Log-Linear Models!

1 Introduction

Log-linear models are advantageous because they allow a far richer set of features, that provide better representation for what we want to model especially for various natural language processing problems.

Consider a sequence of words and a general lagnauge modeling problem i.e., we want to estimate the probability of word i being a certain word, given the previous $i-1$ words. We therefore want to model the distribution over the word w_i given the previous words $w1...w_{i-1}$. For an n-gram language model we assume that

$p(w_i \mid w_1...w_{i-1}) = q(w_i \mid w_{i-n+1}, .., w_{i-1})$

where $q(w_i \mid w_{i-n+1}, .., w_{i-1})$ is a parameter of the model for the n-gram. You can estimate the q parameters in a number of ways e.g., linear interpolation.

n-gram models are restricted in that they condition only on the previous $n-1$ words but not on any other features that those words could incorporate. Linear interpolation of all such features is obviously inefficient beyond a small number of estimates. Here, using log-linear models offers a more satisfactory method to incorporate all the contextual information as features.

2 Log-linear Models

Definition: A log-linear model primarily consists of the following components:

- A set $\mathcal{X}$ of possible inputs
- A set $\mathcal{Y}$ of possible labels
- A positive integer d denoting the number of features and parameters of the model
- A function $f : \mathcal{X} \times \mathcal{Y} \rightarrow R^d$
- A parameter vector $v \in R^d$

For any $x \in \mathcal{X}, y \in \mathcal{Y}$, the model defines a conditional probability:

$$p(y|x; v) = \frac{exp(v \cdot f(x, y))}{\sum_{y' \in \mathcal{Y}} exp(v \cdot f(x, y'))} \tag{1}$$

Here $exp(x) = e^x$ and $v \cdot f(x, y) = \sum_{k=1}^{d} v_k f_k(x, y)$ is the inner product between parameter vector v and $f(x, y)$.

Features: For any pair (x, y), $f(x, y) \in R^d$ is the feature vector representing that pair. Each dimension or component $f_k(x, y)$ for $k = 1...d$ is a distinct feature that represents some property of the input x in conjunction with label y. Each feature has an associated parameter v_k whose value is estimated from the training examples.

Notes on feature construction: In the language modeling problem we have our input x as a sequence of words $w_1..w_{i-1}$ and the label y is the new word in the sequence. You can have features as indicator functions representing various aspects i.e., a feature space of all unigrams where the component for a unigram w is set to 1 if label y is w.

Using feature templates makes this simpler i.e., assign unique IDs to elements seen in training data, to allow efficient lookup from a hash table. For example, if we index every trigram or every (bigram, previous_POS) pair or some other feature combination, we can then easily incorporate indicator functions and estimate parameters when needed.

Efficient feature representation: When implementing log-linear models, sparse features are efficient because we do not need to explicitly represent or manipulate the entire space i.e., d-dimensional feature vectors $f(x, y)$. It is more efficient to compute a function through hash tables s.t., for any pair (x, y), we only want the indices of relevant i.e., non-zero features. For example:

$$Z(x, y) = k : f)k(x, y) = 1 \tag{2}$$

When using hash functions we can now compute $Z(x, y)$ in $O(|Z(x, y)|)$ time, where $|Z(x, y)| << d$. Also consider the equation below, that is a central computation in log-linear models i.e., an inner product of the parameter vector and feature vector:

$$v \cdot f(x, y) = \sum_{k=1}^{d} v_k f_k(x, y) = \sum_{x \in Z(x,y)} v_k \tag{3}$$

Therefore instead of iterating from $1..d$, we can simply look only at non-zero indicator features and compute the inner product in $O(|Z(x,y)|)$ time.

3 Fundamental Model Equation

Consider a pair (x, y) such that $x \in X$ and $y \in Y$ in the training data. The conditional probability under the model is:

$$p(y|x;v) = \frac{exp(v \cdot f(x,y))}{\sum_{y' \in Y} exp(v \cdot f(x,y'))} \tag{4}$$

$$log(p(y|x;v)) = v \cdot f(x,y) = \log(\sum_{y' \in Y} exp(v \cdot f(x,y'))) \tag{5}$$

In the second equation, the inner product is linear in the features $f(x, y)$. The second term depends only on $x \in X$ and not on the label y. The log probability is therefore linear in the features $f(x, y)$ hence the origination of the name of the model.

4 Parameter Estimation

Log-likelihood function and Regularisation: Consider the training set of examples $(x^{(i)}, y^{(i)})$ for $i \in 1.n.n$ and $x^{(i)} \in X, y^{(i)} \in Y$. Given the parameter vector v, we can calculate the log conditional probability $log(p(y|x;v))$ from the equations above. For a particular example, the higher the log conditional probability, the better the model fits this example. Now the log-likelihood considers the sum of the log probabilties of examples in the training data:

$$L(v) = \sum_{i=1}^{n} log(p(y^{(i)}|x^{(i)};v)) \tag{6}$$

Therefore for any parameter vector v, this likelihood is a measure of how well v fits the training examples i.e., we have a function of the parameters v. One estimation method is the maximum likelihood estimation i.e., choose parameters given by:

$$v_{ML} = \arg\max_{x \in R^d} L(v) \tag{7}$$

Therefore this method seeks to find parameters that fit the training data as well as possible.