COMP90042 Natural Language Processing Workshop Week 3

Haonan Li – haonan.li@unimelb.edu.au

16, March 2020

Outline

- Text Classification
- N-gram Language Model
- Smoothing


Text Classification
Tokenization
Word2Vec
K-nearest Neighbor
Conditional Random Field (CRF)
Support Vector Machine (SVM)
Contextualized Word Representations
A Survey
RMDL
HDLTex
Deep Learning
Logistic Regression
Random Forest
Naive Bayes Classifier
FastText
Hierarchical Deep Learning for Text (HDLTex)
Recurrent Neural Networks (RNN)
Recurrent Convolutional Neural Networks (RCNN)
Term Frequency-Inverse Document Frequency
Deep Belief Network (DBN)
Convolutional Neural Networks (CNN)

Text Classification Discussion

1. What is text classification? Give some examples.
2. Why is text classification generally a difficult problem? What are some hurdles that need to be overcome?
3. Consider some (supervised) text classification problem, and discuss whether the following (supervised) machine learning models would be suitable:
   - k-Nearest Neighbour using Euclidean distance
   - k-Nearest Neighbour using Cosine similarity
   - Decision Trees using Information Gain
   - Naive Bayes
   - Logistic Regression
   - Support Vector Machines

What is Text Classification?

What is text classification? Give some examples.

- sentiment analysis
- author identification
- automatic fact-checking
- etc.

Why Text Classification Difficult?

Why is text classification generally a difficult problem?

- **document representation** — how do we identify **features** of the document which help us to distinguish between the various classes?

What are some hurdles that need to be overcome?

- Principal source of features: presence of tokens (words), (known as a **bag–of–words** model).
- many words don't tell you anything about the classes we want to predict, so **feature selection** is important.
- single words are often **inadequate at modelling the meaningful** information in the document
- Multi-word features (e.g. bi-grams, tri-grams) suffer from a **sparse data problem**.

Machine Learning models for Text Classification

Consider some (supervised) text classification problem, and discuss whether the these (supervised) machine learning models would be suitable:

- For a generic genre identification problem using an entire bag–of–words model (similar to the notebook) is as follows:

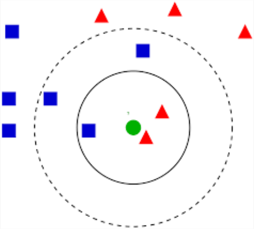
k-Nearest Neighbour

k-Nearest Neighbour using Euclidean distance

- Often this is a bad idea, because Euclidean distance tends to classify documents based upon their length — which is usually not a distinguishing characteristic for classification problems.

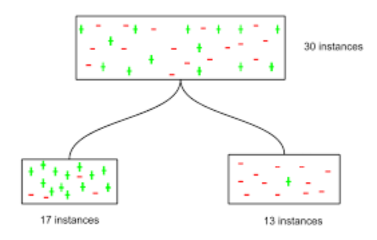
k-Nearest Neighbour using Cosine similarity

- Usually better than the previous, because we're looking at the distribution of terms. However, k-NN suffers from high-dimensionality problems,which means that our feature set based upon the presence of (all) words usually isn't suitable for this model.



Decision Trees using Information Gain

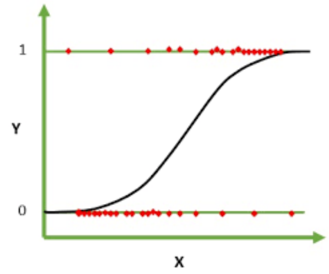
- Decision Trees can be useful for finding meaningful features, however, the feature set is very large, and we might find spurious correlations. More fundamentally, Information Gain is a poor choice because it tends to prefer rare features; in this case, this would correspond to features that appear only in a handful of documents.
- Random Forest?



Naive Bayes

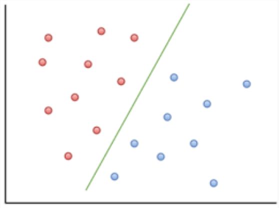
- At first glance, a poor choice because the assumption of the conditional independence of features and classes is highly untrue, e.g.?
- Also sensitive to a large feature set, in that we are multiplying together many (small) probabilities, which leads to biased interpretations based upon otherwise uninformative features.
- Surprisingly somewhat useful anyway!

Logistic Regression



- Useful, because it relaxes the conditional independence requirement of Naive Bayes.
- Since it has an implicit feature weighting step, can handle large numbers of mostly useless features, as we have in this problem

Support Vector Machines



- Linear kernels often quite effective at modelling some combination of features that are useful (together) for characterising the classes.
- How about multi-class?

Language Model

Tasks:

- Speech Recognition
- Spell Checking
- Machine Translation
- etc.

