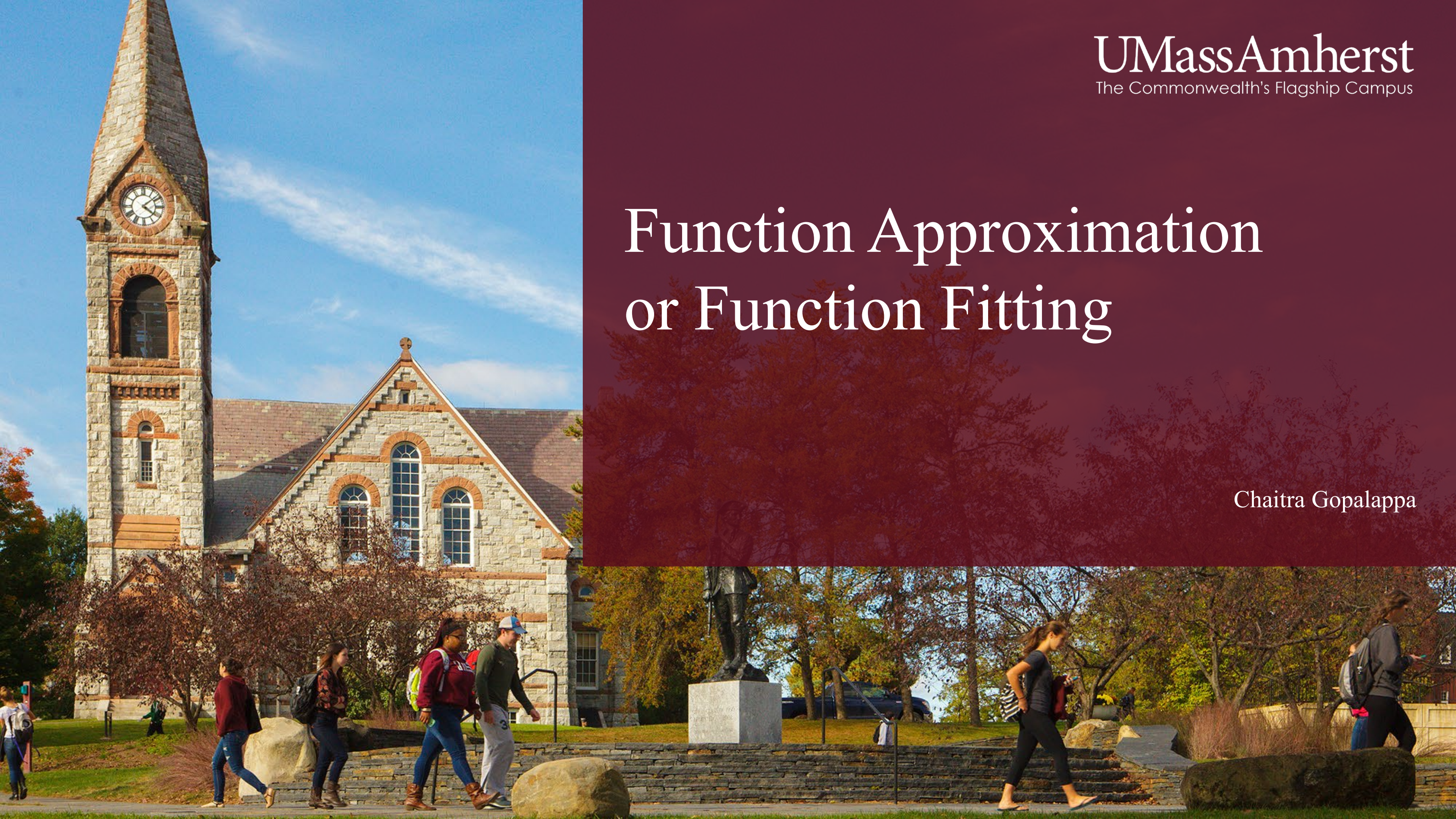


UMassAmherst
The Commonwealth's Flagship Campus

Function Approximation or Function Fitting

Chaitra Gopalappa

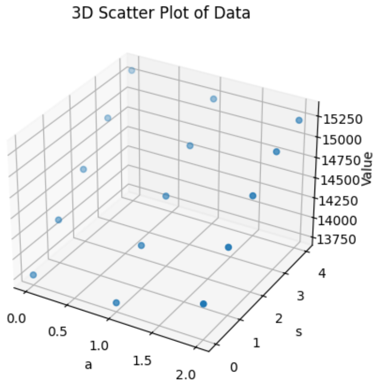
Challenges with Tabular Methods such as Q-Learning, SARSA

