Computational Learning Theory **Fall Semester, 1999/00**

Lecture 6: November 25

Lecturer: Yishay Mansour *Scribe: Adi Akavia, Sivan Sabato, Moti Gindi*

6.1 Finding the Optimal Policy: Value Iteration

In this section we present the Value Iteration algorithm (also referred to as VI) for computing an ϵ-optimal policy[1] for a discounted infinite horizon problem.
In Lecture 5 we showed that the Optimality Equations for discounted infinite horizon problems are:

$$v(s) = \max_{a \in A_s}\{r(s,a) + \lambda \sum_{j \in S} p(j \mid s,a)v(j)\}$$

We also defined the non-linear operator L:

$$L\vec{v} = \max_{d \in \Pi^{MD}}\{\vec{r}_d + \lambda P_d \vec{v}\}$$

For which it was shown, that for any starting point $\vec{v}_0$, the series $\{\vec{v}_n\}$ defined by $\vec{v}_{n+1} = L\vec{v}_n$, converges to the optimal return value $\vec{v}^*_\lambda$.
The idea of VI is to use these results to compute a solution of the Optimality Equations. The VI algorithm finds a Markovian stationary policy that is ϵ-optimal.

6.1.1 The Value Iteration Algorithm

Input: MDP and parameters ϵ and λ

1. Choose an initial return value function $\vec{v}_0$ (by choosing a number for each $s \in S$).
2. $n \leftarrow 0$.
3. Assign the next return value function:

$$\forall s \in S,\ v_{n+1}(s) \leftarrow \max_{a \in A_s}\{r(s,a) + \lambda \sum_{j \in S} p(j \mid s,a)v_n(j)\}$$

4. $n \leftarrow n+1$

[1]An ϵ-optimal policy is a policy with a return value ϵ-close to the return value of the optimal policy.

5. If $\|\vec{v}_{n+1} - \vec{v}_n\| < \epsilon \cdot \frac{1-\lambda}{2\lambda}$, stop,
   Else return to (3).

6. Choose the output policy such that:

$$\pi_\epsilon(s) \in argmax_{a \in A_s}\{r(s,a) + \lambda \sum_{j \in S} p(j \mid s,a) v_{n+1}(j)\}$$

6.1.2 Correctness of Value Iteration Algorithm

In the following theorem we show that the algorithm finds an ϵ-optimal policy in a finite number of steps. Note that the selected policy might in fact be the optimal policy, but we have no way of knowing that in advance.

Theorem 6.1 *For the series $\{\vec{v}_n\}$ and the policy π_ϵ computed by VI the following holds:*

1. $\lim_{n\to\infty} v_n = v^*_\lambda$
2. $\exists N, \forall n > N, \|v_{n+1} - v_n\| < \epsilon \cdot \frac{1-\lambda}{2\lambda}$
3. *The policy π_ϵ is ϵ-optimal*
4. *If $\|v_{n+1} - v_n\| < \epsilon \cdot \frac{1-\lambda}{2\lambda}$ then $\|v_{n+1} - v^*_\lambda\| < \frac{\epsilon}{2}$*

Proof: Parts (1) and (2) follow directly from the properties of the series $v_{n+1} = Lv_n$, that were shown in Lecture 5.
For part (3) we assume that $\|v_{n+1} - v_n\| < \epsilon \cdot \frac{1-\lambda}{2\lambda}$, as is the case when the algorithm stops, and show that $\|v^{\pi_\epsilon}_\lambda - v^*_\lambda\| < \epsilon$, which would make the policy π_ϵ ϵ-optimal.

$$\|v^{\pi_\epsilon}_\lambda - v^*_\lambda\| < \|v^{\pi_\epsilon}_\lambda - v_{n+1}\| + \|v_{n+1} - v^*_\lambda\| \tag{6.1}$$

We now bound each part of the sum individually:

$$\begin{aligned} \|v^{\pi_\epsilon}_\lambda - v_{n+1}\| &= \|L_{\pi_\epsilon} v^{\pi_\epsilon}_\lambda - v_{n+1}\| \qquad (\textit{because } v^{\pi_\epsilon}_\lambda \textit{ is the fixed point of} L_{\pi_\epsilon}) \\ &\leq \|L_{\pi_\epsilon} v^{\pi_\epsilon}_\lambda - Lv_{n+1}\| + \|Lv_{n+1} - v_{n+1}\| \end{aligned}$$

