Graduate Descent

- About
- Archive

KL-divergence as an objective function

Oct 06, 2014 by Tim Vieira statistics machine-learning structured-prediction

It's well-known that KL-divergence is not symmetric, but which direction is right for fitting your model?

Which KL is which? A cheat sheet

If we're fitting q_θ to p using

$\mathbf{KL}(p||q_\theta)$

- mean-seeking, *inclusive* (more principled because approximates the *full* distribution)
- requires normalization wrt p (i.e., often *not* computationally convenient)

$\mathbf{KL}(q_\theta||p)$

- mode-seeking, *exclusive*
- no normalization wrt p (i.e., computationally convenient)

Mnemonic: "When the truth comes first, you get the whole truth" (h/t Philip Resnik). Here "whole truth" corresponds to the *inclusiveness* of $\mathbf{KL}(p||q)$.

As far as remembering the equation, I pretend that "||" is a division symbol, which happens to correspond nicely to a division symbol in the equation (I'm not sure it's intentional).

Inclusive vs. exclusive divergence

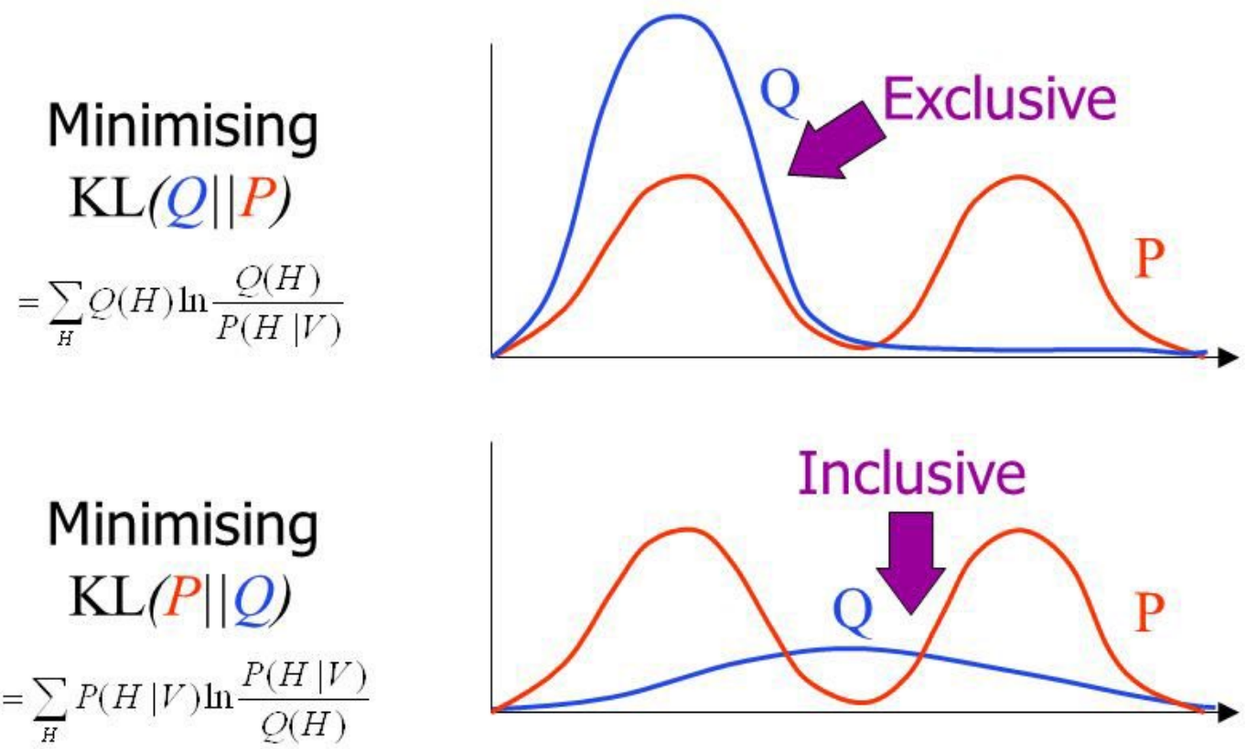

Figure by John Winn.

