Bellman Denklemlerine Giriş:

Çok Basit Bir Problem için Bellman Denkleminin (Değer Fonksiyonu ya da Value Function, V) Yazılması:

Bellman Denklemlerine Giriş:

Çok Basit Bir Problem için Bellman Denkleminin (Değer Fonksiyonu ya da Value Function, V) Yazılması:

- **Örnek için Varsayımlar:**

Bellman Denklemlerine Giriş:

Çok Basit Bir Problem için Bellman Denkleminin (Değer Fonksiyonu ya da Value Function, V) Yazılması:

- **Örnek için Varsayımlar:**
- Zaman sonsuz ve kesikli: $t = 0, 1, 2, \ldots$.

Bellman Denklemlerine Giriş:

Çok Basit Bir Problem için Bellman Denkleminin (Değer Fonksiyonu ya da Value Function, V) Yazılması:

- **Örnek için Varsayımlar:**
- Zaman sonsuz ve kesikli: $t = 0, 1, 2, \ldots$.
- İndirgeme faktörü: $0 < \beta < 1$.

Bellman Denklemlerine Giriş:

Çok Basit Bir Problem için Bellman Denkleminin (Değer Fonksiyonu ya da Value Function, V) Yazılması:

- **Örnek için Varsayımlar:**
- Zaman sonsuz ve kesikli: $t = 0, 1, 2, \ldots$.
- İndirgeme faktörü: $0 < \beta < 1$.
- İki tüketici olsun.

Bellman Denklemlerine Giriş:

Çok Basit Bir Problem için Bellman Denkleminin (Değer Fonksiyonu ya da Value Function, V) Yazılması:

- **Örnek için Varsayımlar:**
- Zaman sonsuz ve kesikli: $t = 0, 1, 2, \ldots$.
- İndirgeme faktörü: $0 < \beta < 1$.
- İki tüketici olsun.
- Başlangıç döneminde ($t = 0$) birinci tüketici 9 birim, ikinci tüketici ise 1 birim gelire sahip olsun.

Bellman Denklemlerine Giriş:

Çok Basit Bir Problem için Bellman Denkleminin (Değer Fonksiyonu ya da Value Function, V) Yazılması:

- **Örnek için Varsayımlar:**
- Zaman sonsuz ve kesikli: $t = 0, 1, 2, \ldots$.
- İndirgeme faktörü: $0 < \beta < 1$.
- İki tüketici olsun.
- Başlangıç döneminde ($t = 0$) birinci tüketici 9 birim, ikinci tüketici ise 1 birim gelire sahip olsun.
- Tüketiciler $0 < \pi < 1$ olasılıkla t dönemindeki gelirlerini bir sonraki dönem olan $t + 1$'de koruyorlar.

Bellman Denklemlerine Giriş:

Çok Basit Bir Problem için Bellman Denkleminin (Değer Fonksiyonu ya da Value Function, V) Yazılması:

- **Örnek için Varsayımlar:**
- Zaman sonsuz ve kesikli: $t = 0, 1, 2, \ldots$.
- İndirgeme faktörü: $0 < \beta < 1$.
- İki tüketici olsun.
- Başlangıç döneminde ($t = 0$) birinci tüketici 9 birim, ikinci tüketici ise 1 birim gelire sahip olsun.
- Tüketiciler $0 < \pi < 1$ olasılıkla t dönemindeki gelirlerini bir sonraki dönem olan $t + 1$'de koruyorlar.
- $1 - \pi$ olasılıkla t dönemindeki gelirleri $t + 1$ döneminde yer değiştirmektedir.

Bellman Denklemlerine Giriş:

Çok Basit Bir Problem için Bellman Denkleminin (Değer Fonksiyonu ya da Value Function, V) Yazılması:

- **Örnek için Varsayımlar:**
- Zaman sonsuz ve kesikli: $t = 0, 1, 2, \ldots$.
- İndirgeme faktörü: $0 < \beta < 1$.
- İki tüketici olsun.
- Başlangıç döneminde ($t = 0$) birinci tüketici 9 birim, ikinci tüketici ise 1 birim gelire sahip olsun.
- Tüketiciler $0 < \pi < 1$ olasılıkla t dönemindeki gelirlerini bir sonraki dönem olan $t + 1$'de koruyorlar.
- $1 - \pi$ olasılıkla t dönemindeki gelirleri $t + 1$ döneminde yer değiştirmektedir.
- Bakınız Ek 1.

Bellman Denklemlerine Giriş:

Çok Basit Bir Problem için Bellman Denkleminin (Değer Fonksiyonu ya da Value Function, V) Yazılması:

- **Örnek için Varsayımlar:**
- Zaman sonsuz ve kesikli: $t = 0, 1, 2, \ldots$.
- İndirgeme faktörü: $0 < \beta < 1$.
- İki tüketici olsun.
- Başlangıç döneminde ($t = 0$) birinci tüketici 9 birim, ikinci tüketici ise 1 birim gelire sahip olsun.
- Tüketiciler $0 < \pi < 1$ olasılıkla t dönemindeki gelirlerini bir sonraki dönem olan $t + 1$'de koruyorlar.
- $1 - \pi$ olasılıkla t dönemindeki gelirleri $t + 1$ döneminde yer değiştirmektedir.
- Bakınız Ek 1.
- Bu durumda Bellman denklemlerini yazabiliriz:

Bellman Denklemlerine Giriş:

Çok Basit Bir Problem için Bellman Denkleminin (Değer Fonksiyonu ya da Value Function, V) Yazılması:

- **Örnek için Varsayımlar:**
- Zaman sonsuz ve kesikli: $t = 0, 1, 2, \ldots$.
- İndirgeme faktörü: $0 < \beta < 1$.
- İki tüketici olsun.
- Başlangıç döneminde ($t = 0$) birinci tüketici 9 birim, ikinci tüketici ise 1 birim gelire sahip olsun.
- Tüketiciler $0 < \pi < 1$ olasılıkla t dönemindeki gelirlerini bir sonraki dönem olan $t + 1$'de koruyorlar.
- $1 - \pi$ olasılıkla t dönemindeki gelirleri $t + 1$ döneminde yer değiştirmektedir.
- Bakınız Ek 1.
- Bu durumda Bellman denklemlerini yazabiliriz:
- $V(9)$: Başlangıç anında 9 birim geliri olan tüketicinin bügune indirgenmiş ömür boyu beklenen geliri.

