Features

Bernd Schomburg

NOTATION. For a finite S we will denote its *cardinality* (= number of elements) by $|S|$. If S is a finite subset of $\mathbb{Z}$, the ring of all integers, then $\min S$ will denote its *minimum*, i.e. $\min S \in S$ and $\min S \leq s$ for all $s \in S$. For a function $f : A \to \mathbb{Z}$ on some finite set A its image $f(A)$ is finite, too, and we will write $\min_A f$ or $\min_{x \in A} f(x)$ for $\min f(A)$. If $f : A \to \mathbb{Z}$ is *one-to-one (injective)*, i.e. if $x \neq x'$, $x, x' \in A$ always implies $f(x) \neq f(x')$, then there is a unique $\bar{x} \in A$ such that $f(\bar{x}) = \min f(A)$, which we will denote by $\arg\min_A f$.

1 Features

Let S a finite set of samples (e.g. documents, emails, websites, PE files), $\mathcal{F}$ a finite set of features (e.g. strings/words) and

$$V : S \times \mathcal{F} \ni (s, f) \mapsto v_f(s) \in \mathbb{R}_0^+$$

a function; we will call $v_f(s)$ the **value** of the feature f for the sample s, $[v_f(s)]_{f \in \mathcal{F}}$ its **feature vector** and

$$\mathcal{F}_s = \{f \in \mathcal{F} \mid v_f(s) > 0\}$$

its **feature set**.

2 Feature extraction for lists

Let T be a finite set of tokens (for instance a set of words from a (finite) alphabet) which will serve as feature set. Moreover, let S be a finite set of finite lists (i.e. vectors) with entries from T; these could have been derived from a text. For samples we define various numerical features as follows.

2.1 Bags for feature extraction

We take $\mathcal{F} = T$ and define feature value function $V : S \times \mathcal{F} \to \mathbb{N}_0$ by

$$V(s, t) = |\{j \mid 1 \leq j \leq N, t_j = t\}|$$

for a sample $s = [t_1, \ldots, t_N] \in S$ and a token $t \in T$.

The function $V(s, \cdot)$ which maps each t to the its feature value $V(s, t)$ for the sample s is called the **bag** of tokens[1] of s. It is simply the *monogram* statistics of T in the sample s.

Example 2.1. *Bags are often encoded as dictionaries. Consider the text*

`John likes to watch movies. Mary likes movies too.`

and the token (feature) set

$$T = \{"Anna", "cinema", "John", "likes", "to", "watch", "movies", "Mary", "likes", "movies", "too"\}.$$

Then:

[1] or Bag of Words (BoW), in particular when the tokens are actually words

- $s = [\text{"}John\text{"}, \text{"}likes\text{"}, \text{"}to\text{"}, \text{"}watch\text{"}, \text{"}movies\text{"}, \text{"}Mary\text{"}, \text{"}likes\text{"}, \text{"}movies\text{"}, \text{"}too\text{"}]$
- $T_s = \{\text{"}John\text{"}, \text{"}likes\text{"}, \text{"}to\text{"}, \text{"}watch\text{"}, \text{"}movies\text{"}, \text{"}Mary\text{"}, \text{"}too\text{"}\}$
- $V(s, \cdot) = \{\text{"}Anna\text{"} : 0, \text{"}cinema\text{"} : 0, \text{"}John\text{"} : 1, \text{"}likes\text{"} : 2, \text{"}to\text{"} : 1, \text{"}watch\text{"} : 1, \text{"}movies\text{"} : 2, \text{"}Mary\text{"} : 1, \text{"}too\text{"} : 1\}$

Algorithm 1: Bag of tokens

Input: A set T of tokens, a list s of tokens
Output: The bag of tokens of s
Initialize B as the dictionary with keys in T and value 0 for each key t.
for $t \in s$ **do**
| $B[t] \leftarrow B[t] + 1$
end
return B

