CMSC 25025 / STAT 37601

Machine Learning and Large Scale Data Analysis

Assignment 5

Out: Tuesday, May 9, 2017
Due: Thursday, May 18, 2017

1. *Conditional probabilities for topic modeling* (30 points)

   The following subproblems have to do with the probability model underlying topic models.

   (a) Let $z_{1:N}$ denote the topic indicator variables for document d in a K-topic latent Dirichlet allocation (LDA) model. The topics are denoted β_k, for $k = 1, \ldots, K$; each of these is a multinomial over a V-word vocabulary.

   Derive the conditional probability distribution

   $$\mathbb{P}(z_n \,|\, z_{-n}, \beta_{1:K}, w_{1:N}, \alpha)$$

   where the topic mixture proportions $\theta_d \sim \text{Dirichlet}(\alpha)$ are integrated out. Give a detailed derivation and explanation of each step. Include all normalizing constants.

   (b) Explain how the distribution above can be used to approximate $\mathbb{E}(\theta_d \,|\, \beta_{1:K}, w_{1:N}, \alpha)$ with a Gibbs sampling algorithm.

   (c) Assuming the same latent Dirichlet allocation model as above, derive a closed form expression for the integral

   $$\mathbb{P}(w_{1:N}, z_{1:N} \,|\, \beta_{1:K}, \alpha) = \int \mathbb{P}(w_{1:N}, z_{1:N} \,|\, \theta_d, \beta_{1:K})\, p(\theta_d \,|\, \alpha)\, d\theta_d.$$

   (d) Let $\boldsymbol{z}$ denote the topic indicator variables for entire collection of words $\boldsymbol{w}$ across all documents in the collection. Derive the conditional probability distribution

   $$\mathbb{P}(z_n \,|\, \boldsymbol{z}_{-n}, \boldsymbol{w}, \alpha, \eta)$$

   of the topic z_n for a specific word w_n in a document, where the topics $\beta_{1:K}$ are integrated out with respect to a prior Dirichlet(η). Give a detailed derivation and explanation of each step. Include all normalizing constants.

Notes and clues

Here are some notes that may help with these problems. The Dirichlet distribution of dimension m is defined by

$$p(x \,|\, \alpha) \propto \prod_{j=1}^{m} x_j^{\alpha_j - 1}. \tag{1}$$

The constant of proportionality, which makes this integrate to one, is

$$\frac{\Gamma(\sum_j \alpha_j)}{\prod_{j=1}^m \Gamma(\alpha_j)}. \tag{2}$$

In other words (other symbols?), we have that

$$\int_{\Delta_m} \prod_{j=1}^m x_j^{\alpha_j - 1}\, dx = \int_0^1 \int_0^{1-x_1} \cdots \int_0^{1-x_1-\cdots x_{m-1}} \prod_{j=1}^m x_j^{\alpha_j-1}\, dx_1 dx_2 \cdots dx_m = \frac{\prod_{j=1}^m \Gamma(\alpha_j)}{\Gamma(\sum_j \alpha_j)}. \tag{3}$$

where Δ_m is the probability simplex. Note that if all of the α_js are equal, the density is

$$p(x \,|\, \alpha) = \frac{\Gamma(m\alpha)}{\Gamma(\alpha)^m} \prod_{j=1}^m x_j^{\alpha-1}. \tag{4}$$

If $X \sim \text{Dirichlet}(\alpha)$ with $\alpha = (\alpha_1, \ldots, \alpha_m)$ then $\mathbb{E}[X_j] = \alpha_j / \sum_k \alpha_k$. This follows easily from the above, since

