Regularization and application to model selection

Contents

0 Reminder

We recall two results from the previous talks.

- (talk 1) Any probability distribution P on X corresponds to an encoding scheme: $x \in X$ is encoded in $-\log_2(P(x))$ bits.
- (talk 2) The Solomonoff universal probability distribution is defined[1] by

$$P_3(x) = \sum_{\text{all random programs}} 2^{-|p|} P(p \,\text{outputs}\, x), \tag{1}$$

  where $|p|$ is the length of the program p. It is reasonable to try to approach this distribution as closely as possible.

1 Motivation for regularization

We are going to study regularization and model selection. Regularization is the following problem:

[1] It is actually one of several possible definitions.

We are given a training dataset $\mathcal{D}$, a family of models $\{P_\theta, \theta \in \Theta\}$, and we try find a good θ to predict future data.

A natural choice would be $\theta^* = \mathrm{argmax}_\theta(p_\theta(\mathcal{D}))$, but in practice, if we try to apply p_{θ^*} to a validation set $\mathcal{D}'$, it will not work in general. We give two examples to illustrate this:

- The "zero-frequency problem": if some element z never occurs in the training data $\mathcal{D}$, P^*_θ will give zero probability to z, which corresponds to an infinite codelength, which is not acceptable (problems will arise if the validation $\mathcal{D}'$ contains z).
- A case of extreme overfitting: P_θ can learn $\mathcal{D}$ perfectly, i.e. $P_\theta = \delta_{\mathcal{D}}$. Then, P_θ will probably be a very bad predictor for new data $\mathcal{D}'$. This can happen when $\dim(\Theta)$ is too large.

We are going to see how these issues can be dealt with.

2 Possible solutions

There exists several approaches to solve the two problems above:

1. Replace the model $\{P_\theta, \theta \in \Theta\}$ with a lower dimensional model. This can be helpful for the overfitting problem, as shown in the following example:

   Suppose we are trying to predict text with a Markov model of order 100. Each string of 100 characters will occur only once and therefore, we will get a large set of deterministic rules (which will probably never be used). This would not happen when using a Markov model of order 2, for example.

   Another example is regression: if we try to fit a polynomial of degree n to n noisy points $(x_i, f(y_i) + \mathcal{N}(0, \varepsilon))$, we can find a polynomial that fits the points exactly, but which will be a very bad estimation of f. Limiting ourselves to "low degree polynomials" would be helpful for this.

   Of course, "low degree polynomials" is not well defined (it should be the set of polynomials of degree inferior to some constant k, but how should we choose k ?): the question "how should we lower the model dimension ?" is an important one.

   It is also important to notice that this method does not solve the zero-frequency problem (although it can help with the *number of occurences* of zero-frequency problems).

2. Use Bayesian models for prediction: instead of using a single value θ^*, use several values of θ (as seen in talk 2). More precisely, instead of p_{θ^*}, use the mixture $\int_\theta p_\theta w(\theta)$, where $w(\theta) \propto \alpha(\theta) p_\theta(\mathcal{D})$, where α is a prior on Θ, and $\mathcal{D}$ is the data.

   We have a canonical prior (Jeffreys' prior, see talk 3), but it is not always well-defined, and moreover, in general (including with other priors), the final prediction cannot be computed algebraically. In that case, there are two possible approximations: either select a few samples $\theta \sim w(\theta)$ (Monte-Carlo), or use an approximation of $w(\theta)$ for which the integral can be computed.

3. Add virtual data to $\mathcal{D}$:[2] if $\mathcal{D} = \{x_1, ..., x_n\}$, train θ on the augmented dataset $\tilde{\mathcal{D}} := (y_1, ..., y_p, x_1, ..., x_n)$, where the y_i are virtual data.

   This approach solves the zero-frequency problem: suppose we have $\mathcal{A} = \{A, B, ..., Z\}$. We can add for example one observation of each letter to the data: if $\mathcal{D} = AAAAA$, then $\tilde{\mathcal{D}} = ABC....XYZAAAAA$, and now, if $x \neq A$, $P_{\theta^*}(x) = \frac{1}{26+5}$. It is also possible to weight the virtual data (instead of adding each letter once, add each letter $\frac{1}{26}$ times, which yields a total of one "virtual observation").