Probabilistic Language Model

Goal: get a probability for an arbitrary sequence of m words:

$$P(w_1, w_2, ...w_n)$$

First step: apply the chain rule to convert joint probabilities to conditional ones:

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

N-gram Model Discussion

For the following “corpus” of two documents:

1. `how much wood would a wood chuck chuck if a wood chuck would chuck wood`
2. `a wood chuck would chuck the wood he could chuck if a wood chuck would chuck wood`

(a). Which of the following sentences: `a wood could chuck`; `wood would a chuck;` is more probable, according to:

- I An unsmoothed uni-gram language model?
- II A uni-gram language model, with Laplacian (“add-one”) smoothing?
- III An unsmoothed bi-gram language model?
- IV A bi-gram language model, with Laplacian smoothing?
- V An unsmoothed tri-gram language model?
- VI A tri-gram language model, with Laplacian smoothing?

N-gram Model Discussion

Word Uni-grams Frequencies, totally 34 appearances:

how	much	wood	would	a	chuck	if	the	he	could	</s>
1	1	8	4	4	9	2	2	1	1	2

Some Bi-grams Frequencies:

<s> a	a wood	wood could	would a	wood would
1	4	0	1	1
<s> wood	could chuck	a chuck	chuck </s>	
0	1	1	0	

An unsmoothed uni-gram language model

- Simply based on the counts of words in the corpus. For example, out of the 34 tokens (including </s>) in the corpus, there were 4 instances of a, so P(a) = $\frac{4}{34}$
- Probability of a sentence: multiply the probabilities of the individual tokens:

$$\begin{aligned} P(A) &= \text{P(a)P(wood)P(could)P(chuck)P(</s>)} \\ &= \frac{4}{34} \cdot \frac{8}{34} \cdot \frac{1}{34} \cdot \frac{9}{34} \cdot \frac{2}{34} \approx 1.27 \times 10^{-5} \\ P(B) &= \text{P(wood)P(would)P(a)P(chuck)P(</s>)} \\ &= \frac{8}{34} \cdot \frac{4}{34} \cdot \frac{4}{34} \cdot \frac{9}{34} \cdot \frac{2}{34} \approx 5.07 \times 10^{-5} \end{aligned}$$

- Clearly sentence B has the greater likelihood, according to this model.

A uni-gram language model, with Laplacian ("add-one") smoothing

- For each probability, we add 1 to the numerator and the size of the vocabulary, which is 11, to the denominator. For example, $P_L(a) = \frac{4+1}{34+11} = \frac{5}{45}$.
- Everything else proceeds the same way:

$$\begin{aligned} P_L(A) &= P_L(a)P_L(wood)P_L(could)P_L(chuck)P_L(\text{</s>}) \\ &= \frac{5}{45} \cdot \frac{9}{45} \cdot \frac{2}{45} \cdot \frac{10}{45} \cdot \frac{3}{45} \approx 1.46 \times 10^{-5} \\ P_L(B) &= P_L(wood)P_L(would)P_L(a)P_L(chuck)P_L(\text{</s>}) \\ &= \frac{9}{45} \cdot \frac{5}{45} \cdot \frac{5}{45} \cdot \frac{10}{45} \cdot \frac{3}{45} \approx 3.66 \times 10^{-5} \end{aligned}$$

- Sentence B is still more likely.

An unsmoothed bi-gram language model i

- This time, we're interested in the counts of pairs of word tokens.
- We include sentence terminals, so that the first probability in sentence A is $P(\text{a}|\text{<s>}) = \frac{1}{2}$ — because one of the two sentences in the corpus starts with a. Now, we can substitute:

$$
\begin{aligned}
P(A) &= P(a|\text{<s>})P(wood|a)P(could|wood)P(chuck|could)P(\text{</s>}|chuck) \\
&= \frac{1}{2} \cdot \frac{4}{4} \cdot \frac{0}{8} \cdot \frac{1}{1} \cdot \frac{0}{9} = 0 \\
P(B) &= P(wood|\text{<s>})P(would|wood)P(a|would)P(chuck|a)P(\text{</s>}|chuck) \\
&= \frac{0}{2} \cdot \frac{1}{8} \cdot \frac{1}{4} \cdot \frac{1}{4} \cdot \frac{0}{9} = 0
\end{aligned}
$$

- Because there is a zero–probability element in both of these calculations, they can't be nicely compared, how to solve?

