Contents

1 Sleeping

Data comes in a stream, say in different languages. We want an algorithm that learns English, learns another language, and when English comes back again, can learn it faster. All the old stuff it learns stay with it and it's easy to relearn. Humans are good at this.

This is a general natural phenomenon. If a bug specialized in a certain environment, if the environment reverts, old genes can be brought up again. Old knowledge lingers and and is quickly brought up again.

Changing one line of code can enable this.

I want sleeping for neural nets.

Some neurons freeze (learning rate decreased or zeroed), and wake up again (learning rate increased) later. Neurons can switch between these states.

We have a piece of intriguing theory. Bayesians will be upset but it works.

Note: does nearest neighbors do this? But label assignments can also change. Note you don't get feedback about there being a transition. You may have to remember all examples.

Use the disk spindown problem as an example (when to timeout laptop).

- experts: n fixed timeouts $\tau_1, \ldots, \tau_n$.
- Master algorithm maintains a set of weights $s_1, \ldots, s_n$, and predict with weighted average.
- Multiplicative update

$$s_{t+1,j} = \frac{s_{t,i} e^{-\eta(\text{energy usage of timeout } i)}}{Z}$$

A multiplicative update can pick out the current best and eliminate everyone else.

If idle times short, spin down after a longer time. If idle times long, spin down immediately. If we have a short period, a long period, then a short period...

Mix in a little bit of a uniform update to obtain uniform recovery.

$$s' = \text{(multiplicative update)} \tag{1}$$
$$s = (1-\alpha)s' + \alpha\left(\frac{1}{N}, \ldots, \frac{1}{N}\right) \tag{2}$$

where α is small. No expert goes to 0 so you can pick it up quickly again.

Regular multiplicative update learns the first language (English) too well and fails to learn anything else. "The curse of the multiplicative update." The multiplicative update with uniform update has a fixed adjustment time each time the language changes, but can learn different languages.

To get long-term, keep track of past average share vector r.

$$s' = \text{(multiplicative update)} \tag{3}$$
$$s = (1-\alpha)s' + \alpha r. \tag{4}$$

Related to momentum? We fixed $\alpha \approx 0.01$.

Wouter M. Koolen and Dmitry Adamskiy found a Bayesian interpretation.

Double track: We have a mixture of experts (RNA strands). Put γ fraction into alive track and $1-\gamma$ into asleep track. In the alive track, do a multiplicative update (selection and PCR). For the asleep track, don't do anything. Split up alive into $(1-\alpha, \alpha)$ alive/asleep, split up asleep into $(\beta, 1-\beta)$ alive asleep, and repeat. Initially,

$$s = \gamma \text{ initial} \tag{5}$$
$$r = 1-\gamma \text{ initial} \tag{6}$$
$$s^m = \text{MultiplicativeUpdate}(s) \tag{7}$$
$$r^m = r \tag{8}$$
$$s = (1-\alpha)s^m + \beta r^m \tag{9}$$
$$r = \alpha s^m + (1-\beta)r^m \tag{10}$$
$$\gamma = \frac{\beta}{\alpha+\beta}. \tag{11}$$

You can vary parameters dynamically.

If you know when the switches happen, loss is very low.

- Regular multiplicative update learns the first language (English) too well and fails to learn anything else (or it takes a long time).
- The multiplicative update with uniform update has a fixed adjustment time each time the language (best expert) changes, but can learn different languages (switch experts).
- Double track does even better.

I plotted logs of weights. Part of it is sent to the other track. When it's need again, it's pulled up faster.

If a weight does good, it leaks over to the sleeping track.

Would it benefit to have a third track? We can do everything continuously!

How can long-term memory be realized in neural nets? How is long-term memory realized in nature? Junk DNA, hidden genes? Sex?

When you throw wild seeds in your yard, part of it sprouts the first year, second year... It takes 7 years to all sprout. A farmer doesn't want this, wants monoculture, breeds out this variation. In time-varying weather, you want to guard against disasters, sleep some.

