ECE-GY 9223 Reinforcement Learning Spring 2019

Lecture 5 — March 1st, 2019

Prof. Quanyan Zhu *Scribe: Muhammad Affan Javed*

1 Overview

In the last lecture we discussed Policy Iteration, introduced neural networks and applied Pontryagin's Maximum Principle to solve the optimal control problem derived from neural networks. In this lecture, we expand our discussion of neural networks by investigating how **Backpropagation** is used in neural networks to calculate the gradients at different layers. We also review **Numerical Algorithms**, specifically **Gradient Descent** and its variations. Finally, we introduce **Stochastic Approximation**.

2 Neural Networks: Backpropagation

We introduced neurons and multi-layer neural networks in the last lecture. Fig.1 shows a single neuron, which is a building block for a neural network. Fig.2 shows a multilayer neural network comprising of multiple layers, each with multiple neurons.

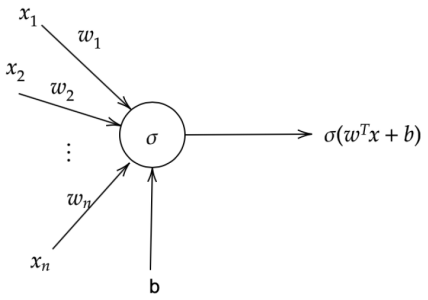

Figure 1: A Single Neuron

The output of a neuron at a given layer can be expressed in the following manner:

$$\begin{aligned} y_1 &= \sigma(w_1 x + b_1) \\ y_2 &= \sigma(w_2 y_1 + b_2) \\ &\vdots \\ y_L &= \sigma(w_L y_{L-1+b_L}) \end{aligned}$$

Moreover, we define:

$$z_l = w_l y_{l-1} + b_l$$

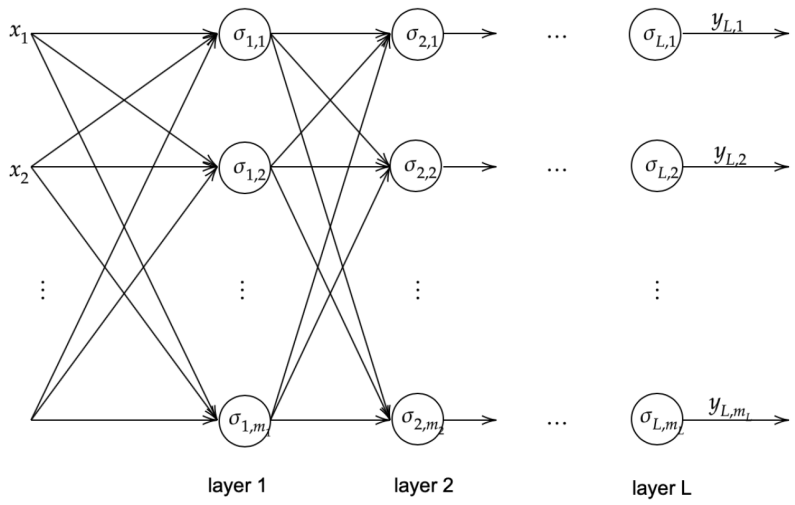

Figure 2: A Multilayer Neural Network

Thus, we can re-write the output of each layer l as $y_l = \sigma(z_l)$

We apply gradient descent on the weights $\{w_L, b_L\}$:

$$w_l(t+1) = w_l(t) - \epsilon_t \nabla_{w_l} J_L \tag{1}$$
$$b_l(t+1) = b_l(t) - \epsilon_t \nabla_{b_l} J_L \tag{2}$$

Observation 1. *Note here that (1) and (2) are not convex in general because σ is not convex.*

To find out the gradient of one data point, we start at the last layer L and apply the chain rule:

$$\begin{aligned}\frac{\partial J_L}{\partial w_{L,ij}} &= \frac{\partial J_L}{\partial y_{L,i}} . \frac{\partial y_{L,i}}{\partial w_{L,ij}} \\ &= \frac{\partial J_L}{\partial y_{L,i}} . \frac{\partial y_{L,i}}{\partial z_{L,i}} . \frac{\partial z_{L,i}}{\partial w_{L,ij}}\end{aligned}$$

