28.4.2015

Optimal and Learning Control for Autonomous Robots

Lecture 10

ADRL

Jonas Buchli
Agile & Dexterous Robotics Lab

FNSNF

ETH Zürich

Class logistics

Have you signed up for the Interview for Ex 2?

http://doodle.com/w2ahzwpdrwa5p5a9
(using your group ID!)

Lecture 10 Goals

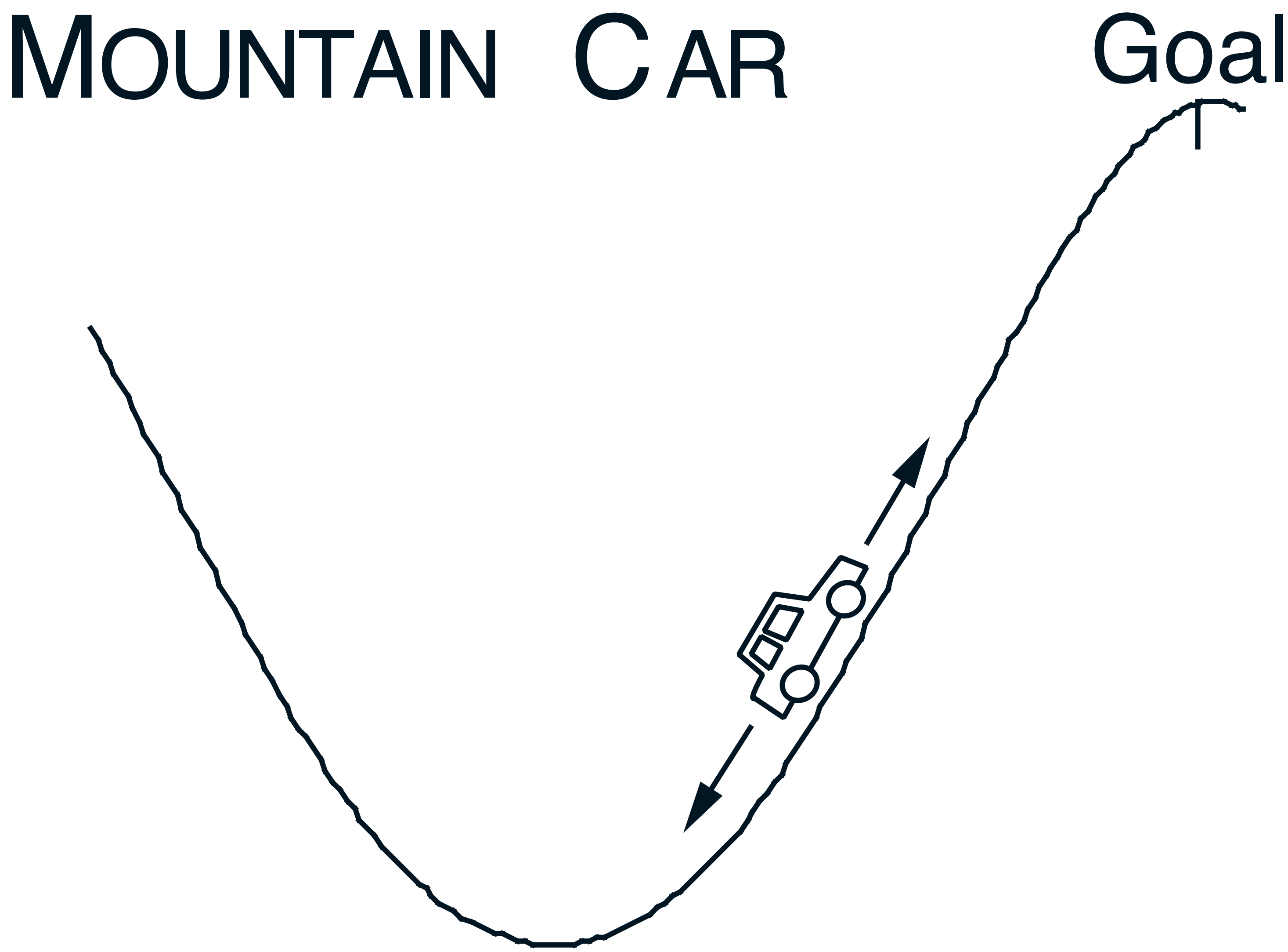
★Function approximation, basis functions
★Path integral stochastic optimal control
★Path integral RL

(Back to) Continuous state action spaces

Function approximation

Mountain Car Problem

A continuous-state problem



Reward

- Goal: +10
- Step: -1

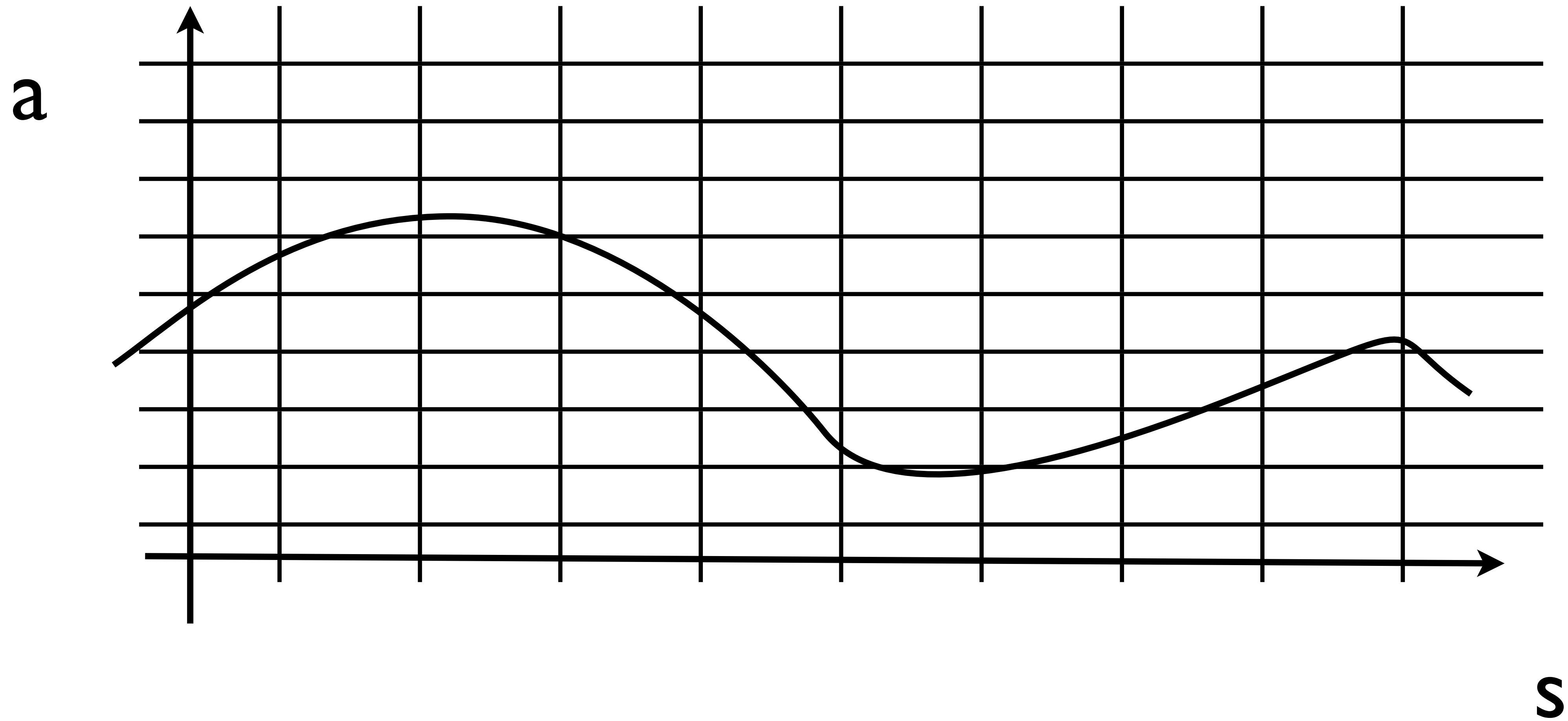
Solving Mountain Car

Practically continuous problems are (almost?) always solved by some sort of discretization!!!

Let's look at the problem of discretization in a bit more detail!

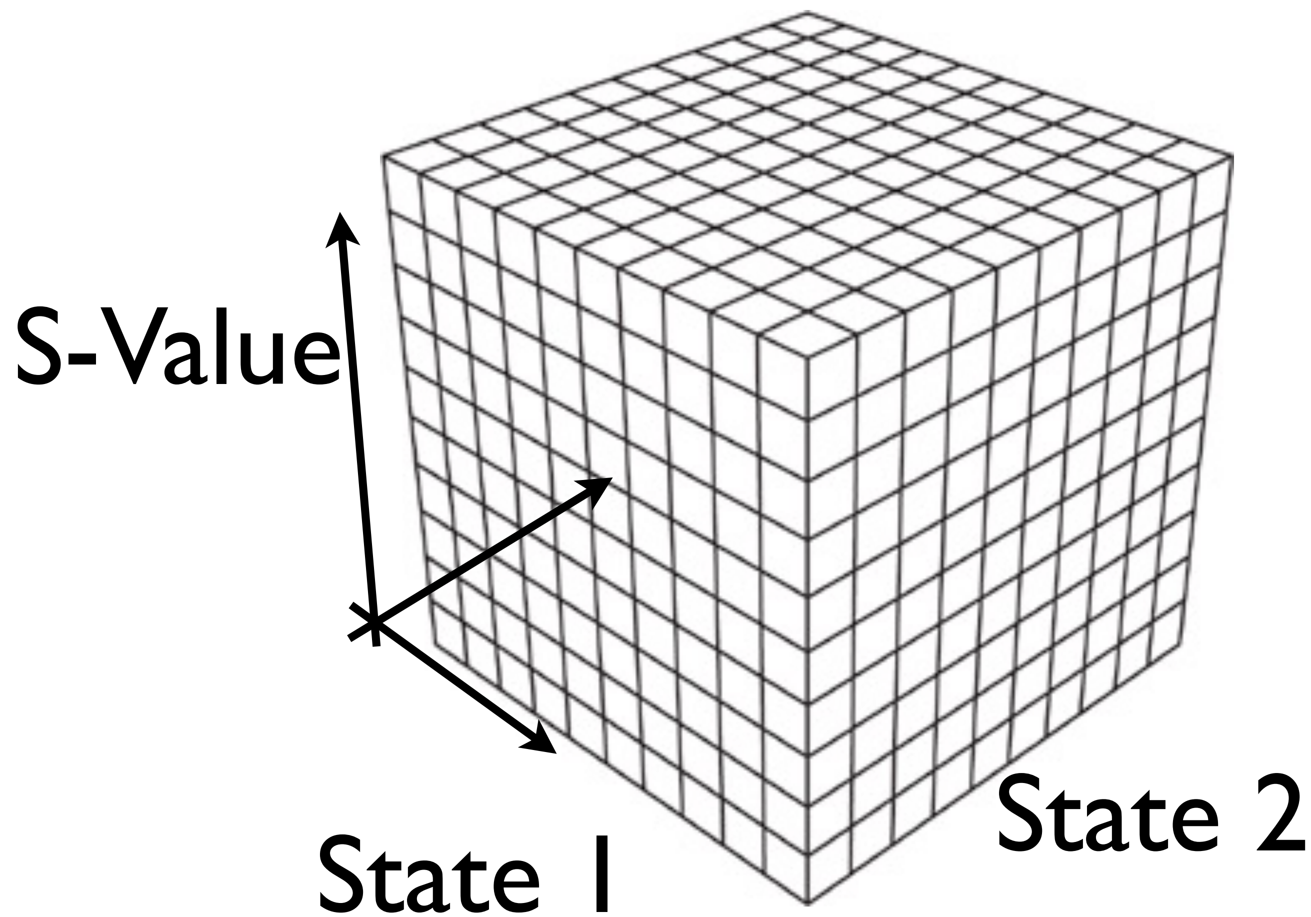
Control Policies

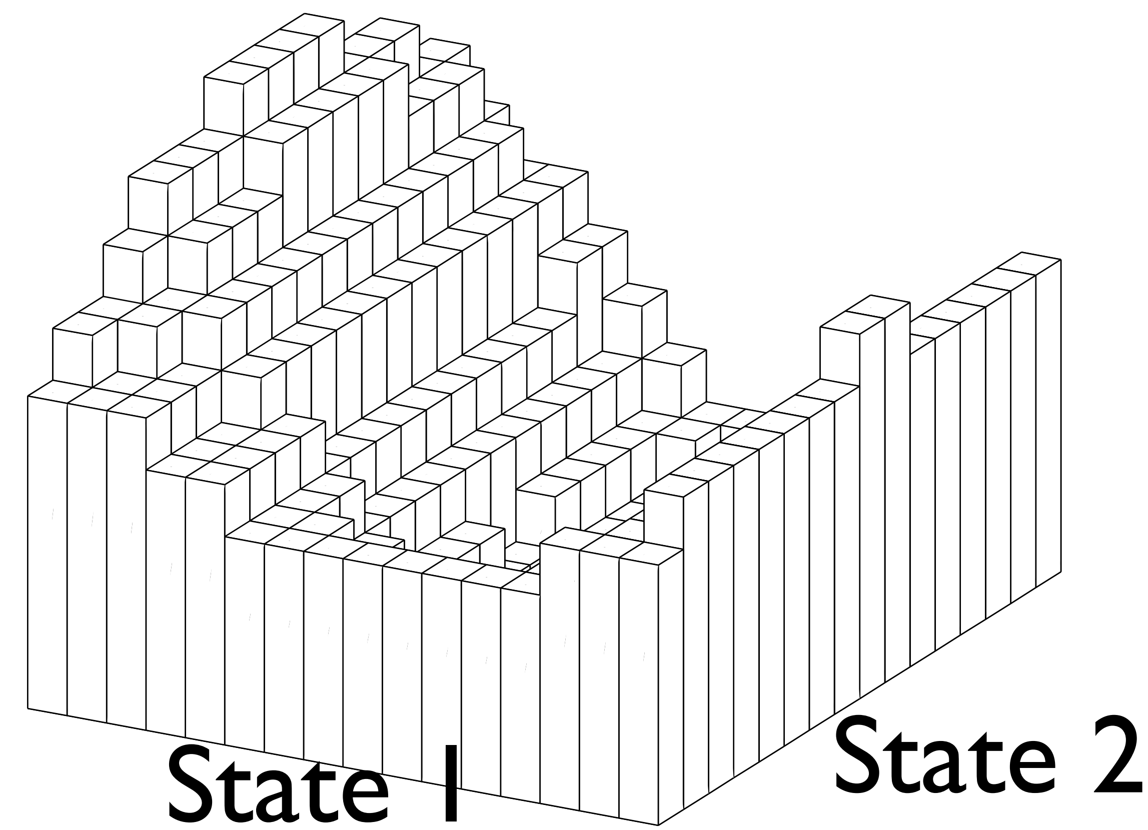
state - action mapping

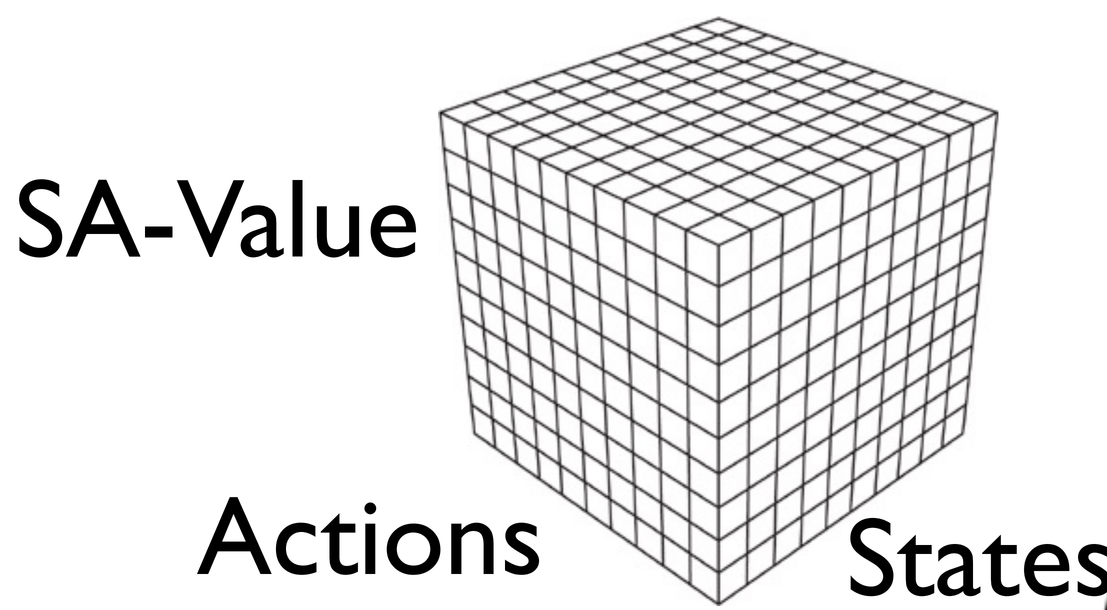


Problem: dimensions!

