Natural Language Processing

gpande3@gatech.eud

Hw-2

1. A:

Q1: (a). $P(+|S) = ?$ $\quad P(-|S) = ?$

Using Conditional Independence of Naive Bayes:

S = ["This", "Film", "was", "hilarous", "I", "didn't", "Yawn", "once", "Not", "single" "bland", "moment", "Every", "Minute", "laugh"]

$P(+|S) = P(+)\,P(\text{hilarous}|+)\,P(\text{Yawn}|+)\,P(\text{bland}|+)\,P(\text{laugh}|+)$

$P(-|S) = P(-)\,P(\text{hilarous}|-)\,P(\text{Yawn}|-)\,P(\text{blan}|-)\,P(\text{laugh}|-)$

$P(+) = \frac{3}{6} = 0.5$

$P(-) = \frac{3}{6} = 0.5$

$P(\text{hilarous}|+) = \frac{2}{13}$ $\qquad P(\text{hilarous}|-) = \frac{2}{16}$

$P(\text{Yawn}|+) = \frac{1}{13}$ $\qquad P(\text{Yawn}|-) = \frac{3}{16}$

$P(\text{bland}|+) = \frac{1}{13}$ $\qquad P(\text{bland}|-) = \frac{2}{16}$

$P(\text{laugh}|+) = \frac{5}{13}$ $\qquad P(\text{laugh}|-) = \frac{2}{16}$

We will calculate log Probabilities, so

$P(+|S) = P(+) \sum_{i=1}^{n} \log P(\text{token}|+) = -3.757$

$P(-|S) = P(-) \sum_{i=1}^{n} \log P(\text{token}|-) = -3.737$

$P(-|S) > P(+|S)$

Hence label assign to given sentence S: '−'

B:

(b) Recomputing Using Laplace add-1 smoothening:

$P(+) = \frac{3}{6} = 0.5$

$P(-) = \frac{3}{6} = 0.5$

$P(\text{hilarous} \mid +) = \frac{2+1}{13+6} = \frac{3}{19}$

$P(\text{Yawn} \mid +) = \frac{1+1}{13+6} = \frac{2}{19}$

$P(\text{bland} \mid +) = \frac{1+1}{13+6} = \frac{2}{19}$

$P(\text{laugh} \mid +) = \frac{5+1}{13+6} = \frac{6}{19}$

$P(\text{hilarous} \mid -) = \frac{2+1}{16+6} = \frac{3}{22}$

$P(\text{Yawn} \mid -) = \frac{3+1}{16+6} = \frac{4}{22}$

$P(\text{bland} \mid -) = \frac{2+1}{16+6} = \frac{3}{22}$

$P(\text{laugh} \mid -) = \frac{2+1}{16+6} = \frac{3}{22}$

$P(+ \mid S) = \log(0.5) + \log(3/19) + \log(2/19) + \log(2/19) + \log(6/19)$

$= -3.558$

$P(- \mid S) = \log(0.5) + \log(3/22) + \log(4/22) + \log(3/22) + \log(3/22)$

$= -3.637$

$P(- \mid s) < P(+ \mid S)$

S would be classify as "+" sentence.

C:

(c): Additional feature that can be extracted from the text to improve classification is to make use of negation words like "didn't", "Not", as this words along with other words changes the meaning of the sentence. "didn't Yawn" = "Not Yawn" and "Not single" = "Every". These words can be useful in bigram or n-gram language modelling.

2:

[redacted] (a). Likelihood equation for unregularized Logistic Regression

$$L(w) = \prod_{i=1}^{n} ((p(z^{(i)})^{y^{(i)}} (1 - p(z^{(i)}))^{1-y^{(i)}}$$

where $p((z))$ is the conditional probability, i.e sigmoid function.

