University of BRISTOL

Machine Learning

Reinforcement Learning and Decisions

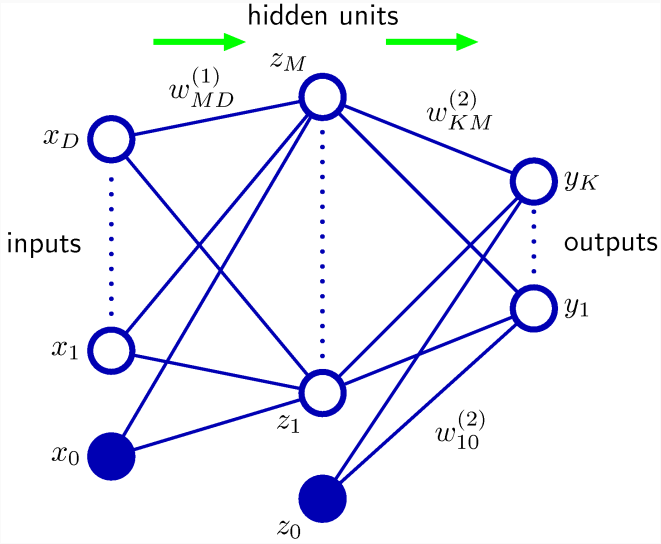
Carl Henrik Ek - carlhenrik.ek@bristol.ac.uk

November 12, 2019

http://www.carlhenrik.com

Introduction

Neural Network



$$y_k(\mathbf{x}, \mathbf{w}) = \sigma\left(\sum_j^M w_{kj}^{(2)} h\left(\sum_j^D w_{ji}^{(1)} x_i + w_{j0}^{(1)}\right) + w_{k0}^{(2)}\right)$$

Composite Functions

Why are composite functions attractive?

$$\mathbf{y} = g(\mathbf{x}) = f_K(f_{K-1}(f_{K-2}(\dots f_1(\mathbf{x})\dots)))$$

Linear Algebra

- Kernel of a function

$$\text{Kern}(f_k) = \left\{(\mathbf{x}, \mathbf{x}') | f_k(\mathbf{x}) = f_k(\mathbf{x}')\right\}$$

Linear Algebra

- Kernel of a function

$$\text{Kern}(f_k) = \left\{(\mathbf{x}, \mathbf{x}') | f_k(\mathbf{x}) = f_k(\mathbf{x}')\right\}$$

- Image of a function

$$\text{Im}(f_k(\mathbf{x})) = \{\mathbf{y} \in Y | \mathbf{y} = f_k(\mathbf{x}), \mathbf{x} \in X\}$$

Linear Algebra

- Rank-Nullity Theorem

$$\dim(\mathrm{Im}(f)) = \dim(V) - \dim(\mathrm{Kern}(f))$$

Linear Algebra

- Rank-Nullity Theorem

$$\dim(\mathrm{Im}(f)) = \dim(V) - \dim(\mathrm{Kern}(f))$$

- Kernel of function

$$\mathrm{Kern}(f_1) \subseteq \mathrm{Kern}(f_{k-1} \circ \ldots \circ f_2 \circ f_1) \subseteq \mathrm{Kern}(f_k \circ f_{k-1} \circ \ldots \circ f_2 \circ f_1)$$

Linear Algebra

- Rank-Nullity Theorem

$$\dim(\mathrm{Im}(f)) = \dim(V) - \dim(\mathrm{Kern}(f))$$

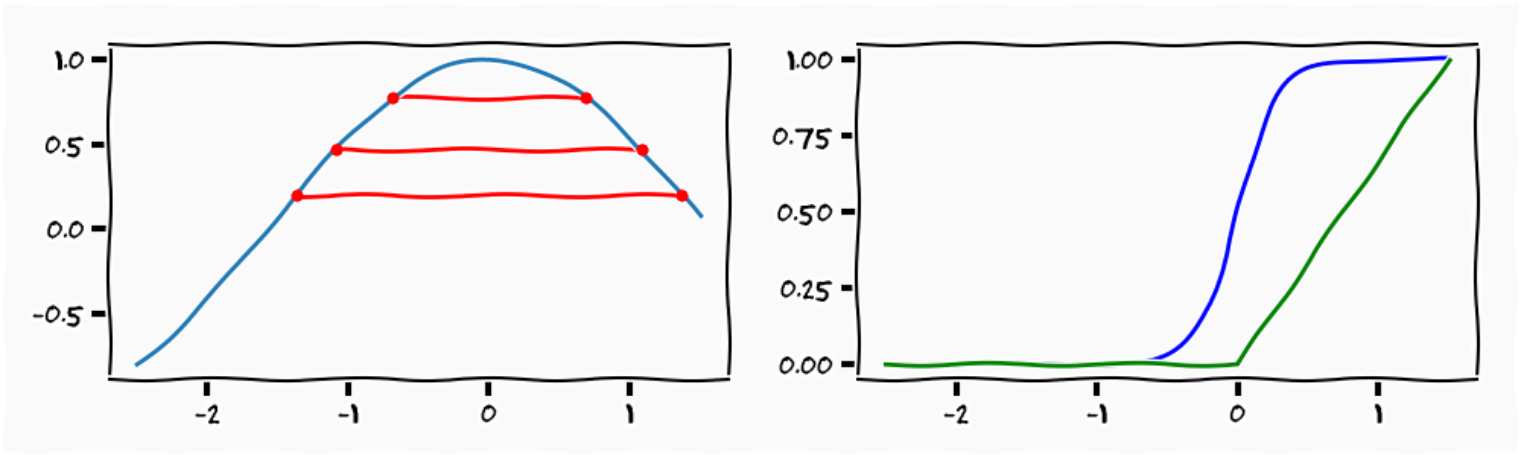
- Kernel of function

$$\mathrm{Kern}(f_1) \subseteq \mathrm{Kern}(f_{k-1} \circ \ldots \circ f_2 \circ f_1) \subseteq \mathrm{Kern}(f_k \circ f_{k-1} \circ \ldots \circ f_2 \circ f_1)$$

- Image of a function

$$\mathrm{Im}(f_k \circ f_{k-1} \circ \ldots \circ f_2 \circ f_1) \subseteq \mathrm{Im}(f_k \circ f_{k-1} \circ \ldots \circ f_2) \subseteq \ldots \subseteq \mathrm{Im}(f_k)$$

Composition functions

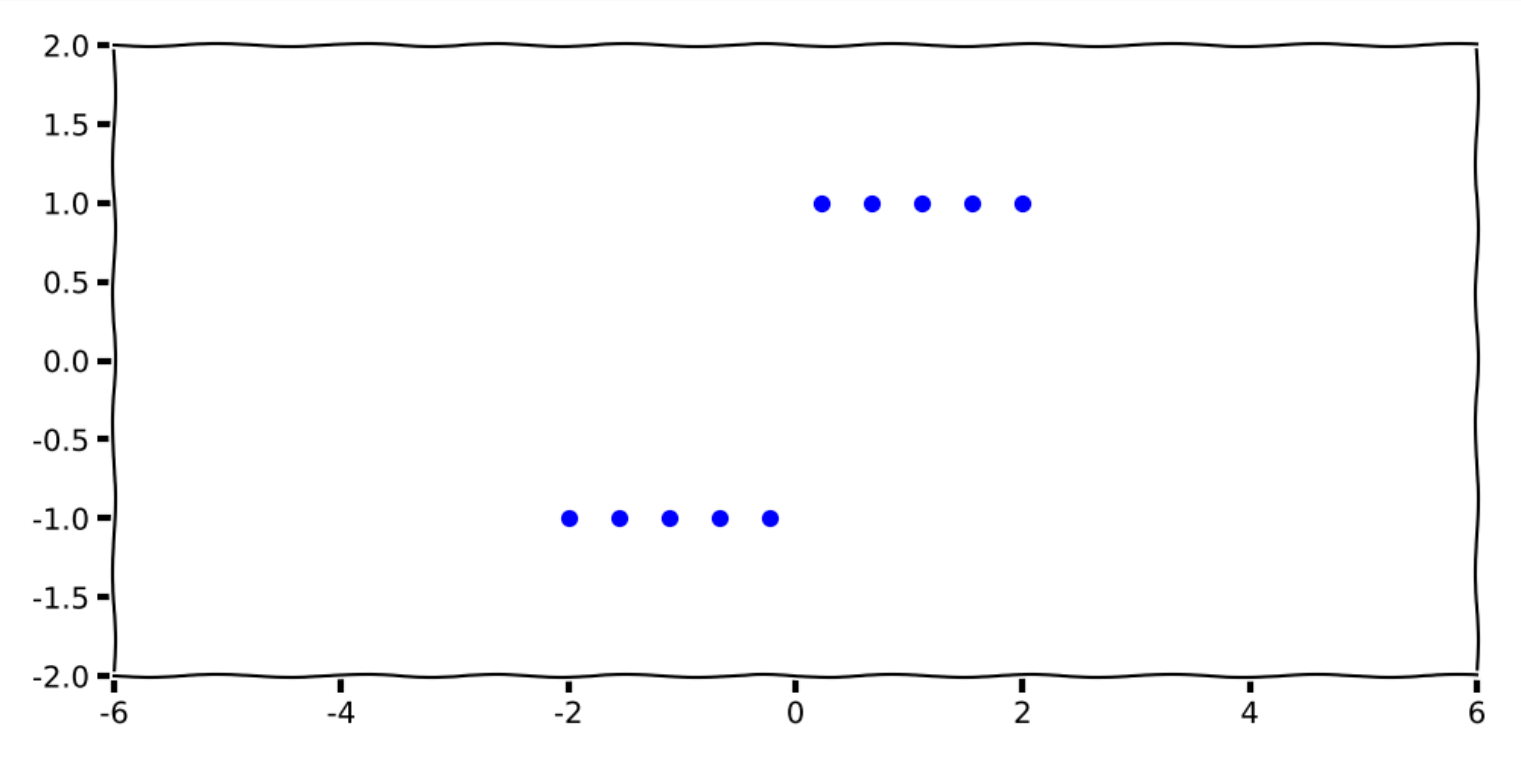


$$y = f_k(f_{k-1}(\dots f_0(x))) = f_k \circ f_{k-1} \circ \dots \circ f_1(x)$$

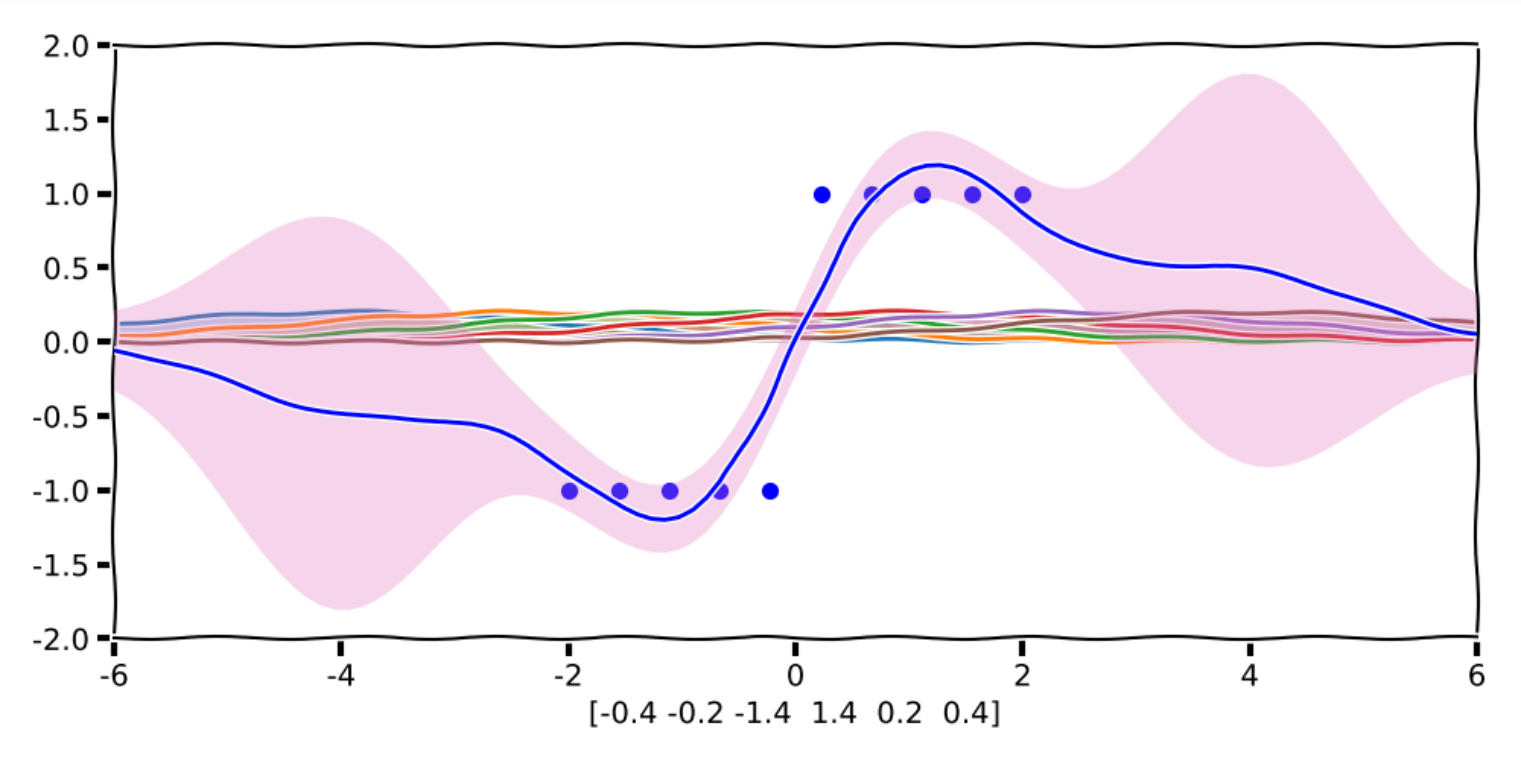
$$\text{Kern}(f_1) \subseteq \text{Kern}(f_{k-1} \circ \dots \circ f_2 \circ f_1) \subseteq \text{Kern}(f_k \circ f_{k-1} \circ \dots \circ f_2 \circ f_1)$$

$$\text{Im}(f_k \circ f_{k-1} \circ \dots \circ f_2 \circ f_1) \subseteq \text{Im}(f_k \circ f_{k-1} \circ \dots \circ f_2) \subseteq \dots \subseteq \text{Im}(f_k)$$

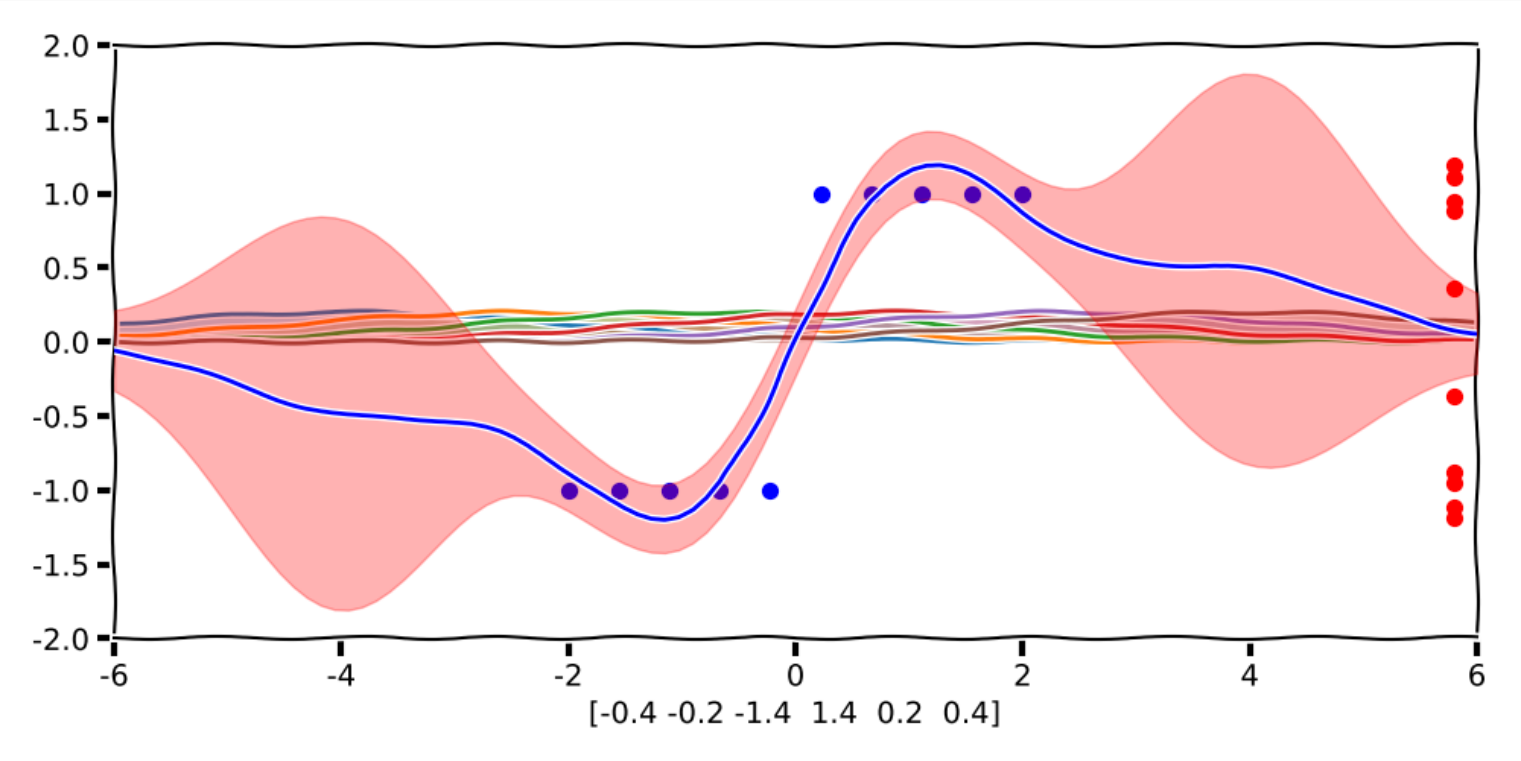
Composite Functions



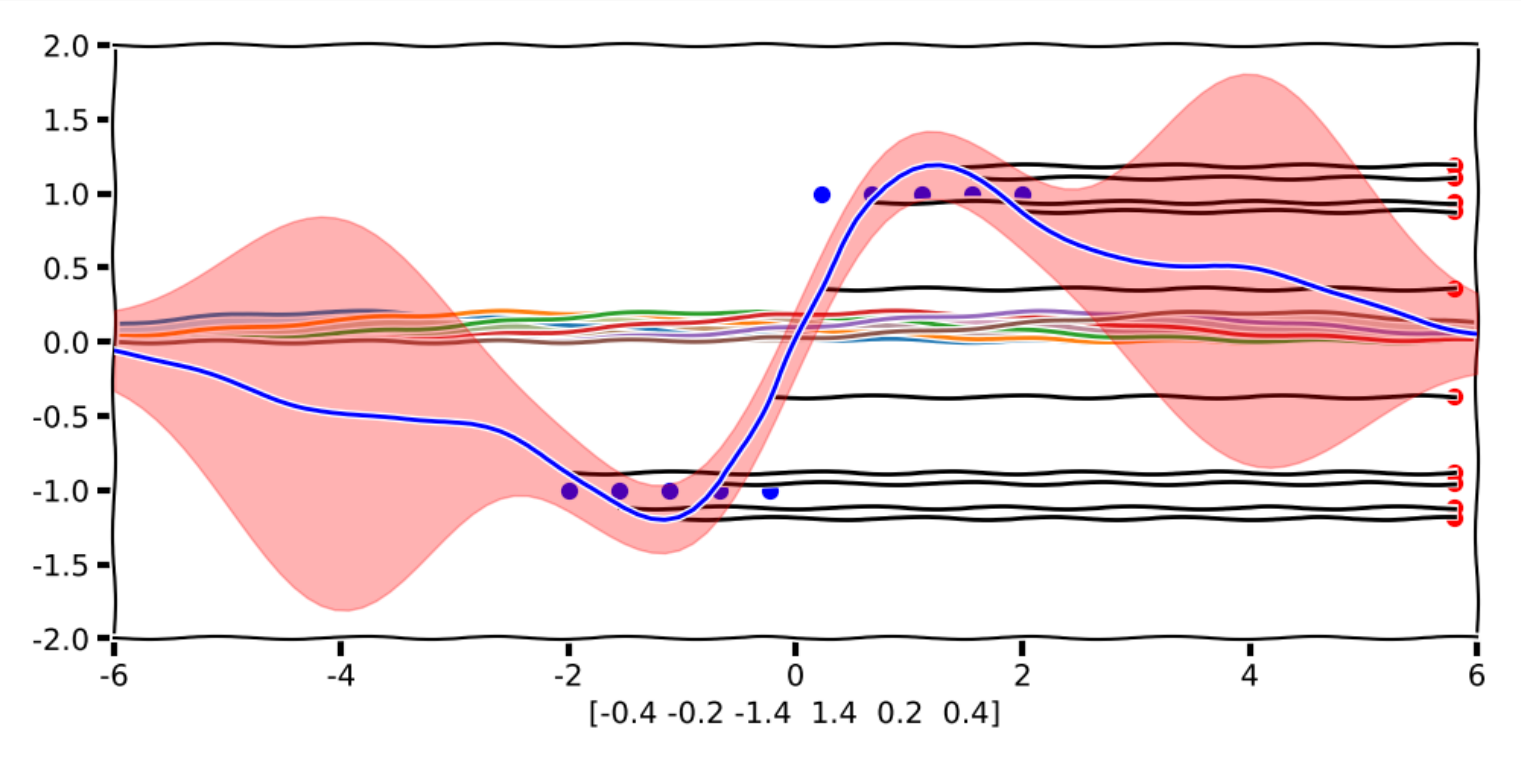
Composite Functions



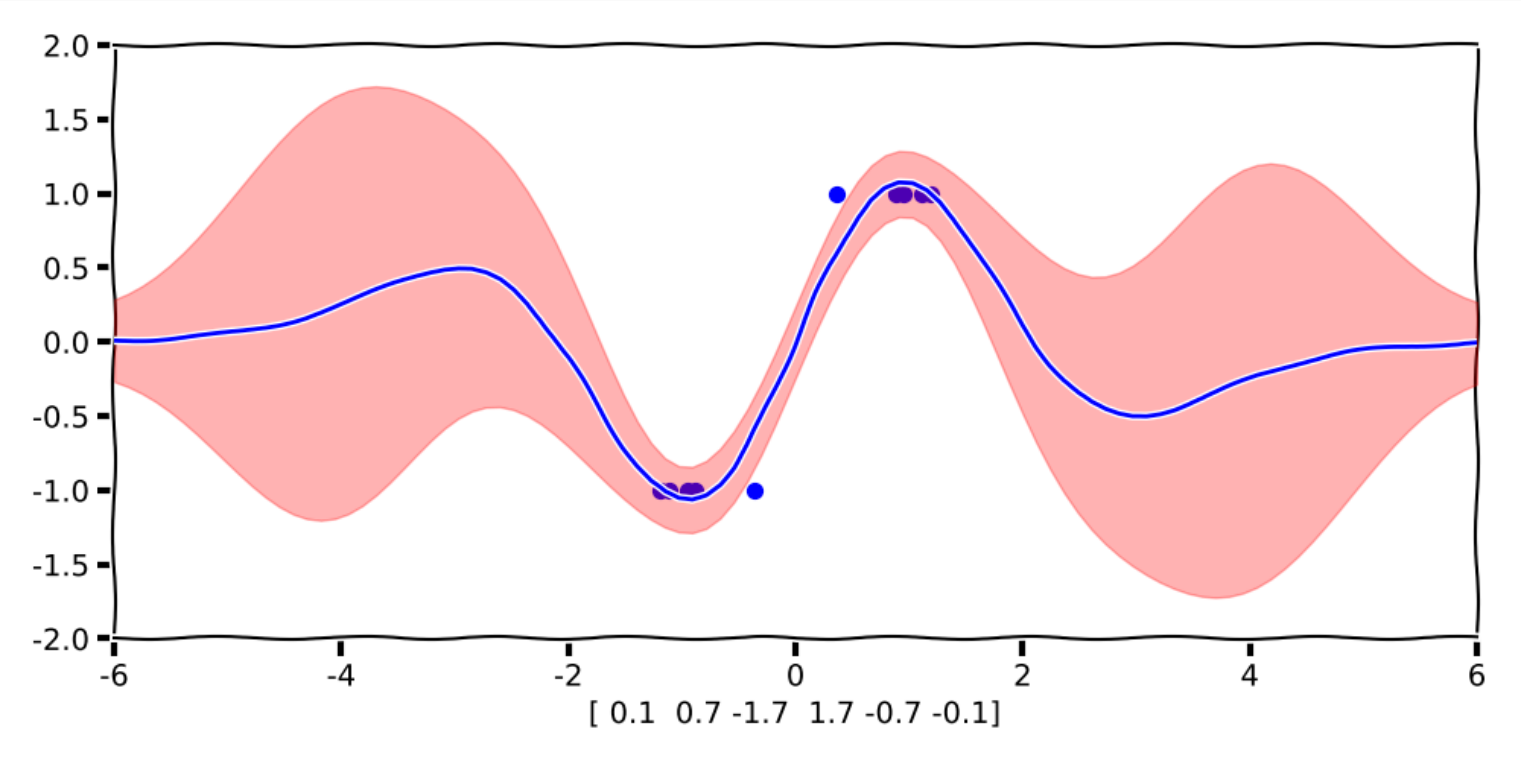
Composite Functions



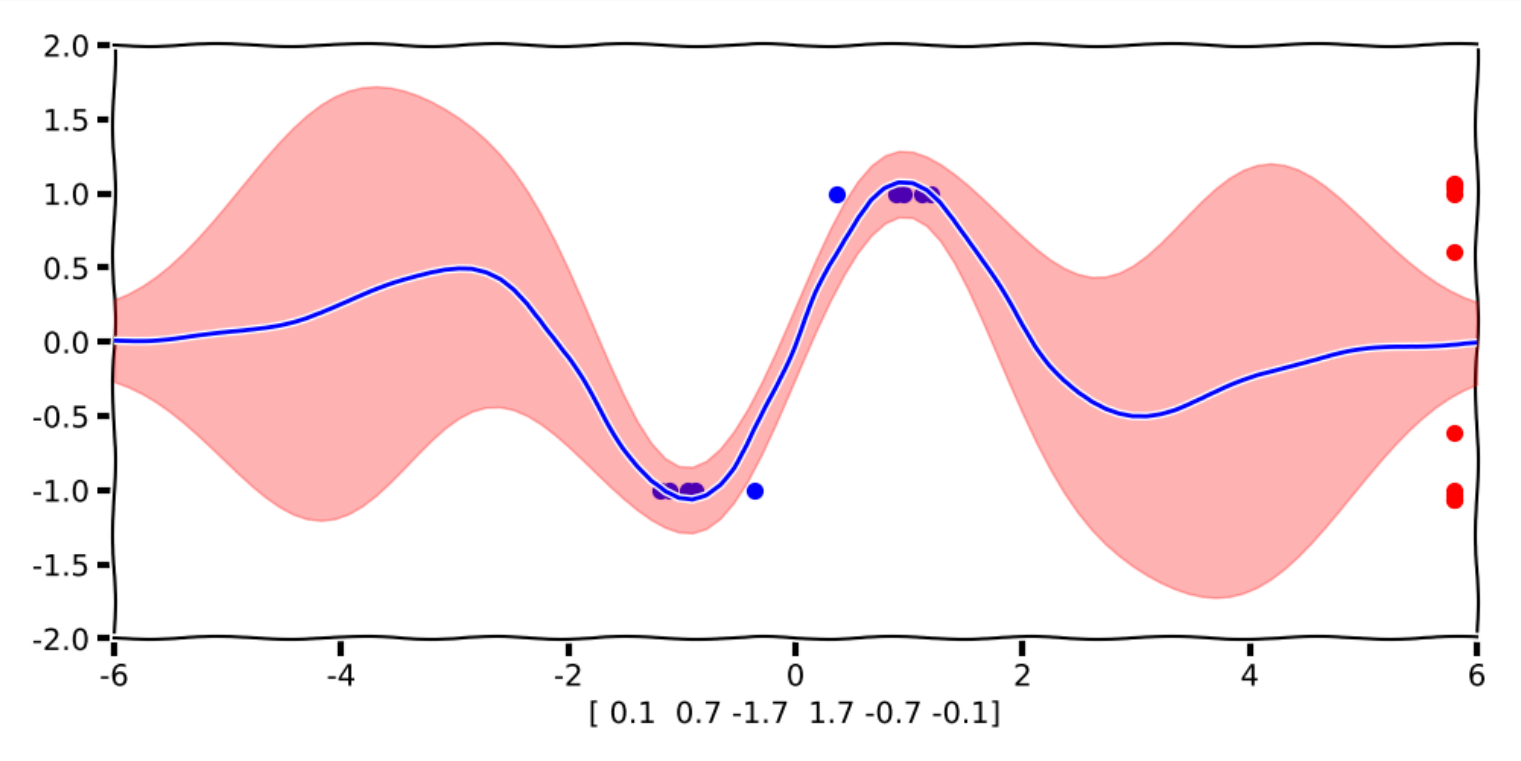
Composite Functions



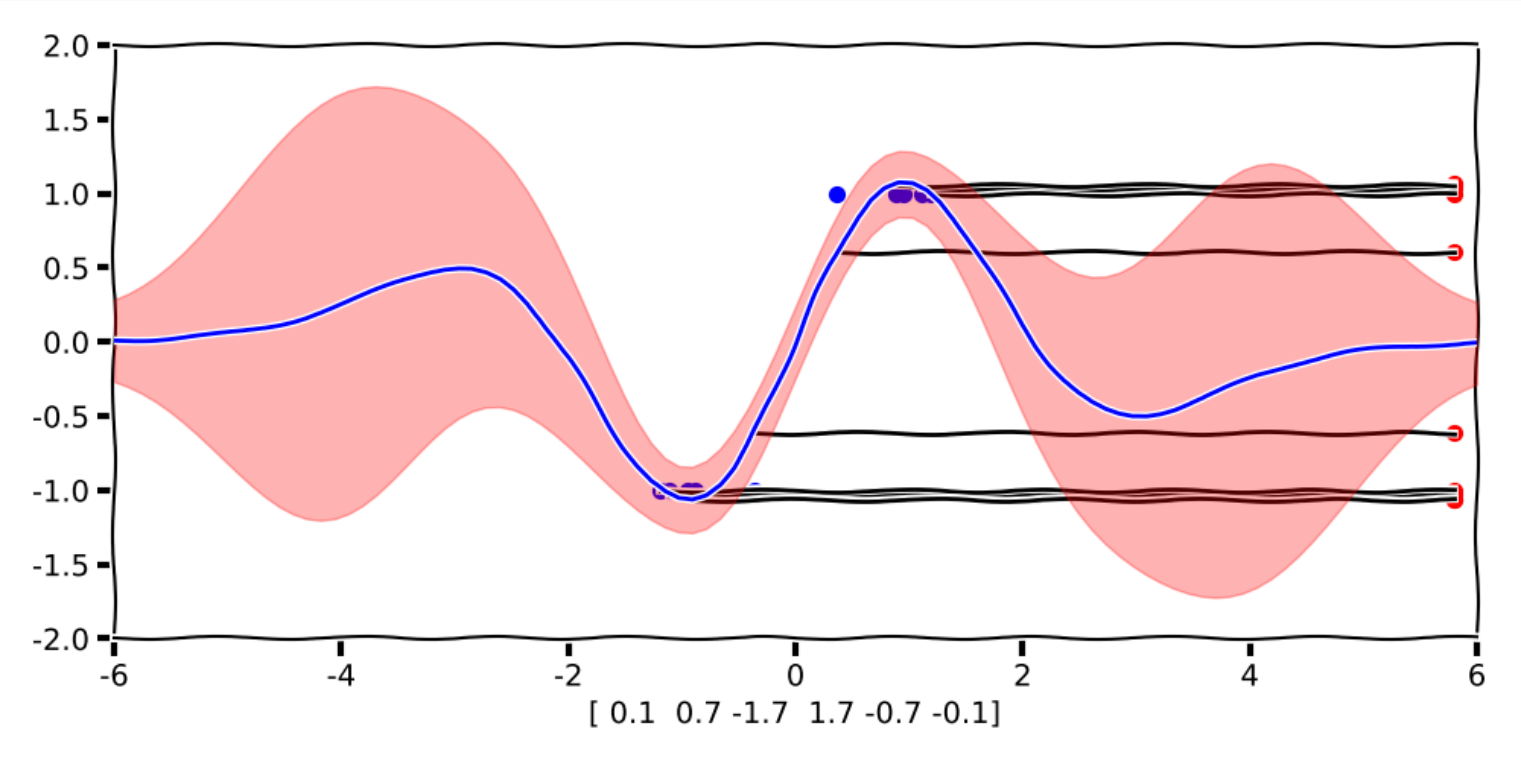
Composite Functions



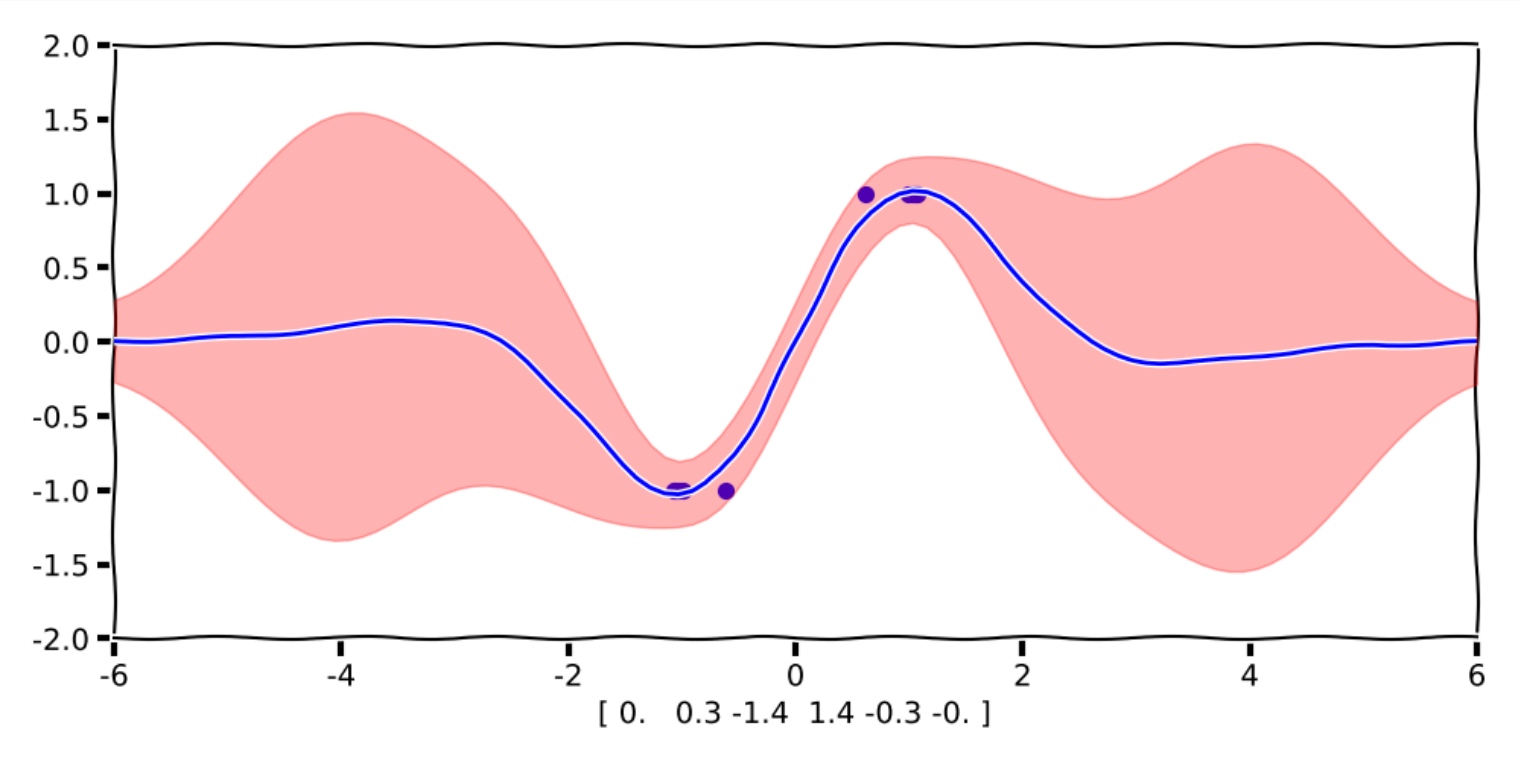
Composite Functions



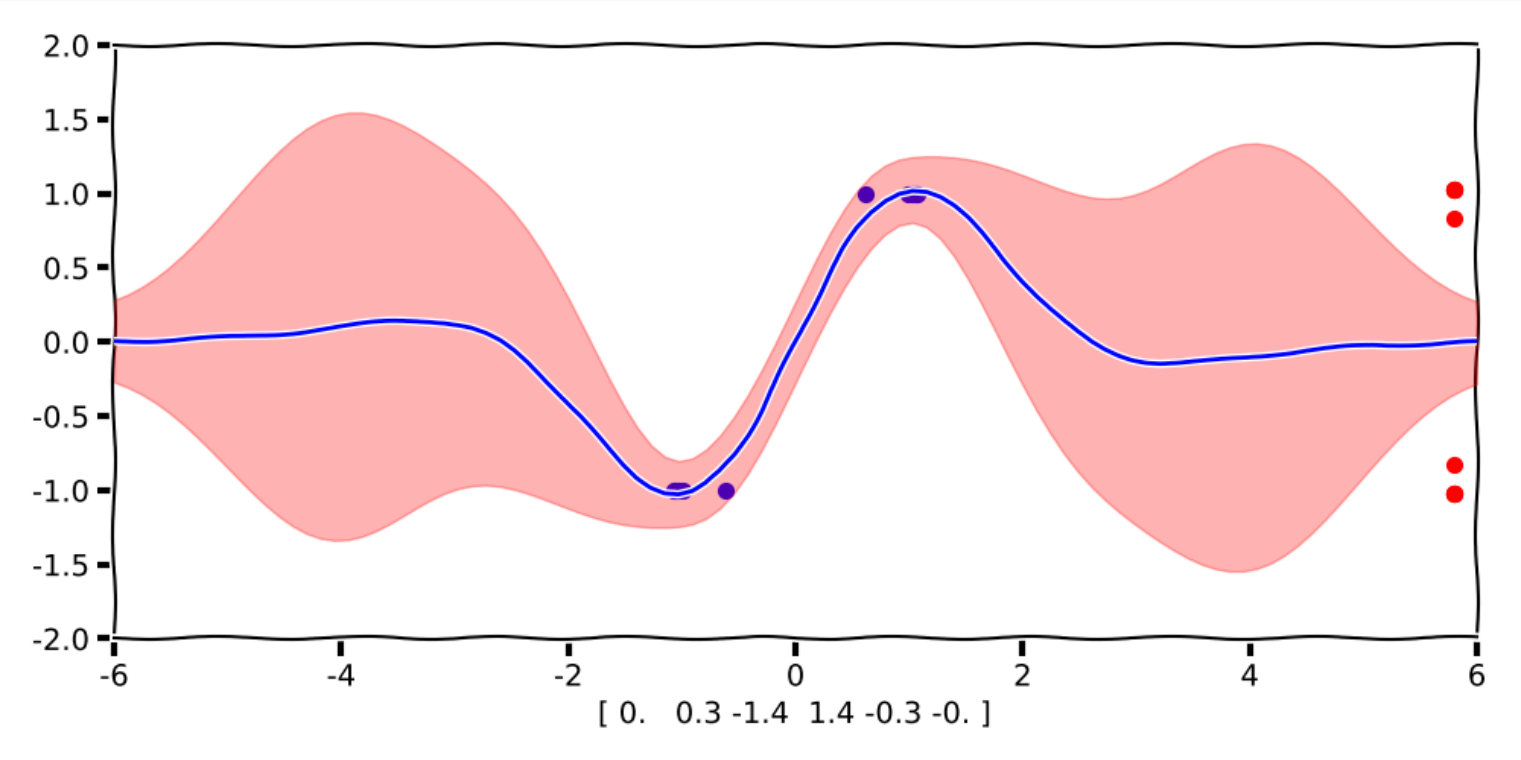
Composite Functions



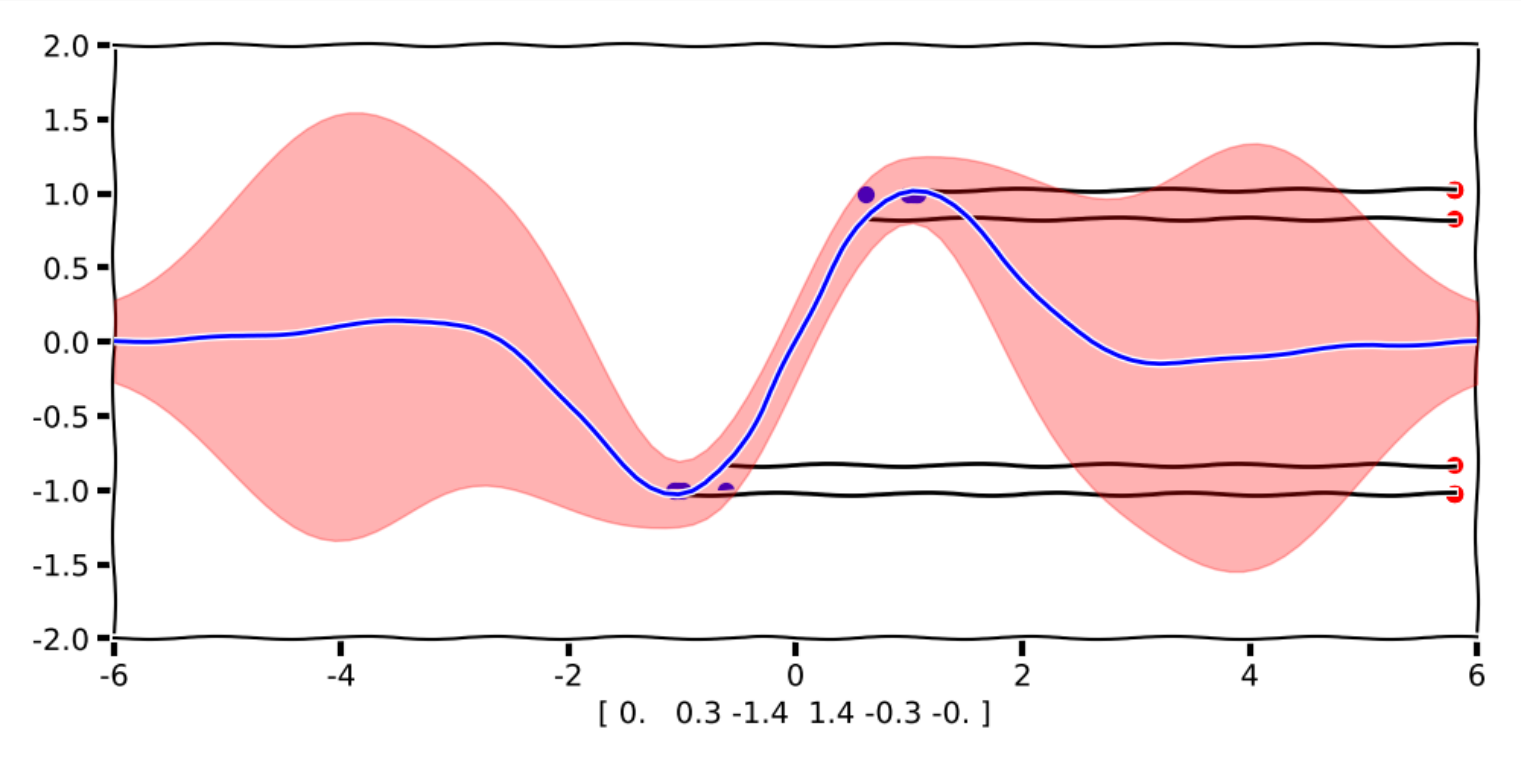
Composite Functions



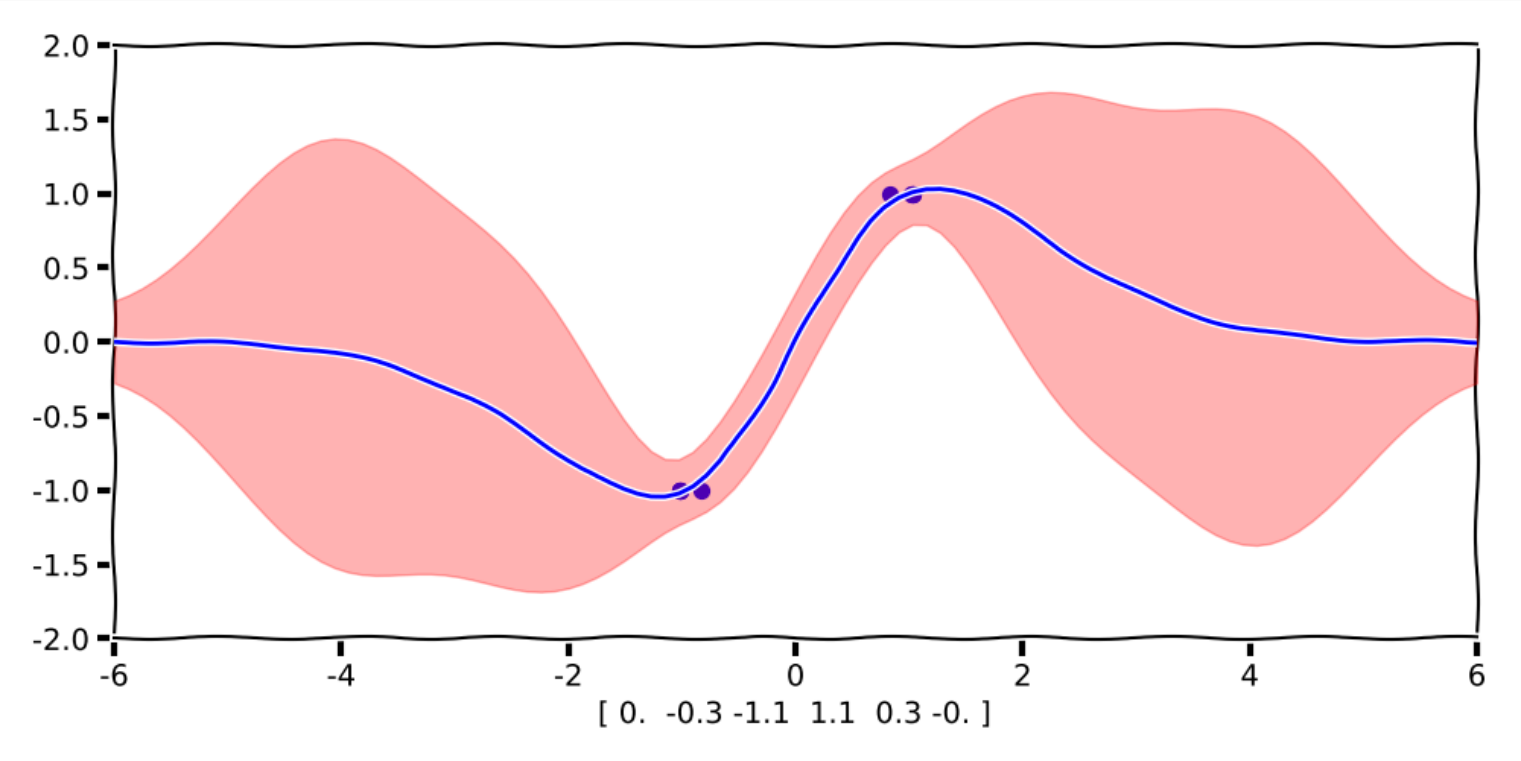
Composite Functions



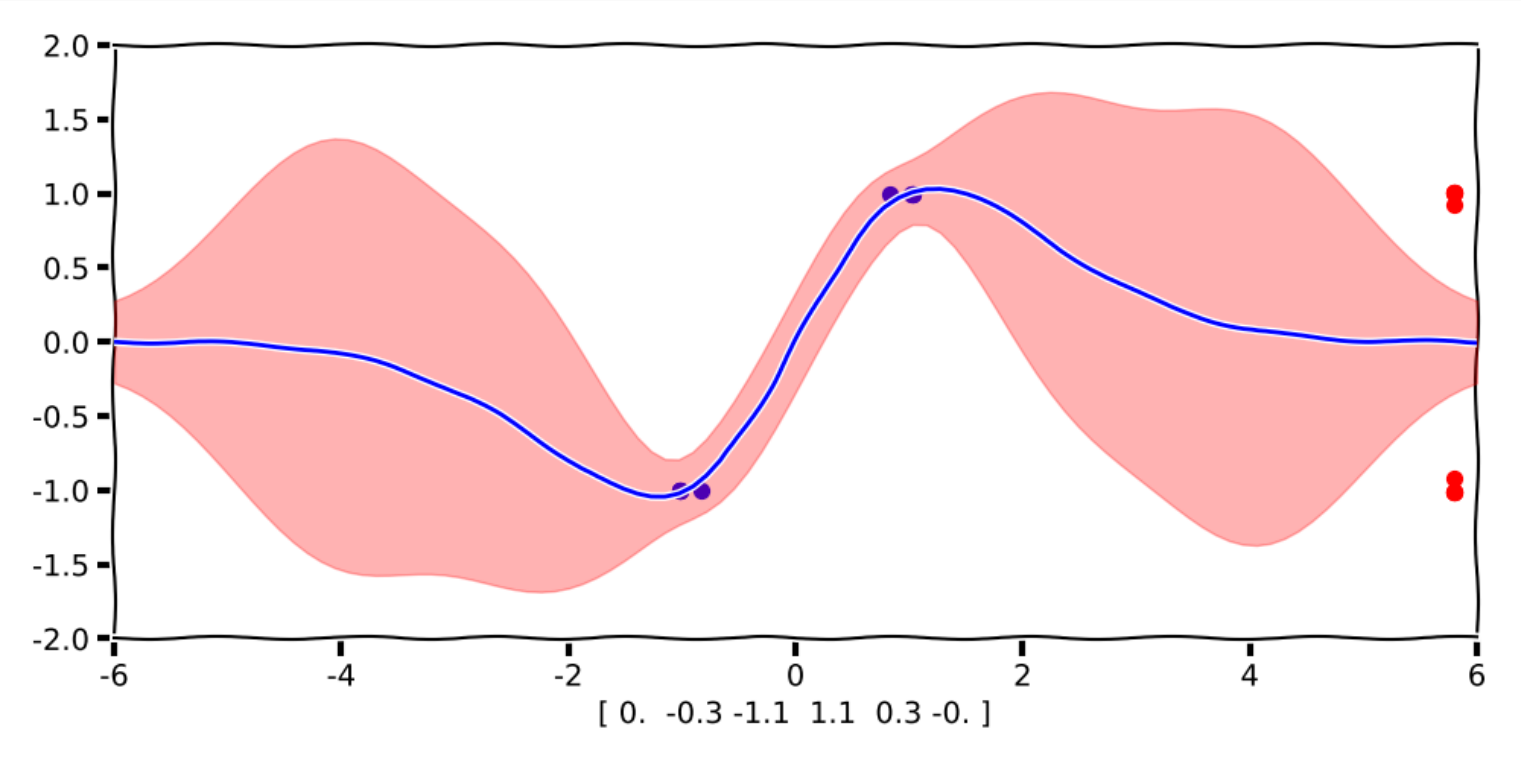
Composite Functions



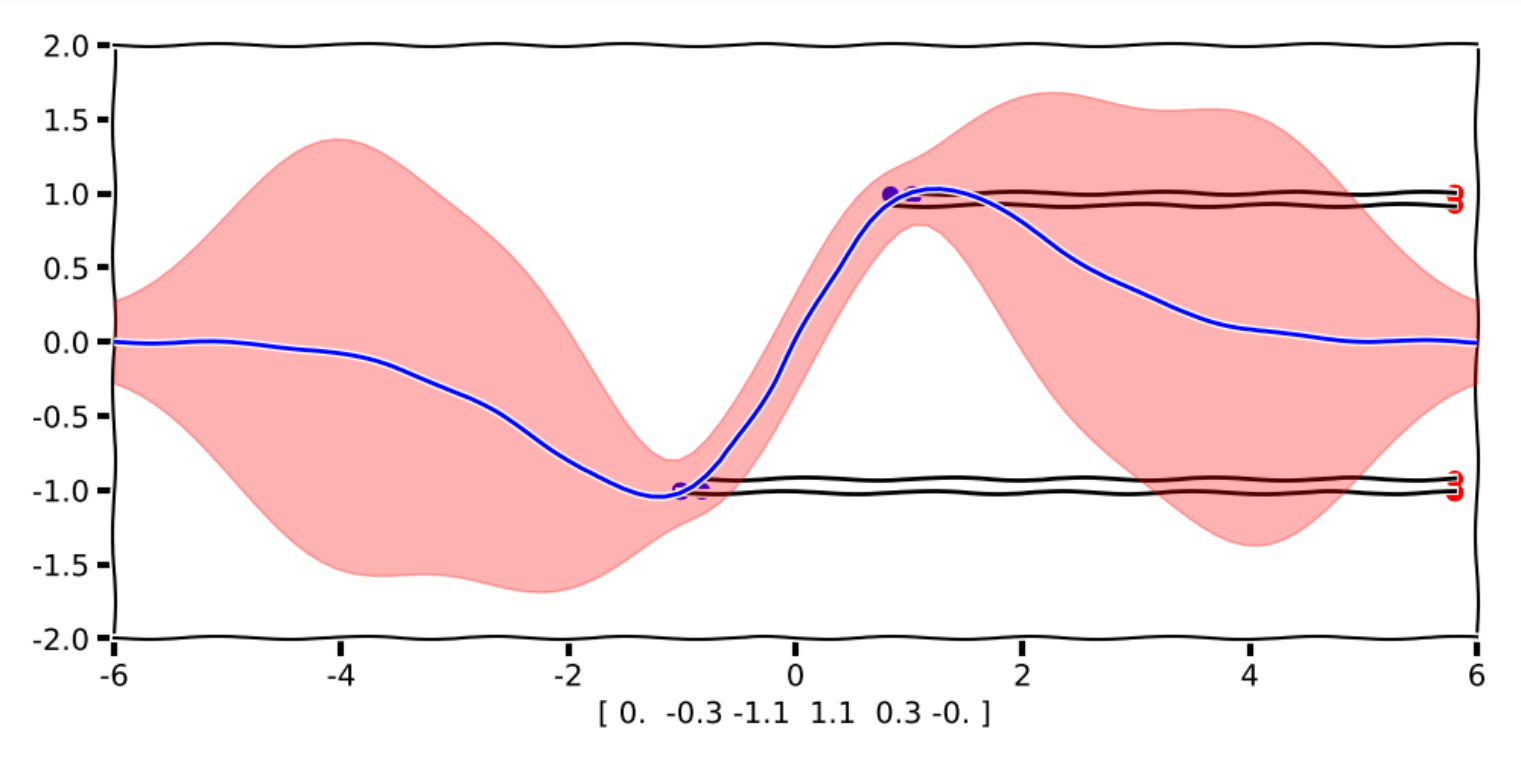
Composite Functions



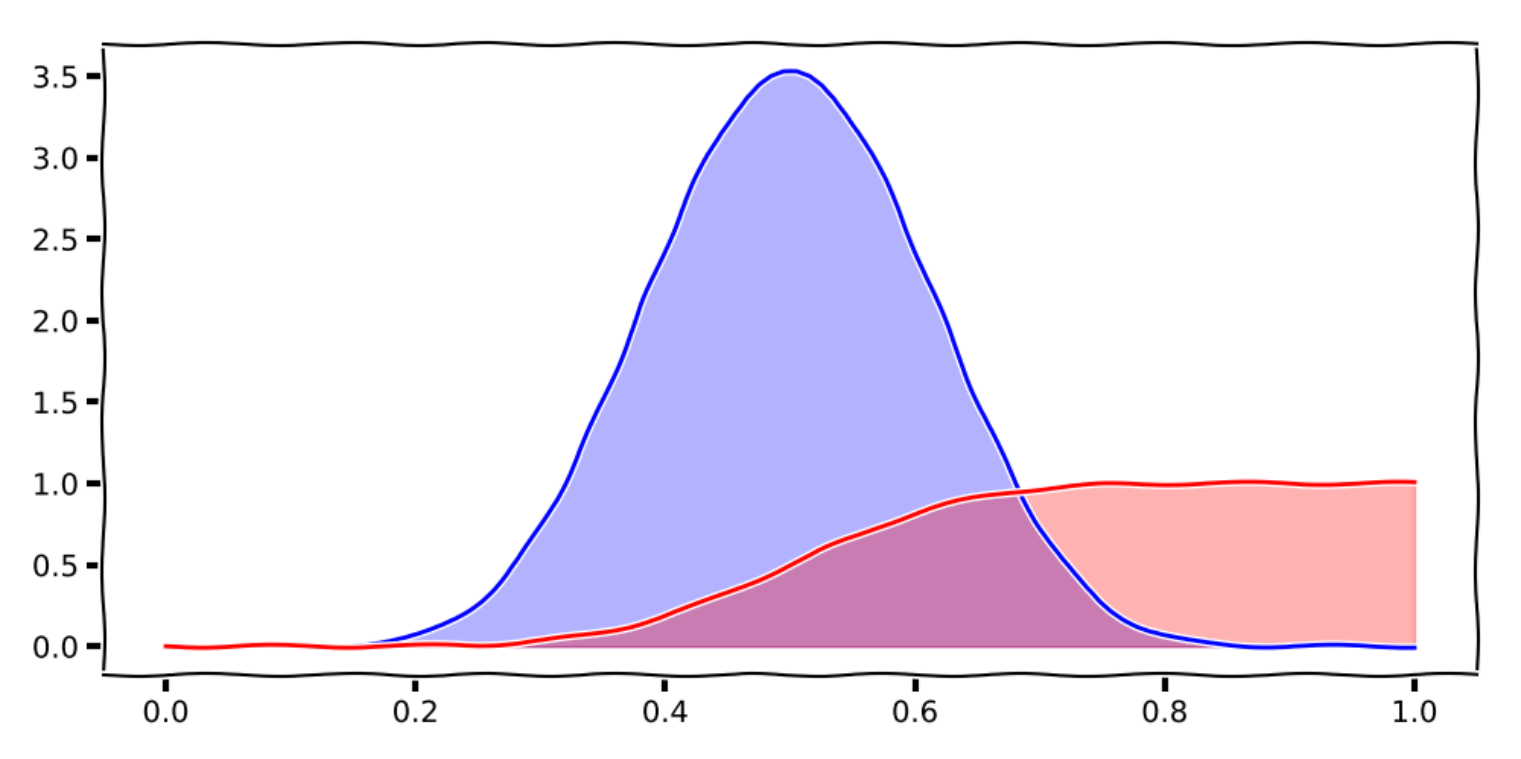
Composite Functions



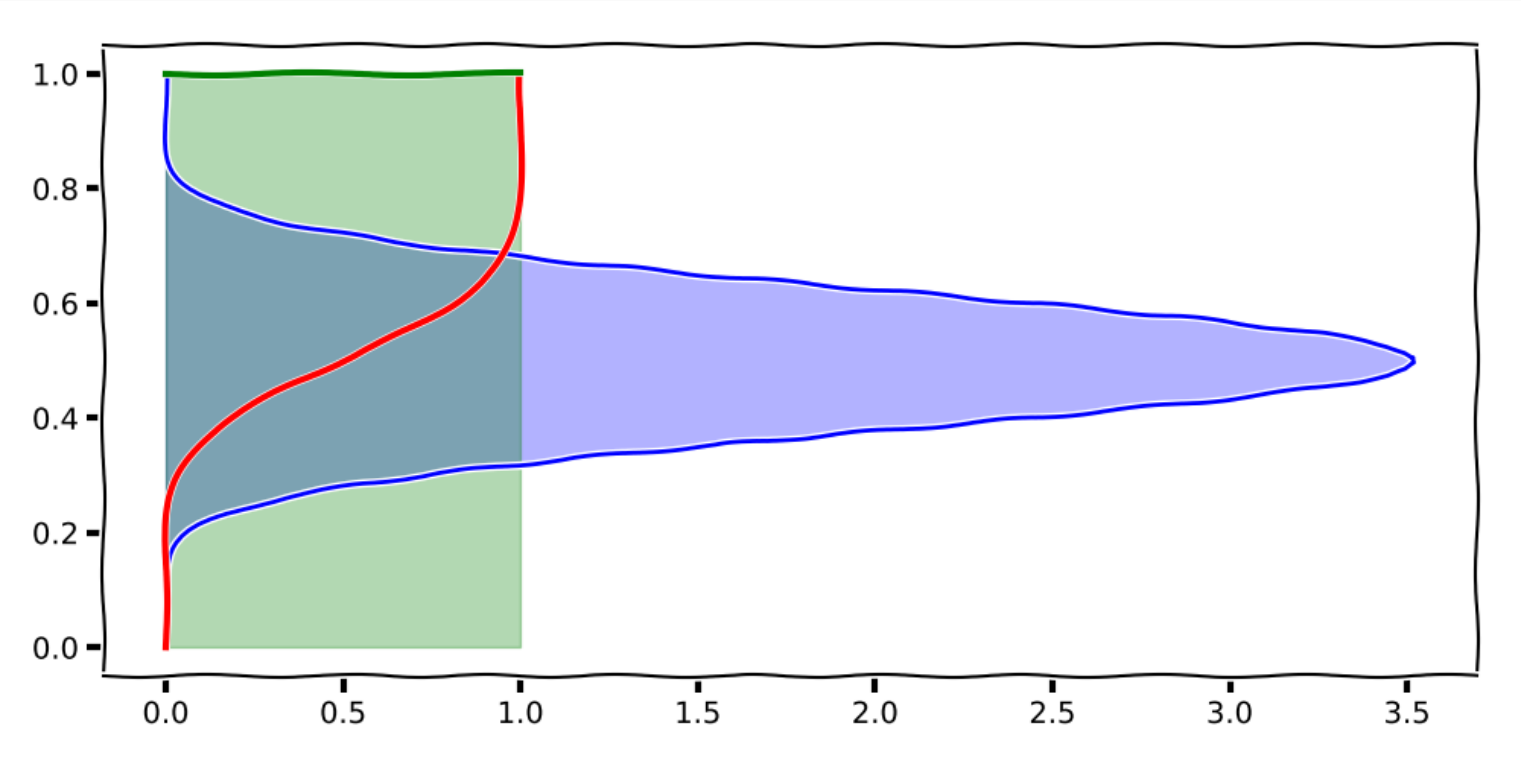
Composite Functions



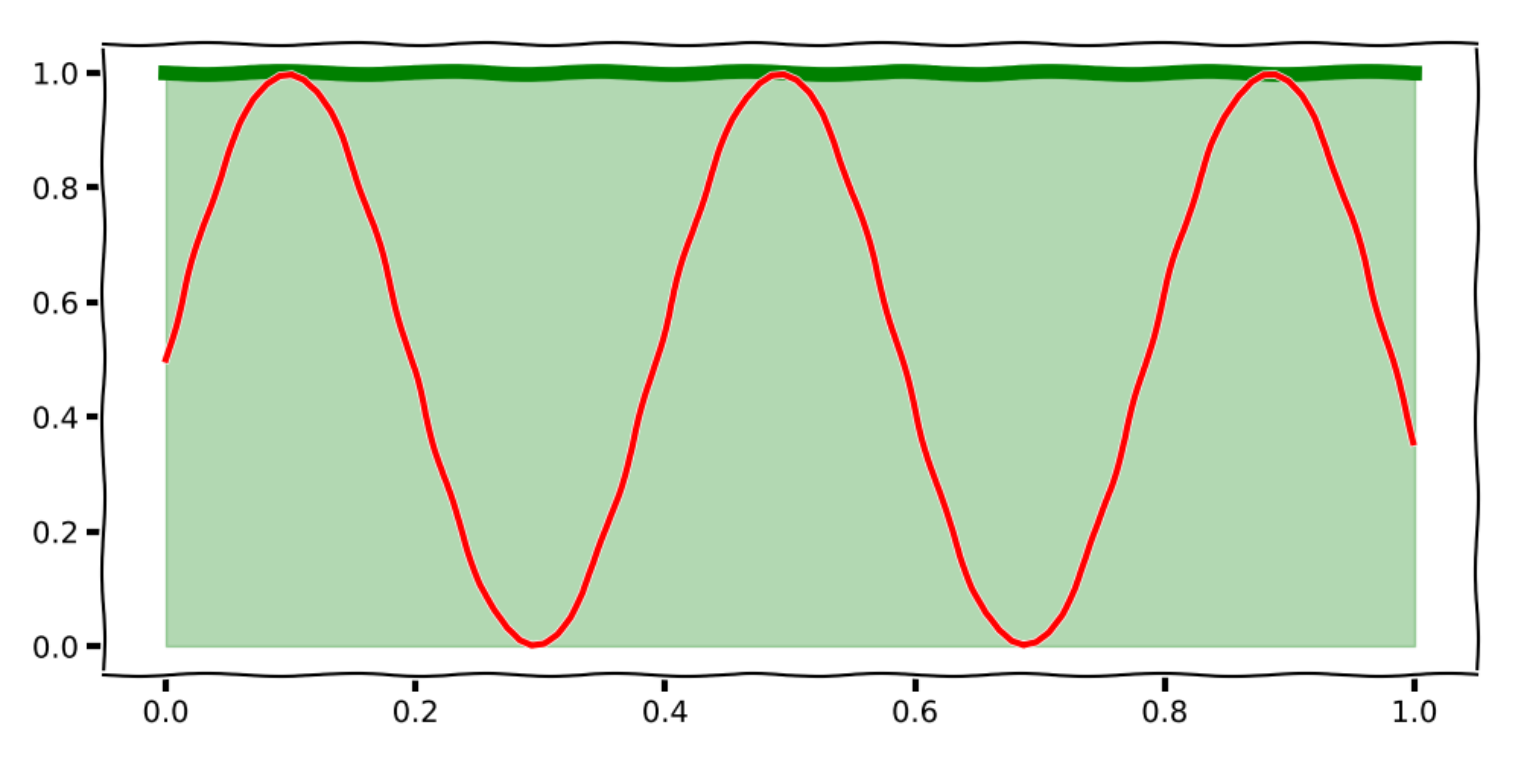
Sampling



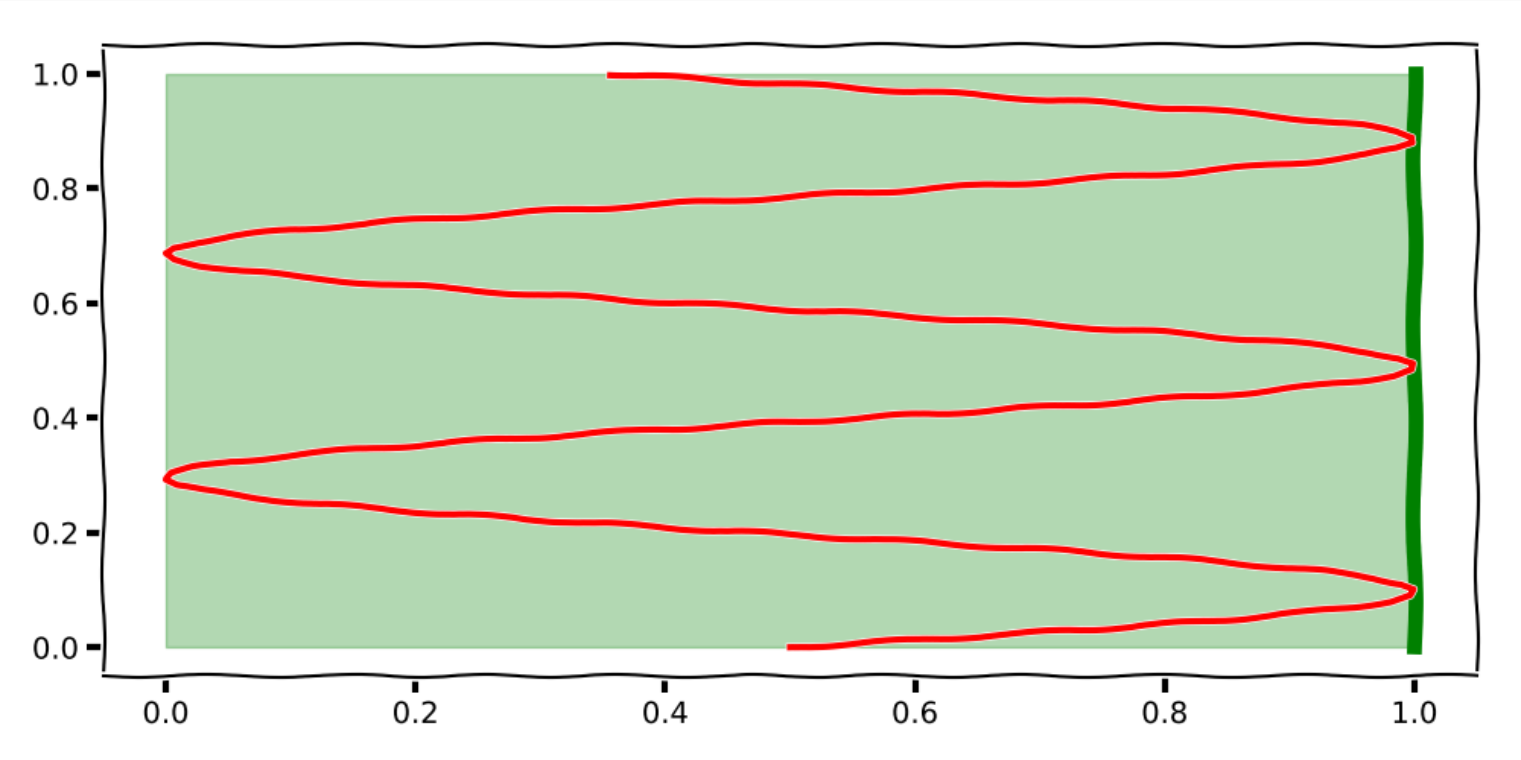
Sampling



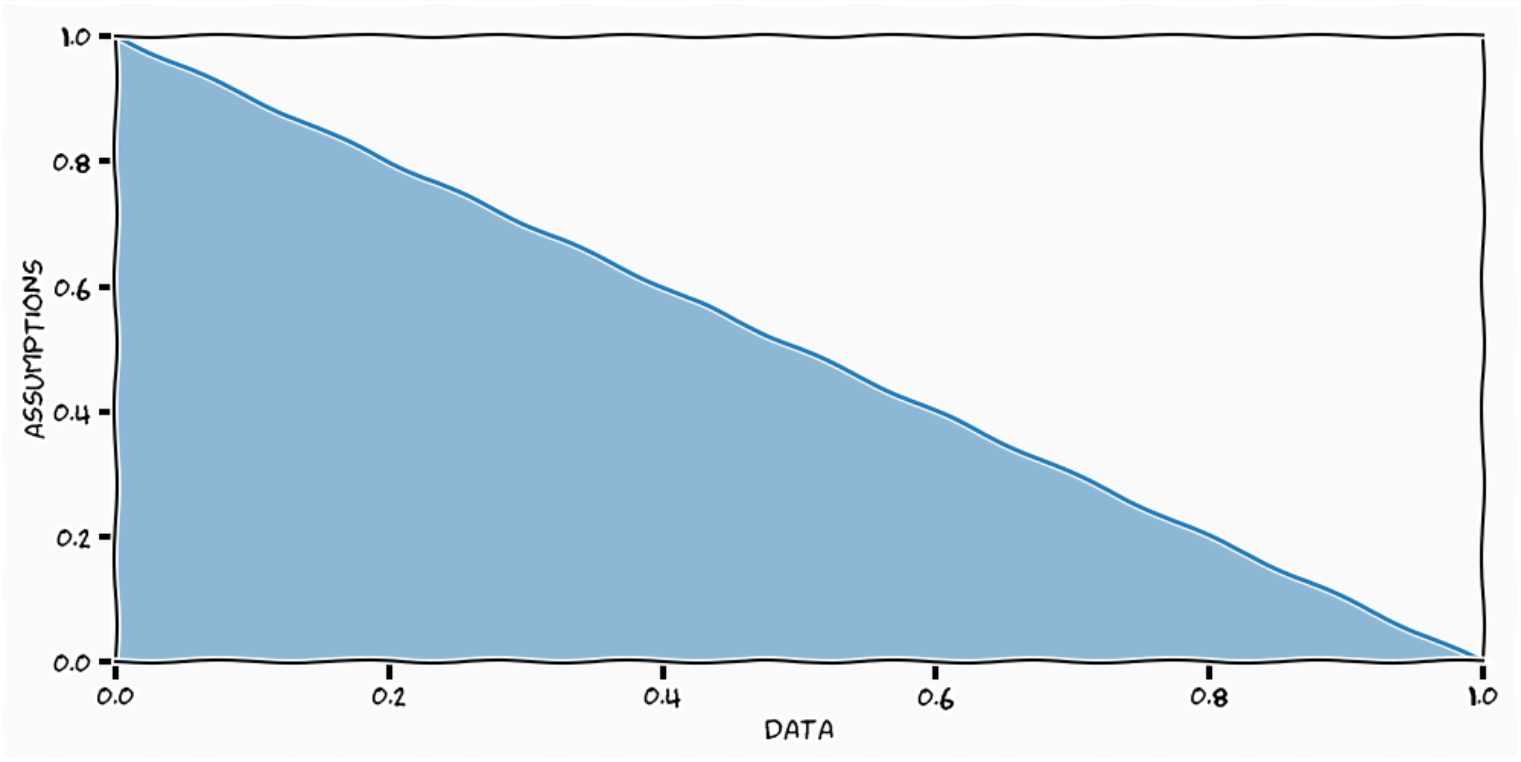
Change of Variables



Change of Variables



Challenge



- We can build a larger range of "functions" by combining simpler ones
- But we reduce our prior assumptions
- However, because its simple we can trade-off with data

Reinforcement Learning

Machine Learning

Supervised Learning predict output from input

$$\mathcal{D} = \{y_i, x_i\}_{i=1}^{N}$$

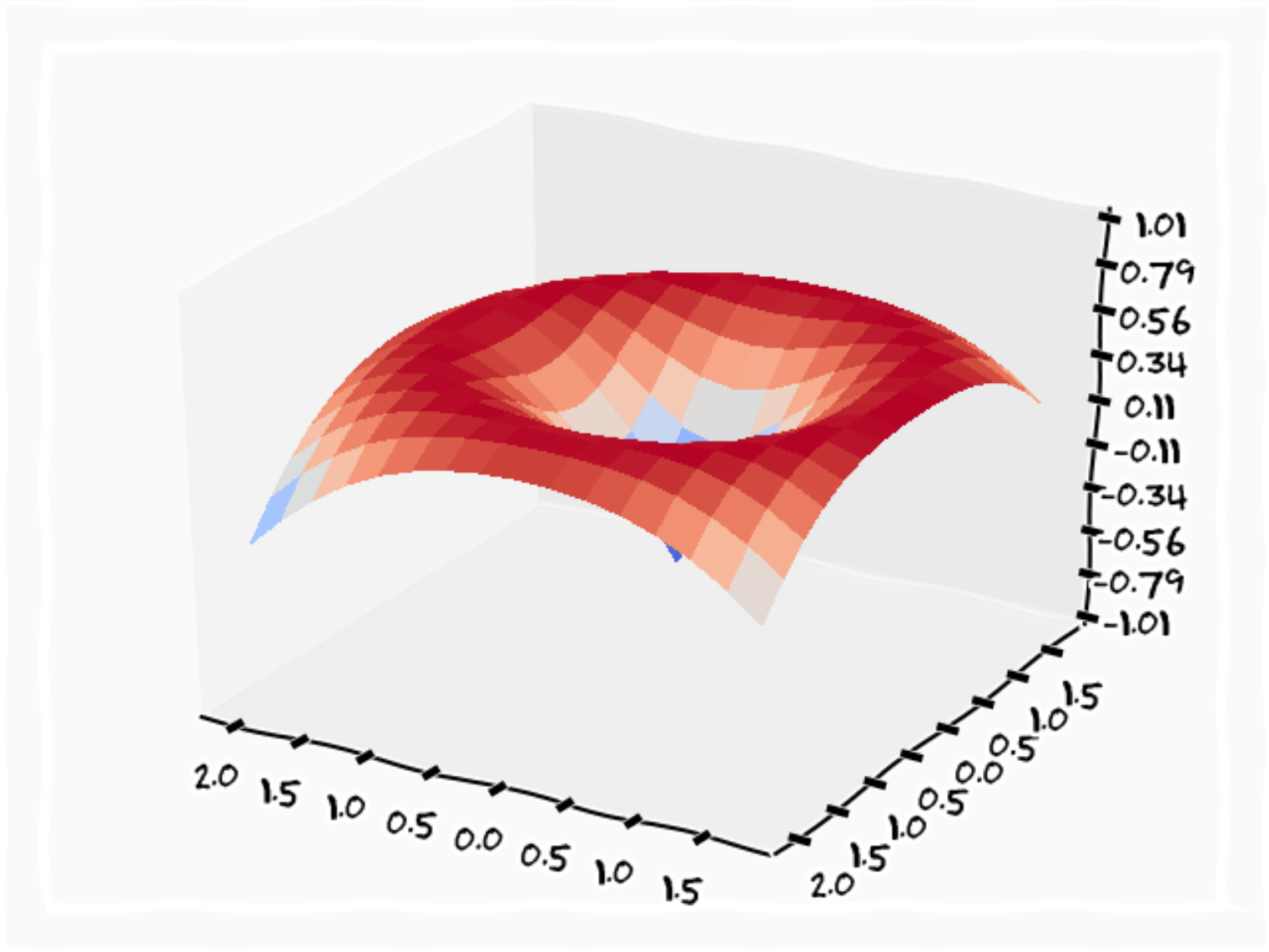
$$p(\mathbf{y}|\mathbf{x}, \theta)$$

Unsupervised Learning model the data

$$\mathcal{D} = \{y_i\}_{i=1}^{N}$$

$$p(\mathbf{y}|\theta)$$

Structure

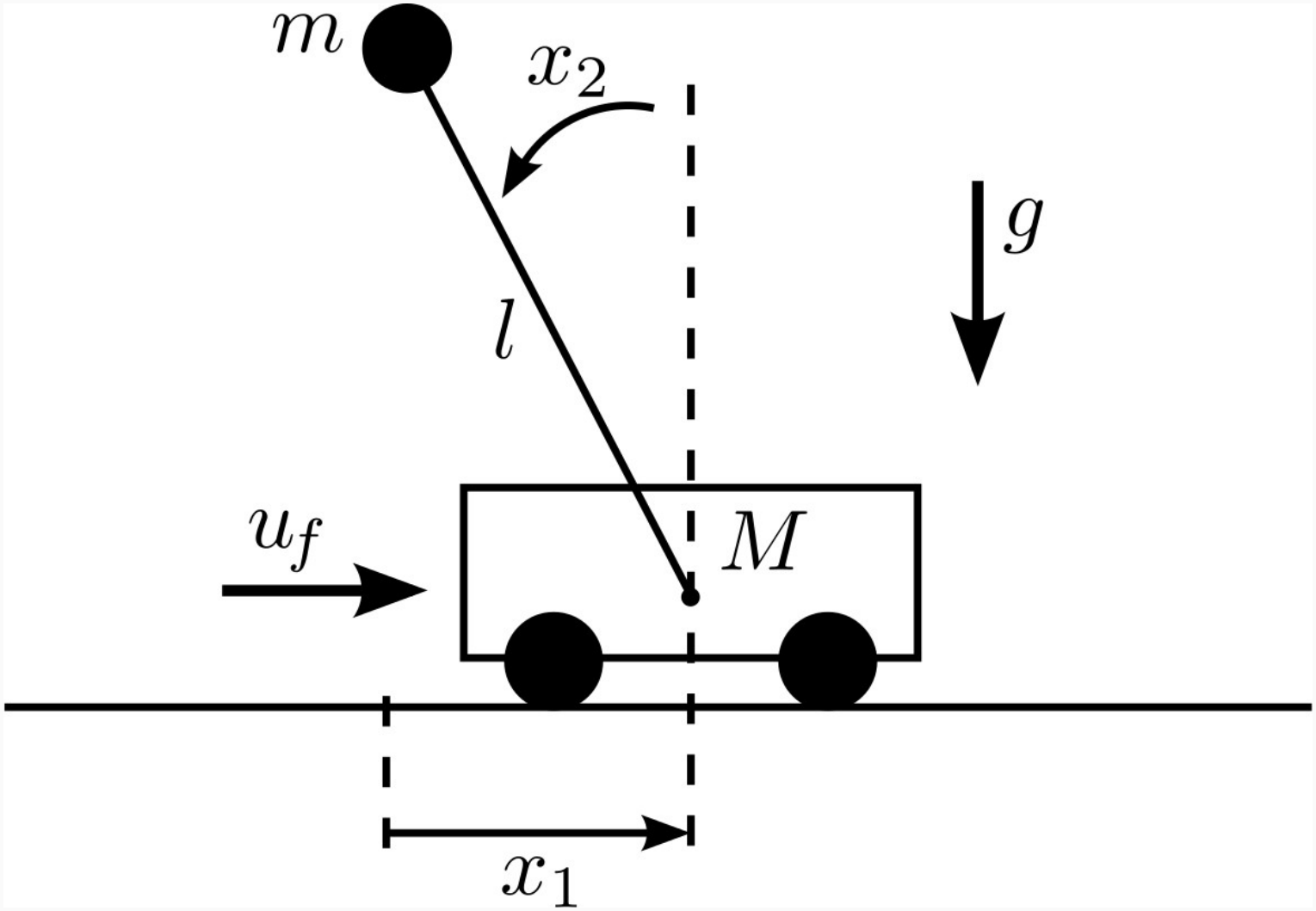

1.01
0.79
0.56
0.34
0.11
-0.11
-0.34
-0.56
-0.79
-1.01
2.0
1.5
1.0
0.5
0.0
0.5
1.0
1.5
2.0
1.5
1.0
0.5
0.0
0.5
1.0
1.5

Reinforcement

Reinforcement Learning

Can we learn without specifying how the task should be achieved by providing, rewards (positive) and punishment (negative)?

Inverted Pendlum[1]

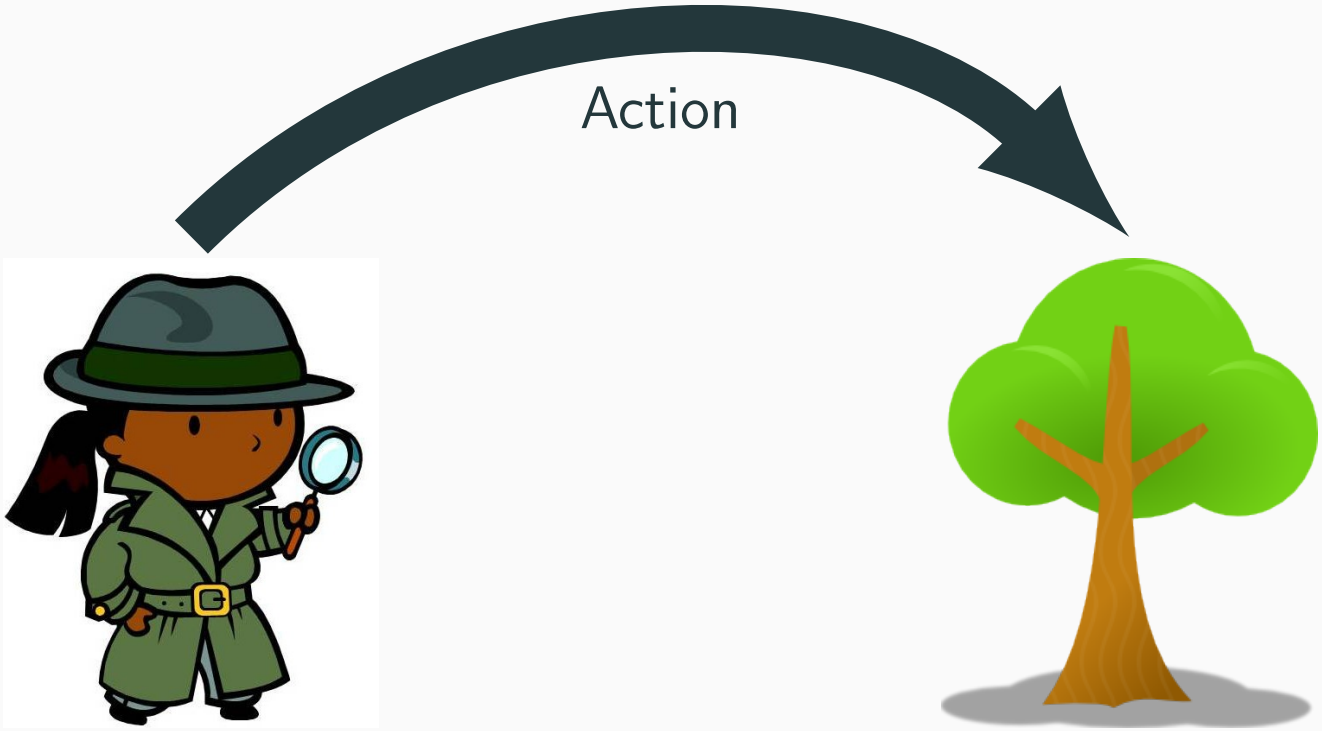


[1] https://www.youtube.com/watch?v=XiigTGKZfks

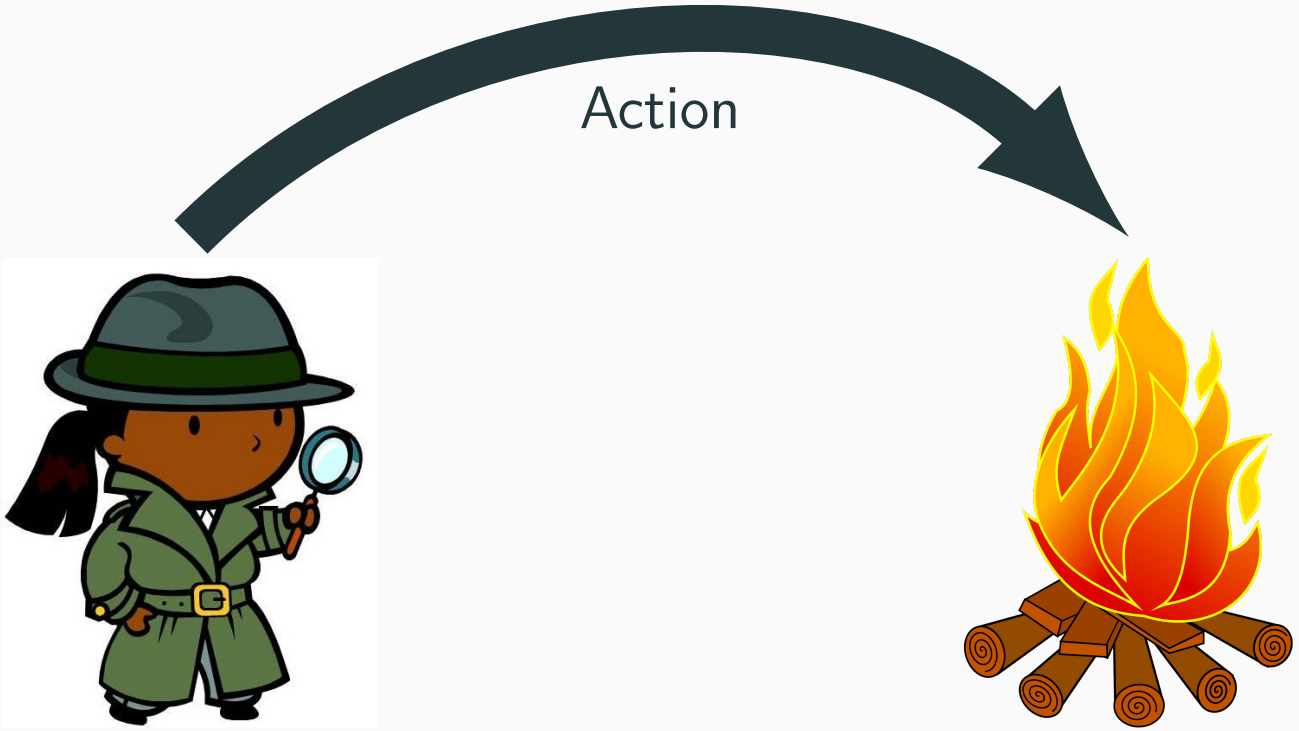
Formalism

Formalism

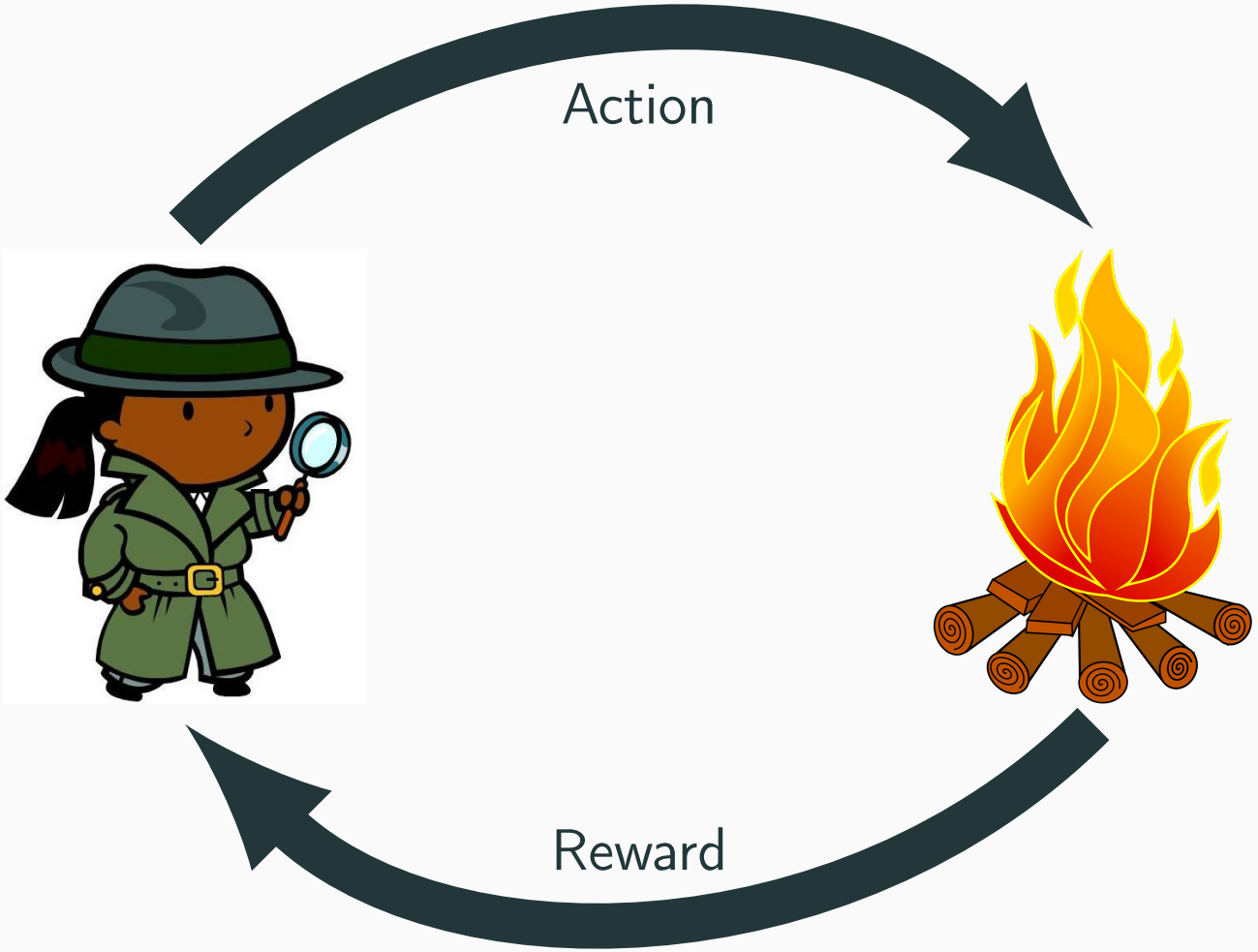
Formalism



Formalism



Formalism



Example: Rollout

Environment you are in state 65 you have 4 possible actions

Example: Rollout

Environment you are in state 65 you have 4 possible actions

Agent I take action 2

Example: Rollout

Environment you are in state 65 you have 4 possible actions

Agent I take action 2

Environment you recieved reinforcement of 7 units, you are now in state 15 you have 2 possible actions

Example: Rollout

Environment you are in state 65 you have 4 possible actions

Agent I take action 2

Environment you recieved reinforcement of 7 units, you are now in state 15 you have 2 possible actions

Agent I take action 1

Example: Rollout

Environment you are in state 65 you have 4 possible actions

Agent I take action 2

Environment you recieved reinforcement of 7 units, you are now in state 15 you have 2 possible actions

Agent I take action 1

Environment you recieved reinforcement of -4 units, you are now in state 65 you have 4 possible actions

Example: Rollout

Environment you are in state 65 you have 4 possible actions

Agent I take action 2

Environment you recieved reinforcement of 7 units, you are now in state 15 you have 2 possible actions

Agent I take action 1

Environment you recieved reinforcement of -4 units, you are now in state 65 you have 4 possible actions

Agent I take action 2

Example: Rollout

Environment you are in state 65 you have 4 possible actions

Agent I take action 2

Environment you recieved reinforcement of 7 units, you are now in state 15 you have 2 possible actions

Agent I take action 1

Environment you recieved reinforcement of -4 units, you are now in state 65 you have 4 possible actions

Agent I take action 2

Environment you recieved reinforcement of 5 units, you are now in state 44 you have 5 possible actions

Optimal Behaviour

Optimal Behaviour

- Finite time horizon

$$E\left[\sum_{t=0}^{h} r_t\right]$$

Optimal Behaviour

- Finite time horizon

$$E\left[\sum_{t=0}^{h} r_t\right]$$

- Average reward

$$\lim_{h\to\infty} E\left[\frac{1}{h}\sum_{t=0}^{h} r_t\right]$$

Optimal Behaviour

- Finite time horizon

$$E\left[\sum_{t=0}^{h} r_t\right]$$

- Average reward

$$\lim_{h\to\infty} E\left[\frac{1}{h}\sum_{t=0}^{h} r_t\right]$$

- Infinite horizon (discounted reward)

$$E\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$$

$$0 \leq \gamma \leq 1$$

Markov Decision Process

Fully observable system

- Transition matrix

$$T(s, a, s') = p\left(s(t+1) = s' | s(t) = s, a(t) = a\right)$$

- Reward matrix

$$R(s, a, s') = E\left(s(t) = s, a(t) = a, s(t+1) = s'\right)$$

- For this set-up the optimal policy can be computed using Dynamic Programming

Markov Decision Process

Fully observable system

- Transition matrix

$$T(s, a, s') = p\left(s(t+1) = s' | s(t) = s, a(t) = a\right)$$

- Reward matrix

$$R(s, a, s') = E\left(s(t) = s, a(t) = a, s(t+1) = s'\right)$$

- For this set-up the optimal policy can be computed using Dynamic Programming
- *We need to be able to enumerate all possible states and actions*

Reinforcement Learning

- Dynamic model

$$p(\mathbf{s}_t|\mathbf{a}_{t-1}, \mathbf{s}_{t-1}, f)$$

- Policy

$$p(\mathbf{a}_t|\mathbf{s}_t, \pi)$$

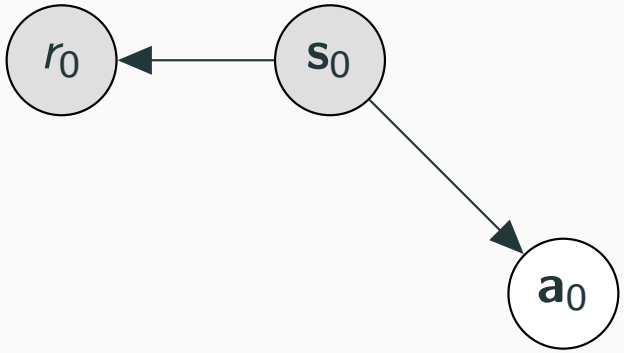
- Reward

$$p(r_t|\mathbf{s}_t)$$

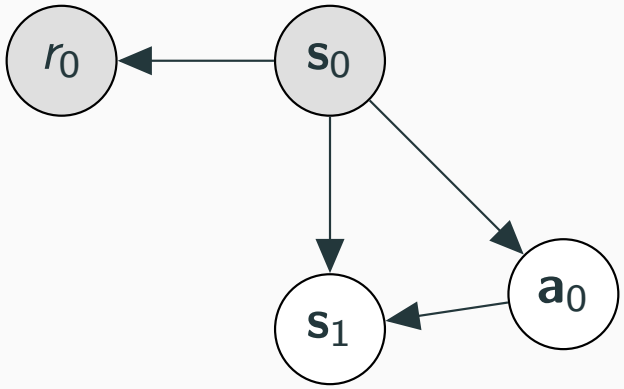
Reinforcement Learning

Reinforcement Learning

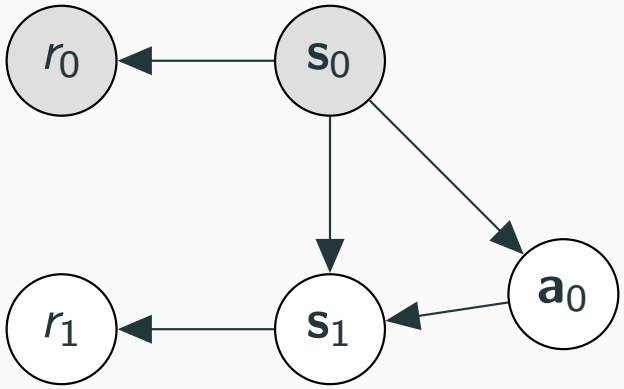
Reinforcement Learning



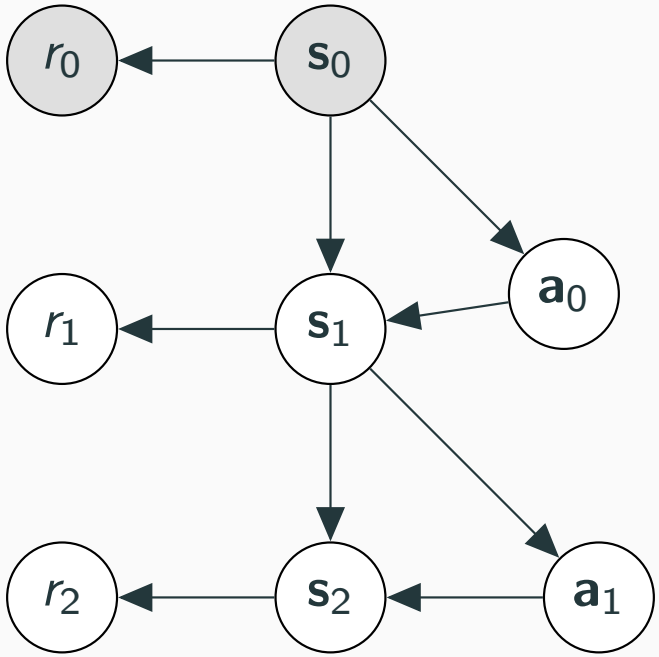
Reinforcement Learning



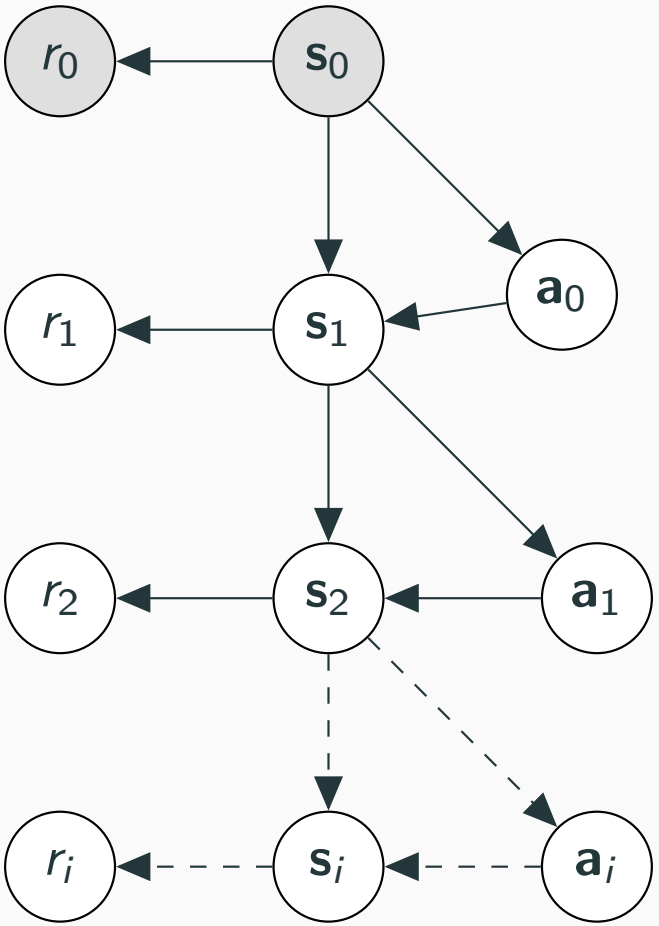
Reinforcement Learning



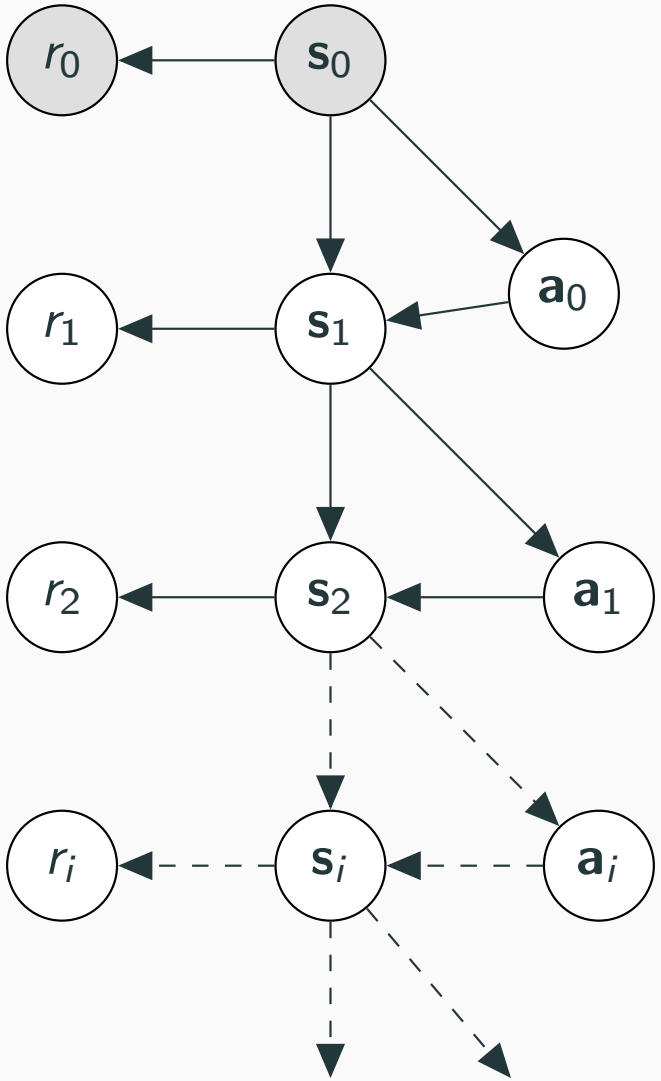
Reinforcement Learning



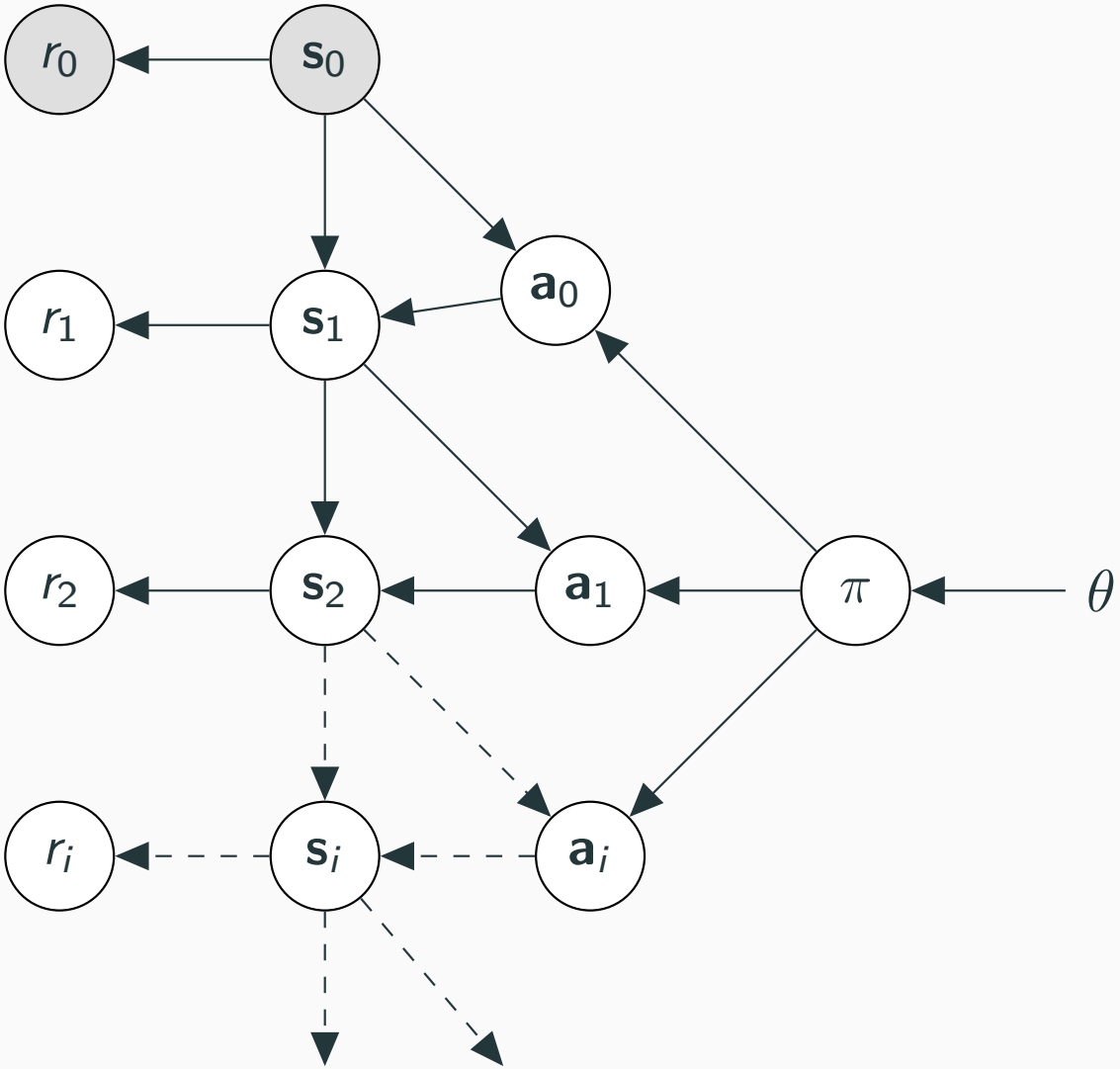
Reinforcement Learning



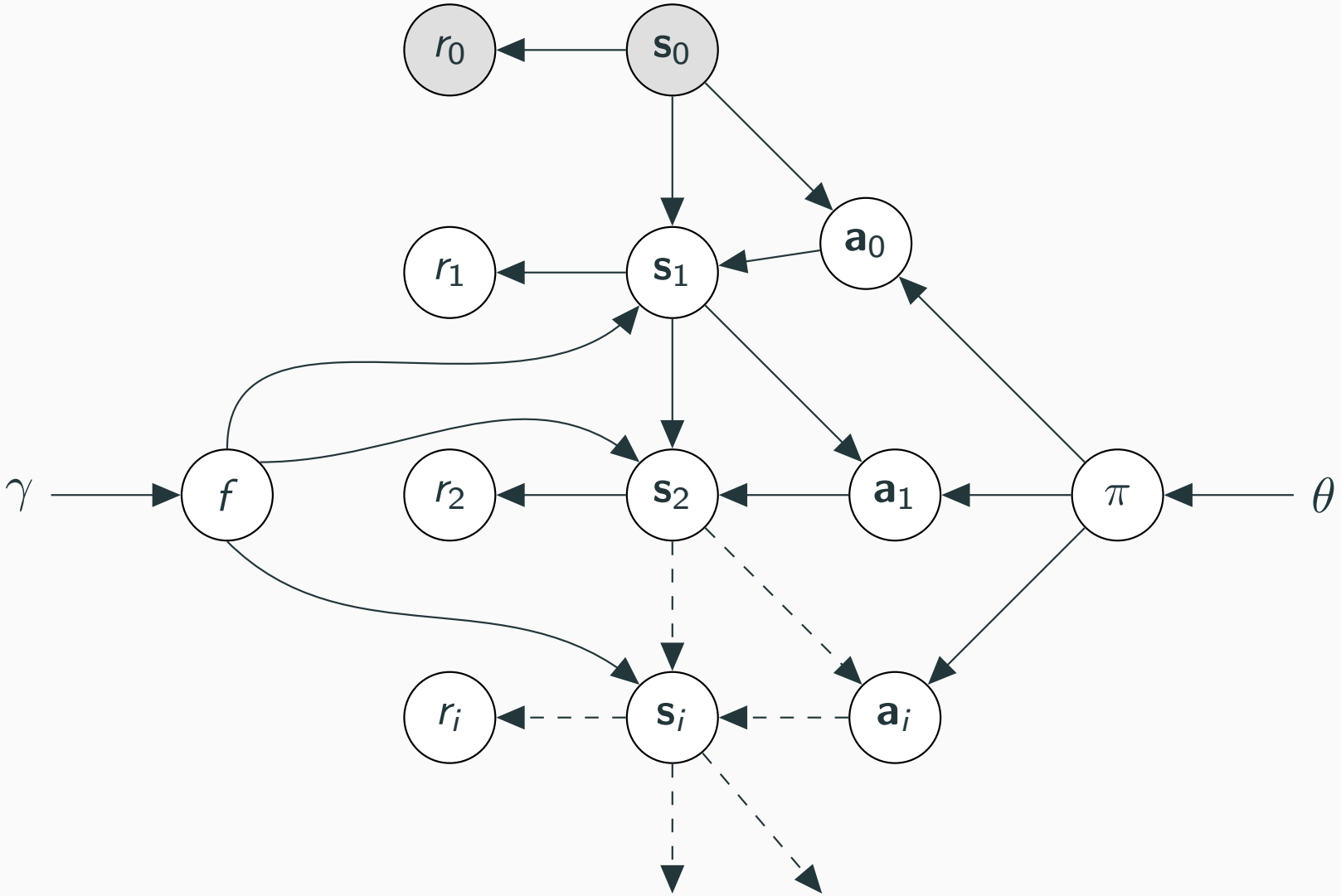
Reinforcement Learning



Reinforcement Learning



Reinforcement Learning



Reinforcment Learning : Rollout

$$
\begin{aligned}
p(\mathbf{s}_{0,\ldots,T}, \mathbf{a}_{0,\ldots,T-1}, r_{0,\ldots,T}, f, \pi, \gamma, \theta | \mathbf{s}_0) = \\
p(\mathbf{s}_T | a_{T-1}, s_{T-1}, f) p(a_{T-1} | s_{T-1}, \pi) \ldots \\
\vdots \\
p(\mathbf{s}_2 | a_1, s_1, f) p(a_1 | s_1, \pi) \\
p(\mathbf{s}_1 | a_0, s_0, f) p(a_0 | s_0, \pi) \\
p(f | \gamma) p(\pi | \theta) p(\gamma) p(\theta)
\end{aligned}
$$

- if we want to learn dynamics and policy we need to marginalise out f, π, θ and γ
- very very hard problem as uncertainty propagated a long way

Reinforcement Learning

$$p(\mathbf{y}_t|\mathbf{s}_t)$$

- We might not be able to observe the state

Model Free

$$Q : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$$

- Learning a model is very hard due to uncertainty propagation
- Directly learn a function from state action sets to reward

Model Free

$$Q : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$$

- Learning a model is very hard due to uncertainty propagation
- Directly learn a function from state action sets to reward
- *Do not build a model, rather a decision system*

Reinforcement Learning

- We can use any availible machine learning method to learn these functions
- Gaussian processes
- Linear regression
- Composite functions (neural networks)
- The tricky thing is how to get the data?

Data

Bandits

Bandit problems

- You are in a room with k slot machines
- Each have a different (unknown) probability of pay-off
- You are permitted h different executions
- Whats the optimal strategy?

Exploration vs. Exploration

- When thinking of a strategy to come up with a policy we need to balance
  - exploring the environment
  - exploiting what we know

Exploration vs. Exploration

- When thinking of a strategy to come up with a policy we need to balance
  - exploring the environment
  - exploiting what we know
- Re-inforcement learning can be thought of in terms of bandit problems
  - sequential bandits
  - delayed reward
  - betting in poker (I think)

Exploration vs. Exploration

- When thinking of a strategy to come up with a policy we need to balance
  - exploring the environment
  - exploiting what we know
- Re-inforcement learning can be thought of in terms of bandit problems
  - sequential bandits
  - delayed reward
  - betting in poker (I think)
- Designing this strategy is the main challenge

Exploration vs. Exploration

- When thinking of a strategy to come up with a policy we need to balance
  - exploring the environment
  - exploiting what we know
- Re-inforcement learning can be thought of in terms of bandit problems
  - sequential bandits
  - delayed reward
  - betting in poker (I think)
- Designing this strategy is the main challenge
- Is Reinforcement Learning Bayesian Optimisation?

Deep Mind

https://www.youtube.com/watch?v=faDKMMwOS2Q

Models vs. Decisions

https://youtu.be/QHcAlAprFxA

Summary

Summary

- Reinforcement learning
  - its nothing different at all, same ideas of modelling
  - how can we learn when we do not know how to do something
  - exploration exploitation
  - if we knew all states, all rewards, all actions it can be done exact

eof

References