

1. Kullback–Leibler divergence

D.J. Wischik

1 Samples from a continuous distribution

The Kullback–Leibler divergence is

$$\begin{aligned}\text{KL}(q \parallel p) &= \mathbb{E}_{X \sim q} \log \frac{q(X)}{p(X)} \\ &= \mathbb{E}_{X \sim q} \log q(X) - \mathbb{E}_{X \sim q} \log p(X)\end{aligned}$$

when q and p are distributions of the same type, i.e. either both probability mass functions so that the expectation is a sum, or both probability density functions so that it is an integral. In lectures we wrote $\text{KL}(\hat{q} \parallel p)$ where $\hat{q}$ is the empirical distribution of the data. What is this?

- When the problem has discrete data and we’re using a discrete distribution p to model it, then $\hat{q}$ is a probability mass function and $\text{KL}(\hat{q} \parallel p)$ is well-defined. As we get more samples, $\hat{q} \rightarrow q$, and $\text{KL}(\hat{q} \parallel p) \rightarrow \text{KL}(q \parallel p)$.
- If the underlying distribution is continuous and we use a continuous distribution to model it, there is a problem with writing $\text{KL}(\hat{q} \parallel p)$, because $\hat{q}$ is a discrete distribution and p is continuous. This is actually infinite, as the example below shows. So what do we mean by it?

Here is what we mean. When we solve

$$\text{minimize } \mathbb{E}_{X \sim q} \log q(X) - \mathbb{E}_{X \sim q} \log p(X) \text{ over parameters of } p$$

it’s only the second term, the model-dependent term, that matters. In machine learning, we give the estimated version a name:

$$\text{cross-entropy} = -\mathbb{E}_{X \sim \hat{q}} \log p(X) = -\frac{1}{n} \sum_i \log p(x_i).$$

I believe it’s useful to retain both terms, for derivations and for intuition. Define the ‘KL by convention’ or ‘KL with a constant’ as like the true KL but using a sample to approximate the model-dependent term:

$$\text{KL}_c(\hat{q} \parallel p) := \mathbb{E}_{X \sim q} \log q(X) - \mathbb{E}_{X \sim \hat{q}} \log p(X).$$

The first term is a constant, which we don’t know. It’s good to be always reminded of this constant, just as indefinite integrals should always come with a $+c$. The second term is computable and optimizable. In the course we write this conventional KL, with abuse of notation, as $\text{KL}(\hat{q} \parallel p)$. In the discrete case, fitting and comparing models is the same whether we’re using actual KL or conventional KL; in the continuous case the actual KL is infinite so only the conventional reading makes sense.

Example (smeared points). Let $\hat{q}$ be the empirical distribution of a set of datapoints $\{x_1, \dots, x_n\}$ in $\mathbb{R}$. Let q_ε be a version in which each datapoint is smeared into a narrow bump:

$$q_\varepsilon(x) = \frac{1}{n} \sum_{i=1}^n r_\varepsilon(x - x_i)$$

where $r_\varepsilon(x) = \varepsilon^{-1} r(x/\varepsilon)$ is a smear of scale ε centered at the origin, based on some underlying density function r . Then

$$\begin{aligned}\text{KL}(q_\varepsilon \parallel p) &\approx \mathbb{E}_{X \sim q_\varepsilon} \log q_\varepsilon(X) - \frac{1}{n} \sum_i \log p(x_i) \quad \text{assuming } p \text{ a cts func} \\ &= \mathbb{E}_{I, R \sim r_\varepsilon} \log q_\varepsilon(x_I + R) - \frac{1}{n} \sum_i \log p(x_i) \quad \text{where } I \text{ unif on datapoints} \\ &\approx \mathbb{E}_{I, R \sim r_\varepsilon} \log \left(\frac{1}{n} r_\varepsilon(R) \right) - \frac{1}{n} \sum_i \log p(x_i) \quad \text{since } r_\varepsilon \text{ small away from } 0 \\ &= -\log n - \log \varepsilon + \mathbb{E}_{R \sim r} \log r(R) - \frac{1}{n} \sum_i \log p(x_i) \quad \text{by algebra.}\end{aligned}$$