However, $\frac{\partial z_{L,i}}{\partial w_{L,ij}} = y_{L-1,j}$ and $\frac{\partial y_{L,i}}{\partial z_{L,i}} = \sigma'(z_L, i)$. So:

$$\frac{\partial J_L}{\partial w_{L,ij}} = \frac{\partial J_L}{\partial y_{L,i}} . \sigma'(z_{L,i}) . y_{L-1,j}$$

Similarly, for b_L:

$$\begin{aligned}\frac{\partial J_L}{\partial b_L, i} &= \frac{\partial J_L}{\partial y_{L,i}} . \frac{\partial y_{L,i}}{\partial z_{L,i}} . \frac{\partial z_{L,i}}{\partial b_{L,i}} \\ &= \frac{\partial J_L}{\partial y_{L,i}} . \sigma'(z_{L,i})\end{aligned}$$

Observation 2. *The gradient at layer L depends on the layer L-1. Generally, the gradient at layer l depends on the layer $l-1$.*

At layer l, $\quad 1 \leq l < L$

$$\begin{aligned}\frac{\partial J_l}{\partial w_{l,ij}} &= \frac{\partial J_l}{\partial y_{l,i}} . \frac{\partial y_{l,i}}{\partial z_{l,i}} . \frac{\partial z_{l,i}}{\partial w_{l,ij}} \\ &= \frac{\partial J_l}{\partial y_{l,i}} . \sigma'(z_{l,i}) . y_{l-1,j}\end{aligned}$$

$$\boxed{\frac{\partial J_l}{\partial w_{l,ij}} = \frac{\partial J_l}{\partial y_{l,i}} . \sigma'(z_{l,i}) . y_{l-1,j}} \tag{3}$$

The first term in (3) can be computed as:

$$\begin{aligned}\frac{\partial J_l}{\partial y_{l,i}} &= \sum_k \frac{\partial J_l}{\partial z_{l+1,k}} . \frac{\partial z_{l+1,k}}{\partial y_{l,i}} \\ &= \sum_k \frac{\partial J_l}{\partial y_{l+1,k}} . \frac{\partial y_{l+1,k}}{\partial z_{l+1,k}} . \frac{\partial z_{l+1,k}}{\partial y_{l,i}} \\ &= \sum_k \frac{\partial J_l}{\partial y_{l+1,k}} . \sigma'(z_{l+1}, k) . w_{l+1,ki}\end{aligned}$$

$$\boxed{\frac{\partial J_l}{\partial y_{l,i}} = \sum_k \frac{\partial J_l}{\partial y_{l+1,k}} . \sigma'(z_{l+1}, k) . w_{l+1,ki}} \tag{4}$$

Similarly,

$$\begin{aligned}\frac{\partial J_l}{\partial b_{l,i}} &= \frac{\partial J_l}{\partial y_{l,i}} . \frac{\partial y_{l,i}}{\partial z_{l,i}} . \frac{\partial z_{l,i}}{\partial b_{l,i}} \\ &= \frac{\partial J_l}{\partial y_{l,i}} . \sigma'(z_{l,i})\end{aligned}$$

$$\boxed{\frac{\partial J_l}{\partial b_{l,i}} = \frac{\partial J_l}{\partial y_{l,i}} . \sigma'(z_{l,i})} \tag{5}$$

Now, we have all the tools we need to compute the gradients at all the layers. We can do this by using the following steps:

Step 1: Compute $\frac{\partial J_L}{\partial y_{L,i}}$ for the last layer L.

Step 2: Use (3),(4) and (5) to compute the gradients for layer L down to layer $l = 1$.

3 Numerical Algorithms

Consider a continuously differentiable function $f : \mathbb{R}^n \to \mathbb{R}$. The unconstrained minimization of f relies on an important idea called ***iterative gradient descent***. Starting at some initial point r_0

(an initial guess, which can be random), successive vectors are generated according to the following update rule:

$$r_{t+1} = r_t + \gamma_t s_t \qquad t = 0, 1, \cdots$$

where γ_t is the step size (usually positive) and s_t is the descent direction, i.e.,:

$$\nabla^T f(r_t) s_t < 0$$

assuming that $\nabla f(r_t) \neq 0$. The termination condition for the gradient descent algorithms is $\nabla f(r_t) = 0$, which implies that $r_{t+1} = r_t$. There is a large number of ways in which the descent direction, s_t, and step size, γ_t, can be chosen. The following sub-sections discuss some examples of these.

