Chapter 12

Learning with the GLM

12.1 Introduction on learning concepts

See learning concepts in Ch 13 of (Barber 2012):

- supervised learning $p(y|x;\theta)$
- reward-based learning $\langle R \rangle$
- unsupervised learning $p(y|\theta)$

Other forms exists such as active learning or semi-supervised learning. In this section, we will see that the learning rule in the different scenarios are very similar and do have some biological support.

12.2 Note on log-concavity of the PDF

Let L be the log-likelihood defined as the logarithm of the probability density function given in Eq. (11.9), i.e.

$$L(\theta) = \log p(y^T) = \int_0^T \log\left(g_{\theta_1}(u_{\theta_2}(t))\right) y(t) - g_{\theta_1}(u_{\theta_2}(t))dt \tag{12.1}$$

where $\theta = (\theta_1, \theta_2) \in \Theta$ are the parameters of the model. How to guarantee that $L(\theta)$ does not have local maxima? A sufficient condition is that L is concave (see Fig. 12.1)

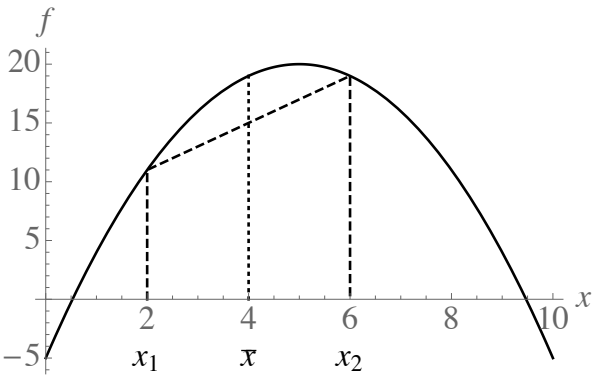

Figure 12.1: For a concave function, $f(\bar{x}) \geq (1-\alpha)f(x_1) + \alpha f(x_2)$ where $\bar{x} = (1-\alpha)x_1 + \alpha x_2$.

Definition 12.1. *C is a convex set if $(1-\alpha)x + \alpha y \in C$ $\forall x, y \in C$ and $\forall \alpha \in [0,1]$.*

Definition 12.2. *A function $f : C \subset \mathbb{R} \to \mathbb{R}$ is concave on the (convex) set C if*

$$f((1-\alpha)x_1 + \alpha x_2) \geq (1-\alpha)f(x_1) + \alpha f(x_2) \quad \forall x_1, x_2 \in C \quad \alpha \in [0,1] \tag{12.2}$$

f is strictly convex if

$$f((1-\alpha)x_1 + \alpha x_2) > (1-\alpha)f(x_1) + \alpha f(x_2) \quad \forall x_1, x_2 \in C \quad \alpha \in [0,1] \tag{12.3}$$

Property 12.1. *A twice differentiable function f is concave iff $f''(x) \leq 0$, $\forall x \in C$.*

Proof 12.1. *Let's prove $"f(x)$ concave $\Rightarrow$ $f''(x) < 0"$. Let $\alpha = 0.5$, $x_1 = x - \Delta x$ and $x_2 = x + \Delta x$. From concavity, we have*

$$f((1-\alpha)x_1 + \alpha x_2) = f(x) \geq \frac{1}{2}f(x+\Delta x) + \frac{1}{2}f(x-\Delta x) \tag{12.4}$$

Hence

$$f''(x) = \lim_{\Delta x \to 0} \frac{1}{\Delta x^2}\left(f(x+\Delta x) - 2f(x) + f(x-\Delta x)\right) \leq 0 \tag{12.5}$$

Property 12.2. *Concavity is preserved under addition, i.e. f_1, f_2 are concave, then $f_1 + f_2$ is concave.*

Proof 12.2. *Since $f_1''(x) \leq 0$ and $f_2''(x) \leq 0$ (from Property 12.1), we have $f_1''(x) + f_2''(x) \leq 0$. Therefore $f_1 + f_2$ is concave.*

Property 12.3. *If $f(x)$ is concave, then $f(\alpha x)$ is also concave.*

Proof 12.3. *Indeed, with $y = \alpha x$*

$$\begin{aligned}\frac{d^2 f}{dx^2} &= \frac{d}{dx}\left(\frac{df}{dy}\frac{dy}{dx}\right) = \frac{d}{dx}\left(\frac{df}{dy}\right)\frac{dy}{dx} + \frac{df}{dy}\frac{d^2y}{dx^2} \\ &= \frac{d^2 f}{dy^2}\underbrace{\left(\frac{dy}{dx}\right)^2}_{=\alpha^2>0} + \underbrace{\frac{df}{dy}\frac{d^2y}{dx^2}}_{=0} \leq 0\end{aligned}$$

Question: Let us assume that u_{θ_2} is linear in θ_2, what are then the conditions on the transfer function g?

1. $\log g_{\theta_1}(u)$ is jointly concave in u and θ_1 (c.f. first term in Eq. (12.1))
2. $g_{\theta_1}(u)$ is jointly convex in u and θ_1 (c.f. second term in Eq. (12.1))
3. Θ is convex

So practically?

