i = TF(i)$
- TF-IDF model: $x^i = TF(i) * IDF(i)$

- Several representations, indexed by $I_1, I_2, ...I_K$ can be united into single feature representation

Properties of standard documents representation

- Properties of standard documents representation:
  - high dimensionality - at least D.
  - very sparse (few features not equal to zero)[1]
- Reduction of feature space
  - remove stop-words
  - remove words which are too frequent or too rare
  - remove words, irrelevant for current task
    - e.g. leave only nouns for topic modelling, adjectives+particles+adverbs for sentiment analysis, etc.
  - stemming / lemmatization
  - feature selection
  - dimensionality reduction
- Linear models (such as linear/logistic regression, SVM) work well.
  - have minimal complexity so overfit less for high D

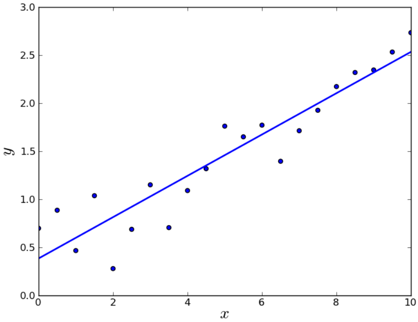
[1] in python use scipy.sparse

Linear regression

- Linear regression

$$\widehat{y}(x) = \beta_0 + \beta^T x = \beta_0 + \beta_1 x^1 + ... + \beta_D x^D$$

- Parameters: $\beta = [\beta_1, ..\beta_D]^T$, β_0



Linear regression estimation

- Usually it is estimated with

$$\sum_{n=1}^{N}\left(\beta_0+\beta^T x_n-y_n\right)^2+\lambda R(\beta)\to\min_{\beta}$$

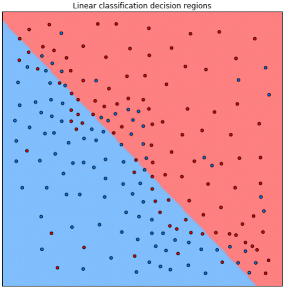
- λ is regularization parameter, $\uparrow\lambda\Leftrightarrow$ complexity$\downarrow$.
- Ridge regression:$R(\beta)=\sum_{d=1}^{D}\beta_d^2$
  - for correlated features spreads weights equally among them
- LASSO regression: $R(\beta)=\sum_{d=1}^{D}|\beta_d|$
  - for correlated features selects one of them
  - performs automatic feature selection

Linear classifier

- Consider binary classification: $y \in \{+1, -1\}$
  - muticlass classification can be performed with many binary classifiers.
- Linear classifier:

$$\widehat{y}(x) = \text{sign}\left(\beta_0 + \beta^T x\right) = \text{sign}\{\beta_0 + \beta_1 x^1 + ... + \beta_D x^D\}$$

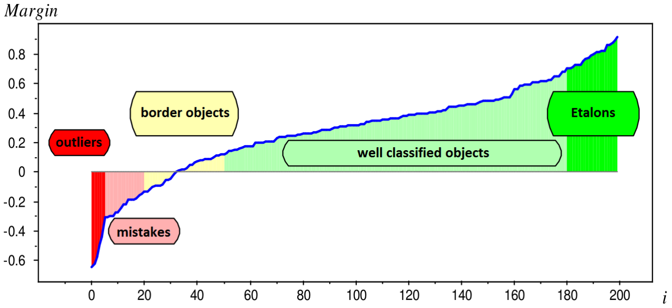
- Estimated parameters: $\beta = [\beta_1, ..\beta_D]^T, \beta_0$.



Margin

- Define the margin $M(x,y) = y\left(\beta_0 + \beta^T x\right)$
  - $M(x,y) \geq 0 <=>$ object x is correctly classified as y
  - $|M(x,y)|$ - confidence of decision
- Margin shows the score of classifying object (x,y).
  - the more, the better

Categorization of objects with respect to margin



Weight optimization

- Weights found with

$$\sum_{n=1}^{N} \mathcal{L}((\beta_0 + \beta^T x_n) y_n) + \lambda R(\beta) \to \min_{\beta_0, \beta}$$

- λ is regularization parameter, $\uparrow \lambda \Leftrightarrow$ complexity$\downarrow$.
- Ridge regression:$R(\beta) = \sum_{d=1}^{D} \beta_d^2$
  - for correlated features spreads weights equally among them
- LASSO regression: $R(\beta) = \sum_{d=1}^{D} |\beta_d|$
  - for correlated features selects one of them
  - performs automatic feature selection
- $\mathcal{L}(M) = max\{1 - M, 0\}$ => SVM
- $\mathcal{L}(M) = \ln(1 + e^{-M})$ => logistic regression