It could be that the third track handles switching small sets of pools.

Bio experiment: tubes with 10^{15} experts. You can do long-term memory with pouring. All the genes that were ever good can stick around.

2 Bayesian interpretation

2.1 Bayes

In the end we want to know how this scales up to NN. To get there we need some good perspectives. This is a second description.

I want to get to long-term memory, but I will first give you 2 things you need: Bayes Rule (learn a single best expert), specialists (not all experts make predictions all the time). Then you can build resilience, things that come back when they need to.

Let past outcomes be $y_1, y_2, \ldots$, models be $m \in M$, so we have probabilities

$$\mathbb{P}(y_t | y_{<t}, m).$$

Get all the pieces and form the joint. A Bayesian puts a **prior**

$$\mathbb{P}(m)$$

and makes a **joint** prediction

$$\mathbb{P}(y_{\leq t}, m) = \prod_{s=1}^{t} \mathbb{P}(y_s | y_{<s}, m).$$

Now **predict**

$$\sum_m \mathbb{P}(y_t | y_{<t}, m) \underbrace{\mathbb{P}(m | y_{<t})}_{\text{posterior}}.$$

The **update** is given by Bayes rule (cf. multiplicative update)

$$\mathbb{P}(m | y_{\leq t}) = \frac{\mathbb{P}(m | y_{<t}) \mathbb{P}(y_t | y_{<t}, m)}{\mathbb{P}(y_t | y_{<t})}.$$

In MU, write $\mathbb{P}(y_t | y_{<t}, m)$ as $e^{\log \text{ loss}}$.

$$\mathbb{P}(y_t | y_{<t}, m) = e^{-\eta l_t(m)}.$$

The regret bound is

$$\sum_{t=1}^{T} -\ln \mathbb{P}(y_t | y_{<t}) + \ln \mathbb{P}(y_t | y_{<t}, m) \leq -\ln \mathbb{P}(m)$$

Ex. for $\frac{1}{N}$, you get logarithmic in the number of experts. There is no dependence on time.

2.2 Specialists

Now I build on this. I want to deal with the case when not all the models are giving a prediction, $\mathbb{P}(y_t|y_{<t}, m)$ might be missing. They are not participating. (A specialist is a model whose predictions might be missing. "I don't know about this area, I abstain.") It's not a partial observation.

How do we deal with this?

Probabilities are available only for $m \in W_t$.

Make up probabilities? Ignore what's missing?

Now there's two types of models: the ones that are there and not.

$$\mathbb{P}(y_t|y_{<t}) = \sum_{m \in W_t} \mathbb{P}(y_t|y_{<t}, m)\mathbb{P}(m|y_{<t}) + \sum_{m \notin W_t} \mathbb{P}(y_t|y_{<t})\mathbb{P}(m|y_{<t}). \tag{12}$$

Plug in your own prediction for the missing predictions. "If you're silent you must agree." This is the same as ignoring and normalizing. Solve for the probability:

$$\mathbb{P}(y_t|y_{<t}) = \frac{\sum_{m \in W_t} \mathbb{P}(y_t|y_{<t}, m)\mathbb{P}(m|y_{<t})}{\sum_{m \in W_t} \mathbb{P}(m|y_t)}.$$

This means I get the Bayesian regret bound for free. When the model is asleep, I assigned my own prediction; there is no contribution to the regret model.

$$\sum_{t:m \in W_t} -\ln \mathbb{P}(y_t|y_{<t}) + \ln \mathbb{P}(y_t|y_{<t}, m) \leq -\ln \mathbb{P}(m).$$

(If no experts are awake, it doesn't matter what we do: if we have no prediction at all, we have no yardstick to evaluate.)

We are plugging in the prediction as a likelihood.

This also works with an infinite number of model; there is another model that was asleep so far. (But you need a prior for models.)

2.3 Resilience

You need a way to deal with the fact that there is a model that is good on the second half of the data. You need an equal amount of goodness to compensate for the badness to even bring it back to the starting point.