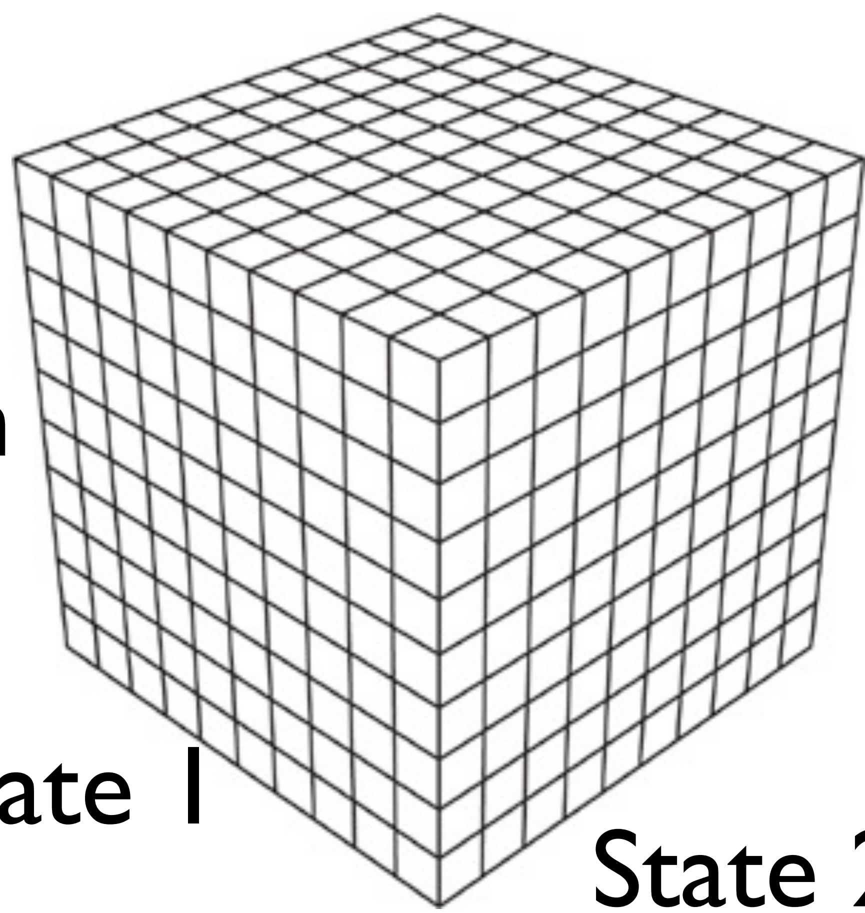
Value discretization

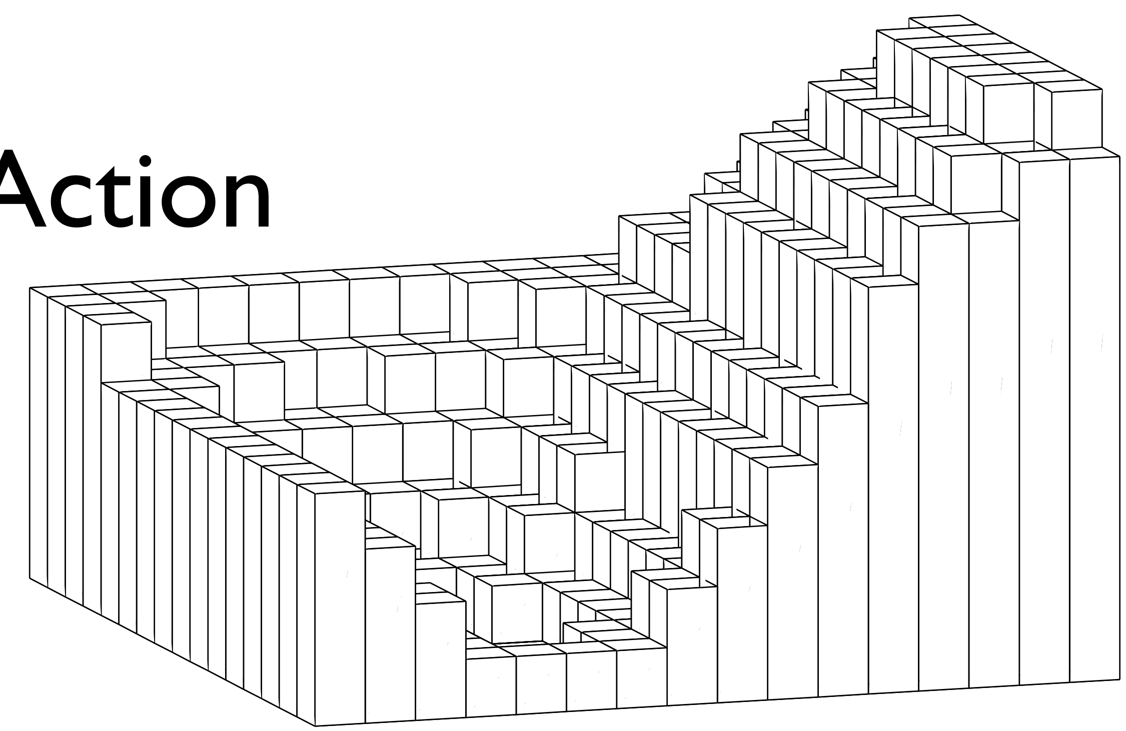






Action discretization

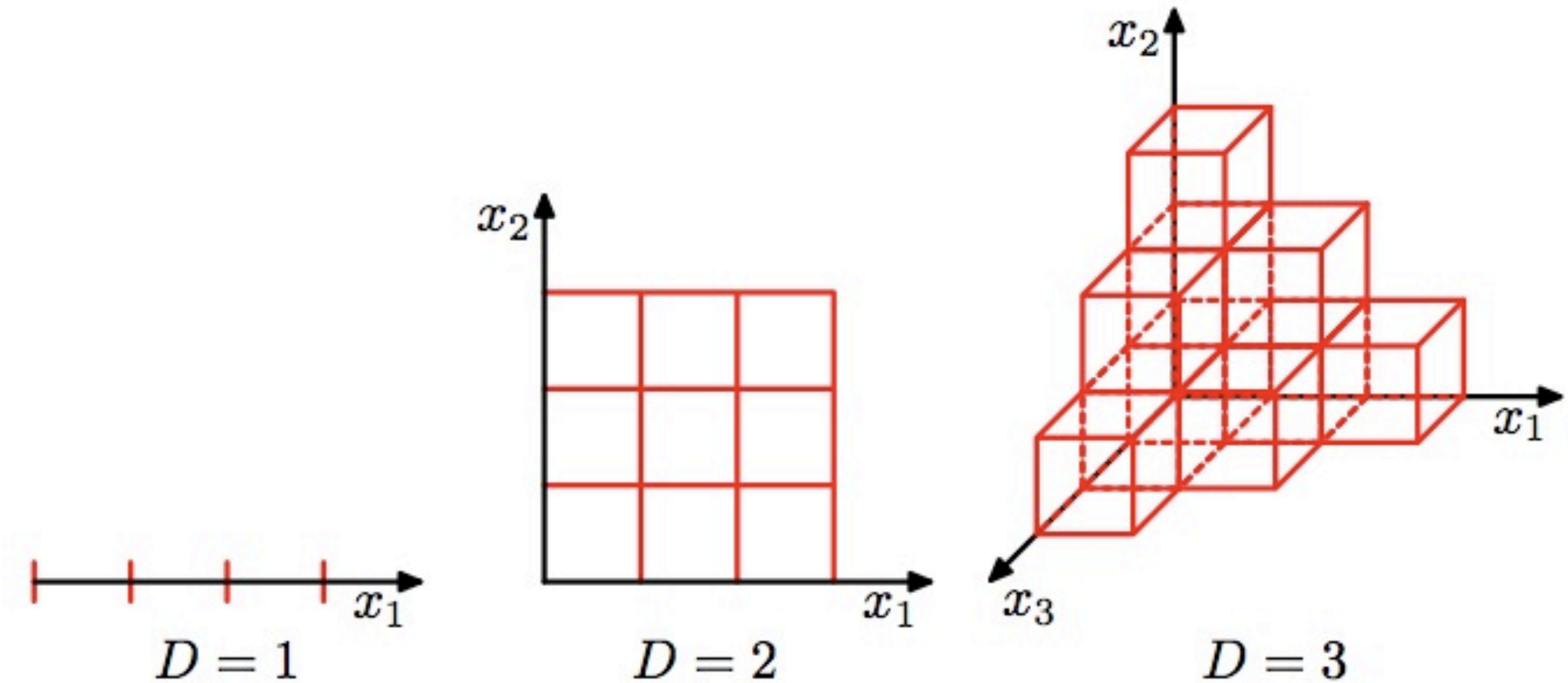




Grid

Policy

Discretization of large/ high-dim. state spaces



[Bishop]

cf. Discussion on exploration

Examples

High dimensional continuous state actions spaces with stochastic dynamics

Optimal(?) control in fluids

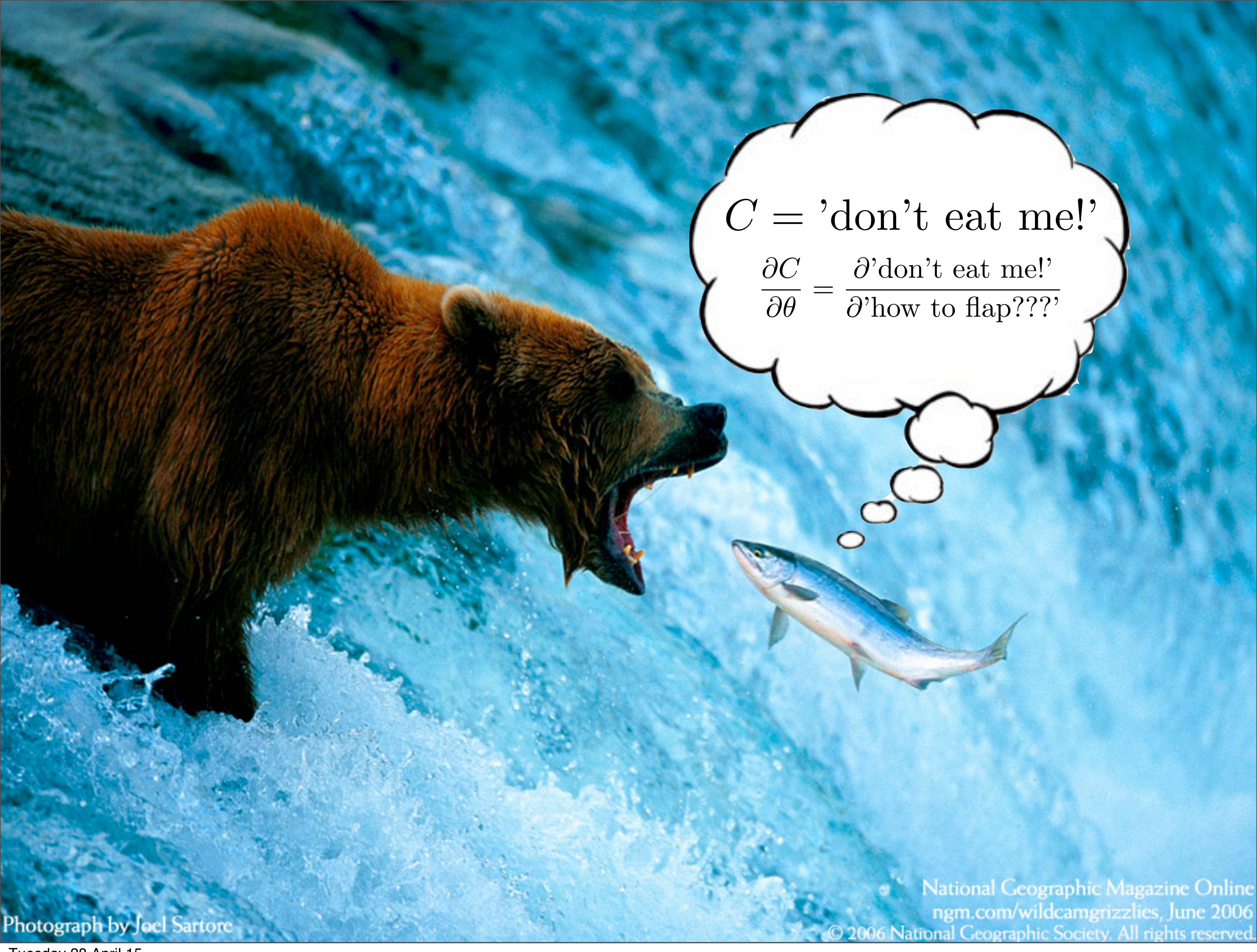
Approach: Computational Fluid dynamics & Evolutionary Algorithm

Stefan Kern and Petros Koumoutsakos
ETH Zurich

Kristina Eschler
hgk Zurich

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

http://www.cse-lab.ethz.ch


$C = \text{'don't eat me!'}$
$\frac{\partial C}{\partial \theta} = \frac{\partial \text{'don't eat me!'}}{\partial \text{'how to flap???'}}$
Photograph by Joel Sartore
National Geographic Magazine Online
ngm.com/wildcamgrizzlies, June 2006
© 2006 National Geographic Society. All rights reserved

$$\rho\left(\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v}\cdot\nabla\mathbf{v}\right) = -\nabla p + \nabla\cdot\mathbf{T} + \mathbf{f}$$

Reinforcement Learning: real-world- sampling based optimal control

Why do high dimensional systems appear ‘repeatable/low-dimensional to us?

)

Discretization issues

- Inflexible (need to decide division ahead of time)
- Inefficient (e.g. if slow varying function but division was decided to be fine)
- or not precise enough... if tiling is too coarse

Is there a way to avoid the issues of tiling and get a handle on the complexity?

Ideally: parameter(s) controlling complexity (in this class)

even more ideal: complexity adjusted automatically (not addressed in this class)

Function approximation

Goal: approximate a given (arbitrary) function

Need:
'Basis functions'
Parameters

Function approximation

Function approximation

Function approximation: $f(x, \theta) \approx y(x)$

function approximator/model → $f(x, \theta)$

target function (e.g. observed) → $y(x)$

$$\min \left(f(x) - y(x) \right) \{\forall x\}$$

does not work, no finite minimum

$$\min \{ ||f(x) - y(x)|| \}$$

$$\theta \, ?$$

need 'approximators' that can 'easily' be tuned and can express arbitrary functions

Norms

inner product

$$\|\boldsymbol{x}\| := \sqrt{\boldsymbol{x} \cdot \boldsymbol{x}}.$$

Euclidian Norm

$$\|\boldsymbol{x}\| := \sqrt{x_1^2 + \cdots + x_n^2}.$$

$$\mathbb{R}^n$$

only defined on Euclidean spaces

p-Norm

$$\|\mathbf{x}\|_p := \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p}.$$

1-Norm

$$\|\boldsymbol{x}\|_1 := \sum_{i=1}^{n} |x_i|.$$

max-Norm

$$\|\mathbf{x}\|_\infty := \max\left(|x_1|, \ldots, |x_n|\right).$$

A D R L

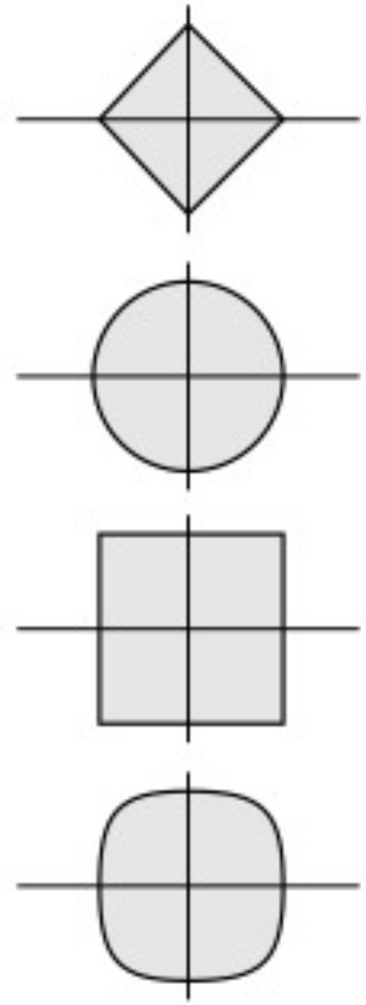
ETH Zürich

$$\|x\|_1 = \sum_{i=1}^{m} |x_i|,$$

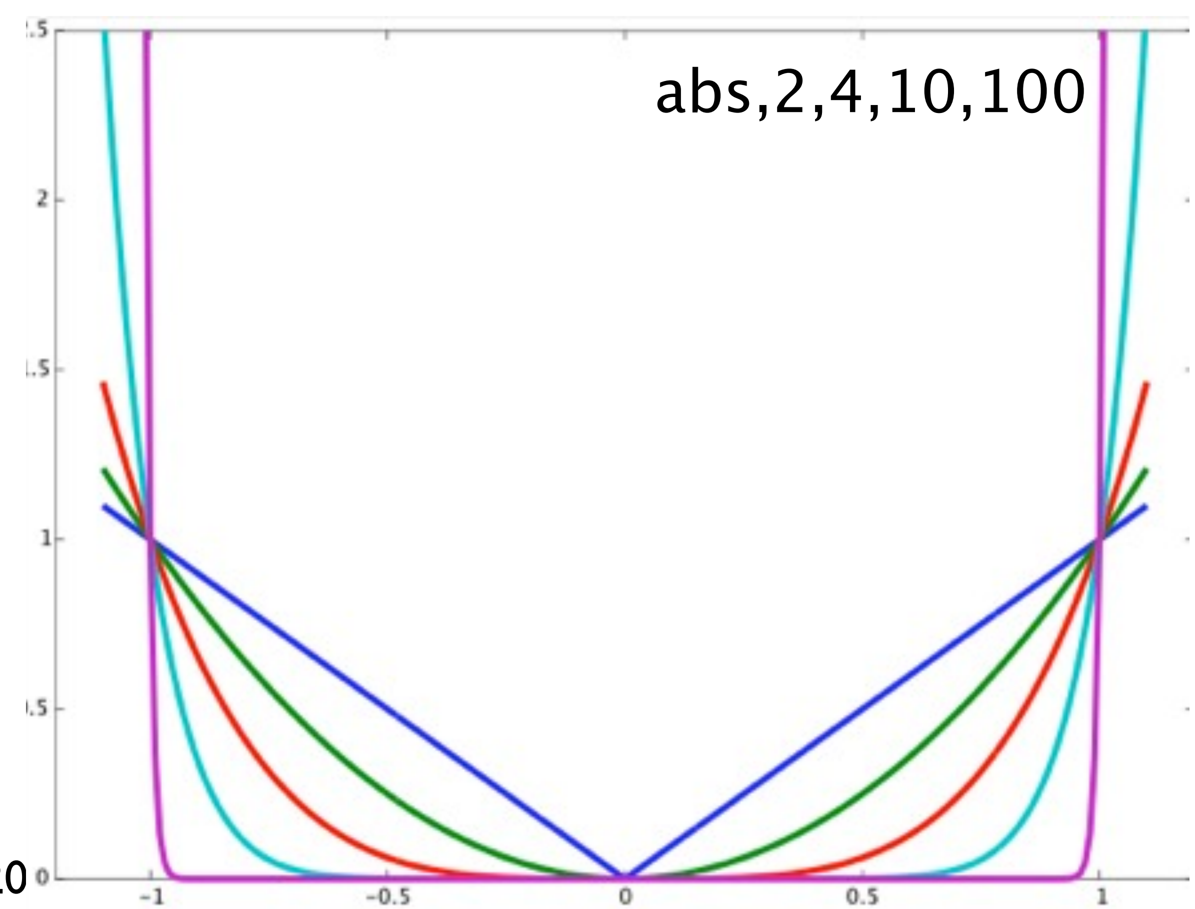
$$\|x\|_2 = \left(\sum_{i=1}^{m} |x_i|^2 \right)^{1/2} = \sqrt{x^* x},$$