Since π_ϵ is maximal over the actions using v_{n+1}, it implies that $L_{\pi_\epsilon} v_{n+1} = Lv_{n+1}$ and we conclude that:

$$\begin{aligned} \|v^{\pi_\epsilon}_\lambda - v_{n+1}\| &\leq \|L_{\pi_\epsilon} v^{\pi_\epsilon}_\lambda - L_{\pi_\epsilon} v_{n+1}\| + \|Lv_{n+1} - Lv_n\| \\ &\leq \lambda\|v^{\pi_\epsilon}_\lambda - v_{n+1}\| + \lambda\|v_{n+1} - v_n\| \end{aligned}$$

From the inequality it follows:

$$\|v^{\pi_\epsilon}_\lambda - v_{n+1}\| \leq \frac{\lambda}{1-\lambda}\|v_{n+1} - v_n\|$$

For the second part of the sum we derive similarly that:

$$\|v_{n+1} - v_\lambda^*\| \leq \frac{\lambda}{1-\lambda}\|v_{n+1} - v_n\|$$

From which part (4) of the theorem also follows.
Returning to inequality 6.1, it follows:

$$\|v_\lambda^{\pi_\epsilon} - v_\lambda^*\| \leq \frac{2\lambda}{1-\lambda}\|v_{n+1} - v_n\| < \epsilon$$

Therefore the selected policy π_ϵ is ϵ-optimal. □

6.1.3 Convergence of Value Iteration Algorithm

In this section we show that the convergence to the optimal policy of VI algorithm is monotonic, and that the convergence is exponentially fast in the parameter λ.

Claim 6.2 *Monotonicity of convergence:*

1. *if* $v \geq u$ *then* $Lv \geq Lu$
2. *if* $Lv_n \geq v_n$ *then* $\forall m \geq 0,\ v_{n+m+1} \geq v_{n+m}$

Proof: For part (1), let

$$\delta \in argmax_{\pi \in \Pi^{MD}}\{r_\pi + \lambda P_\pi u\}$$

Since $P_\delta \geq 0$, it follows that $P_\delta v \geq P_\delta u$. Therefore:

$$Lu = r_\delta + \lambda P_\delta u \leq r_\delta + \lambda P_\delta v \leq \max_{d \in \Pi^{MD}}\{r_d + \lambda P_d v\} = Lv$$

This proves part (1).
From part (1) it follows by induction that if $v \geq u$, then $\forall m \geq 1,\ L^m v \geq L^m u$.
To prove part (2):

$$v_{n+m+1} = L^m L v_n \geq L^m v_n = v_{n+m}$$

□

From the above claim it follows that if $Lv_0 \geq v_0$ (or $Lv_0 \leq v_0$), VI will converge monotonically, which means that running more iterations will always lead to a better policy. If all rewards in a specific MDP are non-negative (or non-positive), we can choose $v_0 = \vec{0}$, which assures the condition is met.

Theorem 6.3 *Let v_n be the sequence of values computed by the VI algorithm.*

1. $\forall n$, $\|v_n - v_\lambda^*\| \leq \frac{\lambda^n}{1-\lambda}\|v_1 - v_0\|$

2. $\|v_\lambda^{\pi_n} - v_\lambda^*\| \leq \frac{2\lambda^n}{1-\lambda}\|v_1 - v_0\|$, *where π_n is the policy that is defined by v_n.*

Proof: Part (1): We will use the result of part (4) in theorem 6.1:

$$If \quad \|v_n - v_{n-1}\| < \epsilon \cdot \frac{1-\lambda}{\lambda} \quad then \quad \|v_n - v_\lambda^*\| < \epsilon$$

To use it we bound:

$$\begin{aligned} \|v_n - v_{n-1}\| &= \|L^{n-1}v_1 - L^{n-1}v_0\| \\ &\leq \lambda^{n-1}\|v_1 - v_0\| \quad (\textit{because } L \textit{ is contracting}) \end{aligned}$$

Let $\epsilon = \frac{\lambda^n}{1-\lambda}\|v_1 - v_0\|$, so that we can use theorem 6.1 to conclude that:

$$\|v_n - v_\lambda^*\| \leq \frac{\lambda^n}{1-\lambda}\|v_1 - v_0\|$$

Which proves part (1).
To prove part (2) we bound:

$$\|v_\lambda^{\pi_n} - v_\lambda^*\| \leq \|v_\lambda^{\pi_n} - v_n\| + \|v_n - v_\lambda^*\|$$

