Mathematical Foundations of Reinforcement Learning

Lecture III: Approximation Architectures & Optimization

Additional refs: "Deep learning: an introduction for applied mathematicians", by C. Higham and D. Higham, SIAM Review 61(4)(2019): 860-891

"Optimization methods for large-scale machine learning", L. Bottou, F.E. Curtis and J. Nocedal, SIAM Review 60(2)(2018): 223-311.

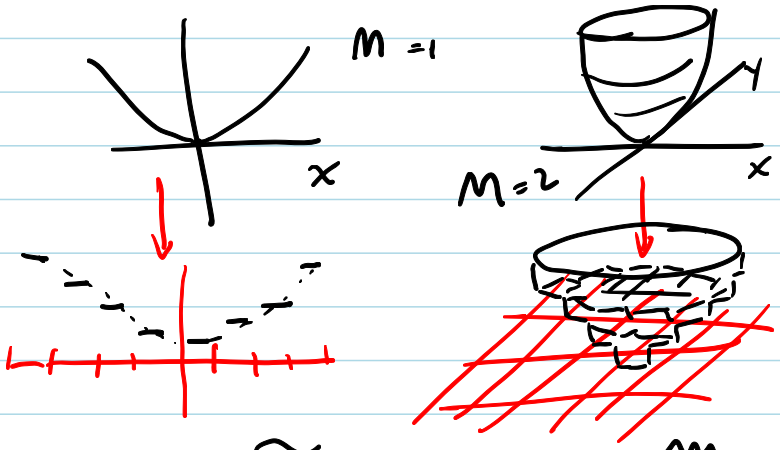
Today:

* Architectures
* ANN's
* Least Squares
* Nonlinear optim.

BASIC APPROXIMATION ARCHITECTURES

$$J^*(x) : \mathbb{R}^n \longrightarrow \mathbb{R}$$

IF $n \leq 3$



, PIECEWISE CONSTANT APPROX. ↕ (POLYNOMIAL)

$$J^*(x) \sim \tilde{J}(x, r) = \sum_{\ell=1}^{m} r_\ell \, \phi_\ell(x), \quad \text{WHERE}$$

$$\phi_\ell(x) = \begin{cases} 1 & \text{if } x \in S_\ell, \\ 0 & \text{if } x \notin S_\ell. \end{cases} \quad \underline{S_1}, \underline{S_2}, \underline{S_3}$$

↳ 1 PARAMETER r_ℓ PER SEGMENT → N SEGMENTS IN 1D

CURSE OF DIMENSIONALITY → N^2 SEGMENTS IN 2D

(ONLY PRACTICAL FOR $d \leq 3$) → N^n % IN nD

AN ALTERNATIVE: GLOBAL POLYNOMIAL APPROXIMATION

SUPPOSE $x := (x_1, \dots, x_m)$, AND WE LOOK FOR A QUADRATIC APPROXIMATION OF $J^*(x)$. DEFINE THE BASIS

$$\phi_0(x) = 1, \quad \phi_i(x) = x_i, \quad \phi_{ij}(x) = x_i x_j, \text{ THEN}$$

$$J^*(x) \sim \tilde{J}_2(x, r) = r_0 \phi_0(x) + \sum_{i=1}^{m} r_i \phi_i(x) + \sum_{i=1}^{m} \sum_{j=1}^{m} r_{ij} \phi_{ij}(x)$$

+: WE NEED $1 + m + m^2$ PARAMETERS $\ll N^m$

−: WE NEED TO INCREASE POLYNOMIAL DEGREE TO IMPROVE ACCURACY (TAYLOR)

$$J^*(x) \sim \tilde{J}_3(x, r) = \tilde{J}_2(x, r) + \sum_{i=1}^{m} \sum_{j=1}^{m} \sum_{k=1}^{m} r_{ijk} \phi_{ijk}(x)$$

↳ $\phi_{ijk}(x) = x_i x_j x_j$

$1 + m + m^2 + m^3 \Rightarrow O(m^{deg})$

FOR EXAMPLE, APPROX. A 10-DIMENSIONAL J^*, $n = 10$

* PIECEWISE CONSTANT APPROX WITH 8 NODES PER DIMENSION:

8^{10} NODES $\approx 10^9$ PARAMETERS (MEMORY?)