$$\|x\|_\infty = \max_{1 \le i \le m} |x_i|,$$

$$\|x\|_p = \left(\sum_{i=1}^{m} |x_i|^p \right)^{1/p} \quad (1 \le p < \infty).$$



$$||\mathbf{x}||_p = (x_1^p + \ldots + x_n^p)^{\frac{1}{p}}$$



Norms

inner product

$$\|\boldsymbol{x}\| := \sqrt{\boldsymbol{x} \cdot \boldsymbol{x}}.$$

Euclidian Norm

$$\|\boldsymbol{x}\| := \sqrt{x_1^2 + \cdots + x_n^2}.$$

$$\mathbb{R}^n$$

only defined on Euclidean spaces

p-Norm

$$\|\mathbf{x}\|_p := \left(\sum_{i=1}^n |x_i|^p\right)^{1/p}.$$

1-Norm

$$\|\boldsymbol{x}\|_1 := \sum_{i=1}^n |x_i|.$$

max-Norm

$$\|\mathbf{x}\|_\infty := \max\left(|x_1|, \ldots, |x_n|\right).$$

Example:

$$x = \begin{bmatrix} p \\ \alpha \end{bmatrix}$$

$$[\mathbf{x}] = \begin{bmatrix} m \\ rad \end{bmatrix}$$

~~$[m^2] + [rad^2]$~~

no 'natural' definition of distance

Weighted p-norms

Weighted Euclidean

$$\|x\|_W = \|Wx\|$$

W: diagonal weighting matrix

$$\|x\|_W = \left(\sum_{i=1}^{m} |w_i x_i|^2 \right)^{1/2}$$

A D R L

ETH Zürich

Function approximation as optimization

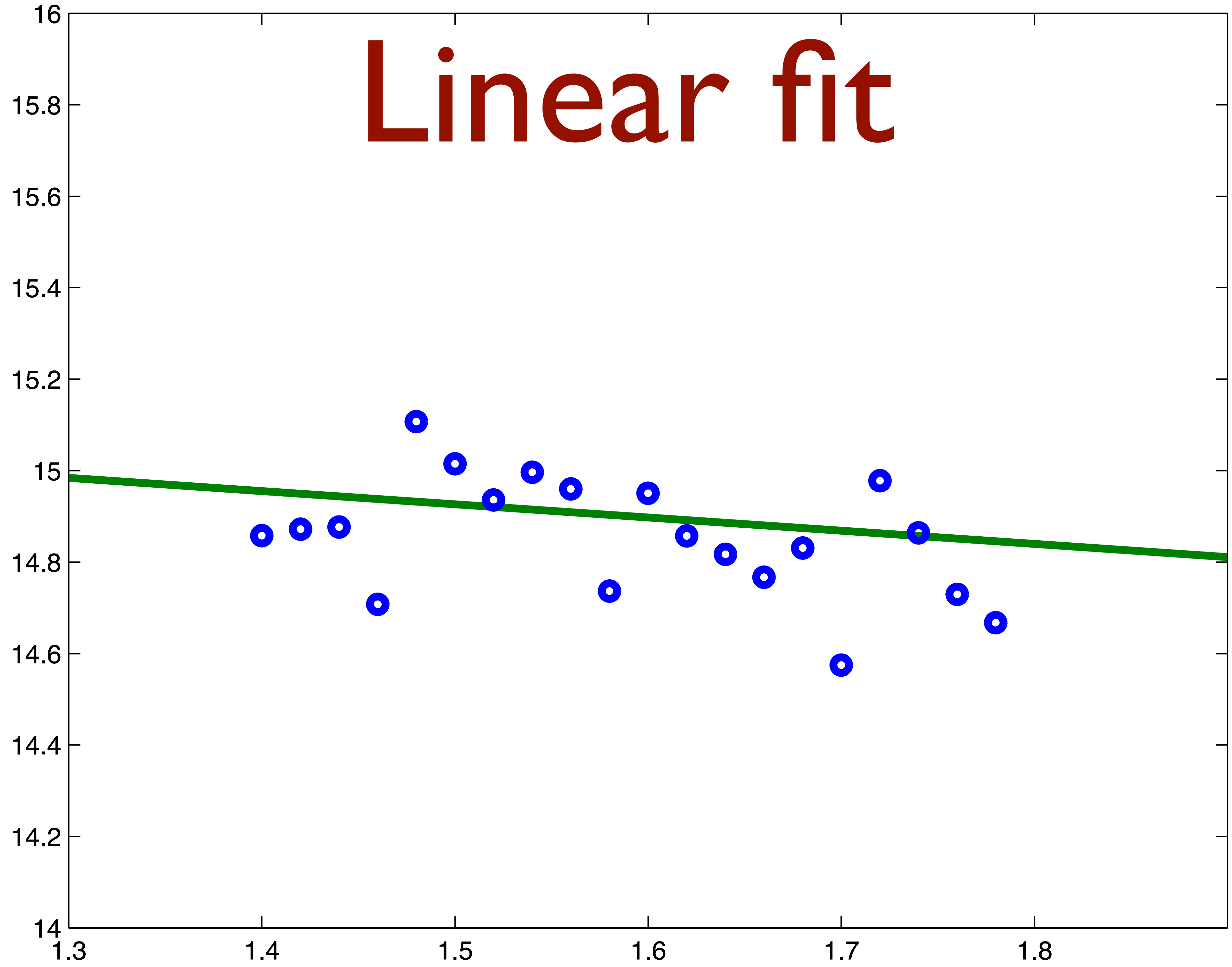
e.g. linear regression:

opt. in least squares sense...

Cost function = error function

Look at the gradient, set it to 0

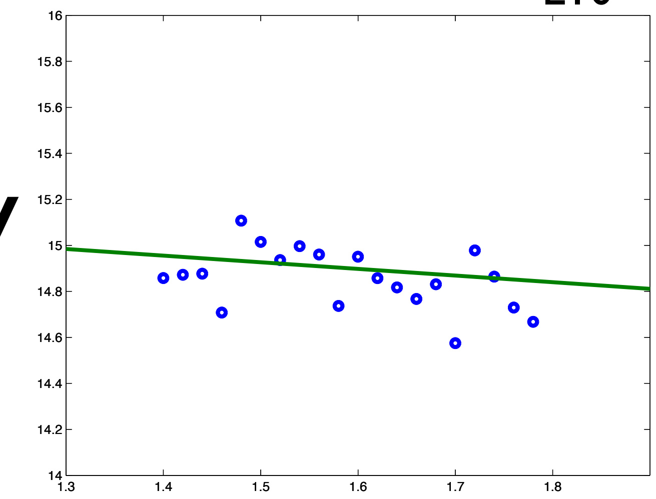
This is key for lots of learning methods



A D R L

ETH Zürich

Least squares linear regression



Input variables: $\mathbf{x} = [x_1, \ldots, x_n]$

Observations: t with gaussian noise: $t = y + \epsilon$

Parameters: $\mathbf{b} = [b_1, \ldots, b_n]$

Linear model $y = \mathbf{b}\mathbf{x}^{\mathbf{T}}$

Find b such that $\min ||y - t||$ at input:

p observations: $\mathbf{t} = \begin{bmatrix} t_1 \\ \ldots \\ t_p \end{bmatrix}$ $\mathbf{X} = \begin{bmatrix} [x_1, \ldots, x_n]_1 \\ \ldots \\ [x_1, \ldots, x_n]_p \end{bmatrix}$

$$\min ||\mathbf{X}\mathbf{b}^{\mathbf{T}} - \mathbf{t}||$$

A D R L

ETH Zürich

Solving for LS fit

same minimum — 'quadratic form'

$$\min ||\mathbf{X}\mathbf{b}^{\mathbf{T}} - \mathbf{t}|| \;\Leftrightarrow\; \min \left[\left(\mathbf{X}\mathbf{b}^{\mathbf{T}} - \mathbf{t}\right)^{T} \left(\mathbf{X}\mathbf{b}^{\mathbf{T}} - \mathbf{t}\right) \right]$$

'cost function'! ('distance')

$$E = \left(\mathbf{X}\mathbf{b}^{\mathbf{T}} - \mathbf{t}\right)^{T} \left(\mathbf{X}\mathbf{b}^{\mathbf{T}} - \mathbf{t}\right)$$

$$\min E \Leftrightarrow \nabla E = 0$$

$$\nabla E = 2\mathbf{X}^{\mathbf{T}}(\mathbf{X}\mathbf{b}^{\mathbf{T}} - \mathbf{t})$$

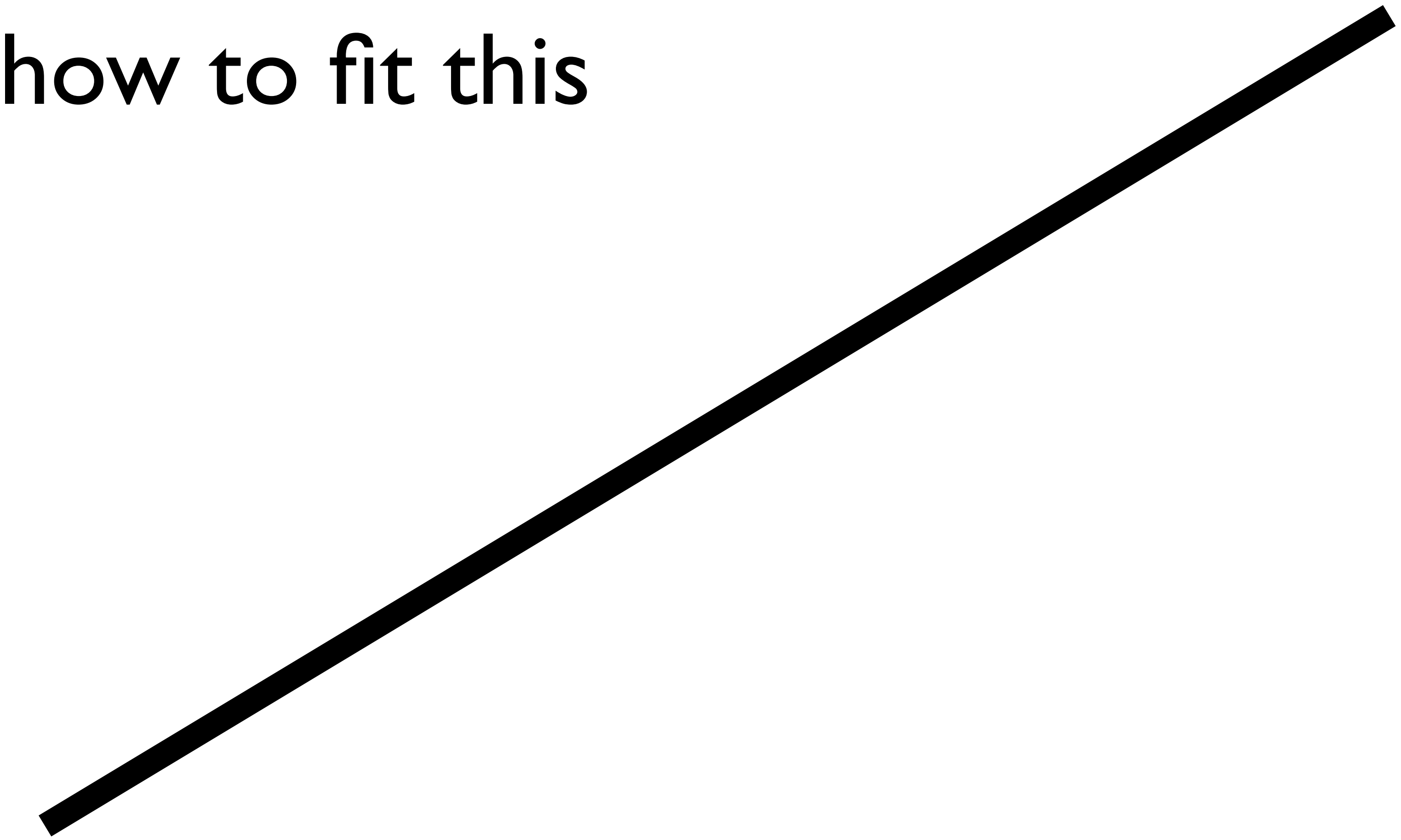
$(\mathbf{X}^{\mathbf{T}}\mathbf{X})$ nxn

$$\mathbf{b}^{\mathbf{T}} = (\mathbf{X}^{\mathbf{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathbf{T}}\mathbf{t}$$

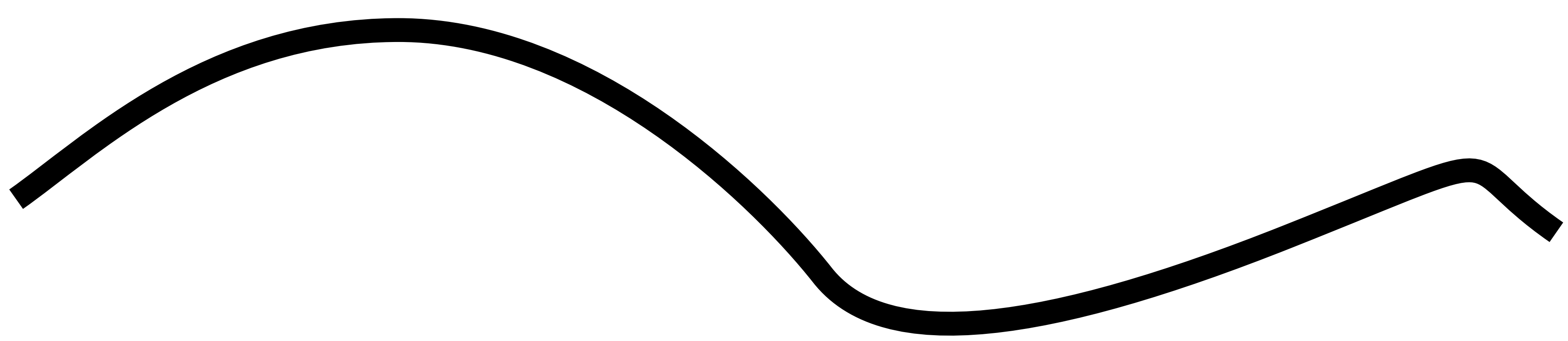
Pseudoinverse!

Matlab: `pinv`

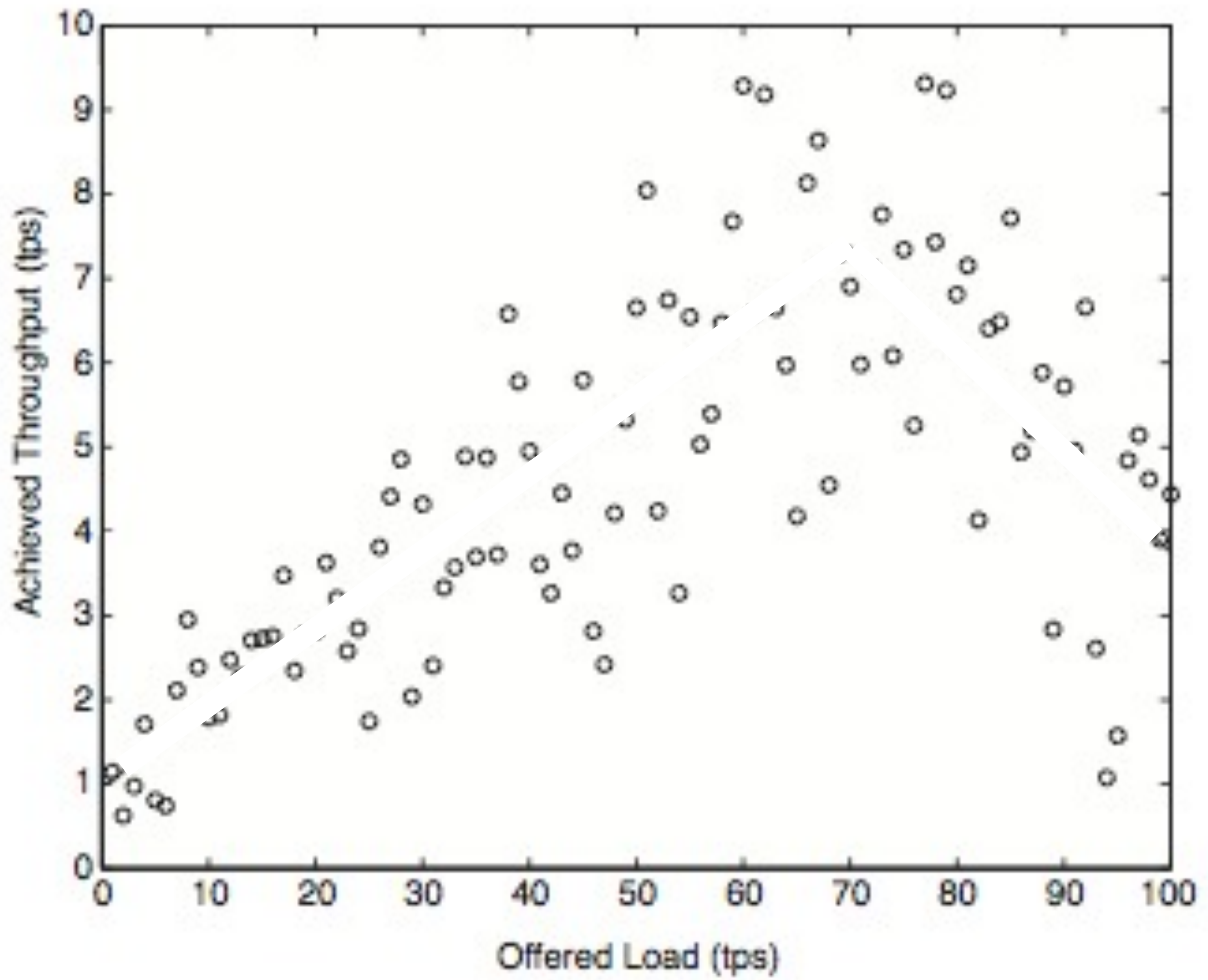
know how to fit this



but what about this?

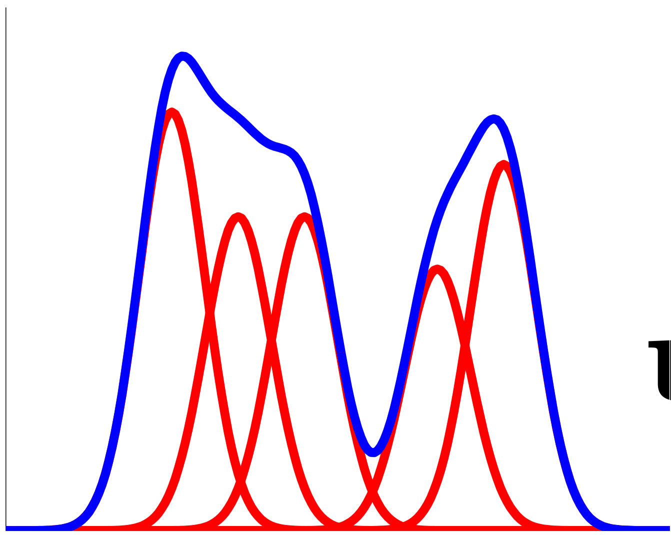


Fit with linear model?