3.1 Examples of Descent Directions s_t

1. **Steepest Descent:**

   Here

   $$s_t = -\nabla^T f(r_t)$$

   Steepest descent is simple, but suffers from slow convergence.

2. **Newton's Method:**

   Here

   $$s_t = -(\nabla^2 f(r_t))^{-1} \nabla f(r_t)$$

   Newton's method maximizes the following quadratic approximation of f around r_t:

   $$\tilde{f}_t(r) = f(r_t) + \nabla^T f(r_t)(r - r_t) + \frac{1}{2}(r - r_t)^T \nabla^2 f(r_t)(r - r_t)$$

   Newton's method finds the global minimum of a positive definite quadratic function in a single iteration (assuming $\gamma_t = 1$). More generally, Newton's method typically converges very fast asymptotically. However, on the other hand, it can be computationally expensive as it requires computing the Hessian matrix and solving a linear system of equations in order to find the Newtonian direction.

3. **Quasi-Newton Method:**

   The Quasi-Newton Method attempts to significantly lower the computational cost of Newton's Method by replacing the inverse of the Hessian with another matrix, D_t, which is a positive definite symmetric matrix. The step-size is given by:

   $$s_t = -D_t \nabla f(r_t)$$

   D_t is an important design parameter and must be carefully chosen in order to achieve fast convergence with low computational complexity. Usually, D_t is chosen to be an approximation of the inverse Hessian $(\nabla^2 f(r_t))^{-1}$. Generally, for non-quadratic functions, Quasi-Newton methods tend to achieve in part the fast convergence rate of Newton's method without incurring the extra overhead for calculating the Newtonian direction.

3.2 Examples of Step-Size Functions γ_t

1. **Constant Step Size:**

$$\gamma_t = \gamma \qquad t = 0, 1, 2, \cdots$$

The step size is the same constant at each time-step. It is immediately clear that choosing the step size such is not good: if γ is too large, the algorithm will miss the optimal point; on the other hand, if γ is too small the algorithm will converge very slowly.

In order to prove convergence, we first impose the following conditions:

$$c_1 \|\nabla f(r_t)\|^2 \leq -\nabla^T f(r_t) s_t \qquad \forall t \tag{6}$$

$$\|s_t\| \leq c_2 \|\nabla f(r_t)\| \tag{7}$$

where c_1, c_2 are some positive scalars.

Theorem 1 (**Convergence for Constant Step Size**). *Let r_t be a sequence generated by a gradient method: $r_{t+1} = r_t + \gamma s_t$, where s_t satisfies (6) and (7). Assume that for some $L > 0$, we have:*

$$\|\nabla f(r) - \nabla f(\bar{r}\| \leq L \|r - \bar{r}\| \qquad \forall r, \bar{r} \in \mathbb{R}^n$$

and

$$0 < \gamma < \frac{2c_1}{Lc_2^2}$$

Then, either $f(r_t) \to -\infty$, or $f(r_t)$ converges to a finite value:

$$\lim_{t\to\infty} \nabla f(r_t) = 0$$

Furthermore, every limit point of r_t is a stationay point of f.

2. **Minimization Rule:**

Another way of choosing the step size is to assign the step size a value such that f is minimized along the direction s_t. This can be expressed formally as:

$$f(r_t + \gamma_t s_t) = \min_{\gamma \geq 0} f(r_t + \gamma s_t)$$

Although choosing the step-size in such a manner leads to faster convergence, it can be quite costly in terms of computation. This is because the algorithm needs to do a blind search for finding the optimal value of γ at every step.

