Equi-gradient TD Learning
TD(\(\lambda\)) on adaptative Basis Functions Network

Manuel Loth,
Manuel Davy, Rémi Coulom, Philippe Preux

INRIA Sequel Lille, France

June 29th 2006
Introduction

- Kernel methods
- Reinforcement learning
Introduction

Kernel methods
new Regression method \sim kernel method
(regularized sample-based linear approximation)

Reinforcement learning
Kernel methods
new *Regression method* \(\sim\) kernel method
(regularized sample-based linear approximation)

*equi-gradient descent*

Reinforcement learning
Introduction

Kernel methods
new Regression method \( \sim \) kernel method
(regularized sample-based linear approximation)

equi-gradient descent

Reinforcement learning
Scheme for using it in \( TD(\lambda) \)
Kernel methods
new *Regression method* $\sim$ kernel method
(regularized sample-based linear approximation)

**equi-gradient descent**

Reinforcement learning
Scheme for using it in $TD(\lambda)$

**equi-gradient $TD(\lambda)$**
Regression
Regression

- unknown $f : \mathcal{X} \rightarrow \mathbb{R}$
Regression

- unknown $f : \mathcal{X} \to \mathbb{R}$
- samples $(x_1, y_1), \ldots, (x_n, y_n) \in \mathcal{X} \times \mathbb{R}$
Regression

- unknown \( f : \mathcal{X} \rightarrow \mathbb{R} \)
- samples \((x_1, y_1), \ldots, (x_n, y_n) \in \mathcal{X} \times \mathbb{R}\)
- **accurate and simple** approximation \( \hat{f} \)
Regression

- unknown \( f : \mathcal{X} \to \mathbb{R} \)
- samples \((x_1, y_1), \ldots, (x_n, y_n) \in \mathcal{X} \times \mathbb{R}\)
- **accurate** and **simple** approximation \( \hat{f} \)

\[
\mathbf{y} = (y_1, \ldots, y_n)^T
\]

\[
\hat{\mathbf{y}} = (\hat{f}(x_1), \ldots, \hat{f}(x_n))^T
\]
Regression

- unknown $f : \mathcal{X} \to \mathbb{R}$
- samples $(x_1, y_1), \ldots, (x_n, y_n) \in \mathcal{X} \times \mathbb{R}$
- accurate and simple approximation $\hat{f}$

$$y = (y_1, \ldots, y_n)^T$$

$$\hat{y} = (\hat{f}(x_1), \ldots, \hat{f}(x_n))^T$$

minimize $\mathcal{L}(y, \hat{y})$
Regression

- unknown \( f : \mathcal{X} \rightarrow \mathbb{R} \)
- samples \( (x_1, y_1), \ldots, (x_n, y_n) \in \mathcal{X} \times \mathbb{R} \)
- accurate and simple approximation \( \hat{f} \)

\[
y = (y_1, \ldots, y_n)^T
\]

\[
\hat{y} = (\hat{f}(x_1), \ldots, \hat{f}(x_n))^T
\]

minimize \( \mathcal{L}(y, \hat{y}) + \lambda \left| \hat{f} \right| \).
Regression

- unknown \( f : \mathcal{X} \rightarrow \mathbb{R} \)
- samples \((x_1, y_1), \ldots, (x_n, y_n) \in \mathcal{X} \times \mathbb{R}\)
- accurate and simple approximation \( \hat{f} \)

\[
\mathbf{y} = (y_1, \ldots, y_n)^T
\]
\[
\hat{\mathbf{y}} = (\hat{f}(x_1), \ldots, \hat{f}(x_n))^T
\]

minimize \( \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}) + \lambda \| \hat{f} \| \).
Regression

- unknown $f : X \rightarrow \mathbb{R}$
- samples $(x_1, y_1), \ldots, (x_n, y_n) \in X \times \mathbb{R}$
- accurate and simple approximation $\hat{f}$

\[
\begin{align*}
\mathbf{y} &= (y_1, \ldots, y_n)^T \\
\hat{\mathbf{y}} &= (\hat{f}(x_1), \ldots, \hat{f}(x_n))^T
\end{align*}
\]

minimize $\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}) + \lambda |\hat{f}|$

kernel methods:
Regression

- unknown \( f : \mathcal{X} \rightarrow \mathbb{R} \)
- samples \((x_1, y_1), \ldots, (x_n, y_n) \in \mathcal{X} \times \mathbb{R}\)
- accurate and simple approximation \( \hat{f} \)

\[
\mathbf{y} = (y_1, \ldots, y_n)^T
\]
\[
\hat{\mathbf{y}} = (\hat{f}(x_1), \ldots, \hat{f}(x_n))^T
\]