[Le Boudec, Performance evaluation, EPFL]

Function approximation using basis functions



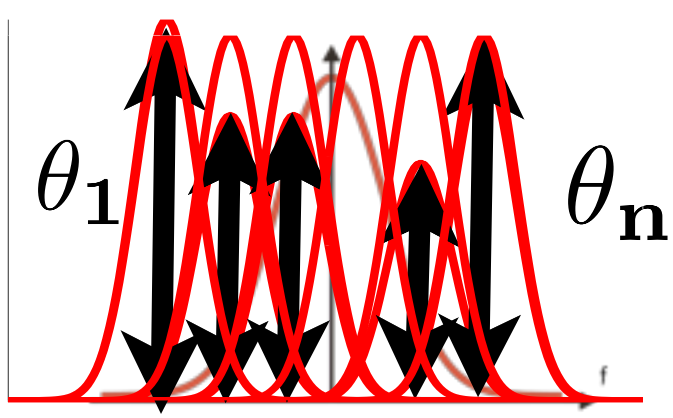
$$\mathbf{u(t)} = \boldsymbol{\phi}\mathbf{(t)}^{\mathbf{T}}\boldsymbol{\theta}$$

Basis functions

Parameters

Learned

Torques
Current
ref. position
'Heading'
...



A D R L

ETH Zürich

'inner product'

$$u(t) = \boldsymbol{\phi}(\mathbf{t})^{\mathbf{T}}\boldsymbol{\theta}$$

$$u(t) = \boldsymbol{\theta}^{\mathbf{T}}\boldsymbol{\phi}(\mathbf{t})$$

Let's look at this expression in more detail

$$\boldsymbol{\phi}(\mathbf{t}) = \begin{bmatrix} \phi_0(t) \\ . \\ . \\ . \\ \phi_n(t) \end{bmatrix} \quad \boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ . \\ . \\ . \\ \theta_n \end{bmatrix} \quad \boldsymbol{\theta} = [\theta_0, \ldots, \theta_n]^T$$

$$u(t) = \theta_0\phi_0(t) + \ldots + \theta_n\phi_n(t)$$

Example: ax+b

$$y = ax + b$$

$$\boldsymbol{\theta} = [a \quad b]^T$$

$$\phi(\mathbf{x}) = \begin{bmatrix} x \\ 1 \end{bmatrix}$$

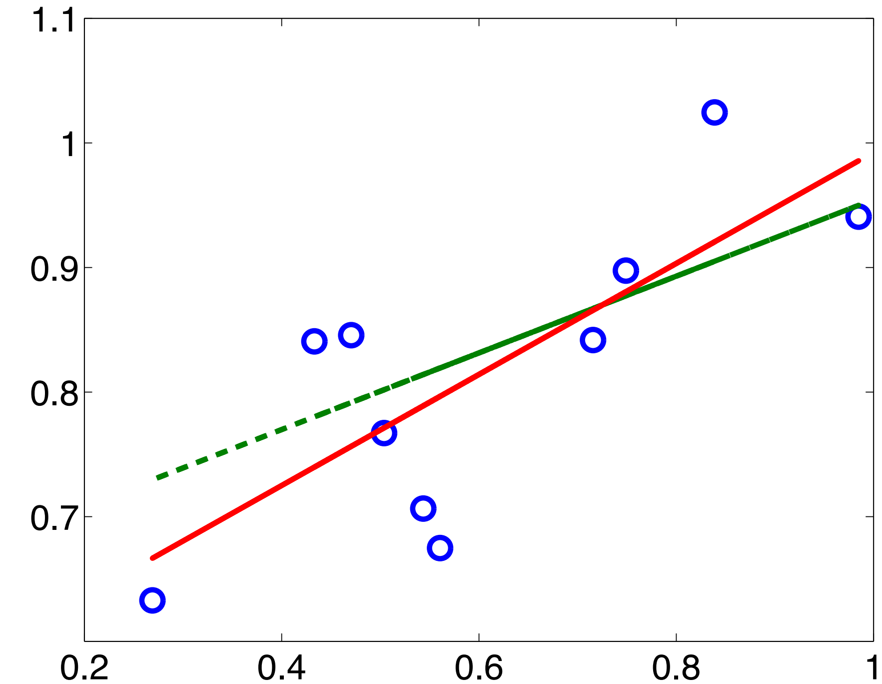
Example: ax+b

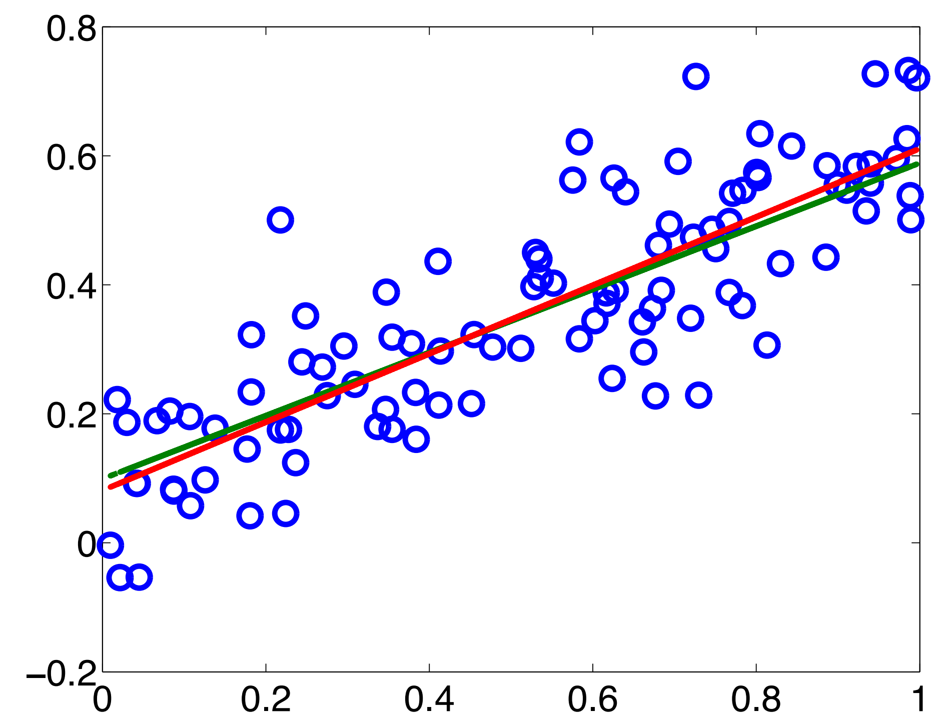
```
% how many observations
p = 10;

x = rand(p,1);
b1 = rand
b2 = rand
eps =  0.1*randn(p,1);
t = b2*x+b1 + eps;
x1 = [x,ones(p,1)];

b_est = pinv(x1)*t

h = plot(x,t,'o',x,t-eps,x,b_est'*x1');
```





Try p = 10

Try p = 100

Basis functions

Goal: function approximation of arbitrary functions
Wanted: 'Good' function approximator
Good: easy to find parameters, expressive, ...

$$f(x) = \sum_i w_i \Psi_i(x)$$

Idea: push nonlinearity into the basis functions, independent of parameters

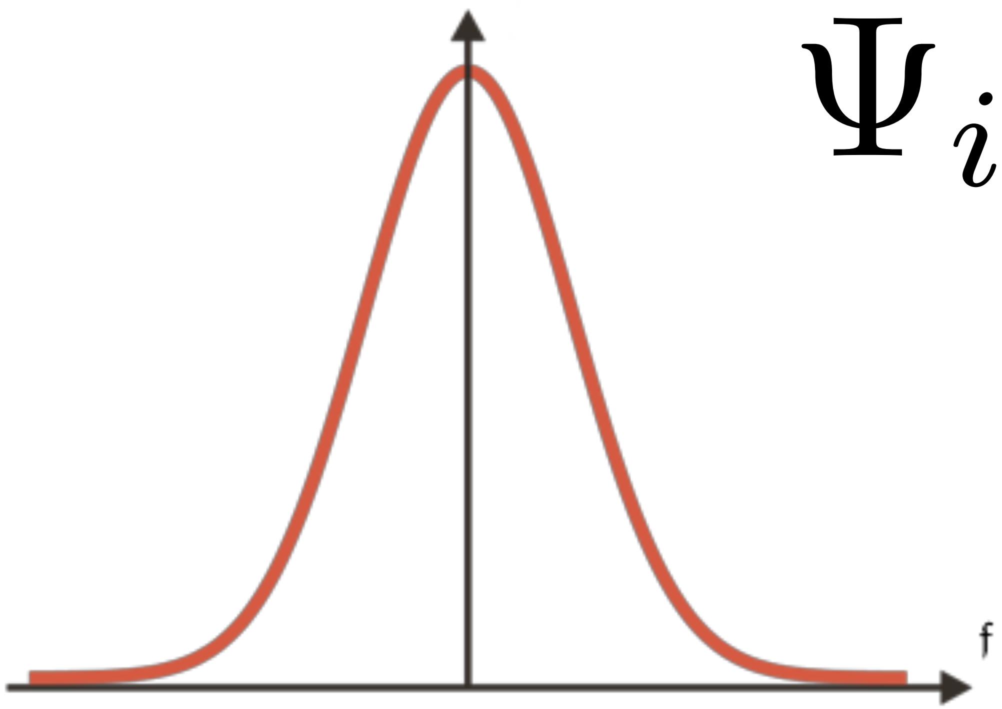
Linear in parameters!!!!!

Fourier basis

$$\frac{a_0}{2} + \sum_{n=1}^{\infty} [a_n \cos(nx) + b_n \sin(nx)]$$

Problem: infinite validity, need infinitely many basis functions to approximate a non-periodic function

Gaussian basis



$$\Psi_i(x) = e^{-\frac{(x-c_i)^2}{b_i}}$$

Idea: push nonlinearity into the basis functions

basis: more localized than sines

$$f(x) = \sum_i w_i \Psi_i(x)$$

Note: can take derivatives of any order of the gaussian basis...

Polynomial basis

$$f(x, \theta) = \theta_0 + \theta_1 x + \theta_2 x^2 + \dots$$

$$f(x, \theta) = \sum_{n=0}^{\infty} \theta_n x^n$$

Other bases

$$f(x,\theta) = \sum_{n=0}^{\infty} \theta_n b_n(x)$$

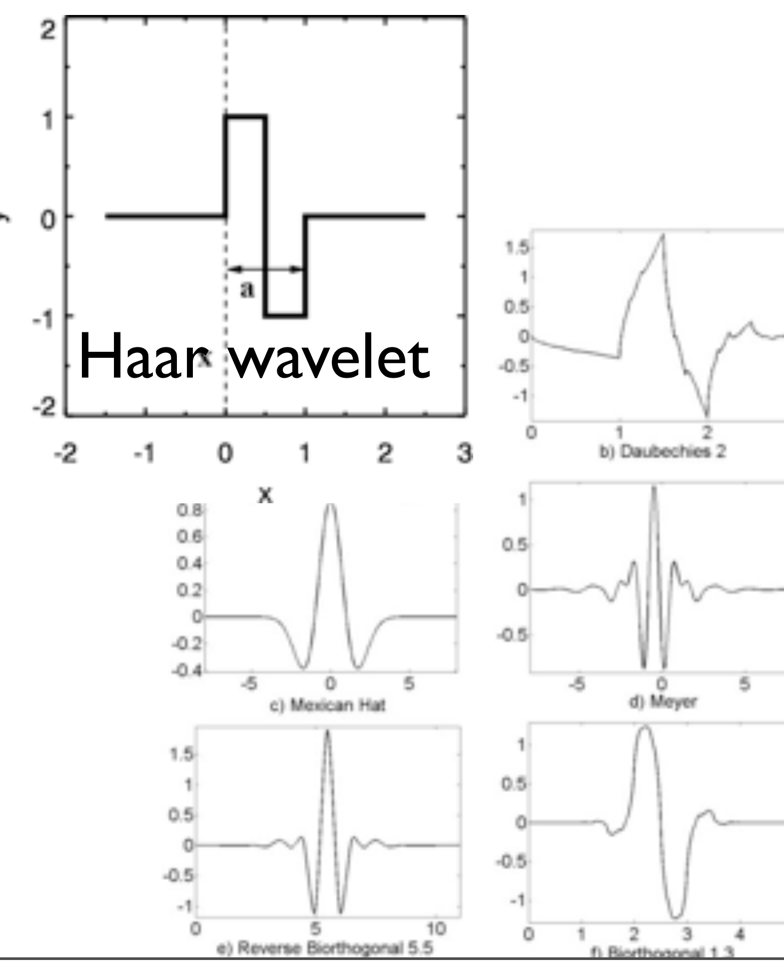
- Index functions

$$b(x) = 1 \; \forall x \in [a, b], \quad 0 \text{ otherwise}$$

- Index functions multiplied with other functions (e.g. local linear models)

$$b(x) = g(x) \; \forall x \in [a, b], \quad 0 \text{ otherwise}$$
$$b(x) = x \; \forall x \in [a, b], \quad 0 \text{ otherwise}$$

- Wavelets: lots of different bases



Basis functions can be defined on $\mathbb{C}$

The function basis zoo

All bases are equal,
but some bases are more equal than others!

Fit nonlinear functions

Linear model: $y(\mathbf{x}, \mathbf{w}) = \sum_{j=0}^{M-1} w_j \phi_j(\mathbf{x}) = \mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x})$

Gaussian basis function: $\phi_j(x) = \exp\left\{-\frac{(x-\mu_j)^2}{2s^2}\right\}$

Observation: $t = y(\mathbf{x}, \mathbf{w}) + \epsilon$

$p(t|\mathbf{x}, \mathbf{w}, \beta) = \mathcal{N}(t|y(\mathbf{x}, \mathbf{w}), \beta^{-1})$

precision (inv. variance) β

Max. likelihood treatment leads to:

$$\begin{aligned} \ln p(\mathbf{t}|\mathbf{w}, \beta) &= \sum_{n=1}^{N} \ln \mathcal{N}(t_n|\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n), \beta^{-1}) \\ &= \frac{N}{2}\ln\beta - \frac{N}{2}\ln(2\pi) - \beta E_D(\mathbf{w}) \end{aligned}$$

$$E_D(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{t_n - \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)\}^2.$$

Gradient: $\nabla \ln p(\mathbf{t}|\mathbf{w}, \beta) = \sum \{t_n - \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)\}\, \boldsymbol{\phi}(\mathbf{x}_n)^{\mathrm{T}}$

A D R L

ETH Zürich

Gradient:

$$\nabla \ln p(\mathbf{t}|\mathbf{w}, \beta) = \sum_{n=1}^{N} \left\{ t_n - \mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x}_n) \right\} \boldsymbol{\phi}(\mathbf{x}_n)^{\mathrm{T}}.$$

Gradient = 0

$$0 = \sum_{n=1}^{N} t_n \boldsymbol{\phi}(\mathbf{x}_n)^{\mathrm{T}} - \mathbf{w}^{\mathrm{T}} \left(\sum_{n=1}^{N} \boldsymbol{\phi}(\mathbf{x}_n) \boldsymbol{\phi}(\mathbf{x}_n)^{\mathrm{T}} \right).$$

$$\mathbf{w}_{\mathrm{ML}} = \left(\boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{\Phi} \right)^{-1} \boldsymbol{\Phi}^{\mathrm{T}} \mathbf{t}$$

$$\boldsymbol{\Phi}^{\dagger} \equiv \left(\boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{\Phi} \right)^{-1} \boldsymbol{\Phi}^{\mathrm{T}}$$

Pseudoinverse!

$$\boldsymbol{\Phi} = \begin{pmatrix} \phi_0(\mathbf{x}_1) & \phi_1(\mathbf{x}_1) & \cdots & \phi_{M-1}(\mathbf{x}_1) \\ \phi_0(\mathbf{x}_2) & \phi_1(\mathbf{x}_2) & \cdots & \phi_{M-1}(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(\mathbf{x}_N) & \phi_1(\mathbf{x}_N) & \cdots & \phi_{M-1}(\mathbf{x}_N) \end{pmatrix}$$

NxM

Why not using pseudo-inverse?

What happens if you have millions of datapoints/observations?

