Probabilistic Models of Discrete Data: Notes on Matt Taddy's Work on Inverse Regression

Scribe: Tom Blazejewski

March 16, 2016

This week we looked at two papers, 'Distributed multinomial regression' and 'Mutlinomial inverse regression for text analysis'. Both papers are by Matt Taddy and reader responses suggested that they found the work interesting.

Before we began, Ido asked why the work is even called 'inverse regression' and Prof. Blei (shortened to PDB) stated that in classical regression, we use X to predict Y whereas in 'inverse regression', we use Y to predict X while also learning something about Y.

Let's first set up some definitions we'll find useful for the rest of the lecture.

- $\vec{c}_i$ - word count for document i
- c_{ij} - count of word j in document i
- $\vec{v}_i$ - features of document i
- m_i - total # of features in document i
- d - # of words in vocabulary
- n - # of documents

The model we intend to fit according to Taddy's work (the inverse regression model is):

$$\begin{aligned} c_i | v_i, m_i &\sim Mult(\vec{q}_i, m_i) \\ &\text{(where } q \text{ is a point on simplex, a distribution on words)} \\ q_{ij} \text{ (word } i \text{ in document } j) &\propto \exp\{\eta_{ij}\} \\ &\text{(like a softmax)} \\ \text{and } \eta_{ij} \text{ is a linear model, } \eta_{ij} &= \alpha_j + \vec{\varphi}_j^T \vec{v}_i \end{aligned}$$

This is noted as a GLM for multiple responses. PDB notes that if we feel invested in learning about statistics history, it is worth reading about multi-response GLMs in Breiman and Friedman. Of particular interest is an empirical study part of that paper.

Back to our model, $\vec{\varphi}_j$ is a vector which can encode sentiments. If we are interested in investigating political discourse, there might be a difference in seeing 'healthcare' or 'social' in left/right sides of aisle.

There's also a normalizer,

$$\Lambda_i = \sum_{k=1}^{d} \exp\{\eta_{ik}\}$$

which is different for each document, and that makes sense, as each document will have different contents.

We see that already here, there is a connection to word embeddings where we fix v and $\vec{\varphi}_j$ is the embedding we're looking for. There's also a similar problem, with a log-normalizer that is no fun to calculate and which is here also tied together across corpora.

Here, Kui comments that it doesn't seem exactly like word embedding, and PDB specifies that the connection is pretty close if you consider v as a 'context', especially in the sense that Levy/Goldberg uses, i.e. filling a matrix with counts.

Jaan then says that the context is a document here, not a window, and suggests you could do something like this with a window where each 10 words is considered a separate 'document'. Ben suggests that this is maybe more capturing information about a document, and not quite a context. PDB notes that it is also important they are using other info about document, too.

Taking a cue from something that Adji had noted in her reading response, PDB notes that φ_i would be unknown for word embeddings and that it is interesting that Taddy ultimately allows for a random effect part in his model.

Maja adds at this point that it seems a little unsatisfactory that we are not actually embedding the categorical/binary labels we are using in this model. She suggests that it might be better to use distributed representations vs. simply labeling politicians as Democrats/Republicans.

In any case, this model has been discussed for a bit, but we are actually pivoting toward a change in model from the currently presented model (**model M**) to one based on Poisson distributions (**model P**).

PDB notes that while we'll briefly go over some key facts about Poisson distributions here, if there are any remaining questions, he has traditionally enjoyed the Bayesian Data Analysis 3 ed. textbook as a more comprehensive resource.

Poisson:

$$p(x|\alpha) = \frac{1}{x!}\alpha^x \exp\{-\alpha\} \tag{1}$$

Here, x is a count and α is a positive rate/intensity. We are recommended as an exercise to turn this into its exponential family form.

Then here, Taddy suggests:

$$c_{ij} \sim Poi(\exp\{\eta_{ij}\})$$

This is a new model, but it relates to the original one. It is equivalent to:

- first, drawing m_i from Poisson whose rate is Λ_i... ($m_i \sim Poi(\Lambda_i)$)
- then, draw c_i with m_i... ($c_i | m_i \sim Mult(\vec{q}_i, m_i)$)

This equivalence is true due to two facts that are probably worth knowing.

Fact 1 Let's say we have a collection of Poisson variables ($x_i \sim Poi(r_i)$).

- The sum of x_i's is a Poisson variable ($\sum x_i \sim Poi(\sum r_i)$
  - PDB uses this in his own work, with Paisley, Gopalan to develop auxiliary variables.

Fact 2 If you condition on its sum

$x | \sum_j x_j = m \sim Mult(\vec{\pi}, m)$ with $\pi_i \propto r_i$

John then says that this would be true if x_i were independent, but in our case, we would expect the words to be highly dependent. PDB says we're not there yet and this is just to show that everything we've said so far is true. That we are able to legitimately connect both models.

So two models, then: one is implicitly conditioned on the sum, and the other is a Poisson model where we draw a sum and use the first model.

Now it gets a little fuzzy, but fundamentally the goal is to estimate φ_j's and parameters of models. In an example case, we want to figure out how much Republicans/Democrats are going to say a given word, say 'walnut'.

