Apprenticeship learning with DNN (with experiments)

Jeong Gwan Lee

KAIST

(Korea Advanced Institute of Science and Technology)

What I implemented,

1. LSPI
2. basis function : sigmoid, RBF, ANNCP, DAN(h2, output)
3. Batch, off-policy and model-free Apprenticeship learning(LSTD-mu)
4. Deep Action Network(DAN) + DQN

Batch, Off-Policy and Model-Free Apprenticeship Learning(BOMAL)

Klein, Edouard, Matthieu Geist, and Olivier Pietquin. "Batch, off-policy and model-free apprenticeship learning." *European Workshop on Reinforcement Learning*. Springer, Berlin, Heidelberg, 2011.

Apprentice Learning,

is a learning approach, which assumes that the reward function is unknown, but instead, *expert knowledge* is available.

This *expert knowledge* is in the form of sequences of states and actions, trajectories, where the goal state is achieved.

The main idea is to find a optimal policy close to the *expert policy* π^E, to use these trajectories.

Inverse Reinforcement Learning(IRL)

Reward function is parameterized as linear combination of features,

$$R(s) = \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{s}) \quad s.t. \quad \boldsymbol{\phi} : S \rightarrow [0,1]^k,$$

w is the weight vector to be learned,
ϕ is the features vector(basis function),
k is the number of features.

Feature expectations, denoted as μ,

$$\boldsymbol{\mu}(\boldsymbol{\pi}) = E[\sum_{t=0}^{\infty} \gamma^t \boldsymbol{\phi}(\boldsymbol{s_t}) | \pi] \in \mathbb{R}^k.$$

Estimated feature expectations for expert policy,

$$\hat{\boldsymbol{\mu}}_{\boldsymbol{E}} = \frac{1}{m} \sum_{i=0}^{m} \sum_{t=0}^{\infty} \gamma^t \boldsymbol{\phi}(\boldsymbol{s_t^{(i)}}),$$

Learning process would be repeated until, $\|\mu(\pi) - \hat{\mu}_E\| < \epsilon$

MDP\R set-up

Why ”reward(R)”, “parameter(θ)” and “basis function($\phi(s)$)” should be bounded?

$$\mathcal{H}_\phi = \{\theta^T \phi(s), \theta \in \mathbb{R}^p\}, \qquad R = \theta^T \phi(s) = \theta_1 \phi_1(s) + \theta_2 \phi_2(s) + \cdots + \theta_p \phi_p(s)$$

- *(paper)* $R \in \mathbb{R}^S$ the reward function (assumed to be absolutely bounded by 1)

$$|R| \leq 1$$

- *(paper)* we assume the basis functions to be bounded by 1:

$$|\phi_i(s)| \leq 1 \qquad 1 \leq i \leq p$$

- *(paper)* In order to ensure that rewards are bounded, we impose that $\|\theta\|_2 \leq 1$

$$\|\theta\|_2 \leq 1$$

$$\mu^\pi(s) = E[\sum_{t=0}^{\infty} \gamma^t \phi(s_t) | s_0 = s, \pi] \qquad |V^{\pi_E}(s_0) - V^{\tilde{\pi}}(s_0)| = |\theta^T(\mu^{\pi_E}(s_0) - \mu^{\tilde{\pi}}(s_0))| \leq \|\mu^{\pi_E}(s_0) - \mu^{\tilde{\pi}}(s_0)\|_2$$

MDP\R set-up

Why ”reward(R)”, “parameter(θ)” and “basis function($\phi(s)$)” should be bounded?

$$\mathcal{H}_\phi = \{\theta^T \phi(s), \theta \in \mathbb{R}^p\}, \qquad R = \theta^T \phi(s) = \theta_1 \phi_1(s) + \theta_2 \phi_2(s) + \cdots + \theta_p \phi_p(s)$$

- *(paper)* $R \in \mathbb{R}^S$ the reward function (assumed to be absolutely bounded by 1)

$$|R| \leq 1$$

- *(paper)* we assume the basis functions to be bounded by 1: ✓

$$|\phi_i(s)| \leq 1 \qquad 1 \leq i \leq p$$

- *(paper)* In order to ensure that rewards are bounded, we impose that $\|\theta\|_2 \leq 1$ ✓

$$\|\theta\|_2 \leq 1$$

$$\mu^\pi(s) = E[\sum_{t=0}^{\infty} \gamma^t \phi(s_t) | s_0 = s, \pi] \qquad |V^{\pi_E}(s_0) - V^{\tilde{\pi}}(s_0)| = |\theta^T(\mu^{\pi_E}(s_0) - \mu^{\tilde{\pi}}(s_0))| \leq \|\mu^{\pi_E}(s_0) - \mu^{\tilde{\pi}}(s_0)\|_2$$

MDP\R set-up

Why ”reward(R)”, “parameter(θ)” and “basis function($\phi(s)$)” should be bounded?

$$\mathcal{H}_\phi = \{\theta^T \phi(s), \theta \in \mathbb{R}^p\}, \qquad R = \theta^T \phi(s) = \theta_1 \phi_1(s) + \theta_2 \phi_2(s) + \cdots + \theta_p \phi_p(s)$$

- *(paper)* $R \in \mathbb{R}^S$ the reward function (assumed to be absolutely bounded by 1)

$$|R| \leq 1$$

- *(paper)* we assume the basis functions to be bounded by 1: ✓

$$|\phi_i(s)| \leq 1 \qquad 1 \leq i \leq p$$

- *(paper)* In order to ensure that rewards are bounded, we impose that $\|\theta\|_2 \leq 1$ ✓

$$\|\theta\|_2 \leq 1$$

$$\theta = \begin{bmatrix} 0.7 \\ 0.7 \end{bmatrix} \qquad \|\theta\|_2 = \sqrt{0.7^2 + 0.7^2} = \sqrt{0.98} \leq 1 \qquad \phi(s) = \begin{bmatrix} \phi_1(s) \\ \phi_2(s) \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \qquad R = \theta^T \phi(s) = 0.7 + 0.7 = 1.4 > 1$$

MDP\R set-up

Why ”reward(R)”, “parameter(θ)” and “basis function($\phi(s)$)” should be bounded?

$$\mathcal{H}_\phi = \{\theta^T \phi(s), \theta \in \mathbb{R}^p\}, \qquad R = \theta^T \phi(s) = \theta_1 \phi_1(s) + \theta_2 \phi_2(s) + \cdots + \theta_p \phi_p(s)$$

- *(paper)* $R \in \mathbb{R}^S$ the reward function (assumed to be absolutely bounded by 1)

$$|R| \leq 1$$

- *(paper)* we assume the basis functions to be bounded by 1: ✓

$$|\phi_i(s)| \leq 1 \qquad 1 \leq i \leq p$$

- *(paper)* In order to ensure that rewards are bounded, we impose that $\|\theta\|_2 \leq 1$ ✓

$$\|\theta\|_1 \leq 1 \longrightarrow |\theta_1| + |\theta_2| + \cdots + |\theta_p| \leq 1$$

$$\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \cdots \\ \theta_p \end{bmatrix} \qquad \phi(s) = \begin{bmatrix} \phi_1(s) \\ \phi_2(s) \\ \cdots \\ \phi_p(s) \end{bmatrix}$$

$$|\theta_1||\phi_1(s)| + |\theta_2||\phi_2(s)| + \cdots + |\theta_p||\phi_p(s)| \leq 1 \quad (\because |\phi_i(s)| \leq 1)$$
$$|\theta_1 \phi_1(s)| + |\theta_2 \phi_2(s)| + \cdots |\theta_p \phi_p(s)| \leq 1 \quad (\because |A||B| = |AB|)$$
$$|\theta_1 \phi_1(s) + \theta_2 \phi_2(s) + \cdots + \theta_p \phi_p(s)| \leq 1 \quad (\because |A + B| \leq |A| + |B|)$$
$$R = \theta^T \phi(s) = \theta_1 \phi_1(s) + \theta_2 \phi_2(s) + \cdots + \theta_p \phi_p(s) \leq 1$$

MDP\R set-up

Why ”reward(R)”, “parameter(θ)” and “basis function($\phi(s)$)” should be bounded?

$$\mathcal{H}_\phi = \{\theta^T \phi(s), \theta \in \mathbb{R}^p\}, \qquad R = \theta^T \phi(s) = \theta_1 \phi_1(s) + \theta_2 \phi_2(s) + \cdots + \theta_p \phi_p(s)$$

- *(paper)* $R \in \mathbb{R}^S$ the reward function (assumed to be absolutely bounded by 1)

$$|R| \leq 1$$

- *(paper)* we assume the basis functions to be bounded by 1: ✓

$$|\phi_i(s)| \leq 1 \qquad 1 \leq i \leq p$$

- *(paper)* In order to ensure that rewards are bounded, we impose that $\|\theta\|_2 \leq 1$ ✓

$$\|\theta\|_2 \leq 1$$

$$\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \cdots \\ \theta_p \end{bmatrix} \qquad \phi(s) = \begin{bmatrix} \phi_1(s) \\ \phi_2(s) \\ \cdots \\ \phi_p(s) \end{bmatrix} \qquad R = \theta^T \phi(s) < C \ (C : constant, p \ is \ bounded.)$$

MDP\R set-up

Why ”reward(R)”, “parameter(θ)” and “basis function($\phi(s)$)” should be bounded?

$$\mathcal{H}_\phi = \{\theta^T \phi(s), \theta \in \mathbb{R}^p\}, \qquad R = \theta^T \phi(s) = \theta_1 \phi_1(s) + \theta_2 \phi_2(s) + \cdots + \theta_p \phi_p(s)$$

- *(paper)* $R \in \mathbb{R}^S$ the reward function (assumed to be absolutely bounded by 1)

$$|R| \leq 1$$

- *(paper)* we assume the basis functions to be bounded by 1: ✓

$$|\phi_i(s)| \leq 1 \qquad 1 \leq i \leq p$$

- *(paper)* In order to ensure that rewards are bounded, we impose that $\|\theta\|_2 \leq 1$ ✓

$$\|\theta\|_2 \leq 1$$

$$\|\theta\|_2 > 1$$

$$\mu^\pi(s) = E[\sum_{t=0}^{\infty} \gamma^t \phi(s_t) | s_0 = s, \pi] \qquad |V^{\pi_E}(s_0) - V^{\tilde{\pi}}(s_0)| = |\theta^T(\mu^{\pi_E}(s_0) - \mu^{\tilde{\pi}}(s_0))|$$

$$\|\mu^{\pi_E}(s_0) - \mu^{\tilde{\pi}}(s_0)\|_2 \leq \epsilon \text{ for some (small) } \epsilon > 0$$

IRL Algorithm

initial policy -> $\mu(s)$ -> $R = \theta^T \phi(s)$ -> policy(LSPI) -> $\mu(s)$ -> $R = \theta^T \phi(s) =$ -> policy(LSPI) …

1. Starts with some initial policy $\pi^{(0)}$ and compute $\mu^{\pi^{(0)}}(s_0)$. Set $j = 1$;

2. Compute $t^{(j)} = \max_{\theta:\|\theta\|_2 \leq 1} \min_{k \in \{0,j-1\}} \theta^T (\mu^{\pi_E}(s_0) - \mu^{\pi^{(k)}}(s_0))$ and let $\theta^{(j)}$ be the value attaining this maximum. At this step, one searches for the reward function which maximizes the distance between the value of the expert at s_0 and the value of *any* policy computed so far (still at s_0). This optimization problem can be solved using a quadratic programming approach or a projection algorithm [1];

[1] Abbeel, Pieter, and Andrew Y. Ng. "Apprenticeship learning via inverse reinforcement learning." *Proceedings of the twenty-first international conference on Machine learning*. ACM, 2004.

IRL Algorithm – Projection method

3.1. A simpler algorithm

The algorithm described above requires access to a QP (or SVM) solver. It is also possible to change the algorithm so that no QP solver is needed. We will call the previous, QP-based, algorithm the **max-margin** method, and the new algorithm the **projection** method. Briefly, the projection method replaces step 2 of the algorithm with the following:

- Set $\bar{\mu}^{(i-1)} = \bar{\mu}^{(i-2)} + \frac{(\mu^{(i-1)} - \bar{\mu}^{(i-2)})^T (\mu_E - \bar{\mu}^{(i-2)})}{(\mu^{(i-1)} - \bar{\mu}^{(i-2)})^T (\mu^{(i-1)} - \bar{\mu}^{(i-2)})} (\mu^{(i-1)} - \bar{\mu}^{(i-2)})$
  (This computes the orthogonal projection of μ_E onto the line through $\bar{\mu}^{(i-2)}$ and $\mu^{(i-1)}$.)
- Set $w^{(i)} = \mu_E - \bar{\mu}^{(i-1)}$
- Set $t^{(i)} = \|\mu_E - \bar{\mu}^{(i-1)}\|_2$

[1] Abbeel, Pieter, and Andrew Y. Ng. "Apprenticeship learning via inverse reinforcement learning." *Proceedings of the twenty-first international conference on Machine learning*. ACM, 2004.

IRL Algorithm

initial policy -> $\mu(s)$ -> $R = \theta^T \phi(s)$ -> policy(LSPI) -> $\mu(s)$ -> $R = \theta^T \phi(s) =$ -> policy(LSPI) …

3. if $t^{(j)} \leq \epsilon$, terminate. The algorithm outputs a set of policies $\{\pi^{(0)}, \ldots, \pi^{(j-1)}\}$ among which the user chooses manually or automatically the closest to the expert (see [1] for details on how to choose this policy). Notice that the last policy is not necessarily the best (as illustrated in Section 4);

LSPI(Least Square Policy Iteration)[2] : LSTD-Q + Policy Evaluation

4. solve the MDP with the reward function $R^{(j)}(s) = (\theta^{(j)})^T \phi(s)$ and denote $\pi^{(j)}$ the associated optimal policy. Compute $\mu^{\pi^{(j)}}(s_0)$;

state-action LSTD-μ

[2]Lagoudakis, Michail G., and Ronald Parr. "Least-squares policy iteration." *Journal of machine learning research* 4.Dec (2003): 1107-1149.

IRL Algorithm

1. Initialize $\mu^E(s_0)$ $\pi^{(0)}$ $\mu^{(0)}(s_0)$

2. Projection method

$\bar{\mu}^{(0)} = \mu^{(0)}(s_0)$ $\theta^{(1)} = \mu^E(s_0) - \bar{\mu}^{(0)}$ $t^{(1)} = \left\|\mu^E(s_0) - \bar{\mu}^{(0)}\right\|_2$

3. if $t^{(j)} \leq \epsilon$, terminate.

4. Solve MDP using LSPI, get $\mu^{\pi^{(1)}}(s_0)$ using LSTD-μ.

$R^{(1)}(s) = (\theta^{(1)})^T \phi(s) \xrightarrow[\text{LSPI}]{} \pi^{(1)} \xrightarrow[\text{LSTD-}\mu]{} \mu^{\pi^{(1)}}(s_0)$

---

2. Projection method

$\bar{\mu}^{(1)} =$ Set $\bar{\mu}^{(i-1)} = \bar{\mu}^{(i-2)} + \frac{(\mu^{(i-1)} - \bar{\mu}^{(i-2)})^T (\mu_E - \bar{\mu}^{(i-2)})}{(\mu^{(i-1)} - \bar{\mu}^{(i-2)})^T (\mu^{(i-1)} - \bar{\mu}^{(i-2)})} (\mu^{(i-1)} - \bar{\mu}^{(i-2)})$ $\theta^{(2)} = \mu^E(s_0) - \bar{\mu}^{(1)}$ $t^{(2)} = \left\|\mu^E(s_0) - \bar{\mu}^{(1)}\right\|_2$

3. if $t^{(j)} \leq \epsilon$, terminate.

4. Solve MDP using LSPI, get $\mu^{\pi^{(2)}}(s_0)$ using LSTD-mu.

$R^{(2)}(s) = (\theta^{(2)})^T \phi(s) \xrightarrow[\text{LSPI}]{} \pi^{(2)} \xrightarrow[\text{LSTD-}\mu]{} \mu^{\pi^{(2)}}(s_0)$

IRL Algorithm – LSPI(Least Square Policy Iteration)

LSTDQ $(D,\ k,\ \phi,\ \gamma,\ \pi)$ // Learns $\widehat{Q}^{\pi}$ from samples

// D : Source of samples (s, a, r, s')
// k : Number of basis functions
// ϕ : Basis functions
// γ : Discount factor
// π : Policy whose value function is sought

$\widetilde{\mathbf{A}} \leftarrow \mathbf{0}$ // $(k \times k)$ matrix
$\widetilde{b} \leftarrow \mathbf{0}$ // $(k \times 1)$ vector

for each $(s, a, r, s') \in D$ — random action "a"

$\widetilde{\mathbf{A}} \leftarrow \widetilde{\mathbf{A}} + \phi(s,a)\Big(\phi(s,a) - \gamma\phi\big(s', \pi(s')\big)\Big)^{\mathsf{T}}$

$\widetilde{b} \leftarrow \widetilde{b} + \phi(s,a)r$

$\widetilde{w}^{\pi} \leftarrow \widetilde{\mathbf{A}}^{-1}\widetilde{b}$

$$\widehat{Q}^{\pi} = \mathbf{\Phi} w^{\pi}$$

return $\widetilde{w}^{\pi}$

LSPI $(D,\ k,\ \phi,\ \gamma,\ \epsilon,\ \pi_0)$ // Learns a policy from samples

// D : Source of samples (s, a, r, s')
// k : Number of basis functions
// ϕ : Basis functions
// γ : Discount factor
// ϵ : Stopping criterion
// π_0 : Initial policy, given as w_0 (default: $w_0 = 0$)

$\pi' \leftarrow \pi_0$ // $w' \leftarrow w_0$

repeat
  $\pi \leftarrow \pi'$ // $w \leftarrow w'$
  $\pi' \leftarrow$ **LSTD**Q $(D,\ k,\ \phi,\ \gamma,\ \pi)$ // $w' \leftarrow$ **LSTD**Q $(D,\ k,\ \phi,\ \gamma,\ w)$
until $(\pi \approx \pi')$ // **until** $(\|w - w'\| < \epsilon)$

return π // **return** w

[2]Lagoudakis, Michail G., and Ronald Parr. "Least-squares policy iteration." *Journal of machine learning research* 4.Dec (2003): 1107-1149.

IRL Algorithm – LSPI(Least Square Policy Iteration)

Approximate Q(prediction)

$$\widehat{Q}^{\pi} = \mathbf{\Phi} w^{\pi} \qquad \mathbf{\Phi} = \begin{pmatrix} \phi(s_1, a_1)^{\intercal} \\ \cdots \\ \phi(s, a)^{\intercal} \\ \cdots \\ \phi(s_{|\mathcal{S}|}, a_{|\mathcal{A}|})^{\intercal} \end{pmatrix}$$

Find w^{π}

$$w^{\pi} = \left(\mathbf{\Phi}^{\intercal} \Delta_{\mu} (\mathbf{\Phi} - \gamma \mathbf{P} \mathbf{\Pi}_{\pi} \mathbf{\Phi})\right)^{-1} \mathbf{\Phi}^{\intercal} \Delta_{\mu} \mathcal{R}$$

$$w^{\pi} = A^{-1} b$$

$$\mathbf{A} = \mathbf{\Phi}^{\intercal} \Delta_{\mu} (\mathbf{\Phi} - \gamma \mathbf{P} \mathbf{\Pi}_{\pi} \mathbf{\Phi}) \qquad \text{and} \qquad b = \mathbf{\Phi}^{\intercal} \Delta_{\mu} \mathcal{R}$$

[2]Lagoudakis, Michail G., and Ronald Parr. "Least-squares policy iteration." *Journal of machine learning research* 4.Dec (2003): 1107-1149.

