Exp-normalize trick

Feb 11, 2014

This trick is the very close cousin of the infamous log-sum-exp trick (scipy.misc.logsumexp (http://docs.scipy.org/doc/scipy/reference/generated/scipy.misc.logsumexp.html)),
Supposed you'd like to evaluate a probability distribution $\boldsymbol{\pi}$ parametrized by a vector $\boldsymbol{x} \in \mathbb{R}^n$ as follows:

$$\pi_i = \frac{\exp(x_i)}{\sum_{j=1}^n \exp(x_j)}$$

The exp-normalize trick leverages the following identity to avoid numerical overflow. For any $b \in \mathbb{R}$,

$$\pi_i = \frac{\exp(x_i - b)\exp(b)}{\sum_{j=1}^n \exp(x_j - b)\exp(b)} = \frac{\exp(x_i - b)}{\sum_{j=1}^n \exp(x_j - b)}$$

In other words, the $\boldsymbol{\pi}$ is shift-invariant. A reasonable choice is $b = \max_{i=1}^n x_i$. With this choice, overflow due to $\exp$ is impossible—the largest number exponentiated after shifting is 0.

Exp-normalize v. log-sum-exp

If what you want to remain in log-space, that is, compute $\log(\boldsymbol{\pi})$, you should use logsumexp. However, if $\boldsymbol{\pi}$ is your goal, then exp-normalize trick is for you! Since it avoids additional calls to $\exp$, which would be required if using log-sum-exp and more importantly exp-normalize is more numerically stable!

Log-sum-exp for computing the log-distibution

$$\log \pi_i = x_i - \mathrm{logsumexp}(\boldsymbol{x})$$

where

$$\mathrm{logsumexp}(\boldsymbol{x}) = b + \log \sum_{j=1}^n \exp(x_j - b)$$

Typically with the same choice for b as above.

Numerically-stable sigmoid function

The sigmoid function can be computed with the exp-normalize trick in order to avoid numerical overflow. In the case of $\mathrm{sigmoid}(x)$, we have a distribution with unnormalized log probabilities $[x, 0]$, where we are only interested in the probability of the first event. From the exp-normalize identity, we know that the distributions $[x, 0]$ and $[0, -x]$ are equivalent (to see why, plug in $b = \max(0, x)$). This is why sigmoid is often expressed in one of two equivalent ways:

$$\mathrm{sigmoid}(x) = 1/(1 + \exp(-x)) = \exp(x)/(\exp(x) + 1)$$

Interestingly, each version covers an extreme case: $x = \infty$ and $x = -\infty$, respectively. Below is some python code which implements the trick:

```
def sigmoid(x):
    "Numerically-stable sigmoid function."
    if x >= 0:
        z = exp(-x)
        return 1 / (1 + z)
    else:
        # if x is less than zero then z will be small, denom can't be
        # zero because it's 1+z.
        z = exp(x)
        return z / (1 + z)
```

Posted by Tim Vieira Feb 11, 2014 numerical (http://timvieira.github.io/blog/tag/numerical.html)

Tweet

Comments

0 Comments **Graduate Descent** 1 **Login**

Recommend 1 Share Sort by Best

Start the discussion…

Be the first to comment.

Recent Posts

- Counterfactual reasoning and learning from logged data (http://timvieira.github.io/blog/post/2016/12/19/counterfactual-reasoning-and-learning-from-logged-data/)
- Heaps for incremental computation (http://timvieira.github.io/blog/post/2016/11/21/heaps-for-incremental-computation/)
- Reversing a sequence with sublinear space (http://timvieira.github.io/blog/post/2016/10/01/reversing-a-sequence-with-sublinear-space/)
- Evaluating ∇f(x) is as fast as f(x) (http://timvieira.github.io/blog/post/2016/09/25/evaluating-fx-is-as-fast-as-fx/)
- Fast sigmoid sampling (http://timvieira.github.io/blog/post/2016/07/04/fast-sigmoid-sampling/)

Tags

datastructures (http://timvieira.github.io/blog/tag/datastructures.html), calculus (http://timvieira.github.io/blog/tag/calculus.html), statistics (http://timvieira.github.io/blog/tag/statistics.html), importance-sampling (http://timvieira.github.io/blog/tag/importance-sampling.html), structured-prediction (http://timvieira.github.io/blog/tag/structured-prediction.html), crf (http://timvieira.github.io/blog/tag/crf.html), decision-making (http://timvieira.github.io/blog/tag/decision-making.html), misc (http://timvieira.github.io/blog/tag/misc.html), sampling (http://timvieira.github.io/blog/tag/sampling.html), deep-learning (http://timvieira.github.io/blog/tag/deep-learning.html), rant (http://timvieira.github.io/blog/tag/rant.html), optimization (http://timvieira.github.io/blog/tag/optimization.html), algorithms (http://timvieira.github.io/blog/tag/algorithms.html), visualization (http://timvieira.github.io/blog/tag/visualization.html), machine-learning (http://timvieira.github.io/blog/tag/machine-learning.html), Gumbel (http://timvieira.github.io/blog/tag/gumbel.html), autodiff (http://timvieira.github.io/blog/tag/autodiff.html), numerical (http://timvieira.github.io/blog/tag/numerical.html), counterfactual-reasoning (http://timvieira.github.io/blog/tag/counterfactual-reasoning.html)

GitHub Repos

- timvieira.github.com (https://github.com/timvieira/timvieira.github.com)
  Personal webpage.
- arsenal (https://github.com/timvieira/arsenal)
  Arsenal of python utilities.
- skid (https://github.com/timvieira/skid)
  bookmarks, simply kept in directories.
- learning-to-prune (https://github.com/timvieira/learning-to-prune)
  Learning to Prune: Exploring the Frontier of Fast and Accurate Parsing

@timvieira (https://github.com/timvieira) on GitHub

Latest Tweets

- Status updating...

Follow @xtimv 616 followers