Different account for different features

- Optimization task for regression and classification:

$$\sum_{n=1}^{N}\mathcal{L}(x_n, y_n|\beta,\beta_0)+\lambda R(\beta)\to\min_{\beta_0,\beta}$$

- Suppose we have K groups of features with indices: $I_1, I_2, ... I_K$
    - nouns, adjectives, verbs, etc.
    - indicators, TF, TF-IDF
    - unigrams, bigrams
    - etc.
- We may control the impact of each group on the model:

$$\sum_{n=1}^{N}\mathcal{L}(\widehat{y}_n, y_n|\beta,\beta_0)+\lambda_1 R(\{\beta_i|i\in I_1\})+...+\lambda_K R(\{\beta_i|i\in I_K\})\to\min_{\beta_0,\beta}$$

- $\lambda_1, \lambda_2, ...\lambda_K$ can be set using cross-validation.
- Scikit-learn allows to set only single λ. But we can control impact of each feature group by different feature scaling.

Metric methods of text classification

- Metric methods typically use:
  - Euclidean distance $\sqrt{\sum_d (x_d - z_d)^2}$
  - cosine similarity: $\frac{\langle x,z \rangle}{\|x\|\|z\|}$
    - equal to cosine of angle between x and z
    - invariant to document size (norms of x and z)
    - cosine distance = 1-cosine similarity
- Rochio method
  - equivalent name - nearest centroid
  - $O(ND)$ training time, $O(CD)$ test time
  - fails for non-linear boundary
- K-NN
  - weighted K-NN can use weights $\propto$ cosine similarity (x, x_n)
  - $O(ND)$ training time (memorization), $O(ND)$ test time.

Table of Contents

Naive Bayes assumption

Bayesian minimum error decision rule:

$$\widehat{y}(x) = \arg\max_y p(y|x) = \arg\max_y \frac{p(y,x)}{p(x)} = \arg\max_y p(y)p(x|y)$$

Naive Bayes assumption

Bayesian minimum error decision rule:

$$\widehat{y}(x) = \arg\max_y p(y|x) = \arg\max_y \frac{p(y,x)}{p(x)} = \arg\max_y p(y)p(x|y)$$

$$p(x^1, x^2, ...x^D|y) = p(x^1|y)p(x^2|y, x^1)...p(x^D|y, x^1, x^2, ...x^{D-1})$$

Problem: exponential to D number of observations needed for estimation.

Naive Bayes assumption

Bayesian minimum error decision rule:

$$\widehat{y}(x) = \arg\max_y p(y|x) = \arg\max_y \frac{p(y,x)}{p(x)} = \arg\max_y p(y)p(x|y)$$

$$p(x^1, x^2, ...x^D|y) = p(x^1|y)p(x^2|y, x^1)...p(x^D|y, x^1, x^2, ...x^{D-1})$$

Problem: exponential to D number of observations needed for estimation.

Naive Bayes assumption in classification

Individual features are **class conditionally** independent:
$p(x|y) = p(x^1|y)p(x^2|y)...p(x^D|y)$

With Naive Bayes max-posterior probability rule becomes:

$$\widehat{y}(x) = \arg\max_y p(y)p(x^1|y)p(x^2|y)...p(x^D|y)$$

Generative text models

- Restrict attention to D words $w_1, w_2, ...w_D$
- Two major models:
  - Bernoulli
    - considers $x^i = \mathbb{I}[w_i \text{ appeared in the document}]$
  - Multinomial
    - considers $x^i =$[number of times w_i appeared in the document]

Bernoulli model[4]

- $w_1, w_2, ... w_D$-all unique words (tokens) in dictionary
- Decision rule:

$$\widehat{y}(x) = \arg\max_y p(y)p(x|y)$$

- $x \in \mathbb{R}^D$, $x^i = \mathbb{I}[w_i \text{ appeared in the document}]$, $i = \overline{1, D}$
- Document generation of class y: **for each word** w_d **generate its occurence in document** with $Bernoulli(\theta_y^d)$.
- $p(y) = \frac{N_y}{N}$
- $p(x|y) = \prod_{d=1}^D \left(\theta_y^d\right)^{x^d} \left(1 - \theta_y^d\right)^{1-x^d}$
- $\theta_y^d = p(x^d = 1|y) = \frac{N_{yx^d}}{N_y}$
- Smoothed variant[2][3]: $\theta_y^d = \frac{N_{yx^d}+\alpha}{N_y+2\alpha}$

[2] interpret this in terms of adding artificial observations

[3] modify for smoothing towards uncoditional word distribution

[4] is it linear classifier?