IRL Algorithm – LSPI(Least Square Policy Iteration)

$$
\begin{aligned}
\mathbf{A} &= \mathbf{\Phi}^{\intercal}\Delta_{\mu}(\mathbf{\Phi} - \gamma \mathbf{P}\mathbf{\Pi}_{\pi}\mathbf{\Phi}) \\
&= \sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}} \phi(s,a)\mu(s,a)\Big(\phi(s,a) - \gamma \sum_{s'\in\mathcal{S}} \mathcal{P}(s,a,s')\phi\big(s',\pi(s')\big)\Big)^{\intercal} \\
&= \sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}} \mu(s,a) \sum_{s'\in\mathcal{S}} \mathcal{P}(s,a,s')\Big[\phi(s,a)\Big(\phi(s,a) - \gamma\phi\big(s',\pi(s')\big)\Big)^{\intercal}\Big]
\end{aligned}
$$

$$
\begin{aligned}
b &= \mathbf{\Phi}^{\intercal}\Delta_{\mu}\mathcal{R} \\
&= \sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}} \phi(s,a)\mu(s,a) \sum_{s'\in\mathcal{S}} \mathcal{P}(s,a,s')R(s,a,s') \\
&= \sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}} \mu(s,a) \sum_{s'\in\mathcal{S}} \mathcal{P}(s,a,s')\Big[\phi(s,a)R(s,a,s')\Big] \quad .
\end{aligned}
$$

$$
D = \Big\{ \; (s_i, a_i, r_i, s_i') \;\Big|\; i = 1, 2, \ldots, L \; \Big\}
$$

$$
\widetilde{\mathbf{A}} = \frac{1}{L}\sum_{i=1}^{L}\Big[\phi(s_i,a_i)\Big(\phi(s_i,a_i) - \gamma\phi\big(s_i',\pi(s_i')\big)\Big)^{\intercal}\Big]
\qquad
\lim_{L\to\infty} \widetilde{\mathbf{A}} = \mathbf{\Phi}^{\intercal}\Delta_{\mu_D}(\mathbf{\Phi} - \gamma\mathbf{P}\mathbf{\Pi}_{\pi}\mathbf{\Phi})
\qquad \text{and} \qquad
\lim_{L\to\infty} \widetilde{b} = \mathbf{\Phi}^{\intercal}\Delta_{\mu_D}\mathcal{R}
$$

$$
\widetilde{b} = \frac{1}{L}\sum_{i=1}^{L}\Big[\phi(s_i,a_i)r_i\Big] \quad ,
$$

state-action LSTD-μ

Compute $\mu^{\pi^{(j)}}(s_0)$ LSTD-μ : to estimate feature expectation of intermediate policies

$$\mu_i^\pi(s) = E[\sum_{t=0}^{\infty} \gamma^t \phi_i(s_t) | s_0 = s, \pi].$$

$$\mu^\pi(s) = \xi^T \psi(s) \qquad \xi \in \mathbb{R}^q$$

$$\xi_i^* = \left(\sum_{t=1}^{n} \psi(s_t, a_t) \left(\psi(s_t, a_t) - \gamma \psi(s_t', \pi(s_t')) \right)^T \right)^{-1} \sum_{t=1}^{n} \psi(s_t, a_t) \phi(s_t, a_t)$$

$$V^\pi(s) = E[\sum_{t=0}^{\infty} \gamma^t R(s_t) | s_0 = s, \pi]$$

$$R = \theta^T \phi(s)$$

$$w^\pi = \left(\mathbf{\Phi}^\intercal \Delta_\mu (\mathbf{\Phi} - \gamma \mathbf{P} \mathbf{\Pi}_\pi \mathbf{\Phi}) \right)^{-1} \mathbf{\Phi}^\intercal \Delta_\mu \mathcal{R}$$

$$w_i^* = \left(\sum_{t=1}^{n} \phi(s_t, a_t) \left(\phi(s_t, a_t) - \gamma \phi(s_t', \pi(s_t')) \right)^T \right)^{-1} \sum_{t=1}^{n} \phi(s_t, a_t) r(s_t, a_t)$$

a set of transitions $\{(s_t, r_t, s_{t+1})_{1 \le t \le n}\}$ sampled according to the policy π

$$(\hat{\mu}^\pi(s_0))^T = \psi(s_0)^T (\Psi^T \Delta \Psi)^{-1} \Psi^T \Phi$$

$$\hat{\psi}(s_0) = \frac{\sum_{i=1, a \sim random}^{N} \psi(s_0, a)}{N}$$

IRL Algorithm – LSPI(Least Square Policy Iteration)

LSTDQ $(D,\ k,\ \phi,\ \gamma,\ \pi)$ // Learns $\widehat{Q}^{\pi}$ from samples

// D : Source of samples (s, a, r, s')
// k : Number of basis functions
// ϕ : Basis functions
// γ : Discount factor
// π : Policy whose value function is sought

$\widetilde{\mathbf{A}} \leftarrow \mathbf{0}$ // $(k \times k)$ matrix
$\widetilde{b} \leftarrow \mathbf{0}$ // $(k \times 1)$ vector

random action "a"

for each $(s, a, r, s') \in D$
$\widetilde{\mathbf{A}} \leftarrow \widetilde{\mathbf{A}} + \phi(s,a)\Big(\phi(s,a) - \gamma\phi\big(s', \pi(s')\big)\Big)^{\mathsf{T}}$
$\widetilde{b} \leftarrow \widetilde{b} + \phi(s,a)r$

$\widetilde{w}^{\pi} \leftarrow \widetilde{\mathbf{A}}^{-1}\widetilde{b}$

return $\widetilde{w}^{\pi}$

$$\widehat{Q}^{\pi} = \mathbf{\Phi} w^{\pi}$$

LSPI $(D,\ k,\ \phi,\ \gamma,\ \epsilon,\ \pi_0)$ // Learns a policy from samples

// D : Source of samples (s, a, r, s')
// k : Number of basis functions
// ϕ : Basis functions
// γ : Discount factor
// ϵ : Stopping criterion
// π_0 : Initial policy, given as w_0 (default: $w_0 = 0$)

$\pi' \leftarrow \pi_0$ // $w' \leftarrow w_0$

repeat
$\pi \leftarrow \pi'$ // $w \leftarrow w'$
$\pi' \leftarrow$ **LSTD**Q $(D,\ k,\ \phi,\ \gamma,\ \pi)$ // $w' \leftarrow$ **LSTD**Q $(D,\ k,\ \phi,\ \gamma,\ w)$
until $(\pi \approx \pi')$ // **until** $(\|w - w'\| < \epsilon)$

return π // **return** w

[2]Lagoudakis, Michail G., and Ronald Parr. "Least-squares policy iteration." *Journal of machine learning research* 4.Dec (2003): 1107-1149.

LSTD vs. LSTD-Q

LSTD for estimating $\hat{v} \approx v_\pi$ ($O(d^2)$ version)

Input: feature representation $\mathbf{x}(s) \in \mathbb{R}^d$, for all $s \in \mathcal{S}, \mathbf{x}(\text{terminal}) \doteq \mathbf{0}$

$\widehat{\mathbf{A}^{-1}} \leftarrow \varepsilon^{-1}\mathbf{I}$ — An $d \times d$ matrix

$\widehat{\mathbf{b}} \leftarrow \mathbf{0}$ — An d-dimensional vector

Repeat (for each episode):

  Initialize S; obtain corresponding $\mathbf{x}$

  Repeat (for each step of episode):

    Choose $A \sim \pi(\cdot|S)$ — behavior from target policy

    Take action A, observe R, S'; obtain corresponding $\mathbf{x}'$

    $\mathbf{v} \leftarrow \widehat{\mathbf{A}^{-1}}^\top(\mathbf{x} - \gamma\mathbf{x}')$

    $\widehat{\mathbf{A}^{-1}} \leftarrow \widehat{\mathbf{A}^{-1}} - (\widehat{\mathbf{A}^{-1}}\mathbf{x})\mathbf{v}^\top/(1 + \mathbf{v}^\top\mathbf{x})$ — on-policy

    $\widehat{\mathbf{b}} \leftarrow \widehat{\mathbf{b}} + R\mathbf{x}$

    $\theta \leftarrow \widehat{\mathbf{A}^{-1}}\widehat{\mathbf{b}}$

    $S \leftarrow S'$; $\mathbf{x} \leftarrow \mathbf{x}'$ — target = behavior

  until S' is terminal

LSTDQ (D, k, ϕ, γ, π) // Learns $\widehat{Q}^\pi$ from samples

// D : Source of samples (s, a, r, s')
// k : Number of basis functions
// ϕ : Basis functions
// γ : Discount factor
// π : Policy whose value function is sought

$\widetilde{\mathbf{A}} \leftarrow \mathbf{0}$ // $(k \times k)$ matrix

$\widetilde{b} \leftarrow \mathbf{0}$ // $(k \times 1)$ vector

for each $(s, a, r, s') \in D$ — random generation

  $\widetilde{\mathbf{A}} \leftarrow \widetilde{\mathbf{A}} + \phi(s, a)\big(\phi(s, a) - \gamma\phi(s', \pi(s'))\big)^\intercal$ — off-policy

  $\widetilde{b} \leftarrow \widetilde{b} + \phi(s, a)r$

$\widetilde{w}^\pi \leftarrow \widetilde{\mathbf{A}}^{-1}\widetilde{b}$

$\widehat{Q}^\pi = \mathbf{\Phi} w^\pi$

return $\widetilde{w}^\pi$

LSTD	**LSTDQ**
Learns the **state** value function V^π	Learns the **state-action** value function Q^π
Basis functions of **state**	Basis functions of **state and action**
Samples **cannot** be reused on-policy	Samples **can** be reused off-policy
Training samples collected **by** π on-policy	Training samples collected **arbitrarily**
Sampling distribution **cannot** be controlled	Sampling distribution **can** be controlled
Bias by the **stationary distribution of** π	Bias by the **sampling distribution**

Backup slide : on-policy vs. off-policy

When the model of environment (such as transition probability) is unknown, the alternative way is depending on “sample experience”.
For episodic task, experience = “multiple episodes”

Obviously, a policy can make episodes(sample experience).

On-policy :
policy being evaluated and improved = policy generating data
(therefore, need ε-greedy for exploration)

Off-policy :
policy being evaluated and improved(target policy) $\neq$ policy generating data(behavior policy)

Why LSTD-μ is off-policy method?

$$
\begin{aligned}
\mathbf{A} &= \mathbf{\Phi}^\intercal \Delta_\mu (\mathbf{\Phi} - \gamma \mathbf{P} \mathbf{\Pi}_\pi \mathbf{\Phi}) \\
&= \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} \phi(s,a) \mu(s,a) \Big(\phi(s,a) - \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s,a,s') \phi(s', \pi(s')) \Big)^\intercal \\
&= \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} \mu(s,a) \sum_{s' \in \mathcal{S}} \mathcal{P}(s,a,s') \Big[\phi(s,a) \Big(\phi(s,a) - \gamma \phi\big(s', \pi(s')\big) \Big)^\intercal \Big]
\end{aligned}
$$

$$
\begin{aligned}
b &= \mathbf{\Phi}^\intercal \Delta_\mu \mathcal{R} \\
&= \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} \phi(s,a) \mu(s,a) \sum_{s' \in \mathcal{S}} \mathcal{P}(s,a,s') R(s,a,s') \\
&= \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} \mu(s,a) \sum_{s' \in \mathcal{S}} \mathcal{P}(s,a,s') \Big[\phi(s,a) R(s,a,s') \Big] \quad .
\end{aligned}
$$

for each $(s, a, r, s') \in D$

$$
\begin{aligned}
\widetilde{\mathbf{A}} &\leftarrow \widetilde{\mathbf{A}} + \phi(s,a) \Big(\phi(s,a) - \gamma \phi\big(s', \pi(s')\big) \Big)^\intercal \\
\widetilde{b} &\leftarrow \widetilde{b} + \phi(s,a) r
\end{aligned}
$$

TD(0) or one-step TD update rule

$$
V(S_t) \leftarrow V(S_t) + \alpha \Big[R_{t+1} + \gamma V(S_{t+1}) - V(S_t) \Big]
$$

Additional degree of freedom($a_0 = a$) allows off-policy learning.(LSTD-Q)

LSTD-Q (Q-function) → LSTD-μ (state-action feature expectation)

TD 의 경우 behavior policy가 필요하다. 그래야 a를 해서 s'를 얻을 수 있으니까
그래서 평가되고 향상되는 policy와 sampling (behavior) policy 가 같으면 on-policy
서로 다르거나, 랜덤하게 제시해주는 경우 off-policy 이다!

Implementation & Experiment

Experiment Setting

CartPole

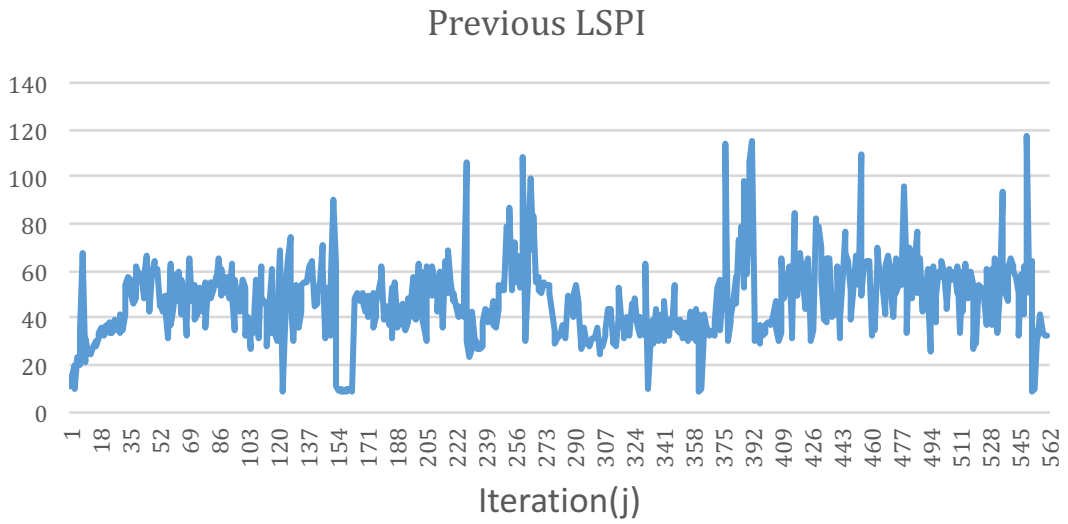
state dimension : 4
# of actions : 2

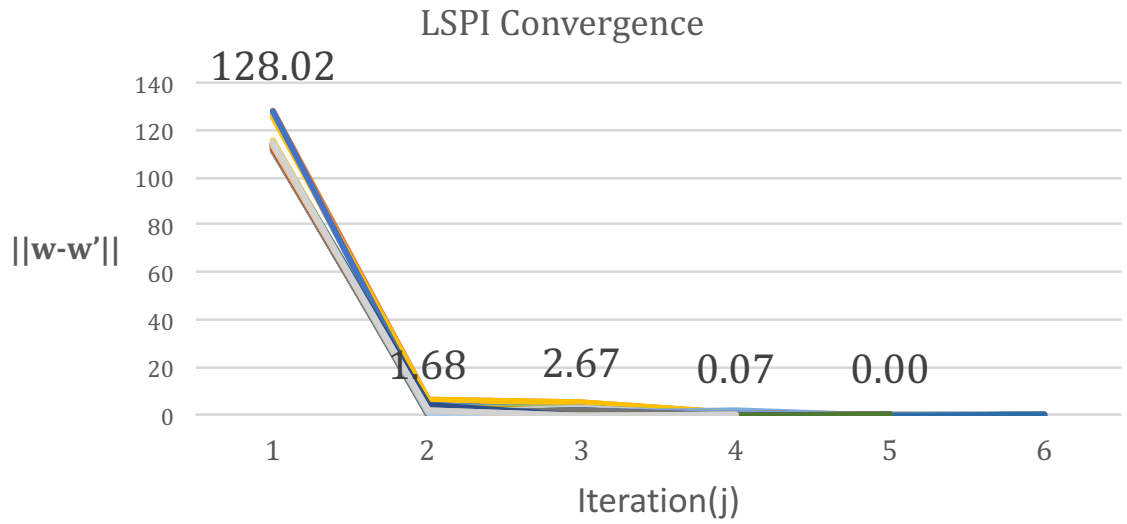
My LSPI's Critical Problem

LSPI $(D,\ k,\ \phi,\ \gamma,\ \epsilon,\ \pi_0)$ // Learns a policy from samples

// D : Source of samples (s, a, r, s')
// k : Number of basis functions
// ϕ : Basis functions
// γ : Discount factor
// ϵ : Stopping criterion
// π_0 : Initial policy, given as w_0 (default: $w_0 = 0$)

$\pi' \leftarrow \pi_0$	// $w' \leftarrow w_0$
repeat	
$\pi \leftarrow \pi'$	// $w \leftarrow w'$
$\pi' \leftarrow$ **LSTDQ** $(D,\ k,\ \phi,\ \gamma,\ \pi)$	// $w' \leftarrow$ **LSTDQ** $(D,\ k,\ \phi,\ \gamma,\ w)$
until $(\pi \approx \pi')$	// **until** $(\|w - w'\| < \epsilon)$
return π	// **return** w





loop_budget : 20

My LSPI's stabilization

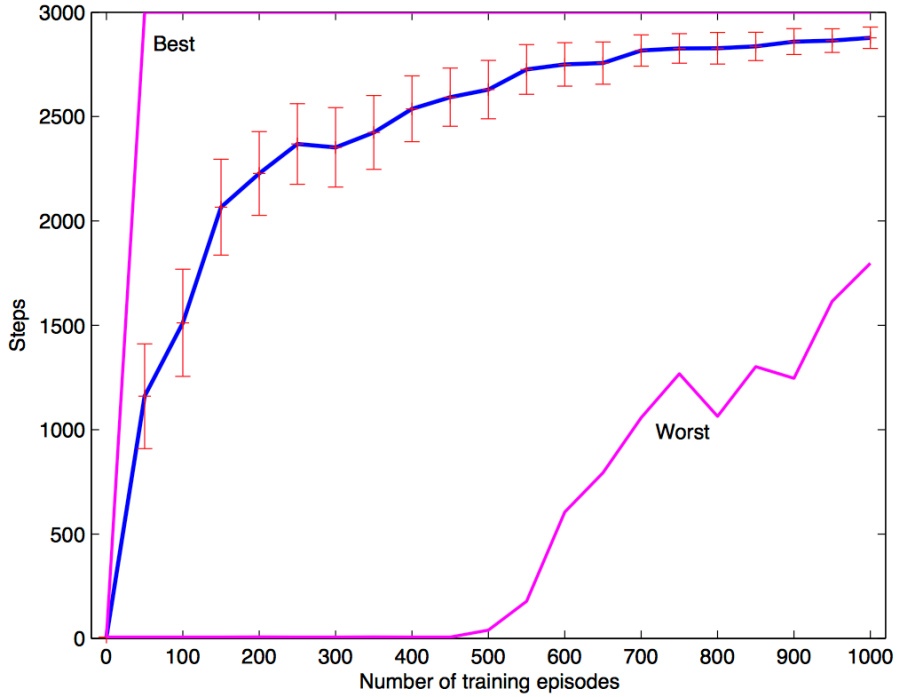
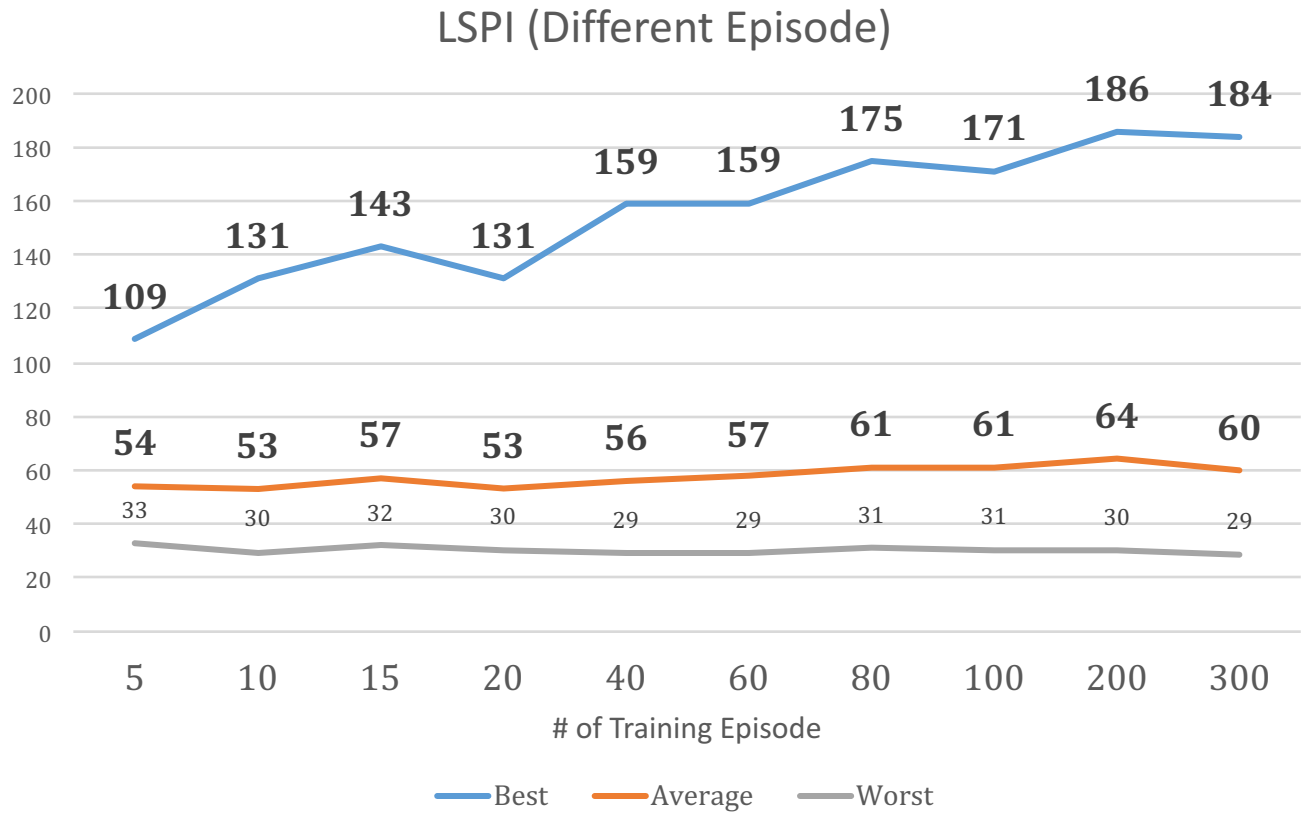
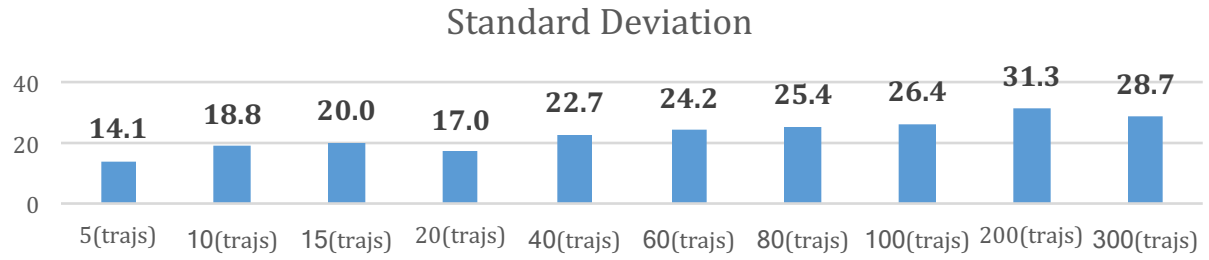


Figure 16: Inverted pendulum (LSPI): Average balancing steps.



Experiment Setting
Evaluation for each policy(w^{π}) : 1000
Trials(Experiments) : 100
basis function : RBF



[2]Lagoudakis, Michail G., and Ronald Parr. "Least-squares policy iteration." *Journal of machine learning research* 4.Dec (2003): 1107-1149.