Computational perspecive

Let's look at what's involved in fitting a model q_θ in each direction. In this section, I'll describe the gradient and pay special attention to the issue of normalization.

Notation: p, q_θ are probabilty distributions. $p = \bar{p}/Z_p$, where Z_p is the normalization constant. Similarly for q.

The easy direction $\mathbf{KL}(q_\theta||p)$

$$
\begin{aligned}
\mathbf{KL}(q_\theta || p) &= \sum_d q(d) \log\left(\frac{q(d)}{p(d)}\right) \\
&= \sum_d q(d) \left(\log q(d) - \log p(d)\right) \\
&= \underbrace{\sum_d q(d) \log q(d)}_{-\text{entropy}} \underbrace{- \sum_d q(d) \log p(d)}_{\text{cross-entropy}}
\end{aligned}
$$

Let's look at normalization of p, the entropy term is easy because there is no p in it.

$$
\begin{aligned}
\sum_d q(d) \log p(d) &= \sum_d q(d) \log(\bar{p}(d)/Z_p) \\
&= \sum_d q(d) \left(\log \bar{p}(d) - \log Z_p\right) \\
&= \sum_d q(d) \log \bar{p}(d) - \sum_d q(d) \log Z_p \\
&= \sum_d q(d) \log \bar{p}(d) - \log Z_p
\end{aligned}
$$

In this case, $-\log Z_p$ is an additive constant, which can be dropped because we're optimizing.

This leaves us with the following optimization problem:

$$
\begin{aligned}
&\operatorname*{argmin}_\theta \mathbf{KL}(q_\theta || p) \\
&\quad = \operatorname*{argmin}_\theta \sum_d q_\theta(d) \log q_\theta(d) - \sum_d q_\theta(d) \log \bar{p}(d)
\end{aligned}
$$

Let's work out the gradient

$$
\begin{aligned}
&\nabla \left[\sum_d q_\theta(d) \log q_\theta(d) - \sum_d q_\theta(d) \log \bar{p}(d)\right] \\
&\quad = \sum_d \nabla \left[q_\theta(d) \log q_\theta(d)\right] - \sum_d \nabla \left[q_\theta(d)\right] \log \bar{p}(d) \\
&\quad = \sum_d \nabla \left[q_\theta(d)\right] \left(1 + \log q_\theta(d)\right) - \sum_d \nabla \left[q_\theta(d)\right] \log \bar{p}(d) \\
&\quad = \sum_d \nabla \left[q_\theta(d)\right] \left(1 + \log q_\theta(d) - \log \bar{p}(d)\right) \\
&\quad = \sum_d \nabla \left[q_\theta(d)\right] \left(\log q_\theta(d) - \log \bar{p}(d)\right)
\end{aligned}
$$

We killed the one in the last equality because $\sum_d \nabla \left[q(d)\right] = \nabla \left[\sum_d q(d)\right] = \nabla \left[1\right] = 0$, for any q which is a probability distribution.

This direction is convenient because we don't need to normalize p. Unfortunately, the "easy" direction is nonconvex in general—unlike the "hard" direction, which (as we'll see shortly) is convex.

Harder direction $\mathbf{KL}(p||q_\theta)$

$$
\begin{aligned}
\mathbf{KL}(p || q_\theta) &= \sum_d p(d) \log\left(\frac{p(d)}{q(d)}\right) \\
&= \sum_d p(d) \left(\log p(d) - \log q(d)\right) \\
&= \sum_d p(d) \log p(d) - \sum_d p(d) \log q(d)
\end{aligned}
$$

Clearly the first term (entropy) won't matter if we're just trying optimize wrt θ. So, let's focus on the second term (cross-entropy).

$$
\begin{aligned}
\sum_d p(d) \log q(d) &= \frac{1}{Z_p} \sum_d \bar{p}(d) \log(\bar{q}\,(d)/Z_q) \\
&= \frac{1}{Z_p} \sum_d \bar{p}(d) \left(\log \bar{q}\,(d) - \log Z_q\right) \\
&= \left(\frac{1}{Z_p} \sum_d \bar{p}(d) \log \bar{q}\,(d)\right) - \left(\frac{1}{Z_p} \sum_d \bar{p}(d) \log Z_q\right) \\
&= \left(\frac{1}{Z_p} \sum_d \bar{p}(d) \log \bar{q}\,(d)\right) - (\log Z_q) \left(\frac{1}{Z_p} \sum_d \bar{p}(d)\right) \\
&= \left(\frac{1}{Z_p} \sum_d \bar{p}(d) \log \bar{q}\,(d)\right) - \log Z_q
\end{aligned}
$$