   Another approach to add noise to the data (for example, if we are considering a regression problem with data (x_i, y_i), replace y_i by $y_i + \mathcal{N}(0, \varepsilon)$). An advantage of this method is that by construction, we can expect a good behavior with noisy data.

   These two approches are very similar: replacing a letter by a uniformly chosen letter with probability p is equivalent to having this same proportion p of "uniform letters" at the beginning of the word. In our example above, having $\tilde{D} = ABCD...ZAAAAA$ is equivalent to having a $\frac{26}{31}$ probability to replace the next letter by a random letter. Notice however that this probability decreases when data is added.

4. Cross-validation: split the data $\mathcal{D} = (\mathcal{D}_1, \mathcal{D}_2)$, train θ on $\mathcal{D}_1$, and test it on $\mathcal{D}_2$. This method has several shortcomings:

   - It is reasonable only in the i.i.d. case. Using cross-validation for weather prediction over forty years by splitting in forty periods of one year would miss long-term trends, for example.
   - Cross-validation does not help with the zero-frequency problem either (if some value z never appears in any of the $\mathcal{D}_i$ there is no reason to select a θ giving z a non-zero probability).
   - "train θ on $\mathcal{D}_1$" is vague: a certain number of methods will be used θ, and then, cross-validation will help to select the "best" θ *among these values.*

     It is however something reasonable to do, and it is widely used.

5. Finally, it is possible to introduce "regularization terms" into the optimization problem. For example, instead of $\theta^* = \text{argmax}_\theta P_\theta(\mathcal{D})$, pick

   $$\theta^* = \text{argmax}_\theta \ln P_\theta(\mathcal{D}) - R(\theta), \tag{2}$$

   where $R(\theta)$ is a penalization for some values of θ, for example $R(\theta) = \lambda\|\theta\|^2$.

   One of the shortcomings of this method is that *it is not invariant with respect to reparametrization of* θ: we can change the answer to our problem by simply formulating it differently!

   The choice of the penalty is also an important question: even if we suppose that the general form of the penalty $R(\theta) = \lambda\|\theta\|^2$ is fixed, the choice of λ will have a huge influence .

[2] In some cases, the Bayesian prediction can be written that way. See talk 3, footnote 5.

Finally, this method uses only one value of θ, which is usually not a good idea.[3]

It is interesting to notice that there exists a relationship between the regularization term and the Bayesian predictor. For example, the penalty $R(\theta) = \lambda\|\theta\|^2$ corresponds to a Gaussian prior on θ. The posterior of a given θ is then proportional to $P_\theta(\mathcal{D})e^{-\lambda\|\theta\|^2}$, and we can see that the regularization method (2) simply maximizes the (logarithm of the) posterior (however, it still uses only one value of θ, contrary to the Bayesian prediction).

We are now going to discuss the zero-frequency problem more precisely.

3 Around the zero-frequency problem

We start by giving the Kolmogrov complexity point of view about the zero-frequency problem, and we show that while it always gives a strictly positive probability to each character (thus eliminating the zero-frequency problem), the number of bits wasted if there really is a deterministic rule is bounded by a constant.

3.1 Solomonoff universal predictor

We recall the universal predictor:

$$P_u = \sum_{\text{all random programs}} 2^{-|p|} P(p \,\text{outputs}\, x) \tag{3}$$

where $|p|$ is the length of p. This is the model we are trying to approach.

Suppose now that we have a model p_θ built from the data. We define $p_{\theta,t,x}$ as the program which has the same output as p_θ, except at position t where it prints x. We have

$$|p_{\theta,t,x}| \leqslant |p_\theta| + K(t) + K(x) \tag{4}$$

Consequently, we have the relationship:

$$P_u(x_1, ..., x_{t-1}, x'_t, x_{t+1}, ...) \geqslant P_u(x_1, ..., x_{t-1}, x_t, x_{t+1}, ...)2^{-K(t)}2^{-K(x'_t)}. \tag{5}$$

For example, if $x \in \mathcal{A}$, where $\mathcal{A}$ is a finite alphabet:

$$K(x) \leqslant \log_2 |\mathcal{A}|, \,\text{and}\, K(t) \leqslant \log_2 t + 2\log_2 \log_2 t. \tag{6}$$

Therefore , we have

$$P_u(x_1, ..., x_{t-1}, x'_t, x_{t+1}, ...) \geqslant P_u(x_1, ..., x_{t-1}, x_t, x_{t+1}, ...)\frac{1}{|\mathcal{A}|}\frac{1}{t(\log_2 t)^2}. \tag{7}$$

In other words, the probability of a series of observations should not be "too far" from the probability of a series that is identical everywhere, except at time t.