As $\varepsilon \rightarrow 0$, in other words $q_\varepsilon \rightarrow \hat{q}$, this goes to $+\infty$. Informally, $\hat{q}$ puts a bolus of probability mass on its datapoints whereas p puts an infinitesimal amount on each, and so p is a terrible descriptor. $\diamond$

2 The maths of overfitting

We want $\min_{\theta} \text{KL}(q \parallel p_{\theta})$. The first part is constant wrt θ , so we might as well seek $\max_{\theta} \mathbb{E}_{X \sim q} \log p_{\theta}(X)$. But when we do this maximization over a training dataset rather than the truth, we overfit.

To see this algebraically, first some definitions. Let $L_r(p_{\theta}) := \mathbb{E}_{X \sim r} \log p_{\theta}(X)$, and let $\hat{q}^{\mathcal{D}}$ be the empirical distribution over a dataset $\mathcal{D}$. With these definitions, what we really want is $\max_{\theta} L_q(p_{\theta})$, but the training goal is $\max_{\theta} L_{\hat{q}^{\text{tr}}}(p_{\theta})$ where tr is the training dataset. Also, let Tr be an IID sample from q of the same size, n , as the training dataset.

The overfitting gap arises as follows.

$$\begin{aligned} \max_{\theta} L_q(p_{\theta}) &= \max_{\theta} \frac{1}{n} \sum_{i=1}^n L_q(p_{\theta}) && \text{since all } n \text{ terms identical} \\ &= \max_{\theta} \mathbb{E}_{\text{Tr}} L_{\hat{q}^{\text{tr}}}(p_{\theta}) && \text{by linearity of expectation} \\ &\leq \mathbb{E}_{\text{Tr}} \max_{\theta} L_{\hat{q}^{\text{tr}}}(p_{\theta}) && \text{by Jensen's inequality} \\ &\approx \max_{\theta} L_{\hat{q}^{\text{tr}}}(p_{\theta}) && \text{single-sample Monte Carlo.} \end{aligned}$$

We can also see the gap in terms of the score of the fitted model. Let $\hat{\theta}^{\text{tr}}$ be the optimizer from fitting the training data. Then, wrapping the derivation above,

$$L_q(p_{\hat{\theta}^{\text{tr}}}) \leq \text{LHS} \lesssim \text{RHS} = L_{\hat{q}^{\text{tr}}}(p_{\hat{\theta}^{\text{tr}}}).$$

In words, if we naively train to find $\max_{\theta} L_{\hat{q}^{\text{tr}}}(p_{\theta})$, we incur the Jensen gap compared to what we actually wanted to solve which is $\max_{\theta} L_q(p_{\theta})$.

We incur the Jensen gap whenever we choose θ (or more generally whenever we choose between models) based on a sample rather than on the true distribution. In high-capacity models, the naive optimization pushes p_{θ} towards memorizing $\hat{q}^{\text{tr}}$, which makes the Jensen gap huge. So instead we throw in an ad-hoc regularizer that we hope will produce a better $\hat{\theta}$. Standard regularizers reduce the objective's incentive or the model's ability to memorize.

Holdout approximation. We use holdout scores because they give us a genuine estimate of the true score. Suppose we've found parameters θ_0 in training; the true score is $L_q(p_{\theta_0})$, and by the law of large numbers (i.e. many-sample Monte Carlo) this is

$$L_q(p_{\theta_0}) \approx L_{\hat{q}^{\text{ho}}}(p_{\theta_0}).$$

This approximation requires θ_0 to be chosen independently of the holdout set. In particular, if we choose the regularization level using a validation dataset, then we need a separate untouched holdout to estimate the score.

For further reading, see Shalev-Shwartz and Ben-David, Understanding Machine Learning (2014), chapter 11. Or take a course on Statistical Learning Theory. It will teach you about the 'uniform law of large numbers', which is the theorem behind the holdout approximation's 'independently of the holdout set'.