A bi-gram language model, with Laplacian smoothing

- We do the same idea as uni-gram add–one smoothing. The vocabulary size is 11.

$$
\begin{aligned}
P_L(A) &= P_L(a|\text{<s>})P_L(wood|a)P_L(could|wood)P_L(chuck|could)P_L(\text{</s>}|chuck) \\
&= \frac{2}{13}\cdot\frac{5}{15}\cdot\frac{1}{19}\cdot\frac{2}{12}\cdot\frac{1}{20} \approx 2.25\times 10^{-5} \\
P_L(B) &= P_L(wood|\text{<s>})P_L(would|wood)P_L(a|would)P_L(chuck|a)P_L(\text{</s>}|chuck) \\
&= \frac{1}{13}\cdot\frac{2}{19}\cdot\frac{2}{15}\cdot\frac{1}{15}\cdot\frac{1}{20} \approx 3.60\times 10^{-6}
\end{aligned}
$$

- This time, sentence A has the greater likelihood, mostly because of the common bi-gram `a wood`.

An unsmoothed tri-gram language model

- Same idea, longer contexts. Note that we now need two sentence terminals.

$$
\begin{aligned}
P(A) &= P(a|\text{<s> <a>})P(wood|\text{<s> a})\ldots P(\text{</s>}|\text{could chuck}) \\
&= \frac{1}{2}\cdot\frac{1}{1}\cdot\frac{0}{4}\cdot\frac{0}{0}\cdot\frac{0}{1} = ? \\
P(B) &= P(wood|\text{<s> <s>})P(would|\text{<s> wood})\ldots P(\text{</s>}|\text{a chuck}) \\
&= \frac{0}{2}\cdot\frac{0}{0}\cdot\frac{1}{1}\cdot\frac{0}{1}\cdot\frac{0}{0} = ?
\end{aligned}
$$

- Given that the unsmoothed bi-gram probabilities were zero, that also means that the unsmoothed tri-gram probabilities will be zero. Why?
- In this case, they aren't even well–defined, because of the $\frac{0}{0}$ terms, but we wouldn't be able to meaningfully compare these numbers in any case.

A tri-gram language model, with Laplacian smoothing

- The vocabulary size is 11. Everything proceeds the same way:

$$
\begin{aligned}
P_L(A) &= P_L(a|\text{<s> <a>})P_L(wood|\text{<s> a})\ldots P_L(\text{</s>}|\text{could chuck}) \\
&= \frac{2}{13} \cdot \frac{2}{12} \cdot \frac{1}{15} \cdot \frac{1}{11} \cdot \frac{1}{12} \approx 1.30 \times 10^{-5} \\
P_L(B) &= P_L(wood|\text{<s> <s>})P_L(would|\text{<s> wood})\ldots P_L(\text{</s>}|\text{a chuck}) \\
&= \frac{1}{13} \cdot \frac{1}{11} \cdot \frac{2}{12} \cdot \frac{1}{12} \cdot \frac{1}{11} \approx 8.83 \times 10^{-6}
\end{aligned}
$$

- Notice that the problem of unseen contexts is now solved (they are just $\frac{]1}{11}$).
- Sentence A has a slightly greater likelihood here, mostly because of the `a wood` at the start of one of the sentences.
- The numbers are getting very small, what to do?

Other Smoothing Strategies

- Add-k smoothing
- Absolute discounting
- Backoff
- Kneser-Ney smoothing

Continuation Probability

(b). Based on the “corpus”, the vocabulary = {`a, chuck, could, he, how, if, much, the, wood, would, </s>`}, and the continuation counts of the following words are given as follows:

a = 2, could = 1, he = 1, how = 0, if = 1, much = 1, the = 1, would = 2

What is the continuation probability of `chuck` and `wood`?

Continuation Probability

Continuation counts:

a = 2, could = 1, he = 1, how = 0, if = 1, much = 1, the = 1, would = 2

What is the continuation probability of `chuck` and `wood`?

- unique context words before `chuck: {wood, would, could, chuck }`
- unique context words before `wood: {the, much, a, chuck }`

$$
\begin{aligned}
P_{count}(chuck) &= \frac{\#chuck}{\#a + \#could + ... + \#chuck + \#would} \\
&= \frac{4}{2+1+1+0+1+1+1+2+1+4+4} \\
P_{count}(wood) &= \frac{\#wood}{\#a + \#could + ... + \#chuck + \#would} \\
&= \frac{4}{2+1+1+0+1+1+1+2+1+4+4}
\end{aligned}
$$

Back-off and Interpolation

What does **back–off** mean, in the context of smoothing a language model? What does **interpolation** refer to?

- Back–off is a smoothing strategy, where we incorporate lower–order n-gram models (in particular, for unseen contexts). For example, if we have never seen some tri-gram from our sentence, we can instead consider the bigram probability.
- Interpolation is a similar idea, but instead of only “falling back” to lower–order n-gram models for unseen events, we can instead consider every probability as a linear combination of all of the relevant n-gram models, where the weights are once more chosen to ensure that the probabilities of all events, given some context, sum to 1.

Questions ☺