- Limited to discrete state, discrete action
- Issues for problems with large dimensionality

Solution: Function approximation also called Function Fitting

Function approximation: Context of Q-Table

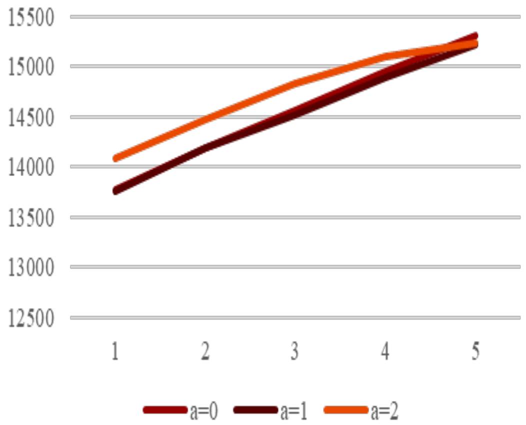
	a=0	a=1	a=2
s=0	13774.6	13760.6	14085
s=1	14187.7	14183.6	14478.7
s=2	14558.7	14522.4	14824.2
s=3	14950.3	14886.4	15095.4
s=4	15307.8	15218.8	15229.7



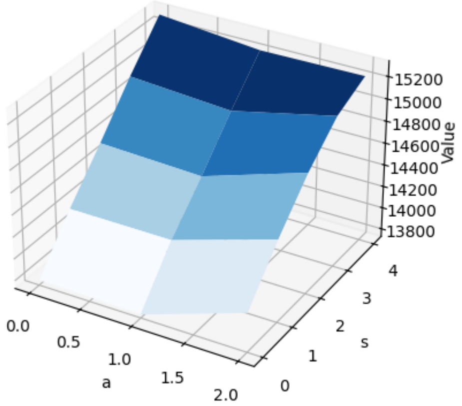
- Fit a function for $Q(s, A = a)$, for each a
- Fit a function for $Q(s, a)$, for each a