Backup slide : Radial Basis Functions

RBFs are the natural generalization of coarse coding(binary features {0, 1}) to continuous-valued features([0, 1]).

A typical RBF feature has a Gaussian response dependent on the distance between the state, s, and the feature's center state, c_i, and the feature's width, σ_i.

$$x_i(s) \doteq \exp\left(-\frac{||s - c_i||^2}{2\sigma_i^2}\right)$$

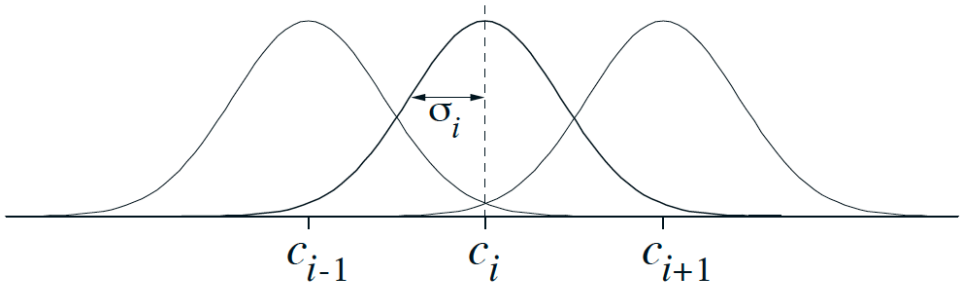
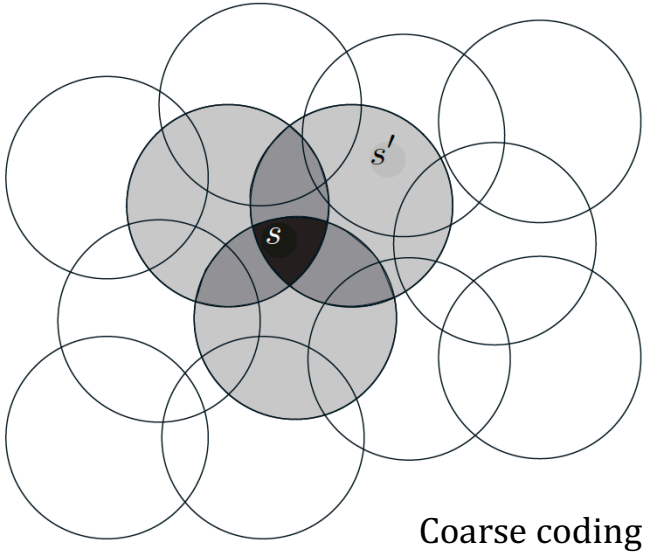


Figure 9.13: One-dimensional radial basis functions.



Coarse coding

Advantage : vary smoothly, and differentiable.

Basis function (RBF; gaussian)

N : dimension of states

A : # of actions

P : # of basis features

$s = (s_1, s_2, ..., s_N)^T$

$\mu_j = [\mu_{j,1}, \mu_{j,2}, ..., \mu_{j,N}]^T$, $j \in \{1, 2, ..., P\}$

$\mu_{j,i}$ is a random float number in (-1, 1)

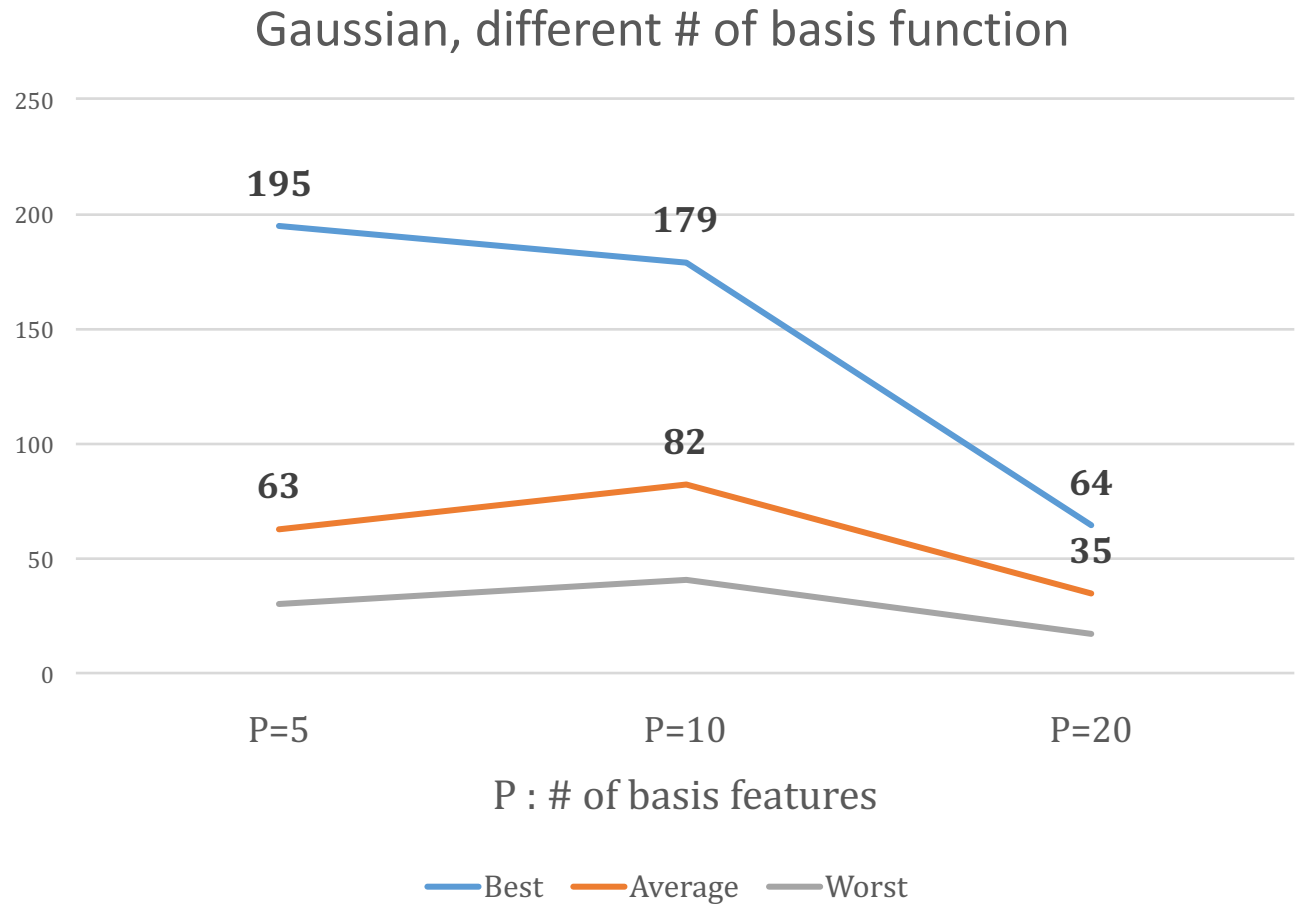
Radial Basis Functions;

given discrete true action $\hat{a}$,

$\phi_j(s) = exp(-\|s - \mu_j\|^2) = exp(-\frac{\|s-\mu_j\|^2}{2\sigma^2})$ $(\sigma = 1, \text{skip } \frac{1}{2})$

$\phi(s) = [1, \phi_1(s), \phi_2(s), ..., \phi_P(s)] \in R^P$

$\Phi(s; a = \hat{a}) = [\underbrace{[0, 0, ..., 0]}_{a \neq 0}, ..., \underbrace{\phi(s)}_{a = \hat{a}}, ... \underbrace{[0, 0, ..., 0]}_{a \neq A-1}] \in R^{A \times P}$



Episode for training w^π(LSTD-Q) : 100 (2000 samples)

Evaluation for each policy(w^π) : 1000

Trials(Experiments) : 100

Basis Function Dimension : (A(=2) X P(=5,10,20)) ➔ 10,20,40

RBF

P : # of basis features

$t = \|w^{\pi_{j-1}} - w^{\pi_j}\|_2$

LSPI $(D,\ k,\ \phi,\ \gamma,\ \epsilon,\ \pi_0)$ // Learns a policy from samples

// D : Source of samples (s, a, r, s')
// k : Number of basis functions
// ϕ : Basis functions
// γ : Discount factor
// ϵ : Stopping criterion
// π_0 : Initial policy, given as w_0 (default: $w_0 = 0$)

$\pi' \leftarrow \pi_0$ // $w' \leftarrow w_0$

repeat
  $\pi \leftarrow \pi'$ // $w \leftarrow w'$
  $\pi' \leftarrow$ **LSTDQ** $(D,\ k,\ \phi,\ \gamma,\ \pi)$ // $w' \leftarrow$ **LSTDQ** $(D,\ k,\ \phi,\ \gamma,\ w)$
until $(\pi \approx \pi')$ // **until** $(\|w - w'\| < \epsilon)$

return π // **return** w

LSTDQ $(D,\ k,\ \phi,\ \gamma,\ \pi)$ // Learns $\widehat{Q}^{\pi}$ from samples

// D : Source of samples (s, a, r, s')
// k : Number of basis functions
// ϕ : Basis functions
// γ : Discount factor
// π : Policy whose value function is sought

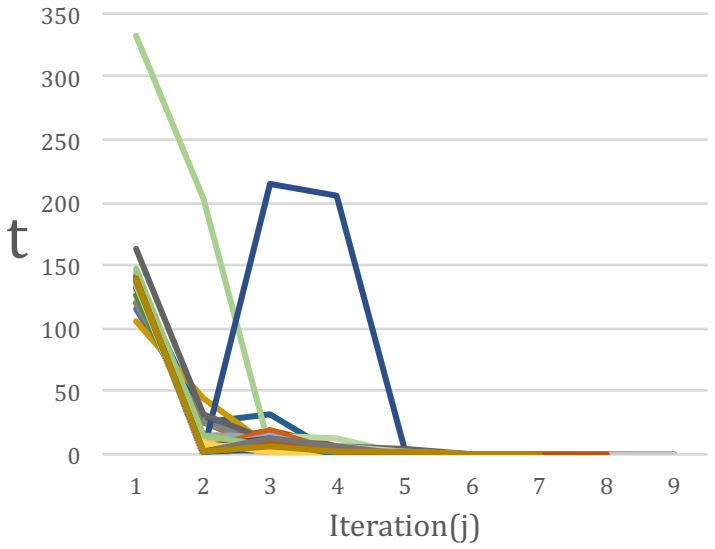
$\widetilde{\mathbf{A}} \leftarrow \mathbf{0}$ // $(k \times k)$ matrix
$\widetilde{b} \leftarrow \mathbf{0}$ // $(k \times 1)$ vector

for each $(s, a, r, s') \in D$
  $\widetilde{\mathbf{A}} \leftarrow \widetilde{\mathbf{A}} + \phi(s, a)\left(\phi(s, a) - \gamma\phi(s', \pi(s'))\right)^{\mathsf{T}}$
  $\widetilde{b} \leftarrow \widetilde{b} + \phi(s, a)r$

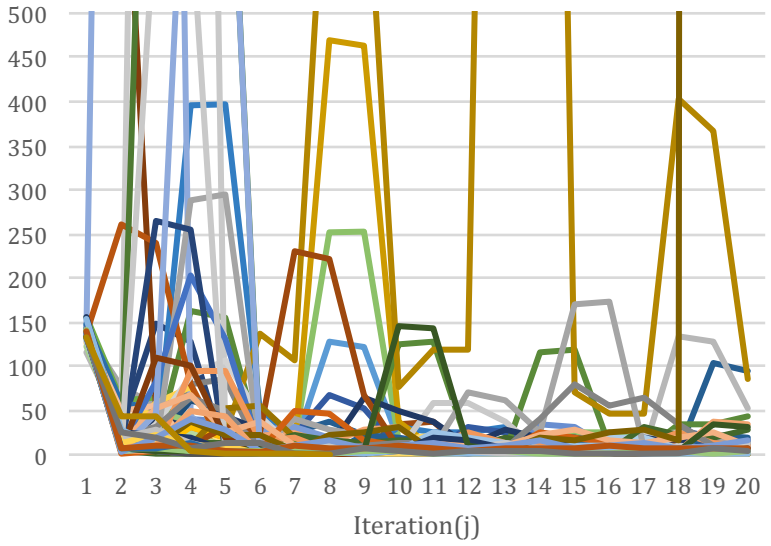
$\widetilde{w}^{\pi} \leftarrow \widetilde{\mathbf{A}}^{-1}\widetilde{b}$

return $\widetilde{w}^{\pi}$

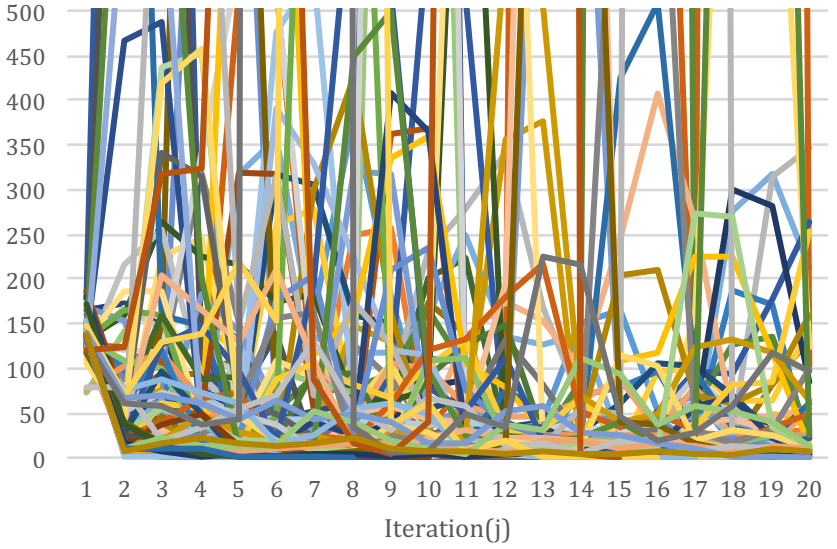
P=5 Convergence



P=10 Convergence



P=20 convergence



Basis function (Sigmoid)

N : dimension of states

A : # of actions

P : # of basis features

$s = (s_1, s_2, ..., s_N)^T$

$\mu_j = [\mu_{j,1}, \mu_{j,2}, ..., \mu_{j,N}]^T$, $j \in \{1, 2, ..., P\}$

$\mu_{j,i}$ is a random float number in (-1, 1)

sigmoid;

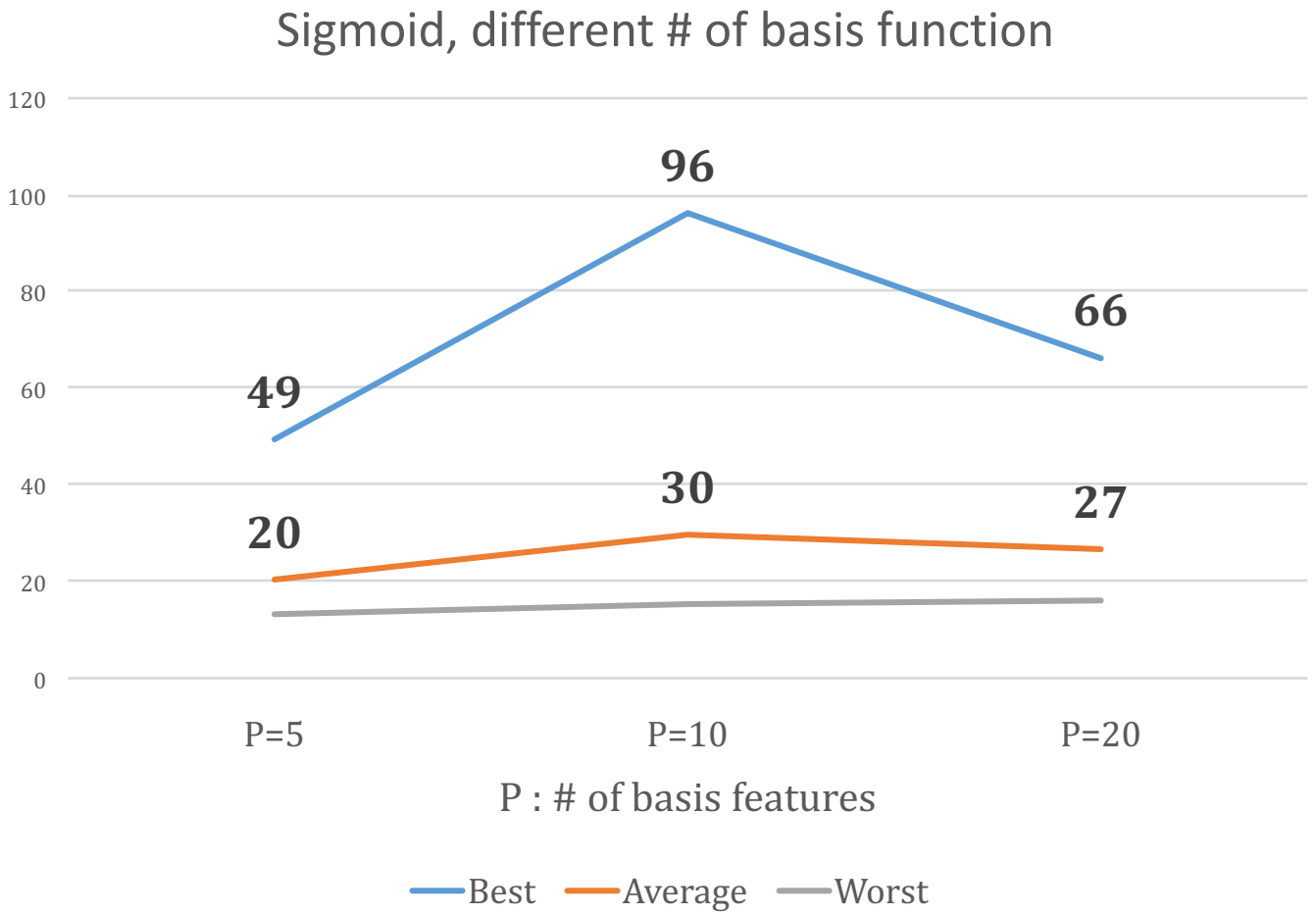
given discrete true action $\hat{a}$,

$\sigma(a) = \frac{1}{1+exp(-a)}$

$\phi_j(s) = \sigma(\|s - \mu_j\|_2)$

$\phi(s) = [1, \phi_1(s), \phi_2(s), ..., \phi_P(s)] \in R^P$

$\Phi(s; a = \hat{a}) = [\underbrace{[0, 0, ..., 0]}_{a \neq 0}, ..., \underbrace{\phi(s)}_{a = \hat{a}}, ... \underbrace{[0, 0, ..., 0]}_{a \neq A-1}] \in R^{A \times P}$



Episode for training w^{π}(LSTD-Q) : 100 (2000 samples)