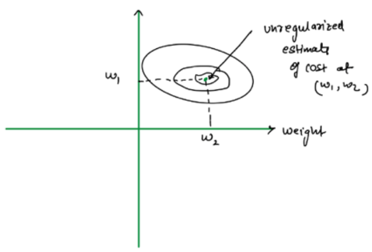
$$p((z)) = \frac{1}{1+e^{-z}}$$

taking log of the likehood function:

$$l(w) = \log(L(w))$$

$$= \sum_{i=1}^{n} y^{(i)} \log(p(z^i)) + (1-y^i)\log(1-p(z^i))$$

Now, the objective of this function is to minimize the cost by varying weights. for the unregularized cost, we could say that we find the global cost minimum for a perticular value of weights, and these weights are unconstrainted weights.

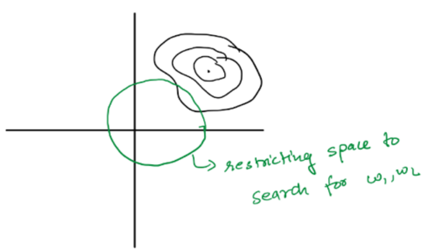


Now adding regularization term (L2)

L2:

$$l(w) = \left[\sum_{i=1}^{n} -y^i \log(p(z^i)) - (1-y^i)\log(1-p(z^i))\right] + \lambda \|w\|_2^2$$

Addition of regularization means we are increasing the cost by the euclidean norm of the weight vector. So, we penalize the cost if we go too far on our weights to over come overfitting. we restrict our search space to the regularized estimate.



Therefor the weight will reduce in regularized logistic regression.

$$\|\theta'\|_2^2 \leq \|\hat{\theta}\|_2^2$$

3: a:

(3). (a) $P(W) = P(w_1, w_2, \ldots w_n)$

$= P(w_1) P(w_2|w_1) P(w_3|w_1,w_2) \ldots P(w_n|w_1,w_2 \ldots w_{n-1})$ — (1)

$= \prod_{i=1}^{n} P(w_i | w_1^{i-1})$

To make n-gram model work, we make an assumption

$P(w_m | w_{m-1} \ldots w_1) \approx P(w_m | w_{m-1}, \ldots w_{m-n+1})$ — (2)

meaning we always only look for previous n-1 tokens

So, putting value of eq^n (2) in eq^n (1).

$$\boxed{\begin{aligned} P(w) &= P(w_1) P(w_2|w_1) P(w_3|w_1,w_2) \ldots P(w_m|w_1, \ldots w_{m-n+1}) \\ &= \prod_{m=1}^{M} P(w_m | w_{m-1}, \ldots, w_{m-n+1}) \end{aligned}}$$

for Bi-gram, $n=2$

$$\boxed{\begin{aligned} P(w) &= \prod_{m=1}^{M} P(w_m | w_{m-1}, \ldots w_{m-n+1}) \\ &= \prod_{m=1}^{M} P(w_m | w_{m-1}) \end{aligned}}$$

B:

(b): S: [BOS] I like cheese made at home [EOS]

Count ([BOS]) = 4

Count ([EOS]) = 4

Count (I) = 3

Count (like) = 2

Count (cheese) = 4

Count (made) = 3

Count (home) = 3

Count (is) = 2

Count (at) = 2

Count (tasty) = 1

Count (salty) = 1

Count (that) = 1

P([BOS] I like cheese made at home [EOS]) =

P(I | [BOS]) × P(like | I) × P(cheese | like) × P(made | cheese) × P(home | ma

$$= \frac{3}{4} \times \frac{2}{3} \times \frac{1}{2} \times \frac{1}{4} \times \frac{1}{3} \times \frac{1}{3}$$

$$= \frac{1}{12 \times 12}$$

$$= \frac{1}{144}$$

C:

(c): Perplexity : It is the measure of the likelihood.
lower perplexity corresponds to higher likelihood
higher perplexity corresponds to lower likelihood.

$$\text{Perplexity}(w) = 2^{-\frac{l(w)}{M}}$$

In our case of bigram model.

$$l(w) = \log_2 \frac{1}{144}$$

$$\text{Perplexity}(w) = 2^{-\frac{1}{M}\log(1/144)}$$

$M = 8$ [∵ total 8 tokens in the give Sentence]

$$= 2^{\frac{1}{8}\log(144)}$$

$$= (144)^{1/8} \qquad \left[\because 2^{\log_2 6} = 6^{\log_2 2} = 6\right]$$

D:

(d) Laplace smoothening.