- g has to be positive (by construction, it is a probability density)
- g' is monotonically increasing functions. From condition 2, we have $0 \leq g''$ and from condition 1, we have $g'' \leq (g')^2/g$. Therefore $0 \leq g'' \leq (g')^2/g$.
- g grows at least linearly (from cond. 2) and at most exponentially (from cond. 1)
- additional subtleties c.f. (Paninski et al. 2004)

Example 1. *Typical functions that satisfy those conditions*

- *exponential*
- *rectified linear*

Counter-example 1. *Typical functions that do not satisfy those conditions*

- *sigmoidal*
- *quadratic*

12.3 Supervised learning

12.3.1 From the neuron perspective

Let us consider a feedforward network with M input neurons and N output neurons. Input spike trains are denoted as $\mathbf{x} = (x_1, \dots, x_M)$ and output spike trains as $\mathbf{y} = (y_1, \dots, y_N)$. Let us further assume that during training the output neuron is forced to produce spike trains $\mathbf{y}$ for a corresponding $\mathbf{x}$ which are drawn from a teacher distribution $p^*(\mathbf{y}|\mathbf{x})$. Practically those spikes could be forced through some strong input current that comes from other neurons.
The learning task can be formulated as minimizing the KL divergence between the target distribution $p^*(\mathbf{y}|\mathbf{x})$ and the distribution $p_\theta(\mathbf{y}|\mathbf{x})$ produced by the network averaged over the distribution $p(\mathbf{x})$ of input spike trains:

$$\begin{aligned} \overline{D}_{\mathrm{KL}} &= \langle D_{\mathrm{KL}}(p^*(\mathbf{y}|\mathbf{x}) \| p_\theta(\mathbf{y}|\mathbf{x})) \rangle_{p(\mathbf{x})} \\ &= \sum_{\mathbf{x}} p(\mathbf{x}) \sum_{\mathbf{y}} p^*(\mathbf{y}|\mathbf{x}) \log \frac{p^*(\mathbf{y}|\mathbf{x})}{p_\theta(\mathbf{y}|\mathbf{x})} \end{aligned} \tag{12.6}$$

where θ are the network parameters (such as the connectivity matrix).

One possibility is to perform a gradient descent (even though gradient decent has some pathologies)

$$\begin{aligned} \Delta\theta &= -\alpha \nabla_\theta \overline{D}_{\mathrm{KL}} \\ &= \alpha \left\langle \nabla_\theta \log p_\theta(\mathbf{y}|\mathbf{x}) \right\rangle_{p^*(\mathbf{y}|\mathbf{x})p(\mathbf{x})} \\ &= \alpha \left\langle \sum_{i=1}^{N} \int_0^T \frac{g_i'(t)}{g_i(t)} (y_i(t) - g_i(t)) \nabla_\theta u_i(t) dt \right\rangle_{p^*(\mathbf{y}|\mathbf{x})p(\mathbf{x})} \end{aligned} \tag{12.7}$$

where $g_i'(t) = dg(u)/du|_{u=u_i(t)}$ and $g_i(t) = g(u_i(t))$ and α is the learning rate. Let us now consider the (online) learning rule for the synaptic weight w_{ij}:

$$\Delta w_{ij} = \alpha \frac{g_i'(t)}{g_i(t)} (y_i(t) - g_i(t))(\epsilon_{ij} * x_j)(t) \tag{12.8}$$

This learning rule is very interesting for several reasons

- it is local
- it is online
- it closely matches experimental on STDP. See Fig. 12.2.
- it matches 2 fields of research. Statistical learning and synaptic plasticity

Exercise 12.1. *Derive the synaptic plasticity learning for neuron model which is given with a differential equation of the type $\dot{u} = f(u, w)$ where f is linear in u and in w.*

Solution 12.1.

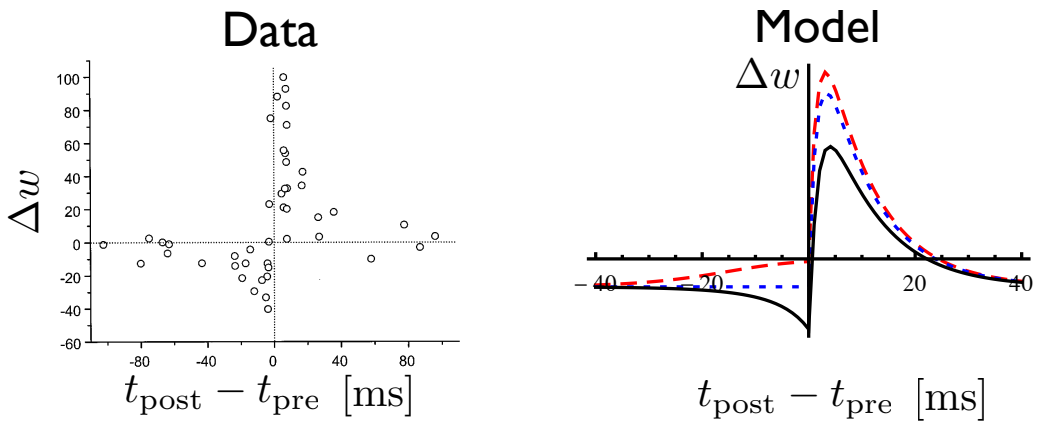

Figure 12.2: Spike-Timing Dependent Plasticity data from Bi and Poo, 1998 (left) and predicted by Eq. (12.8) for various spike after-potential kernels η.

