Chapter 3

Topic Modeling

3.1 Introduction

Counting words can be a pretty good way of quantifying what a document is about, but not perfect. Suppose that we are trying to categorize documents into different genres. If a document contains the word "laser" or "robot" or other similar words, that is a pretty good sign that it's an engineering article or a science-fiction story. Suppose further that many of these technology-related words just occur one time – say, "photon" occurs only in an engineering article and "ramjet" appears only in a science-fiction story. That wouldn't be very good evidence that documents mentioning "photon" are engineering and documents mentioning "ramjet" are science-fiction. It would be more reasonable for a model to learn that these are all technology-related words, and that technology-related words are indicative of both engineering and science fiction. That is, all these words belong to the *topic* of technology, and both engineering and science fiction use words in the technology topic. How might we learn such topics automatically? It turns out that these topics aren't all that helpful for classification; nevertheless, topics are interesting in their own right.

3.2 Naïve Bayes with topics

Let's modify the NBC as follows:

$$P(k,d) = p(k) \prod_{w \in d} \sum_{t} p(t \mid k) p(w \mid t). \tag{3.1}$$

What's new is the introduction of the variable t (topic), which is drawn from a finite set of predetermined size. We are given training data consisting of documents d_i and classes k_i, but we are not given the topics of the words in the d_i. The topics are called *hidden* variables.

3.2.1 Training

We want to maximize the log-likelihood,

$$\log L = \sum_i \log p(k_i) + \sum_i \sum_{w \in d_i} \log \sum_t p(t \mid k_i) p(w \mid t).$$

The $\sum_i \log p(k_i)$ is off by itself and can be maximized separately. Just as in the NBC, this is done by simply counting and dividing:

$$p(k) = \frac{c(k)}{\sum_k c(k)}. \tag{3.2}$$

The rest of the model is not as simple because of the hidden variables. There are two ways we might proceed. The way that is traditional in NLP is to use *expectation-maximization*, which involves computing a distribution over the hidden variables and then reestimating the model as if the hidden variables were actually observed.

The other way is to directly maximize the log-likelihood using stochastic gradient ascent. To do that, we need to do a few things. First, note that we're going to be maximizing with respect to $p(t \mid k)$ and $p(w \mid t)$, which are probabilities, so they must be non-negative and normalized. We can accomplish this with a change of variables:

$$p(t \mid k) = \operatorname*{softmax}_{t'=t} \lambda(t' \mid k) = \frac{\exp \lambda(t \mid k)}{\sum_{t'} \exp \lambda(t' \mid k)}$$
$$p(w \mid t) = \operatorname*{softmax}_{w'=w} \lambda(w' \mid t) = \frac{\exp \lambda(w \mid t)}{\sum_{w'} \exp \lambda(w' \mid t)}$$

Now we can optimize with respect to the λ's, which are unconstrained.

Automatic differentiation can figure out the gradient of the log-likelihood for you, but it may be illuminating to see what the answer is. For a single document d with correct class k, the gradient with respect to $p(t \mid k)$ is:

$$\begin{aligned}
\frac{\partial}{\partial p(t \mid k)} \log L &= \sum_{w \in d} \frac{\sum_{t'} \frac{\partial p(t' \mid k)}{\partial \lambda(t \mid k)} p(w \mid t')}{\sum_{t'} p(t' \mid k) p(w \mid t')} \\
&= \sum_{w \in d} \frac{\sum_{t'} p(t' \mid k)(\delta(t, t') - p(t \mid k)) p(w \mid t')}{\sum_{t'} p(t' \mid k) p(w \mid t')} \\
&= \sum_{w \in d} \left(\frac{p(t \mid k) p(w \mid t)}{\sum_{t'} p(t' \mid k) p(w \mid t')} - p(t \mid k) \right) \\
&= \sum_{w \in d} \big(P(t \mid w, k) - p(t \mid k) \big).
\end{aligned}$$

The gradient with respect to $p(t \mid k')$ for $k' \neq k$ is zero. Similarly, the gradient with respect to $p(w \mid t)$ is:

$$\frac{\partial}{\partial p(w \mid t)} \log L = \sum_{w \in d} \big(P(w \mid t, k) - p(w \mid t) \big).$$

Intuitively, the negative terms cause the model to "forget" a little bit of what it thought before, and the positive terms push the model a little bit towards the observed document d and class k. For example, suppose $d = \{w\} = \{\text{write}\}$ and $k = \text{female}$. If we previously thought that topic 12 favored "write," but didn't know that females favor topic 12, then we'll increase $p(12 \mid \text{female})$. On the other hand, if we previously thought that females favor topic 12, but didn't know that topic 12 favored "write," then we'll increase $p(\text{write} \mid 12)$.