minimize \( \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}) + \lambda \left\| \hat{f} \right\| \)

kernel methods:
- \( k : \mathcal{X}^2 \rightarrow \mathbb{R} \sim \) similarity
unknown $f : \mathcal{X} \to \mathbb{R}$

samples $(x_1, y_1), \ldots, (x_n, y_n) \in \mathcal{X} \times \mathbb{R}$

accurate and simple approximation $\hat{f}$

$\mathbf{y} = (y_1, \ldots, y_n)^T$

$\hat{\mathbf{y}} = (\hat{f}(x_1), \ldots, \hat{f}(x_n))^T$

minimize $\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}) + \lambda \left\| \hat{f} \right\|$.

kernel methods:

- $k : \mathcal{X}^2 \to \mathbb{R} \sim$ similarity

- 1 basis function / learning point: $\phi_i(x) = k(x_i, x)$
Regression

- unknown $f : \mathcal{X} \rightarrow \mathbb{R}$
- samples $(x_1, y_1), \ldots , (x_n, y_n) \in \mathcal{X} \times \mathbb{R}$
- accurate and simple approximation $\hat{f}$

$$y = (y_1, \ldots , y_n)^T$$

$$\hat{y} = (\hat{f}(x_1), \ldots , \hat{f}(x_n))^T$$

minimize $L(y, \hat{y}) + \lambda \left\| \hat{f} \right\|$.

Kernel methods:

- $k : \mathcal{X}^2 \rightarrow \mathbb{R} \sim \text{similarity}$
- 1 basis function / learning point: $\phi_i(x) = k(x_i, x)$
- $\hat{f}(x) = \sum_i w_i \phi_i(x)$
regularized sample-based linear approximators
regularized sample-based linear approximators

- samples
regularized sample-based linear approximators

- samples $\sim$ basis functions $\phi_1, \ldots, \phi_m : \mathcal{X} \rightarrow \mathbb{R}$
regularized sample-based linear approximators

- samples $\sim$ basis functions $\phi_1, \ldots, \phi_m : \mathcal{X} \rightarrow \mathbb{R}$
- $\hat{f}(x) = \sum_i w_i \phi_i(x)$
regularized sample-based linear approximators

- samples \( \sim \) basis functions \( \phi_1, \ldots, \phi_m : \mathcal{X} \rightarrow \mathbb{R} \)
- \( \hat{f}(x) = \sum_i w_i \phi_i(x) \)

\[
\Phi = \begin{pmatrix}
\phi_1(x_1) & \ldots & \phi_m(x_1) \\
\vdots & \ddots & \vdots \\
\phi_1(x_n) & \ldots & \phi_m(x_n)
\end{pmatrix}, \quad \mathbf{w} = (w_1, \ldots, w_m)^T
\]
regularized sample-based linear approximators

- samples \( \sim \) basis functions \( \phi_1, \ldots, \phi_m : \mathcal{X} \to \mathbb{R} \)
- \( \hat{f}(x) = \sum_i w_i \phi_i(x) \)

\[
\Phi = \begin{pmatrix}
\phi_1(x_1) & \cdots & \phi_m(x_1) \\
\vdots & \ddots & \vdots \\
\phi_1(x_n) & \cdots & \phi_m(x_n)
\end{pmatrix}, \quad w = (w_1, \ldots, w_m)^T
\]

\[
\hat{y} = \Phi w
\]
regularized sample-based linear approximators

- samples $\sim$ basis functions $\phi_1, \ldots, \phi_m : X \rightarrow \mathbb{R}$
- $\hat{f}(x) = \sum_i w_i \phi_i(x)$

$$
\Phi = \begin{pmatrix}
\phi_1(x_1) & \cdots & \phi_m(x_1) \\
\vdots & \ddots & \vdots \\
\phi_1(x_n) & \cdots & \phi_m(x_n)
\end{pmatrix}, \quad w = (w_1, \ldots, w_m)^T
$$

$$
\hat{y} = \Phi w
$$

$$
\mathcal{L}(y, \Phi w) + \lambda \vert \Phi w \vert
$$
LASSO

\[ \mathcal{L}(y, \Phi w) + \lambda |\Phi w| \]
LASSO

$$(y - \Phi w)^2 + \lambda |\Phi w|$$
LASSO

\[(y - \Phi w)^2 + \lambda \sum_i |w_i| \quad \text{(LASSO)}\]
LASSO

\[(y - \Phi w)^2 + \lambda \sum_i |w_i| \quad \text{(LASSO)}\]