Bellman Denklemlerine Giriş:

Çok Basit Bir Problem için Bellman Denkleminin (Değer Fonksiyonu ya da Value Function, V) Yazılması:

- **Örnek için Varsayımlar:**
- Zaman sonsuz ve kesikli: $t = 0, 1, 2, \ldots$.
- İndirgeme faktörü: $0 < \beta < 1$.
- İki tüketici olsun.
- Başlangıç döneminde ($t = 0$) birinci tüketici 9 birim, ikinci tüketici ise 1 birim gelire sahip olsun.
- Tüketiciler $0 < \pi < 1$ olasılıkla t dönemindeki gelirlerini bir sonraki dönem olan $t + 1$'de koruyorlar.
- $1 - \pi$ olasılıkla t dönemindeki gelirleri $t + 1$ döneminde yer değiştirmektedir.
- Bakınız Ek 1.
- Bu durumda Bellman denklemlerini yazabiliriz:
- $V(9)$: Başlangıç anında 9 birim geliri olan tüketicinin bügune indirgenmiş ömür boyu beklenen geliri.
- $V(1)$: Başlangıç anında 1 birim geliri olan tüketicinin bügune indirgenmiş ömür boyu beklenen geliri.

Bellman Denklemlerine Giriş:

Çok Basit Bir Problem için Bellman Denkleminin (Değer Fonksiyonu ya da Value Function, V) Yazılması:

- **Örnek için Varsayımlar:**
- Zaman sonsuz ve kesikli: $t = 0, 1, 2, \ldots$.
- İndirgeme faktörü: $0 < \beta < 1$.
- İki tüketici olsun.
- Başlangıç döneminde ($t = 0$) birinci tüketici 9 birim, ikinci tüketici ise 1 birim gelire sahip olsun.
- Tüketiciler $0 < \pi < 1$ olasılıkla t dönemindeki gelirlerini bir sonraki dönem olan $t + 1$'de koruyorlar.
- $1 - \pi$ olasılıkla t dönemindeki gelirleri $t + 1$ döneminde yer değiştirmektedir.
- Bakınız Ek 1.
- Bu durumda Bellman denklemlerini yazabiliriz:
- $V(9)$: Başlangıç anında 9 birim geliri olan tüketicinin bügune indirgenmiş ömür boyu beklenen geliri.
- $V(1)$: Başlangıç anında 1 birim geliri olan tüketicinin bügune indirgenmiş ömür boyu beklenen geliri.
-

$$V(9) = 9 + \beta\left((\pi)V(9) + (1 - \pi)V(1)\right)$$

$$V(1) = 1 + \beta\left((1 - \pi)V(9) + \pi V(1)\right)$$

Bellman Denklemlerine Giriş:

Çok Basit Bir Problem için Bellman Denkleminin (Değer Fonksiyonu ya da Value Function, V) Yazılması:

- **Örnek için Varsayımlar:**
- Zaman sonsuz ve kesikli: $t = 0, 1, 2, \ldots$.
- İndirgeme faktörü: $0 < \beta < 1$.
- İki tüketici olsun.
- Başlangıç döneminde ($t = 0$) birinci tüketici 9 birim, ikinci tüketici ise 1 birim gelire sahip olsun.
- Tüketiciler $0 < \pi < 1$ olasılıkla t dönemindeki gelirlerini bir sonraki dönem olan $t + 1$'de koruyorlar.
- $1 - \pi$ olasılıkla t dönemindeki gelirleri $t + 1$ döneminde yer değiştirmektedir.
- Bakınız Ek 1.
- Bu durumda Bellman denklemlerini yazabiliriz:
- $V(9)$: Başlangıç anında 9 birim geliri olan tüketicinin bügune indirgenmiş ömür boyu beklenen geliri.
- $V(1)$: Başlangıç anında 1 birim geliri olan tüketicinin bügune indirgenmiş ömür boyu beklenen geliri.
-

$$V(9) = 9 + \beta\left((\pi)V(9) + (1 - \pi)V(1)\right)$$

$$V(1) = 1 + \beta\left((1 - \pi)V(9) + \pi V(1)\right)$$

- İki bilinmeyenli iki denklem çözülerek $V(9)$ ve $V(1)$ elde edilir.

Bellman Denklemlerine Giriş:

Çok Basit Bir Problem için Bellman Denkleminin (Değer Fonksiyonu ya da Value Function, V) Yazılması:

- **Örnek için Varsayımlar:**
- Zaman sonsuz ve kesikli: $t = 0, 1, 2, \ldots$.
- İndirgeme faktörü: $0 < \beta < 1$.
- İki tüketici olsun.
- Başlangıç döneminde ($t = 0$) birinci tüketici 9 birim, ikinci tüketici ise 1 birim gelire sahip olsun.
- Tüketiciler $0 < \pi < 1$ olasılıkla t dönemindeki gelirlerini bir sonraki dönem olan $t + 1$'de koruyorlar.
- $1 - \pi$ olasılıkla t dönemindeki gelirleri $t + 1$ döneminde yer değiştirmektedir.
- Bakınız Ek 1.
- Bu durumda Bellman denklemlerini yazabiliriz:
- $V(9)$: Başlangıç anında 9 birim geliri olan tüketicinin bügune indirgenmiş ömür boyu beklenen geliri.
- $V(1)$: Başlangıç anında 1 birim geliri olan tüketicinin bügune indirgenmiş ömür boyu beklenen geliri.
-

$$V(9) = 9 + \beta\left((\pi)V(9) + (1 - \pi)V(1)\right)$$

$$V(1) = 1 + \beta\left((1 - \pi)V(9) + \pi V(1)\right)$$

- İki bilinmeyenli iki denklem çözülerek $V(9)$ ve $V(1)$ elde edilir.
- Bu yazıma "Recursive Formulation" adı da verilir.

Dinamik Programlama

Dinamik programlama, dinamik optimizasyon yöntemlerinden biridir:

Dinamik Programlama

Dinamik programlama, dinamik optimizasyon yöntemlerinden biridir:

- Neo-Klasik büyüme modeli için SPP'yi (Social Planner Problemi) **Sequential Formda** yazalım (basitlik açısından $\delta = 1$ olsun):

Dinamik Programlama

Dinamik programlama, dinamik optimizasyon yöntemlerinden biridir:

- Neo-Klasik büyüme modeli için SPP'yi (Social Planner Problemi) **Sequential Formda** yazalım (basitlik açısından $\delta = 1$ olsun):
-

$$\max_{c_t, k_{t+1}} \sum_{t=0}^{\infty} \beta^t U(c_t)$$

Dinamik Programlama

Dinamik programlama, dinamik optimizasyon yöntemlerinden biridir:

- Neo-Klasik büyüme modeli için SPP'yi (Social Planner Problemi) **Sequential Formda** yazalım (basitlik açısından $\delta = 1$ olsun):
- $$\max_{c_t, k_{t+1}} \sum_{t=0}^{\infty} \beta^t U(c_t)$$
- s.t.
$$c_t + k_{t+1} = f(k_t)$$

Dinamik Programlama

Dinamik programlama, dinamik optimizasyon yöntemlerinden biridir:

- Neo-Klasik büyüme modeli için SPP'yi (Social Planner Problemi) **Sequential Formda** yazalım (basitlik açısından $\delta = 1$ olsun):
- $$\max_{c_t, k_{t+1}} \sum_{t=0}^{\infty} \beta^t U(c_t)$$
- s.t.
$$c_t + k_{t+1} = f(k_t)$$
- $$k_0 \geq 0 \ \text{(veri)}$$

Dinamik Programlama

Dinamik programlama, dinamik optimizasyon yöntemlerinden biridir:

- Neo-Klasik büyüme modeli için SPP'yi (Social Planner Problemi) **Sequential Formda** yazalım (basitlik açısından $\delta = 1$ olsun):
- $$\max_{c_t, k_{t+1}} \sum_{t=0}^{\infty} \beta^t U(c_t)$$
- s.t.
$$c_t + k_{t+1} = f(k_t)$$
- $$k_0 \geq 0 \ \text{(veri)}$$
- $$c_t, k_{t+1} \geq 0$$

Dinamik Programlama

Dinamik programlama, dinamik optimizasyon yöntemlerinden biridir:

- Neo-Klasik büyüme modeli için SPP'yi (Social Planner Problemi) **Sequential Formda** yazalım (basitlik açısından $\delta = 1$ olsun):
- $$\max_{c_t, k_{t+1}} \sum_{t=0}^{\infty} \beta^t U(c_t)$$
- s.t.
$$c_t + k_{t+1} = f(k_t)$$
- $$k_0 \geq 0 \ \text{(veri)}$$
- $$c_t, k_{t+1} \geq 0$$
- Problemi sadece k cinsinden de **Sequential Formda** yazabiliriz:

Dinamik Programlama

Dinamik programlama, dinamik optimizasyon yöntemlerinden biridir:

- Neo-Klasik büyüme modeli için SPP'yi (Social Planner Problemi) **Sequential Formda** yazalım (basitlik açısından $\delta = 1$ olsun):
- $$\max_{c_t, k_{t+1}} \sum_{t=0}^{\infty} \beta^t U(c_t)$$
- s.t.
$$c_t + k_{t+1} = f(k_t)$$
- $$k_0 \geq 0 \ \text{(veri)}$$
- $$c_t, k_{t+1} \geq 0$$
- Problemi sadece k cinsinden de **Sequential Formda** yazabiliriz:
- $$\max_{k_{t+1}} \sum_{t=0}^{\infty} \beta^t U(f(k_t) - k_{t+1})$$

s.t.

Dinamik Programlama

Dinamik programlama, dinamik optimizasyon yöntemlerinden biridir:

- Neo-Klasik büyüme modeli için SPP'yi (Social Planner Problemi) **Sequential Formda** yazalım (basitlik açısından $\delta = 1$ olsun):
- $$\max_{c_t, k_{t+1}} \sum_{t=0}^{\infty} \beta^t U(c_t)$$
- s.t.
- $$c_t + k_{t+1} = f(k_t)$$
- $$k_0 \geq 0 \ \text{(veri)}$$
- $$c_t, k_{t+1} \geq 0$$
- Problemi sadece k cinsinden de **Sequential Formda** yazabiliriz:
- $$\max_{k_{t+1}} \sum_{t=0}^{\infty} \beta^t U(f(k_t) - k_{t+1})$$

  s.t.
- $$0 \leq k_{t+1} \leq f(k_t)$$

Dinamik Programlama

Dinamik programlama, dinamik optimizasyon yöntemlerinden biridir:

- Neo-Klasik büyüme modeli için SPP'yi (Social Planner Problemi) **Sequential Formda** yazalım (basitlik açısından $\delta = 1$ olsun):
- $$\max_{c_t, k_{t+1}} \sum_{t=0}^{\infty} \beta^t U(c_t)$$
- s.t.
- $$c_t + k_{t+1} = f(k_t)$$
- $$k_0 \geq 0 \ \text{(veri)}$$
- $$c_t, k_{t+1} \geq 0$$
- Problemi sadece k cinsinden de **Sequential Formda** yazabiliriz:
- $$\max_{k_{t+1}} \sum_{t=0}^{\infty} \beta^t U(f(k_t) - k_{t+1})$$

  s.t.
- $$0 \leq k_{t+1} \leq f(k_t)$$
- $$k_0 \geq 0 \ \text{(veri)}$$

Dinamik Programlama

Neo-Klasik Büyüme Neden Recursive Formdadır?

Dinamik Programlama

Neo-Klasik Büyüme Neden Recursive Formdadır?

- Çünkü her dönem başlangıç "k" değeri de değişse de hep aynı yukarıda tanımlanan problem çözülür:

Dinamik Programlama

Neo-Klasik Büyüme Neden Recursive Formdadır?

- Çünkü her dönem başlangıç "k" değeri de değişse de hep aynı yukarıda tanımlanan problem çözülür:
- 0. dönemde k_0 veri iken faydayı maksimize eden k_1 seçilir.

Dinamik Programlama

Neo-Klasik Büyüme Neden Recursive Formdadır?

- Çünkü her dönem başlangıç "k" değeri de değişse de hep aynı yukarıda tanımlanan problem çözülür:
- 0. dönemde k_0 veri iken faydayı maksimize eden k_1 seçilir.
- Daha sonra 1. dönemde k_1 veri iken aynı problemde faydayı maksimizde eden k_2 seçilir.

Dinamik Programlama

Neo-Klasik Büyüme Neden Recursive Formdadır?

- Çünkü her dönem başlangıç "k" değeri de değişse de hep aynı yukarıda tanımlanan problem çözülür:
- 0. dönemde k_0 veri iken faydayı maksimize eden k_1 seçilir.
- Daha sonra 1. dönemde k_1 veri iken aynı problemde faydayı maksimizde eden k_2 seçilir.
- Daha sonra 2. dönemde k_2 veri iken aynı problemde faydayı maksimizde eden k_3 seçilir.

Dinamik Programlama

Neo-Klasik Büyüme Neden Recursive Formdadır?

- Çünkü her dönem başlangıç "k" değeri de değişse de hep aynı yukarıda tanımlanan problem çözülür:
- 0. dönemde k_0 veri iken faydayı maksimize eden k_1 seçilir.
- Daha sonra 1. dönemde k_1 veri iken aynı problemde faydayı maksimizde eden k_2 seçilir.
- Daha sonra 2. dönemde k_2 veri iken aynı problemde faydayı maksimizde eden k_3 seçilir.
- Bu mantık genellenirse, her dönem sadece başlangıç değeri değişen aynı problemi çözdüğümüzden bu problem Recursive formda tanımlanabilir.