2.2 Bigram statistics as features

We can create feature values for pairs of tokens, i.e. use $\mathcal{F} = T \times T$ as a feature set as follows. We define $V_{bigr} : S \times \mathcal{F} \to \mathbb{N}_0$ by

$$V_{bigr}(s, (t, t')) = |\{j \mid 1 \leq j < N, t_j = t, t_{j+1} = t'\}|$$

for $s = [t_1, ..., t_N] \in S$ and $t, t' \in T$, i.e. $V_{bigr}(s, \cdot)$ is the *bigram* statistics of T in the sample s.We can normalise this feature value function as follows: Take

$$m(s, t) = \sum_{t' \in T} V_{bigr}(s, (t, t'))$$

for $t \in T$. Note that $V(s, t) - 1 \leq m(s, t) \leq V(s, t)$

$$v_{(t,t')}(s) = \begin{cases} 0, & \text{if } m(s,t) = 0, \\ \frac{V_{bigr}(s,(t,t'))}{m(s,t)}, & \text{if } m(s,t) \neq 0. \end{cases}$$

Then the matrix $v(s) = [v_{(t,t')}(s)]_{t,t' \in T} \in [0, 1]^{T \times T}$ can be used as a feature (array) for s.

3 Feature hashing

If $\mathcal{F}$ is very large, the feature vector $[v_f(s)]_{f \in \mathcal{F}}$ of a sample s could be simply too large for numerical calculations. In this case one can use the so-called **hashing trick**. Let

$$H : \mathcal{F} \to \mathcal{X}$$

be a hash function with $m = |\mathcal{X}|$ small. Then we use $\mathcal{X}$ as the new feature set where we define the value of a feature x for a sample s by

$$\hat{v}_x(s) = \sum_{f : H(f) = x} v_f(s),$$

so that the feature vector $[\hat{v}_x(s)]_{x \in \mathcal{X}}$ has only m entries.

In pratical implementations one takes $\mathcal{X} = \{0, ..., m - 1\}$ (e.g. $m = 1024$), and

$$H = \mathcal{H} \mod m,$$

where $\mathcal{H}$ is a standard n-bit hash function with values between 0 and $2^n - 1$ such that $2^n - 1 > m$ (e.g. $n = 32$) and we can assume that $\mathcal{F}$ is contained in the domain of definition of $\mathcal{H}$.

The following algorithms apply the hashing trick to bags and calculate the feature vector $[v_x]_{0 \leq x < m}$ with $v_x = |\{j \mid H(t_j) = x\}|$ for a given list $s = [t_1, ..., t_N]$.

Algorithm 2: Feature extraction

Input: A list s of tokens.
Output: A list of length m of feature values.
Initialize $feature\_values$ as a list of length m.
for $x \leftarrow 0$ **to** $m-1$ **do**
  $feature\_values[x] \leftarrow 0$
  for $t \in s$ **do**
    if $H(t) = x$ **then**
      $feature\_values[x] \leftarrow feature\_values[x] + 1$
    end
  end
end
return $feature\_values$

Algorithm 3: Implementation of Algorithm 1 in Python

```
import numpy as np
def extract_features(list_of_tokens):
    hash_buckets = [H(w) for w in list_of_tokens]
    buckets, counts = np.unique(hash_buckets, return_counts=True)
    feature_values = np.zeros(m)
    for bucket, count in zip(buckets, counts):
        feature_values[bucket] = count
    return feature_values
```

4 Similarity and MinHash

For the material in this section see also [1].

Definition 4.1. *Let $\mathcal{X}$ be finite (and non-empty) and $A, B \subseteq \mathcal{X}$, not both empty. The* **Jaccard**[2] **index** *$J(A,B)$ is defined by*

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|};$$

it measures to what extent the sets have elements in common. Obviously, $J(A,B) = 0 \Leftrightarrow A \cap B = \emptyset$ and $J(A,B) = 1 \Leftrightarrow A = B$.

Remark 4.2. *(i) With this notion we consider two sets A and B as close or similar if*

$$J(A,B) \geq \alpha,$$

where $\alpha \in (0,1)$ is some number close to 1, e.g. 0.8.
(ii) $d(A,B) = 1 - J(A,B)$ defines a metric on the set of all non-empty subsets of $\mathcal{X}$.

In pratice, it will be difficult to actually calculate the Jaccard index. We have to take recourse to statistical methods.

Let $H : \mathcal{X} \to \mathbb{Z}$ be a one-to-one function.