- Basis Pursuit
- Chen 95
LASSO

\[(y - \phi w)^2 + \lambda \sum_i |w_i|\]  \text{(LASSO)}

- Basis Pursuit \hspace{1cm} \text{Chen 95}
- Adaptative Ridge Regression \hspace{1cm} \text{Granvalet 98}
LASSO

\[(y - \Phi w)^2 + \lambda \sum_i |w_i| \quad \text{(LASSO)}\]

- Basis Pursuit \hspace{1cm} Chen 95
- Adaptative Ridge Regression \hspace{1cm} Granvalet 98
- Matching Pursuit \hspace{1cm} Mallat & Zhang 93
LASSO

\[(y - \Phi w)^2 + \lambda \sum_{i} |w_i| \quad \text{(LASSO)}\]

- Basis Pursuit \quad Chen 95
- Adaptative Ridge Regression \quad Granvalet 98
- Matching Pursuit \quad Mallat & Zhang 93
- iterative gradient descent \quad Osborne et al. 00
Equi-gradient descent

*Least-Angle Regression Stagewise/laSSO*
Equi-gradient descent

Least-Angle Regression Stagewise/laSSO

- Efron 2002 variable selection
Equi-gradient descent

Least-Angle Regression Stagewise/laSSO

- Efron 2002 variable selection
- Guigue 2005 kernelization
Equi-gradient descent

**Least-Angle Regression Stagewise/laSSO**
- Efron 2002 variable selection
- Guigue 2005 kernelization
- Generalization, simplification → *Equi-gradient descent*
Equi-gradient descent
Equi-gradient descent

Given $\lambda$
Given $\lambda$

Given $\text{sgn}(\mathbf{w}) = (0, 0, -1, 0, 1, \ldots)^T$
Equi-gradient descent

Given $\lambda$

Given $\text{sgn}(\mathbf{w}) = (0, 0, -1, 0, 1, \ldots)^T$

$\mathbf{w}_a = \textit{active}$ (non-zero) weights
Equi-gradient descent

Given $\lambda$
Given $\text{sgn}(\mathbf{w}) = (0, 0, -1, 0, 1, \ldots)^T$
$\mathbf{w}_a = \textbf{active}$ (non-zero) weights
$\Phi_a = \text{active basis functions}$
Given $\lambda$

Given $\text{sgn}(w) = (0, 0, -1, 0, 1, \ldots)^T$

$w_a = \textbf{active}$ (non-zero) weights

$\Phi_a = \text{active basis functions}$

$\hat{y} = \Phi_a w_a$
Equi-gradient descent

Given $\lambda$
Given $\text{sgn}(w) = (0, 0, -1, 0, 1, \ldots)^T$
$w_a = \text{active}$ (non-zero) weights
$\Phi_a = \text{active basis functions}$
$\hat{y} = \Phi_a w_a$

minimize $(y - \Phi_a w_a)^2 + \lambda \sum_i |w_i|$
Equi-gradient descent

Given $\lambda$

Given $\text{sgn}(\mathbf{w}) = (0, 0, -1, 0, 1, \ldots)^T$

$\mathbf{w}_a = \text{active} \ (\text{non-zero}) \ \text{weights}$

$\Phi_a = \text{active basis functions}$

$\hat{\mathbf{y}} = \Phi_a \mathbf{w}_a$

$$\text{minimize} \ (\mathbf{y} - \Phi_a \mathbf{w}_a)^2 + \lambda \sum_i |w_i|$$

$$\Phi_a^T(\mathbf{y} - \Phi_a \mathbf{w}_a) = \lambda \text{sgn}_a$$
Equi-gradient descent

Given $\lambda$

Given $\text{sgn}(w) = (0, 0, -1, 0, 1, \ldots)^T$

$w_a = \text{active}$ (non-zero) weights

$\Phi_a = \text{active basis functions}$

$\hat{y} = \Phi_a w_a$

minimize $(y - \Phi_a w_a)^2 + \lambda \sum_i |w_i|$

$\Phi_a^T(y - \Phi_a w_a) = \lambda \text{sgn}_a$

for all active $w_i$, $\left|\frac{\partial L}{\partial w_i}\right| = \lambda$
Equi-gradient descent