Evaluation for each policy(w^{π}) : 1000

Trials(Experiments) : 100

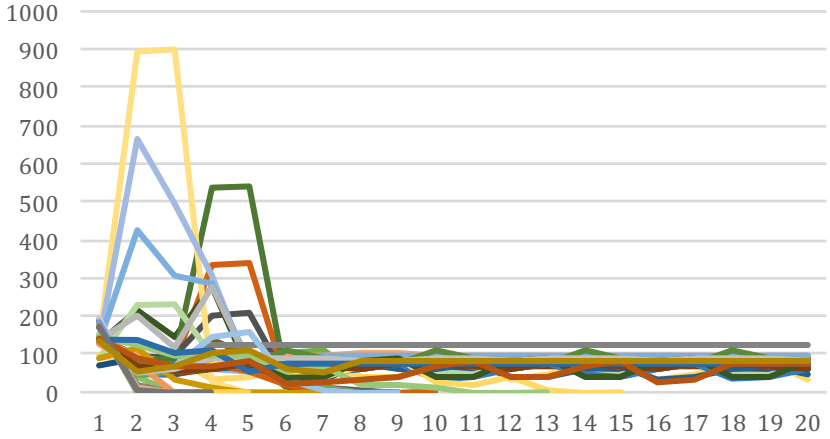
basis feature(P) = {5, 10, 20}

Basis Function Dimension : (A(=2) X P(=5,10,20)) ➔ 10,20,40

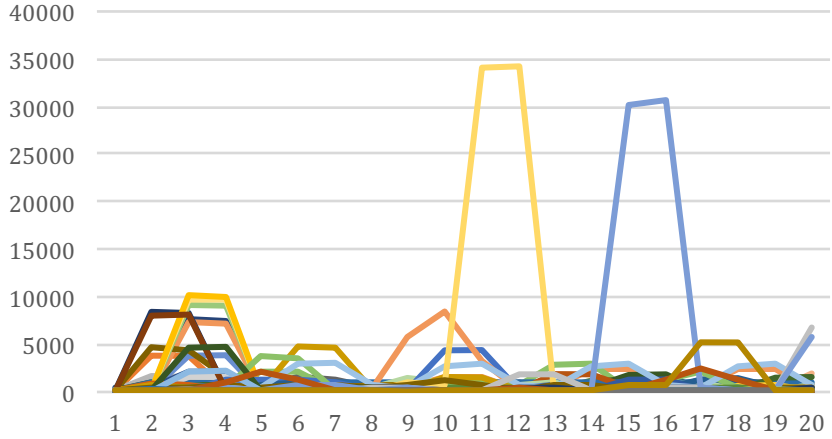
Basis function (Sigmoid)

$$t = \|w^{\pi_{j-1}} - w^{\pi_j}\|_2$$

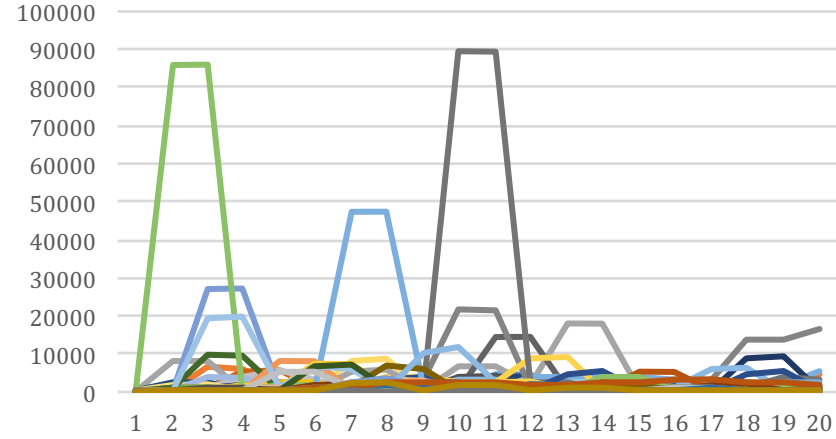
P=5



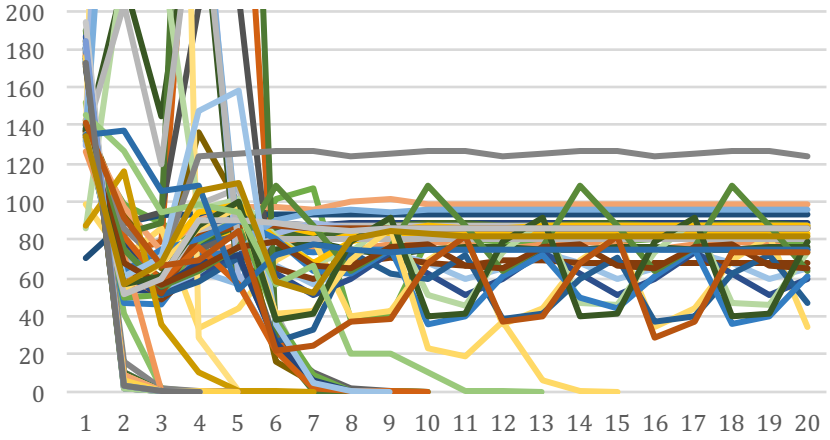
P=10



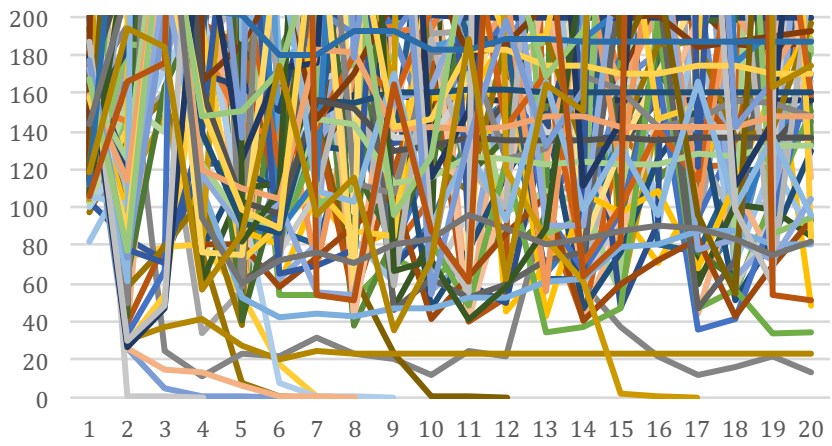
P=20



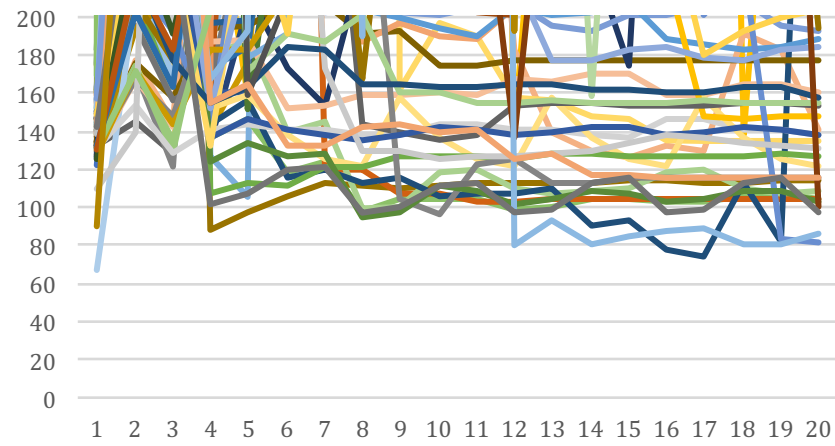
P=5 (0~200)



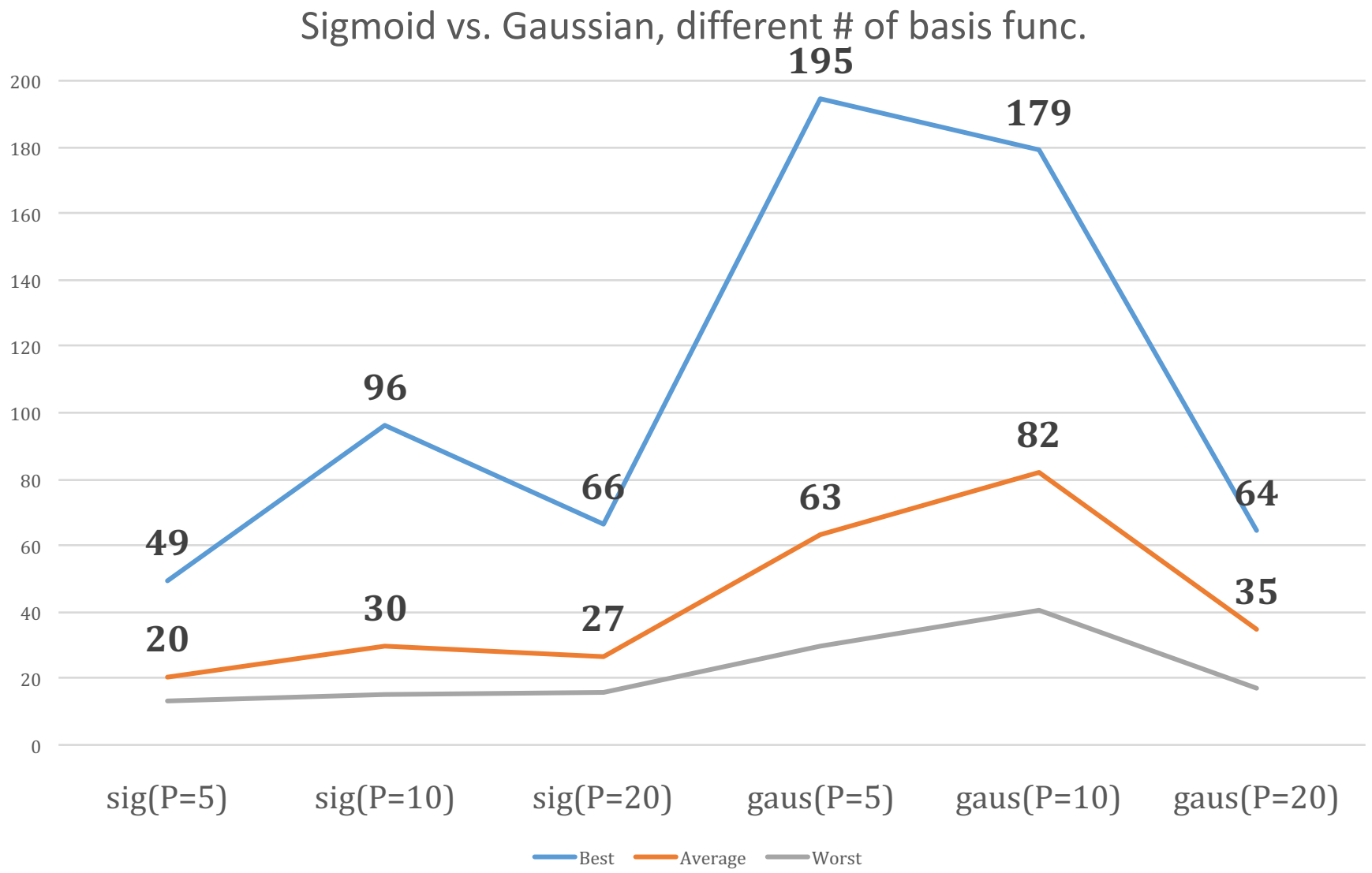
P=10 (0~200)



P=20 (0~200)



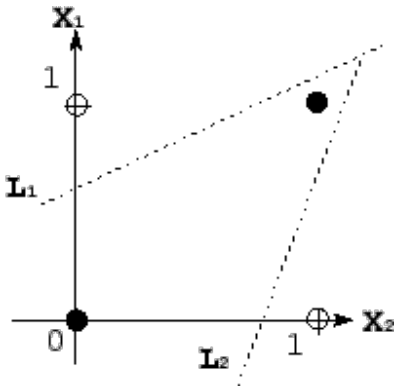
Basis function (Gaussian vs. Sigmoid)



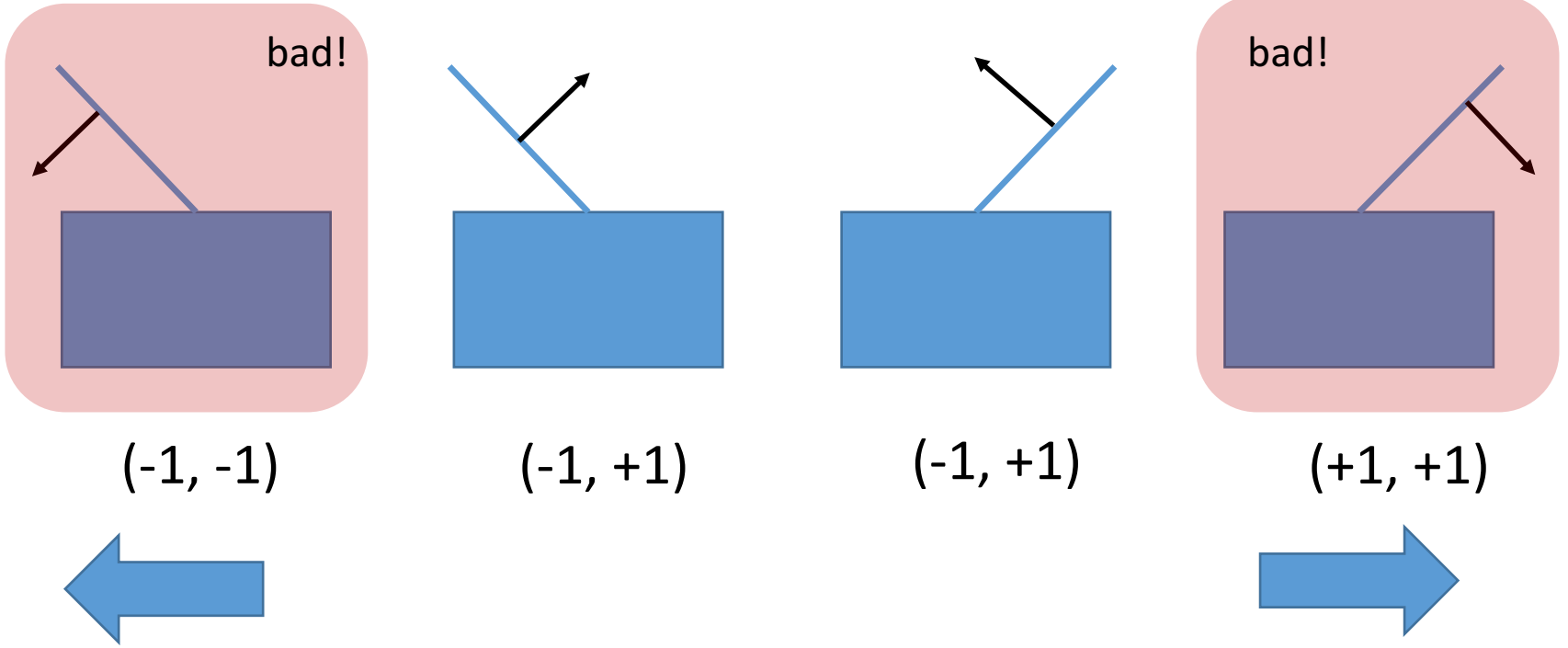
CartPole Problem Analysis (manual policy)

X_1	X_2	Y
0	0	0
0	1	1
1	0	1
1	1	0

$Y = X_1 \oplus X_2$



XOR



(-1, -1) (-1, +1) (-1, +1) (+1, +1)

Go left
label 1

Safe
label 0

Safe
label 0

Go right
label 2

We need ANN with activation function!

ANN for CartPole

ANN for CartPole

$$softmax\_cross\_entropy$$

$$loss = -\sum_{i}^{N} a_t log(\hat{a}_t)$$

Input : 4

W1 : 4 x 2

b1 : 2

Relu

h1 : 2

W2 : 2 x 3

b2 : 3

Softmax

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$

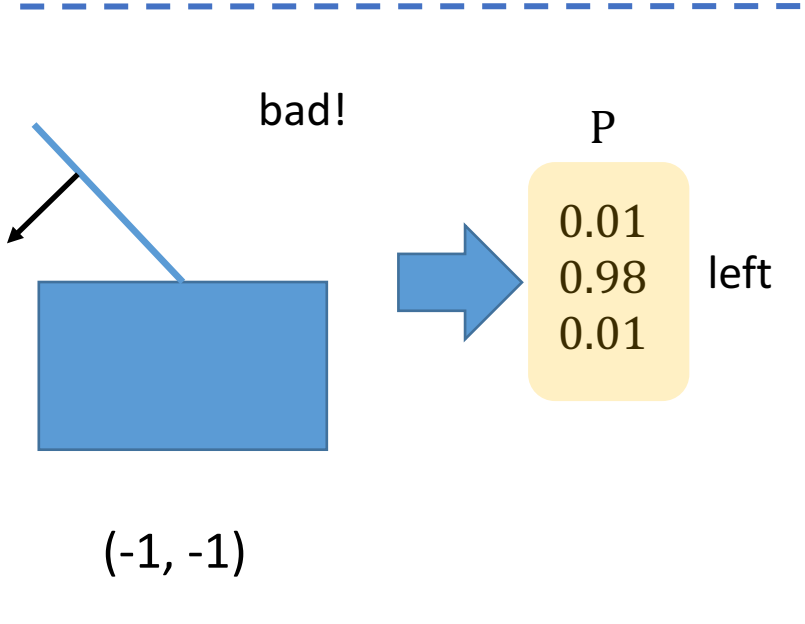
output : 3

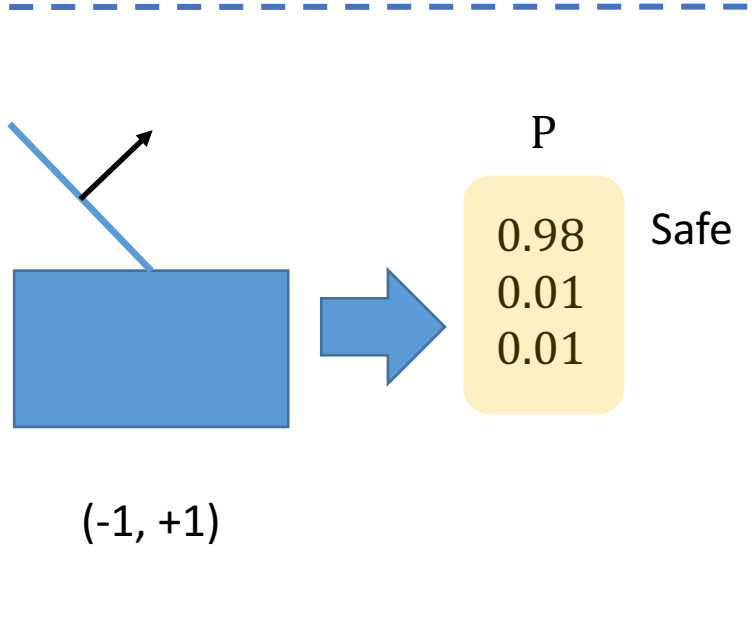
$$\begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix}$$

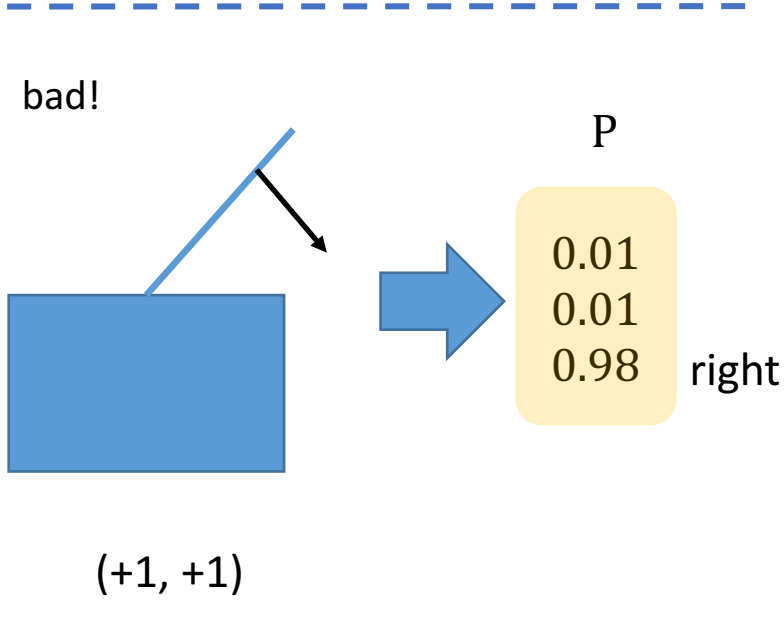
label 0(safe)

label 1(go left)

label 2(go right)



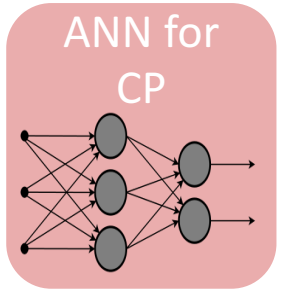




Basis function (ANN for CartPole + RBF)

ANN for CartPole + RBF

$$S = \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \end{bmatrix}$$



$$P = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix}$$

RBF

$$exp(-\frac{\|s-\mu_j\|^2}{2\sigma^2})$$

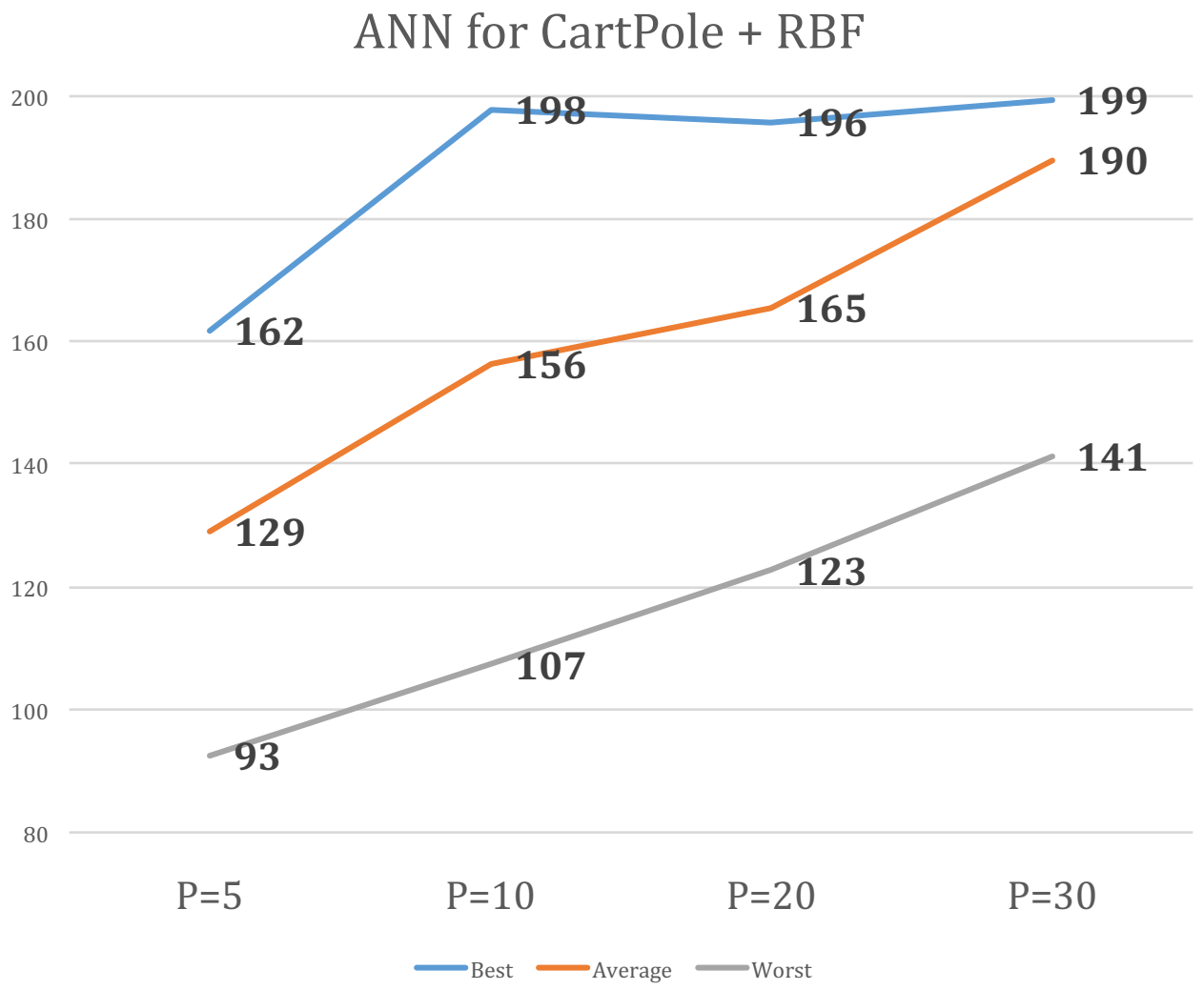
$$\phi_j(P)$$

Episode for training w^{π}(LSTD-Q) : 100 (2000 samples)