The gradient, when q is in the exponential family, is intuitive:

$$
\begin{aligned}
\nabla \left[\frac{1}{Z_p} \sum_d \bar{p}(d) \log \bar{q}\,(d) - \log Z_q\right] &= \frac{1}{Z_p} \sum_d \bar{p}(d) \nabla \left[\log \bar{q}\,(d)\right] - \nabla \log Z_q \\
&= \frac{1}{Z_p} \sum_d \bar{p}(d) \phi_q(d) - \mathbb{E}_q\left[\phi_q\right] \\
&= \mathbb{E}_p\left[\phi_q\right] - \mathbb{E}_q\left[\phi_q\right]
\end{aligned}
$$

Why do we say this is hard to compute? Well, for most interesting models, we can't compute $Z_p = \sum_d \bar{p}(d)$. This is because p is presumed to be a complex model (e.g., the real world, an intricate factor graph, a complicated Bayesian posterior). If we can't compute Z_p, it's highly unlikely that we can compute another (nontrivial) integral under $\bar{p}$, e.g., $\sum_d \bar{p}(d) \log \bar{q}(d)$.

Nonetheless, optimizing KL in this direction is still useful. Examples include: expectation propagation, variational decoding, and maximum likelihood estimation. In the case of maximum likelihood estimation, p is the empirical distribution, so technically you don't have to compute its normalizing constant, but you do need samples from it, which can be just as hard to get as computing a normalization constant.

Optimization problem is *convex* when q_θ is an exponential family—i.e., for any p the *optimization* problem is "easy." You can think of maximum likelihood estimation (MLE) as a method which minimizes KL divergence based on samples of p. In this case, p is the true data distribution! The first term in the gradient is based on a sample instead of an exact estimate (often called "observed feature counts"). The downside, of course, is that computing $\mathbb{E}_p[\phi_q]$ might not be tractable or, for MLE, require tons of samples.

Remarks

- In many ways, optimizing exclusive KL makes no sense at all! Except for the fact that it's computable when inclusive KL is often not. Exclusive KL is generally regarded as "an approximation" to inclusive KL. This bias in this approximation can be quite large.
- Inclusive vs. exclusive is an important distinction: Inclusive divergences require $q > 0$ whenever $p > 0$ (i.e., no "false negatives"), whereas exclusive divergences favor a single mode (i.e., only a good fit around a that mode).
- When q is an exponential family, $\mathbf{KL}(p||q_\theta)$ will be convex in θ, no matter how complicated p is, whereas $\mathbf{KL}(q_\theta||p)$ is generally nonconvex (e.g., if p is multimodal).
- Computing the value of either KL divergence requires normalization. However, in the "easy" (exclusive) direction, we can optimize KL without computing Z_p (as it results in only an additive constant difference).
- Both directions of KL are special cases of α-divergence. For a unified account of both directions consider looking into α-divergence.

Acknowledgments

I'd like to thank the following people:

- Ryan Cotterell for an email exchange which spawned this article.
- Jason Eisner for teaching me all this stuff.
- Florian Shkurti for a useful email discussion, which caugh a bug in my explanation of why inclusive KL is hard to compute/optimize.
- Sabrina Mielke for the suggesting the "inclusive vs. exclusive" figure.

Comments

ALSO ON **GRADUATE DESCENT**

Backprop is not just the chain rule — ...

6 years ago • 7 comments

Almost everyone I know says that "backprop is just the chain rule." Although ...

Algorithms for sampling without ...

4 years ago • 21 comments

The probability of the sampling without replacement scheme can ...

How to test gradient implementations ...

6 years ago • 7 comments

Setup: Suppose we have a function, \(f: \mathbb{R}^n \rightarrow ...

Black-box optimization — ...