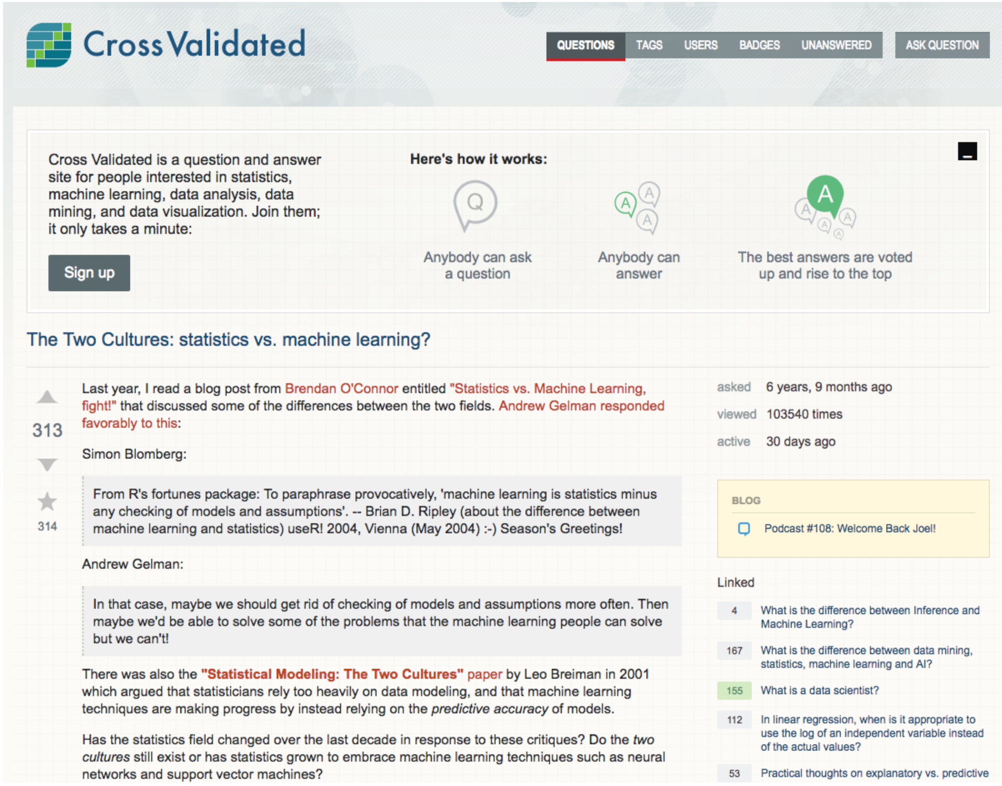
$$\begin{aligned}
\mathbb{E}(X_j \,|\, \alpha) &= \frac{\Gamma(\sum_k \alpha_k)}{\prod_k \Gamma(\alpha_k)} \int_{\Delta_m} x_j \prod_{k=1}^m x_k^{\alpha_k - 1}\, dx && (5)\\
&= \frac{\Gamma(\sum_k \alpha_k)}{\prod_k \Gamma(\alpha_k)} \frac{\Gamma(\alpha_j + 1) \prod_{k \neq j} \Gamma(\alpha_k)}{\Gamma(\sum_k \alpha_k + 1)} && (6)\\
&= \frac{\Gamma(\sum_k \alpha_k)}{\Gamma(\sum_k \alpha_k + 1)} \frac{\Gamma(\alpha_{j+1})}{\Gamma(\alpha_j)} && (7)\\
&= \frac{\alpha_j}{\sum_k \alpha_k} && (8)
\end{aligned}$$

using the fact that $\Gamma(z+1) = z\Gamma(z)$.

These calculations can be summarized as follows. Suppose that we have a weighted V-sided die, with probability θ_j that face j comes up. Our prior distribution on θ is Dirichlet(α) with $\alpha = (\alpha, \ldots, \alpha)$. We roll the die N times and observe face j a total of n_j times, so $\sum_j n_j = N$. Our posterior distribution over θ is then Dirichlet($(n_1 + \alpha, \ldots, n_V + \alpha)$). If we now roll the die one more time, we believe the probability that j comes up is

$$p_j = \frac{n_j + \alpha}{N + \alpha V}. \tag{9}$$

In the following problems, you will model the statistics and machine learning repository of the online question and answer site "StackExchange," called "CrossValidated." Screen shots from a couple of the posts are shown below.



Cross Validated

QUESTIONS TAGS USERS BADGES UNANSWERED ASK QUESTION

Cross Validated is a question and answer site for people interested in statistics, machine learning, data analysis, data mining, and data visualization. Join them; it only takes a minute:

Sign up

Here's how it works:

Anybody can ask a question

Anybody can answer

The best answers are voted up and rise to the top

The Two Cultures: statistics vs. machine learning?