12.3.2 From the experimenter perspective

Let us know consider the case where a set of neurons are recorded. Those neurons generate spike trains $\mathbf{y}$ in response to stimuli I. At this stage several questions can be asked

- the encoding question: given a new stimulus, can I predict the corresponding spike train?
- the decoding question: given a recorded spike trains, can I estimate what was the stimulus that generated them?

Here we will answer only the first question. Mathematically, this problem is analogous to the supervised learning problem for sSRM neurons that we considered so far, i.e. we replace x by I. Every neuron has its own receptive field κ_i (see Eq. (11.23)) which can be written as a sum of basis functions $b_j(t)$

$$\kappa_i = \sum_j k_{ij} b_j(t) \tag{12.9}$$

The task is to minimize

$$\overline{D}_{\text{KL}} = \langle D_{\text{KL}}(p^*(\mathbf{y}|I) \| p_\theta(\mathbf{y}|I)) \rangle_{p(I)} \tag{12.10}$$

with respect to all model parameters θ. Those parameters now include w_{ij} as well as $k_i j$. This gives

$$\Delta\theta = \alpha \left\langle \sum_{i=1}^{N} \int_0^T \frac{g_i'(t)}{g_i(t)} (y_i(t) - g_i(t)) \nabla_\theta u_i(t) dt \right\rangle_{p^*(\mathbf{y}|I)p(I)} \tag{12.11}$$

We can use this equation to express the learning rule for the weight matrix w. The result is identical to the one obtained previously by Eq. (12.8). Learning the receptive fields coefficients k_{ij} gives

$$\Delta k_{ij} = \alpha \frac{g_i'(t)}{g_i(t)} (y_i(t) - g_i(t))(b_j * I)(t) \tag{12.12}$$

12.4 Reward-based learning with spiking neurons

Learning from reward or punishment is a challenging task for several reasons

1. **High dimensionality problem**. The sensory inputs to an agent are of high dimensions, so it is hard to know which part of the input is responsible for the reward. A lot of algorithms assume that the agent is in a given state, but this is already a massive simplification as the "state identification" is not a trivial problem. This high-dimensionality problem is often termed as the *spatial-credit assignment problem*. Furthermore, it is often the case that the reward does not come right after the action has been taken, but after a possibly long and variable delay. Example: In response to a tough lecture (input), a student works hard (action) and gets a good mark (reward) few months after. This is called the *temporal-credit assignement problem*. Overall this is termed as *the spatio-temporal credit assignment problem*. .
2. The reward function can be pretty bad containing several local minima and could be non-smooth - which makes the problem particularly hard

Let us now make the scenario more concrete. A set of input neurons with activity $\mathbf{x}$ stimulate a recurrent population of neuron with activity $\mathbf{y}$. The environment rewards the network for certain combinations of $\mathbf{x}$ and $\mathbf{y}$, i.e. $R(\mathbf{x},\mathbf{y})$. The task is here to maximize the expected reward

$$\langle R\rangle = \sum_{\mathbf{x},\mathbf{y}} R(\mathbf{x},\mathbf{y}) p_\theta(\mathbf{x},\mathbf{y}) \tag{12.13}$$

Note that here the stochasticity plays a crucial role. It smooths out the possibly non-smooth reward function R and makes it differentiable.
The learning rule can be written as

$$\begin{aligned}\Delta\theta &= \alpha\nabla_\theta \langle R\rangle \\ &= \sum_{\mathbf{x},\mathbf{y}} R(\mathbf{x},\mathbf{y})\nabla_\theta \log p_\theta(\mathbf{x},\mathbf{y}) p_\theta(\mathbf{x},\mathbf{y}) \\ &= \alpha\left\langle (R-\bar{R})\nabla_\theta \log p_\theta(\mathbf{y}|\mathbf{x})\right\rangle \end{aligned} \tag{12.14}$$

Exercise 12.2. *Convince yourself that $\bar{R}$ can be arbitrary*

Solution 12.2.

Exercise 12.3. *Calculate the optimal $\bar{R}$ such that the variance of the estimator is minimized.*

Solution 12.3.

So, for the synaptic weights, the corresponding online learning rule yields

$$\Delta w_{ij} = \alpha(R-\bar{R})\frac{g_i'(t)}{g_i(t)}(y_i(t)-g_i(t))(\epsilon_{ij} * x_j)(t) \tag{12.15}$$

Note again that this learning rule is very similar to the one obtained in Eq. (12.8).

Exercise 12.4. *Calculate the reward-based learning rule for a network of rate-based neurons, i.e. $\mathbf{y} = \mathbf{w}\mathbf{x} + \epsilon$ where $\epsilon_j \sim \mathcal{N}(x;0,\sigma^2)$*

Solution 12.4.