Smoothing

9

- One drawback of word counting is that the number of parameters to estimate in the joint distribution over words and labels grows quadratically with the size of the vocabulary and label set. Thus many word-label pairs are never seen in the training set (Zipf's law makes this scaling even worse). One way to address this issue is via smoothing; which reduces the estimators variance at the expense of increasing its bias.

$$\hat{\phi}_{y,j} = \frac{\alpha + \sum_{i\,:\,y^{(i)}=y}^{M} x_j^{(i)}}{N\alpha + \sum_{j'=1}^{N} \sum_{i\,:\,y^{(i)}=y}^{M} x_{j'}}$$

where α is the smoothing parameter

Smoothing

- One drawback of word counting is that the number of parameters to estimate in the joint distribution over words and labels grows quadratically with the size of the vocabulary and label set. Thus many word-label pairs are never seen in the training set (Zipf's law makes this scaling even worse). One way to address this issue is via smoothing; which reduces the estimators variance at the expense of increasing its bias.

$$\hat{\phi}_{y,j} = \frac{\alpha + \sum_{i\,:\,y^{(i)}=y}^{M} x_j^{(i)}}{N\alpha + \sum_{j'=1}^{N} \sum_{i\,:\,y^{(i)}=y}^{M} x_{j'}}$$

where α is the smoothing parameter

Naive Bayes' classifier example w/Laplace smoothing

	Cat	Documents
Training	-	just plain boring
	-	entirely predictable and lacks energy
	-	no surprises and very few laughs
	+	very powerful
	+	the most fun film of the summer
Test	?	predictable with no fun

function TRAIN NAIVE BAYES(D, C) **returns** log $P(c)$ and log $P(w|c)$

for each class $c \in C$ # Calculate $P(c)$ terms
  N_{doc} = number of documents in D
  N_c = number of documents from D in class c
  $logprior[c] \leftarrow \log \frac{N_c}{N_{doc}}$
  $V \leftarrow$ vocabulary of D
  $bigdoc[c] \leftarrow$ **append**(d) **for** d $\in$ D **with** class c
  for each word w in V # Calculate $P(w|c)$ terms
    $count(w,c) \leftarrow$ # of occurrences of w in $bigdoc[c]$
    $loglikelihood[w,c] \leftarrow \log \frac{count(w,c) + 1}{\sum_{w' \text{ in } V} (count(w',c) + 1)}$
return $logprior$, $loglikelihood$, V

function TEST NAIVE BAYES($testdoc$, $logprior$, $loglikelihood$, C, V) **returns** best c

for each class $c \in C$
  $sum[c] \leftarrow logprior[c]$
  for each position i in $testdoc$
    $word \leftarrow testdoc[i]$
    if $word \in V$
      $sum[c] \leftarrow sum[c] + loglikelihood[word,c]$
return $\text{argmax}_c\ sum[c]$

- Taken from Jurafsky & Martin, Chp 4