313

314

Last year, I read a blog post from Brendan O'Connor entitled "Statistics vs. Machine Learning, fight!" that discussed some of the differences between the two fields. Andrew Gelman responded favorably to this:

Simon Blomberg:

> From R's fortunes package: To paraphrase provocatively, 'machine learning is statistics minus any checking of models and assumptions'. -- Brian D. Ripley (about the difference between machine learning and statistics) useR! 2004, Vienna (May 2004) :-) Season's Greetings!

Andrew Gelman:

> In that case, maybe we should get rid of checking of models and assumptions more often. Then maybe we'd be able to solve some of the problems that the machine learning people can solve but we can't!

There was also the **"Statistical Modeling: The Two Cultures"** paper by Leo Breiman in 2001 which argued that statisticians rely too heavily on data modeling, and that machine learning techniques are making progress by instead relying on the *predictive accuracy* of models.

Has the statistics field changed over the last decade in response to these critiques? Do the *two cultures* still exist or has statistics grown to embrace machine learning techniques such as neural networks and support vector machines?

asked 6 years, 9 months ago

viewed 103540 times

active 30 days ago

BLOG

Podcast #108: Welcome Back Joel!

Linked

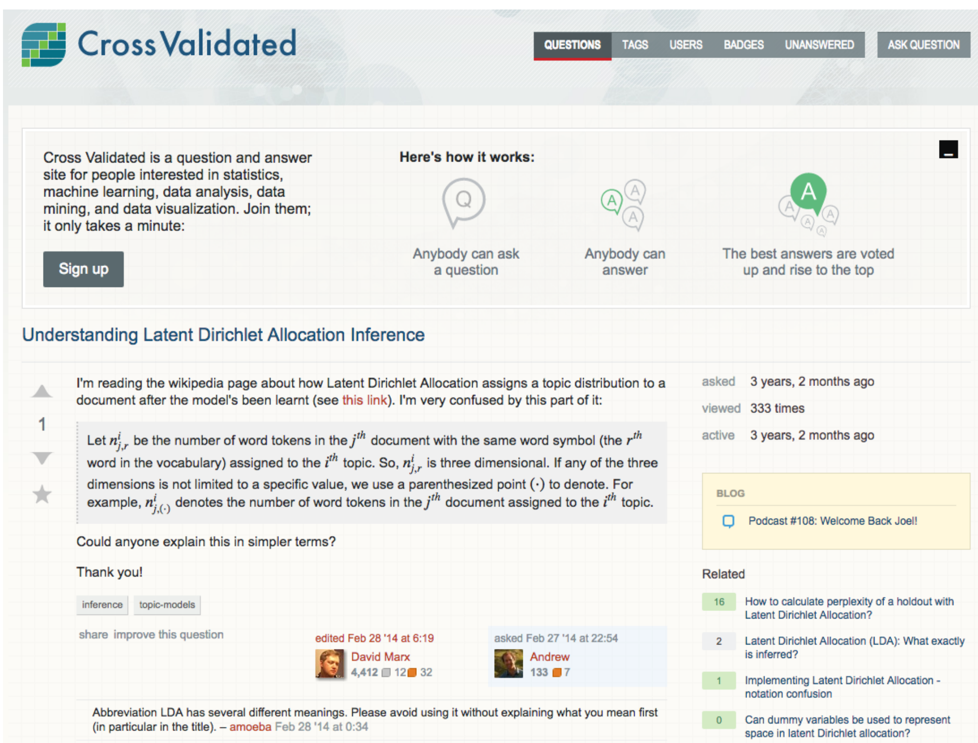
4 What is the difference between Inference and Machine Learning?

167 What is the difference between data mining, statistics, machine learning and AI?

155 What is a data scientist?

112 In linear regression, when is it appropriate to use the log of an independent variable instead of the actual values?

53 Practical thoughts on explanatory vs. predictive



Cross Validated

QUESTIONS TAGS USERS BADGES UNANSWERED ASK QUESTION

Cross Validated is a question and answer site for people interested in statistics, machine learning, data analysis, data mining, and data visualization. Join them; it only takes a minute:

Sign up

Here's how it works:

Anybody can ask a question

Anybody can answer

The best answers are voted up and rise to the top

Understanding Latent Dirichlet Allocation Inference

1

I'm reading the wikipedia page about how Latent Dirichlet Allocation assigns a topic distribution to a document after the model's been learnt (see this link). I'm very confused by this part of it:

> Let $n^i_{j,r}$ be the number of word tokens in the j^{th} document with the same word symbol (the r^{th} word in the vocabulary) assigned to the i^{th} topic. So, $n^i_{j,r}$ is three dimensional. If any of the three dimensions is not limited to a specific value, we use a parenthesized point $(\cdot)$ to denote. For example, $n^i_{j,(\cdot)}$ denotes the number of word tokens in the j^{th} document assigned to the i^{th} topic.

