

Data Preprocessing

The dataset contains two newsgroups, one is baseball and the other is hockey.

Each document contains 'From', 'Subject' headers and the main body. You should extract features for each document, for example:

You can simply cast each word in the document as a feature and assign the value to each word as the frequency of the word appears in the document. Of course, you can do advanced preprocessing like stop word removal, word stemming, or define specific features by yourself.

An example feature file is given :

```
+1 1:0 2:6.87475407647932 3:3.58860800796358 4:0 2:2.7 2.725475021777 30:0.228447564110819 49:2.1380940541094 56:3.84284214634782 90:2.83603740693284 114:8.60474895145252 131:2.41152410385398  
-1 1:0 4:0 30:0 2.28447564110819 78:0.915163336038847 116:2.4241825007259 304:3.47309512084174 348:13.941644886054 384:1.46427114637585 626:2.85545549278994 650:3.003091491596
```

where each line "[label] [keyword1]:[value1] [keyword2]:[value2]..." represents a document

Label(+1, -1) is the classification of the document;

keyword_i is the global sequence number of a word in the whole dataset;

value_i is the frequency of the word appears in the document.

The goal of data preprocessing:

Generate a feature file for the whole dataset, and then split it into five equal set of files say s1 to s5 for 5-fold Cross-validation.

Learning & Test Processing

Input: feature files s1 to s5

Steps:

- 1) Set $i=1$, take s_i as test set and others as training set
- 2) Take the training set and learn the SMO (\mathbf{w} and b for the data points) and store the learnt weights.
- 3) Take the test set and using the weights learnt from Step 2, classify the documents as hockey or baseball.
- 4) Calculate Precision, Recall and F1 score.
- 5) $i++$, Repeat Step 1, 2, and 3 until $i > 5$

Training

Objective Function

$$\begin{aligned} \min_{\vec{\alpha}} \quad & \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N y_i y_j K(\vec{x}_i, \vec{x}_j) \alpha_i \alpha_j - \sum_{i=1}^N \alpha_i, \\ & 0 \leq \alpha_i \leq C, \forall i, \\ & \sum_{i=1}^N y_i \alpha_i = 0. \end{aligned}$$

- The lecture in class showed us, we can:
 - Solve the dual more efficiently (fewer unknowns)
 - Add parameter C to allow some misclassifications
 - Replace $\mathbf{x}_i^\top \mathbf{x}_j$ by more more general kernel term

Intuitive Introduction to SMO

- SMO is essentially doing same thing with Perceptron learning algorithm
 - find a linear separator by adjusting weights on misclassified examples
- Unlike perceptron, SMO has to maintain sum over examples of example weight times example label
$$\sum_i \alpha_i y_i = 0.$$
- Therefore, when SMO adjusts weight of one example, it must also adjust weight of another

SMO Algorithm

- Input: C, kernel, kernel parameters, epsilon
- Initialize b and all α' s to 0
- Repeat until KKT satisfied (to within epsilon):
 - Find an example $e1$ that violates KKT (prefer unbound examples ($0 < \alpha_i < C$) here, choose randomly among those)
 - Choose a second example $e2$. Prefer one to maximize step size (in practice, faster to just maximize $|E_1 - E_2|$). If that fails to result in change, randomly choose unbound example. If that fails, randomly choose example. If that fails, re-choose $e1$.
 - Update α_1 and α_2 in one step
 - Compute new threshold b

Updating Two α

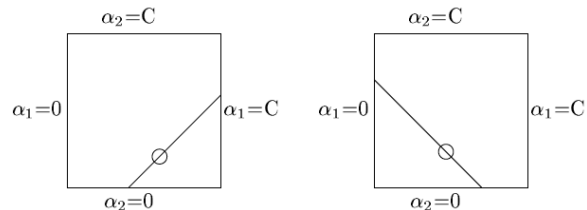
- Given examples e1 and e2, set

$$\alpha_2^{\text{new}} = \alpha_2 + \frac{y_2(E_1 - E_2)}{\eta} \quad \text{where:}$$

$$\eta = K(\vec{x}_1, \vec{x}_1) + K(\vec{x}_2, \vec{x}_2) - 2K(\vec{x}_1, \vec{x}_2)$$