Dinamik Programlama

Bu problemin "Bellman denklemini" ya da "Recursive formunu" (zaman indislerinden arındırıp) şöyle yazabiliriz:

Dinamik Programlama

Bu problemin "Bellman denklemini" ya da "Recursive formunu" (zaman indislerinden arındırıp) şöyle yazabiliriz:

- $$V(k) = \max_{0 \leq k' \leq f(k)} U(f(k) - k') + \beta V(k')$$

Dinamik Programlama

Bu problemin "Bellman denklemini" ya da "Recursive formunu" (zaman indislerinden arındırıp) şöyle yazabiliriz:

- $$V(k) = \max_{0 \leq k' \leq f(k)} U(f(k) - k') + \beta V(k')$$

- $V(k)$, cari dönemde elinde k sermaye stoku bulunduran bir kişinin ömür boyunca elde edeceği bugüne indirgenmiş (optimal) faydayı temsil etmektedir.

Dinamik Programlama

Bu problemin "Bellman denklemini" ya da "Recursive formunu" (zaman indislerinden arındırıp) şöyle yazabiliriz:

- $$V(k) = \max_{0 \leq k' \leq f(k)} U(f(k) - k') + \beta V(k')$$

- $V(k)$, cari dönemde elinde k sermaye stoku bulunduran bir kişinin ömür boyunca elde edeceği bugüne indirgenmiş (optimal) faydayı temsil etmektedir.
- $V(k)$ denklemi Bellman Denklemi ya da Değer Fonksiyonu (Value Function) ya da Recursive Formulation olarak adlandırılmaktadır.

Dinamik Programlama

Bu problemin "Bellman denklemini" ya da "Recursive formunu" (zaman indislerinden arındırıp) şöyle yazabiliriz:

- $$V(k) = \max_{0 \leq k' \leq f(k)} U(f(k) - k') + \beta V(k')$$

- $V(k)$, cari dönemde elinde k sermaye stoku bulunduran bir kişinin ömür boyunca elde edeceği bugüne indirgenmiş (optimal) faydayı temsil etmektedir.
- $V(k)$ denklemi Bellman Denklemi ya da Değer Fonksiyonu (Value Function) ya da Recursive Formulation olarak adlandırılmaktadır.
- k cari dönemindeki sermaye stokunu göstermekte olup durum (state) degişkenidir. Cari dönemdeki tüm bilgileri durum değişkeni yani k özetler.

Dinamik Programlama

Bu problemin "Bellman denklemini" ya da "Recursive formunu" (zaman indislerinden arındırıp) şöyle yazabiliriz:

- $$V(k) = \max_{0 \leq k' \leq f(k)} U(f(k) - k') + \beta V(k')$$

- $V(k)$, cari dönemde elinde k sermaye stoku bulunduran bir kişinin ömür boyunca elde edeceği bugüne indirgenmiş (optimal) faydayı temsil etmektedir.
- $V(k)$ denklemi Bellman Denklemi ya da Değer Fonksiyonu (Value Function) ya da Recursive Formulation olarak adlandırılmaktadır.
- k cari dönemindeki sermaye stokunu göstermekte olup durum (state) degişkenidir. Cari dönemdeki tüm bilgileri durum değişkeni yani k özetler.
- k' ise bir sonraki dönemdeki sermaye stokunu gösterir. Seçim (control/choice) değişkenidir.

Dinamik Programlama

Bu problemin "Bellman denklemini" ya da "Recursive formunu" (zaman indislerinden arındırıp) şöyle yazabiliriz:

- $$V(k) = \max_{0 \leq k' \leq f(k)} U(f(k) - k') + \beta V(k')$$

- $V(k)$, cari dönemde elinde k sermaye stoku bulunduran bir kişinin ömür boyunca elde edeceği bugüne indirgenmiş (optimal) faydayı temsil etmektedir.
- $V(k)$ denklemi Bellman Denklemi ya da Değer Fonksiyonu (Value Function) ya da Recursive Formulation olarak adlandırılmaktadır.
- k cari dönemindeki sermaye stokunu göstermekte olup durum (state) degişkenidir. Cari dönemdeki tüm bilgileri durum değişkeni yani k özetler.
- k' ise bir sonraki dönemdeki sermaye stokunu gösterir. Seçim (control/choice) değişkenidir.
- $g(k) = k'$ policy function (politika fonksiyonu) olarak adlandırılmaktadır.

Dinamik Programlama

Bu problemin "Bellman denklemini" ya da "Recursive formunu" (zaman indislerinden arındırıp) şöyle yazabiliriz:

- $$V(k) = \max_{0 \leq k' \leq f(k)} U(f(k) - k') + \beta V(k')$$

- $V(k)$, cari dönemde elinde k sermaye stoku bulunduran bir kişinin ömür boyunca elde edeceği bugüne indirgenmiş (optimal) faydayı temsil etmektedir.
- $V(k)$ denklemi Bellman Denklemi ya da Değer Fonksiyonu (Value Function) ya da Recursive Formulation olarak adlandırılmaktadır.
- k cari dönemindeki sermaye stokunu göstermekte olup durum (state) degişkenidir. Cari dönemdeki tüm bilgileri durum değişkeni yani k özetler.
- k' ise bir sonraki dönemdeki sermaye stokunu gösterir. Seçim (control/choice) değişkenidir.
- $g(k) = k'$ policy function (politika fonksiyonu) olarak adlandırılmaktadır.
- Bu fonksiyon, cari dönemde k bilindiğinde bir sonraki dönem için optimal k' miktarını verir.

Dinamik Programlama

Bu problemin "Bellman denklemini" ya da "Recursive formunu" (zaman indislerinden arındırıp) şöyle yazabiliriz:

- $$V(k) = \max_{0 \leq k' \leq f(k)} U(f(k) - k') + \beta V(k')$$

- $V(k)$, cari dönemde elinde k sermaye stoku bulunduran bir kişinin ömür boyunca elde edeceği bugüne indirgenmiş (optimal) faydayı temsil etmektedir.
- $V(k)$ denklemi Bellman Denklemi ya da Değer Fonksiyonu (Value Function) ya da Recursive Formulation olarak adlandırılmaktadır.
- k cari dönemindeki sermaye stokunu göstermekte olup durum (state) degişkenidir. Cari dönemdeki tüm bilgileri durum değişkeni yani k özetler.
- k' ise bir sonraki dönemdeki sermaye stokunu gösterir. Seçim (control/choice) değişkenidir.
- $g(k) = k'$ policy function (politika fonksiyonu) olarak adlandırılmaktadır.
- Bu fonksiyon, cari dönemde k bilindiğinde bir sonraki dönem için optimal k' miktarını verir.
- Daha genel bir ifadeyle "Recursive Formulation" şu şekilde yazılabilir:

Dinamik Programlama

Bu problemin "Bellman denklemini" ya da "Recursive formunu" (zaman indislerinden arındırıp) şöyle yazabiliriz:

- $$V(k) = \max_{0 \leq k' \leq f(k)} U(f(k) - k') + \beta V(k')$$

- $V(k)$, cari dönemde elinde k sermaye stoku bulunduran bir kişinin ömür boyunca elde edeceği bugüne indirgenmiş (optimal) faydayı temsil etmektedir.
- $V(k)$ denklemi Bellman Denklemi ya da Değer Fonksiyonu (Value Function) ya da Recursive Formulation olarak adlandırılmaktadır.
- k cari dönemindeki sermaye stokunu göstermekte olup durum (state) degişkenidir. Cari dönemdeki tüm bilgileri durum değişkeni yani k özetler.
- k' ise bir sonraki dönemdeki sermaye stokunu gösterir. Seçim (control/choice) değişkenidir.
- $g(k) = k'$ policy function (politika fonksiyonu) olarak adlandırılmaktadır.
- Bu fonksiyon, cari dönemde k bilindiğinde bir sonraki dönem için optimal k' miktarını verir.
- Daha genel bir ifadeyle "Recursive Formulation" şu şekilde yazılabilir:

- $$V(x) = \max_{x' \in X} U(x, x') + \beta V(x')$$

Dinamik Programlama

Bu problemin "Bellman denklemini" ya da "Recursive formunu" (zaman indislerinden arındırıp) şöyle yazabiliriz:

- $$V(k) = \max_{0 \leq k' \leq f(k)} U(f(k) - k') + \beta V(k')$$

- $V(k)$, cari dönemde elinde k sermaye stoku bulunduran bir kişinin ömür boyunca elde edeceği bugüne indirgenmiş (optimal) faydayı temsil etmektedir.
- $V(k)$ denklemi Bellman Denklemi ya da Değer Fonksiyonu (Value Function) ya da Recursive Formulation olarak adlandırılmaktadır.
- k cari dönemindeki sermaye stokunu göstermekte olup durum (state) degişkenidir. Cari dönemdeki tüm bilgileri durum değişkeni yani k özetler.
- k' ise bir sonraki dönemdeki sermaye stokunu gösterir. Seçim (control/choice) değişkenidir.
- $g(k) = k'$ policy function (politika fonksiyonu) olarak adlandırılmaktadır.
- Bu fonksiyon, cari dönemde k bilindiğinde bir sonraki dönem için optimal k' miktarını verir.
- Daha genel bir ifadeyle "Recursive Formulation" şu şekilde yazılabilir:

- $$V(x) = \max_{x' \in X} U(x, x') + \beta V(x')$$

- Burada x :durum (state) değişkeni; x' : kontrol (control) değişkeni ve $x' \in X$ kontrol/seçim değişkeni üzerindeki kısıttır.

Kısa ve Uzun Dönem Sonuçlarının Bulunması için Alternatif Yöntemler

Kısa Dönem Sonuçlarını Elde Etmenin Alternatif Yöntemeleri (Hepsi Bellman Denklemi/Recursive Form ile çözülür):

Kısa ve Uzun Dönem Sonuçlarının Bulunması için Alternatif Yöntemler

Kısa Dönem Sonuçlarını Elde Etmenin Alternatif Yöntemeleri (Hepsi Bellman Denklemi/Recursive Form ile çözülür):

1. Tahmin & Doğrulama metodu (Guess and Verify Method) (Çok kısıtlı durumlar için analitik çözüm)

Kısa ve Uzun Dönem Sonuçlarının Bulunması için Alternatif Yöntemler

Kısa Dönem Sonuçlarını Elde Etmenin Alternatif Yöntemeleri (Hepsi Bellman Denklemi/Recursive Form ile çözülür):

1. Tahmin & Doğrulama metodu (Guess and Verify Method) (Çok kısıtlı durumlar için analitik çözüm)
2. **Değer fonksiyonu iterasyonu** (Value Function Iteration) (nümerik çözüm)

Kısa ve Uzun Dönem Sonuçlarının Bulunması için Alternatif Yöntemler

Kısa Dönem Sonuçlarını Elde Etmenin Alternatif Yöntemeleri (Hepsi Bellman Denklemi/Recursive Form ile çözülür):

1. Tahmin & Doğrulama metodu (Guess and Verify Method) (Çok kısıtlı durumlar için analitik çözüm)
2. **Değer fonksiyonu iterasyonu** (Value Function Iteration) (nümerik çözüm)
3. Politika fonksiyonu iterasyonu (Policy function iteration/Howard's improvement algorithm) (nümerik çözüm)

Kısa ve Uzun Dönem Sonuçlarının Bulunması için Alternatif Yöntemler

Kısa Dönem Sonuçlarını Elde Etmenin Alternatif Yöntemeleri (Hepsi Bellman Denklemi/Recursive Form ile çözülür):

1. Tahmin & Doğrulama metodu (Guess and Verify Method) (Çok kısıtlı durumlar için analitik çözüm)
2. **Değer fonksiyonu iterasyonu** (Value Function Iteration) (nümerik çözüm)
3. Politika fonksiyonu iterasyonu (Policy function iteration/Howard's improvement algorithm) (nümerik çözüm)

Uzun Dönem Sonuçlarının Bulunması: Euler Denklemi ve Durağan Durum Sonuçlarını Elde Etmenin Alternatif Yöntemleri:

Kısa ve Uzun Dönem Sonuçlarının Bulunması için Alternatif Yöntemler

Kısa Dönem Sonuçlarını Elde Etmenin Alternatif Yöntemeleri (Hepsi Bellman Denklemi/Recursive Form ile çözülür):

1. Tahmin & Doğrulama metodu (Guess and Verify Method) (Çok kısıtlı durumlar için analitik çözüm)
2. **Değer fonksiyonu iterasyonu** (Value Function Iteration) (nümerik çözüm)
3. Politika fonksiyonu iterasyonu (Policy function iteration/Howard's improvement algorithm) (nümerik çözüm)

Uzun Dönem Sonuçlarının Bulunması: Euler Denklemi ve Durağan Durum Sonuçlarını Elde Etmenin Alternatif Yöntemleri:

1. Lagrange Metodu (Sequential form ile çözülür)

Kısa ve Uzun Dönem Sonuçlarının Bulunması için Alternatif Yöntemler

Kısa Dönem Sonuçlarını Elde Etmenin Alternatif Yöntemeleri (Hepsi Bellman Denklemi/Recursive Form ile çözülür):

1. Tahmin & Doğrulama metodu (Guess and Verify Method) (Çok kısıtlı durumlar için analitik çözüm)
2. **Değer fonksiyonu iterasyonu** (Value Function Iteration) (nümerik çözüm)
3. Politika fonksiyonu iterasyonu (Policy function iteration/Howard's improvement algorithm) (nümerik çözüm)

