

Working with text

Victor Kitov

`v.v.kitov@yandex.ru`

Text mining

- In text mining feature space is usually high dimensional and sparse.
- To handle sparsity design matrix X may be stored in *sparse matrix format*.
- Linear models work well in high dimensional spaces
 - models are already complex due to many features
 - non-linear models have much more parameters and overfit
- Examples of linear models:
 - regression: linear regression with different regularization
 - classification: logistic regression, SVM
- We may also use arbitrary models in diminished feature space with
 - feature extraction (using, for example, PCA)
 - feature selection (using correlation, mutual information, etc.)

Token set

- 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 $\{t_1, t_2, \dots\}$.
 - 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*
 - faster
 - *lemmatization*
 - more accurate

Term frequency

- Term-frequency model: $TF(i) = \frac{n_i}{n}$
 - n_i is the number of times t_i appeared in d
 - n total number of tokens in d .
- $TF(i)$ measures how common is token t_i in the document.
- To make $TF(i)$ less skewed it is usually calculated as

$$TF(i) = \ln \left(1 + \frac{n_i}{n} \right)$$

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, \dots, I_K can be united into single feature representation.

Different account for different features

- Optimization task with regularization:

$$\sum_{n=1}^N \mathcal{L}(\hat{y}_n, y_n | \mathbf{w}) + \lambda R(\mathbf{w}) \rightarrow \min_{\mathbf{w}}$$

- Here λ controls complexity of the model:

Different account for different features

- Optimization task with regularization:

$$\sum_{n=1}^N \mathcal{L}(\hat{y}_n, y_n | \mathbf{w}) + \lambda R(\mathbf{w}) \rightarrow \min_{\mathbf{w}}$$

- Here λ controls complexity of the model: $\uparrow \lambda \Leftrightarrow \text{complexity} \downarrow$.

Different account for different features

- Optimization task with regularization:

$$\sum_{n=1}^N \mathcal{L}(\hat{y}_n, y_n | \mathbf{w}) + \lambda R(\mathbf{w}) \rightarrow \min_{\mathbf{w}}$$

- Here λ controls complexity of the model: $\uparrow \lambda \Leftrightarrow \text{complexity} \downarrow$.
- Suppose we have K groups of features with indices:

$$I_1, I_2, \dots, I_K$$

- We may control the impact of each group on the model:

$$\sum_{n=1}^N \mathcal{L}(\hat{y}_n, y_n | \mathbf{w}) + \lambda_1 R(\{\mathbf{w}_i | i \in I_1\}) + \dots + \lambda_K R(\{\mathbf{w}_i | i \in I_K\}) \rightarrow \min_{\mathbf{w}}$$

- $\lambda_1, \lambda_2, \dots, \lambda_K$ 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 scaling.