Evaluation for each policy(w^{π}) : 1000

Trials(Experiments) : 100

basis feature(P) = {5, 10, 20, 30}

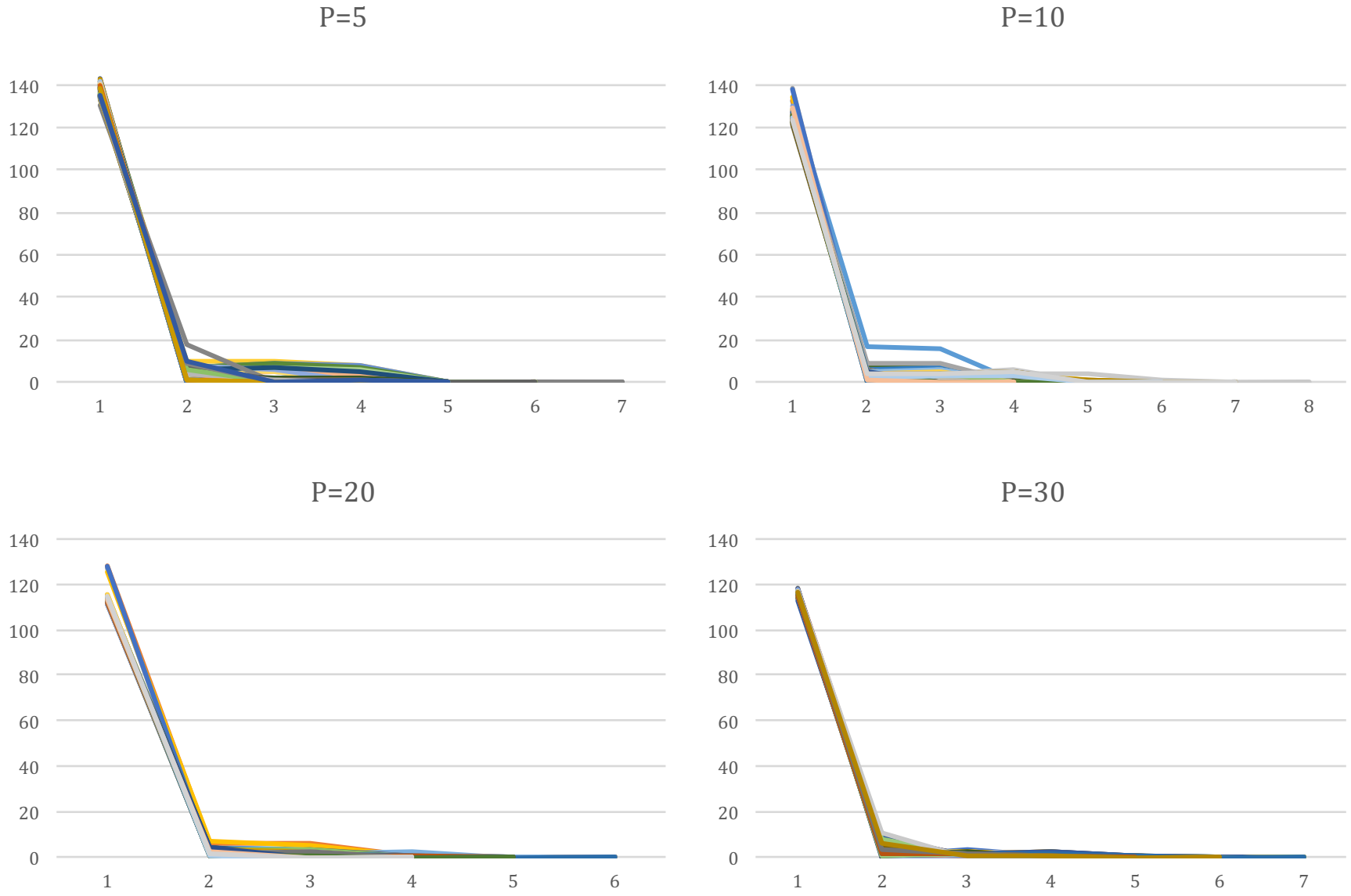


Basis function (ANN for CartPole + RBF)

P : # of basis features

Trials(Experiments) : 100

$t = \|w^{\pi_{j-1}} - w^{\pi_j}\|_2$



Experiment Setting

1. Initialize $\mu^E(s_0)$ $\pi^{(0)}$ $\mu^{(0)}(s_0)$

2. Projection method

$\bar{\mu}^{(0)} = \mu^{(0)}(s_0)$ $\theta^{(1)} = \mu^E(s_0) - \bar{\mu}^{(0)}$ $t^{(1)} = \left\|\mu^E(s_0) - \bar{\mu}^{(0)}\right\|_2$

$t^{(2)} = \left\|\mu^E(s_0) - \bar{\mu}^{(1)}\right\|_2$

4. Solve MDP using LSPI, get $\mu^{\pi^{(1)}}(s_0)$ using LSTD-μ.

$R^{(1)}(s) = (\theta^{(1)})^T \phi(s)$ → (LSPI) $\pi^{(1)}$ → (LSTD-μ) $\mu^{\pi^{(1)}}(s_0)$ **Baseline 1**

---

2. Projection method

$\bar{\mu}^{(1)} =$ Set $\bar{\mu}^{(i-1)} = \bar{\mu}^{(i-2)} + \frac{(\mu^{(i-1)} - \bar{\mu}^{(i-2)})^T (\mu_E - \bar{\mu}^{(i-2)})}{(\mu^{(i-1)} - \bar{\mu}^{(i-2)})^T (\mu^{(i-1)} - \bar{\mu}^{(i-2)})} (\mu^{(i-1)} - \bar{\mu}^{(i-2)})$

$t^{(2)} = \left\|\mu^E(s_0) - \bar{\mu}^{(1)}\right\|_2$

$\mu^{\pi^{(1)}}(s_0)$ **Baseline 2**

compute feature expectation by MC approach(like expert mu)

$\theta^{(2)} = \mu^E(s_0) - \bar{\mu}^{(1)}$ $t^{(2)} = \left\|\mu^E(s_0) - \bar{\mu}^{(1)}\right\|_2$

4. Solve MDP using LSPI, get $\mu^{\pi^{(2)}}(s_0)$ using LSTD-mu.

$R^{(2)}(s) = (\theta^{(2)})^T \phi(s)$ → (LSPI) $\pi^{(2)}$ → (LSTD-μ) $\mu^{\pi^{(2)}}(s_0)$

IRL + MC approach

of Expert Episode : 100 (100 x 200 = 20000 samples(s,a,s',r))

# of Episode for training w^{π}(LSTD-Q) : 100 (100 x 20 = 2000 samples)

# of Episode for calculating μ^{π} : 100 (100 x ? samples)

# of Evaluation of each policy(w^{π}) : 100

$$R(s) = \theta^{\intercal}\phi_{reward}(s)$$

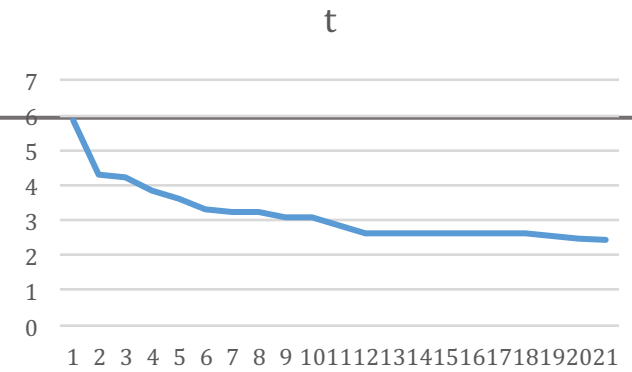
Reward basis function : ANN cartpole + RBF

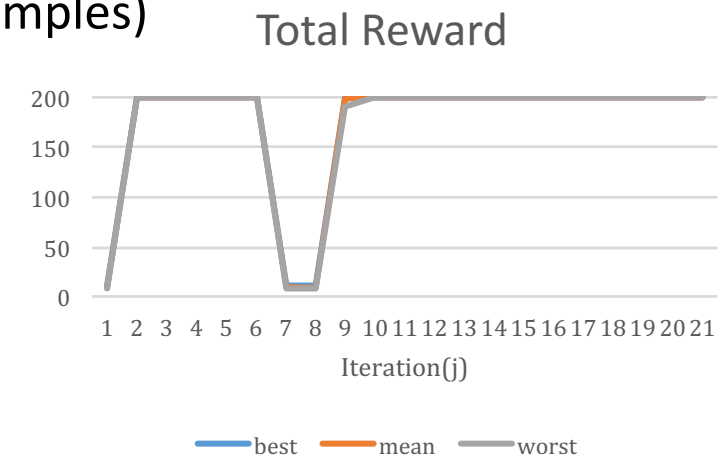
Reward basis feature dim(ϕ_{reward}) : 5

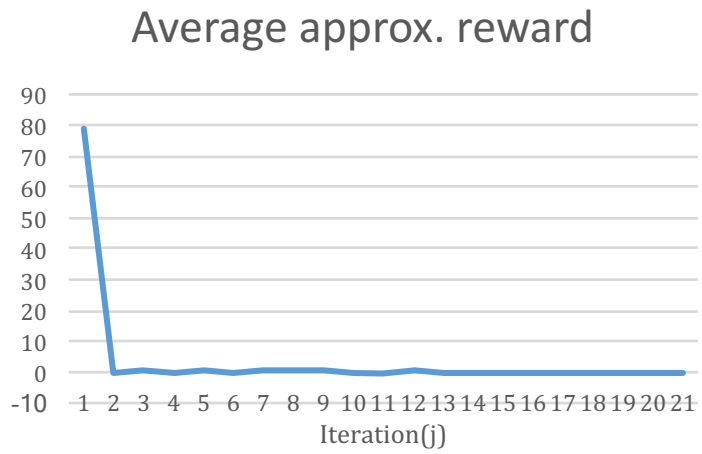
$$\hat{Q}^{\pi} = \Phi_{LSTD} w^{\pi}$$

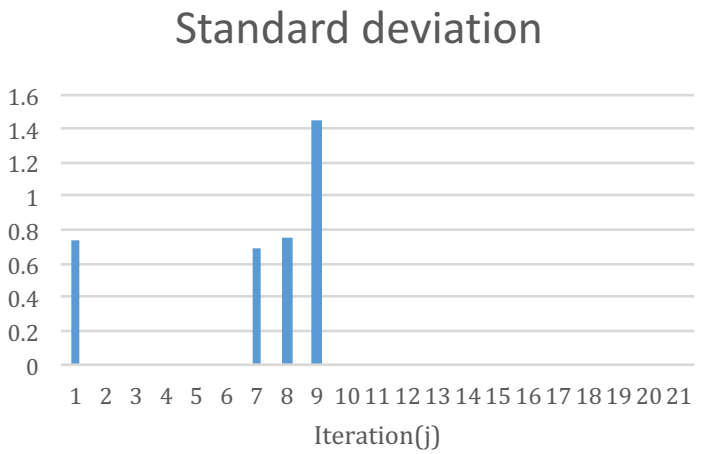
LSTD-Q basis function : ANN cartpole + RBF

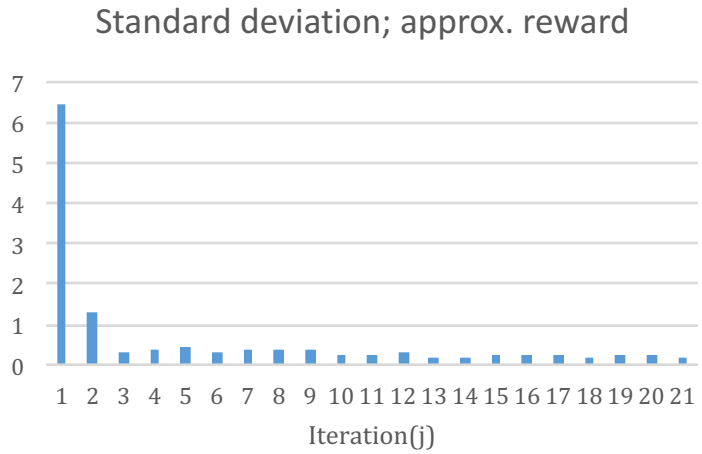
LSTD-Q basis dimension(ϕ_{LSTD}) : 5











IRL + MC approach

of Expert Episode : **1 (= 200 samples (s,a,s',r))**

# of Episode for training w^{π}(LSTD-Q) : 100

# of Episode for calculating μ^{π} : 100

# of Evaluation of each policy(w^{π}) : 100

$$R(s) = \theta^{\mathsf{T}} \phi_{reward}(s)$$

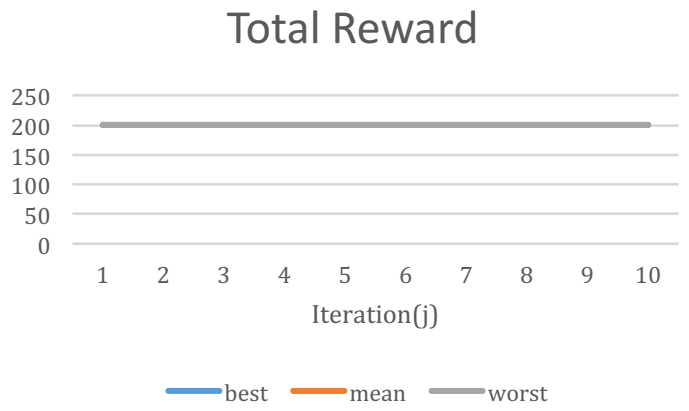
Reward basis function : ANN cartpole + RBF

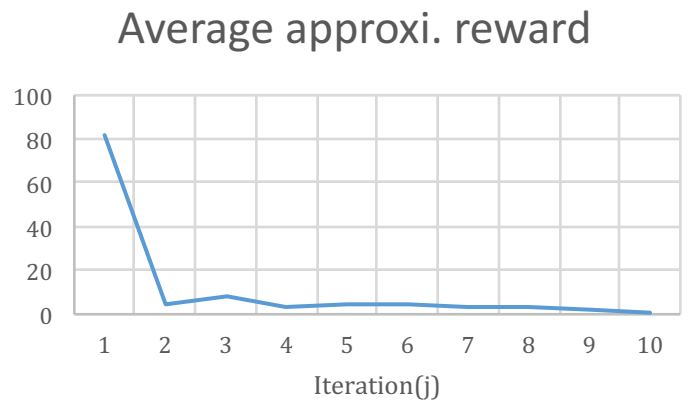
Reward basis feature dim(ϕ_{reward}) : 5

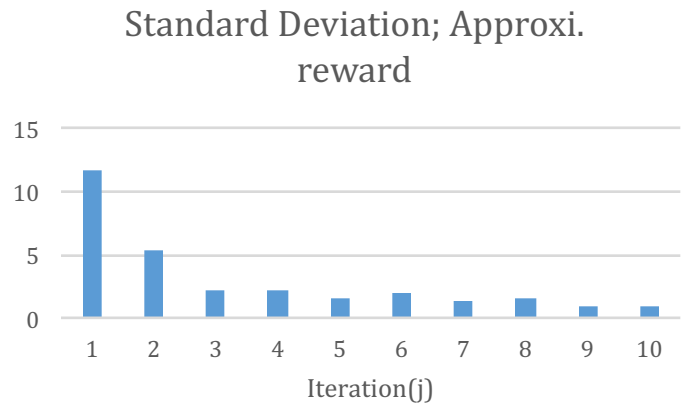
$$\hat{Q}^{\pi} = \Phi_{LSTD} w^{\pi}$$

LSTD-Q basis function : ANN cartpole + RBF

LSTD-Q basis dimension(ϕ_{LSTD}) : 5







→ 200 expert trajectories are enough!

IRL + MC approach

$$\textbf{for each } (s,a,r,s') \in D$$
$$\tilde{\mathbf{A}} \leftarrow \tilde{\mathbf{A}} + \phi(s,a)\left(\phi(s,a) - \gamma\phi(s',\pi(s'))\right)^{\intercal}$$
$$\tilde{b} \leftarrow \tilde{b} + \phi(s,a)r$$

# of Expert Episode : 100

# of Episode for training w^{π}(LSTD-Q) : 10 (10 x 20 = 200 samples, |D|=200)

# of Episode for calculating μ^{π} : 100

# of Evaluation of each policy(w^{π}) : 100

$$R(s) = \theta^{\intercal}\phi_{reward}(s)$$

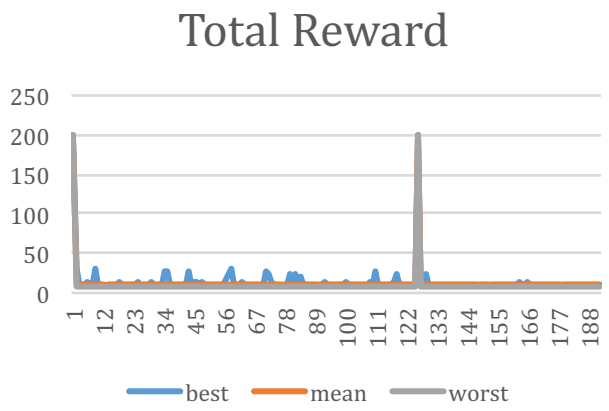
Reward basis function : ANN cartpole + RBF

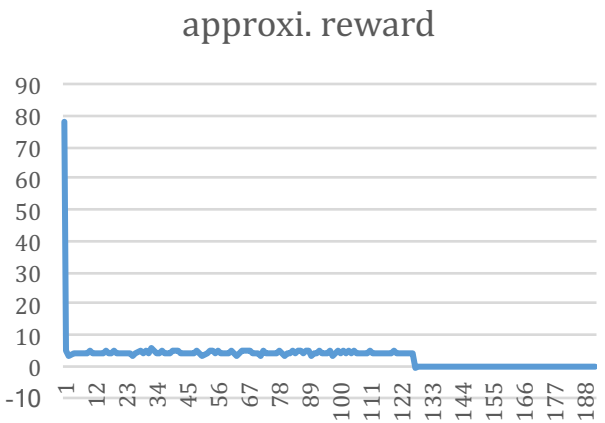
Reward basis feature dim(ϕ_{reward}) : 5

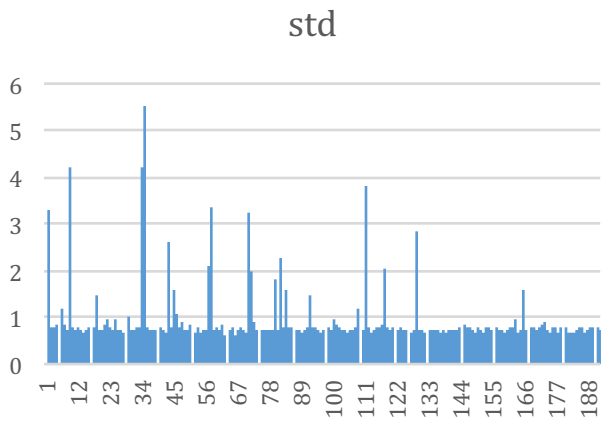
$$\hat{Q}^{\pi} = \Phi_{LSTD} w^{\pi}$$

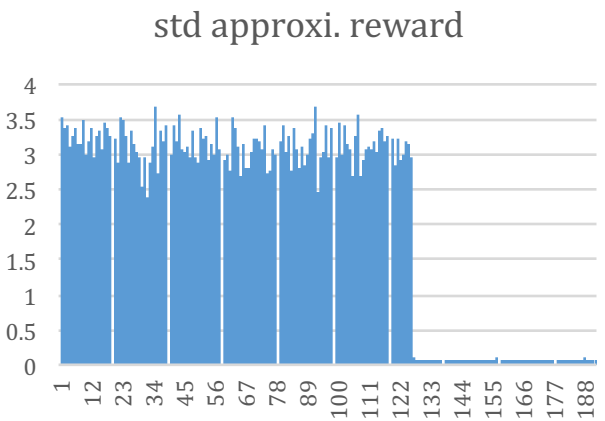
LSTD-Q basis function : ANN cartpole + RBF

LSTD-Q basis dimension(ϕ_{LSTD}) : 5









IRL + MC approach

$$\textbf{for each } (s,a,r,s') \in D$$
$$\tilde{\mathbf{A}} \leftarrow \tilde{\mathbf{A}} + \phi(s,a)\left(\phi(s,a) - \gamma\phi(s',\pi(s'))\right)^{\intercal}$$
$$\tilde{b} \leftarrow \tilde{b} + \phi(s,a)r$$

# of Expert Episode : 100

# of Episode for training w^{π}(LSTD-Q) : 1 (1 x 20 = 20 samples, |D|=20)

# of Episode for calculating μ^{π} : 100

# of Evaluation of each policy(w^{π}) : 100

$$R(s) = \theta^{\intercal}\phi_{reward}(s)$$

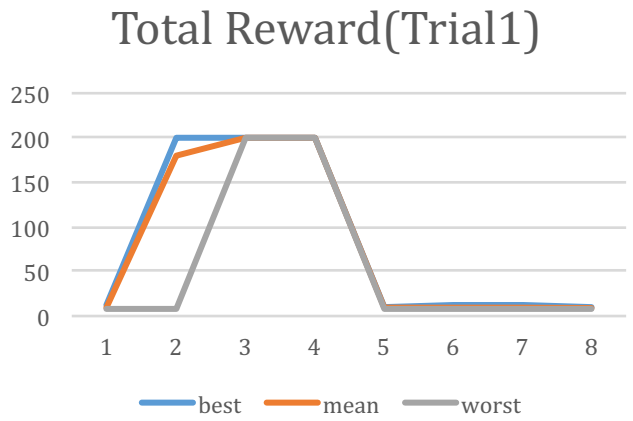
Reward basis function : ANN cartpole + RBF

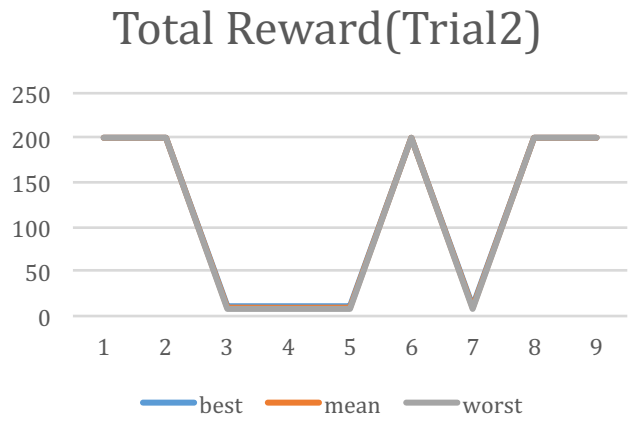
Reward basis feature dim(ϕ_{reward}) : 5

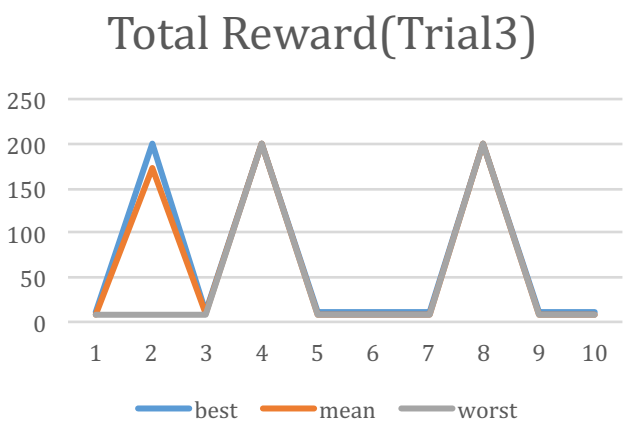
$$\hat{Q}^{\pi} = \Phi_{LSTD} w^{\pi}$$

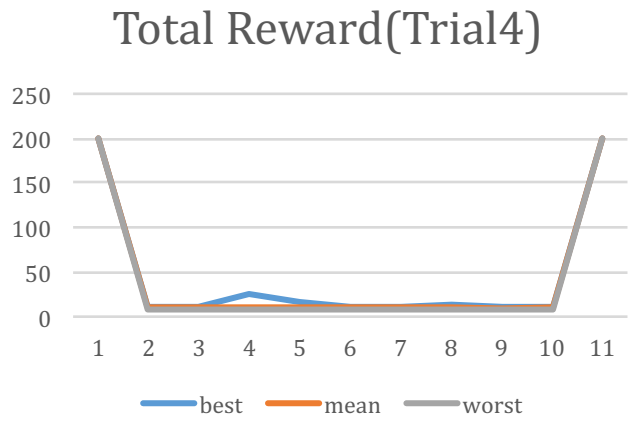
LSTD-Q basis function : ANN cartpole + RBF

LSTD-Q basis dimension(ϕ_{LSTD}) : 5









→ 20 trajectories for training w^{π} are enough!