* GLOBAL POLYNOMIAL OF DEGREE 8:

10^8 PARAMETERS $\longrightarrow$ 1 ORDER OF MAGNITUDE LESS.

(GOOD FOR $n \approx 10$)

BOTH PIECEWISE AND GLOBAL POLYNOMIAL APPROXIMATIONS ARE LINEAR IN r. HOW DO WE FIND BEST r'S?

SUPPOSE WE HAVE A DATASET $\{x_i, J^*(x_i)\}_{i=1}^{N_S}$ WITH N_S SAMPLES OF THE OPTIMAL COST: INTERPOLATION OR REGRESSION

$$r^* = \underset{r}{\operatorname{argmin}} \sum_{i=1}^{N_S} \| J^*(x_i) - \tilde{J}(i, r) \|^2$$

$\tilde{J}(i,r) \hookrightarrow \sum r_i \phi_i(x_i)$

LINEAR LEAST SQUARES! (NORMAL EQNS., SVD...)

ARTIFICIAL NEURAL NETWORKS

WE NEED APPROXIMATION ARCHITECTURES TO HANDLE $n >>> 1$

$\Rightarrow$ NONLINEAR APPROXIMATION / NONLINEAR OPTIMIZATION

1-LAYER (SHALLOW) NETWORK:

$$\tilde{J}(x, v, r) = \sum_{\ell=1}^{\tilde{m}} r_\ell \, \phi_\ell(x, v) = \langle r, \phi(x, v) \rangle$$

WHERE $\phi_\ell(x, v) = \sigma((Ay(x)+b)_\ell)$ → coordinate

$A \in \mathbb{R}^{m \times n}$

$b \in \mathbb{R}^m$

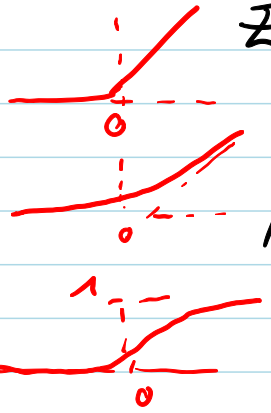
ENCODER $y : \mathbb{R}^n \to \mathbb{R}^n$ NONLINEAR

ACTIVATION FUNCTION:

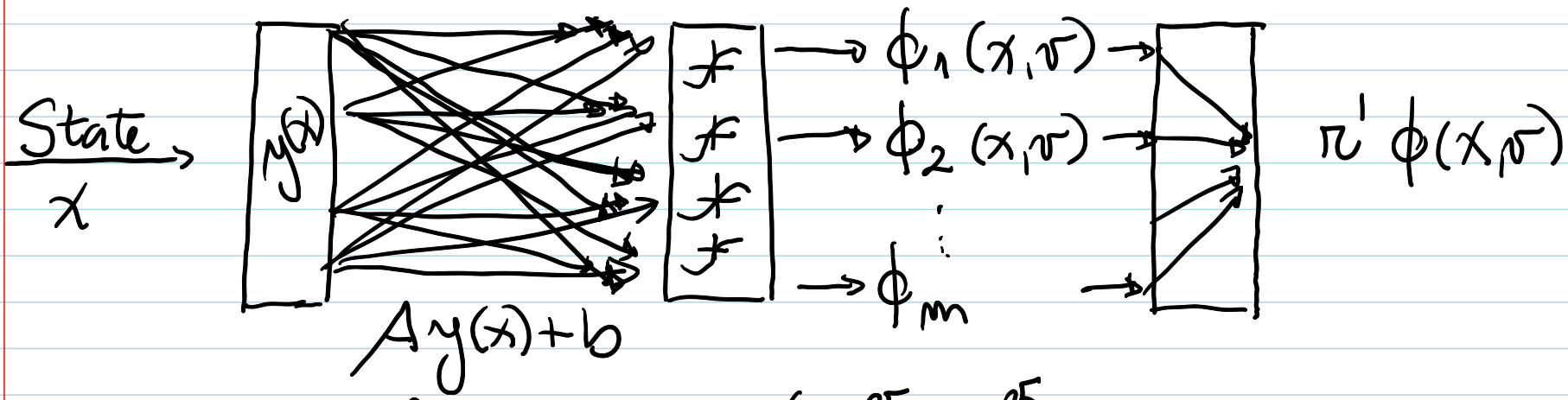
$\sigma(\xi) = \max\{0, \xi\}$

$\sigma(\xi) = \ln(1 + e^\xi)$

$\sigma(\xi) = \frac{1}{1 + e^{-\xi}}$



ENCODER - LINEAR - NONLINEAR $\longrightarrow$ LINEAR WEIGHTS.



DEFINING $\mathcal{L}^v(x) := \sigma(A^v x + b^v)$, COMPONENTWISE

MULTILAYER (DEEP) NEURAL NETWORK

$$\tilde{f}(x,v) = \mathcal{L}^{v_L} \circ \cdots \circ \mathcal{L}^{v_3} \circ \mathcal{L}^{v_2} \circ \mathcal{L}^{v_1}(y(x))$$

PARAMETERS
- # OF LAYERS (DEPTH)
- # OF NEURONS (WIDTH)
- CHOICE OF σ's, A's, b's
- ARCHITECTURE (FFN, RESNET, INN)

TRAINING NEURAL NETWORKS

NOTATION: $\tilde{J}(x, v, r), \tilde{J}(x, v) \longrightarrow \tilde{J}(x, r)$

GIVEN N_S SAMPLES $\{x_i, J^*(x_i)\}_{i=1}^{N_S}$, WE SOLVE

$$r^* = \arg\min_r \sum_{i=1}^{N_S} \frac{1}{2} \| J^*(x_i) - \tilde{J}(x_i, r) \|^2$$

$$:= \| g_i(r) \|^2$$

HOWEVER, $\tilde{J}(x, r)$ IS NONLINEAR IN r.

$\Rightarrow$ NONLINEAR, NON-CONVEX, LARGE-SCALE OPTIMIZATION.

NEED TO COMPUTE OPTIMALITY CONDITIONS REQUIRING $\nabla_r g_i(r)$, USE CHAIN RULE/BACKPROPAGATION.

In particular,

$$\frac{\partial \|g_k(r)\|^2}{\partial A^v(i,j)} = -g_k(r) L_{m+1} \sum_m L_m \cdots L_{v+1} \sum_v I_{ij} \sum_{k-1} \cdots$$

$A x+b$

$[\sigma^1 \cdots \sigma^1]$

$[\sigma \cdots \sigma]$

j

Optimality Conditions

$$\min_{r \in \mathbb{R}^m} g(r)$$

r^* is a LOCAL MINIMUM IF $\exists\ \varepsilon > 0$ SUCH THAT

$g(r^*) \leq g(r)$, $\forall\ r$ WITH $\|r - r^*\| < \varepsilon$

r^* IS A GLOBAL MINIMUM IF $g(r^*) \leq g(r)\ \ \forall\ r \in \mathbb{R}^m$

Necessary Optimality Conditions:

Let x^* be a local min, $g \in C^1$. Then $\nabla g(x^*) = 0$ (1^{st}-order), and $\nabla^2 g(x^*)$ is positive semidefinite. (2^{nd})

Sufficient Optimality Conditions:

$\nabla g(x^*) = 0$ and $\nabla^2 g(x^*)$ positive definite

$\Rightarrow$ x^* is a strict local min of g,

$\exists\ \alpha > 0$ and $\varepsilon > 0$ such that $g(x) \geq g(x^*) + \alpha \|x - x^*\|^2$

for $\|x - x^*\| < \varepsilon$.

Exercise: Go through the problem

$$\min_{x \in \mathbb{R}^n} \frac{1}{2} x^t Q x - b^t x, \quad Q \text{ symmetric } n \times n$$

GRADIENT METHODS:

SOLVING $\min_{x \in \mathbb{R}^n} g(x)$ THROUGH THE SEQUENCE $x_{t+1} = x_t + \gamma_t s_t$, $t = 0, \ldots,$

WHERE $\gamma_t > 0$ $\forall t$, AND s_t IS A DESCENT DIRECTION $\nabla g(x_t)^t s_t < 0$, UNTIL

$\nabla g(x_t) = 0$, OR SUFFICIENTLY SMALL.