Theorem 4.3. *We have*

$$J(A,B) = \frac{1}{m!} |\{\pi \in Sym(\mathcal{X}) \mid \min_A (H \circ \pi) = \min_B (H \circ \pi)\}|,$$

where $m = |\mathcal{X}|$ and $Sym(\mathcal{X})$ denotes the symmetric group of all permutations on $\mathcal{X}$.

[2] Paul Jaccard (1868-1944) Swiss botanist and plant physiologist

Algorithm 4: Calculation of the Jaccard index, cf. [2, p.73]

```
def jaccard(A,B):
    intersection= A.intersection(B)
    intersection_length = float(len(intersection))
    union= A.union(B)
    union_length = float(len(union))
    return intersection_length/union_length
```

Proof. For $S \subseteq \mathcal{X}$ and $s \in S$ put $\Lambda_s = \{\pi \mid \arg\min_S(H \circ \pi) = s\}$. Then

$$Sym(\mathcal{X}) = \bigcup_{s \in S} \Lambda_s,$$

and, since H is one-to-one, $\Lambda_s \cap \Lambda_t = \emptyset$ if $s \neq t$. Thus

$$m! = |Sym(\mathcal{X})| = \sum_{s \in S} |\Lambda_s|.$$

Finally, if $s, t \in S$, then

$$\Lambda_s \ni \pi \mapsto \pi \circ (st) \in \Lambda_t,$$

where (st) is the transposition of s and t, is a bijection. Thus all the Λ_s, $s \in S$, have the same cardinality $m!/|S|$, and, if T is a subset of S, then

$$|\{\pi \mid \arg\min_S(H \circ \pi) \in T\}| = \sum_{t \in T} |\Lambda_t| = |T|\frac{m!}{|S|}.$$

In particular, if $S = A \cup B$ and $T = A \cap B$, then

$$J(A, B) = \frac{1}{m!}|\{\pi \mid \arg\min_{A \cup B}(H \circ \pi) \in A \cap B\}|.$$

The assertion now follows from the following lemma. ☐

Lemma 4.4. *Let $f : \mathcal{X} \to \mathbb{Z}$ be one-to-one. Then we have*

$$\min_A f = \min_B f \Leftrightarrow \arg\min_{A \cup B} f \in A \cap B.$$

Proof. Exercise. ☐

Remark 4.5. *For π put*

$$J_\pi(A, B) = \begin{cases} 1, & \min_A(H \circ \pi) = \min_B(H \circ \pi), \\ 0, & \text{otherwise.} \end{cases}$$

Then

$$J(A, B) = \frac{1}{m!} \sum_\pi J_\pi(A, B). \tag{4.1}$$

Now let Π be a uniformly distibuted random permutation on $\mathcal{X}$, i.e. a measurable function with values in $Sym(\mathcal{X})$ such that

$$\mathbb{P}(\Pi = \pi) = \frac{1}{m!}$$

for each $\pi \in Sym(X)$, and consider the random variable

$$J_\Pi(A, B) = \begin{cases} 1, & \min_A(H \circ \Pi) = \min_B(H \circ \Pi), \\ 0, & \text{otherwise.} \end{cases}$$

Theorem 4.6. $\mathbb{E}(J_\Pi(A,B)) = J(A,B)$.

Proof. By (4.1) we have

$$\mathbb{E}(J_\Pi(A,B)) = \sum_\pi J_\pi(A,B)\mathbb{P}(\Pi=\pi) = \frac{1}{m!}\sum_\pi J_\pi(A,B) = J(A,B).$$

□

Lemma 4.7. *Let $X_1, ..., X_k$ be independent identically distibuted (i.i.d.) random variables such that $0 \le X_j \le 1$ and $X = \frac{1}{k}\sum_{j=1}^k X_j$. Then for $\epsilon > 0$*

$$\mathbb{P}(|X - \mathbb{E}(X)| \ge \epsilon) \le 2e^{-\frac{k\epsilon^2}{2}}. \tag{4.2}$$

