Exercise Sheet 1

Probabilistic models of modern AI

vassilis.papadopoulos@epfl.ch – www.vassi.life/teaching/aiproba

Exercise 1 The Bregman Divergence

A divergence $D(p||q)$ is a sort of 'statistical distance' of distribution $p(x)$ compared to $q(x)$, although its definition is looser than a distance. More precisely, a divergence is a binary function $D : M \times M \to [0, \infty)$, where M is a manifold, which we will most often take to be $\mathbb{R}^n$ (for discrete probability distributions[1]). It satisfies :

1 $D(p||q) \geq 0 \ \forall \ p, q$

2 $D(p||q) = 0 \Leftrightarrow p = q$

3 ($D(p||p + dp)$ is a positive-definite quadratic form in dp)

The Bregman divergence B_C is a class of divergences defined with the help of a function $C : M \to \mathbb{R}$. We can define $B_C(p||q)$ geometrically as the distance between $C(p)$ and the Taylor expansion of C at q, evaluated at p, see Fig. 1.

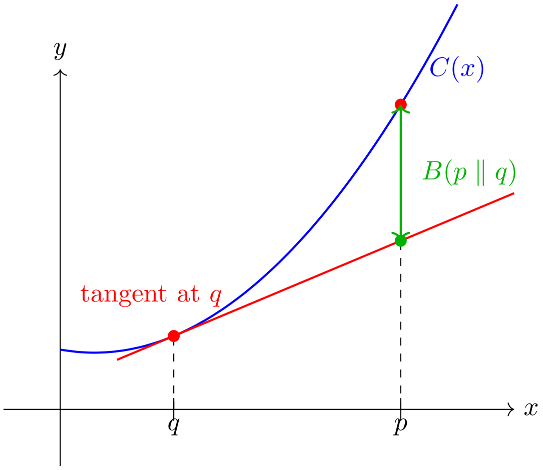

Figure 1: Bregman divergence

1. Show that this geometric definition corresponds to $B(p \parallel q) = C(p) - C(q) - \langle p - q, \nabla C(q) \rangle$
2. Show that $B_C(p \parallel q)$ is a divergence iff C is convex.
3. Find all functions $\hat{C}(p)$ s.t. $B_C(p \parallel q) = B_{\hat{C}}(p \parallel q), \ \forall p, q$
4. Find a function $C(p)$ such that $B_C(p \parallel q) = \sum_x |p(x) - q(x)|^2$
5. Find a function $C(p)$ such that $B_C(p \parallel q) = KL(p \parallel q) = \sum_x p(x) \log(\frac{p(x)}{q(x)})$

$KL(p \parallel q)$ is called the Kullback-Leibler divergence, and is used very commonly to train neural networks, but also has an important interpretation in coding theory.

[1] For general density functions, the manifold would be infinite dimensional, which would complicate things. Usually, we restrict to a parametrised sub-class, e.g. μ and σ for Gaussians, or $\theta \in \mathbb{R}^N$, for a neural network with N parameters)

6. Assume that you have a stream of symbols x_i, that you believe is generated with propability distribution $p(x_i)$.
   a) Compute the average number of bits per symbol you will need to transmit that data, if it is indeed generated by $p(x_i)$
   b) Compute the average number of bits per symbol you will need if it is actually generated by $q(x_i)$. How many 'extra bits' are you using on average in this case?
7. Find all the functions C s.t. $B_C(p \parallel q)$ is symmetric.[2]

Exercise 2 Proper scoring rules, in general

Consider the situation where you want to elicit an accurate prediction from an *expert*, who is tasked to give you probabilities p_i for events i happening, where we assume that the events are disjoint. To motivate the *expert*, you come up with a scoring rule $s_i(p)$, which corresponds to the payment the *expert* will receive if event i is realized.

1. Consider the simple scheme where $s_i(p) = p_i$, that is, you reward the expert with the amount of probability mass he has allocated to the true event.
   a) Let's say we are eliciting a binary prediction (e.g. will it rain, or will it not). If the expert considers a probability $q = 0.7$ of rain, what should he report to maximize his expected score?
   b) More generally, for n possibilities, if the expert believes probabilities q_i, what should he report so that he maximizes his expected score?

Clearly, we need to be more thoughtful about the choice of *scoring rule*. We would like to design it such that the expert is incentivized to report his beliefs honestly. Such a scoring rule is called *proper*.