3.2.2 Experiment

We trained this model on the blog dataset using expectation-maximization. (This experiment dates from an earlier version of the notes which used only expectation-maximization, not stochastic gradient as-

cent.) We used 20 topics, 100 iterations, and add-10^{-6} smoothing (add-1 didn't work at all), and got an accuracy of 61.6%, which is no improvement over the original model (65.1%).

But what is more interesting is the topics learned by the model. We show the top seven words in each topic, sorted according to a magic formula:[1]

t	$P(t \mid \text{m})$	$P(t \mid \text{f})$	words
0	0.04	0.05	`write ever together friends days recently took`
1	0.03	0.06	`sweet sugar wonderful loved dog n child`
2	0.04	0.06	`weekend wait favorite can't women pretty took`
3	0.01	0.09	`amelia prints lunch lovely en wedding raw`
4	0.10	0.00	`gay java users vacuum copyright eu lardo`
5	0.06	0.03	`war companies version engine box form number`
6	0.00	0.14	`sudan darfur vous slaw etsy dans deedee`
7	0.00	0.12	`une darfur garlic deedee etsy harriet sudan`
8	0.05	0.04	`seems ice power paper front thanks later`
9	0.12	0.00	`topps não linecook avrdude drillard liverpool ironpython`
10	0.05	0.04	`topics ways thats they've computer they're slowly`
11	0.06	0.03	`clear trend low question added themselves easily`
12	0.05	0.04	`quickly cut 2008 enough level care story`
13	0.05	0.04	`play search each sounds which goal include`
14	0.13	0.00	`drillard python ironpython topps wifi google's balloons`
15	0.05	0.04	`statement below less technical starting wrong these`
16	0.03	0.07	`herself sun dog coffee kids images tea`
17	0.04	0.05	`exhausted home until girl me positive house`
18	0.07	0.02	`stage film reference states system telephone google`
19	0.00	0.10	`le sauce du ladies à lunch et`

Some of the topics are fairly coherent, and you can see that some topics are heavily favored by male bloggers and some topics are favored by female bloggers. In the next section, we'll leave text classification behind and focus our attention on discovering topics.

3.3 Probabilistic LSA

We can think of a document represented as a bag of words as a vector, where each component of the vector is a different word type. In many applications, this vector representation is useful in its own right. For example, we can compute the similarity between two documents using the cosine of the angle between the two document vectors.

As argued above, however, words might be too fine-grained a way of quantifying what a document is about. So what if we use vectors where each component of the vector is a different *topic*? This is the idea behind *probabilistic latent semantic analysis* (PLSA) or *probabilistic latent semantic indexing* (PLSI) (Hofmann, 1999). The only difference between PLSA and the NBC with topics is that in PLSA, every document is in its own class. Then, the probabilities $p(t \mid k_i)$ can be thought of as a vector representation of document d_i.

Training PLSA is identical to above. But after training a PLSA model, there isn't any inference step. The end product of PLSA training is the parameters themselves. We ran PLSA (expectation-maximization, 20

[1] Sorted according to $p(w \mid t)/(c(w)+10)$. There isn't any deep motivation for this formula, but it seems to work okay.

topics, 100 iterations, add-10^{-6} smoothing). We used the same half-baked formula as before to find the most interesting words in each topic, and they were:

topic	words
0	`deedee kohl eric fatty adhd earthquake spud peppa beetner jeans`
1	`woke tomorrow mom bed snow birthday ride went sleep randa`
2	`garden seed gardening gardeners lambing burpee seeds sheep jacob katahdin`
3	`slaw não é uma para reservation um o nsobject ser`
4	`mangalitsa quiz nutrition lardo kotak skin vegan wax band's organizational`
5	`  balloons baking rice butter flour sugar psi tiramisu cream`
6	`drillard kombu reform republicans democrats congress sen obama directx mona`
7	`  les linecook le une dans à vous et pas`
8	`divorce weight charter overweight bimal haw exercises webkit dating muscles`
9	`darfur sudan sudanese rebels khartoum jungle al-nur rebel marra barrel`
10	`printmaking prints ds yourself stable jailbird teaching sasha math forgive`
11	`liverpool dreamer conversations benitez gage arc reds horror arsenal liverpool's`
12	`harriet { } geographers amelia font-size plone zope jsf hvar`
13	`# avrdude avr reddit pololu vikki orangutan hex kindle avr-gcc`
14	`arruda habenula ponzi brasilia heap dsn tile gems hop dominion`
15	`ironpython mobile microsoft lumber user marketing developers web users applications`
16	`baseball topps tournament mcgriff sticker fred knitting sox stickers dodgers`
17	`sous-vide choir low-temperature choral low-temp temperature mp3 albums album pokey`
18	`n ur ~ abt ll tat wat coz jus gal`
19	`israel hudson hezbollah borscht iran enrichment git github rake integrity`