Uzun Dönem Sonuçlarının Bulunması: Euler Denklemi ve Durağan Durum Sonuçlarını Elde Etmenin Alternatif Yöntemleri:

1. Lagrange Metodu (Sequential form ile çözülür)
2. Zarf Teoremi (Envelope Theorem ya da Benveniste-Scheinkman (BS) Yöntemi) (Recursive form ile çözülür)

Zarf Teoreminin Uygulanması:

Dinamik Programlama

Zarf Teoreminin Uygulanması:

- $$V(k_t) = \max_{0 \leq k_{t+1} \leq f(k_t)} [U(g(k_t, k_{t+1})) + \beta V(k_{t+1})]$$

Dinamik Programlama

Zarf Teoreminin Uygulanması:

- $$V(k_t) = \max_{0 \leq k_{t+1} \leq f(k_t)} [U(g(k_t, k_{t+1})) + \beta V(k_{t+1})]$$
- $V(k_t)$ değeri k_{t+1}'e göre maksimize edilmiş değerdir.

Dinamik Programlama

Zarf Teoreminin Uygulanması:

- $$V(k_t) = \max_{0 \leq k_{t+1} \leq f(k_t)} [U(g(k_t, k_{t+1})) + \beta V(k_{t+1})]$$
- $V(k_t)$ değeri k_{t+1}'e göre maksimize edilmiş değerdir.
- FOC w.r.t. k_{t+1} (zincir kuralı ile):

Dinamik Programlama

Zarf Teoreminin Uygulanması:

- $$V(k_t) = \max_{0 \leq k_{t+1} \leq f(k_t)} [U(g(k_t, k_{t+1})) + \beta V(k_{t+1})]$$
- $V(k_t)$ değeri k_{t+1}'e göre maksimize edilmiş değerdir.
- FOC w.r.t. k_{t+1} (zincir kuralı ile):
- $$\frac{\partial V(k_t)}{\partial k_{t+1}} = U'(g(k_t, k_{t+1})) \frac{\partial g(k_t, k_{t+1})}{\partial k_{t+1}} + \beta V'(k_{t+1}) = 0 \quad (*1)$$

Dinamik Programlama

Zarf Teoreminin Uygulanması:

- $$V(k_t) = \max_{0 \leq k_{t+1} \leq f(k_t)} [U(g(k_t, k_{t+1})) + \beta V(k_{t+1})]$$
- $V(k_t)$ değeri k_{t+1}'e göre maksimize edilmiş değerdir.
- FOC w.r.t. k_{t+1} (zincir kuralı ile):
- $$\frac{\partial V(k_t)}{\partial k_{t+1}} = U'(g(k_t, k_{t+1})) \frac{\partial g(k_t, k_{t+1})}{\partial k_{t+1}} + \beta V'(k_{t+1}) = 0 \quad (*1)$$
- Şimdi de $V(k_t)$ değişkeninin k_t ile nasıl değiştiğine bakalım:

Dinamik Programlama

Zarf Teoreminin Uygulanması:

- $$V(k_t) = \max_{0 \leq k_{t+1} \leq f(k_t)} [U(g(k_t, k_{t+1})) + \beta V(k_{t+1})]$$
- $V(k_t)$ değeri k_{t+1}'e göre maksimize edilmiş değerdir.
- FOC w.r.t. k_{t+1} (zincir kuralı ile):
- $$\frac{\partial V(k_t)}{\partial k_{t+1}} = U'(g(k_t, k_{t+1})) \frac{\partial g(k_t, k_{t+1})}{\partial k_{t+1}} + \beta V'(k_{t+1}) = 0 \quad (*1)$$
- Şimdi de $V(k_t)$ değişkeninin k_t ile nasıl değiştiğine bakalım:
- $$\frac{dV(k_t)}{dk_t} = \frac{\partial V(k_t)}{\partial k_t} + \frac{\partial V(k_t)}{\partial k_{t+1}} \frac{\partial k_{t+1}}{\partial k_t}$$

Dinamik Programlama

Zarf Teoreminin Uygulanması:

- $$V(k_t) = \max_{0 \leq k_{t+1} \leq f(k_t)} [U(g(k_t, k_{t+1})) + \beta V(k_{t+1})]$$
- $V(k_t)$ değeri k_{t+1}'e göre maksimize edilmiş değerdir.
- FOC w.r.t. k_{t+1} (zincir kuralı ile):
- $$\frac{\partial V(k_t)}{\partial k_{t+1}} = U'(g(k_t, k_{t+1})) \frac{\partial g(k_t, k_{t+1})}{\partial k_{t+1}} + \beta V'(k_{t+1}) = 0 \quad (*1)$$
- Şimdi de $V(k_t)$ değişkeninin k_t ile nasıl değiştiğine bakalım:
- $$\frac{dV(k_t)}{dk_t} = \frac{\partial V(k_t)}{\partial k_t} + \frac{\partial V(k_t)}{\partial k_{t+1}} \frac{\partial k_{t+1}}{\partial k_t}$$
- $\frac{\partial V(k_t)}{\partial k_{t+1}} = 0$ olur çünkü zaten $V(k_t)$ değeri k_{t+1}'e göre maksimize edilmiş değerdir.

Dinamik Programlama

Zarf Teoreminin Uygulanması:

- $$V(k_t) = \max_{0 \leq k_{t+1} \leq f(k_t)} [U(g(k_t, k_{t+1})) + \beta V(k_{t+1})]$$
- $V(k_t)$ değeri k_{t+1}'e göre maksimize edilmiş değerdir.
- FOC w.r.t. k_{t+1} (zincir kuralı ile):
- $$\frac{\partial V(k_t)}{\partial k_{t+1}} = U'(g(k_t, k_{t+1})) \frac{\partial g(k_t, k_{t+1})}{\partial k_{t+1}} + \beta V'(k_{t+1}) = 0 \quad (*1)$$
- Şimdi de $V(k_t)$ değişkeninin k_t ile nasıl değiştiğine bakalım:
- $$\frac{dV(k_t)}{dk_t} = \frac{\partial V(k_t)}{\partial k_t} + \frac{\partial V(k_t)}{\partial k_{t+1}} \frac{\partial k_{t+1}}{\partial k_t}$$
- $\frac{\partial V(k_t)}{\partial k_{t+1}} = 0$ olur çünkü zaten $V(k_t)$ değeri k_{t+1}'e göre maksimize edilmiş değerdir.
- $$\frac{dV(k_t)}{dk_t} = V'(k_t) = \frac{\partial V(k_t)}{\partial k_t} = U'(g(k_t, k_{t+1})) \frac{\partial g(k_t, k_{t+1})}{\partial k_t}$$