Given $\lambda$

Given $\text{sgn}(w) = (0, 0, -1, 0, 1, \ldots)^T$

$w_a = \textbf{active}$ (non-zero) weights

$\Phi_a = \text{active basis functions}$

$\hat{y} = \Phi_a w_a$

minimize $(y - \Phi_a w_a)^2 + \lambda \sum_i |w_i|$

$\Phi_a^T(y - \Phi_a w_a) = \lambda \text{sgn}_a$

for all active $w_i$, $\left| \frac{\partial L}{\partial w_i} \right| = \lambda$

for all inactive $w_i$, $\left| \frac{\partial L}{\partial w_i} \right| \leq \lambda$
Regularization path
Regularization path

samples

\((x_1, y_1)\) \quad (x_2, y_2)\)
Regularization path

samples

$(x_1, y_1)$  $(x_2, y_2)$
Regularization path

\[ (x_2, y_2) \]

\[ y_2 \]

\[ y_1 \]

\[ (x_1, y_1) \]

\[ (x_2, y_2) \]

\[ \text{samples} \]

\[ (x_1, y_1) \]

\[ (x_2, y_2) \]
Regularization path

\[ \begin{align*}
\phi_2(x_2) &\rightarrow y_2 \\
\phi_1(x_1) &\rightarrow y_1 \\
(x_2, y_2) &\text{ samples} \\
(x_1, y_1) &\text{ basis functions } \phi_1, \phi_2
\end{align*} \]
Regularization path

Samples
\((x_1, y_1)\) \((x_2, y_2)\)

Basis functions
\(\phi_1, \phi_2\)

Reg. coefficient \(\lambda\)
Regularization path

samples
$(x_1, y_1) \quad (x_2, y_2)$

basis functions
$\phi_1, \phi_2$

reg. coefficient $\lambda$
$\hat{f}_\lambda = w_1 \phi_1(x) + w_2 \phi_2(x)$
Regularization path

\[ (x_2) \]

\[ y_2 \]

\[ \hat{f}_\lambda(x_2) \]

\[ \phi_2(x_2) \]

\[ \Phi_1 \]

\[ \phi_2(x_1) \]

\[ \hat{f}_\lambda(x_1) \]

\[ y_1 \]

\[ \hat{y}_\lambda \]

samples

\[ (x_1, y_1) \quad (x_2, y_2) \]

basis functions

\[ \phi_1, \phi_2 \]

reg. coefficient \( \lambda \)

\[ \hat{f}_\lambda = w_1 \phi_1(x) + w_2 \phi_2(x) \]

\[ \hat{y}_\lambda = w_1 \Phi_1 + w_2 \Phi_2 \]
Regularization path

**Samples**
\((x_1, y_1) \quad (x_2, y_2)\)

**Basis functions**
\(\phi_1, \phi_2\)

**Regularization coefficient** \(\lambda\)
\(\hat{f}_\lambda = w_1 \phi_1(x) + w_2 \phi_2(x)\)
\(\hat{y}_\lambda = w_1 \Phi_1 + w_2 \Phi_2\)

**Path**
\(\{\hat{y}_\lambda\}_{0 \leq \lambda < \infty}\)
Regularization path

samples
$(x_1, y_1)$ $(x_2, y_2)$

basis functions
$\phi_1, \phi_2$

reg. coefficient $\lambda$

$\hat{f}_\lambda = w_1 \phi_1(x) + w_2 \phi_2(x)$
$\hat{y}_\lambda = w_1 \phi_1 + w_2 \phi_2$

path
$\{\hat{y}_\lambda\}_{0 \leq \lambda < \infty}$
Regularization path

\[ f_\lambda(x_2) \]

\[ f_\lambda(x_1) \]

\[ y \]

\[ y_\lambda \]

\[ \Phi_1 \]

\[ \Phi_2 \]

Linear parts
Regularization path

\[
\begin{align*}
(y_1) &= (x_1) + \beta y_2 \\
(y_2) &= f(x_2) \\
\hat{f}_\lambda(x_2) &= \Phi_1 \\
\hat{f}_\lambda(x_1) &= \Phi_2 \\
\end{align*}
\]

linear parts
constant signs

\[
\begin{align*}
\hat{y}_\lambda &= \text{equi-gradient descent} \\
\end{align*}
\]
Regularization path

\[ y_1 = \Phi_2 \]

\[ y_2 = \Phi_1 \]

linear parts
constant signs
direction = LS/residual

\[ \hat{f}_\lambda(x_2) \]

\[ (x_1) \]

\[ (x_2) \]

\[ \hat{f}_\lambda(x_1) \]

\[ y \]

\[ y_1 \]

\[ y_2 \]
Regularization path

\[ (x_2) \]
\[ y_2 \]
\[ \hat{f}_\lambda(x_2) \]
\[ \Phi_1 \]
\[ \Phi_2 \]
\[ (x_1) \]
\[ \hat{f}_\lambda(x_1) \]
\[ y_1 \]
\[ y \]

linear parts
constant signs
direction = LS/residual
break points
Regularization path

- Linear parts
- Constant signs
- Direction = LS/residual
- Break points
- New equi-gradient $\phi$