4 years ago • 1 comment

Black-box optimization algorithms are a fantastic tool that everyone should ...

W **Wittawat**

9 years ago

Nice post on KL divergence ! About the two directions of KL, I have one intuitive way to remember which direciton does what. For KL[p || q], if you expand out, you will have log(p(x)/q(x)) inside the integral. When is this quantity high ? It is when p(x) is high and q(x) is low. So to prevent it from being high (we want to minimize it), you must spread the mass of q(x) everywhere that p(x) has some mass. That is why it is mean seeking.

For KL[q || p], expanding out gives you log(q(x)/p(x)). This one is high when q(x) is high and p(x) is low. So to minimize, you want to allocate mass of q in the region where p is high (imagine that q(x) and p(x) are roughly equal in the local region and log(q(x)/p(x)) is roughly log(1)=0. Now what if q(x) is low when p(x) is high ? This is okay because the quantity is still low. That is to say that q may completely omit some mode. That is why it is mode seeking.

Hope it makes some sense...

8 0 Reply • Share ›

jaggi → Wittawat

5 years ago edited

what are typically values for KL[q||p] (how negative can it go?), it seems like q being tiny greater than 0, satisfy the minization condition. It doesn't need to find any mode. what am I missing here? mutliplication by q with log(q/p)?

0 0 Reply • Share ›

Tim Vieira Mod → jaggi

5 years ago

KL (in both directions) is always >= 0.

Describing KL(q||p) as "mode seeking" is a caricature, if there are several bumps that are 'near enough' then KL(q*||p) might cover both of them.

1 0 Reply • Share ›

jaggi → Tim Vieira

5 years ago

ahh, yes. thank you.
putting link for future reference on why - https://stats.stackexchange...

0 0 Reply • Share ›

Tim Vieira Mod → Wittawat

9 years ago

Thanks Wittawat! Nice explanation. I meant to incorporate something like this into the post, but simplified to just the zeros of p and q. I ended up not using it because it required too many special cases for division by zero and log(0).

0 0 Reply • Share ›

C **Cunxiao Du**

10 months ago

very nice post!
But for KL (P_\theta || P_\real), we need to access the log P_\real. I think that's also one of the reason why the reverse KL term is hard to compute? actually the reverse KL has very good character w.r.t some specific problems like generation (you only need to generate one corret instead of all the

possible one)

0 0 Reply • Share ›

Ahlad Rajput
4 years ago

I have two questions (1) Why you are normalizing the pdfs (2) why you have normalized only p in the easire direction and both p and q in the harder direction

0 0 Reply • Share ›

Tim Vieira Mod → Ahlad Rajput
4 years ago

Computing normalizing constants is generally the /computationally difficult/ part in doing probabilistic inference. Variational inference was created to avoid the computation of normalizing constants for `p`, which is typically an intractable Bayesian posterior distribution.

1 0 Reply • Share ›

Jae Duk Seo
5 years ago

very good post thank you for this!

0 0 Reply • Share ›

Jae Duk Seo
5 years ago

very very good blog post!

0 0 Reply • Share ›

Tim Vieira Mod → Jae Duk Seo
5 years ago

Thank you!

1 0 Reply • Share ›

zhenli
6 years ago edited

Much clearer now! Many thanks!

0 0 Reply • Share ›

Tim Vieira Mod → zhenli
6 years ago

Glad it was useful! :-)

0 0 Reply • Share ›

rasmusab
9 years ago

Just what I needed! Thanks!

0 0 Reply • Share ›

Recent Posts

- Fast rank-one updates to matrix inverse?
- On the Distribution of the Smallest Indices
- On the Distribution Functions of Order Statistics
- Animation of the inverse transform method
- Generating truncated random variates

Tags

numerical, efficiency, sampling-without-replacement, statistics, notebook, ordered-sampling, sampling, algorithms, Gumbel, decision-making, reservoir-sampling, optimization, rl, machine-learning, calculus, automatic-differentiation, implicit-function-theorem, Lagrange-multipliers, testing, counterfactual-reasoning, importance-sampling, datastructures, incremental-computation, data-structures, rant, hyperparameter-optimization, crf, deep-learning, structured-prediction, visualization
Follow @xtimv