We have given $k = 0.1$

# of Unique tokens in the given vocab = 12

Hence

$$P(w_1|w_2) = \frac{\text{Count}(w_2 w_1) + 0.1}{\text{Count}(w_2) + 12 \times 0.1}$$

$$P(w) = \frac{3+0.1}{4+1.2} \times \frac{2+0.1}{3+1.2} \times \frac{1+0.1}{2+1.2} \times \frac{1+0.1}{4+1.2} \times \frac{1+0.1}{3+1.2} + \frac{1+0.1}{3+1.2}$$

$$= \frac{4.1}{5.2} \times \frac{3.1}{4.2} \times \frac{2.1}{3.2} \times \frac{2.1}{5.2} \times \frac{2.1}{4.2} \times \frac{2.1}{4.2}$$

$$= \frac{4.1 \times 3.1 \times 2.1^4}{5.2^2 \times 4.2^3 \times 3.2}$$

$$= 0.000977$$

E:

(e): After adding [UNK] our vocab will become:

[BOS] i made cheese at home [EOS]

[BOS] i like home made cheese [EOS]

[BOS] cheese made at home is [UNK] [EOS]

[BOS] i like cheese [UNK] is [UNK] [EOS]

S: [BOS] i like pepperjack cheese [EOS]

adding [UNK] to the unknown vocab words.

S: [BOS] I like [UNK] cheese [EOS]

Now, count ([BOS]) = 4 count (home) = 3

count ([EOS]) = 4 count (at) = 2

count ([UNK]) = 3 count (is) = 2

count (cheese) = 4 count (I) = 3

count (like) = 2

count (made) = 3

$P(S) = P(I|[BOS]) \times P(like|I) \times P([UNK]|like) \times P(cheese|[UNK])$
$\times P([EOS]|cheese)$

$$= \frac{3+0.1}{4+1} \times \frac{2+0.1}{3+1} \times \frac{0.1}{2+1} \times \frac{1+0.1}{3+1} \times \frac{1+0.1}{4+1}$$

$$= \frac{3.1}{5} \times \frac{2.1}{4} \times \frac{0.1}{3} \times \frac{1.1}{4} \times \frac{1.1}{5}$$

$$= \frac{3.1 \times 2.1 \times 1.1^2 \times 0.1}{5^2 \times 4^2 \times 3}$$

$$= 0.000656$$

4:a:

(4) (a): Most hateful Words in Naive Bayes

["groid", "non-white", "dumb", "ape", "liberal", "scum", "asian", "mud", "non", "jews"]

Least Hateful Words

["-", "thanks", "html", "sf", "irish central", "facebook", "email", "10.00", "radio", "match"]

from the result, we can see that most hateful words are the common hate speech words and the model is able to identify it correctly. Also similarly for the least hateful words we can see that the trends follows the least hateful words in the vocab.

B:

[redacted]

(6). for logistic Regression with regularization:

Best tuned hyperparameters as

Learning Rate = 0.9 | Epochs = 2000

Regularization	Train Accuracy	Test Accuracy
0.0001	99.35	69.01
0.001	98.82	69.27
0.01	89.34	67.19
0.1	76.78	65.10
1	67.95	56.25
10	66.38	59.90

for unregularized version or

$\lambda = 0$

Train Accuracy = 99

Test Accuracy = 69

We can clearly observe from the above results and regularization parameter that when our lambda / regularization parameter reaches or close to zero, then it tends towards the overfitting solution which can be seen by the test accuracy approaching ≈ 99.9% while the test accuracy does not improve much.

Another observation is if the Lambda or regularization parameter is increased and reaches 1 or above model tends to underfit due to decrease in the train / test accuracy.

C: Implemented Bigram and remove all stop words using nltk library. Also I removed all the punctuations and numbers from the vocabulary, but since the size of the dataset was too small, the accuracy wasn't improved rather it decreased to 90 percent for train dataset and 50 percent for test dataset.