... or lots of dimensions in the problem?

$$\mathbf{w} = (\mathbf{\Phi}^{\mathbf{T}}\mathbf{\Phi})^{-1}\mathbf{\Phi}^{\mathbf{T}}\mathbf{t}$$

Google

$$\mathbf{\Phi}^{\#} = (\mathbf{\Phi}^{\mathbf{T}}\mathbf{\Phi})^{-1}\mathbf{\Phi}^{\mathbf{T}}$$

dimensions?

square matrix MxM

(each entry N multiplications)

→ use SVD (found iteratively)

→ Iterative least squares

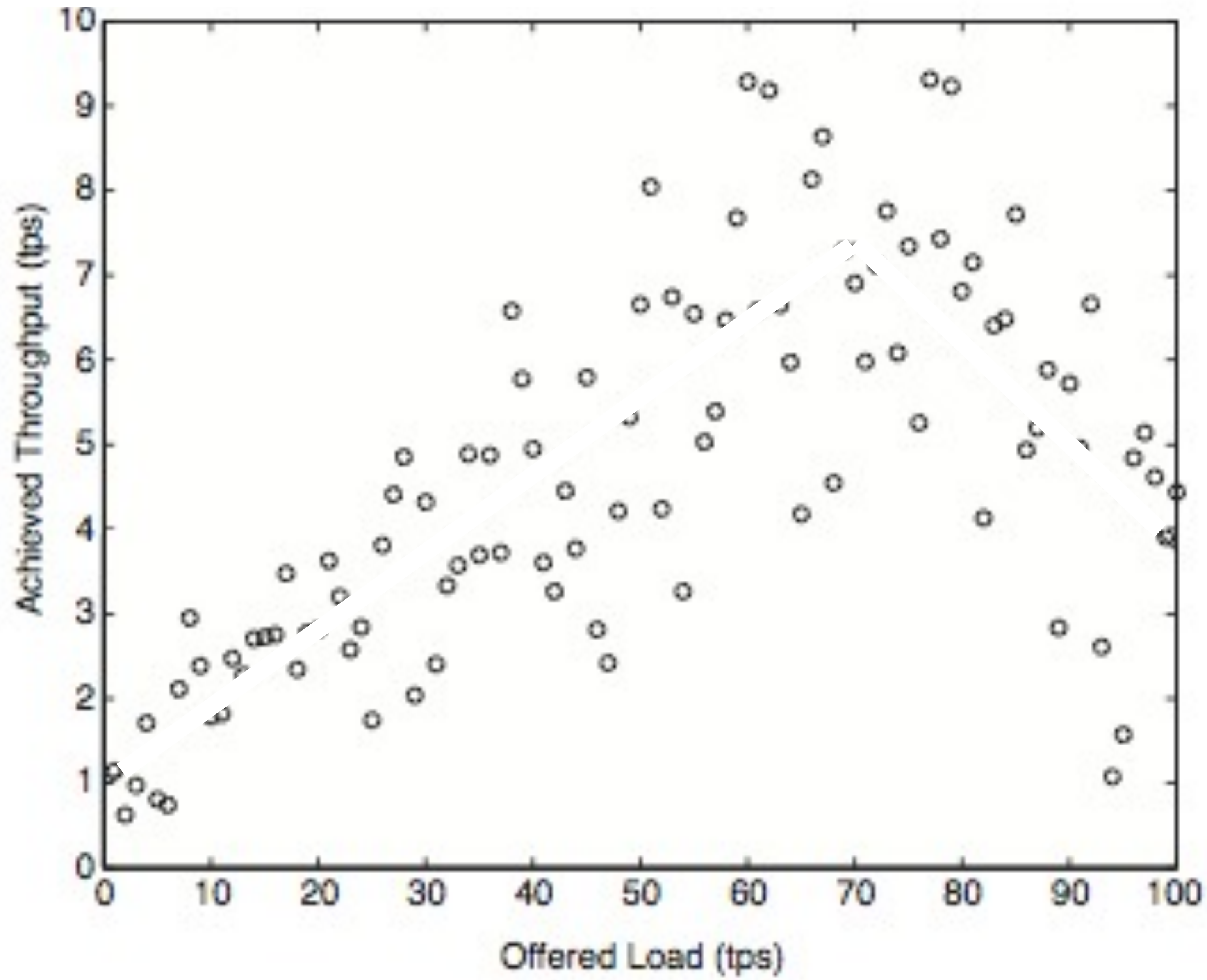
Matlab: `svd`

A D R L

ETH Zürich

What basis function is used in Le Boudec's example?

$$Y_i = (a + bx_i)1_{x_i \leq \xi} + (c + dx_i)1_{\{x_i > \xi\}} + \epsilon_i$$



[Le Boudec, Performance evaluation, EPFL]

Nonlinear fitting

It's easy to fit a linear function

... or anything where the parameters show up linear!!!

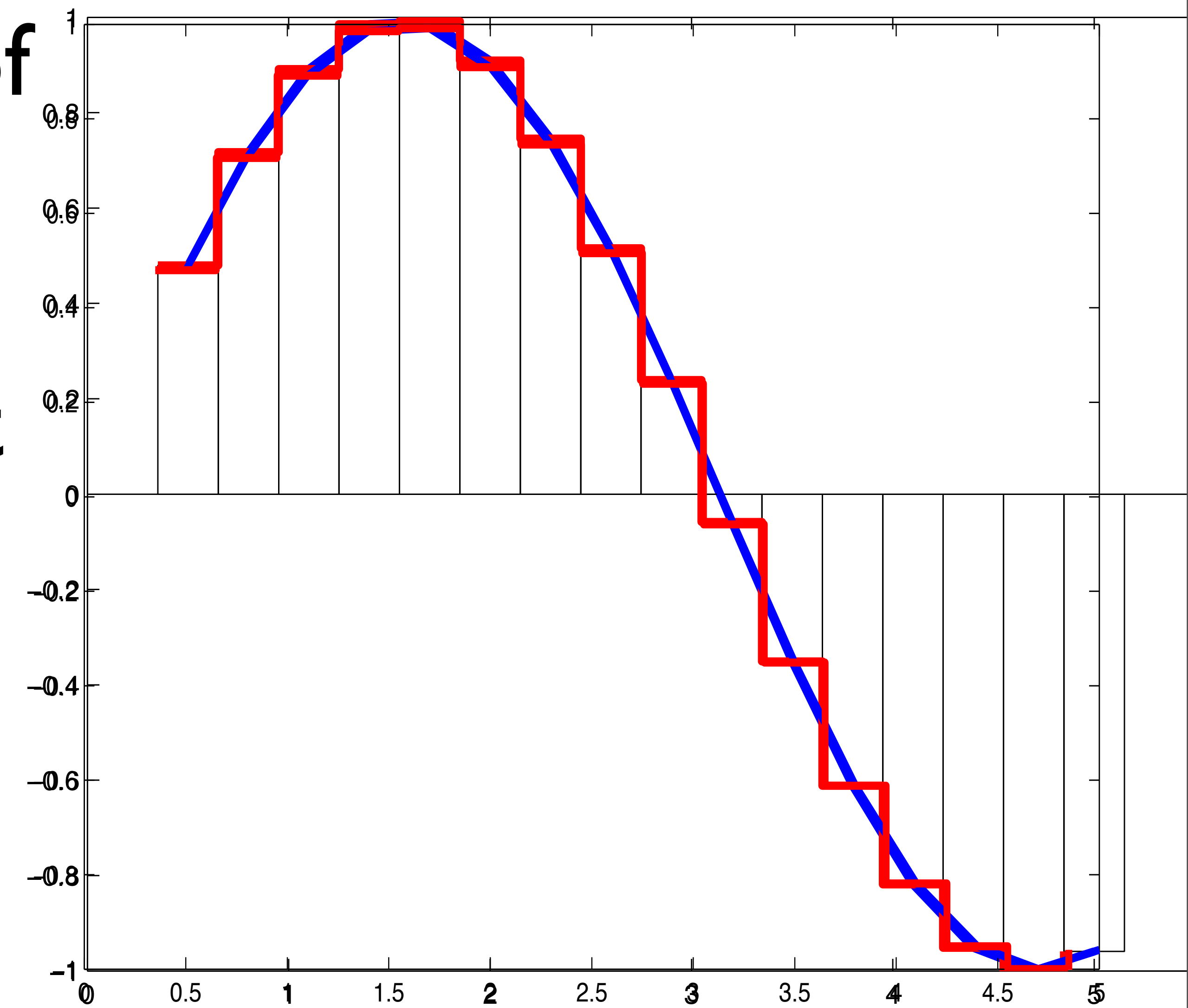
Example: Sampling

Function approximation view of sampling

discretization is special case of function approximation

can make the same argument with dirac pulse sampling

remember: no overlap / no generalization



A D R L

ETH Zürich

Function approximation as ‘bridge’ between continuous and discrete world

Can lower dimensionality of the problem (dimensionality becomes an open parameter!!!)

Tradeoff: high N, good approximation, optimal policy, curse of dimensionality

Approximate what?

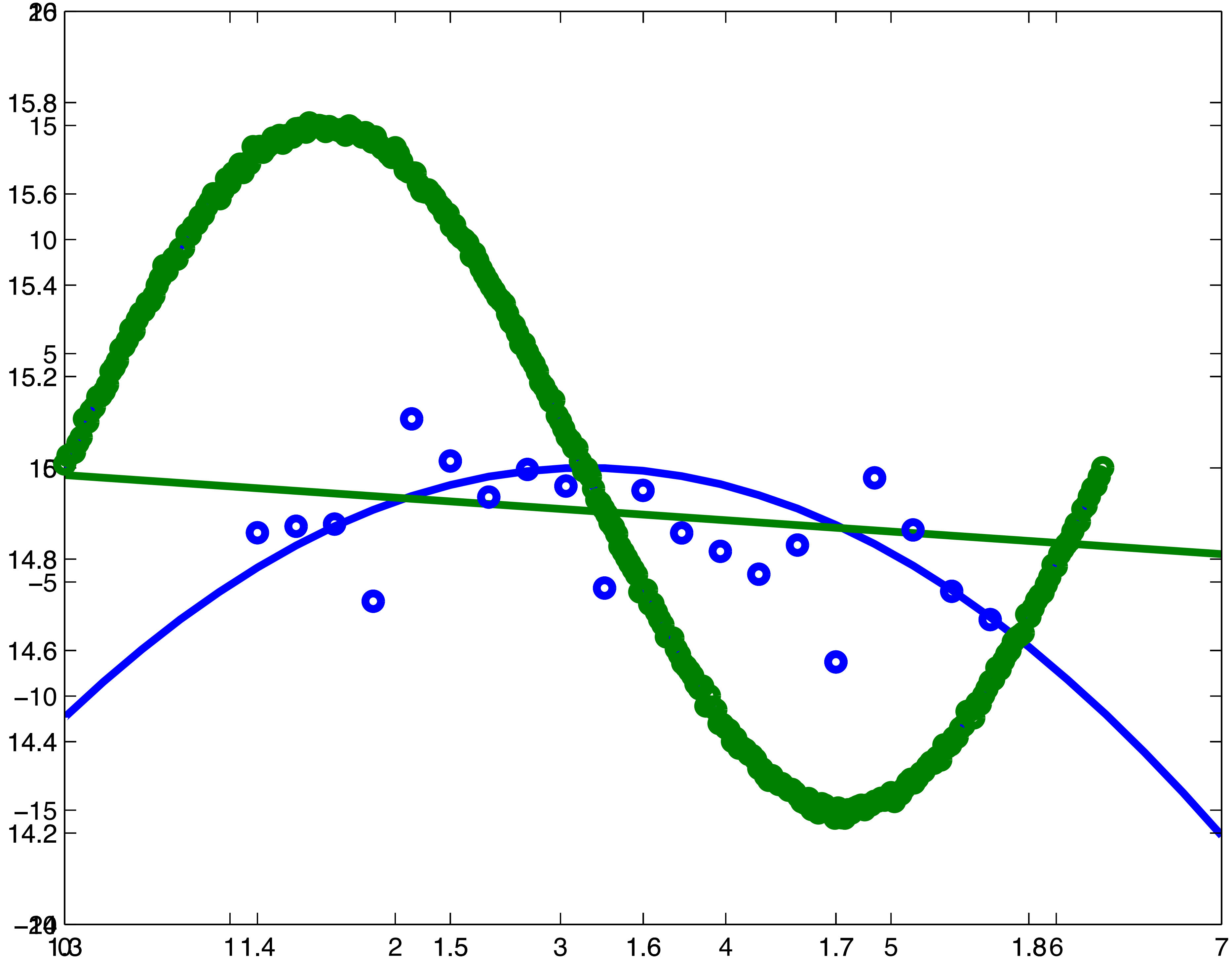
Supervised learning: Target function known → difference is cost

Reinforcement learning: Cost is sampled from examples
But think of it this way: There is a 'target function', i.e. the optimal control/or value function and the approximator has to minimize the distance between the 'guess' and this function

- Direct policy learning (e.g. policy gradients)
- Value function approximation

Some issues...

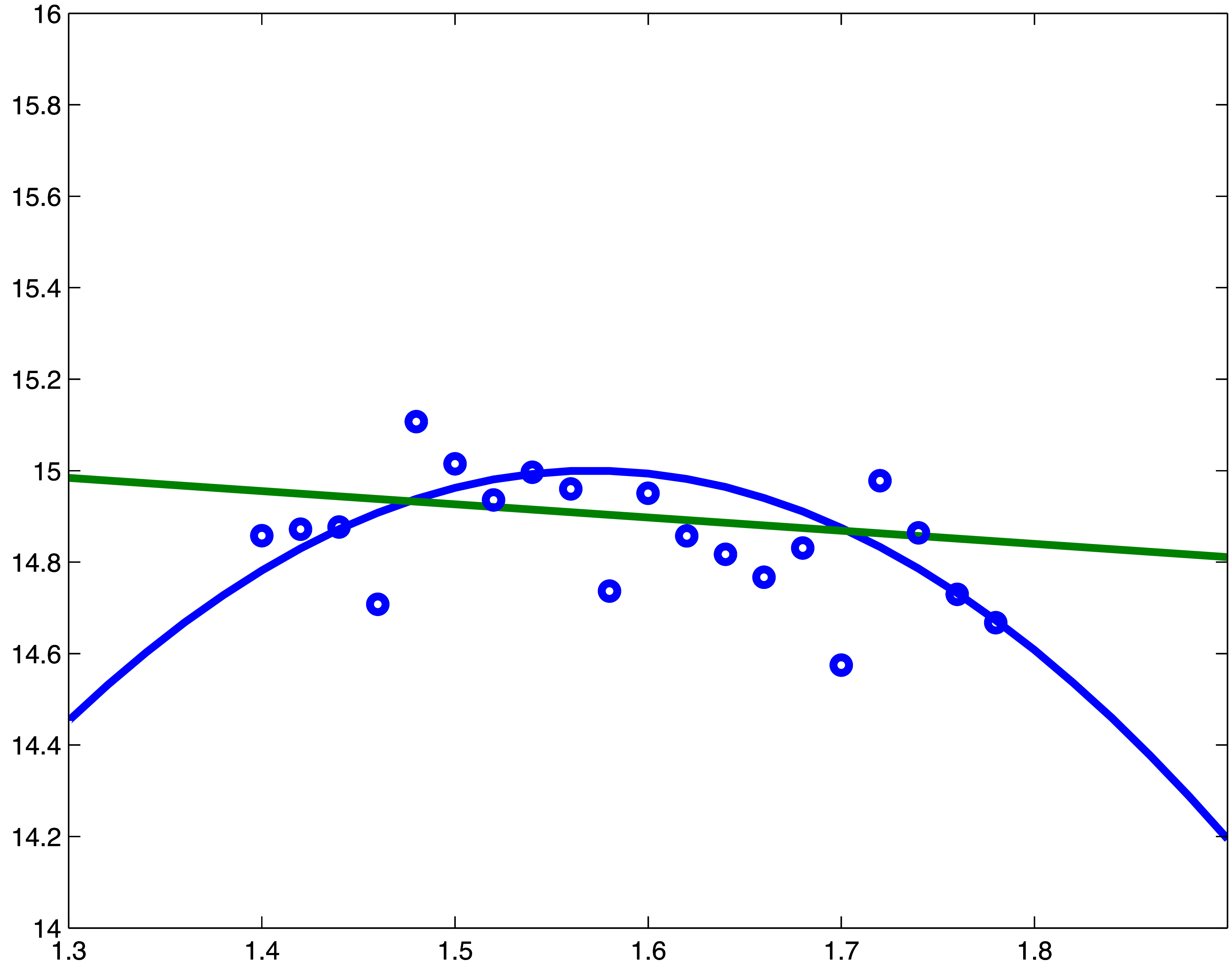
Extrapolation vs. Interpolation



Extrapolate?

Need a model... let's fit one...

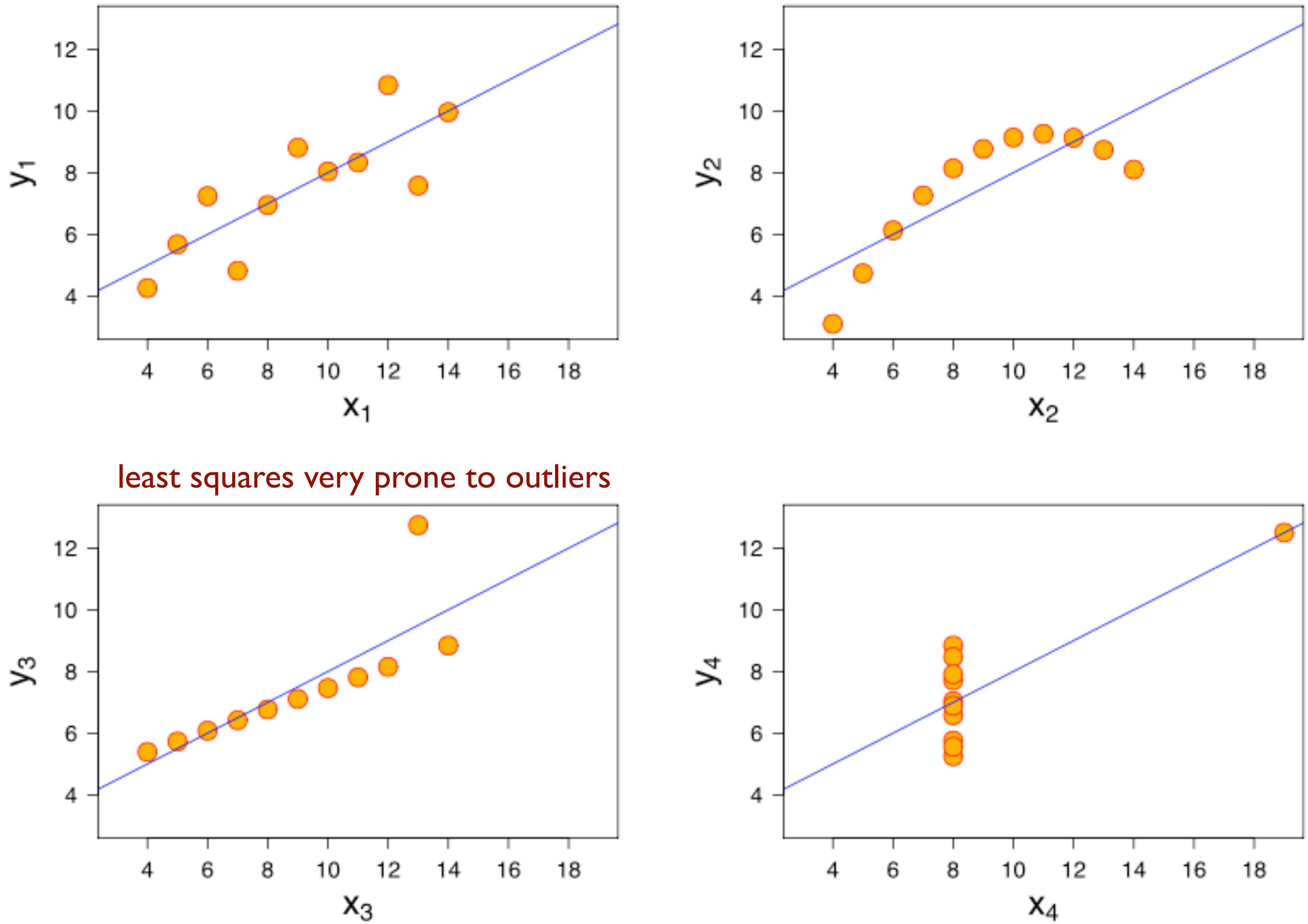
Buchli - OLCAR - 2015



Interpolation?