3. **Diminishing Step Size:**

A diminishing step-size is one which satisfies the following condition:

$$\gamma_t \to 0$$

However, we impose another condition as well, i.e. the step size can not diminish too quickly. This condition can be expressed formally as:

$$\sum_{t=0}^{\infty} \gamma_t = \infty$$

This condition is imposed in order to deal with the convergence issue. If the step-size diminishes too quickly, the algorithm may stop at an in-optimal point.

3.3 Gradient Methods with Noise

We now extend our discussion of gradient methods in the presence of noise or errors. The noise, denoted by w_t, can be formally incorporated into the framework in the following manner:

$$\begin{aligned} r_{t+1} &= r_t + \gamma_t(s_t) \\ &= r_t - \gamma_t(\nabla f(r_t) + w_t) \end{aligned}$$

The following subsections discuss various types of noise and the conclusions regarding convergence and stability of the gradient descent algorithms that we can consequently draw from them.

3.3.1 Noise (w_t) is small relative to the gradient

Mathematically, this condition can be written as:

$$\|w_t\| < \|\nabla f(r_t)\| \qquad \forall t$$

Assuming $\nabla f(r_t) \neq 0$, s_t is still a direction of descent. This can be shown in the following manner:

$$\begin{aligned} \nabla^T f(r_t) s_t &= \nabla^T f(r_t)(-\nabla f(r_t) - w_t) \\ &\leq -\|\nabla f(r_t)\|^2 + \|\nabla f(r_t)\| \, \|(\| \, w_t) \\ &< 0 \end{aligned}$$

3.3.2 Noise (w_t) is bounded

This is a more general case where w_t is bounded by some δ, and not necessarily the gradient. This can be written as:

$$\|w_t\| < \delta \qquad \text{for some} \;\; \delta > 0$$

It can easily be shown that such a generic bound is problematic. Specifically, the algorithm does not converge to the right point.

Let $w_t = w$. Then, $r_{t+1} = r_t - \gamma_t(\nabla f(r_t) + w)$ Consequently, we need the noise to be diminishing in order for the algorithm to converge to the optimal point.

3.3.3 Noise (w_t) is proportional to the step-size γ_t

Here,

$$\|w_t\| \leq q\gamma_t, \quad \forall t$$

where q is some scalar.

For convergence, it is sufficient to have for some positive scalars p and q,

$$\|w_t\| \leq \gamma_t(q + p\,\|\nabla f(r_t)\|), \quad \forall t$$

3.3.4 Noise (w_t) is stochastic

An example of stochastic noise is when w_t are independent zero mean random vectors with finite variance. An important special case is when f is of the form:

$$f(r) = \mathbb{E}_v(F(r, v))$$

where $F : \mathbb{R}^{m+n} \mapsto \mathbb{R}$ is a continuously differentiable function, v is a random vector in $\mathbb{R}^m$, and $\mathbb{E}_v[.]$ denotes the expected value with respect to v. Then:

$$\begin{aligned} \nabla f(r) &= \nabla \mathbb{E}_v[F(r, v)] \\ &= \mathbb{E}_v[\nabla_r F(r, v)] \end{aligned}$$

The gradient descent equation can then be written as:

$$\begin{aligned} r_{t+1} &= r_t + \gamma_t s_t \\ &= r_t - \gamma_t \mathbb{E}_v[\nabla_r F(r, v)] \end{aligned} \tag{8}$$

Observation 2. *If we know the distribution of v, we can solve (8) in closed form.*

However, generally we do not know the distribution of v. In such cases, an approximation is computed by using a limited number of samples of $\nabla_r F(r_t, v)$. In the extreme case, we have only one sample v_t:

$$s_t = -\nabla_r F(r_t, v_t)$$

Then, the error (or noise) is:

$$\begin{aligned} w_t &= \nabla_r F(r_t, v_t) - \nabla f(r_t) \\ &= \nabla_r F(r_t, v_t) - \mathbb{E}_v[\nabla_r F(r_t, v)] \end{aligned} \tag{9}$$

The gradient descent equation then becomes:

$$r_{t+1} = r_t - \gamma_t[\mathbb{E}_v[\nabla_r F(r, v)] + w_t] \tag{10}$$

where w_t is the zero-mean noise given in (9).