The following bounds derive (a) from the same result of theorem 6.1 and (b) from part (1) of this theorem, respectively:

$$\begin{aligned} \|v_\lambda^{\pi_n} - v_\lambda^*\| &\leq \overbrace{\frac{\lambda}{1-\lambda}\|v_n - v_{n-1}\|}^{(a)} + \overbrace{\frac{\lambda^n}{1-\lambda}\|v_1 - v_0\|}^{(b)} \\ &\leq \frac{2\lambda^n}{1-\lambda}\|v_1 - v_0\| \end{aligned}$$

The last inequality follows form L being a contracting operator, thus ending the proof of part (2) □

From this theorem it follows that each iteration of the VI algorithm is closer to v_λ^* by a factor of λ. As λ approaches 1, the rate of convergence decreases. The bounds we have shown can be used to determine the number of iterations needed for a specific problem.

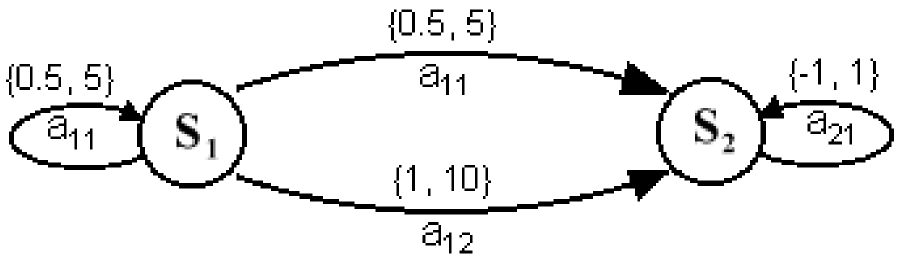

Figure 6.1: Example Diagram

6.1.4 Example: Running Value Iteration Algorithm

(Consider the MDP in figure 6.1)
Let:

$$\begin{aligned}
\lambda &= \frac{1}{2} \\
v_{n+1}(s_1) &= MAX\{5 + \lambda[\frac{1}{2}v_n(s_1) + \frac{1}{2}v_n(s_2)],\ 10 + \lambda v_n(s_2)\} \\
v_{n+1}(s_2) &= -1 + \lambda v_n(s_2)
\end{aligned}$$

Step 1: We Initialize:

$$v_0(s_1) = v_0(s_2) = -10$$

Steps 2-5:

$$\begin{aligned}
v_1(s_2) &= -1 + 0.5(-10) = -6 \\
v_1(s_1) &= MAX\{5 + 0.25(-10) + 0.25(-10),\ 10 + 0.5(-10)\} = 5 \\
\\
v_2(s_2) &= -1 + 0.5(-6) = -4 \\
v_2(s_1) &= MAX\{5 + 0.25 \cdot 5 + 0.25(-6),\ 10 + 0.5(-6)\} = 7 \\
\\
v_3(s_2) &= -1 + 0.5(-4) = -3 \\
v_3(s_1) &= MAX\{5 + 0.25 \cdot 7 + 0.25 \cdot (-3),\ 10 + 0.5 \cdot (-3)\} = 8.5
\end{aligned}$$

Step 6:

$$\begin{aligned} \pi_\epsilon(s_1) &= a_{12} \\ \pi_\epsilon(s_2) &= a_{21} \end{aligned}$$

Note that the iterated value approaches v_λ^*, which is:

$$\begin{aligned} v_\lambda^*(s_2) &= -2 \\ v_\lambda^*(s_1) &= 9 \end{aligned}$$

6.2 Policy Iteration

In this section we present the Policy Iteration algorithm (also referred to as PI) for finding the optimal policy in a discounted infinite horizon problem. As opposed to the Value Iteration algorithm, the output of PI is not an approximation of the optimal policy, but the optimal policy itself.

6.2.1 Policy Iteration Algorithm

Input: MDP, and λ

1. Initialize: $d_0 \in \Pi^{MD}$, $n \leftarrow 0$

2. *(policy evaluation)*
   Find v_n (the value of d_n) by solving the equations:

$$(I - \lambda P_{d_n})v = r_{d_n}$$

3. *(policy improvement)*
   Choose a greedy policy with respect to v_n:
   Choose the next policy, d_{n+1}, s.t.:

$$d_{n+1} \in argmax_{d \in \Pi^{MD}} \{r_d + \lambda P_d v_{d_n}\}$$

   Choose $d_{n+1} = d_n$ if possible.