Model choice, model bias and explanatory power of models

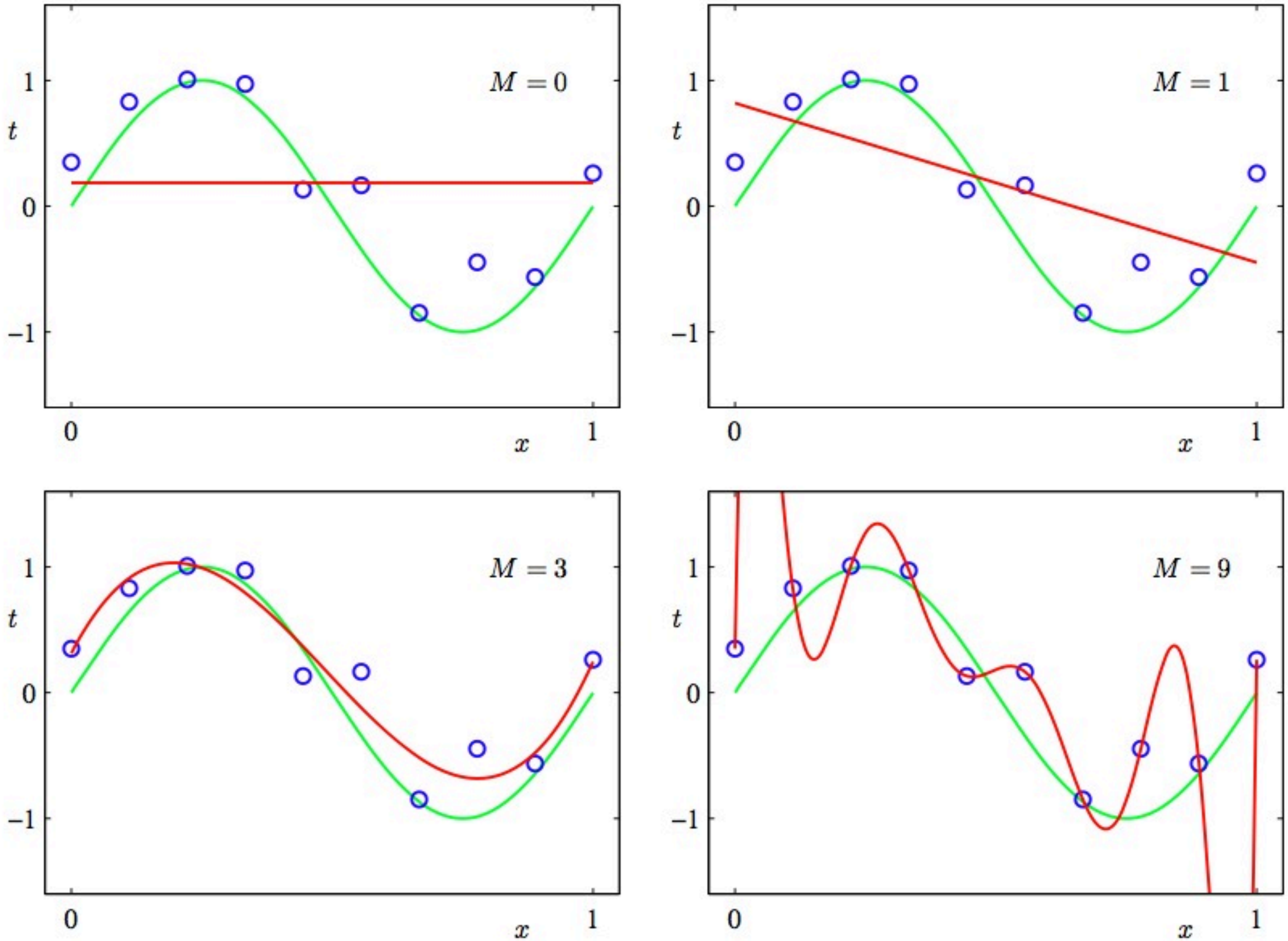
Anscombe's quartet



Overfitting

The approximator has to have enough expressive power to capture the essentials,

... but it should not try to over-interpret the data!

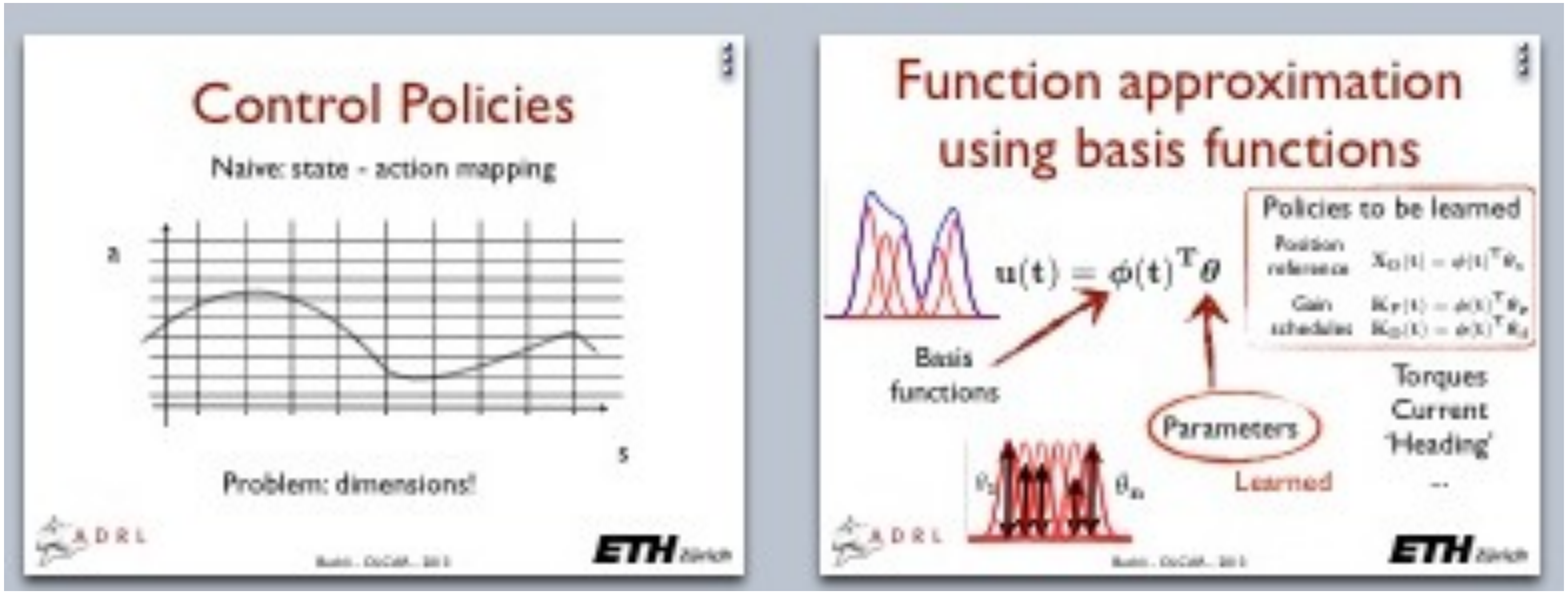

$M = 0$
$M = 1$
$M = 3$
$M = 9$
t
x
1
0
−1
0
1

State vs time based policies

Policy: basis functions

function of states $\Psi(x)$

function of time $\Psi(t)$



‘classic RL’
Sutton & Barto

p 9 on efficiency of ‘evolutionary’ vs. direct value based methods

p17 on how opt. control IS learning and emphasis on ‘incremental’!

Parametrized Policy Learning (applied RL in Robotics in last ~10y)

see Examples/Papers

⇔ Planning

Path integral RL

Goal: Solve continuos time stochastic optimal control problem by sampling

⇒ First look at Path Integral Optimal Control

(Reminder from L3)

Reminder
Continuos time optimal control

A D R L

ETH Zürich

(Reminder from L3)

Continuous time system

System dynamics

$$\dot{\mathbf{x}}(t) = \mathbf{f}_t(\mathbf{x}(t), \mathbf{u}(t))$$

Cost

$$J = e^{-\beta(t_f - t_0)}\Phi(\mathbf{x}(t_f)) + \int_{t_0}^{t_f} e^{-\beta(t-t_0)} L(\mathbf{x}(t), \mathbf{u}(t))dt$$

$0 \leq \beta$ discount / decay rate

'exponential decay'

(Reminder from L3)

Continuous time optimal control problem

Find control $u^*(t) = \mu^*(t, x(t))$ minimizing

control (input) policy

$$J = e^{-\beta(t_f - t_0)}\Phi(\mathbf{x}(t_f)) + \int_{t_0}^{t_f} e^{-\beta(t-t_0)} L(\mathbf{x}(t), \mathbf{u}(t))dt$$

Given constraints

$$\dot{\mathbf{x}}(t) = \mathbf{f}_t(\mathbf{x}(t), \mathbf{u}(t))$$

Goal: Optimal policy

$$\mu^* = \arg\min_u J$$

(Reminder from L3)

Hamilton Jacobi Bellman Equation

Carl Gustav Jacob Jacobi
(1804-1851)

William Rowan Hamilton
(1805-1865)

Richard Bellman
1920-84

$$\frac{\partial V^*}{\partial t} = \beta V^* - \min_{\mathbf{u}\in\mathbf{U}} \left\{ L(\mathbf{x},\mathbf{u}) + \left(\frac{\partial V^*}{\partial \mathbf{x}}\right)^T \mathbf{f}(\mathbf{x},\mathbf{u}) \right\}$$

$$\beta V^* - \frac{\partial V^*}{\partial t} = \min_{\mathbf{u}\in\mathbf{U}} \left\{ L(\mathbf{x},\mathbf{u}) + \left(\frac{\partial V^*}{\partial \mathbf{x}}\right)^T \mathbf{f}(\mathbf{x},\mathbf{u}) \right\}$$

In general: Nonlinear, Partial Differential Equation

Has no analytical solution... : (

Backwards in time! $V^*(t_f, \mathbf{x}) = \Phi(\mathbf{x})$

(Reminder from L3)

Reminder
Stochastic optimal control

(Reminder from L3)

Stochastic system

$$\dot{\mathbf{x}}(t) = \mathbf{f}_t(\mathbf{x}(t), \mathbf{u}(t)) + \mathbf{B}(t)\mathbf{w}(t), \quad \mathbf{x}(0) = \mathbf{x}_0$$

Mean: $E[\mathbf{w}(t)] = \bar{\mathbf{w}} = 0$ — mean-free

Co-variance: $E[\mathbf{w}(t)\mathbf{w}(\tau)^T] = \mathbf{W}(t)\delta(t-\tau)$ — uncorrelated over time

$$E[\mathbf{w}(t)\mathbf{w}(\tau)^T] = 0 \quad t \neq \tau$$

Expected cost:

$$J = E\left\{e^{-\beta(t_f - t_0)}\Phi(\mathbf{x}(t_f)) + \int_{t_0}^{t_f} e^{-\beta(t' - t_0)} L(\mathbf{x}(t'), \mathbf{u}(t'))dt'\right\}$$

A D R L

Stoch. HJB

(Reminder from L3)

$$\beta V^*(t,\mathbf{x}) - V_t^*(t,\mathbf{x}) = \min_{\mathbf{u}(t)} \left\{ L(\mathbf{x},\mathbf{u}(t)) + V_{\mathbf{x}}^{*T}\mathbf{f}_t(\mathbf{x},\mathbf{u}(t)) + \frac{1}{2}\mathrm{Tr}[V_{\mathbf{xx}}^*\mathbf{B}(t)\mathbf{W}(t)\mathbf{B}^T(t)] \right\}$$

Hamilton Jacobi Bellman Equation

$$V^*(t_f,\mathbf{x}) = \Phi(\mathbf{x})$$

A D R L

Towards Path-integral SOC

Goal: Solve continuos time stochastic optimal control problem by sampling

- ★ Can solve SOC with certain assumptions, eg. LQ.
- ★ Here a less stringent form of the dynamics (ctl. affine) and (almost) NO assumptions on cost

Why important?

Cost is key design 'handle' to tell the system what to do...

Control affine opt control problem

Note similarity to stoch. system (L3)

$$\dot{\mathbf{x}}(t) = \mathbf{f}_t(\mathbf{x}(t), \mathbf{u}(t)) + \mathbf{B}(t)\mathbf{w}(t)$$

$$\dot{\mathbf{x}}_t = \mathbf{f}(\mathbf{x}_t, t) + \mathbf{G}(\mathbf{x}_t)\mathbf{u}_t$$

$$V(\mathbf{x}_0) = \min_{\mathbf{u}_{t_0:t_N}} \mathbb{E}\left[R(\tau)\right]$$

$$R(\tau) = \phi(t_N) + \int_{t_0}^{t_N} r_t dt \qquad r_t = q(\mathbf{x}_t) + \frac{1}{2}\mathbf{u}_t^T R \mathbf{u}_t$$

$$V(x_t) = \min_{u_t} E_\tau[R(\tau)]$$

$$\int p(\tau)R(\tau)d\tau \qquad \int p(\tau)\left(\frac{1}{\lambda}\phi + \frac{1}{\lambda}\int rdt\right)d\tau$$

Discretize and use EM-like idea: PoWER - Problem: pseudo-probability - restriction on cost function

Other idea: Treat probability as a diffusion process - Connection with statistical physics Forward dynamics! Sampling (Monte Carlo)!

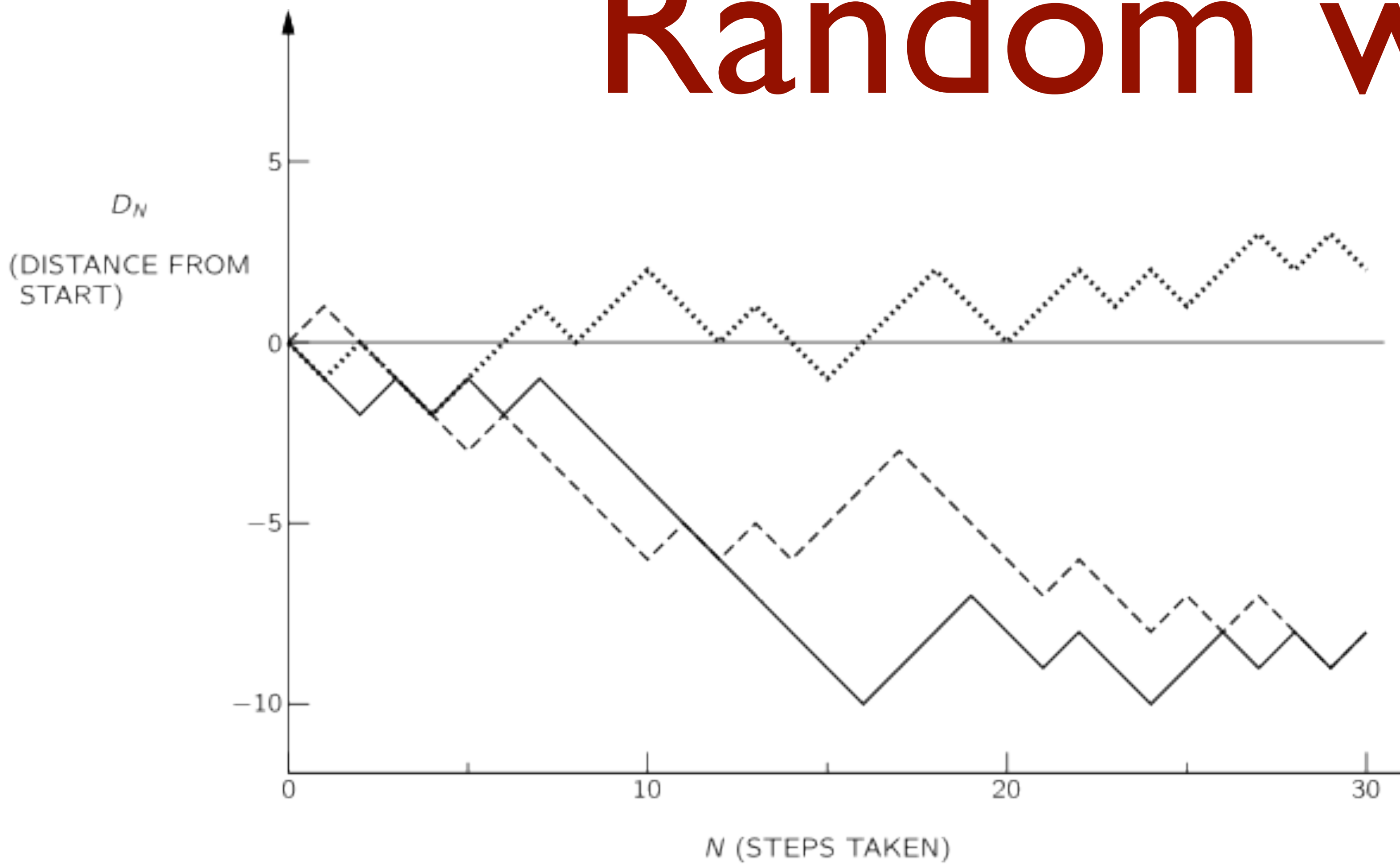
Continuous Random Processes

$$d\mathbf{x} = \mathbf{f}\left(\mathbf{x}, \mathbf{u}\right) dt + F\left(\mathbf{x}, \mathbf{u}\right) d\boldsymbol{\omega}$$

Major difficulty: Definition of stochastic processes in continuous time!