IRL + MC approach

of Expert Episode : 100

# of Episode for training w^{π}(LSTD-Q) : 100

# of Episode for calculating μ^{π} : 100

# of Evaluation of each policy(w^{π}) : 100

$$R(s) = \theta^{\intercal}\phi_{reward}(s) \qquad exp(-\frac{\|s-\mu_j\|^2}{2\sigma^2})$$

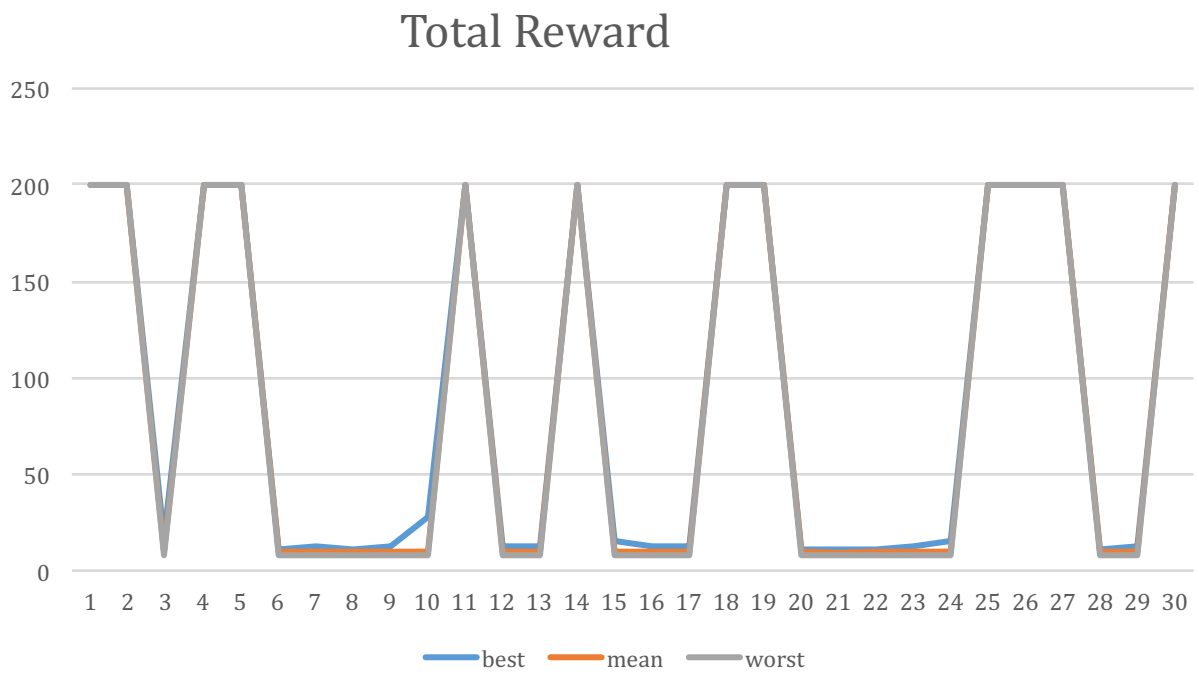
Reward basis function : **RBF** ~~(ANN cartpole + RBF)~~

Reward basis feature dim(ϕ_{reward}) : 5

$$\hat{Q}^{\pi} = \Phi_{LSTD} w^{\pi}$$

LSTD-Q basis function : ANN cartpole + RBF

LSTD-Q basis dimension(ϕ_{LSTD}) : 5



IRL + MC approach

of Expert Episode : 100

# of Episode for training w^{π}(LSTD-Q) : 100

# of Episode for calculating μ^{π} : 100

# of Evaluation of each policy(w^{π}) : 100

$$R(s) = \theta^{\intercal}\phi_{reward}(s)$$

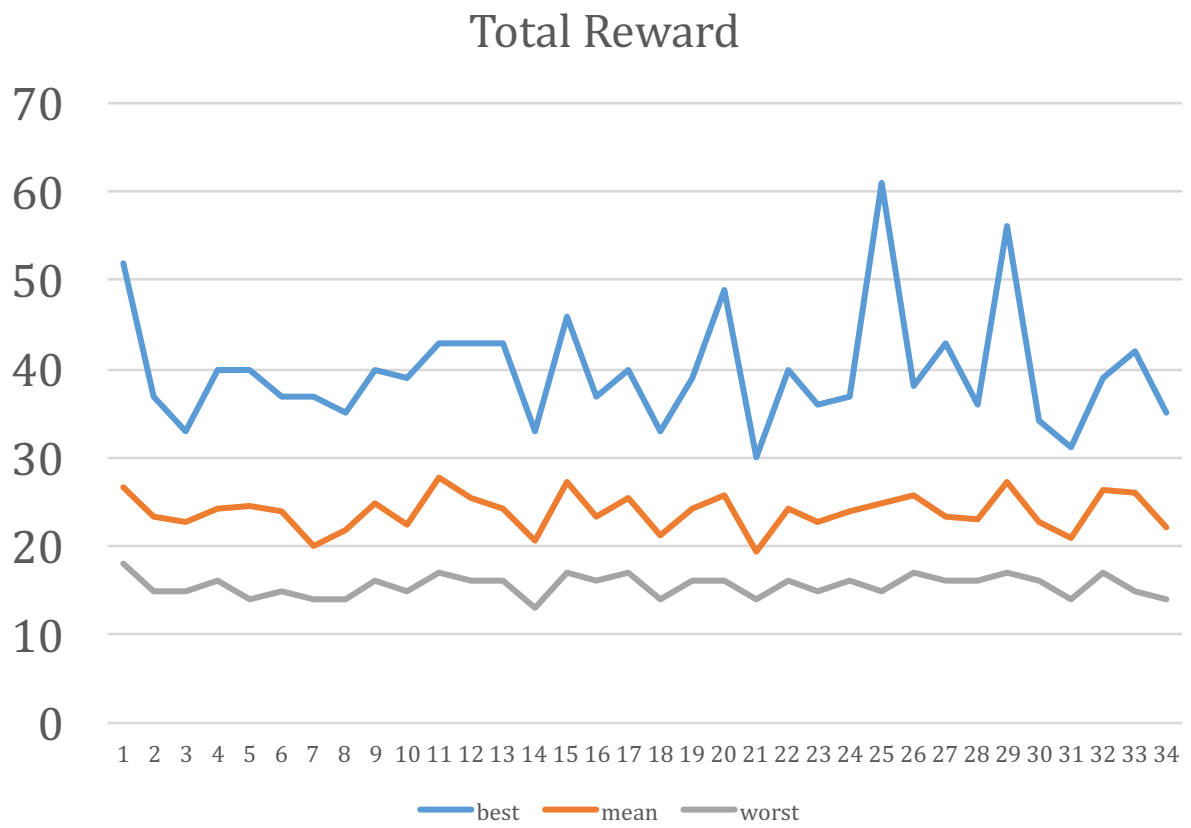
Reward basis function : ANN cartpole + RBF

Reward basis feature dim(ϕ_{reward}) : 5

$$\hat{Q}^{\pi} = \Phi_{LSTD} w^{\pi}$$

LSTD-Q basis function : **RBF** ~~(ANN cartpole + RBF)~~

LSTD-Q basis dimension(ϕ_{LSTD}) : 5



IRL + MC approach

of Expert Episode : 100

# of Episode for training w^{π}(LSTD-Q) : 100

# of Episode for calculating μ^{π} : 100

# of Evaluation of each policy(w^{π}) : 100

$$R(s) = \theta^{\intercal}\phi_{reward}(s)$$

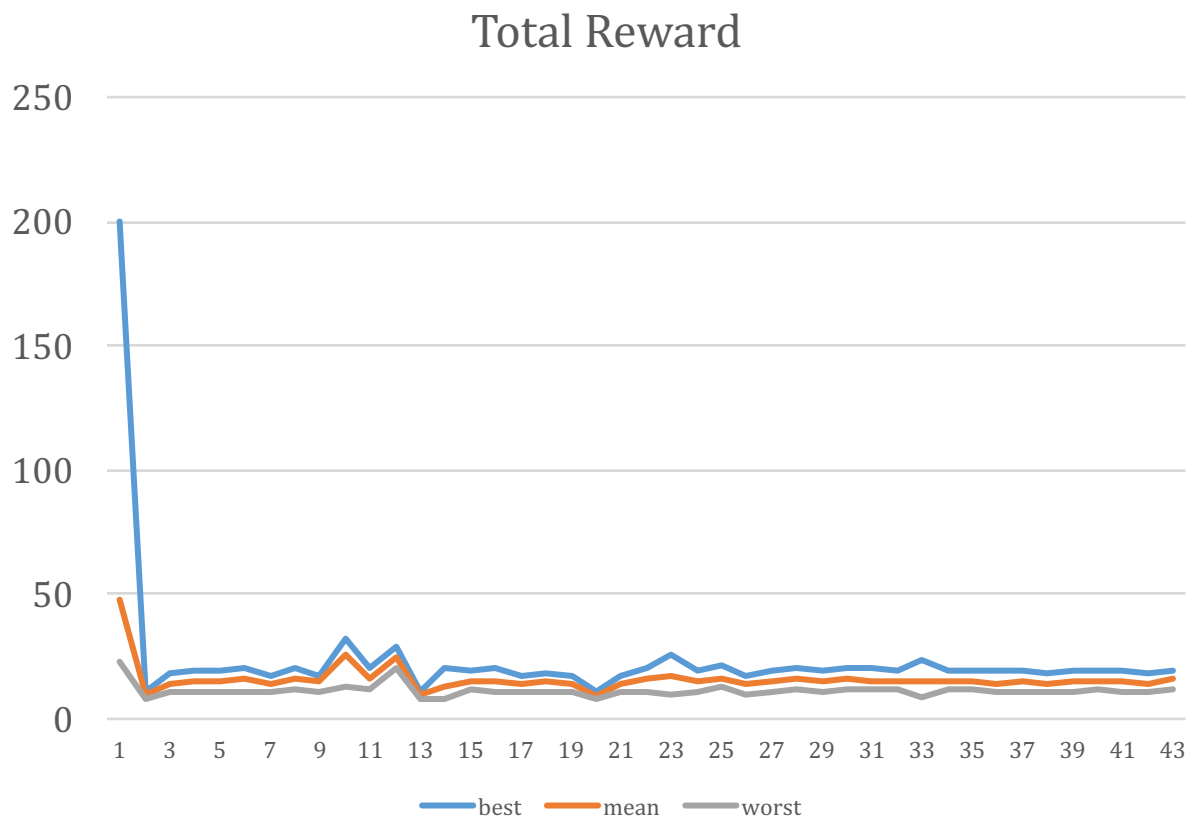
Reward basis function : **RBF** ~~(ANN cartpole + RBF)~~

Reward basis feature dim(ϕ_{reward}) : 5

$$\hat{Q}^{\pi} = \Phi_{LSTD} w^{\pi}$$

LSTD-Q basis function : **RBF** ~~(ANN cartpole + RBF)~~

LSTD-Q basis dimension(ϕ_{LSTD}) : 5



IRL + LSTD-mu

of Expert Episode : 100

# of Episode for training w^{π}(LSTD-Q) : 100

# of Episode for calculating μ^{π} : 100 (100 x ? samples) **(LSTD-mu)**

# of Evaluation of each policy(w^{π}) : 100

$$R(s) = \theta^{\intercal}\phi_{reward}(s)$$

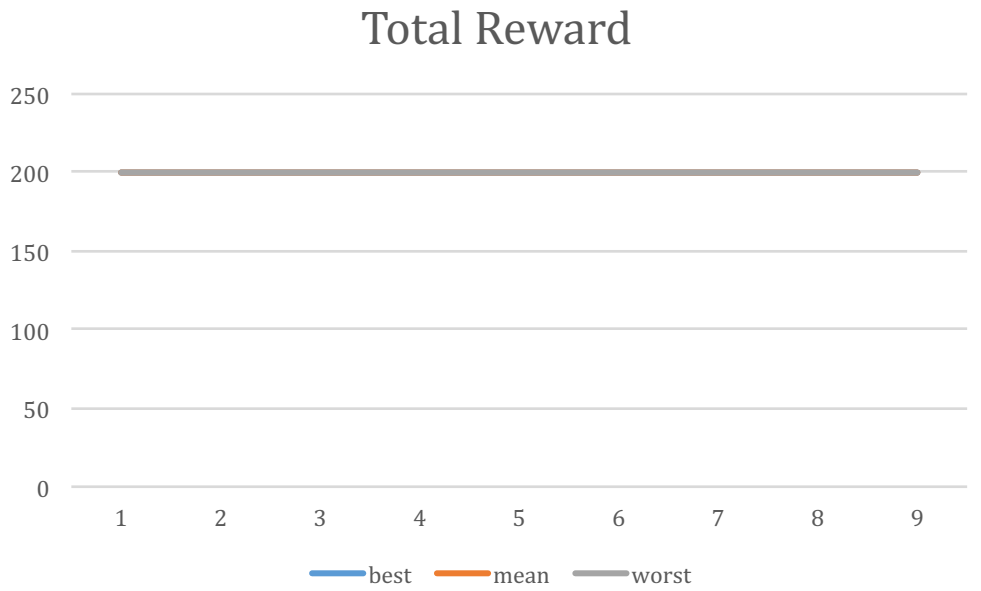
Reward basis function : ANN cartpole + RBF

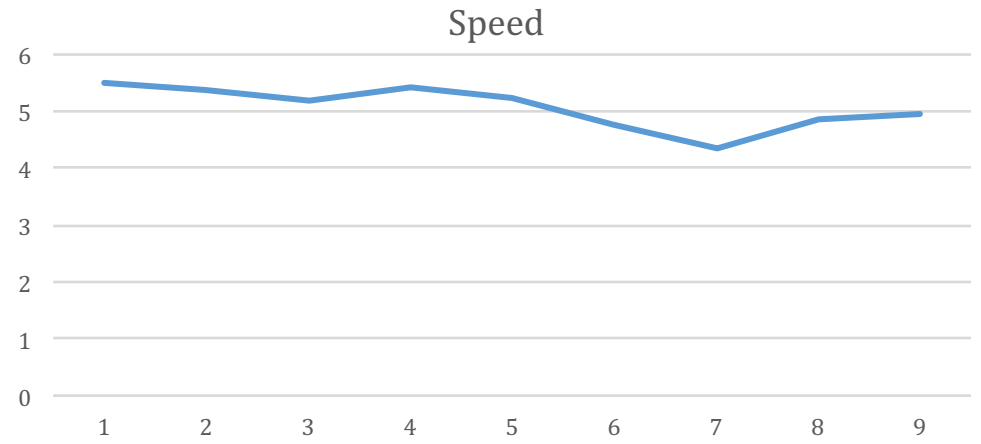
Reward basis feature dim(ϕ_{reward}) : 5

$$\hat{Q}^{\pi} = \Phi_{LSTD} w^{\pi}$$

LSTD-Q basis function : ANN cartpole + RBF

LSTD-Q basis dimension(ϕ_{LSTD}) : 5





Average speed : x5.06 (compared to MC approach with # Epi 100)

IRL + LSTD-mu

of Expert Episode : 100

# of Episode for training w^{π}(LSTD-Q) : 100

# of Episode for calculating μ^{π} : **1** ~~(100)~~ (1 x ? samples) **(LSTD-mu)**

# of Evaluation of each policy(w^{π}) : 100

$$R(s) = \theta^{\intercal}\phi_{reward}(s)$$

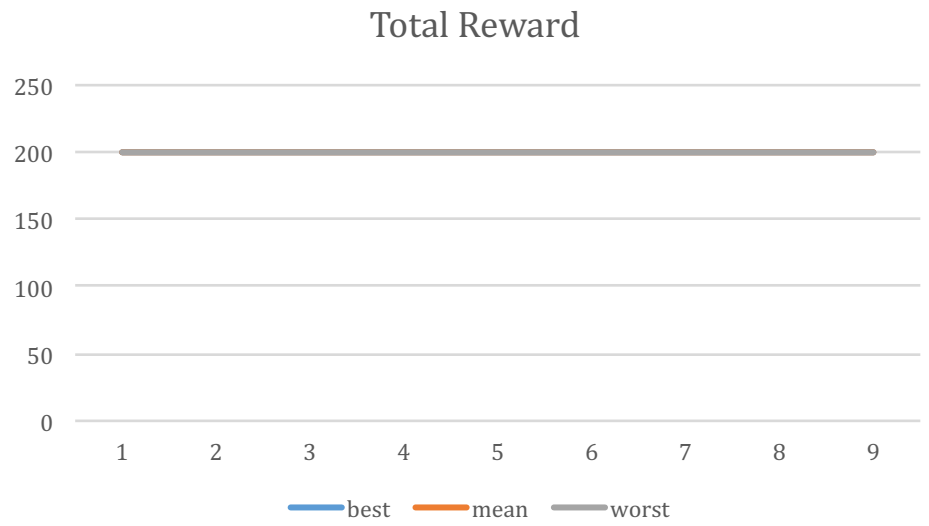
Reward basis function : ANN cartpole + RBF

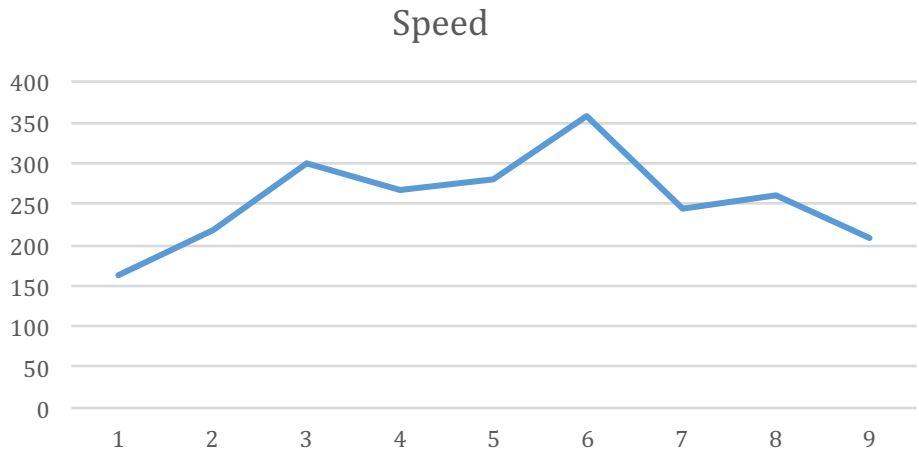
Reward basis feature dim(ϕ_{reward}) : 5

$$\hat{Q}^{\pi} = \Phi_{LSTD} w^{\pi}$$

LSTD-Q basis function : ANN cartpole + RBF

LSTD-Q basis dimension(ϕ_{LSTD}) : 5





Average speed : x255 (compared to MC approach with # Epi 100)

Given information, Expert Trajectories.