Multinomial model

- $w_1, w_2, ...w_D$-all unique words (tokens) in dictionary
- Decision rule:

$$\widehat{y}(x) = \arg\max_y p(y)p(x|y)$$

- $x \in \mathbb{R}^D$, x^i =[number of times w_i appeared in the document], $i = \overline{1, D}$
- Document generation of class y: **for each word-position** $i = 1, 2, ...n_{document}$ **generate word** z_i with Categorical$(\theta_1^y, \theta_2^y, ...\theta_D^y)$.
- θ_i^y =[probability of w_i on word position]

Multinomial model[7]

- $\left(\sum_i x^i\right)!$ - number of permutations of all words
- $\prod_i \left(x^i\right)!$ - number of permutations of words withing groups (of the same word)
- $\frac{\left(\sum_i x^i\right)!}{\prod_i (x^i)!}$ - number of permutations of word groups.
- Since permutation of word groups do not affect word counts $[x^1, ...x^D]$ in the document:

$$p(x|y) = \frac{\left(\sum_i x^i\right)!}{\prod_i (x^i)!} \prod_{i=1}^{D} \left(\theta_i^y\right)^{x^i}$$

- $p(y) = \frac{N_y}{N}$, $\theta_i^y = n_{yi}/n_y$ where
  - n_{yi} - number of times word w_i appeared in documents$\in y$
  - n_y - number of words in documents$\in y$
- Smoothed version[5][6]: $\theta_y^d = \frac{n_{yd+\alpha}}{n_y+\alpha D}$

[5] interpret this in terms of adding artificial observations

[6] modify for smoothing towards unconditional word distribution

Discussion

- For prediction discriminative models are preferred to generative
    - they do not model high dimensional $p(x|y)$
    - do not rely upon Naive Bayes assumption
- Advantages of generative models
    - can adapt to changes in $p(y)$
    - can filter outliers by $p(x)$
    - Multinomial and Bernoulli fit in $O(N)$.

Table of Contents

Feature selection

- Feature selection - select words with most discriminative information about document classes.
- We estimate criterion $I(w)$, order words by decreasing $I(w)$ and select features to top K values of $I(w)$.
- Define $p(c|w) = p(y = c|\text{word } w \text{ is present})$ - conditional probability of c-th class of document, given it contains word w.

- **Information gain** (as in decision trees) measures difference in class uncertainty defore and after obsering presense of word w:

$$\begin{aligned} I(w) &= Entropy(c) - Entropy(c|w) \\ &= -\sum_c p(c) \ln p(c) + p(w) \sum_c p(c|w) \ln p(c|w) \\ &+ (1 - p(w)) \sum_c (1 - p(c|w)) \ln(1 - p(c|w)) \end{aligned}$$

Word informativeness criteria

- Natural measures of discrimination by w:

$$I(w) = std.dev\left(\{p(c|w)\}_{c=1}^{C}\right)$$
$$I(w) = \max\left(\{p(c|w)\}_{c=1}^{C}\right) - \min\left(\{p(c|w)\}_{c=1}^{C}\right)$$

- Gini index for word w:

$$G(w) = \sum_{c=1}^{C} p(c|w)^2$$

- To avoid misleading results for these 2 measures when classes are unbalances ($\max_c p(y=c) - \min_c p(y=c)$ is large) we replace $p(c|w)$ with $p'(c|w)$:

$$p'(c|w) = \frac{p(y=c|w)/p(y=c)}{\sum_i p(y=i|w)/p(y=i)}$$

Word informativeness criteria

- Mutual information

$$I_c(w) = \ln\left(\frac{p(w,c)}{p(w)p(c)}\right) = \ln\left(\frac{p(w)p(c|w)}{p(w)p(c)}\right) = \ln\left(\frac{p(c|w)}{p(c)}\right)$$

- χ^2-statistic (test H_0: occurence of w and occurence of class c are independent)

$$I_c(w) = \frac{Np(w)^2\left(p(c|w) - p(w)\right)^2}{p(w)\left(1 - p(w)\right)p(c)\left(1 - p(c)\right)}$$

- 2 previous mesures estimate word informativeness with respect to class.
- Informativness of w for all classes can be generated by:

$$I(w) = \sum_c p(c) I_c(w)$$

$$I(w) = \max_c I_c(w)$$