Observation 3. *It can be seen that (10) is a noisy version of (8), where the noise arises due to the fact that a single sample v_t is used in the approximation.*

Moreover, the noise w_t need not diminish with $\|\nabla f(r_t)\|$ but under appropriate conditions its effects are "averaged out". What is happening here is that the descent condition $\nabla^T f(r_t) s_t < 0$ holds **on the average** on non-stationary points of $f(r_t)$. With a diminishing step-size, the occasional use of a *bad* direction will not cause the algorithm to oscillate, given that on *average* the algorithm uses *good* directions.

4 Stochastic Approximation

Now we expand the discussion of gradient methods with noise to a more general setting. Suppose we are interested in solving for r an equation of the form:

$$\mathbb{E}_v[g(r, v)]$$

where v is a random variable and g is a known function. One possibility is to use a determinsitic algorithm:

$$r_{t+1} = (1 - \gamma)r_t + \gamma\mathbb{E}_v[g(r_t, v)] \tag{11}$$

If it converges, it converges to a fixed point. This can be shown in the following manner. Convergence implies $r_{t+1} = r_t$. Then:

$$\begin{aligned} r_t &= (1 - \gamma)r_t + \gamma\mathbb{E}_v[g(r_t, v)] \\ \gamma r_t &= \gamma\mathbb{E}_v[g(r_t, v)] \\ r_t &= \mathbb{E}_v[g(r_t, v)] \end{aligned}$$

$$\boxed{r = \mathbb{E}_v[g(r, v)]}$$

Instead of directly computing the expectation $\mathbb{E}_v[g(r, v)]$, which is a multi-dimensional integral and hence computationally difficult to solve, we can estimate $\mathbb{E}_v[g(r, v)]$ by generating random samples $\tilde{v}$ of v. For example, we can generate k samples and carry out an update using the sample mean:

$$\frac{1}{k}\sum_{i=1}^{k} g(r, \tilde{v}_i)$$

As k becomes large, the sample mean converges to the true mean and we recover the deterministic algorithm given in (11).

At the other extreme is what is known as the **Robbins-Monro stochastic approximation algorithm**. Here, we let $k = 1$ and base an update on a single sample $\tilde{v}$:

$$\begin{aligned} r_{t+1} &= (1 - \gamma)r_t + \gamma g(r, \tilde{v}) \\ &= (1 - \gamma)r_t + \gamma(\mathbb{E}[g(r_t, \tilde{v}] + g(r_t, \tilde{v} - \mathbb{E}[g(r_t, \tilde{v}])) \\ &= (1 - \gamma)r_t + \gamma(\mathbb{E}[g(r_t, \tilde{v}] + w_t) \end{aligned}$$

$$\boxed{r_{t+1} = (1 - \gamma)r_t + \gamma(\mathbb{E}[g(r_t, \tilde{v}] + w_t)} \tag{12}$$

Observation 4. *We can leverage the Robbins-Monro algorithm to compute the fixed point equation without knowing the underlying distributions.*

Observation 5. *(12) looks similar to the Bellmann equation. Maybe we can use this for Dynamic Programming equations.*

In general:

$$x_{n+1} = x_n + \epsilon_n(h(x_n) + \mu_n) \tag{13}$$

Here, x_0 is given, $x_n \in \mathbb{R}^d$, $h(.) : \mathbb{R}^d \mapsto \mathbb{R}^d$ and μ_n is noise.

Now consider the following ODE:

$$\dot{x} = h(x) \tag{14}$$

where x_0 is given.

Theorem 6. *If ODE admits a continuously differentiable Lyapanov function, then it implies the convergence of the stochastic iteration.*

The roadmap for proving this theorem can be encapsulated in the following ideas:

1. Show that the general trajectory for the ODE and the interpolated trajectory of $\{x_n\}$ remain arbitrarily close.
2. Show that there is a Lyapanov function that allows the ODE to go to the stationary point.

Next, we define some preliminary mathematical theorems that will aid us in proving the main Stochastic Approximation theorem.

