On Trace Norm Regularization for Document Modeling

Jason D. M. Rennie
jrennie@gmail.com

November 1, 2005

Abstract

We propose a model for text. At the lowest level, we use a simple language model which (at best) can only capture local structure. However, the model allows for these lowest-level components to be structured so that higher-level ordering and grouping which is present in the data may be reflected in the model.

1 The Data

We assume that the text data is made up of a set of units. We represent the word counts of each unit as a vector, $\vec{x}$. The unit may range from as small as a sentence to as large as a document or thread. The choice of the "unit" is a fundamental choice of the model designer. It should reflect the nature of the data along with the desired detail in which the text should be modeled.

We assume that each "unit" ($\vec{x}$) has a set of parameters ($\vec{\theta}$) specifying the likelihood of text in that unit. One simple likelihood which might be used is the multinomial,

$$P(\vec{x}|\theta) = \frac{\left(\sum_j x_j\right)!}{\prod_j x_j!} \prod_j \left(\frac{\exp(\theta_j)}{\sum_{j'} \exp(\theta_j)}\right)^{x_j}. \tag{1}$$

Note that $\vec{\theta}$ represents a natural parameterization of the multinomial. We assume that the unit parameters ($\vec{\theta}$) bottleneck information about the unit likelihood—if the unit parameters are given/known, no other model variables influence the unit likelihood.

2 The Model

A very important aspect of our model involves how the unit parameters are produced. In a simple model of text, one might use a single parameter vector that is optimized for the entire collecton. Or, if there are clear divisions within the data (such as classification labels), one might optimize one parameter vector per division. Alternatively, if no divisions are given a priori, one might try to learn them, e.g. find principal components, or cluster the data.

An alternative approach, which we favor and include in our model, is to allow one parameter vector per unit, but to regularize these parameter vectors so that complexity is added to the model only when there is a clear mandate in the data. In fact, this approach results naturally from a Bayesian viewpoint. Of course, the Bayesian viewpoint can be used to support many approaches. However, the Bayesian motivation along with the flexibility offered in our approach makes it, we think, more compelling than older approaches.

2.1 The Hierarchy

Now, we have specified from where the word-counts come—they are generated from the unit parameter vectors via a simple language model, such as the multinomial. But, from where do the unit parameter vectors originate? A Bayesian perspective says that we must answer this question. In particular, we must specify a distribution over the unit parameter vectors. We answer this question in two stages.

First, we establish a hierarchy over the units. Each unit's parameter vector corresponds to a leaf node in the hierarchy. There is a single root node; intermediary nodes (if any) link the leaf nodes to the root nodes. The hierarchy should be constrained to structures that are feasible according to the data. If the units are ordered (e.g. sentences within a document), the hierarchy should respect that ordering. We assume that all (valid) hierarchies are equally likely. One might imagine that the first step of our generative process is to draw a hierarchy out of a bag of hierarchies. We make this assumption for simplicity and because there does not seem to be any reason to prefer one structure over another a priori. Of course, if the data clearly indicates that a simpler or more complex hierarchy is preferred, then this should be reflected in the distribution over hierarchies.

Our second step is to establish a process for generating the unit parameter vectors themselves. For this, we assume that each node of the hierarchy has a parameter vector that is equal in dimension and domain to the unit parameter vectors. Furthermore, we assume that the conditional likelihood of a set of children nodes' parameter vectors is a function of a trace norm of a particular matrix. Let $\vec{\theta}_1, \ldots, \vec{\theta}_n$ be the parameter vectors corresponding to the children of a parent node, which has parameter vector $\vec{\phi}$. Furthermore, define the matrix where each row is the difference between a child's parameter vector and the parent's parameter vector,

$$\Theta = \begin{bmatrix} \vec{\theta}_1 - \vec{\phi} \\ \vdots \\ \vec{\theta}_n - \vec{\phi} \end{bmatrix} \tag{2}$$

Then, we define the distribution of child parameter vectors as proportional to the exponential of the negative trace norm of this matrix,

$$P(\vec{\theta}_1, \ldots, \vec{\theta}_n | \vec{\phi}) \propto \exp\left(-\lambda \|\Theta\|_{\mathrm{tr}}\right). \tag{3}$$

λ controls the flatness of this distribution. Note that the trace norm of a matrix is the sum of its singular values. The trace norm of Θ indicates the degree to which the child parameter vectors can depart from the parent parameter vector. We use the trace norm because it encourages the child parameter vector differences to lie in a subspace—it encourages a low-rank solution. This is in contrast to an exponentiated Frobenius norm or Normal distribution which would encourage a full rank solution.