Equi-gradient descent
Equi-gradient TD(\(\lambda\))
Experiments
Conclusion
Regularization path

\[ \hat{f}_\lambda(x_2) \]

- **Linear parts**
- **Constant signs**
- **Direction** = LS/residual
- **Break points**
- New equi-gradient \( \phi \)
- Weight \( \rightarrow 0 \)
Regularization path
Regularization path
Regularization path
Regularization path
Regularization path
Regularization path
Regularization path
Regularization path
Regularization path
Regularization path
Regularization path
Regularization path
Regularization path
Regularization path
Regularization path
Complexity

Each step (activation/deactivation) is $O(|D| + |A|^2)$

- linear in dictionary size
- quadratic in number of active basis functions
Reinforcement Learning

- TD(λ)
- continuous state space
- discrete time
- updates after each trajectory
Radial Basis Functions Network

\[ \hat{V}(x) = \sum w_i \phi_i(x) \]
Radial Basis Functions Network

\[ \hat{V}(x) = \sum w_i \phi_i(x) \]
Update

\[ \hat{V}(x) = \sum (w_i + \Delta w_i) \phi_i(x) \]
Update

\[ \hat{V}(x) = \sum w_i \phi_i(x) + \sum \Delta w_i \phi_i(x) \]
Independant regression

temporal differences
Gradient descent on fixed basis network

temporal differences
Equi-gradient descent on extended network

temporal differences
Temporal regularization
Temporal regularization

Continuously add basis functions?!
Temporal regularization

*Continuously add basis functions*?!  
- Put a preference on existing basis functions
Temporal regularization

**Continuously add basis functions?!**

- Put a preference on existing basis functions
- Remove zero-weighted basis functions
Temporal regularization

**Continuously add basis functions?!**

- Put a preference on existing basis functions
  \[ \phi_i \leftarrow \rho \phi_i \]
- Remove zero-weighted basis functions
Temporal regularization

**Continuously add basis functions?!**

- Put a preference on existing basis functions
  \[ \phi_i \leftarrow \rho \phi_i \Rightarrow w_i \leftarrow \frac{1}{\rho} w_i \]
- Remove zero-weighted basis functions
Preliminary simple experiments
Preliminary simple experiments

- Inverted pendulum
- Gaussian basis functions on normalized state space
- Updates after each episode
- Stopping EG descents at $|\hat{y}|^2 = 70\%|y|^2$
Experiments: preference

The graph illustrates the cumulative rewards and active features over episodes. The x-axis represents the episode number, ranging from 0 to 300, while the y-axes show the cumulative rewards and active features. The lines correspond to different values, with each line labeled for distinction.
Experiments: multi-kernels
Experiments: symmetry

![Graph showing cumulated rewards and active features over episodes.](image)

- **Cumulated Rewards**
  - No symmetry: Red line
  - Symmetry: Green dotted line

- **Active Features**
  - Episode # 0 to 300

The graph compares the performance of two scenarios: one without symmetry and one with symmetry. The cumulated rewards increase with each episode, and the number of active features stabilizes over time.
Experiments: comparison

![Comparison Graph](image-url)

- GPTD
- TD
- EGTD

[Graph Description]

Introduction
Equi-gradient descent
Equi-gradient TD(λ)
Experiments
Conclusion
Experiments: comparison
<table>
<thead>
<tr>
<th>Introduction</th>
<th>Equi-gradient descent</th>
<th>Equi-gradient TD((\lambda))</th>
<th>Experiments</th>
<th>Conclusion</th>
</tr>
</thead>
</table>

**Conclusion**
Conclusion

**Summary:**
Conclusion

**Summary:**

- **efficient** and **easy** way to select basis functions in TD(\(\lambda\))
Summary:
• efficient and easy way to select basis functions in TD(\(\lambda\))
• robust, no unintuitive parameters
Conclusion

Summary:
- **efficient** and **easy** way to select basis functions in TD($\lambda$)
- **robust**, no unintuitive parameters

Perspectives:
Conclusion

**Summary:**

- **efficient** and **easy** way to select basis functions in TD($\lambda$)
- **robust**, no unintuitive parameters

**Perspectives:**

- experiments on other problems
Conclusion

**Summary:**
- **efficient** and **easy** way to select basis functions in TD(λ)
- **robust**, no unintuitive parameters

**Perspectives:**
- experiments on other problems
- automatically build basis function dictionary based on topology, TD variance, . . . (wavelets, low-dimensional projections, . . .)
Take home message

- Feature selection is easy!
- Basic use of it in RL $\rightarrow$ efficient & easy-to-tune TD($\lambda$)