We can look at gradients w.r.t. η_{ij} and then through chain rule get to original parameters in **model M**.

$$\mathcal{L}_M = \sum_i \vec{c}_i^T \vec{n}_i - m_i \log(\sum_j \exp\{\eta_{ij}\})$$

Here in the first term, we are multiplying an unnormalized log-probability by the number of times we see the word and summing across all documents. This should make sense. In the second term, we are using m_i log-normalizers because each time we see a word, we have a contribution from a normalized and unnormalized probability.

This equation is very difficult to compute. If we take the gradient with the log(sum(exp)), then all gradients depend on other η_{ik}'s. This is a common ML problem.

In his articles, Taddy makes a point here, that if we add a constant to this equation, we do not change the gradient. You can add μ_i to η_i and the log-likelihood does not change with η_{ij}.

Now, taking the same idea about being able to add constant μ_i without changing gradients, let's look at a possible likelihood for the Poisson model (**Model P**). The first term $(c_{ij}(\eta_{ij} + \mu_i))$ comes from the equation I labeled as (1). Note that the η is from $\log(\exp(\eta_{ij}))$ and the factorial drops out.

$$\mathcal{L}_P = \sum_{j=1}^{d} \sum_{i=1}^{n} \vec{c}_i^T \vec{n}_i - m_i \log(\sum_j \exp\{\eta_{ij}\})$$

Note that whereas in the multinomial model (**model M**) we do not model zeroes, here we do. However, we are now sensitive to an intercept term. Maja asks what is the difference between this intercept and α. PDB responds that the intercept here is for the document. Maja then posits that it would potentially be preferable to just include this in the model so that the term could actually be worked out while modeling. PDB says that that is an idea but that the parameterization in its current form is useful and we'll get to details in a bit.

Now if we look at the derivative...

$$\frac{d\mathcal{L}_P}{d\mu_i} = m_i - \sum_j \exp\{\eta_{ij} + \mu_i\}$$

We can rewrite this as, $m_i - \exp\{\mu_i\} \sum_j \exp\{\eta_{ij}\}$.

So our gradient on η_{ij} basically depends on μ_i. So let's say we want to just plug in the μ_i so that it is set to its MLE.

Then $\mu_i^* = \log \frac{mi}{\sum_j \exp\{\eta_{ij}\}}$.

The big point, then, is that when $\mu_i = \mu_i^*$, this likelihood (from **model P**) is equivalent to the one from **model M**.

Ben says there's a sleight of hand about to happen here, because of the use of μ_i as an implicit parameter. We are calling for the use of a sufficient statistic for μ_i and then losing μ_i. So it's a sleight of hand because the models are becoming equivalent through an implicit conditioning on μ_i if we set μ_i to be based on m, which is all we needed from the data to set μ originally. The model equivalence is motivated then abrogated.

Now if we think about this in terms of recommendation systems, the log of the rate of the Poisson is $\alpha_j + \vec{\varphi_j} v_i + \mu_i$. The α_j term would correspond to item intercepts and the μ_i would correspond to user intercepts.

If φ_j is zero everywhere, and α_j is also zero, μ_i would just be indicating how many words to pepper out. It would be like, 'I have words at random to pepper out and if I am small, there will not be many words and if I am big, I will pepper out many words.' The word 'pepper' is used several more times here and ultimately connected to an analogy with actual pepper granules being ground up and falling uniformly into bins (words).

Now depending on the sentiment, φv will affect rates of words, pulling some up, others down. And in general, α is going to be high for words like 'the', 'and', 'pepper', 'or'. Meanwhile, μ_i is just going to indicate how long a document is, with no info about word distribution.

In a case of political sentiment, we might expect 'social' to be brought up by φ when we are talking about left-wing politics and 'taxes' to be brought up when considering right-wing politics.

As reasonable as this all sounds, η_{ij} is still dependent on η_{ik}. This is therefore a calculation that is hard to distribute across many computers (and that is what Matt Taddy wants to do). So what we do is we just set $\hat{\mu_i} = \log m_i$ and then we don't look back. It's wrong, and it's not the MLE, but it allows us to do distributed calculations by crowbarring in some independence. In a minor form of consolation, the estimate is true in some very basic models.

At this point, PDB notes that he is not sure whether it is really the best estimate, and we return to that later. But so anyway, let's say that we set $\eta_{ij} = \alpha_j + \vec{\varphi}_j^T v_i$ and we have $\hat{\mu_i} = \log m_i$.

So the contribution to the log likelihood is:

$$\ell(\alpha_j, \vec{\varphi}_j) = \sum_{i=1}^{n} c_{ij}(\alpha_j + \vec{\varphi}_j \vec{v}_i) - m_i \exp\{\alpha_j + \vec{\varphi}_j^T v_i\}$$

A few notes: we removed $\hat{\mu}_i$ because no dependence. The m_i is coming from $\exp(\hat{\mu}_i)$ where $\hat{\mu}_i = \log m_i$.