Martingales: $\{Y_n\}$ is a Martingale if:

1. $\mathbb{E}(|Y_n|) < \infty \quad \forall n$
2. $\mathbb{E}(Y_{n+1}|\mathbb{F}_n) = Y_n$ for some $\mathbb{F}_n$

Theorem 7. ***Martingale Convergence Theorem***
Let Y_n be a Martingale and $\sum_{n\geq 0} \mathbb{E}(\|Y_{n+1} - Y_n\|^2 |\mathbb{F}_n) < \infty$ almost surely.

Then, $\exists Y_\infty \in \mathbb{R}^d, \|Y_\infty\| < \infty$ almost surely and $Y_n \to Y_\infty$ almost surely as $n \to \infty$

Theorem 8. ***Gronwall's Inequality***
Let $x : [0, T] \mapsto \mathbb{R}$ be non-negative for which there exists a contour L such that

$$\dot{x}(t) \leq Lx(t) \quad \forall t \in [0, T]$$

Then:

$$x(t) \leq e^{Lt}x(0) \quad \forall t \in [0, T]$$

Extension: If $L(t)$ is a non-negative, summable function, i.e.:

$$\dot{x}(t) \leq L(t)x(t)$$

Then,

$$x(t) \leq x(0) \exp\left(\int_0^t L(\tau)d\tau\right)$$

Extension: Let $x : [0, T] \mapsto \mathbb{R}$ be a bounded, non-negative, measurable function. Let $L : [0, T] \mapsto \mathbb{R}$ be a non-negative, measurable function, and let $B \geq 0$ be some constant. Also,

$$x(t) \leq B + \int_0^t L(\tau)x(\tau)d\tau \quad \forall t \in [0, T]$$

Then,

$$x(t) \leq B \exp\left(\int_0^t L(\tau)d\tau \right)$$

Theorem 9. ***Discrete Time Gronwall's Inequality***
Let $\{x_n\}$ and $\{a_n\}$ be non-negative sequences and $k \geq 0$. Also,

$$x_{n+1} \leq k + \sum_{m=0}^{n} a_m x_m$$

Then,

$$x_n \leq k \exp\left(\sum_{m=0}^{n} a_m \right)$$

Now we move on to the main theorem for Stochastic Approximation. First, consider the following assumptions:

Assumption 1. ***Lipschitz Continuity of*** *h:*

$$\exists L \geq 0 \quad \textit{such that} \quad \forall x, y \in \mathbb{R}^d : \|h(x) - h(y)\| \leq L \|x - y\|$$

Assumption 2. ***Diminishing Step-Size:***

$$\sum_{n \geq 0} \epsilon_n = \infty$$

and

$$\sum_{n \geq 0} \epsilon_n^2 < \infty$$

Assumption 3. ***Martingale Difference Noise:***

$$\begin{aligned} &\exists K \geq 0 \quad \textit{such that:} \\ &\mathbb{E}(\mu_{n+1}|F_n) = 0 \\ &\mathbb{E}(\|\mu_{n+1}\|^2 |F_n) \leq K(1 + \|x_n\|) \end{aligned}$$

Assumption 4. ***Bounded Iterates:***

$$\sup_n(\|x_n\|) < \infty \quad \textit{almost surely}$$

Assumption 5. ***Lyapanov Condition:*** *For the dynamical system $\dot{x} = h(x)$, there exists a positive, radially unbounded and continuously differentiable function $V : \mathbb{R}^d \mapsto \mathbb{R}$ such that for all $x \in \mathbb{R}^d$:*

$$\langle \nabla V(x), h(x) \rangle \leq 0$$

with strict inequality if $V(x) \neq 0$

Theorem 6. ***Stochastic Approximation***
Assume Assumptions 1-5 hold. Then $V(x_n) \to 0$ *almost surely as* $n \to \infty$*, i.e.* x_n *converges to the stationary point of the ODE almost surely.*

The detailed proof of this theorem will be discussed in subsequent lectures.

References

[1] Bertsekas, Dimitri P., and John N. Tsitsiklis. Neuro-dynamic programming. Vol. 5. Belmont, MA: Athena Scientific, 1996.

[2] Borkar, Vivek S. Stochastic approximation: a dynamical systems viewpoint. Vol. 48. Springer, 2009