Optimal policy $\pi(s) = argmax_a Q(s, a)$

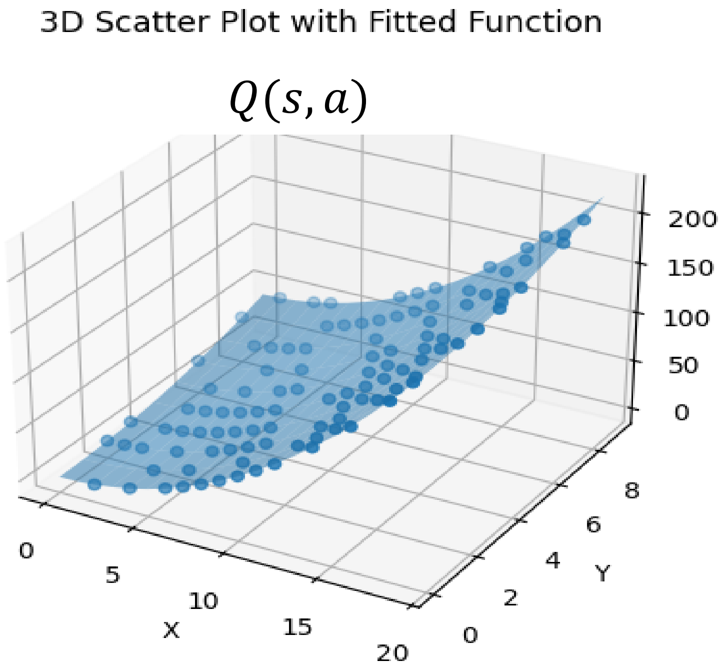
$Q(s, A = a)$



$Q(s, a)$



	0	1	2	3	4	5	6	7	8	9
0	19.2	2.6	-	96.2	-	-	-	13.7	-	-
1	88.2	55.6	36.4	121.1	-	132.0	21.7	-	-	-
2	89.8	-	119.8	-	-	-	-	37.9	120.0	28.8
3	-	-	59.2	58.7	11.5	65.5	28.1	9.4	59.1	144.0
4	98.1	-	55.4	122.3	106.6	42.7	-	116.8	12.3	-
5	81.9	-	55.4	-	-	69.8	-	0.4	76.2	-
6	-	86.4	130.2	-	46.1	77.3	-	-	-	-
7	58.7	71.8	39.7	25.4	101.2	88.3	35.2	4.5	68.4	-
8	67.4	57.8	-	-	-	7.4	39.1	3.1	179.6	-
9	97.4	-	15.6	-	-	118.3	-	-	-	35.2
10	95.7	-	70.7	-	88.2	-	-	101.2	0.7	16.7
11	-	-	-	33.3	-	-	106.6	-	-	4.3
12	-	70.4	25.4	-	15.9	122.0	40.6	-	48.8	46.7
13	56.8	22.6	-	84.1	-	-	22.1	-	70.2	11.1
14	60.5	1.3	-	-	63.8	-	-	-	2.9	-
15	-	-	-	-	-	-	58.3	95.9	45.3	-
16	67.7	32.9	72.6	44.8	53.5	118.0	55.7	40.6	-	-
17	-	115.8	8.8	68.0	27.2	6.4	-	76.0	49.2	-
18	94.3	-	-	-	12.9	-	-	-	57.9	-
19	128.4	-	-	-	-	113.5	142.4	68.4	-	-



Optimal policy $\pi(s) = argmax_a Q(s,a)$

We only sample to sufficiently represent the space (and not sample for every (s,a) pair); This is suitable for both discrete and continuous spaces

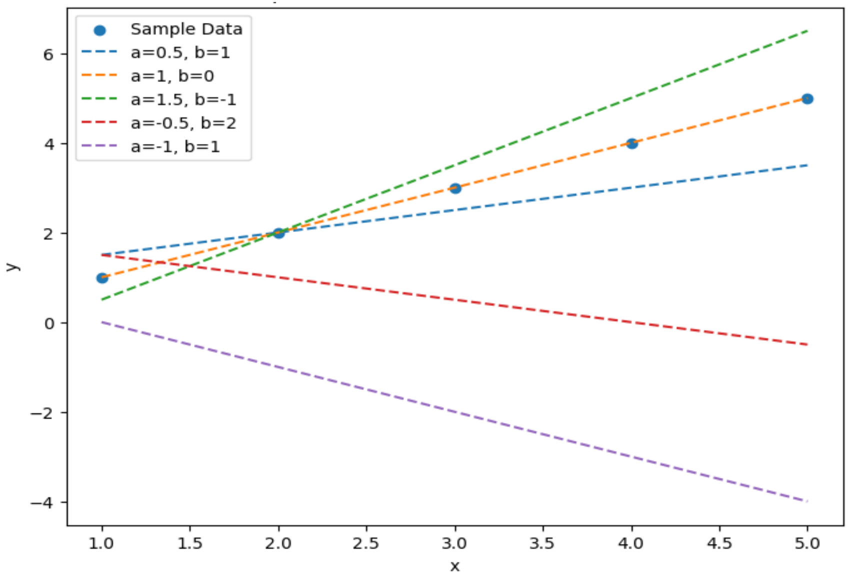
Generalizing: Function approximation/ function fitting

- **Problem**: Given $y = f(\vec{x})$, develop a mathematical representation of $f(\vec{x})$
- Typical dataset format:

	$\vec{x}$ (independent variables)					$y = f(\vec{x})$
Data samples (p)	$x[0]$	$x[1]$	$x[2]$	……	$x[N]$	$y = f(x[0], x[1], \ldots.)$
1						
2 . . .						
P						

