Statistical Physics of Computation - Exercises

Emanuele Troiani, Vittorio Erba, Yizhou Xu

September 2024

Week 6

Gibbs estimator for the teacher-student binary perceptron

This week we study again the linear classification (perceptron) problem, but with a twist. As before we have $P = \alpha N$ points $\{\xi^\mu\}_{\mu=1}^P$ of dimension N, each with a label $\sigma^\mu = \pm 1$. We want to train a perceptron (which we will call the student) to distinguish between the two samples. In other works we want a vector J with $\|J\|^2 = N$ such that

$$\frac{\sigma^\mu}{\sqrt{N}} J^T \xi^\mu > 0\,. \tag{1}$$

The twist here is that the labels are not random, instead they are a function of the position of the points ξ^μ, generated using a teacher vector J^* (uniformly distributed on the sphere of radius $\sqrt{N}$)

$$\text{sign}\left(\frac{1}{\sqrt{N}} J^{*T} \xi^\mu\right) = \sigma^\mu\,. \tag{2}$$

Thus, we have non-trivial correlations between points and label, which allows us to speak about *learning*.

1. Why does it make no sense to study the satisfiability (SAT/UNSAT) transition in this setting?

We can imagine a setting in which we know everything about the data and its generation process (as in we know the ξ^μ, the σ^μ, i.e. we know the training set, and we know that the labels are generated with a perceptron) but we don't have access to the specifics of the teacher (we know that J^* has a certain distribution, i.e. uniform on the sphere as J but not its values). Two interesting question to ask are:

- How do we estimate J^* from the data?
- Given an estimate of J^*, which may be imprecise, how likely are we to correctly classify a new sample (usually called *generalisation error*)?

We saw during the lecture that one strategy to estimate J^* is *Gibbs learning*, where we take our estimate $\hat{J}$ uniformly from the space of classifiers that correctly classify the whole training set (let's call this the solution space). This is not necessarily a practical choice, as it may not be simple to sample uniformly from the solution space, and in general actual algorithms may only be able to access a limited part of the solution space. Yet, it is an instructive computation, and

it will allow us to study more realistic algorithms later in the course. Thus, today we will study Gibbs learning, and we will estimate the generalisation error of the Gibbs estimator.

As we saw in the lecture, given that we want to compute average properties over the uniform measure on the solution set, it is useful to introduce the measure

$$p(J) = \frac{1}{Z}\delta\left(J \in \text{solution space}\right) . \tag{3}$$

and we recognised that the normalisation factor Z, our partition function, is just Gardner's volume

$$Z = \int d\mu(J)\Pi_{\mu=1}^{p}\theta\left(\frac{\sigma^{\mu}}{\sqrt{N}}\sum_{i=1}^{N}J_i\xi_i^{\mu}\right) . \tag{4}$$

Thus, we may study the averaged free entropy associated to Gardner's volume to extract an order parameter (likely the overlap), which will hopefully help us estimate the generalisation error.

Derivation of the state equation

During the lecture, we saw that the (replicated) averaged partition function can be written as

$$\mathbb{E}_{\xi,\sigma,J^*}Z^n = 2^P\mathbb{E}_{\xi}\prod_{a=0}^{n}\left[\int d\mu(J^a)\prod_{\mu=1}^{P}\theta\left(\frac{1}{\sqrt{N}}\sum_{i=1}^{N}J_i\xi_i^{\mu}\right)\right] . \tag{5}$$

Notice that we are now averaging also over the teacher weights J^*, as they are an integral part of the data generation procedure. Notice also that we explicitly keep the overall factor 2^P coming from summing over the labels: in previous computations we did not keep them as the alter the partition function by an overall factor, but in this case it will be nice to be a bit more precise.

2. Re-derive Eq. (5) even if we derived it already during the lecture. Being at ease with setting up a replica computation is a skill that you need to practice.