Could anyone explain this in simpler terms?

Thank you!

inference topic-models

share improve this question

edited Feb 28 '14 at 6:19
David Marx
4,412 12 32

asked Feb 27 '14 at 22:54
Andrew
133 7

Abbreviation LDA has several different meanings. Please avoid using it without explaining what you mean first (in particular in the title). – amoeba Feb 28 '14 at 0:34

asked 3 years, 2 months ago

viewed 333 times

active 3 years, 2 months ago

BLOG

Podcast #108: Welcome Back Joel!

Related

16 How to calculate perplexity of a holdout with Latent Dirichlet Allocation?

2 Latent Dirichlet Allocation (LDA): What exactly is inferred?

1 Implementing Latent Dirichlet Allocation - notation confusion

0 Can dummy variables be used to represent space in latent Dirichlet allocation?

LSDA

2. *Topic modeling of CrossValidated* (40 points)

Our data were taken from the December 15, 2016 Stack Exchange data dump[1], and processed by Paul Hively. You will find two files,

```
/project/cmsc25025/stackexchange/20161215StatsPostsRaw.csv
/project/cmsc25025/stackexchange/20161215StatsPostsMerged.csv
```

The cleaned file has 92,335 documents, created by combining questions and associated answers, then removing HTML, LaTeX, code, and stopwords. See the `README` file for further details.

Here is part of an entry from the cleaned up version of the collection:

```
124,"Statistical classification of text I'm a programmer without
statistical background, and I'm currently looking at different
classification methods for a large number of different documents that
I want to classify into pre-defined categories. I've been reading
about kNN, SVM and NN. However, I have some trouble getting
started. What resources do you recommend? I do know single variable
and multi variable calculus quite well, so my math should be strong
enough. I also own Bishop's book on Neural Networks, but it has proven
to be a bit dense as an introduction. [...]
```

(a) Process the data to determine a word vocabulary. You should get a vocabulary of size around 10,000 words or so—it's up to you to decide. Describe the steps you take to process the data and the criteria you use to select the vocabulary.

(b) Now fit topic models on the collection. Divide the corpus into training and validation documents–use a 90%/10% split, holding out about 9,000 documents. You will need to write a parser that maps each entry to a sequence of word-id/count pairs.

You may use the LDA implementation from either spark.mllib or spark.ml. The following resources may be helpful:

`http://goo.gl/Y308G4` (Topic modeling with LDA: MLlib meets GraphX)
`http://goo.gl/lIacvm` (spark.mllib.clustering.LDA)
`http://goo.gl/Ro2Fnt` (spark.ml.clustering.LDA)

Train topic models using different numbers of topics; a good starting point would be around 30 topics. Display the top 10 or so words (in decreasing order of probability β_{kw}) in each topic. Comment on the "meaning" or interpretation of several of the topics.

Select several documents, and display the most probable topics for each of them (according to the posterior distribution over θ). Do the assigned topics make sense? Comment on your findings.

[1]Licensed under Creative Commons Share Alike 3.0, https://creativecommons.org/licenses/by-sa/3.0/

You will need to read the documentation for the implementation that you choose (mllib or ml), to learn how to carry out these steps.

(c) Now you will investigate how to evaluate the model more quantitatively[2]. Recall that (for a model that is exchangeable at the document level), the perplexity of a model θ is

$$\text{Perpexity}(\theta) = \left(\prod_D p_\theta(D) \right)^{-1/\sum_D |D|}$$

where D is a test document with $|D|$ words. Explain how this corresponds to the definition

$$\text{Perpexity}(\theta) = \left(\prod_{n=1}^{N} p_\theta(w_n \,|\, w_1, \ldots, w_{n-1}) \right)^{-1/N}$$

which is the inverse geometric mean of the predictions.

Now, explain (mathematically) how to evaluate the test set perplexity for the latent Dirichlet allocation model. Why is this difficult? Can you propose a computationally efficient approximation? For clues you may wish to study the spark.ml code that implements perplexity.