Dinamik Programlama

Zarf Teoreminin Uygulanması:

- $$V(k_t) = \max_{0 \leq k_{t+1} \leq f(k_t)} [U(g(k_t, k_{t+1})) + \beta V(k_{t+1})]$$
- $V(k_t)$ değeri k_{t+1}'e göre maksimize edilmiş değerdir.
- FOC w.r.t. k_{t+1} (zincir kuralı ile):
- $$\frac{\partial V(k_t)}{\partial k_{t+1}} = U'(g(k_t, k_{t+1})) \frac{\partial g(k_t, k_{t+1})}{\partial k_{t+1}} + \beta V'(k_{t+1}) = 0 \quad (*1)$$
- Şimdi de $V(k_t)$ değişkeninin k_t ile nasıl değiştiğine bakalım:
- $$\frac{dV(k_t)}{dk_t} = \frac{\partial V(k_t)}{\partial k_t} + \frac{\partial V(k_t)}{\partial k_{t+1}} \frac{\partial k_{t+1}}{\partial k_t}$$
- $\frac{\partial V(k_t)}{\partial k_{t+1}} = 0$ olur çünkü zaten $V(k_t)$ değeri k_{t+1}'e göre maksimize edilmiş değerdir.
- $$\frac{dV(k_t)}{dk_t} = V'(k_t) = \frac{\partial V(k_t)}{\partial k_t} = U'(g(k_t, k_{t+1})) \frac{\partial g(k_t, k_{t+1})}{\partial k_t}$$
- Yukarıdaki denklemi 1 dönem ilerisi için yazarsak:

$$V'(k_{t+1}) = U'(g(k_{t+1}, k_{t+2})) \frac{\partial g(k_{t+1}, k_{t+2})}{\partial k_{t+1}} \quad (*2)$$

Dinamik Programlama

Zarf Teoreminin Uygulanması:

- $$V(k_t) = \max_{0 \leq k_{t+1} \leq f(k_t)} [U(g(k_t, k_{t+1})) + \beta V(k_{t+1})]$$
- $V(k_t)$ değeri k_{t+1}'e göre maksimize edilmiş değerdir.
- FOC w.r.t. k_{t+1} (zincir kuralı ile):
- $$\frac{\partial V(k_t)}{\partial k_{t+1}} = U'(g(k_t, k_{t+1})) \frac{\partial g(k_t, k_{t+1})}{\partial k_{t+1}} + \beta V'(k_{t+1}) = 0 \quad (*1)$$
- Şimdi de $V(k_t)$ değişkeninin k_t ile nasıl değiştiğine bakalım:
- $$\frac{dV(k_t)}{dk_t} = \frac{\partial V(k_t)}{\partial k_t} + \frac{\partial V(k_t)}{\partial k_{t+1}} \frac{\partial k_{t+1}}{\partial k_t}$$
- $\frac{\partial V(k_t)}{\partial k_{t+1}} = 0$ olur çünkü zaten $V(k_t)$ değeri k_{t+1}'e göre maksimize edilmiş değerdir.
- $$\frac{dV(k_t)}{dk_t} = V'(k_t) = \frac{\partial V(k_t)}{\partial k_t} = U'(g(k_t, k_{t+1})) \frac{\partial g(k_t, k_{t+1})}{\partial k_t}$$
- Yukarıdaki denklemi 1 dönem ilerisi için yazarsak:

$$V'(k_{t+1}) = U'(g(k_{t+1}, k_{t+2})) \frac{\partial g(k_{t+1}, k_{t+2})}{\partial k_{t+1}} \quad (*2)$$

- (*1) ve (*2) denklemi birlikte çözülerek önce "Euler denklemi" ve sonra "Durağan Durum sonuçları" elde edilir.

Dinamik Programlama

Zarf Teoreminin Uygulanması:

- $$V(k_t) = \max_{0 \leq k_{t+1} \leq f(k_t)} [U(g(k_t, k_{t+1})) + \beta V(k_{t+1})]$$
- $V(k_t)$ değeri k_{t+1}'e göre maksimize edilmiş değerdir.
- FOC w.r.t. k_{t+1} (zincir kuralı ile):
- $$\frac{\partial V(k_t)}{\partial k_{t+1}} = U'(g(k_t, k_{t+1})) \frac{\partial g(k_t, k_{t+1})}{\partial k_{t+1}} + \beta V'(k_{t+1}) = 0 \quad (*1)$$
- Şimdi de $V(k_t)$ değişkeninin k_t ile nasıl değiştiğine bakalım:
- $$\frac{dV(k_t)}{dk_t} = \frac{\partial V(k_t)}{\partial k_t} + \frac{\partial V(k_t)}{\partial k_{t+1}} \frac{\partial k_{t+1}}{\partial k_t}$$
- $\frac{\partial V(k_t)}{\partial k_{t+1}} = 0$ olur çünkü zaten $V(k_t)$ değeri k_{t+1}'e göre maksimize edilmiş değerdir.
- $$\frac{dV(k_t)}{dk_t} = V'(k_t) = \frac{\partial V(k_t)}{\partial k_t} = U'(g(k_t, k_{t+1})) \frac{\partial g(k_t, k_{t+1})}{\partial k_t}$$
- Yukarıdaki denklemi 1 dönem ilerisi için yazarsak:

$$V'(k_{t+1}) = U'(g(k_{t+1}, k_{t+2})) \frac{\partial g(k_{t+1}, k_{t+2})}{\partial k_{t+1}} \quad (*2)$$

- (*1) ve (*2) denklemi birlikte çözülerek önce "Euler denklemi" ve sonra "Durağan Durum sonuçları" elde edilir.
- Zarf teoreminin Neo-Klasik Büyüme Modeline uyarlanması da bu adımlar izlenerek basit bir şekilde yapılabilir.