We notice that Eq. (5) is the same as the averaged Gardner's volume for random labels (the model we looked at during lectures 3 and 4), just with $n+1$ replicas instead of n: the teacher acts as an additional replica, as its distribution, given the training data (ξ, σ), is uniform over the solution set by definition. Thus, we can follow the same steps of the computation (keeping track of the additional 2^P) to get

$$\begin{aligned}\mathbb{E}_{\xi,\sigma,J^*}Z^n = \int \Pi_{a<b}d\Sigma^{ab}\left[2\int \Pi_a d\Delta^a \mathcal{N}\Big(\{\Delta^a\}_{a=1}^{n}\Big|0,\Sigma^{ab}\Big)\theta(\Delta^a)\right]^P \\ \times\left[\int \Pi_a d\mu(J^a)\Pi_{a<b}\delta\left(N\Sigma^{ab}-\sum_{i=1}^{N}J_i^aJ_i^b\right)\right]\end{aligned} \tag{6}$$

where $\Sigma_{ab} \in \mathbb{R}^{n+1\times n+1}$ is the overlap order parameter for this problem. We switched notations from q to Σ to highlight that we are dealing here with an *a priori* different order parameter, as it also encodes the overlap between the teacher (0-th replica) and all the other replicas (entries $\Sigma_{0,a}$ for $a = 1\dots n$).

The most general RS ansatz we could impose in this case is given by

$$\Sigma^{ab} = \begin{pmatrix} 1 & m & m & \dots & m \\ m & 1 & q & \dots & q \\ m & q & 1 & \dots & q \\ \vdots & \vdots & \vdots & \ddots & q \\ m & q & q & \dots & 1 \end{pmatrix} \tag{7}$$

where the meaning of m and q is

$$m = \frac{J^{*T} J^a}{N}\,, \qquad\qquad q = \frac{J^{aT} J^b}{N} \tag{8}$$

for $a, b = 1, \dots, n$. In words, m is the overlap between the teacher and one of the replicas of the student, while q is the overlap between two replicas of the student. We will see in the future problems in which $m \neq q$, but in our case we can take $m = q$.

3. Explain why it makes sense to impose $m = q$.

4. Show that in the RS ansatz

$$\frac{1}{nN} \log \int \Pi_a d\mu(J^a) \Pi_{a<b} \delta \left(N\Sigma^{ab} - \sum_{i=1}^{N} J_i^a J_i^b \right) = \frac{1}{2}(\log(1-q) + q) \tag{9}$$

   in the large N limit, followed by the small n limit.

5. Show that in the RS ansatz and small n limit

$$\frac{1}{n} \log 2 \int \Pi_a d\Delta^a \mathcal{N}\Big(\{\Delta^a\}_{a=1}^n \Big| 0, \Sigma^{ab} \Big) \theta(\Delta^a) = 2 \int Dz H\left(\frac{\sqrt{q} z}{\sqrt{1-q}} \right) \log H\left(\frac{\sqrt{q} z}{\sqrt{1-q}} \right) \tag{10}$$

   You may find useful to know that

$$\int Dz H(az) = \frac{1}{2}\,. \tag{11}$$

6. Conclude that the averaged free entropy associated to the partition function Z is given by $\phi(q)$ evaluated at it's critical point, where

$$\phi(q) = \frac{1}{2}\log(1-q) + \frac{q}{2} + 2\alpha \int Dz H\left(\frac{\sqrt{q} z}{\sqrt{1-q}} \right) \log H\left(\frac{\sqrt{q} z}{\sqrt{1-q}} \right)\,. \tag{12}$$

One can show that the state equation is given by

$$\frac{q}{\sqrt{1-q}} = \frac{\alpha}{\pi} \int Dz\, e^{-\frac{qz^2}{2}} H\left(\sqrt{q} z\right)^{-1} \tag{13}$$

The generalisation error

In principle q has all the information on the problem, but it's not what people care about in learning. As we mentioned, what we would really like to compute is the generalisation error, i.e. the probability of correctly classifying a new, previously unseen data pair $(\xi_{\text{new}}, \sigma_{\text{new}})$.