We want a model to join in at the time that it's useful. Let's design specialists that do this for us. Let's enrich the model class.

$$M^{res} = \{(m, s) : m \in M, s \in \mathbb{N}\}$$

Let

$$N_t = \{(m, s) : s \leq t\}.$$

Here (m, s) predicts like m. Let

$$\mathbb{P}(m, s) = \mathbb{P}(s)\mathbb{P}(m).$$

Let $\mathbb{P}(s) = \frac{1}{T}$.

My loss from s to T is given by

$$-(\text{loss against } m) \leq -\ln \mathbb{P}(m) + \ln T.$$

This is fixed share: mixing against uniform.

There are only those that have woken up and are still asleep. If I keep on playing, it doesn't break my regret at time T! I can sum them to cover all my data.

2.4 Long-term memory

The experts need to participate, go away, and come back.

We want experts of the form sssswwwsssswww....

Let

$$M^{LTM} = M \times \{w, s\}^{\mathbb{N}}.$$

We have $(m, \overline{a}) \in W_t$ iff $a_t = w$.

The prior is

$$\mathbb{P}(m, \overline{a}) = \mathbb{P}(m)\text{Markov}(\overline{a}).$$

We want a stretch of asleep/awake. The more switches, the more complicated. The most reasonable are the sticky sequences. There are 2 states in the Markov process: given I'm awake I have high likelihood of being awake. Given I'm asleep I have a high likelihood of staying asleep. There is a small probability of changing tracks.

This comes back in the regret bound.

You can group everything that's awake/asleep. That's exactly the two tubes.

The regret depends on $\mathbb{P}(m)$: Roughly you pay $\ln T$ for every switch. You can tune the prior to match the frequency. You can make $\frac{1}{T}$ for switching: smaller probability of switching as time goes on. This is appropriate for meteorites, not seasons.

Going to 0 at known rate is close to knowing the proportion given it's sublinear.

If you have hard-core loss there are standard reductions. It's orthogonal.

Finding the best sequence in hindsight is NP-hard. Can I make the best such partition?

The model can compete with all partitions.

Use sleeping to avoid putting prior on partitions and trying to solve a NP-hard problem? (This is not Markov: you want to do well with models that have appeared in the past, which corresponds to models in the past being more likely to recur. A Bayesian would use a Dirichlet process prior, but this is intractable to do inference on.)

Can you equate "sleeping" with wildcard? No, you have to produce a likelihood.

From a Bayesian perspective it is disturbing, but it works. If you imagine any predictions it becomes fully defined.

When something is asleep keep on going. You have to cover your timeline.

You have 2 modes of using it: the sleeping things are real, or build your own model where they are defined by you: virtual experts, you define where they are sleeping.

2.5 Neural networks

For a single neuron, once in a while you want to freeze it, and the whole graph underneath. How to freeze and wake up again? Put a new feature.

Problem: the layers depend on each other.

The other question: maybe there is another track where you leak; it goes at a slower rate. Maybe you can do it hierarchically, and show English a small pool, German is another pool, and you shift between pools. With very little additional mechanism (in the single neuron case), have layered long-term memory, layered time scales.

How about continuous things? The math goes through if you have partial awake/asleep. As you vary "asleep" parameter, you are more asleep.

We had fixed sleep/awake factors. You can learn those. One step do likelihoods, the next step get feedback, have heuristic for being awake or asleep. This is allowed for in Bayesian thinking. If you have likelihood to waking up/falling asleep, the two things interleave.

References

[AWK12] Dmitry Adamskiy, Manfred K Warmuth, and Wouter M Koolen. "Putting Bayes to sleep". In: *Advances in Neural Information Processing Systems*. 2012, pp. 135–143.

[BW02] Olivier Bousquet and Manfred K Warmuth. "Tracking a small set of experts by mixing past posteriors". In: *Journal of Machine Learning Research* 3.Nov (2002), pp. 363–396.

[Fre+97] Yoav Freund et al. "Using and combining predictors that specialize". In: *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*. ACM. 1997, pp. 334–343.