4. If $d_{n+1} = d_n$ stop,
   else $n \leftarrow n + 1$, return to (2).

6.2.2 Convergence of Policy Iteration Algorithm

We shall see that when the number of states - —S—, and the number of actions - —A— are finite, there are no two non-consecutive iterations with the same policy, unless we have an optimal policy. Therefore d_n converges to the optimal policy in a finite number of steps. The key to the convergence of d_n is the monotonicity of $\{v_n\}$.

Claim 6.4 *Let v_n, v_{n+1} be the values of consecutive iterations of the above algorithm, then $v_n \leq v_{n+1} \leq v_\lambda^*$.*

Proof: Let d_{n+1} be the policy in the *policy improvement* step, then

$$\begin{aligned} r_{d_{n+1}} + \lambda P_{d_{n+1}} v_n &\geq r_{d_n} + \lambda P_{d_n} v_n \\ &= v_n \qquad (by\ definition\ of\ v_n) \end{aligned}$$

Therefore: $r_{d_{n+1}} \geq (I - \lambda P_{d_{n+1}})v_n$. Multiplying by $(I - \lambda P_{d_{n+1}})^{-1}$ we get:

$$v_{n+1} = (I - \lambda P_{d_{n+1}})^{-1} r_{d_{n+1}} \geq v_n$$

□

Note: The above claim is valid even when S or A are infinite.

Theorem 6.5 *Let S and A be finite, then the* Policy Iteration *algorithm converges to the optimal policy after at most $|A|^{|S|}$ iterations.*

Proof: Clearly, $|A|^{|S|} \geq |\Pi^{MD}|$. According to claim 6.4, in each step $v_{n+1} \geq v_n$ except for the last step in which $v_{n+1} = v_n$. Therefore no policy $\pi \in \Pi^{MD}$ can appear in two different iterations. Hence the number of iterations $\leq |\Pi^{MD}| \leq |A|^{|S|}$ □

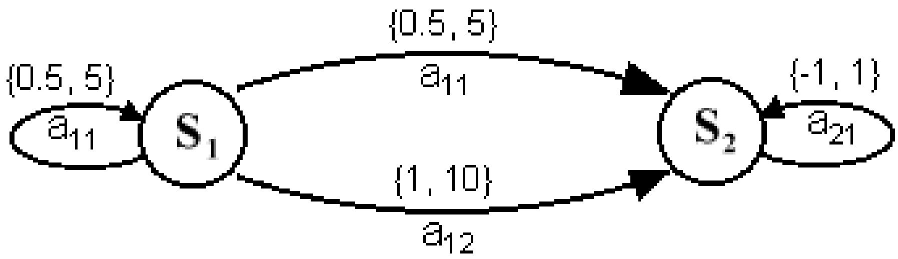

Figure 6.2: Example Diagram

6.2.3 Example: Running Policy Iteration Algorithm

(Consider the MDP in figure 6.2)
Let:

$$\lambda = 0.95$$

Step 1:

$$\begin{aligned} d_0(s_1) &= a_{12} \\ d_0(s_2) &= a_{21} \end{aligned}$$

Policy Evaluation:

$$\begin{aligned} v_0(s_1) &= 10 + 0.95 v_0(s_2) \\ v_0(s_2) &= -1 + 0.95 v_0(s_2) \\ &\Downarrow \\ v_0(s_1) = -9 \quad &, \quad v_0(s_2) = -20 \end{aligned}$$

Policy Improvement:

$$MAX\{5 + 0.475 v_0(s_1) + 0.475 v_0(s_2),\ 10 + 0.95 v_0(s_2)\} = MAX\{-0.8775,\ -9\}$$

Therefore:

$$\begin{aligned} d_1(s_1) &= a_{11} \\ d_2(s_2) &= a_{21} \end{aligned}$$

Policy Evaluation:

$$\begin{aligned} v_1(s_1) &= 5 + 0.475v_1(s_1) + 0.475v_1(s_2) \\ v_1(s_2) &= 10 + 0.95v_1(s_2) \\ &\Downarrow \end{aligned}$$

$$v_1(s_1) = -8.571 \quad , \quad v_1(s_2) = -20$$

The next policy improvement step shows that $d_2 = d_1$, and therefore the algorithm terminates and outputs d_1 as the optimal policy.