Proof. We will make use of Hoeffding[3]'s inequality which says that if $Y_1, \dots, Y_k$ are i.i.d. random variables such that $\mathbb{E}(Y_j) = 0$ and $a_j \le Y_j \le b_j$ for $j = 1, \dots, k$ and if $\epsilon > 0$, then

$$\mathbb{P}(\sum_{j=1}^k Y_j \ge \epsilon) \le e^{-t\epsilon}\prod_{j=1}^k e^{\frac{t^2(b_j-a_j)^2}{8}} \tag{4.3}$$

for all $t > 0$, see [3, Theorem 4.4].
Now let $\epsilon > 0$ and put

$$Y_j = \frac{1}{k}(X_j - \mathbb{E}(X_j)).$$

Then $Y_j, j = 1, ..., k$, are i.i.d. with $\mathbb{E}(Y_j) = 0$ and

$$-\frac{1}{k} \le Y_j \le \frac{1}{k}.$$

Thus, if we insert $a_j = -1/k, b_j = 1/k$ into (4.3), then we obtain

$$\mathbb{P}(X - \mathbb{E}(X) \ge \epsilon) \le e^{-t\epsilon + \frac{t^2}{2k}}.$$

Analogously,

$$\mathbb{P}(-(X - \mathbb{E}(X)) \ge \epsilon) \le e^{-t\epsilon + \frac{t^2}{2k}},$$

so that

$$\mathbb{P}(|X - \mathbb{E}(X)| \ge \epsilon) \le 2e^{-t\epsilon + \frac{t^2}{2k}} \tag{4.4}$$

for all $t > 0$. (4.2) now follows if we take $t = k\epsilon$, for which the right hand side in (4.4) is minimized.

□

Theorem 4.8. *Let $\Pi_1, ..., \Pi_k$ be independent uniformly distributed random permutations on $\mathcal{X}$ and*

$$X = \frac{1}{k}|\{j \in \{1, ..., k\} \mid \min_A(H \circ \Pi_j) = \min_B(H \circ \Pi_j)\}|.$$

Then

$$\mathbb{E}(X) = J(A,B)$$

and

$$\mathbb{P}(|J(A,B) - X| < \epsilon) > 1 - \delta,$$

whenever $\epsilon > 0$, $0 < \delta < 1$ and $k > \frac{2}{\epsilon^2}\ln(\frac{2}{\delta})$.

[3] Wassily Hoeffding (1914–1991), Finnish-American statistician.

Proof. For $j = 1, ..., k$ let

$$X_j = \begin{cases} 1, & \min_A(H \circ \Pi_j) = \min_B(H \circ \Pi_j), \\ 0, & \text{otherwise.} \end{cases}$$

Then $X = \frac{1}{k}\sum_{j=1}^{k} X_j$ and assertion follows from Lemma 4.7. □

Remark 4.9. *Let $\mathcal{Y} = h(\mathcal{X})$. Theorem 4.8. can be reformulated as follows:*
Let $H_1, ..., H_k$ be independent uniformly distributed random bijection from $\mathcal{X}$ onto $\mathcal{Y}$ (that are uniformly distributed over the space of all bijections from $\mathcal{X}$ onto $\mathcal{Y}$) and

$$X = \frac{1}{k}|\{j \in \{1, ..., k\} \mid \min_A H_j = \min_B H_j\}|.$$

Then the conclusions of Theorem 4.8 hold.

Algorithm 5: MinHash estimator of the Jaccard index of two sets A and B

```
NUM_HASHES = 512
import numpy as np
import mmh3
def minhash(feature_set):
    minhashes = []
    for i in range(NUM_HASHES):
        minhashes.append(
            min([mmh3.hash(feature, i) for feature in feature_set])
            )
    return np.array(minhashes)

def similarity(A,B):
    mh_A = minhash(A)
    mh_B = minhash(B)
    return (mh_A == mh_B).sum()/float(NUM_HASHES)
```

Algorithm 6: MinHash on a set A

Input: A set $A \subseteq X$, k functions $H_k : X \to ?$
Output: The vector $[m_1(A), ..., m_k(A)]$
For $j = 1, ..., k$ initialize $c_j = \infty$.
for $x \in A$ **do**
    for $j = 1$ **to** k **do**
        if $h_j(x) < c_j$ **then**
            $c_j \leftarrow h_j(x)$
        end
    end
end
return $c_1, ..., c_k$

References

[1] J.M. Phillips. *Mathematical Foundations for Data Analysis.* Springer, 2021.

[2] J. Saxe. *Malware Data Science - Attack Detection and Attribution.* no starch press, 2018.

[3] L. Wasserman. *All of Statistics.* Springer, 2004.