The analysis above can be extended to more general cases: for example, the uniform distribution on $\mathcal{A}$ can be replaced by any probability distribution p on $\mathcal{A}$, and the $\frac{1}{|\mathcal{A}|}$ in (7) simply becomes $p(x)$.

[3] In the case of the virtual data, the problem is less important, since it is possible to ensure that each symbol is given a strictly positive probability.

3.2 Link with the zero-frequency problem

We consider data $\mathcal{D}$ sampled from a finite alphabet $\mathcal{A}$, and we suppose that we have a true deterministic rule in the dataset. In this section, we take $\mathcal{D} = (A, A, A...)$ (i.e. $x_i = A$ for all i) for simplicity, but the same treatment can be applied to any deterministic rule.

If we know that the real probability distribution is a Dirac in A, we find that the $\mathcal{D}$ can be encoded in 0 bits.

We are interested in the number of additional bits needed to encode the data $\mathcal{D}$ (this number is called the *regret*) when using methods that avoid the zero-frequency problem.

We will use the notation $\mathcal{D}_t := (x_1, ..., x_t)$ (in the case we will be studying here, $\mathcal{D}_t = (A, ..., A)$).

3.2.1 Solomonoff predictor

By using equation (7), we find that for each $x \in \mathcal{A}$,

$$P_u(x_t = x|\mathcal{D}_{t-1}) \geqslant P_u(x_t = A|A, ..., A)\frac{1}{|\mathcal{A}|}\frac{1}{t(\log_2 t)^2}. \tag{8}$$

However, it can be argued that in this case, asymptotically, we almost have the equality $P_u(x_t = x|\mathcal{D}_{t-1}) \approx P_u(x_t = A|\mathcal{D}_{t-1})\frac{1}{|\mathcal{A}|}\frac{1}{t(\log_2 t)^2}$ for $x \neq A$, since the shortest description of the sequence would really be "only A, except at time t, where it is x". Using this approximation, we find

$$P_u(x_t = A|\mathcal{D}_{t-1}) \approx 1 - (|\mathcal{A}| - 1)\frac{1}{|\mathcal{A}|}\frac{1}{t(\log_2 t)^2}, \tag{9}$$

and we can compute the regret:

$$-\sum_t \log_2 P_u(x_t = A|\mathcal{D}_{t-1}) \approx \sum_t (|\mathcal{A}| - 1)\frac{1}{|\mathcal{A}|}\frac{1}{t(\log_2 t)^2}, \tag{10}$$

which is bounded.

In other words, with this method, we waste only a finite number of bits to avoid the zero-frequency problem when predicting a Dirac. Let us show this is not the case when adding virtual data.

3.2.2 virtual data

The smoothing due to virtual data is $O(\frac{|\mathcal{A}|}{t})$, and consequently, we have: $P^{\text{virtual}}(x_{t+1} = x) \geqslant \frac{1}{t+|\mathcal{A}|}$ for each $x \in \mathcal{A}$. Let us show that this probability is too high. We have:

$$P^{\text{virtual}}(x_{t+1} = A|\mathcal{D}_t) = 1 - \frac{|\mathcal{A}| - 1}{t + |\mathcal{A}|}, \tag{11}$$

whereas $P^{\text{opt}}(x_{t+1} = A|\mathcal{D}) = 1$. Consequently, the number of additional bits needed to encode $\mathcal{D}$ is:

$$-\sum_t \log_2 P^{\text{virtual}}(x_{t+1} = A|\mathcal{D}_t) \approx \sum_t \frac{|\mathcal{A}|}{t}, \tag{12}$$

which tends to infinity when $t \to \infty$.

There exists a classical workaround for this, which has already been mentioned in talk 3 as a "zero-redudancy estimator", let us recall it when predicting series of 0 and 1: instead of using only the θ computed with the virtual points, use a Bayesian mixture of P_θ, δ_0 and δ_1: this estimator will have exactly the same behavior as P_θ except when faced with a sequence of identical characters, and in that case, it will give a higher probability to the unique character of the sequence.