And now this is simply a Poisson regression.

$$c_{ij} \sim Poi(\exp\{\alpha_j + \vec{\varphi}_j \vec{v}_i + \log m_i\})$$

We could include $\log m_i$ but we don't have to. And the big result is that we can run this regression across many different computers for each word. Further contributions from Taddy include some regularization and a few other bells and whistles, but we've decoupled these things into separate computations and that's the important part. François notes a similarity to Arora here, that there is a normalizer set to a constant for convenience.

PDB now has a few questions for Taddy... we started with a multinomial, but that was hard to fit because Λ_i ties together normalizers. Then we made Poisson, and said if we set intercept to MLE, we get equivalence. Then out of a RCMP campaign hat we have a plug-in estimator, and that's ocassionally right and we use it out of a general desire to have fun with large datasets.

Question 1. In work on correlated topic models, PDB went back and forth inferring and then setting $\exp\{\eta_{ij}\}$. Can we cycle back and forth here, and set up a coordinate ascent that starts with a plug-in estimate, but updates η_{ij} based on the last iteration? Could this help?

Question 2. Why does it matter? If we go back to pepper shaker, we allow coefficients to be the same thing across documents (*scribe note: my notes don't seem to clearly articulate this question*).

Question 3. If μ_i is supposed to capture document length, then we would expect $\mathrm{E}[X] = r$ (from fact 2). We should expect everything to sum to the document length, but that doesn't actually happen... $\mathrm{E}[m_i | \varphi v + \alpha = 0] = \exp\{\mu_i\} d$. That's not document length. Would $\hat{\mu}_i = \log \frac{m_i}{d}$ be more appropriate/motivated by theory?

Ben at this point suggests we'd never have $\alpha = 0$ and that the parameters should adjust to μ. Maja further points out that model M easily handles proportionality, and we should have no problem with scaling.

Ryan at this point opines that we are sort of losing the whole point of this entire model because there are no guarantees that any of these parameters are going to end up 'true', in that we can actually pull out any sort of inference from this. He wonders what the point of having such a Frankenstein model might be where we have a nice original model and the end result is computable, but not inferentially tied to the thing we set out to find in the first place.

This is noted as a good question and it's not clear whether Taddy himself could answer.

Let's move to inverse regression, because according to PDB, it is a little underutilized.

Here's the idea... we have some exponential r.v.

$$\begin{aligned} p(Z_i) &= \exp\{\eta_i^T c_i - \alpha(\eta_i)\} \\ \eta_{ij} &= \alpha_j + \vec{\varphi}_j^T \vec{v}_i \\ \Phi &= p \text{ x } d \text{ matrix of coefficients (generally, } d > p) \\ p(\vec{c}_i) &= \exp\{\alpha^T c_i\} \exp\{(\Phi \vec{c}_i)^T \vec{v}_i - a(\Phi, \vec{\alpha}_i, \vec{v}_i)\} \end{aligned}$$

In the last line, we see there is a contribution of a function of $\vec{c}_i$ and one of $\Phi\vec{c}_i, v_i$. This implies that $\vec{v}_i$ is independent of $\vec{c}_i$ given $\Phi\vec{c}_i$.

PDB says we are being a little loose here and Ben notes it is not exactly clear when this is actually a sufficient statistic, i.e. is it when we are actually predicting one from the other or...

To put it a just a little bit more precisely...

$$\begin{gathered} v_i \sim ... \\ c_i | v_i \sim ... \\ p(v_i|c_i) ... \text{ and } \mathrm{E}[v_i|c_i] = \mathrm{E}[v_i|\Phi c_i] \end{gathered}$$

Now what Taddy notes is that we can compute for each document and come up with a p-vector where $\vec{Z} = \Phi\vec{c}_i$.

This is intuitive. If we have a set of 'cool' features, we might want to sum across words hitting those cool features and we would end up with some sort of estimate of how cool a document is.

Dawen says that this is sort of a glimpse of a generative vs. discriminative learning coming in to play a role here. PDB agrees, and says that if we care about v, we could just do $v|c$. That would be reasonable and maybe we should do that. The issue is that as v gets high-dimensional, it is harder to do that and at that point, getting to a generative model may actually get us somewhere.

Finally, in the last five minutes, we turned to figuring out how all this relates to word embeddings.

Imagine w_i occurs in a context c_i. We have some potential function, and why not just make it based on a Poisson...

$$\log p(w_i, v = 1 | c_i) = \lambda_v^T \gamma_{ci} - \exp\{\lambda_v^T \gamma_{ci}\}$$

w_i is an indicator vector with one 1. Well then,

$$\log p(w_i | c_i) = \lambda_{wi}^T \gamma_{ci} - \sum_v \exp(\lambda_v^T \gamma_{ci})$$

We might consider fitting this with pseudolikelihood. This looks like regression-based LL. Except we don't ignore zeros, and we use Poisson which is probably nicer/more efficient to fit.

There's a final note that this is a truncated Poisson and apparently you can do a fancy correction of this or not bother and it still works fine.

Organizational note: no class next week (spring break).