- Clip this value in the natural way:

$$\alpha_2^{\text{new,clipped}} = \begin{cases} H & \text{if } \alpha_2^{\text{new}} \geq H; \\ \alpha_2^{\text{new}} & \text{if } L < \alpha_2^{\text{new}} < H; \\ L & \text{if } \alpha_2^{\text{new}} \leq L. \end{cases}$$



if $y_1 = y_2$ then:

$$L = \max(0, \alpha_2 + \alpha_1 - C)$$

$$H = \min(C, \alpha_2 + \alpha_1)$$

otherwise:

$$L = \max(0, \alpha_2 - \alpha_1)$$

$$H = \min(C, C + \alpha_2 - \alpha_1)$$

- Set $\alpha_1^{\text{new}} = \alpha_1 + s(\alpha_2 - \alpha_2^{\text{new,clipped}})$ where $s = y_1 y_2$

Karush-Kuhn-Tucker (KKT) Conditions

- It is necessary and sufficient for a solution to our objective that all α 's satisfy the following:

$$\begin{aligned}\alpha_i = 0 &\Leftrightarrow y_i u_i \geq 1, \\ 0 < \alpha_i < C &\Leftrightarrow y_i u_i = 1, \\ \alpha_i = C &\Leftrightarrow y_i u_i \leq 1.\end{aligned}$$

- An α is 0 iff that example is correctly labeled with room to spare
- An α is C iff that example is incorrectly labeled or in the margin
- An α is properly between 0 and C (is “unbound”) iff that example is “barely” correctly labeled (is a support vector)
- We just check KKT to within some small epsilon, typically 10^{-3}

At End of Each Step, Need to Update b and w

- Compute b_1 and b_2 as follows

$$b_1 = E_1 + y_1(\alpha_1^{\text{new}} - \alpha_1)K(\vec{x}_1, \vec{x}_1) + y_2(\alpha_2^{\text{new,clipped}} - \alpha_2)K(\vec{x}_1, \vec{x}_2) + b$$

$$b_2 = E_2 + y_1(\alpha_1^{\text{new}} - \alpha_1)K(\vec{x}_1, \vec{x}_2) + y_2(\alpha_2^{\text{new,clipped}} - \alpha_2)K(\vec{x}_2, \vec{x}_2) + b$$

- Choose $b = (b_1 + b_2)/2$
- When α_1 and α_2 are not at bound, it is in fact guaranteed that $b = b_1 = b_2$

- $$\mathbf{w} = \sum_{i=1}^N \alpha_i y_i \mathbf{x}_i,$$

$$\Delta \mathbf{w} = \Delta \alpha_1 y_1 \mathbf{x}_1 + \Delta \alpha_2 y_2 \mathbf{x}_2.$$

Testing

Testing for SMO Algorithm

Pseudocode:

```
for (i = 0; i < D; i++) {  
    p = 0;  
    for (j = 0; j < D(i); j++) {  
        p += w[j] * x[j] ;  
    }  
    p+=b;  
  
    if (p <= 0)  
        predicated -> class 1  
    else  
        predicated -> class 2
```

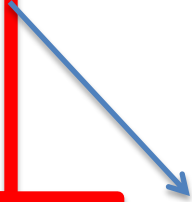
D-> Set of given Files (or) All unique words in given set of files.

D(i)-> Number of unique words in current processing file

y[i]-> label

x[i]-> input variant.

Example: Frequency of a word in the current processing file



Processing the current document (text file).

Precision & Recall

predicted class (observation)	actual class (expectation)	
	tp	fp
	(true positive) Correct result	(false positive) Unexpected result
	fn	tn
	(false negative) Missing result	(true negative) Correct absence of result

Precision is the probability that a (randomly selected) retrieved document is relevant.

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

Recall is the probability that a (randomly selected) relevant document is retrieved in a search.

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

F1 Score

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

Reference Code

- Pseudo-code
 - Reference 2
- C++ code
 - Reference 3

Submission

- Implementation
 - Report Precision, Recall, F1 measure
 - Report time cost
 - Submit as **zip file**, including
 - 1) Source Code with necessary comments, including
 1. Data preprocessing
 2. SMO training, testing and evaluation
 - 2) ReadMe.txt explaining
 1. How to extract features for the dataset
 2. Explain main functions, e.g., select α_1 and α_2 , update α_1 and α_2 , update w and b .
 3. How to run your code, from data preprocessing to training/testing/evaluation, step by step.
 4. What's the final average precision, recall and F1 Score and time cost.

References

1. *SVM by Sequential Minimal Optimization (SMO)*. Algorithm by John Platt. Lecture by David Page.
pages.cs.wisc.edu/~dpage/cs760/MLlectureSMO.ppt
2. Platt(1998):<http://research.microsoft.com/~jplatt/smoTR.pdf>
3. *Sequential Minimal Optimization for SVM* .
www.cs.iastate.edu/~honavar/smo-svm.pdf