# function approximation

goal: architectures & algorithms for approximating (optimal) value and policy functions

Bertsekas & Tsitsiklis Ch 3

- we'd like to consider MDP

$$\min_u E[c(x,u)] \text{ s.t. } x^+ \sim P(x,u)$$

with infinite state and action spaces, e.g. $X = \mathbb{R}^n$, $\mathcal{U} = \mathbb{R}^m$

- now value $v : X \to \mathbb{R}$ and policy $\mu : X \to \Delta(\mathcal{U})$ cannot be represented with a finite-dimensional vector

* to use a digital computer, must approximate these functions

i.e. need: - architectures
- algorithms

# architectures for approximation

- we seek to approximate $f: X \to \mathbb{R}$ using $\tilde{f}: X \times \Theta \to \mathbb{R}$ (e.g. $f = v^*, \pi^*, \mu, v^\mu$) where $\theta \in \Theta$ are approximator parameters

→ what properties are desirable?

- provide good approximation of $f$
- easy to choose parameters $\theta$, e.g. linear (convex, at least)

- Broadly, can consider architectures that are linear or nonlinear in parameters $\theta$

we'll use semicolon to remind that $\theta$ is a parameter

- linear: $\tilde{f}(x;\theta) = \sum_{k=0}^{K} \theta_k \cdot b_k(x)$

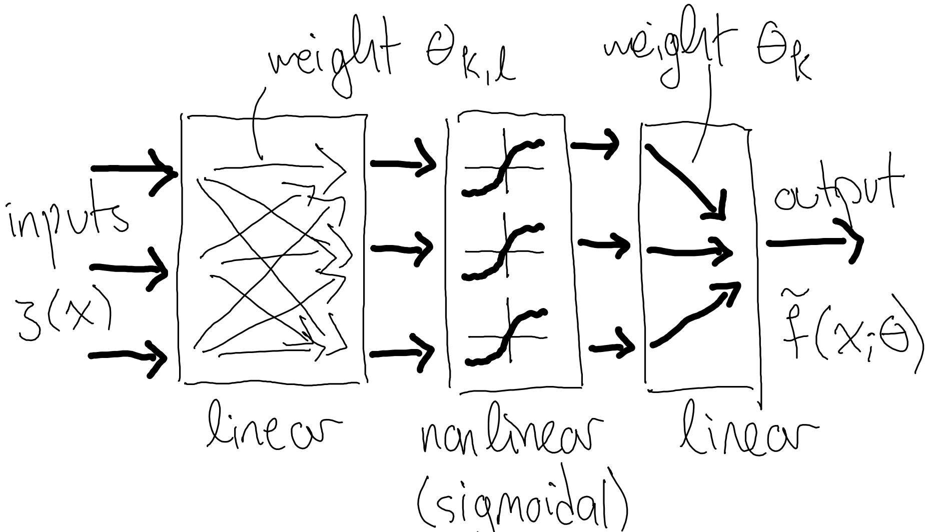
  where $\{b_k\}_{k=0}^{K}$ are called <u>basis functions</u>

  (though they might not form a

vector space basis)

- nonlinear: huge variety of options; (so-called) "neural" networks are popular:

ex: (single-layer perceptron)




$$\tilde{f}(x;\theta) = \sum_{k=1}^{K} \theta_k \cdot \sigma\left( \sum_{l=1}^{L} \theta_{k,l} \cdot z_l(x) \right)$$

where $\sigma : \mathbb{R} \to \mathbb{R}$ is <u>sigmoidal</u>,
i.e. differentiable, monotonic,

$$-\infty < \lim_{\tau \to -\infty} \sigma(\tau) < \lim_{\tau \to +\infty} \sigma(\tau) < +\infty$$


$\dfrac{e^{\tau} - e^{-\tau}}{e^{\tau} + e^{-\tau}}$ hyperbolic tangent

$\dfrac{1}{1+e^{-\tau}}$ logistic


$\overline{1+e^{-z}}$ logistic

and $z: X \to \mathbb{R}^L$ is <u>feature vector</u>,
i.e. (partial) state observation/
(approximate) sufficient statistic in
a finite-dimensional vector space

<u>fact</u>: any continuous function over
a closed and bounded domain can
be approximated arbitrarily well
by single-layer perceptron
(with sufficiently large $K, L$ and
the right choice of weights)
* multiple layers are allowed;
"in practice, the number of hidden
layers is usually one or two,
and almost never more than three" !

idea: backpropagation (ie chain rule)
- for a multilayer network, function is composition

$$\tilde{f}(x;\theta) = \sigma_N(W_N \cdot \sigma_{N-1}(W_{N-1} \cdots \sigma_1(W_1 \cdot x) \cdots)$$

- so gradient wrt $\theta$ is product

- so gradient wrt $\theta$ is product

$$D_\theta \tilde{f} = \left[ D\sigma_N \cdot DW_N \cdot D\sigma_{N-1} \cdot DW_{N-1} \cdots D\sigma_1 \cdot DW_1 \right](x;\theta)$$

* forward pass to evaluate derivatives, then <u>backward</u> pass to compute vector-matrix multiplication

---

## algorithms for approximation

- assuming there exists $\theta^* \in \Theta$ for which $\tilde{f}_{\theta^*} \simeq f$, how to find $\theta^*$?

→ propose an algorithmic approach to compute / approximate $\theta^*$ (specify algorithm inputs & any assumptions)

- formulate an optimization problem:

$$\min_{\theta \in \Theta} \underbrace{\| \tilde{f}_\theta - f \|^2_{\mathbb{R}^X}}_{\text{norm on function space } \mathbb{R}^X}$$

- since norm $\| \cdot \|_{\mathbb{R}^X}$ can't be evaluated on infinite-dimensional $\mathbb{R}^X$, assume given a finite dataset

$$\{(x, v(x))\}_{x \in \Xi} \text{ where } x \in X$$

$\{(x, v(x))\}_{x \in \Xi}$ where $x \in X$

and $v(x) \simeq v^*(x)$ or $v^\pi(x)$,

i.e. $v(x)$ is an estimate of optimal value or value of policy $\pi$ at state $x \in X$:

$$\min_{\theta \in \Theta} \sum_x \| \tilde{v}(x;\theta) - v(x) \|^2$$

- can then apply standard algorithms to this finite-dimensional (nonlinear) least-squares optimization problem

<u>ex</u>: linear case (3.2.2)

- suppose $\tilde{f}(x;\theta) = \sum_{k=1}^{N} \theta_k b_k(x)$

$$\Rightarrow D_\theta \tilde{f}(x;\theta) = [b_1(x) \cdots b_N(x)]$$

- then $\min_\theta \sum_x \| \tilde{f}(x;\theta) - f(x) \|_2^2$ is linear least-squares problem

$\longrightarrow$ solve this NLP

- stationarity: $\sum_x (\tilde{f}(x;\theta) - f(x)) D\tilde{f}(x,\theta)$

- stationarity: $\sum_x \left( f(x;\theta) - f(x) \right) \nabla f(x,\theta)$

$$= \underbrace{\sum_x \left( \sum_{k=1}^{N} \theta_k b_k(x) - f(x) \right) \left[ b_1(x) \cdots b_N(x) \right]}$$

$*$ <u>affine</u> in $\theta$ !

$$= A \cdot \theta + b = 0 \iff \theta = -A^{-1} b$$