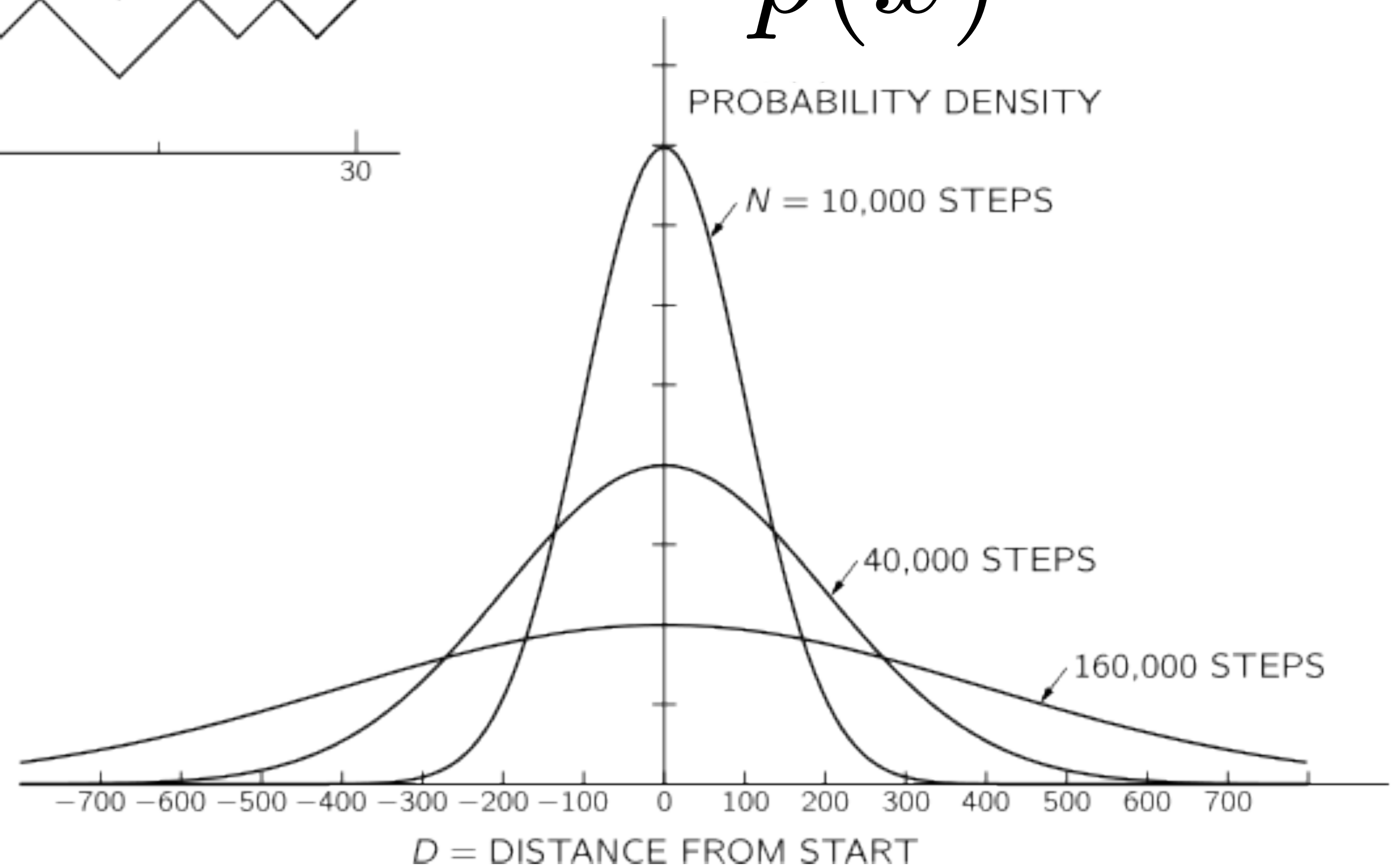
		states	
		discrete	continuous
time	discrete	$dx \to 0$ $\int \to \sum$	$\int \to \sum$ $dt \to 0$
	continuous		Lots of gnarly math!

Random walks



'paths diffuse over time'

$p(x)$



'density'

A D R L

ETH Zürich

Expectations over paths

$$\Psi_{t_i} = E_{\tau_i}\left(\Psi_{t_N} e^{-\int_{t_i}^{t_N} \frac{1}{\lambda} q_t dt}\right) = E_{\tau_i}\left[\exp\left(-\frac{1}{\lambda}\phi_{t_N} - \frac{1}{\lambda}\int_{t_i}^{t_N} q_t\, dt\right)\right]$$

forward!

... but stochastic

$$\int p(\tau) \exp\left(-\frac{1}{\lambda}\phi - \frac{1}{\lambda}\int q dt\right) d\tau$$

$$\tau = x(t \ldots t_N) \sim p(x, u)$$

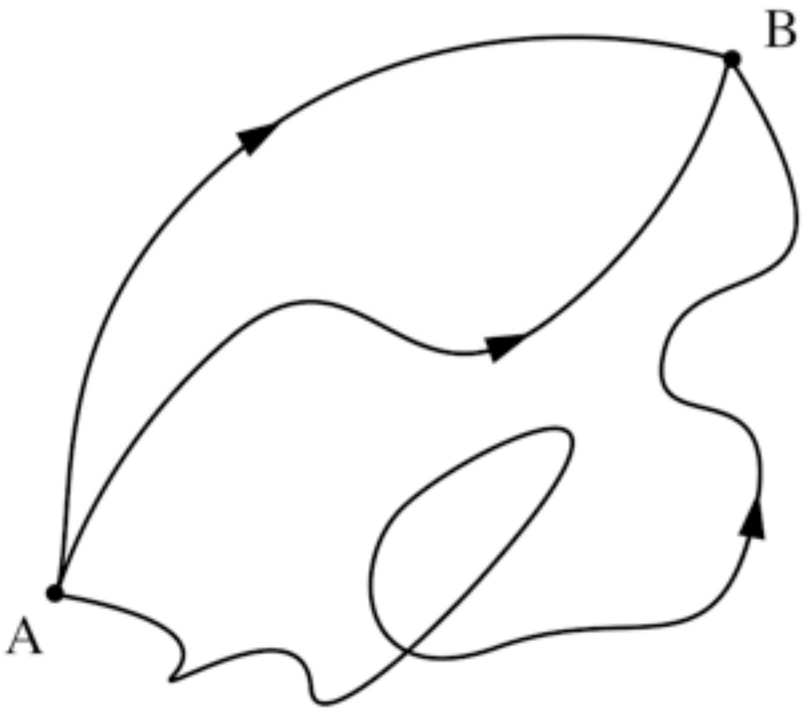
an instance of a random path segment (a random 'number', but in spaces of functions)

$$E[X] = \int x p(x) dx$$

Continuous time, x is function of time $x = f(t)$

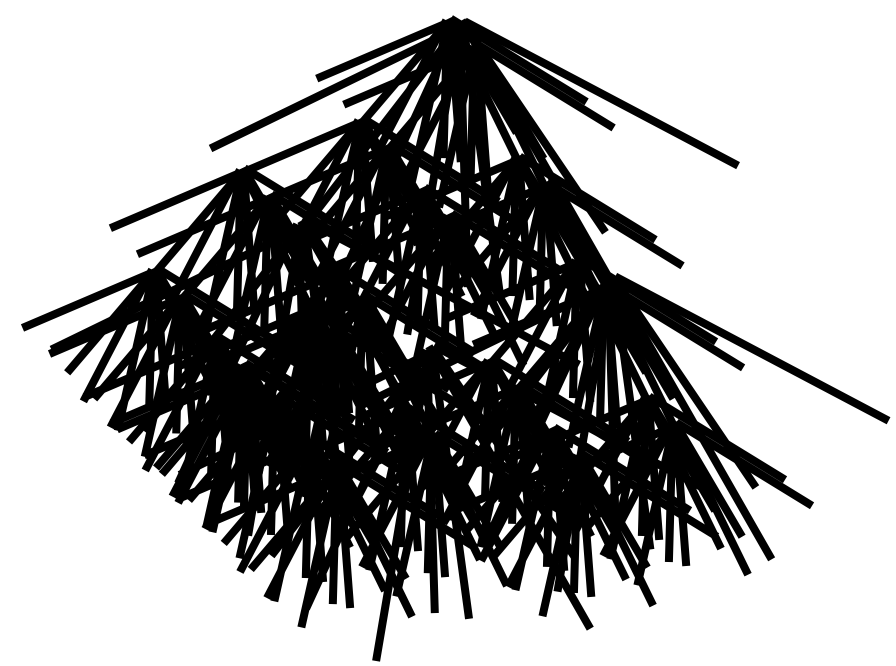
Comparison to graphs

can think of all possibilities of a random walk as graph



There are several ways to end up in a certain state, each path has an associated probability

When does 'branching' occur?



Idea: do discrete time and take limit

$$dt \to 0$$

ADRL

ETH Zürich

Continuous decision processes

Take random walk and take limits

$dx \to 0$ probability densities

$dt \to 0$ probability flow

for all times $\int p(x)dx = 1$ conservation law!

Conserved flow?

You know how to do that!

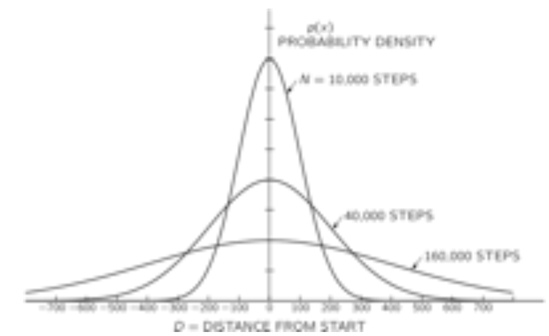
A D R L

ETH Zürich

Pre-requisite 1: Brownian motion

Assume process with probability distribution

$$\mathbb{P}_{\mathbf{w}}(t,w) = \frac{1}{\sqrt{2\pi\sigma^2 t}} \exp\left(-\frac{(w-\mu t)^2}{2\sigma^2 t}\right)$$



at any time:

$$\mathbb{E}\{w(t)\} = \mu t$$

$$\mathbb{V}ar\{w(t)\} = \sigma^2 t$$

Defined via increment process:

$$dw(t) = \lim_{\Delta t \to 0} w(t+\Delta t) - w(t)$$

1. The increment process, $dw(t)$, has a Gaussian distribution with the mean and the variance, $\mu\Delta t$ and $\sigma^2\Delta t$ respectively.

2. The increment process, $dw(t)$, is statistically independent of $w(s)$ for any $s \leq t$.

A D R L

ETH Zürich

Simulate Brownian Motion

Integrate discretized increment process:

$$w(t+\Delta t) = w(t) + \mu \Delta t + \sqrt{\Delta t \sigma^2}\varepsilon, \qquad w(0) = 0$$

$$\varepsilon \sim N(0,1)$$

... this is a discretized SDE

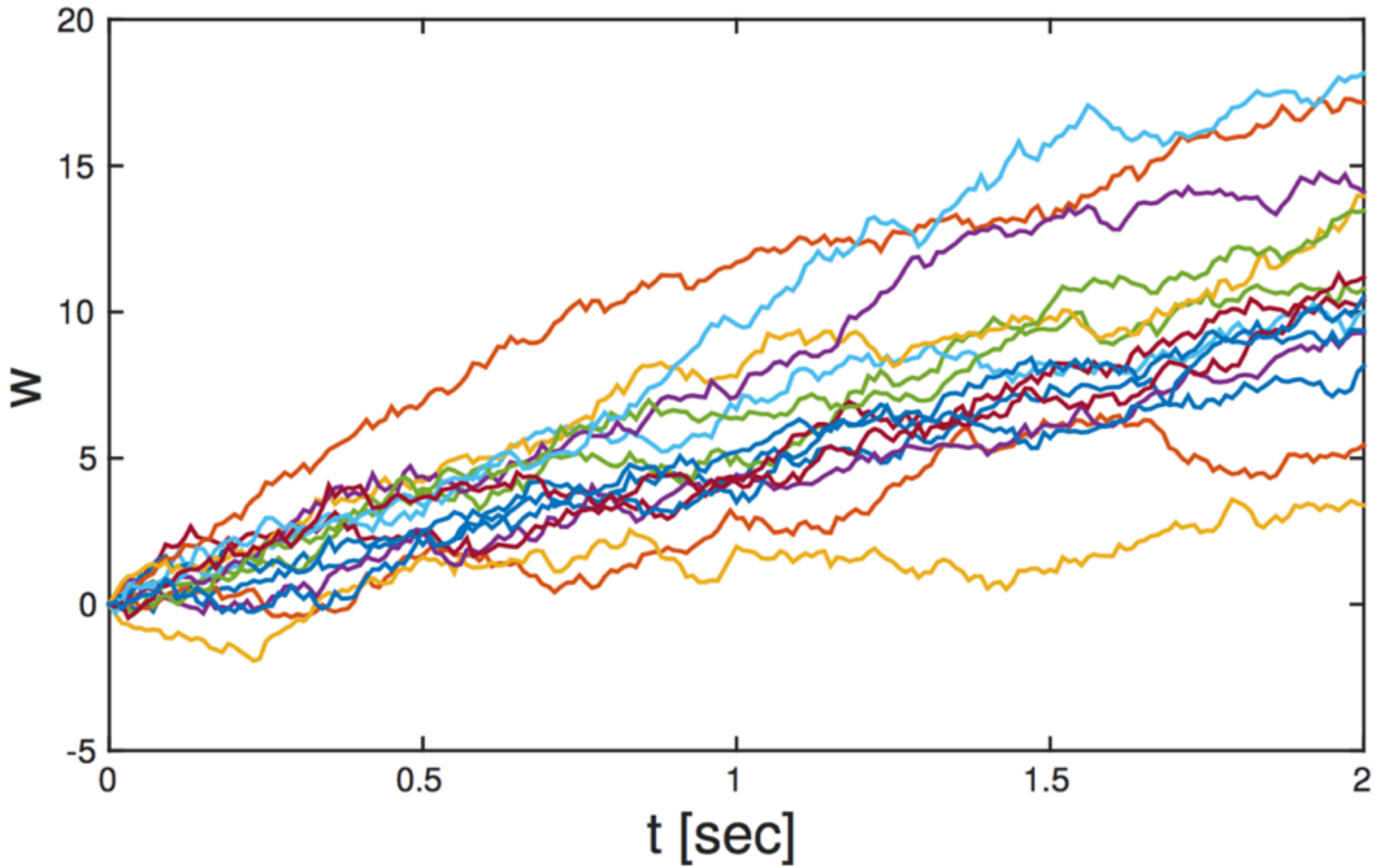


Figure 3.1: Brownian Motion with $\mu = 5$ and $\sigma^2 = 4$

Pre-requisite 2: Stochastic Differential Equations (SDE)

SDE: $$d\mathbf{x} = \mathbf{f}(t, \mathbf{x})dt + \mathbf{G}(t, \mathbf{x})d\mathbf{w}$$

$\mathbf{w}(t)$ Brownian motion (Wiener Process) (zero mean, covariance = I)

SDE Integration

$$d\mathbf{x} = \mathbf{f}(t, \mathbf{x})dt + \mathbf{G}(t, \mathbf{x})d\mathbf{w}$$

Numerical integration:

use: small time step Δt $d\mathbf{w} = \sqrt{\Delta t}\varepsilon$

assumes constant increment: $w(t + \Delta t) = w(t) + \mu\Delta t + \sqrt{\Delta t\sigma^2}\varepsilon, \quad w(0) = 0$

$$\mathbf{x}(t_{n+1}) = \mathbf{x}(t_n) + \mathbf{f}(t_n, \mathbf{x}(t_n))\Delta t + \mathbf{G}(t_n, \mathbf{x}(t_n))\sqrt{\Delta t}\varepsilon, \quad \varepsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

Process is nonlinear through f(t,x): Non-gaussian

But, for $\Delta t \to 0$

$$\mathbb{P}_{\mathbf{x}}(t + \Delta t, \mathbf{x} \mid t, \mathbf{y}) = \mathcal{N}\Big(\mathbf{y} + \mathbf{f}(t, \mathbf{y})\Delta t, \mathbf{G}(t, \mathbf{y})\mathbf{G}^T(t, \mathbf{y})\Delta t\Big)$$

Probabilistic Dynamics

Discrete time:

Markov chains

☞ Master Equation

Continuous time:

Jumps: Continuous-time Markov chain

Smooth: Markov Process

☞ Fokker-Planck

Pre-requisite 3: The Fokker-Planck Equation

Time evolution of probability distribution?
(Not a Gaussian process)

Probability distribution is solution of a nonlinear PDE: The Fokker-Planck Equation

Fokker-Planck Equation

formal definitions

Assuming stochastic process: $d\mathbf{x} = \mathbf{f}(t, \mathbf{x})dt + \mathbf{G}(t, \mathbf{x})d\mathbf{w}$

$\mathbb{P}_{\mathbf{x(t)}}(t = s, \mathbf{x} \mid \tau, \mathbf{y})$ conditional probability distribution of x(t)

given that process at initial time s has value $\mathbf{y} = \mathbf{x}(s)$

Time evolution of prob. distribution:

$$\partial_t \mathbb{P} = -\nabla_x^T(\mathbf{f}\mathbb{P}) + \frac{1}{2}\mathrm{Tr}\left[\nabla_{xx}(\mathbf{G}\mathbb{P})\right]$$

$$\nabla_x() = \begin{bmatrix} \frac{\partial}{\partial x_1} \\ \frac{\partial}{\partial x_2} \\ \vdots \\ \frac{\partial}{\partial x_n} \end{bmatrix} \qquad \nabla_{xx}() = \begin{bmatrix} \frac{\partial}{\partial x_{11}} & \cdots & \frac{\partial}{\partial x_{1n}} \\ \vdots & \ddots & \vdots \\ \frac{\partial}{\partial x_{n1}} & \cdots & \frac{\partial}{\partial x_{nn}} \end{bmatrix}$$

A D R L

ETH Zürich

Fokker-Planck Equation

PDE for time evolution of probability distribution

Diffusion

$$\frac{\partial}{\partial t}p(x,t) = -\frac{\partial}{\partial x}[\mu(x,t)p(x,t)] + \frac{\partial^2}{\partial x^2}[D(x,t)p(x,t)]$$

Drift

$$dx = \mathbf{f}(x,t)dt + \mathbf{G}(x)d\omega$$

conservation law!