(d) Now evaluate the test set perplexity for a range of models, fit with $K = 10, 20, \ldots, 200$ topics (or an appropriate range of your own choice). Plot the test set perplexity as a function of number of topics. Which is the best model? Do you notice any qualitative difference in the topics as K increases? Comment on your overall findings.

LSDA

3. *Fake news* (30 points)

In this problem, you will train Recurrent Neural Networks (RNN) language models to generate fake news. You will give the RNN some text and ask it to model the probability distribution of the next element in the sequence given a sequence of previous elements.

We have already prepared some code `fake_new.py` for you to start. It trains a two layer Long Short Term Memory machine (LSTM) and then generates fake news. Load `python`, then use `python fake_news.py -h` to see the input arguments. You can find their default values in the code. For example, the default mini-batch size is 2048.

```
parser.add_argument('--batch_size', '-b', type=int, default=2048,
                    help='Number of examples in each mini-batch')
```

The input/output type could be either characters or words. Character-level RNNs are faster to train as the vocabulary size is much smaller, but word-level RNNs give better performance.

[2] This is a "theory" problem. Write up your solution in a markdown cell in your IPython notebook with your code.

We use a deep learning package called `Chainer`. Our LSTM is implemented in class `FooRNN`. The zero-th layer is word embedding, which converts the id of your element into a vector. The first and second layers contain standard LSTM units. The last layer does a linear transformation. The class function `__call__` defines how this LSTM is executed.

```
def __call__(self, x):
    h0 = self.embed(x)
    h1 = self.l1(F.dropout(h0, train=self.train))
    h2 = self.l2(F.dropout(h1, train=self.train))
    y = self.l3(F.dropout(h2, train=self.train))
    return y
```

For an input element x, it outputs a vector $y \in \mathbb{R}^{|V|}$, where V is the vocabulary. As we specified later that our task is classification,

```
model = L.Classifier(FooRNN(n_vocab, args.n_units, train=True))
```

`Chainer` will pass y into a softmax function to obtain a distribution $\bar{y}$:

$$\bar{y}_i = \frac{\exp(y_i)}{\sum_j \exp(y_j)}.$$

The local loss of this unit is the cross entropy between $\bar{y}$ and the true distribution of next element, which is represented as a one-hot vector. This part is automated by `Chainer` and you won't see it in our code. To prevent overfitting, we randomly drop out half of the units in our RNN during training, while in prediction all units are used.

The training text we use is a subset of the UCI news aggregator dataset. It contains four types of news: business, technology, entertainment and health, stored in

```
/project/cmsc25025/uci-news-aggregator/{b,t,e,m}_article.json
```

where one line contains a new story.

After training over the concatenation of 6 million words, with 128 hidden units in each LSTM layer and 100 epochs, we obtained the following generated text:

the ice-company cautioned in earnings of risk claims during the city price of commercial therapy processing in operating price rate prices in florida and the euro fell in march to 29.4 per cent in 2012. it was 82 cents, beating $10.5 per share, the most low estimate of half to 13 days to 0.01%. even analysts recently helped no rates can likely constitute liquidity subsidies. in rising modules, alibaba was unlikely to consolidate their profit for the country to prohibit its organic apparel cushion the date of both counters expanded in business produced counter on the internet business and focus on interim capital, spokesman andrew to meet technology data's apple person that owns an outperform lafarge.

The generated text is not very grammatical or fluent. Your goal will be to modify the architecture or training of the LSTM in order to improve performance, as measured subjectively by the quality of the generated text.

You can run `fake_news.py` on either a CPU or GPU. To use a CPU, set the input argument `-g -1`. GPU nodes are limited on Midway clusters, so we will use `SBATCH` scripts to submit the job. Details will be posted to Piazza.

Your task is to understand the code and modify it. Calculate the perplexity on the validation data, and tune the parameters such as the number of units and the embedding dimension in order to minimize the perplexity. Generate 10 fake news stories of reasonable length, and write them to a file named `generated_text.txt`.

Feel free to change other parts of the code or architecture if you like, such as the structure of the RNN. You can also use your own code framework or other neural network packages such as `theano` or `pytorch`. (`TensorFlow` is currently unsupported.) If you would like to use other packages, please post them on Piazza.

Submit your modified `fake_news.py` and `generated_text.txt`.