Expert Trajectory,

$$T_i = \{(s_t, a_t, s_{t+1})\ 1 \leq t \leq Done\}$$

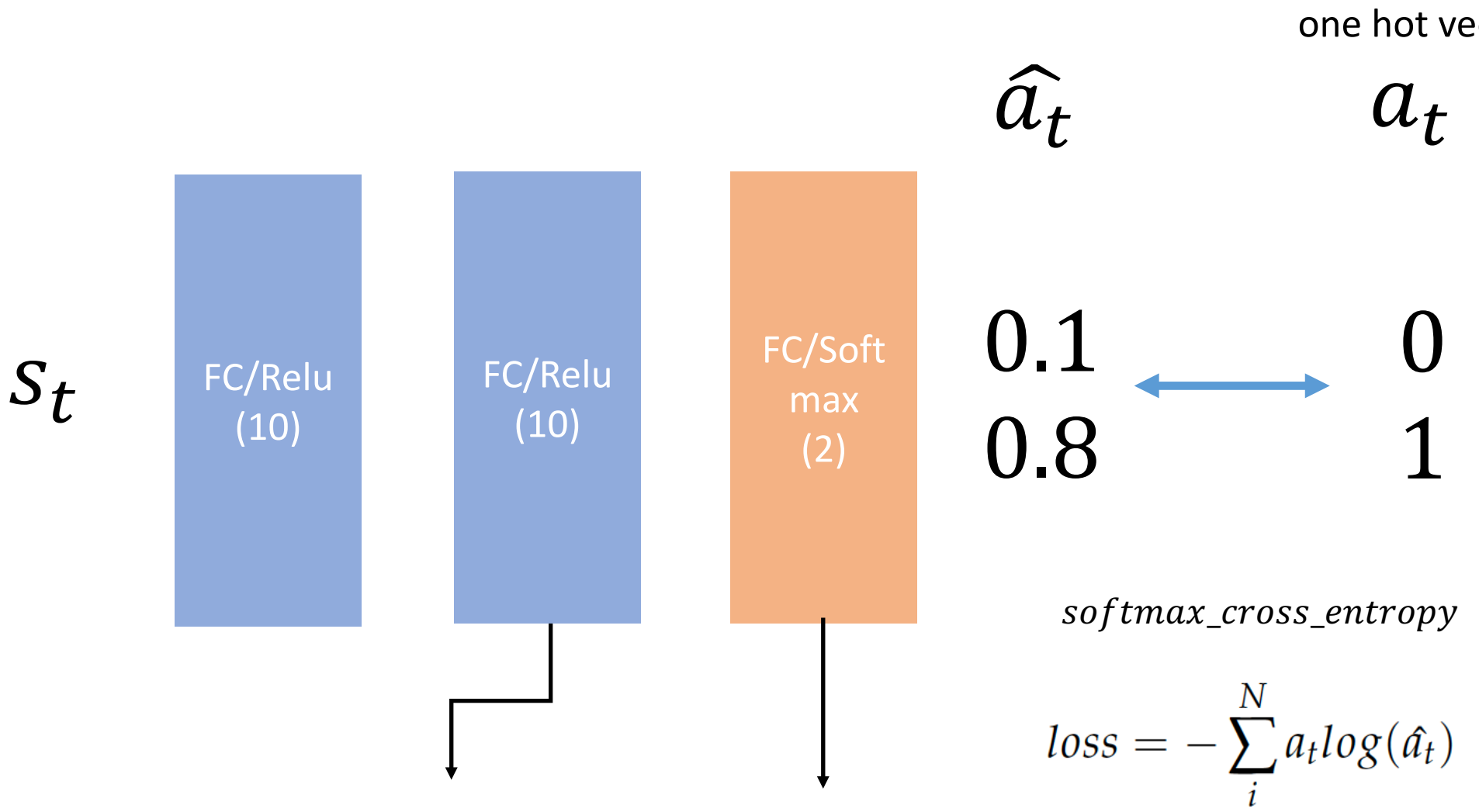
Expert Trajectories,

$$T_{collection} = \{T_i, 1 \leq i \leq n\}$$

Supervised Action Estimator? (not DQN but quite similar)

$$s_t(input) \rightarrow a_t(output)?$$

Deep Action Network(DAN)

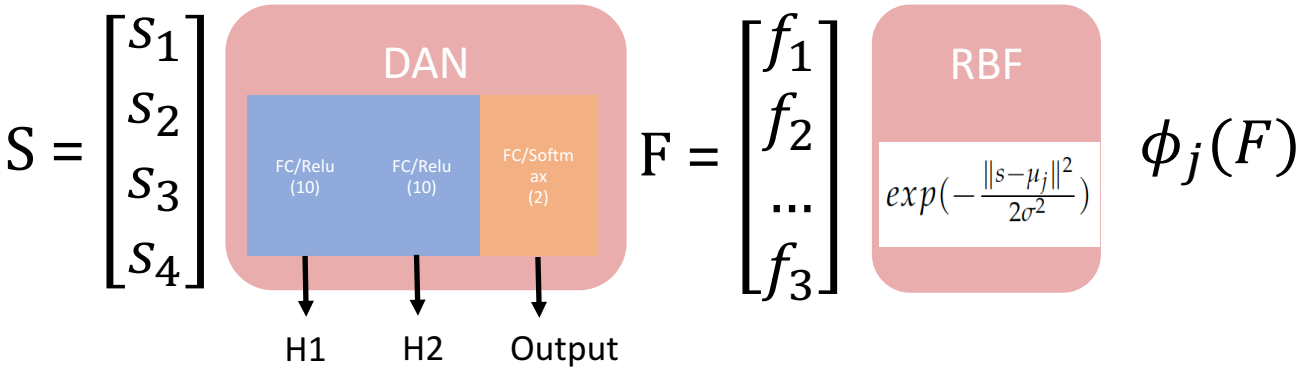


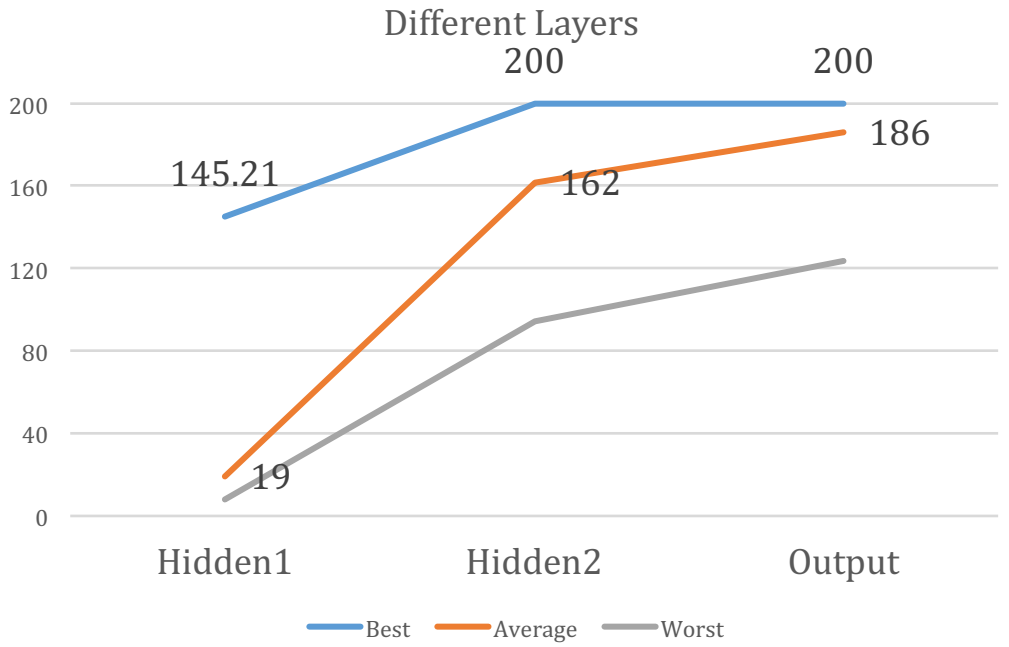
$$softmax\_cross\_entropy$$

$$loss = -\sum_{i}^{N} a_t log(\hat{a}_t)$$

DAN + RBF as basis function

DAN + RBF





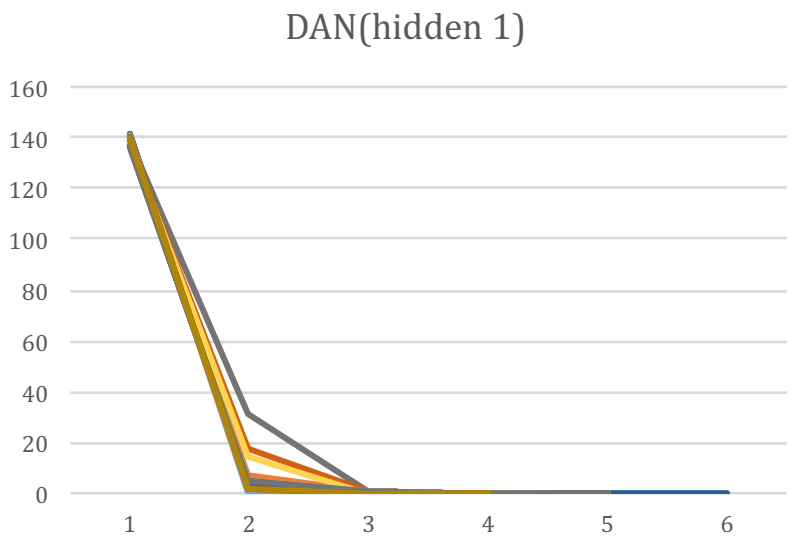
Episode for training w^{π}(LSTD-Q) : 100 (2000 samples)

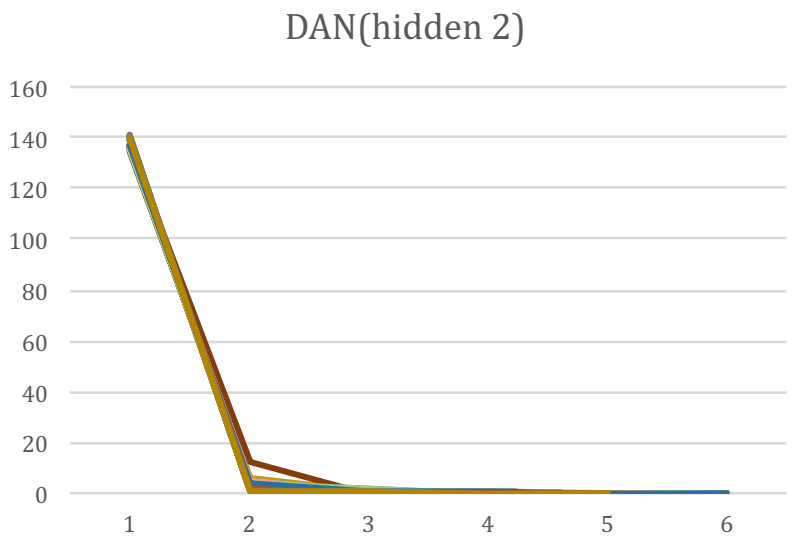
Evaluation for each policy(w^{π}) : 1000

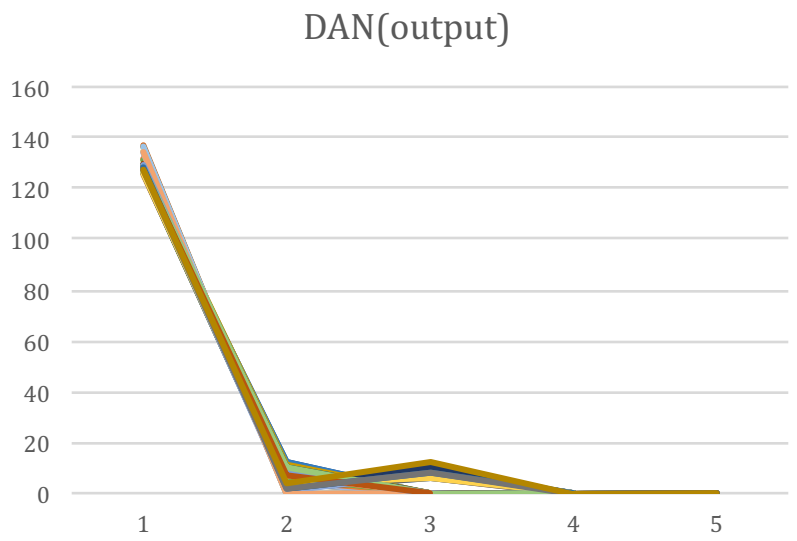
Trials(Experiments) : 100

basis feature(P) = 10

DAN + RBF as basis function







$$t = \|w^{\pi_{j-1}} - w^{\pi_j}\|_2$$

DAN + IRL + MC approach

of Expert Episode : 100

# of Episode for training w^{π}(LSTD-Q) : 100

# of Episode for calculating μ^{π} : 100

# of Evaluation of each policy(w^{π}) : 100

$$R(s) = \theta^{\intercal}\phi_{reward}(s)$$

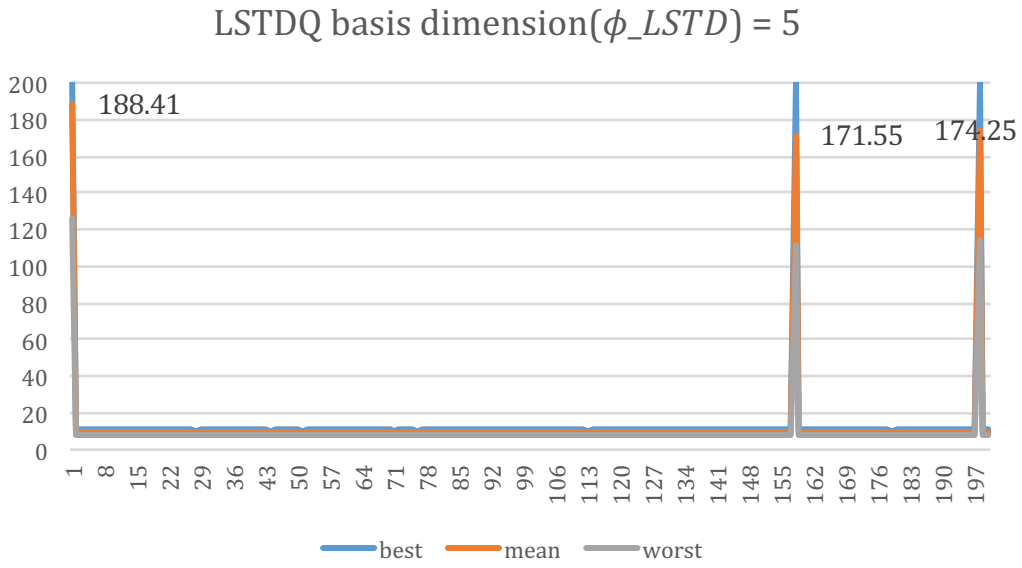
Reward basis function : RBF (Tensorflow conflict)

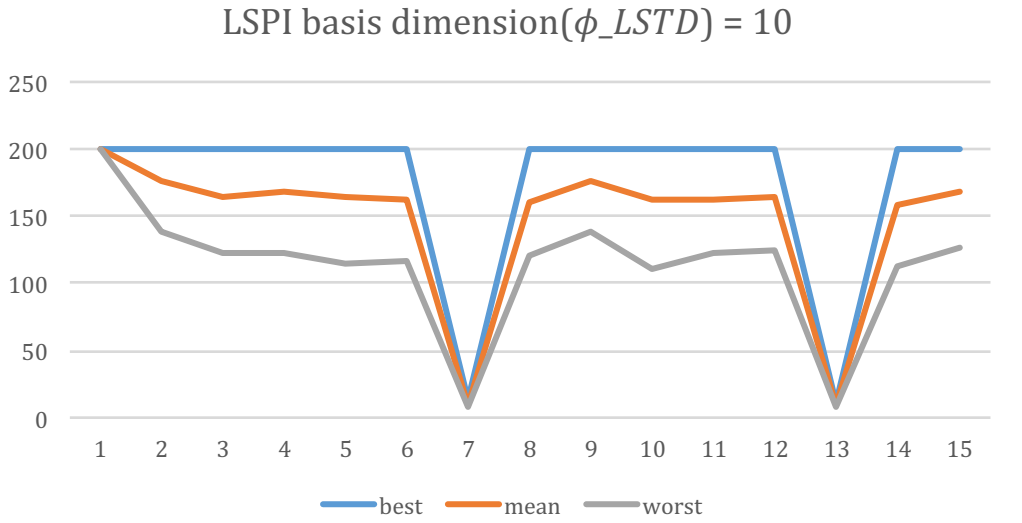
Reward basis feature dim(ϕ_{reward}) : 5

$$\hat{Q}^{\pi} = \Phi_{LSTD} w^{\pi}$$

LSTD-Q basis function : DAN**(output layer)** + RBF

LSTD-Q basis dimension(ϕ_{LSTD}) : {5, 10}





DAN + IRL + MC approach

of Expert Episode : 100

# of Episode for training w^{π}(LSTD-Q) : 100

# of Episode for calculating μ^{π} : 100

# of Evaluation of each policy(w^{π}) : 100

$$R(s) = \theta^{\intercal}\phi_{reward}(s)$$

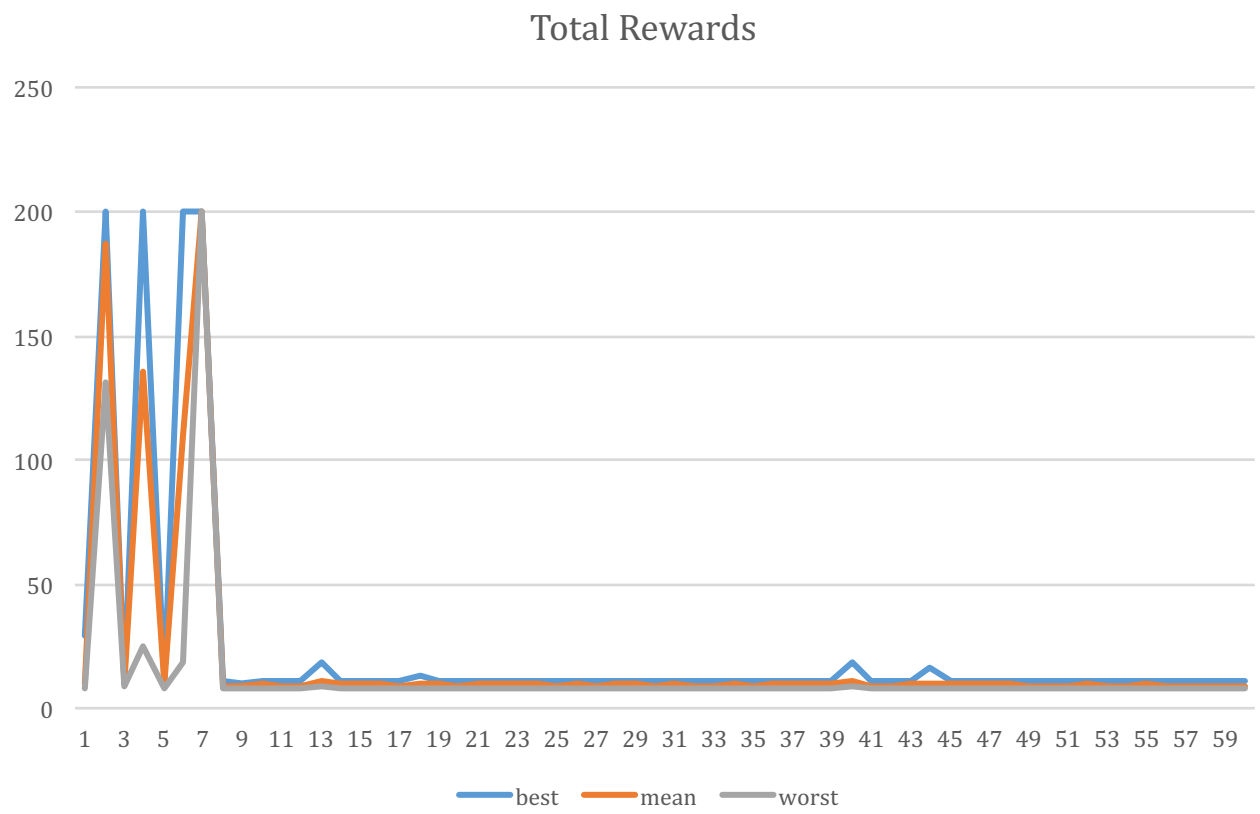
Reward basis function : RBF (Tensorflow conflict)

Reward basis feature dim(ϕ_{reward}) : 5

$$\hat{Q}^{\pi} = \Phi_{LSTD} w^{\pi}$$

LSTD-Q basis function : DAN**(hidden2)** + RBF

LSTD-Q basis dimension(ϕ_{LSTD}) : 5



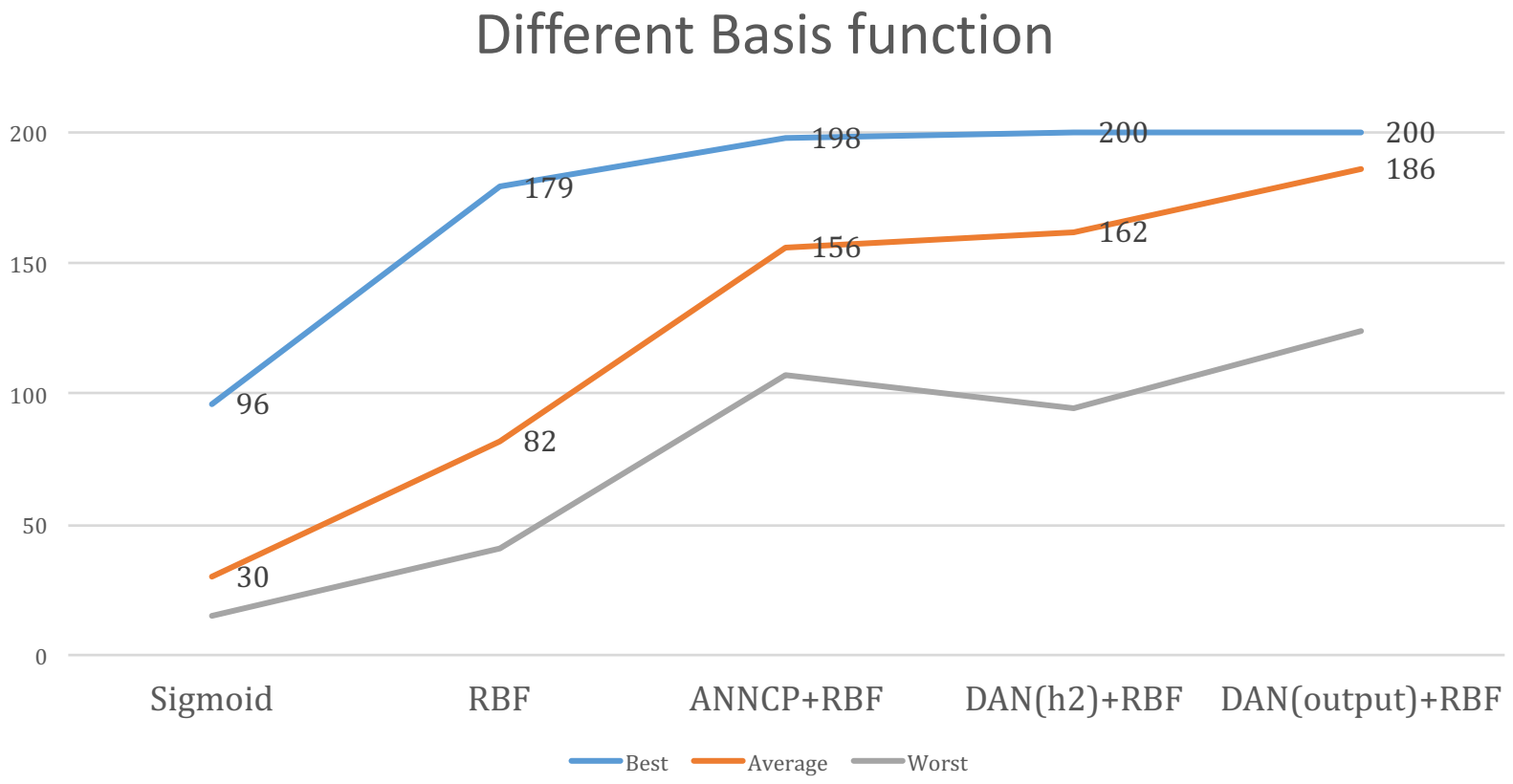
ANN for cartpole vs. Deep Action Network (for LSTD-Q basis func.)