Some topics pick up foreign words, and some topics are fairly specific subjects (2: gardening; 5: baking; 6: American politics; 9: Sudan; 11: soccer; 15: programming; 16: baseball; 17: cooking and music; 19: Middle East politics and programming). Maybe you can see some others.

3.4 Optional: Latent Dirichlet Allocation

No one uses PLSA anymore. *Latent Dirichlet Allocation* (Blei, Ng, and Jordan, 2003) is a further development of PLSA that is by far the most widely known and used topic model. LDA is quite math-heavy and we don't give a full description here, but just a little bit of the basic idea.

In Section 2.3.2, we introduced the notion of a *prior* distribution:

$$\underbrace{P(\theta \mid \text{data})}_{\text{posterior}} = \frac{\overbrace{P(\theta)}^{\text{prior}}\overbrace{P(\text{data} \mid \theta)}^{\text{likelihood}}}{\underbrace{P(\text{data})}_{\text{evidence}}}, \tag{3.3}$$

and we just assumed that the prior was uniform. But we can choose other distributions too. The easiest choice of prior, other than uniform, is the (symmetric) *Dirichlet distribution*:

$$\text{Dir}(\theta; \alpha) \propto \prod_i \theta_i^{\alpha-1}. \tag{3.4}$$

The parameter α is called the *concentration*.

Now we have a number of options. First, we can try to find the best model (θ) just like before. Since we are now taking the prior into account, we're not doing maximum likelihood estimation anymore; we're doing *maximum a posteriori (MAP) estimation*. It's not too hard to see that the resulting estimate is exactly what the MLE estimate would have been if there were pseudocounts of $(\alpha - 1)$ for each outcome. In other words, MAP estimation with a Dirichlet prior with concentration α is equivalent to MLE estimation with add-$(\alpha - 1)$ smoothing.

But another way of thinking about it is, if we're only interested in the topics, and not θ, then we shouldn't try to estimate θ. Instead, we should integrate over all possible values of θ:

$$P(\text{topics} \mid \text{data}) = \frac{P(\text{topics}, \text{data})}{P(\text{data})} \tag{3.5}$$

$$= \frac{\int P(\text{topics}, \text{data}, \theta)\, \mathrm{d}\theta}{\int P(\text{data}, \theta) \mathrm{d}\theta} \tag{3.6}$$

That looks bad, and in all but the simplest cases, it is bad, but that doesn't stop people from trying to do it anyway.

Asuncion et al. (2009) provide a good overview and comparison of both PLSA and LDA. The LDA model looks like this:

$$P(\text{data} \mid \alpha, \beta) = \underbrace{\prod_t P(\phi_t \mid \beta) \prod_i P(\theta_i \mid \alpha)}_{\text{prior}} \underbrace{\prod_{w \in d_i} \sum_t p(t \mid i, \theta)\, p(w \mid t, \phi)}_{\text{likelihood}}. \tag{3.7}$$

The part labeled "likelihood" is more or less the same as PLSA. The part labeled "prior" has two parts; $P(\phi_t \mid \beta)$ is a Dirichlet prior on the word distributions $p(w \mid t)$, and $P(\theta_i \mid \alpha)$ is a Dirichlet prior on the topic distributions $p(t \mid i)$.

From here, we can use one of several different approximation methods to proceed. One of these (Variational Bayes or mean-field approximation) is operationally quite similar to EM. For further information, see the references listed below.

References

Asuncion, Arthur et al. (2009). "On Smoothing and Inference for Topic Models". In: *Proc. UAI*, pp. 27–34.

Blei, David M., Andrew Y. Ng, and Michael I. Jordan (2003). "Latent Dirichlet Allocation". In: *J. Machine Learning Research* 3, pp. 993–1022.

Hofmann, Thomas (1999). "Probabilistic Latent Semantic Analysis". In: *Uncertainty in Artificial Intelligence (UAI)*, pp. 289–296.