6.3 A Comparison between VI and PI Algorithms

In this section we will compare the convergence rate of the VI and PI algorithms. It will be shown that, assuming that the two algorithms begin with the same approximated value, the PI algorithm converges faster to the optimized value.

Theorem 6.6 *Let $\{u_i\}$ be the series of values created by the VI algorithm (where $u_{n+1} = Lu_n$) and let $\{v_i\}$ be the series of values created by PI algorithm. If $u_0 = v_0$, then $\forall n,\ u_n \leq v_n \leq v^*_\lambda$.*

Proof: We will use induction to prove the theorem.
Induction Basis: We assume that $u_0 = v_0$. v_0 is the return value of a specific policy, and therefore it is clearly $\leq$ the optimal return value. Therefore: $u_0 \leq v_0 \leq v^*_\lambda$.
Induction Step:

$$u_{n+1} = Lu_n = L_{p_n}u_n \quad [Where\ p_n \in argmax_{d\in\Pi^{MD}}\{r_d + \lambda P_d u_n\}]$$

From the induction hypothesis $u_n \leq v_n$, and since L_{p_n} is monotonic it follows that:

$$L_{p_n}u_n \leq L_{p_n}v_n$$

Since L is taking the maximum over all policies:

$$L_{p_n}v_n \leq Lv_n$$

We denote the policy determined by PI algorithm in iteration n as d_n and therefore:

$$Lv_n = L_{d_n}v_n$$

From the Optimality Equations we get:

$$L_{d_n}v_n \leq L_{d_n}v_\lambda^{d_n}$$

From the definition of v_{n+1} we have:

$$L_{d_n} v_\lambda^{d_n} = v_{n+1}$$

And we get $u_{n+1} \leq v_{n+1}$. In Theorem 6.4 it was proven that $v_{n+1} \leq v_\lambda^*$ and therefore $u_{n+1} \leq v_{n+1} \leq v_\lambda^*$. □

From Theorem 6.6 it follows that, assuming the same starting point, PI algorithm requires less stages then VI algorithm to converge to the optimal policy. Yet, it should be noticed that each single stage of PI requires a solution of a set of linear equations (the *policy evaluation* stage) and therefore it is computationally more expensive than a single stage of VI algorithm.

6.4 Linear Programming

6.4.1 Introduction to Linear Programming

In a general **linear-programming problem**, we wish to optimize a linear function, subject to a set of linear inequalities. We are given an $m \times n$ matrix $\mathbf{A}$, an m-vector $\vec{b}$, and an n-vector $\vec{c}$. We wish to find a vector $\vec{x}$ of n elements that maximizes the **objective function**: $\sum_{i=1}^{n} c_i x_i$ subject to the m constraints given by $\mathrm{A}\vec{x} \geq \vec{b}$, and $\forall i,\ x_i \geq 0$.
This problem can be written formally as a **linear-program** of the following structure (also called the primal linear program):
Minimize: $\vec{c}^T \vec{x}$
Subject To: $A\vec{x} \geq \vec{b}$ *and* $\vec{x} \geq \vec{0}$
Many problems can be expressed as linear programs, and for this reason much work has gone into algorithms for linear programming. Today we know some polynomial algorithms that can be used to solve this problem, and there are many software packages can solve it.
Each minimization problem can easily be transformed to a dual maximization problem of the form:
Maximize: $\vec{b}^T \vec{y}$
Subject To: $A^T \vec{y} \leq \vec{c}$ *and* $\vec{y} \geq \vec{0}$

Theorem 6.7 *(no proof)* $\vec{c}^T \hat{\vec{x}} = \vec{b}^T \hat{\vec{y}}$, *where* $\hat{\vec{x}}$ *and* $\hat{\vec{y}}$ *are the solutions of a minimization problem and its dual maximization problem.*

The above theorem suggests that we can either solve the primal or the dual linear program.