- ANN for cartpole

label : should be provided how to act, according to danger situation(state)
only Cartpole problem

- Deep Action Network(DAN)

label : expert trajectory (s, a)
more general



# of Expert Episode : 100

# of Episode for training w^{π}(LSTD-Q) : 100

# of Episode for calculating μ^{π} : 100

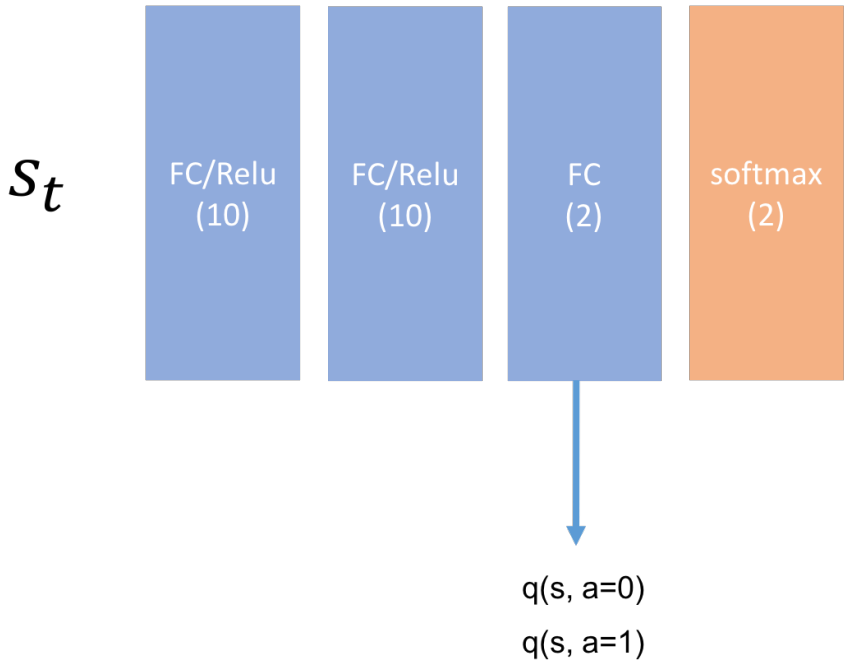
# of Evaluation of each policy(w^{π}) : 100

basis feature(P) = 10

DAN + DQN

Deep Action Network

Dataset = $\{(s_{i,t}, a_{i,t})\}$ i ∈ Trajectories, $1 \le t \le Done$



$\hat{a}_t$ one hot vector a_t

0.1 ⟷ 0

0.8 ⟷ 1

$softmax\_cross\_entropy$

$$loss = -\sum_{i}^{N} a_t log(\hat{a}_t)$$

Now, we know a_t (only for s_t)

$$s_t \rightarrow a_t \rightarrow s_{t+1} \rightarrow a_{t+1} \dots$$

$$Dataset = \{(s_{i,t}, a_{i,t}, s_{i,t+1})\ _{i \in Traj, 1 \le t \le Done}\}$$

$q(s,a) = r(s,a) + \gamma\, q(s', \pi(s'))$ (by Bellman equation)

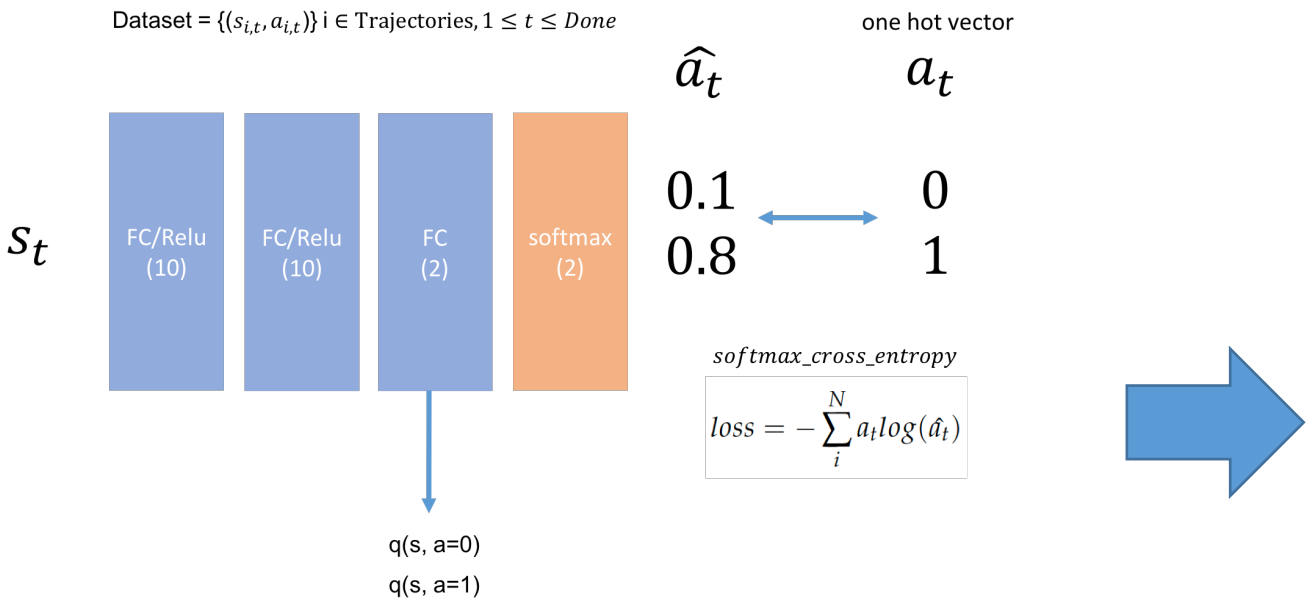
$r(s,a) = q(s,a) - \gamma\, q(s', \pi(s'))$

$r(s,a) = q(s,a) - \gamma \max_{a'} q(s', a')$

known known

DAN + DQN

1. DAN : Estimate Action and get q(s,a)



$$q(s,a) = r(s,a) + \gamma\, q(s', \pi(s')) \quad \text{(by Bellman equation)}$$

$$r(s,a) = q(s,a) - \gamma\, q(s', \pi(s'))$$

$$r(s,a) = q(s,a) - \gamma \max_{a'} q(s', a')$$

known known

3. MDP solver(control solver)

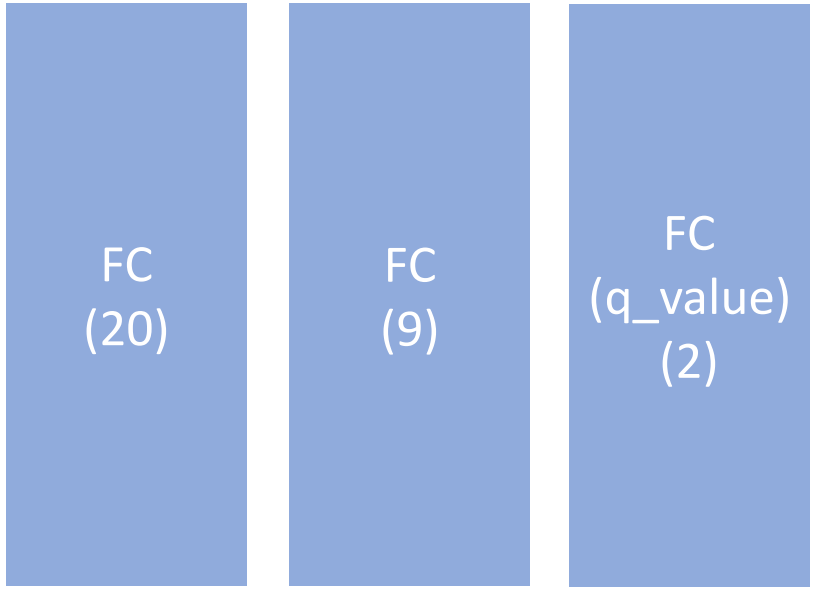
a. DQN

Deep Q Network(DQN)

Dataset = $\{(s_{i,t}, a_{i,t}, r_{i,t})\}$ i ∈ Trajectories, $1 \leq t \leq Done$

$\hat{q}_t$ q_t

s_t, a_t



1.4
6.4

$q_{t(s,a=0)}$
$q_{t(s,a=1)}$

$$Loss(w) = \sum (\hat{q}_t - q_t)^2$$

$$q_t(s,a) = r_t(s,a) - \gamma \max_{a'} q_t(s',a')$$

$$r_t(s,a) = q_t(s,a) - \gamma \max_{a'} q_t(s',a')$$

from DAN

DAN->DRN->DQN (Previous Results)

```
Deep Action Network Training iteration 1810, 0.03702983260154724
Deep Action Network Training iteration 1820, 0.015642434358596802
Deep Action Network Training iteration 1830, 0.010453480295836926
Deep Action Network Training iteration 1840, 0.050940681248903275
Deep Action Network Training iteration 1850, 0.011584173887968063
Deep Action Network Training iteration 1860, 0.006984895560890436
Deep Action Network Training iteration 1870, 0.020961787551641464
Deep Action Network Training iteration 1880, 0.024781856685876846
Deep Action Network Training iteration 1890, 0.039983492344617844
Deep Action Network Training iteration 1900, 0.04822589084506035
Deep Action Network Training iteration 1910, 0.01002182811498642
Deep Action Network Training iteration 1920, 0.012212877161800861
Deep Action Network Training iteration 1930, 0.05930352210998535
Deep Action Network Training iteration 1940, 0.05499160289764404
Deep Action Network Training iteration 1950, 0.040910910815000534
Deep Action Network Training iteration 1960, 0.013394010253250599
Deep Action Network Training iteration 1970, 0.03657654672861099
Deep Action Network Training iteration 1980, 0.021590299904346466
Deep Action Network Training iteration 1990, 0.030237708240747452
Testing Deep Action Network... 5 times
2019-01-09 10:17:15.416 Python[4242:642329] ApplePersistenceIgnoreStat
will not be touched. New state will be written to /var/folders/fw/st6g
Test DAN 0 : 30 timesteps
Test DAN 1 : 27 timesteps
Test DAN 2 : 31 timesteps
Test DAN 3 : 48 timesteps
Test DAN 4 : 24 timesteps
```

Expert trajectories were wrong.

```
Deep Reward Network iteration : 900, loss : 0.49980152130126954
Deep Reward Network iteration : 910, loss : 0.4997991180419922
Deep Reward Network iteration : 920, loss : 0.4997987747192383
Deep Reward Network iteration : 930, loss : 0.49979644775390625
Deep Reward Network iteration : 940, loss : 0.4997944641113281
Deep Reward Network iteration : 950, loss : 0.49979454040527344
Deep Reward Network iteration : 960, loss : 0.49979267120361326
Deep Reward Network iteration : 970, loss : 0.49979217529296877
Deep Reward Network iteration : 980, loss : 0.49979118347167967
Deep Reward Network iteration : 990, loss : 0.49979095458984374
DQN Training episode 0 timestep 1 reward -0.36829471588134766
DQN Training episode 0 timestep 2 reward -0.3732357621192932
DQN Training episode 0 timestep 3 reward -0.3663175404071808
DQN Training episode 0 timestep 4 reward -0.36510223150253296
DQN Training episode 0 timestep 5 reward -0.3578903079032898
DQN Training episode 0 timestep 6 reward -0.3613246977329254
DQN Training episode 0 timestep 7 reward -0.3559782803058624
DQN Training episode 0 timestep 8 reward -0.35866567492485046
DQN Training episode 0 timestep 9 reward -0.35467588901519775
DQN Training episode 0 timestep 10 reward -0.3565685749053955
DQN Training episode 0 timestep 11 reward -0.3541203737258911
DQN Training episode 0 timestep 12 reward -0.3483473062515259
DQN Training episode 0 timestep 13 reward -0.3440174162387848
DQN Training episode 0 timestep 14 reward -0.35596293210983276
DQN Training Episode 0 finished after 15 timesteps
average reward : -0.3854454053299768
```

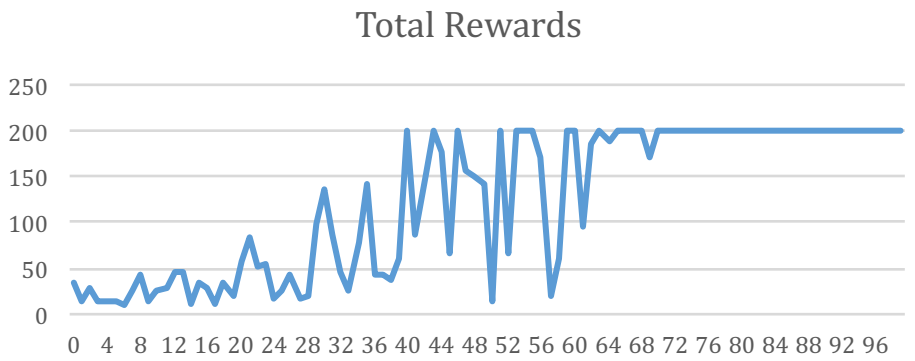
fortunately, reward are same!

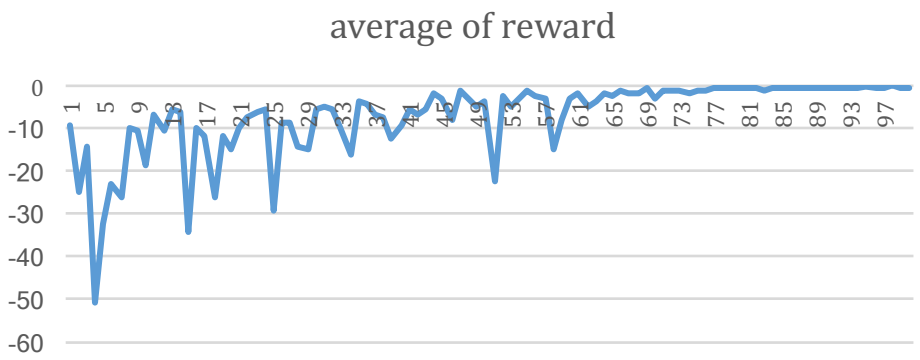
DAN->DQN

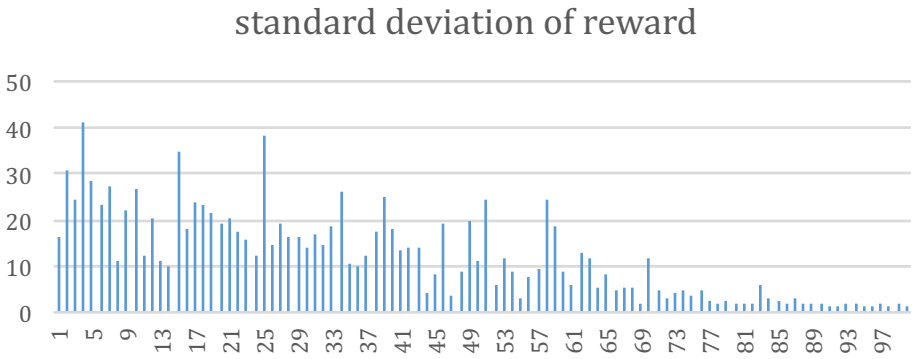
DAN

```
Deep Action Network Training iteration 9960, 0.5171039593219757
Deep Action Network Training iteration 9970, 0.5489578205347061
Deep Action Network Training iteration 9980, 0.5222935244441033
Deep Action Network Training iteration 9990, 0.5158185619115829
saveing our trained weights!!
Testing Deep Action Network... 100 times
Test DAN 0 : 200 timesteps
Test DAN 1 : 200 timesteps
Test DAN 2 : 200 timesteps
Test DAN 3 : 200 timesteps
Test DAN 4 : 200 timesteps
Test DAN 5 : 200 timesteps
Test DAN 6 : 200 timesteps
Test DAN 7 : 200 timesteps
Test DAN 8 : 200 timesteps
Test DAN 9 : 200 timesteps
Test DAN 10 : 200 timesteps
Test DAN 11 : 200 timesteps
Test DAN 12 : 200 timesteps
Test DAN 13 : 200 timesteps
Test DAN 14 : 200 timesteps
Test DAN 15 : 200 timesteps
Test DAN 16 : 200 timesteps
Test DAN 17 : 200 timesteps
Test DAN 18 : 200 timesteps
Test DAN 19 : 150 timesteps
```

average reward : 194.85







What I implemented,

1. LSPI
2. basis function : sigmoid, RBF, ANNCP, DAN(h2, output)
3. Batch, off-policy and model-free Apprenticeship learning(LSTD-mu)
4. Deep Action Network(DAN) + DQN

Next steps, (in order of importance)

1. ~~Fix LSPI problem(oscillation)~~
2. ~~Analyze Deep Action, Reward Network~~
3. Other simulator (CartPole might be easy)
4. IRL + DQN (as MDP solver) or apply other Deep RL
5. Projection method → Quadratic programming (reward bounding part)
6. ~~Apply deep features for LSTD-mu~~

The code is available on the my github page(https://github.com/jeonggwanlee/LSTD-mu)

Backup slide

Backup slide : Radial Basis Functions

RBFs are the natural generalization of coarse coding(binary features {0, 1}) to continuous-valued features([0, 1]).

A typical RBF feature has a Gaussian response dependent on the distance between the state, s, and the feature's center state, c_i, and the feature's width, σ_i.

$$x_i(s) \doteq \exp\left(-\frac{||s - c_i||^2}{2\sigma_i^2}\right)$$

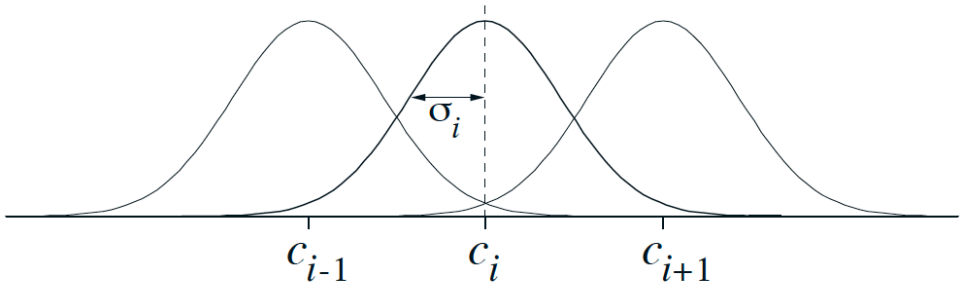
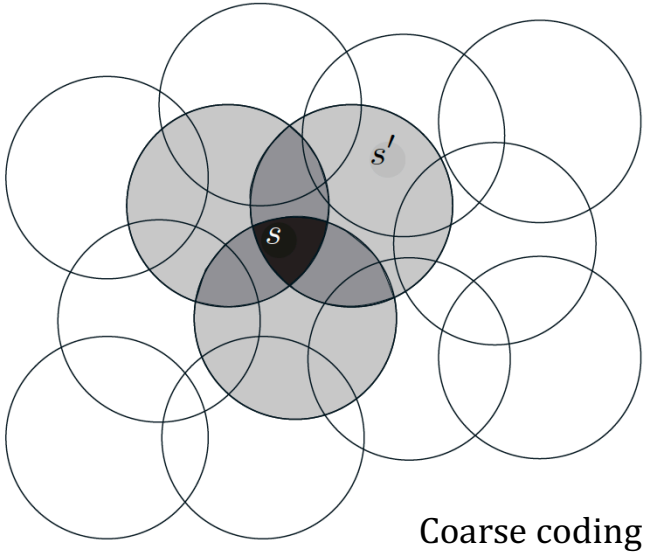


Figure 9.13: One-dimensional radial basis functions.



Coarse coding

Advantage : vary smoothly, and differentiable.