WordVecs Lab

Julia L. Wang

wordVecArticles.txt: set of text from 10 wikepedia articles (D), 1 article / line (d)

wordVecTitles.txt: corresponding titles of the articles

wordVecWords.txt: pre-processed set for the union of words in all articles W, let each word be w with index t

wordVecV.mat: a 1651x10 vector V (t x d) of *term frequencies* $f_{\text{term}}(t, d)$: the number of times word w_t appears in d

```
% initialization
clear
load 'wordVecV.mat' %t row and d column
D = 10; %there are 10 articles/documents
W = 1651;
```

Part A

Finding the 2 articles closest in Euclidean distance and angle distance.
$v_d = (f_{\text{term}}(1, d), f_{\text{term}}(2, d), \ldots, f_{\text{term}}(|W|, d))$ for each d (article).

```
% each column of V is v_d, the frequency for each word in the d-th article
% take the transpose of V to get the dth article to be the dth row
v = transpose(V);

% finding closest Euclidean distance and angle distance
euclid_distance(v, D) % = 7 and 8
```

```
ans = 7
```

```
angle_distance(v, D) % = 9 and 10
```

```
ans = 9
```

```

%they could be different - they are the same when the vectors are
%normalized
```

Part B

Normalized vectors are the column vector (v_d) divided by the summation of all frequencies in that article.

$$v_d' = \frac{v_d}{\sum_{t=1}^{|W|} f_{\text{term}}(t, d)}$$

Which 2 are closest in distance and angle? Are they the same pair as Part A? What reason would we use this normalization?

```
% get new v after normalization
v_d = transpose( V ./ sum(V));
```

```
ans = 1×10
   365   386   367   588   258   644   234   207   540   505
```

```
% finding closest Euclidean distance and angle distance
euclid_distance(v_d, D) % = 9 and 10
```

```
ans = 9
```

```
angle_distance(v_d, D) % = 9 and 10
```

```
ans = 9
```

```
% Answer is diff: when normalized, the shortest euclidean and angle
% distances are the same.

% We do this normalization because we get the weight to which specific
% words are used compared to the total words in each document. Rather than
% counting the number of occurances, we look at the percentage of the word
% in that document.
```

Document frequency $f_{\mathrm{doc}}(t) = \sum_{d=1}^{|D|} \mathrm{I}[f_{\mathrm{term}}(t, d) > 0]$ with the indicator function: counts how many the t-th appears.

Term-frequency inverse document frequency score: $w(t, d) = \frac{f_{\mathrm{term}}(t, d)}{\sum_{t=1}^{|W|} f_{\mathrm{term}}(t, d)} \sqrt{\log\left(\frac{|D|}{f_{\mathrm{doc}}(t)}\right)}$

Part C

$w_d = (w(1, d), w(2, d), \ldots, w(|W|, d))$ find the 2 articles with smallest euclidean distance

```
f_doc = sum( transpose( min(v, 1)), 2); %f_doc is a 1651x1 matrix
    % counts how many documnets a word appears in

% square root part of w(t, d)
w_2 = sqrt( log(D./f_doc));

w = v_d  * w_2;

% euclidean distance
euclid_distance(w, D) % 8 and 9
```

```
ans = 8
```

Part D

Why use the inverse document frequency adjustment? What does the adjustment do geometrically?

```
% We use this frequency adjustment to assign weights to specific words.
% This is because some words are used more frequently for all documents,
% and not just a single one. For instance, "the" "a" "for" are words that
% appear at high frequency across multiple documents, however, do not
% provide keywords or insight into what the document is about.
```

```
% Geometrically: we multiply the normalized vectors by
    % sqrt(log(#documents/ # times a word shows up in a document))
% which acts as a weight -> as the # of times the word appears in a
% document increases, #doc/#word in doc decreases, and the log function
% decreases.
```

Distance Functions

```
function d = euclid_distance(v, rows)
    % initialize
    euc = 999;
    euc_article = 0;

    for i = 1:(rows-1) % for all of the rows
        e = pdist(v(i:(i+1), :)); %calculate distance to the next row (article)
        if e < euc
            euc = e;
            euc_article = i;
        end
    end
    d= euc_article;
end

function a = angle_distance(v, rows)
    % initialize
    angle = 360;
    ang_article = 0;

    for i = 1:(rows-1) % for all of the rows
        a = pdist(v(i:(i+1), :), 'cosine'); % calc angle to next article
        if a < angle
        angle = a;
        ang_article = i;
        end
    end
    a = ang_article;
end
```