2. In this section we will recover the general form of proper scoring rules :

$$s_i(q) = G(\delta_i) - B_G(\delta_i \parallel q) + C_i \tag{1}$$

   Where $G(q)$ is a strictly convex function, and C_i arbitrary constants. We will use a derivation inspired by (Hanson, 2002)[3]. We will be considering an expert who has beliefs q_i about the probabilities of n events.
   a) Show that a scoring rule is *proper* if $F(r) = \sum_i q_i s_i(r)$ has a unique maximum at $r = q$.
   b) Verify that 1 is indeed a proper scoring rule.
   c) Using a Lagrange multiplier, show that if the scoring rule is proper, then there exists a $\lambda(r)$ s.t. :

$$\sum_i q_i \partial_j s_i(q) - \lambda(q) = 0 \quad \forall\ j \tag{2}$$

   d) Deduce that we must have [4]

$$s_i(q) = \partial_i \rho(x) - \lambda(x) + C_i \tag{3}$$

   We will set $C_i = 0$ in what follows.
   e) Show that we can rewrite $s_i(q)$ as a function only of $G(q) = \sum_i q_i s_i(q)$ and its derivative.
   f) Show that we can write :

$$s_i(q) = G(\delta_i) - B(\delta_i \parallel q) \tag{4}$$

   g) The form we have found so far only guarantees that $F(r)$ has a critical point at q. Show that if it is a maximum, then $G(q)$ must necessarily be strictly convex (on the domain where $\sum_i q_i = 1$)![5]
3. So far, we have shown only that a proper scoring rule will successfully elicit the belief of an expert who tries to maximize his expected score. However, is the score 'fair'? Namely, if an expert provides a better prediction, will he obtain a higher score, on average?

[2] Hint, rot encoded: Wzijk, NCFX jyfn kyrk kyv vhlrkzfe kf jrkzjwp zj $\backslash tufk \backslash erscrT(g) = 2T(g)$. Kyve, wzeu kyv xvevirc jfclkzfe nzky kyv dvkyfu fw tyrirtkvizjkztj, fi sp xfzex kf iruzrc tffiuzerkvj.

[3] The derivation in (Hanson, 2002) is, however, wrong, and we correct it here

[4] Hint, rot encoded : Qvssreragvngr gur rdhngvba j.e.g. d_x

[5] Hint, rot encoded: Knwxy, ywd yt uwtaj ny bnymtzy hfwnsl fgtzy ymj htsxywfnsy $\sum_n v_n = 1$. Tshj ny nx itsj, wjujfy ymj xfrj xyjux, stb yfpnsl ymj htsxywfnsy nsyt fhhtzsy

Now, we may think we are done and that we will get accurate reports from the agents. However, we notice now that agents are not maximizing for the expected score, but for the expected *utility* of the score! In essence, the expert may assign their own value (utility) to the obtained score. For example, would you consider it twice as good to gain 2 cents rather than 1? Probably not, as they are both essentially worthless (0 utility).

This can be modelized by a utility function $u : \mathbb{R} \to \mathbb{R}$, that takes as input a score s, and outputs its utility $u(s)$. Generally, we will assume that u is an increasing function, but in some situations it might not be (what if winning 100 million dollars in cryptocurrency might make you a target, and so you would rather win 'only' 10 million?). In any case, the expert is now maximizing :

$$U = \sum_i p_i u(s_i(r)) \tag{5}$$

And all our efforts to design a proper scoring rule are now void. Unless ?

4. Instead of rewarding the experts directly with a score s (you can think of the score as money), we will instead give them some quantity s of lottery tickets (you can imagine having fractional lottery tickets). The prize of the lottery is some large prize M, which does not depend on the prediction of the expert. Show that this reward system restores the 'proper' property of the scoring rule, independently of the utility function of the expert (so long as $U(M) > 0$).
5. (open) How can we deal with negative scores in this case? Do we need to consider negative scores?

Exercise 3 BONUS : Scoring rules in code

This is a hands-on exercise that will require to code (just a little bit) in Python. The easiest way to do it is to head to https://tinyurl.com/aiproba1. Alternatively, if you want to work on your own computer, you can download the .ipynb Colab file though not recommended as I haven't added dependencies yet.

The goal of this practical exercise is to get a feel of how the scoring rules affect the agent's report when that agent is a neural network. In this case, there isn't really a notion of 'beliefs' that are reported correctly or incorrectly, and we can access only the report of the neural network. Still, if we want the neural network to output correct probabilities as much as possible, we will need to use a proper scoring rule.