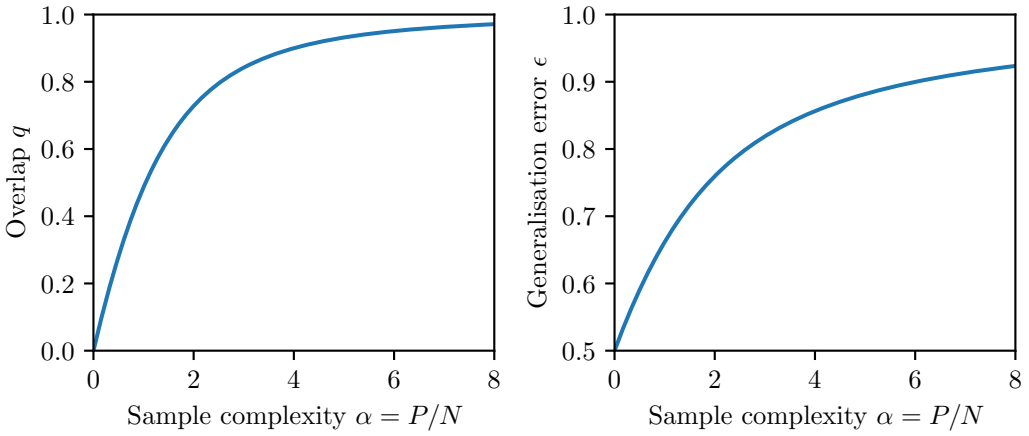

Figure 1: Overlap and generalisation error for our problem

10. Show that the generalisation error for a classifier $\hat{J}$ is given by

$$\epsilon(\hat{J}) = \mathbb{E}_{\xi_{\text{new}}} \theta \left(\frac{1}{\sqrt{N}} \hat{J}^T \xi_{\text{new}} \frac{1}{\sqrt{N}} J^{*T} \xi_{\text{new}} \right) \tag{14}$$

11. Manipulate the expression to write

$$\epsilon(\hat{J}) = \mathbb{E}_{z,z^*} \theta \left(z z^* \right) \tag{15}$$

where z, z^* are zero mean Gaussian variables. What is their covariance?

Finally, one can solve analytically the Gaussian expectation to get

$$\epsilon(\hat{J}) = \frac{1}{\pi\sqrt{1-q^2}} \int_0^\infty \int_0^\infty dz dz^* \exp\left\{ -\frac{z^2 + z^{*2} + 2qzz^*}{2(1-q^2)} \right\} = \frac{1}{2} + \frac{1}{\pi} \arcsin(q) \tag{16}$$

Notice that the generalisation error does not depend on $\hat{J}$ in the high dimensional limit. This is due to the concentration of the overlap: in the solution set, if we fix a random vector (the teacher) and extract a second random vector (the student), their overlap will always be q (with probability approaching 1 as $N \to \infty$). The non-trivial observation here is that, given the data, the teacher is a totally normal random vector in the solution set, which is what we where arguing when we put $m = q$ in point 3. In other problems, where $m \neq q$, the teacher vector is special. In that case, two random vectors from the solution set would have overlap q, but a random vector from the solution set would have overlap m with the teacher.

Numerical solution of the state equation

Eq 13 is not really useful unless we can actually plot $q(\alpha)$. Fortunately, there is a standard way of solving numerically such equations. The idea is to guess a starting value for q^0, say $q^0 = 0$, and then iterates a suitable map $q^{t+1} = F[q^t]$ until convergence to a solution of eq (13). More precisely, define

$$F[q] = \frac{\alpha\sqrt{1-q}}{\pi} \int Dz \frac{e^{-\frac{qz^2}{2}}}{H\left(\sqrt{q}z\right)} \tag{17}$$

and iterate the map

$$q^{t+1} = \delta F[q^t] + (1-\delta)q^t \tag{18}$$

for some $\delta \in (0,1)$. δ is called damping, and is a helpful feature to stabilise and speed-up convergence of these iterative algorithm.

7. Show that the fixed points of Eq. 18 are solutions to the state equation Eq. 13.

Such an algorithm is very similar to what we will see in the rest of the class, but for the moment you can think of it as a gradient descent iteration on the free entropy. For your reference, we have implemented this procedure and plotted the results in Figure 1.

8. Code Eq. 18 and reproduce Figure 1. You can use the programming language that you prefer, but we recommend Python. Hints:

   - You should regularise the denominator in Eq 18, i.e. substitute $H\left(\sqrt{q}z\right) \to H\left(\sqrt{q}z\right) + 0.0001$, where 0.0001 is any small numeric constant. This will prevent the denominator to diverge when H goes to zero.
   - There is no need to code the function H. Indeed, $H(x) = \mathrm{erfc}(x/\sqrt{2})/2$, and the erfc function is implemented in most languages (such as in the Scipy python library).
   - To integrate scalar function you can find many libraries, such as *scipy.integrate.quad* in Python. For the integration interval, which a priori is $\mathbb{R}$, use $(-M, M)$ for a large but finite M. $M \approx 10$ should work. Indeed, the Gaussian measure basically puts to zero the integrand ouside this modified interval.