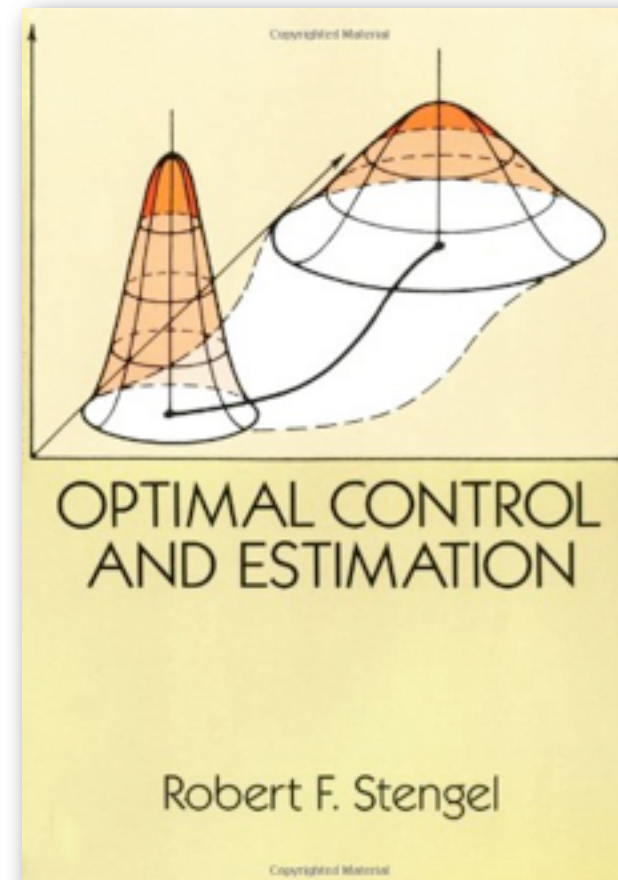
$$\int p(x)dx = 1$$

brownian motion, no drift $\mathbf{f}(\mathbf{x}_t, t)dt = 0$ $\mathbf{G}(\mathbf{x}_t) = 1$

$$d\mathbf{x} = d\omega$$

$$\Rightarrow \quad \frac{\partial p(x,t)}{\partial t} = \frac{1}{2}\frac{\partial^2 p(x,t)}{\partial x^2}$$

$$\Rightarrow p(x,t) = \frac{1}{\sqrt{2\pi t}}e^{-\frac{x^2}{2t}}$$



A D R L

[BB EXAMPLE]

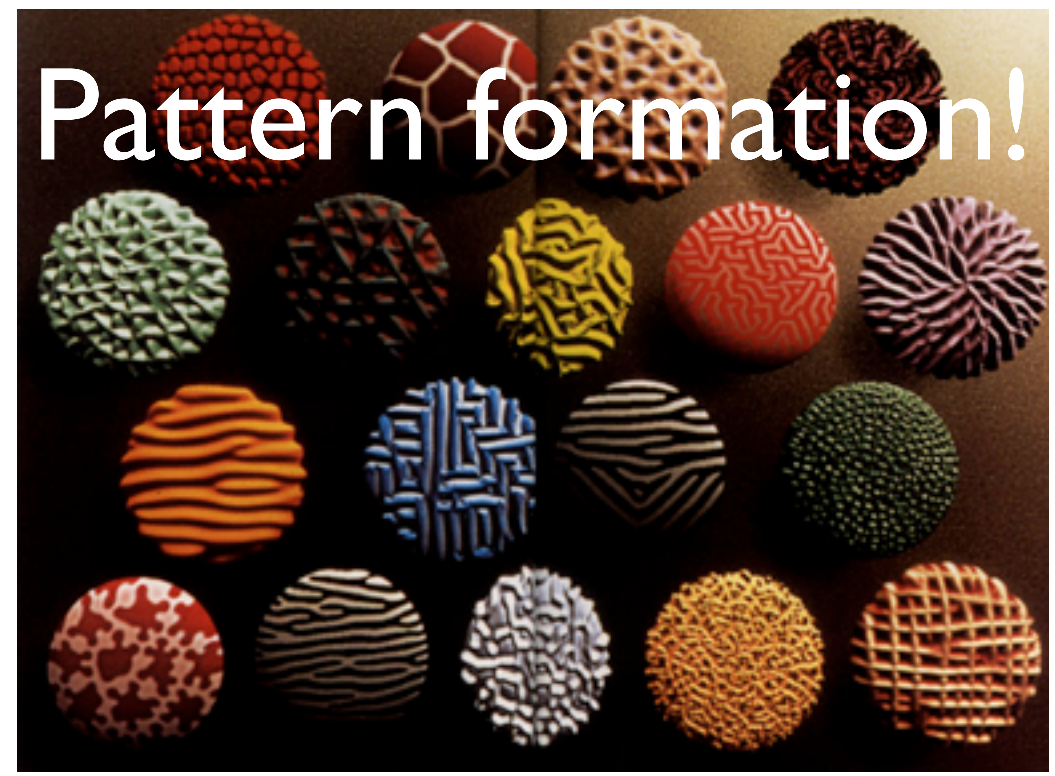
Fokker-Planck Equation

PDE for time evolution of probability distribution

$$\frac{\partial}{\partial t}p(x,t) = -\frac{\partial}{\partial x}[\mu(x,t)p(x,t)] + \frac{\partial^2}{\partial x^2}[D(x,t)p(x,t)]$$

Drift Diffusion

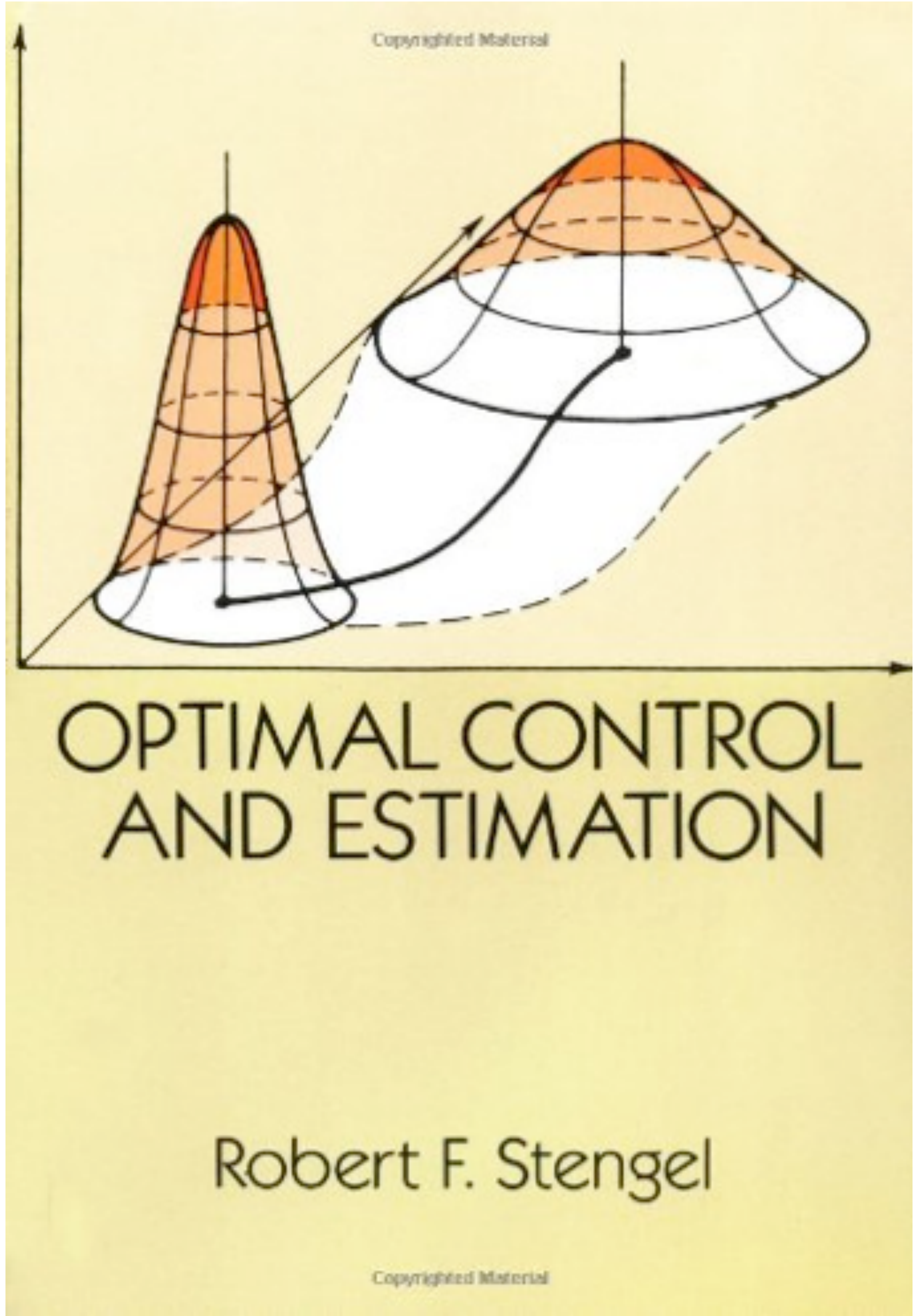
cf. Fluid Dynamics
Heat and Charge diffusion
cf. Particle filters



Finance, Biology, Chemistry, Physics, Sociology, Anthropology, Control & Machine Learning

Stochastic Control

'Controlled Diffusion'
Controlled Brownian Motion



Fokker-Planck modeling is conceptually very useful

... e.g. to develop algorithms

But, naively applying the concept is often not practical

Linearly-Solvable Markov Decision Process

Class of stochastic optimal control problems for which HJB is linear (in Value)

From SOC to LMDP

Assume SDE with control/noise affine form

$$d\mathbf{x} = \mathbf{f}(t,\mathbf{x})dt + \mathbf{g}(t,\mathbf{x})\left(\mathbf{u}dt + d\mathbf{w}\right), \qquad d\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}dt)$$

divide by dt substitute $\frac{d\mathbf{w}}{dt}$ by ε

$$\dot{\mathbf{x}} = \mathbf{f}(t,\mathbf{x}) + \mathbf{g}(t,\mathbf{x})\left(\mathbf{u} + \varepsilon\right), \qquad \varepsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma})$$

Cost $$J = E\left\{\Phi(\mathbf{x}(t_f)) + \int_{t_0}^{t_f} q(t,\mathbf{x}) + \frac{1}{2}\mathbf{u}^T\mathbf{R}\mathbf{u}\, dt\right\}$$

Not LMDP (nonlinear HJB)

Towards linear HJB

of control affine SOC problem

Use HJB for general stoch. cont. time control problem

$$\beta V^*(t,\mathbf{x}) - V_t^*(t,\mathbf{x}) = \min_{\mathbf{u}(t)} \left\{ L(\mathbf{x},\mathbf{u}(t)) + V_{\mathbf{x}}^{*T} \mathbf{f}_t(\mathbf{x},\mathbf{u}(t)) + \frac{1}{2}\mathrm{Tr}[V_{\mathbf{xx}}^* \mathbf{B}(t)\mathbf{W}(t)\mathbf{B}^T(t)] \right\}$$

Substitute (assumptions/definitions):

$$\beta \leftarrow 0 \qquad L(\mathbf{x},\mathbf{u}(t)) \leftarrow q(t,\mathbf{x}) + \frac{1}{2}\mathbf{u}^T\mathbf{R}\mathbf{u}$$

$$\mathbf{f}_t(\mathbf{x},\mathbf{u}(t)) \leftarrow \mathbf{f}(t,\mathbf{x}) + \mathbf{g}(t,\mathbf{x})\mathbf{u} \qquad \mathbf{B}(t) \leftarrow \mathbf{g}(t,\mathbf{x})$$

$$\mathbf{W}(t) \leftarrow \mathbf{\Sigma}$$

$$-\partial_t V^*(t,\mathbf{x}) = \min_{\mathbf{u}} \left\{ q(t,\mathbf{x}) + \frac{1}{2}\mathbf{u}^T\mathbf{R}\mathbf{u} + \nabla_x^T V^*(t,\mathbf{x})(\mathbf{f}(t,\mathbf{x}) + \mathbf{g}(t,\mathbf{x})\mathbf{u}) + \frac{1}{2}\mathrm{Tr}[\nabla_{xx} V^*(t,\mathbf{x})\mathbf{g}(t,\mathbf{x})\mathbf{\Sigma}\mathbf{g}^T(t,\mathbf{x})] \right\}$$

HJB Equation

of control affine opt. ctrl problem

$$-\partial_t V^*(t,\mathbf{x}) = \min_{\mathbf{u}} \left\{ q(t,\mathbf{x}) + \frac{1}{2}\mathbf{u}^T\mathbf{R}\mathbf{u} + \nabla_x^T V^*(t,\mathbf{x})(\mathbf{f}(t,\mathbf{x}) + \mathbf{g}(t,\mathbf{x})\mathbf{u}) + \frac{1}{2}\mathrm{Tr}[\nabla_{xx} V^*(t,\mathbf{x})\mathbf{g}(t,\mathbf{x})\boldsymbol{\Sigma}\mathbf{g}^T(t,\mathbf{x})] \right\}$$

gradient of RHS = 0 yields

$$\mathbf{u}^*(t,\mathbf{x}) = -\mathbf{R}^{-1}\mathbf{g}^T(t,\mathbf{x})\nabla_x V(t,\mathbf{x})$$

Optimal control

'improve value'
negative gradient

$$\mathbf{u}^*(t,\mathbf{x}) = -\mathbf{R}^{-1}\mathbf{g}^T(t,\mathbf{x})\nabla_x V(t,\mathbf{x})$$

'coord. trafo.'
'states to control'

'Cost of controls to get improvement'

Optimal HJB

substitute opt. control back into HJB ⇒

$$\mathbf{u}^*(t,\mathbf{x}) = -\mathbf{R}^{-1}\mathbf{g}^T(t,\mathbf{x})\nabla_x V(t,\mathbf{x})$$

$$-\partial_t V^* = q - \frac{1}{2}\nabla_x^T V^* \, \mathbf{g}\mathbf{R}^{-1}\mathbf{g}^T \nabla_x V^* + \nabla_x^T V^* \, \mathbf{f} + \frac{1}{2}\mathrm{Tr}[\nabla_{xx} V^* \, \mathbf{g}\boldsymbol{\Sigma}\mathbf{g}^T]$$

Nonlinear PDE!

Optimal HJB

substitute opt. control back into HJB ⇒

$\mathbf{u}^*(t,\mathbf{x}) = -\mathbf{R}^{-1}\mathbf{g}^T(t,\mathbf{x})\nabla_x V(t,\mathbf{x})$

$$-\partial_t V^* = q - \frac{1}{2}\nabla_x^T V^* \, \mathbf{g}\mathbf{R}^{-1}\mathbf{g}^T \nabla_x V^* + \nabla_x^T V^* \, \mathbf{f} + \frac{1}{2}\mathrm{Tr}[\nabla_{xx} V^* \, \mathbf{g}\boldsymbol{\Sigma}\mathbf{g}^T]$$

Nonlinear PDE!

short notation, replace: $\mathbf{g}\,\mathbf{R}^{-1}\mathbf{g}^T$ by $\boldsymbol{\Xi}$

add'l assumption: control cost linked to noise

$$\mathbf{R}\boldsymbol{\Sigma} = \lambda\mathbf{I} \quad \Rightarrow \quad \mathbf{g}\boldsymbol{\Sigma}\mathbf{g}^T = \lambda\boldsymbol{\Xi}$$

$\lambda > 0$

log transform

$$-\partial_t V^* = q - \frac{1}{2}\nabla_x^T V^* \,\Xi \nabla_x V^* + \nabla_x^T V^* \,\mathbf{f} + \frac{\lambda}{2}\mathrm{Tr}[\nabla_{xx} V^* \,\Xi]$$

Nonlinear PDE!

Desirability Ψ

Log transform

$$V^*(t, \mathbf{x}) = -\lambda \log \Psi(t, \mathbf{x})$$

$\Rightarrow$

$$\begin{aligned} \partial_t V^*(t,\mathbf{x}) &= -\lambda \frac{\partial_t \Psi}{\Psi} \\ \nabla_x V^*(t,\mathbf{x}) &= -\lambda \frac{\nabla_x \Psi}{\Psi} \\ \nabla_{xx} V^*(t,\mathbf{x}) &= \frac{1}{\lambda}\nabla_x V^* \,\nabla_x^T V^* - \lambda \frac{\nabla_{xx}\Psi}{\Psi} \end{aligned}$$

rewrite scalar expression:

$$-\frac{1}{2}\nabla_x^T V^* \,\Xi \nabla_x V^* = -\frac{1}{2}\mathrm{Tr}[\nabla_x^T V^* \,\Xi \nabla_x V^*] = -\frac{1}{2}\mathrm{Tr}[\nabla_x V^* \,\nabla_x^T V^* \,\Xi]$$

substitute into HJB:

$$\lambda \frac{\partial_t \Psi}{\Psi} = q - \cancel{\frac{1}{2}\mathrm{Tr}[\nabla_x V^* \,\nabla_x^T V^* \,\Xi]} - \lambda \mathbf{f}^T \frac{\nabla_x \Psi}{\Psi} + \cancel{\frac{1}{2}\mathrm{Tr}[\nabla_x V^* \,\nabla_x^T V^* \,\Xi]} - \frac{\lambda^2}{2}\mathrm{Tr}\left[\frac{\nabla_{xx}\Psi}{\Psi}\Xi\right]$$

multiply both sides by $-\Psi/\lambda$

Linear HJB for desirability

$$-\partial_t \Psi = -\frac{1}{\lambda} q \Psi + \mathbf{f}^T \nabla_x \Psi + \frac{\lambda}{2} \mathrm{Tr}[\Xi \nabla_{xx} \Psi]$$

Linear PDE in desirability Ψ

equivalent form:

$$-\partial_t \Psi = \mathrm{H}[\Psi]$$

$$\begin{aligned} \mathrm{H} &= -\frac{1}{\lambda} q + \mathbf{f}^T \nabla_x + \frac{\lambda}{2} \mathrm{Tr}[\Xi \nabla_{xx}] \\ &= -\frac{1}{\lambda} q + \sum_i \mathbf{f}_i \frac{\partial}{\partial_{x_i}} + \frac{\lambda}{2} \sum_{i,j} \Xi_{ij} \frac{\partial^2}{\partial_{x_i} \partial_{x_j}} \end{aligned}$$

linear, but still no analytic solution for arbitrary q(x,t)

ADRL

could solve backward terminal condition $\Psi(t_f, \mathbf{x}) = \exp\left(-\frac{1}{\lambda} \Phi(\mathbf{x})\right)$

Forward solution through diffusion process

$$-\partial_t \Psi = -\frac{1}{\lambda} q \Psi + \mathbf{f}^T \nabla_x \Psi + \frac{\lambda}{2} \mathrm{Tr}[\Xi \nabla_{xx} \Psi]$$

Can solve this equation using 'forward diffusion process'

$$\Psi_{t_i} = E_{\tau_i}\left(\Psi_{t_N} e^{-\int_{t_i}^{t_N} \frac{1}{\lambda} q_t dt}\right) = E_{\tau_i}\left[\exp\left(-\frac{1}{\lambda}\phi_{t_N} - \frac{1}{\lambda}\int_{t_i}^{t_N} q_t \, dt\right)\right]$$

'expectation in respect to path τ'

path drawn from random process

$$\tau = x(t \ldots t_N) \sim p(x, u)$$

$$d\mathbf{x} = \mathbf{f}(t, \mathbf{x})dt + \mathbf{g}(t, \mathbf{x})d\mathbf{w} \qquad d\omega \sim N(0, \Sigma)$$

forward! ... but stochastic

Remember the forward search in the discrete state, discrete time problem (Lect. 1)

A D R L

ETH Zürich

Credits

material from:

Sutton & Barto's book: http://webdocs.cs.ualberta.ca/~sutton/book/the-book.html

Bishop: Pattern Recognition and Machine Learning

Feynman Lectures