Widrow-Hoff algorithm for function approximation

(When analytical form of function is known, apply W-H algo to estimate coefficients of function)

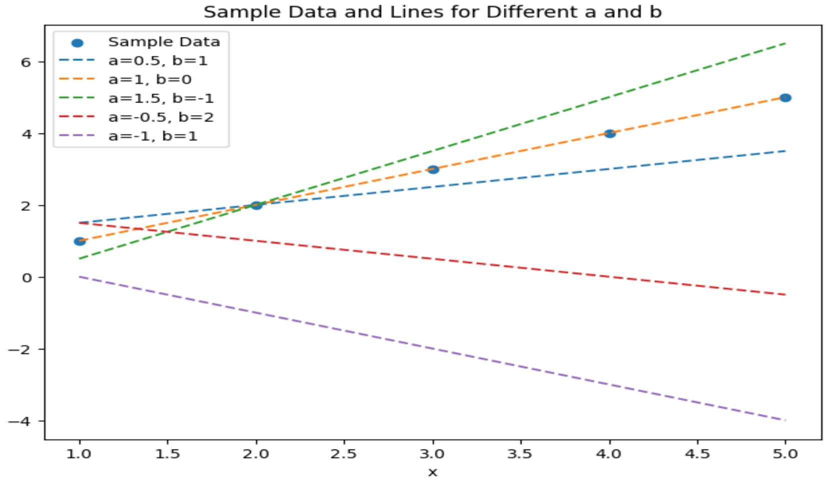


Problem formulation:

- Suppose we know the functional/analytical form of y; and suppose it is linear, i.e., of the form

$$y = \mathrm{w}[0] + \sum_{i=1}^{N} w[i]x[i] = \sum_{i=0}^{N} w[i]x[i] \ \text{ with } x[0] = 1$$

$$y = \vec{w}\vec{x}^T$$



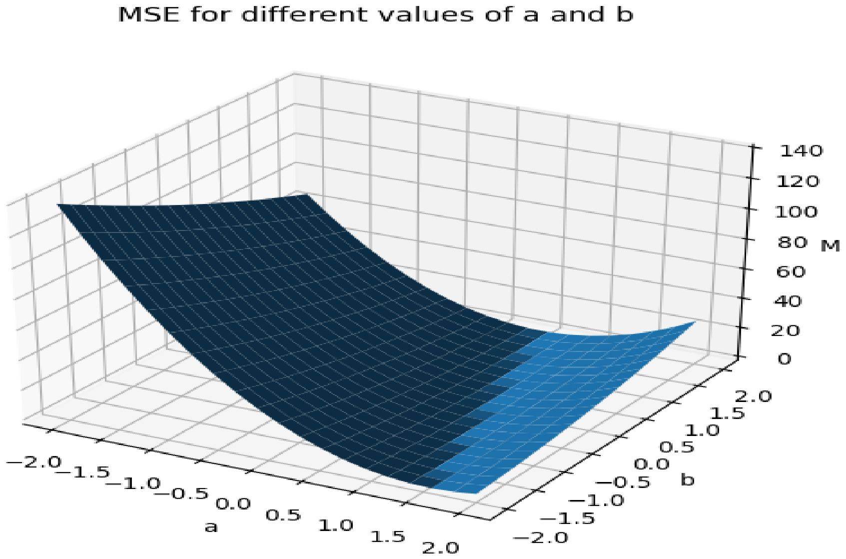
Problem formulation:

- Our objective is to determine the values of the coefficient $w[i]; i \in \{0,1,\ldots,N\}$ that provides the best fit to data by minimizing sum of square error (SSE) or E

$$\min_{\vec{w}} \mathrm{E} = \min_{\vec{w}} \sum_{p=1}^{P} \left[y_p - \sum_{i=0}^{N} w[i]x[i] \right]^2$$

$$\min_{\vec{w}} \mathrm{E} = \min_{\vec{w}} \sum_{p=1}^{P} \left[y_p - \vec{w}\vec{x}^T \right]^2 = \min_{\vec{w}} \sum_{p=1}^{P} \left[y_p - \overline{y_p} \right]^2$$

P is the number of data samples



- W-H uses steepest descent (SD):
- Recollect main transformation of SD

$$x[i] \leftarrow x[i] - \mu \frac{\partial f(\vec{x})}{\partial x[i]}; i = 1, \dots, N$$

$$\vec{x} \leftarrow \vec{x} - \mu \nabla f(\vec{x})$$

- For function fitting

$$f(\vec{w}) = E = \sum_{p=1}^{P} \left[y_p - \sum_{i=0}^{N} w[i]x[i] \right]^2 = \sum_{p=1}^{P} \left[y_p - \vec{w}\vec{x}^T \right]^2$$

Objective:

$$\min_{\vec{w}} \mathrm{E} \sim \min_{\vec{w}} \frac{\mathrm{E}}{2}$$

That is, here, we are solving for the coefficients w's so, replace $\vec{x}$ by $\vec{w}$ *in SD*

$$w[i] \leftarrow w[i] - \mu \frac{\partial f(\vec{w})}{\partial w[i]}; i = 1, \dots, N$$

$$\frac{\partial E/2}{\partial w[i]} = ?$$

Solution algorithm

$$\frac{\partial E/2}{\partial \omega[i]} = \frac{1}{2}\sum_{p=1}^{P} \frac{\partial}{\partial \omega[i]}\left(y_p - \sum_{i=0}^{N} \omega[i]x_p[i]\right)^2$$

$$= \sum_{p=1}^{P} \left(y_p - \sum_{i=0}^{N} \omega[i]x_p[i]\right)(-x_p[i])$$

- Thus, WH transformation is, for each i

$$w[i] \leftarrow w[i] - \mu \frac{\partial f(\vec{w})}{\partial w[i]}; i = 1, \dots, N$$

$$w[i] \leftarrow w[i] - \mu \sum_{p=1}^{P} \left(y_p - \sum_{i=0}^{N} \omega[i]x_p[i]\right)(-x_p[i])$$

Steps in W-H algorithm (solve for $\vec{w}$ by applying steepest descent)

1. Initialize
   1. Set $w[i]$ to values between 0 and 1; Set E_{old} (the SSE) to a large number
   2. Set $m = 0$
   3. Set μ to a small value typically a function of the number of iterations $\left(e.g., \mu = \frac{A}{B+m}\right.$; A and B are scalars; $\left., u = \frac{1}{m}\right)$
2. Compute $\bar{y}_p = \sum_{i=0}^{N} \omega[i] x_p[i]$ for each $p \in \{1,..,P\}$;
   1. y_p is known (data samples)
3. Update $w[i]$ for each $i = 0,1,\ldots,N$
4. $\omega_{m+1}[i] \leftarrow w_m[i] + \mu \sum_{p=1}^{m} (y_p - \bar{y}_p) x_p[i]$
5. Set $m = m + 1$. Calculate E_{new} ;
6. $E_{\text{new}} = \sum_{p=1}^{m} (y_p - \bar{y}_p)^2$
7. Update μ.
8. If $|E_{\text{new}} - E_{\text{old}}|$ < tolerance STOP. Otherwise set $E_{\text{old}} = E_{\text{new}}$ and go back to step 2.

What if:

- We assumed a linear function in previous slides

$$y = \vec{w}\vec{x}$$

- What-if we know that data fits to a non-linear function, e.g.,

$$y = w[1]x[1]^2 + w[2]x[2]^3 + w[3]x[1]x[2]$$

- What would be suitable method to solve for $\vec{w}$?

What if:

- We assumed a linear function in previous slides

$$y = \vec{w}\vec{x}$$

- What-if we know that data fits to a non-linear function, e.g.,

$$y = w[1]x[1]^2 + w[2]x[2]^3 + w[3]x[1]x[2]$$

- What would be suitable method to solve for $\vec{w}$?
- Rewrite using transformed variables and solve for $\vec{w}$ using SD

$$y = \vec{w}\vec{z}$$

where,

$$z[1] = x[1]^2$$
$$z[2] = x[2]^3$$
$$z[3] = x[1]x[2]$$

What if?

- What if functional form is unknown?

UMassAmherst

The Commonwealth's Flagship Campus