6.4.2 A Linear Programming Example

Consider the translation of the following problem to a linear programming problem:
A person undergoing a diet should take N types of vitamins. One should take a quantity of

b_i from each vitamin type. There exist M fruits. Each fruit j contains $v_{j,1}...v_{j,N}$ vitamins of each type. Each fruit j costs p_j. Assuming that it is possible to buy a fraction of a fruit, what is the cheapest combination of fruits that should be bought while keeping the diet constraints?
The following linear program describes this problem:
Minimize: $\sum_{j=1}^{M} x_j p_j$
Subject To:

- $\forall i : \sum_{j=1}^{M} x_j v_{j,i} \geq b_i$
- $\forall j : x_j \geq 0$

Where x_j would denote the quantity bought of fruit j.
The solution of this linear program gives the lowest price needed to achieve the diet constraints. Yet, this problem can be also translated to a dual maximization linear program:
Maximize: $\sum_{i=1}^{N} y_i b_i$
Subject To:

- $\forall j : \sum_{i=1}^{N} y_i v_{j,i} \leq p_j$
- $\forall i : y_i \geq 0$

The intuition behind this translation can be viewed by changing the point of view about the problem. In this case we can assume the following problem:
A vitamins company sells N types of vitamins. Each vitamin price is y_i. It is known that every person is taking a quantity of b_i from each vitamin type. The person can either buy the vitamin directly from the company or get it by eating fruits. There exist M fruits. Each fruit j contains $v_{j,1}...v_{j,N}$ vitamins of each type. Each fruit j costs p_j. The company does not want a combination of vitamins to cost more than the fruit which contains them, since then people will buy the fruit rather than the vitamins. The aim is to maximize the return of the diet..
The variable y_i in the above program denotes the price of each vitamin type i. The solution of this program will give the maximal income of the company. According to Theorem 6.7 the company's maximal income will be equal exactly to buyer's minimal expense.

6.4.3 Use of Linear Programming to Find the Optimal Policy

The general translation of the problem $x = max\{a_1, .., a_n\}$ to a linear program is as follows:
Minimize: x
Subject To: $\forall i : x \geq a_i$
It was shown that the Optimality Equations of the discounted infinite horizon problem are:

$$v(s) = \max_{a \in A_s} \{r(s, a) + \lambda \sum_{j \in S} p(j \mid s, a) v(j)\}$$

Using the above translation we would get the following linear program:
Minimize: $\sum_{j\in S}\alpha(j)v(j)$
Subject To:

- $\forall s\in S, \forall a\in A_s: v(s)\geq r(s,a)+\lambda\sum_{j\in S}p(j\mid s,a)v(j)$

  $\alpha(j)$ is any set of constants with the following characteristics:

- $\forall j: \alpha(j)>0$
- $\sum_j \alpha(j)=1$

These constants can be viewed as the probability distribution of the agent's initial state in the MDP.
Accordingly, the dual linear program would be:
Maximize: $\sum_{s\in S}\sum_{a\in A_s} r(s,a)\chi(s,a)$
Subject To:

- $\forall j\in S: \sum_{a\in A_j}\chi(s,a)=\alpha(j)+\lambda\sum_{a\in A_s}\sum_{s\in S}p(j\mid s,a)\chi(s,a)$
- $\chi(s,a)\geq 0$

Intuitively, $\chi(s,a)$ can be thought of as the probability of being in state s and performing action a, while taking into account the discount factor - λ. In the following theorem $\chi(s,a)$ is defined more formally.

Theorem 6.8 *(no proof)*

1. *For every $d\in\Pi^{MR}$, $s\in S$ and $a\in A_s$, let:*

$$\chi_d(s,a)=\sum_{j\in S}\alpha(j)\sum_{n=1}^{\infty}\lambda^{n-1}Prob[x_n=s\bigwedge y_n=a\mid x_1=j]$$

   $\chi_d(s,a)$ is a feasible solution of the dual linear maximization problem.

2. *Let $\chi_d(s,a)$ be a feasible solution of the dual linear problem. For every $s\in S$ and $a\in A_s$ define $d_x(s)$ such that:*

$$Prob[d_x(s)=a]=\frac{\chi(s,a)}{\sum_{\overline{a}\in A_s}\chi(s,\overline{a})}$$

   and then $\chi_{d_x}(s,a)=\chi(s,a)$.

Theorem 6.8 is assuring that every solution to the dual linear program can be translated to a policy with a return value equal to the maximized element in the program. Thus, the best solution of the program can be translated to a policy with the maximal possible return value. This policy will, obviously, be the best policy.