Notes on Natural Language Processing by Eisenstein

compiled by D.Gueorguiev 9/21/2024

# Linear Text Classification

**Definition**: *Text Classification Problem*

Given a text document, assign a label from a set of discrete labels.

## The Bag of Words

Question: how do we represent a document with text?

Use a column vector of word counts:

where is count for the -th word in a vocabulary . Here the size of the vocabulary is denoted with .

In linear classification the decision is based on a weighted sum of individual word counts where the word set is the feature set of the classification problem. The classification object is the vector ; this object is often called *a bag of words*. With a bag of words representation we are ignoring everything else but the frequency count of each word – we are not accounting for grammatical and syntactic constructs, sentence boundaries, paragraphs.

To predict a label from a bag of words we assign a score to each word in the vocabulary measuring fitness of the word with this label. These word scores are known as weights and are stored in column vector .

Let us consider multi-class classifier where The goal is to predict a label , given the bag of words , using weights . For each label , we compute a score , which is a scalar measure of the compatibility between the bag-of-words and the label . In a linear bag-of-words classifier, this score is the vector inner product between the weights and the output of a *feature function*

(1)

For example, given arguments and , element of this feature vector might be

(2)

returns the count of the word *whale* if the label is , and it returns zero otherwise. The index depends on the position of *whale* in the vocabulary and of in the set of possible labels. The corresponding weight then scores the fitness of the word *whale* for the label . Positive score means that this word makes the label more likely.

The output of the feature function can be formalized as a vector:

(3)

(4)

(5)

where is a column vector of zeros, and the semicolon indicates vertical concatenation.

For each of the possible labels, the feature function returns a vector that is mostly zeros, with a column vector of word counts inserted in a location that depends on the specific label .

This notation may seem awkward but generalizes to a range of learning settings, particularly *structure prediction*.

Given a vector of weights, , we can now compute the score by Eq (1). This inner product gives a scalar measure of the fitness of the observation with respect to the label .

Note that only features and weights are necessary. That is true because we can require that regardless of . With this requirement it is possible to implement any classification rule that can be achieved with features and weights. This is akin to the binary classification rule , where is a vector of weights, is an offset , and the label set is . For we have – that is, we need one dimensional simplex (a line) to separate a label set of size 2.

Question: how can we obtain the weights ?

One option is to manually set those. For example if we want to distinguish English from Spanish, we can use English and Spanish dictionaries, and set the weight to one for each word that appears in the associated dictionary. For example:

,

,

,

,

Usually it is not easy to set classification weights by hand, due to the large number of words and the difficulty of selecting exact numerical weights. We will introduce a probabilistic method to generate such weights.

## Naïve Bayes

Notation:

– the joint probability that the random variables and take specific values and respectively. The subscripts indicating the random variables will be omitted hereon. Thus we end up with the more concise:

– the joint probability of a bag of words and its true label

– a dataset of labeled instances which are i.i.d.

– the joint probability of the entire dataset given with

One approach to classification is to set the weights as to maximize the joint probability of a *training set* of labeled documents. This is known as *maximum likelihood estimation*: