Feature Selection for Value Function Approximation Using Bayesian Model Selection

Tobias Jung and Peter Stone

Department of Computer Sciences

University of Texas at Austin

{tjung,pstone}@cs.utexas.edu

Summary: Model Selection for GPTD

Big Picture

Optimization Over Time

Consider: Time-discrete process $t = 0, 1, 2, \ldots$ with

- $\mathcal{X} \subset \mathbb{R}^D$ state space (continuous), $\mathcal{A}$ action space
- Transition probabilities $p(\mathbf{x}_{t+1}|\mathbf{x}_t, a_t)$ (Markov)
- Reward function $R(\mathbf{x}_{t+1}, \mathbf{x}_t, a_t)$ (immediate payoff)

Define: Utility under a policy (expected sum of rewards)

- **Policy** $\pi : \mathcal{X} \to \mathcal{A}$ (deterministic).
- For a given policy π the **value function** (with $\gamma \in (0, 1)$ being a discount factor)

$$\forall \mathbf{x}: \qquad V^\pi(\mathbf{x}) := \mathbb{E}\{\sum_{t \geq 0} \gamma^t R(\mathbf{x}_{t+1}, \mathbf{x}_t, \pi(a_t))|\mathbf{x}_0 = \mathbf{x}\}$$

(where expectation is wrt the randomness of future events)

Goal: Find a policy π^* with maximum utility, i.e. find $\pi^* := \text{argmax}_\pi V^\pi$, an **optimal** policy.

$\Longrightarrow$ Not surprisingly, a vast number of applications: **robotics**, control, AI, game playing, economics & finance, operations research ...

Dynamic Programming/Reinforcement Learning

In theory: One framework to find π^* is **policy iteration**:

- Guess initial policy π_1. For $k = 1, 2, \ldots$
  - Compute V^{π_k} (policy evaluation)
  - Compute improved policy π_{k+1} from V^{π_k} (policy improvement)

In practice: quite tricky to get it right. Lots of open questions. Our focus here: **policy evaluation**.

Approximate policy evaluation (APE):

- **Problem #1:** State space large. $\Longrightarrow$ Function approximation. One good choice: **linear**

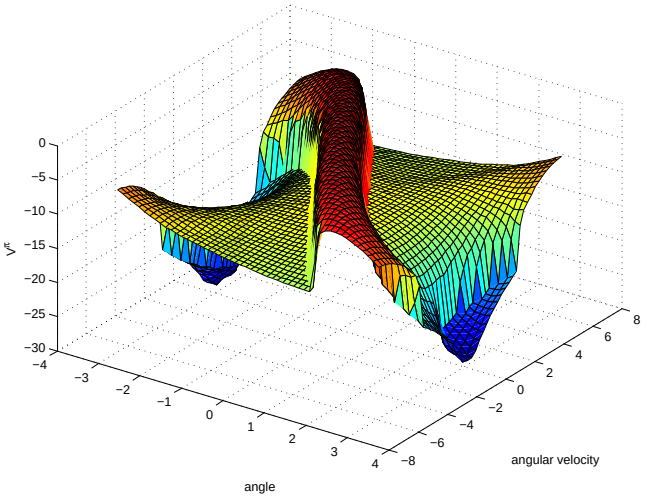
$$V^\pi(\mathbf{x}) \approx \tilde{V}(\mathbf{x}; \mathbf{w}) = \sum_{i=1}^{m} \underbrace{w_i}_{\text{weights}} \underbrace{\phi_i(\mathbf{x})}_{\text{basis functions/features (known)}}$$

- **Problem #2:** System dynamics P, R unknown. $\Longrightarrow$ Instead: sample transitions

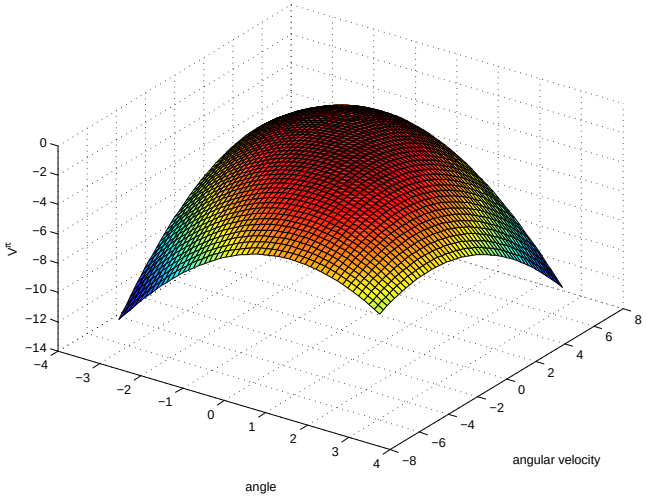
Good news: Given samples and 'good' features, APE is well understood: TD, LSTD, LSPE, BRM, ...

Bad news: What are 'good' features? (How can we find them from the data?)

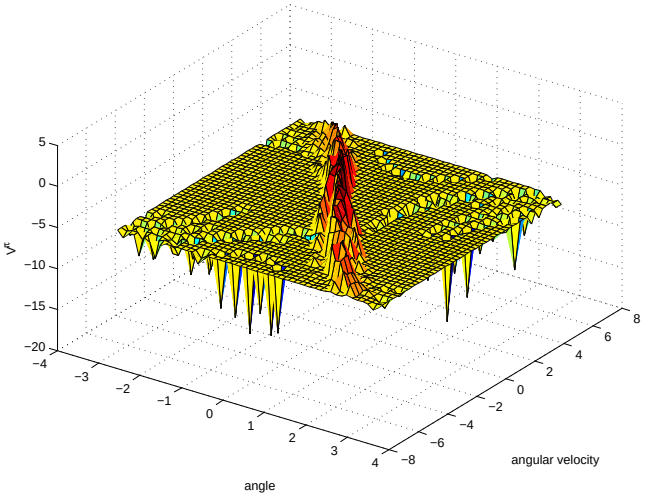
Why is choosing 'good' features difficult?



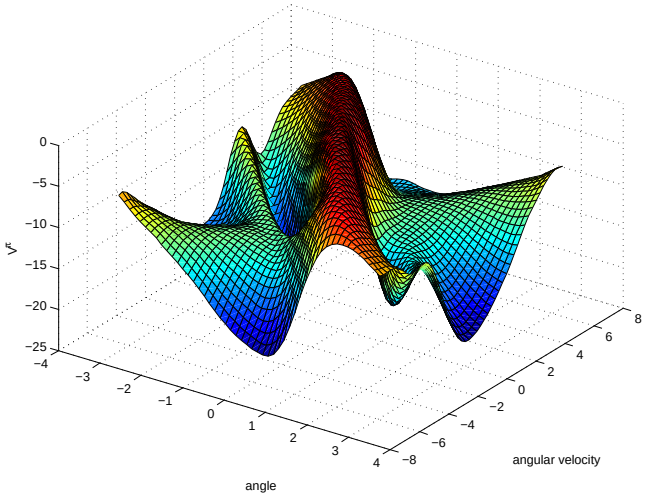
True value function
Error: 0 (MSE)



Manual selection: too smooth
Error: 82.69 (MSE)



Manual selection: too complex
Error: 61.17 (MSE)



Automatic model selection
Error: 12.24 (MSE)

Overview of the talk

Scope: **Dynamic programming/reinforcement learning**

- requires **repeated solution** of least-squares-like problems (policy evaluation)

Problem addressed: **How to find good approximate representations for V^π?**

- **Without the manual tweaking, trial & error usually plaguing RL?**
- Without prior knowledge of the domain? Using just the observed training data?

Our approach: **Leverage modern machine learning techniques:**

- Non-parametric Gaussian processes (no need to worry about individual basis functions)
- Principled framework for model selection (Bayesian)

Novelty:

- Model selection in RL (via marginal likelihood optimization for GPTD)
- Framework for feature selection: find & eliminate irrelevant state variables/directions
  - improves generalization & prediction performance
  - reduces runtime complexity
- Empirical demonstration: it works! (despite minor violation of theoretical assumptions)

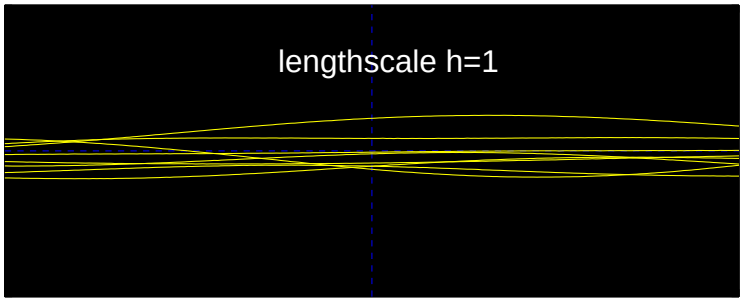
Background GPTD

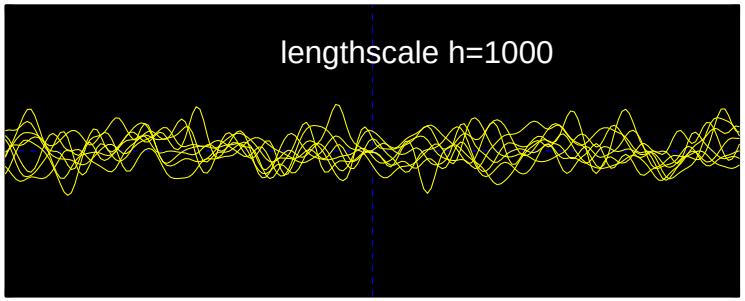
Why Use GPs for APE?

Non-parametric: Instead of *individual basis functions*, specify *class of functions* via

- **Smoothness:** how much $V(\mathbf{x}), V(\mathbf{x}')$ can vary in relation to distance of $\mathbf{x}, \mathbf{x}'$
- **Gaussian process:** class of functions -> distribution over functions (Gaussian) (prior)
  smoothness -> covariance

Example: Let covariance $k_\theta(\mathbf{x}, \mathbf{x}') = \exp\{-h \|\mathbf{x} - \mathbf{x}'\|^2\}$





Important practical advantages:

- **Easy to use:** only have to specify k_θ or its hyperparameter (e.g. one scalar)
- **Linear:** efficient + robust
  - Closed form solution (simple linear algebra, efficient implementation BLAS/LAPACK)
  - Convergence APE
- **Model selection:** good values for hyperparameters can be found automatically (from data)
- **In practice:** good performance; at least equal to well-tuned NNs, but without the hassles ...

GPTD (Summary)

Training data: Observed transitions under π

- sequence of states $\mathbf{X} := [\mathbf{x}_1, \ldots, \mathbf{x}_n]$, where $\mathbf{x}_i \sim p(\cdot\,|\mathbf{x}_{i-1}, \pi(\mathbf{x}_{i-1}))$ 'Inputs'
- associated rewards $\mathbf{r} := [r_1, \ldots, r_{n-1}]$, where $r_i := R(\mathbf{x}_i, \mathbf{x}_{i+1}, \pi(\mathbf{x}_i))$ 'Targets'

Note: Unlike ordinary regression, in RL we cannot observe samples from V directly. Instead: recursion

value of one state = value of successor state + reward (Bellman equation)

GPTD for stochastic transitions (Engel et al. 2003, 2005)

$$\mathbf{r}\,|\,\mathbf{X}, \boldsymbol{\theta} \;\sim\; \mathcal{N}(\mathbf{0}, \mathbf{Q}), \qquad \text{where } \mathbf{Q} := (\mathbf{H}\mathbf{K}\mathbf{H}^\mathsf{T} + \sigma_0^2\mathbf{H}\mathbf{H}^\mathsf{T}), [\mathbf{K}]_{ij} := k_{\boldsymbol{\theta}}(\mathbf{x}_i, \mathbf{x}_j)$$

To predict: the function value $V(\mathbf{x}^*)$ at a new state $\mathbf{x}^*$, we have

$$V(\mathbf{x}^*)\,|\,\mathbf{X}, \mathbf{r}, \mathbf{x}^*, \boldsymbol{\theta} \;\sim\; \mathcal{N}(\boldsymbol{\mu}(\mathbf{x}^*), \sigma^2(\mathbf{x}^*))$$

where

$$\begin{aligned} \boldsymbol{\mu}(\mathbf{x}^*) &:= \overbrace{\mathbf{k}(\mathbf{x}^*)^\mathsf{T}}^{\text{feature vector}}\ \overbrace{\mathbf{H}^\mathsf{T}\mathbf{Q}^{-1}\mathbf{r}}^{\text{weights}} \\ \sigma^2(\mathbf{x}^*) &:= k(\mathbf{x}^*, \mathbf{x}^*) - \mathbf{k}(\mathbf{x}^*)^\mathsf{T}\mathbf{H}^\mathsf{T}\mathbf{Q}^{-1}\mathbf{H}\mathbf{k}(\mathbf{x}^*). \end{aligned}$$

Note: to make all of this work, all we need to know is data + hyperparameters $\boldsymbol{\theta}$ (incl. noise)

SR Approximation for GPTD

Of course, it's not that easy ...

Problem: training $\mathcal{O}(n^3)$, memory $\mathcal{O}(n^2)$, prediction $\mathcal{O}(n)$

Subset of regressors: (well known for ordinary GPs, here for GPTD)

- Approximate kernel from subset: $k(\mathbf{x}, \mathbf{x}') \approx \mathbf{k}_m(\mathbf{x})^\intercal \mathbf{K}_{mm}^{-1} \mathbf{k}_m(\mathbf{x}')$, $m \ll n$
- Solve a reduced problem: training $\mathcal{O}(nm^2)$, memory $\mathcal{O}(m^2)$, prediction $\mathcal{O}(m)$
  (details in paper)

Selection of subset:

- In general, supervised and unsupervised methods possible.
- Here: unsupervised. Use: ICD of $\mathbf{K}$ (dual) $\Leftrightarrow$ partial Gram-Schmidt (primal)
- Note:
  - Number m of selected elements will depend on **effective rank** of $\mathbf{K}$ (eigenspectrum)
  - Eigenspectrum of $\mathbf{K}$ $\Leftrightarrow$ complexity of solution (cf. likelihood)

Thus: simpler solutions $\Longrightarrow$ better generalization + **better runtime** (important for RL!)

Model Selection for GPTD

Model Selection

Model selection = finding good hyperparameters $\boldsymbol{\theta}$ **automatically** (in RL currently done manually)

Marginal likelihood for GPTD:

1. Consider likelihood of the data

$$p(\mathbf{r}|\mathbf{X}, \boldsymbol{\theta}) = \mathcal{N}(\mathbf{0}, \mathbf{Q})$$

   as function of hyperparameters $\boldsymbol{\theta}$: (neg loglike)

$$\mathcal{L}(\boldsymbol{\theta}) := -\frac{1}{2}\log\det\mathbf{Q} - \frac{1}{2}\mathbf{r}^\mathsf{T}\mathbf{Q}^{-1}\mathbf{r} - \frac{n}{2}\log 2\pi$$

2. Find $\boldsymbol{\theta}$ that minimizes $\mathcal{L}$
   - Requires iterative gradient-based solver (like cg)
   - Gradient of $\mathcal{L}$ can be obtained in closed form (see paper)

Note: $\mathcal{L}$ consists of two conflicting terms

- **Complexity** $\log\det\mathbf{Q}$ (note: log det = sum of log eigenvals)
- **Data fit** $\mathbf{r}^\mathsf{T}\mathbf{Q}^{-1}\mathbf{r}$

Generally: it's either/or

- large bandwidth -> high complexity (large effective rank) -> low data error
- small bandwidth -> low complexity (small effective rank) -> high data error

Automatic relevance determination

Automated procedure for hyperparameter selection:

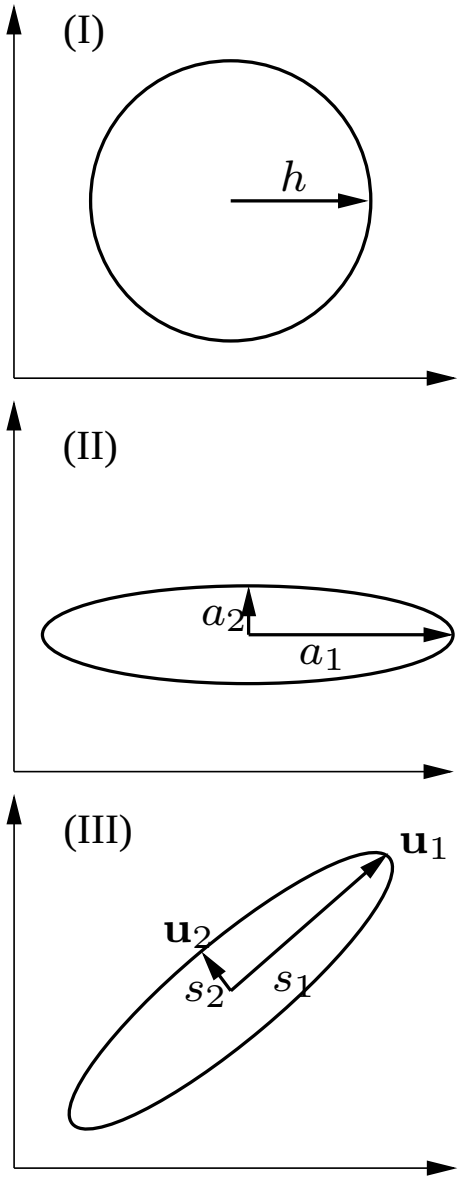
- $\Longrightarrow$ can use cov with larger number of hyperparameters
  (infeasible to set by hand)
- $\Longrightarrow$ better fit regularities of data, remove what is irrelevant

Covariance: We consider three variants of the form:

$$k_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{x}') = v_0 \exp\left\{-\frac{1}{2}(\mathbf{x}-\mathbf{x}')^{\mathsf{T}}\boldsymbol{\Omega}(\mathbf{x}-\mathbf{x}')\right\} + b$$

with scalar hyperparameters v_0, b and matrix $\boldsymbol{\Omega}$ given by

- **Variant I:** $\boldsymbol{\Omega} = h\mathbf{I}$.
- **Variant II:** $\boldsymbol{\Omega} = \mathrm{diag}(a_1, \ldots, a_D)$.
- **Variant III:** $\boldsymbol{\Omega} = \mathbf{M}_k\mathbf{M}_k^{\mathsf{T}} + \mathrm{diag}(a_1, \ldots, a_D)$.



Note:

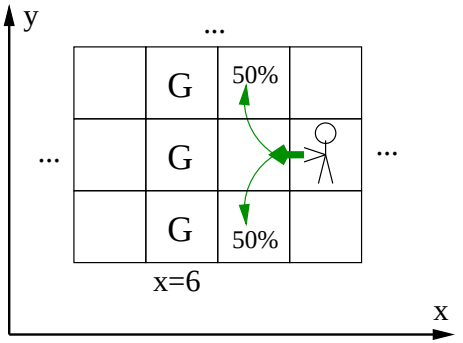
- (II), (III) contain adjustable parameters for every state variable
- Setting them automatically from data $\Longrightarrow$
  Model selection automatically determines their relevance

Experiments

Experiment 1a: 2D gridworld

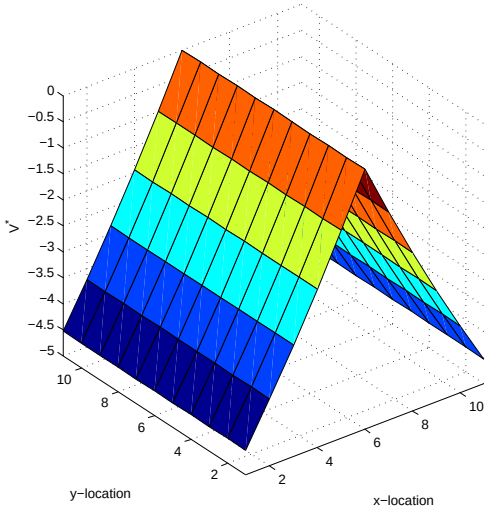
Scenario:

- 2D gridworld (11×11 cells)
- -1 per step, except when in goal G
- stochastic transitions
- **y-coordinate irrelevant for predicting V^π**



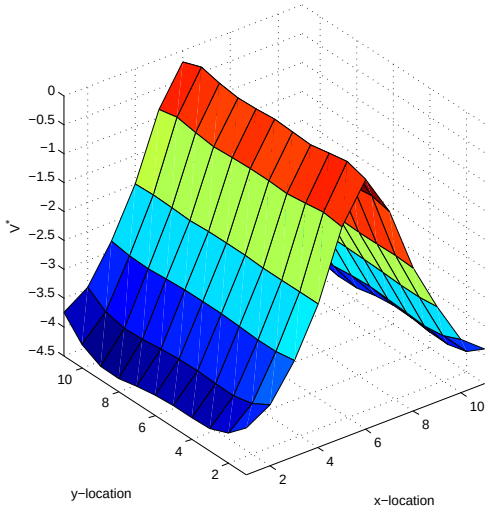
Results:

- sample 500 transitions under optimal policy, GPTD with (I),(II)
- whole learning was fully automated



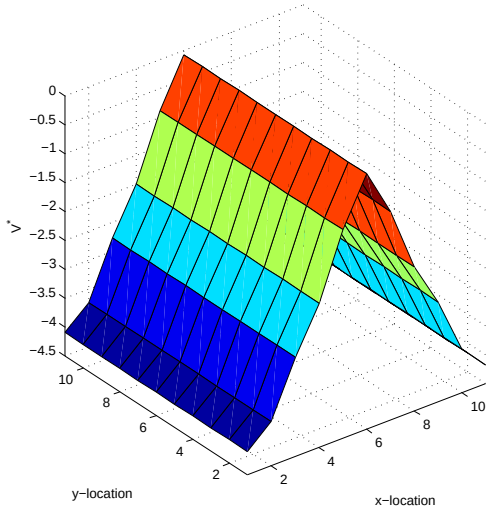
true value function

Error: 0 (MSE)



using (I), no ARD

Error: 0.030 (MSE)



using (II), with ARD

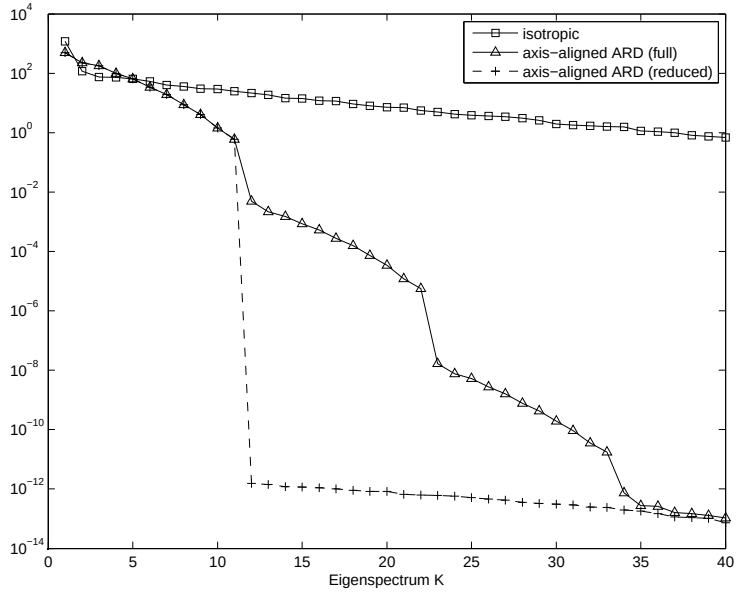
Error: 0.019 (MSE)

Analysis

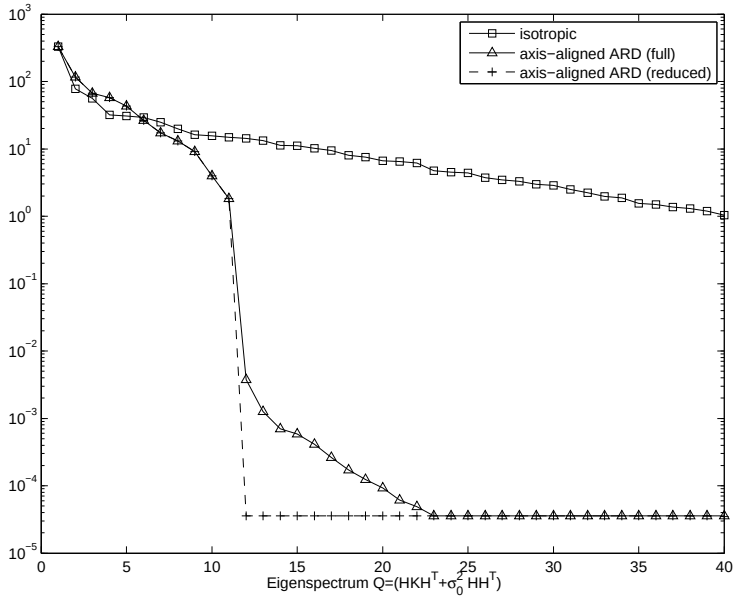
Results from model selection:

	Hyperparameters $\boldsymbol{\theta}$	Complexity	Data fit	$\mathcal{L}$ (smaller is better)
(I)	$h = 2.89$	-2378.2	54.78	-2323.4
(II)	$a_1 = 3.53 \quad a_2 = 10^{-5}$	-2772.7	13.84	-2758.8
(II) without y	$a_1 = 3.53 \quad a_2 = 0$	-2790.7	13.84	-2776.8

Analysis: How a data-adapted covariance reduces complexity of the model



Eigenvalues of $\mathbf{K}$

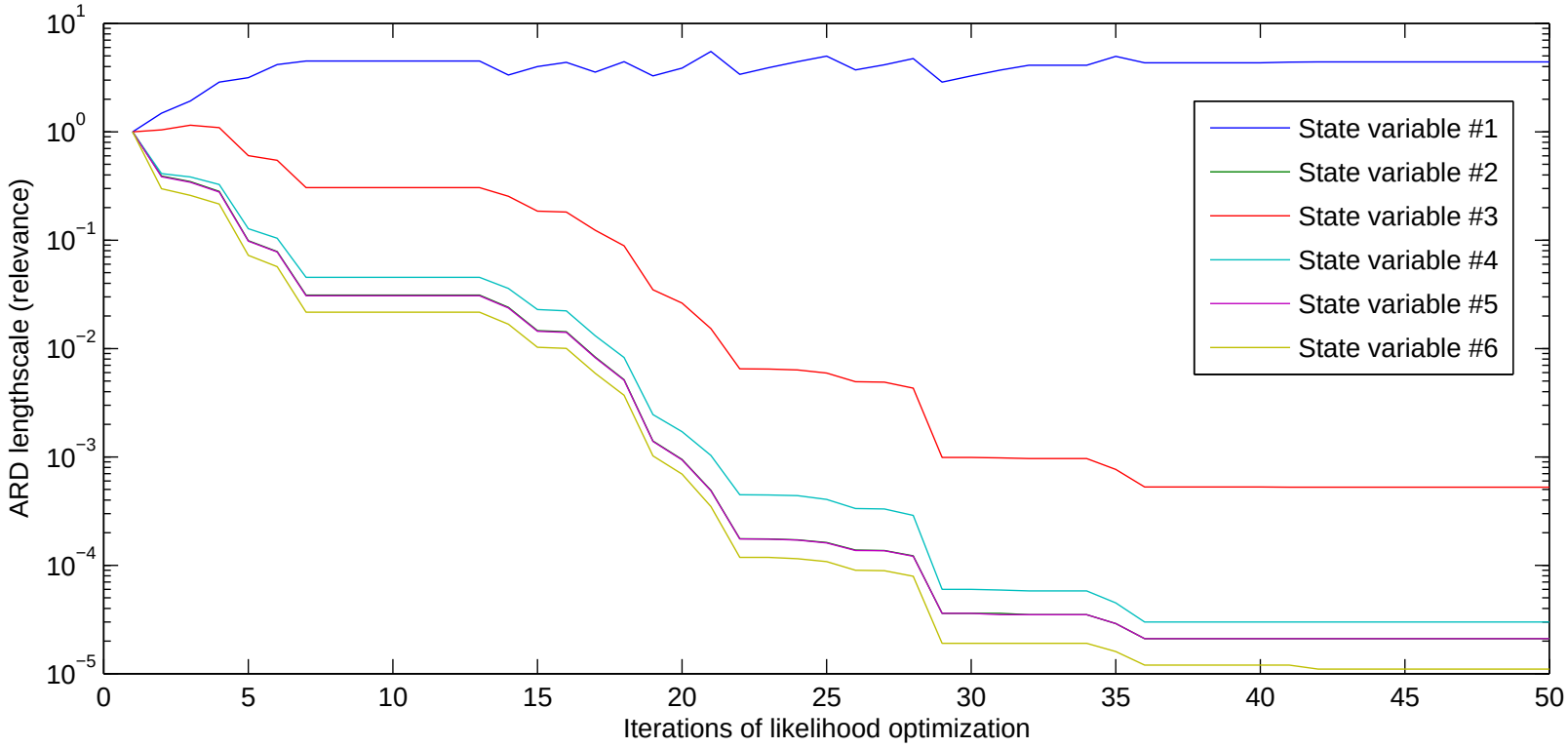


Eigenvalues of $\mathbf{Q}$

Experiment 1b: 6D gridworld

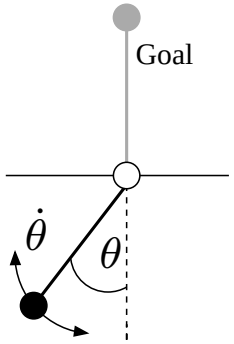
Even as we add more (irrelevant) state variables, optimization of the marginal likelihood with the ARD kernel correctly identifies those that matter:

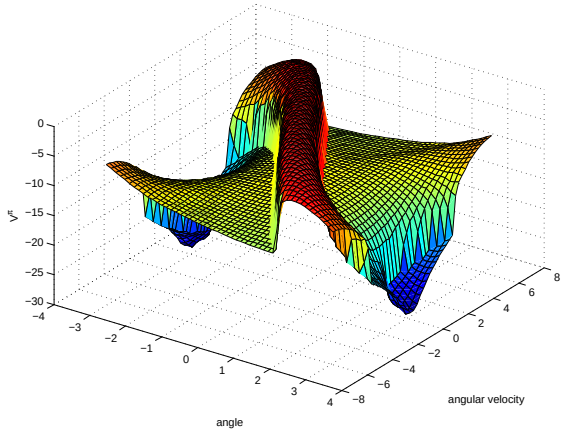
$$\text{6D state } \mathbf{x} = \begin{bmatrix} x & y & x + \text{small noise} & x + \text{large noise} & y + \text{small noise} & y + \text{large noise} \end{bmatrix}^{\mathsf{T}}$$



Experiment 2: Inverted pendulum

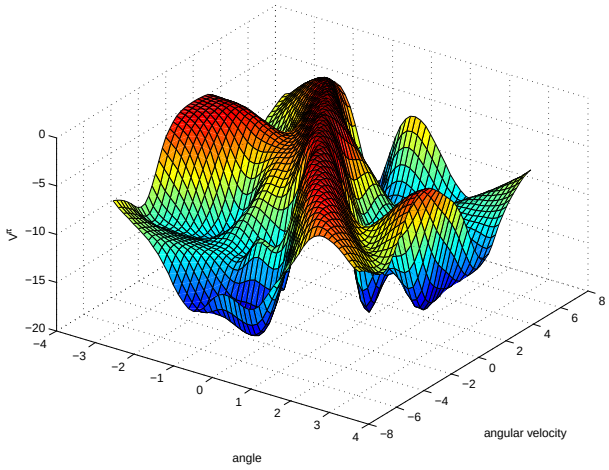
Scenario: a more realistic benchmark



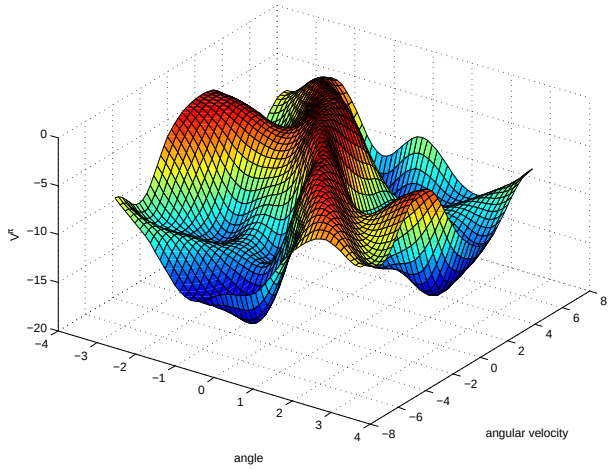


(optimal V^{π^*})

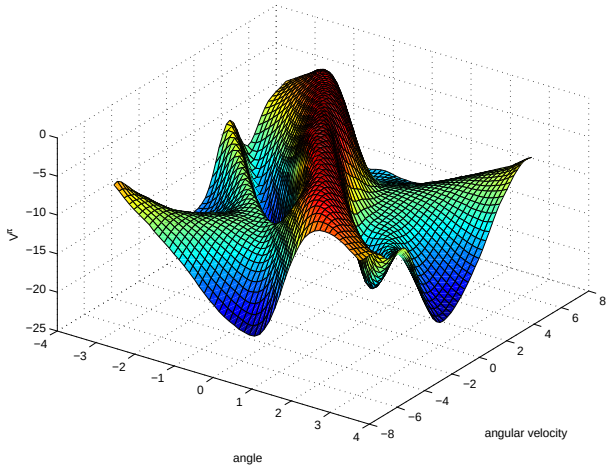
Results: 1000 transitions from optimal policy, GPTD for (I), (II), (III)



(I) Error: 46.36 (MSE)



(II) Error: 48.89 (MSE)

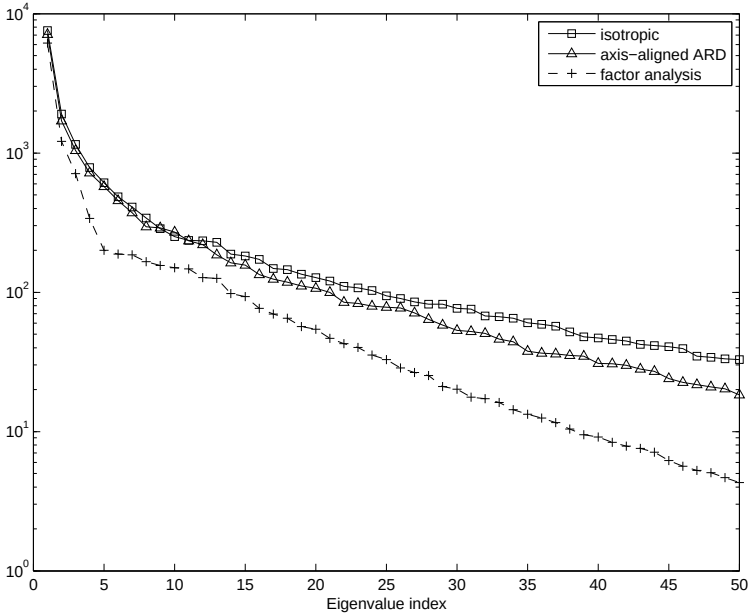


(III) Error: 12.24 (MSE)

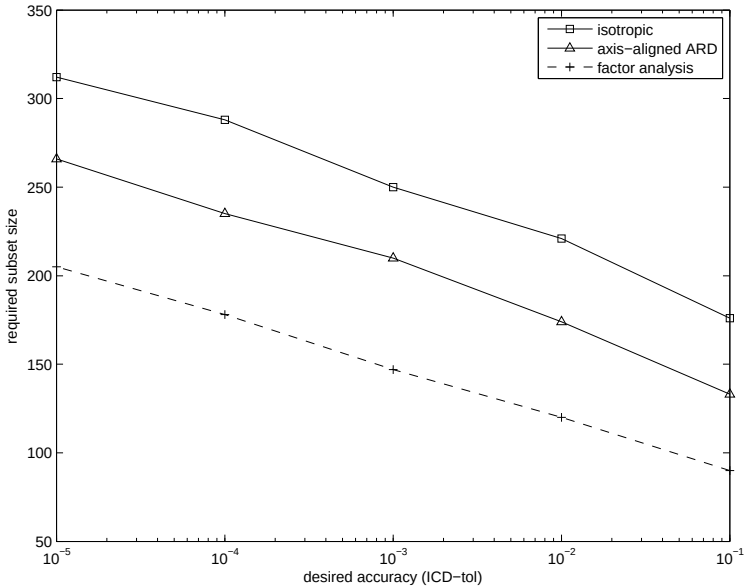
Analysis

Analysis:

- No irrelevant state variable but
  - (III) finds **dominant direction**
  - (II) is restricted to axis aligned directions (same as (I))
- Consequence:
  - (III) achieves best generalization with the least complex model
  - the least complex model also requires the least computational resources



eigenspectrum



number of elements ICD selects

Finish

Summary (so far):

- Framework for **automatic** feature selection/generation in RL
  - based on GPs as underlying function approximator
  - based on Monte-Carlo rollouts/LSTD(1) as policy evaluation method (=GPTD)
  - based on likelihood-based model selection
- Framework doesn't come with theoretical guarantees (violates some independence assumptions)
- **Framework seems to work in practice**

Ongoing work:

- Solve the full optimal control problem, i.e. do policy iteration
  - requires policy improvement
  - requires exploration (or strategy for sample generation)
  - may require extension to joint state-action space (Q-function)
- More complex experiments/simulations.
- Gain more theoretical insights, e.g. when will GPTD fail?
- Compare with other policy evaluation methods, like LSTD, LSPE, ...

Related work

GPTD:

[1] Y.Engel, S. Mannor, and R. Meir. Bayes meets Bellman: The Gaussian process approach to temporal difference learning. ICML 20, 2003

[2] Y.Engel, S. Mannor, and R. Meir. Reinforcement learning with Gaussian processes. ICML 22, 2005

Adaptation of basis functions in RL:

[3] N. Menache, N. Shimkin, and S. Mannor. Basis function adaptation in temporal difference reinforcement learning. Annals of Operations Research, 134:215-238

Other:

[4] P. Keller, S. Mannor, and D. Precup. Automatic basis function construction for approximate dynamic programming and reinforcement learning. ICML 23, 2006

[5] R. Parr, C. Painter-Wakefield, L. Li, and M. Littman. Analyzing feature generation for value-function approximation. ICML 24, 2007

[6] S. Mahadevan and M. Maggioni. Proto-value functions. A Laplacian framework for learning representation and control in Markov decision processes. JMLR, 8:2169-2231, 2007

[7] J. Reisinger, P. Stone, and R. Mikkulainen. Online kernel selection for Bayesian reinforcement learning. ICML 25, 2008