How to choose (γ_t, s_t)?

GRADIENT METHODS:

CHOOSING S_t:

i) STEEPEST DESCENT: $S_t = -\nabla g(x_t)$

ii) NEWTON'S METHOD: $S_t = -(\nabla^2 g(x_t))^{-1} \nabla g(x_t)$

(IDEA: TO MINIMIZE A QUADRATIC APPROX OF g AROUND x_t)

iii) QUASI-NEWTON METHOD: $S_t = -D_t \nabla g(x_t)$

$D_t > 0$, SYMMETRIC (EMULATES $\nabla^2 g(x_t)$)

STEPSIZE RULES: $\gamma_t = C$, OR $\gamma_t \xrightarrow{t \to \infty} 0$

C ↓ SMALL

CONVERGENCE:

- ASSUME $\exists\, c_1, c_2 > 0$ SUCH THAT $\forall t$

$$c_1 \|\nabla g(\theta_t)\|^2 \leq -\nabla g(\theta_t)' s_t$$

$$\|s_t\| \leq c_2 \|\nabla g(\theta_t)\|$$

$$\theta_{t+1} = \theta_t + \gamma_t s_t$$

- ASSUME $L > 0$ $\quad \|\nabla g(\theta) - \nabla g(\bar{\theta})\| \leq L \|\theta - \bar{\theta}\|$

AND $\quad$ i) $\quad 0 < \gamma_1 < 2c_1/(Lc_2^2)$

ii) $\gamma_t \to 0$, $\sum_{t=0}^{+\infty} \gamma_t = \infty$

$\Rightarrow$ $g(\theta_t) \to -\infty$ OR $\lim_{t\to\infty} \nabla g(\theta_t) = 0$.

VARIANTS:

STEEPEST DESCENT WITH MOMENTUM

$$x_{t+1} = x_t - \gamma \nabla g(x_t) + \beta (x_t - x_{t-1}) \; ; \; \gamma > 0, \; 0 \leq \beta < 1$$

OR

$$x_{t+1} = x_t - \gamma \sum_{k=0}^{t} \beta^{t-k} \nabla g(x_k)$$

"PROXIMAL POINT ALGORITHM": (IMPLICIT)

$$x_{t+1} = x_t - \gamma \nabla g(x_{t+1})$$

$$\Downarrow$$

$$x_{t+1} = \text{PROX}_{\gamma g}(x_t) := \underset{x}{\operatorname{argmin}} \; g(x) + \frac{1}{2\gamma} \|x - x_t\|^2$$

STOCHASTIC GRADIENT DESCENT

MOTIVATION: $\min_{r} g(r) := \frac{1}{N_S} \sum_{k=1}^{N_S} g_k(r)$ $:= \|g_i(r)\|^2$

WHAT IF THE TRAINING SET N_S IS TOO LARGE?

→ NEED TO DO 1 BACKPROPAGATION/GRADIENT FOR EACH SAMPLE, AT EACH ITERATION (EXPENSIVE)

LET $K(t)$ BE SEQUENCE OF INDEPENDENT UNIFORM VARIABLES OVER $\{1, \dots, N_S\}$

SGD: $r_{t+1} = r_t - \gamma_t \nabla g_{K(t)}(r_t)$, WITH $\gamma_t > 0$.

→ $\mathbb{E}[\nabla g_{K(t)}(r_t)]$

OR

SGD (BATCHES): $r_{t+1} = r_t - \frac{\gamma_t}{\#S(t)} \sum_{k \in S(t)} \nabla g_k(r_t)$

$\frac{1}{N_S} \sum_{k=1}^{N_S} \nabla g_k(r_t)$

SUMMARY!

* LINEAR APPROXIMATION FOR $J^*(i)$
* ARTIFICIAL NEURAL NETWORKS
* TRAINING ANN'S
* GRADIENT DESCENT METHODS
* STOCHASTIC GRADIENT DESCENT