We have now established a full distribution on the data which is very flexible in the generation process. If there is hierarchical structure in the data, this will be naturally uncovered by this model.

3 Learning

We have established the model, but have yet to discuss how one might do learning. In fact, learning follows naturally from the fact that we have established a distribution over structures and parameters for the model.

3.1 Classification

Consider the task of document classification. We use the document as the fundamental unit and impose a hierarchy restriction that documents in the same class must have a common ancestor (that is different from the common ancestors of the other classes). Our model specifies a joint distribution over model structure, parameters and data. To determine the class of a document for which the class is unobserved, we integrate over all data, structures and parameters where the document is in a particular class (i.e. it is a descendent of the common ancestor for that class). Doing this for all classes, we get $P(\vec{x}, c)$ for each class c. The most likely class is the one with the highest probability. Note that $P(\vec{x}, c) \propto \frac{P(\vec{x},c)}{P(\vec{x})} = P(c|\vec{x})$.

3.2 Segmentation

Consider the task of segmenting transcripted speech by speaker or segmenting a news stream by topic. We can learn a segmentation using this model by maximizing the marginal likelihood of the data while imposing a constraint on the form of the hierarchy. So that we can clearly understand the meaning of a segmentation, we restrict ourselves to 2-level hierarchies. I.e. there is a root node, a level of intermediate nodes and a level of "unit" nodes. The only control is to which intermediate node each unit node associates. We wish to find the hierarchical structure that maximizes the (marginal) likelihood of the data, $P(\vec{x}_1, \ldots, \vec{x}_n)$. To find the best segmentation, we simply calculate the marginal likelihood of the data for each possible segmentation and select the one that yields the highest data likelihood. Again, calculating the marginal likelihood involves integrating out all other variables.

4 Summation

This section is unfinished...

The applications we describe require integrating-out a large number of variables. This can be computationally expensive. The structure of our model allows for a degree of improved efficiency. It corresponds to a factor graph without loops. Hence, the sum-product algorithm can be applied to efficiently calculate marginal probabilities (see [1] for a tutorial).

Each parent-child sub-tree in our model corresponds to a monolithic "factor." I.e. there is nothing that can help us calculate efficiently marginal probabilities within this sub-tree. We must do the integration explicitly. Recall the distribution for each sub-tree,

$$P(\vec{\theta}_1, \ldots, \vec{\theta}_n | \vec{\phi}) \propto \exp\left(-\lambda \|\Theta\|_{\mathrm{tr}}\right). \tag{4}$$

We neglected to include the normalization constant,

$$Z = \int \cdots \int \exp(-\lambda \|\Theta\|_{\mathrm{tr}}) d\vec{\theta}_1 \cdots d\vec{\theta}_n, \tag{5}$$

which is hardly a simple thing to calculate. Note that Z does not depend on $\vec{\phi}$, so we can replace Θ with $\tilde{\Theta} = \begin{bmatrix} \vec{\theta}_1 \\ \vdots \\ \vec{\theta}_n \end{bmatrix}$ in our calculation of Z.

Note that if instead of calculating the trace norm, we were calculating the (exponentiated negative) sum of lengths of the parameter vectors, our distribution would be Gaussian. Our distribution is, in fact, a weighted Gaussian distribution. What we are calculating is the (exponentiated negative) sum of the singular values, or the sum of the lengths of basis vectors (singular vectors times the respective singular value). Why should this be so difficult to calculate, then? Because the the volume over which a summed-length applies is different for the trace norm as it is for the Gaussian. Two overlapping, length-one vectors corresponds to a trace norm of 1, but a summed-length of 2. Adding a new vector adds to the trace norm its length portion which is perpendicular to all other vectors.

References

[1] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519, 2001.