3.2.3 Bayesian case

Let us now show that the Bayesian prediction suffers from the same shortcoming: when the data follow a deterministic rule, its regret is not bounded.

The "plug-in" predictor is defined as follows:

1. Start with a prior distribution α on Θ
2. Compute the posterior probability of θ: $w(\theta) = \alpha(\theta) P_\theta(\mathcal{D}_t)$
3. Predict the missing data according to

$$P^\alpha(x_{t+1}|\mathcal{D}_t) = \frac{\int_\theta w(\theta) P_\theta(x_{t+1})}{\int_\theta w(\theta)} \tag{13}$$

Although this is usually hard to compute, we have the following result in the case of a finite alphabet:

Proposition 1. *Over a finite alphabet $\mathcal{A}$, $\theta = (f_A, f_B, ...,)$ with $\sum_{x\in\mathcal{A}} f_x = 1$:*

- *The uniform prior on θ is equivalent to adding virtual letters (each letter once),*
- *The Jeffreys' prior on θ is equivalent to adding virtual letters (each letter .5 times).*

Consequently, the Bayesian method also regularizes too much: for Jeffreys' prior, with equation (12), we obtain a regret approximately equal to $\frac{\dim(\Theta)}{2} \log_2 t$.

3.3 Model selection: the Bayesian Information Criterion

The two-parts code approach[4] is another way of obtaining this same regret, namely $\frac{\dim(\Theta)}{2} \log_2 t$, when encoding the optimal parameter value:

We have $K(\mathcal{D}) \leqslant \log_2 p_\theta(\mathcal{D}) + K(p_\theta)$, and $K(p_\theta) \leqslant K(p) + \text{cost of encoding } \theta$. The problem is that θ is a real number, so it cannot be encoded exactly. The optimal choice (see talk 2) is to encode θ up to $\sim \frac{1}{\sqrt{t}}$. Now the associated cost is $-\log_2 \frac{1}{\sqrt{t}}$ for each component, so the total cost of encoding θ is $\frac{\dim(\Theta)}{2} \log_2 t$.

This yields the so-called "Bayesian information criterion" (already mentioned in talk 1 and 2): when comparing models of different dimensions, $\theta_1^*, ..., \theta_n^*$, use the one maximizing $\log_2 p_{\theta_k^*}(\mathcal{D}) - \frac{\dim(\Theta_k)}{2} \log_2 t$. However, it can be argued that this model penalizes useless parameters too harshly.

Other penalties, which are not necessarily linked to information theory, exist. Let us for instance discuss the slope heuristics, introduced in [1].

[4] As defined in talk 2: encode the optimal θ for the data, and then, use P_θ to compress the data.

4 Slope heuristics

Suppose that we have a function f (e.g. f is a 2nd degree polynomial), and that we observe n points $y_i = f(x_i) + \mathcal{N}(0, \sigma^2)$ (in practice σ is not known, but it does not matter).

Let f_k be the best approximation of the $x_i \mapsto y_i$ by a polynomial of degree k.

The true error on the data is $e_k := \frac{1}{n} \sum \|f(x_i) - f_k(x_i)\|^2$, and the observed error is $\hat{e}_k := \frac{1}{n} \sum \|y_i - f_k(x_i)\|^2$ (we use the square error because the noise is Gaussian, and the log of a Gaussian is quadratic).

We use the following claim:

Claim 2. *If $k > \deg(f)$:*

- $\hat{e}_{k+1} \approx \hat{e}_k - \sigma^2/n$. *(the observed error decreases)*
- $e_{k+1} \approx e_k + \sigma^2/n$. *(the true error increases, which is normal, since we are fitting the noise)*

Now, the heuristics is the following:

1. Find λ such that $\hat{e}_k = C - \lambda k$ for large k (by the claim above $\lambda = \sigma^2/n$).

   By the claim above $\hat{e} + 2\lambda k$ has the same asymptotic behavior as e_k.

2. Select the value of k that minimizes $\hat{e} + 2\lambda k$.

References

[1] L. Birgé and P. Massart. Minimal penalties for gaussian model selection. *Probability Theory Related Fields*, 138:792–798, 2006.

[2] Peter D. Grünwald. *The Minimum Description Length Principle (Adaptive Computation and Machine Learning)*. The MIT Press, 2007.

[3] Ray J. Solomonoff. A formal theory of inductive inference. *Information and Control*, 7, 1964.