Dinamik Programlama

Tahmin ve Doğrulama Yöntemi:

Dinamik Programlama

Tahmin ve Doğrulama Yöntemi:

- $$\max_{c_t, k_{t+1}} \sum_{t=0}^{\infty} \beta^t log(c_t)$$

Dinamik Programlama

Tahmin ve Doğrulama Yöntemi:

- $$\max_{c_t, k_{t+1}} \sum_{t=0}^{\infty} \beta^t log(c_t)$$
- s.t.

$$k_{t+1} + c_t = A k_t^{\alpha}$$

$$k_0 > 0 \text{ veri}$$

Dinamik Programlama

Tahmin ve Doğrulama Yöntemi:

- $$\max_{c_t, k_{t+1}} \sum_{t=0}^{\infty} \beta^t log(c_t)$$
- s.t.
$$k_{t+1} + c_t = Ak_t^{\alpha}$$
$$k_0 > 0 \text{ veri}$$
- ya da
$$\max_{k_{t+1}} \sum_{t=0}^{\infty} \beta^t log(Ak_t^{\alpha} - k_{t+1})$$

Dinamik Programlama

Tahmin ve Doğrulama Yöntemi:

- $$\max_{c_t, k_{t+1}} \sum_{t=0}^{\infty} \beta^t log(c_t)$$
- s.t.
$$k_{t+1} + c_t = Ak_t^{\alpha}$$
$$k_0 > 0 \text{ veri}$$
- ya da
$$\max_{k_{t+1}} \sum_{t=0}^{\infty} \beta^t log(Ak_t^{\alpha} - k_{t+1})$$
- $$k_0 > 0 \text{ veri}$$

Dinamik Programlama

Tahmin ve Doğrulama Yöntemi:

- $$\max_{c_t, k_{t+1}} \sum_{t=0}^{\infty} \beta^t log(c_t)$$
- s.t.
$$k_{t+1} + c_t = Ak_t^{\alpha}$$
$$k_0 > 0 \text{ veri}$$
- ya da
$$\max_{k_{t+1}} \sum_{t=0}^{\infty} \beta^t log(Ak_t^{\alpha} - k_{t+1})$$
- $$k_0 > 0 \text{ veri}$$
- gibi "sequential" formda tanımlanmış bir Neo-Klasik büyüme modelini Tahmin ve Doğrulama yöntemi ile çözmek için modeli ilk önce "recursive" formda yazalım.

Dinamik Programlama

Tahmin ve Doğrulama Yöntemi:

- $$\max_{c_t, k_{t+1}} \sum_{t=0}^{\infty} \beta^t log(c_t)$$
- s.t.
$$k_{t+1} + c_t = Ak_t^{\alpha}$$
$$k_0 > 0 \text{ veri}$$
- ya da
$$\max_{k_{t+1}} \sum_{t=0}^{\infty} \beta^t log(Ak_t^{\alpha} - k_{t+1})$$
- $$k_0 > 0 \text{ veri}$$
- gibi "sequential" formda tanımlanmış bir Neo-Klasik büyüme modelini Tahmin ve Doğrulama yöntemi ile çözmek için modeli ilk önce "recursive" formda yazalım.
- Bir başka deyişle "Bellman" denklemini yazalım:

Dinamik Programlama

Tahmin ve Doğrulama Yöntemi:

- $$\max_{c_t, k_{t+1}} \sum_{t=0}^{\infty} \beta^t log(c_t)$$
- s.t.
$$k_{t+1} + c_t = Ak_t^{\alpha}$$
$$k_0 > 0 \text{ veri}$$
- ya da
$$\max_{k_{t+1}} \sum_{t=0}^{\infty} \beta^t log(Ak_t^{\alpha} - k_{t+1})$$
- $$k_0 > 0 \text{ veri}$$
- gibi "sequential" formda tanımlanmış bir Neo-Klasik büyüme modelini Tahmin ve Doğrulama yöntemi ile çözmek için modeli ilk önce "recursive" formda yazalım.
- Bir başka deyişle "Bellman" denklemini yazalım:
- $$V(k) = \max_{0 \leq k' \leq Ak^{\alpha}} \underbrace{\log(Ak^{\alpha} - k')}_{\text{bugunku fayda}} + \beta \underbrace{V(k')}_{\text{gelecekteki fayda}}$$

Dinamik Programlama

Tahmin ve Doğrulama Yöntemi:

- $$\max_{c_t, k_{t+1}} \sum_{t=0}^{\infty} \beta^t log(c_t)$$
- s.t.
$$k_{t+1} + c_t = Ak_t^{\alpha}$$
$$k_0 > 0 \text{ veri}$$
- ya da
$$\max_{k_{t+1}} \sum_{t=0}^{\infty} \beta^t log(Ak_t^{\alpha} - k_{t+1})$$
- $$k_0 > 0 \text{ veri}$$
- gibi "sequential" formda tanımlanmış bir Neo-Klasik büyüme modelini Tahmin ve Doğrulama yöntemi ile çözmek için modeli ilk önce "recursive" formda yazalım.
- Bir başka deyişle "Bellman" denklemini yazalım:
- $$V(k) = \max_{0 \leq k' \leq Ak^{\alpha}} \underbrace{\log(Ak^{\alpha} - k')}_{\substack{\text{bugunku} \\ \text{fayda}}} + \beta \underbrace{V(k')}_{\substack{\text{gelecekteki} \\ \text{fayda}}}$$
- Burada $V(k)$ cari dönemde elinde k miktarında sermaye bulunduran bir kişinin ömür boyunca elde edeceği optimal faydanın bugüne indirgenmiş halini göstermektedir.

Tahmin ve Doğrulama Yöntemi için 1.Aşama

Dinamik Programlama

Tahmin ve Doğrulama Yöntemi için 1.Aşama

- **1. aşama: Tahmin**

Tahmin ve Doğrulama Yöntemi için 1.Aşama

- **1. aşama: Tahmin**
- 1. aşama olarak $V(k)$ fonksiyonunun $V(k) = h_0 + h_1 \log k$ formunda olduğunu tahmin edelim.

Tahmin ve Doğrulama Yöntemi için 1.Aşama

- **1. aşama: Tahmin**
- 1. aşama olarak $V(k)$ fonksiyonunun $V(k) = h_0 + h_1 \log k$ formunda olduğunu tahmin edelim.
- Bu tahmin geçerli iken $V(k') = h_0 + h_1 \log k'$ denklemi de geçerli olacaktır.

Tahmin ve Doğrulama Yöntemi için 1.Aşama

- **1. aşama: Tahmin**
- 1. aşama olarak $V(k)$ fonksiyonunun $V(k) = h_0 + h_1 \log k$ formunda olduğunu tahmin edelim.
- Bu tahmin geçerli iken $V(k') = h_0 + h_1 \log k'$ denklemi de geçerli olacaktır.
- Bu durumda maksimizasyon problemi şu şekilde yazılır:

Tahmin ve Doğrulama Yöntemi için 1.Aşama

- **1. aşama: Tahmin**
- 1. aşama olarak $V(k)$ fonksiyonunun $V(k) = h_0 + h_1 \log k$ formunda olduğunu tahmin edelim.
- Bu tahmin geçerli iken $V(k') = h_0 + h_1 \log k'$ denklemi de geçerli olacaktır.
- Bu durumda maksimizasyon problemi şu şekilde yazılır:
-

$$V(k) = \max_{0 \leq k' \leq Ak^{\alpha}} \log(Ak^{\alpha} - k') + \beta V(k')$$

Tahmin ve Doğrulama Yöntemi için 1.Aşama

- **1. aşama: Tahmin**
- 1. aşama olarak $V(k)$ fonksiyonunun $V(k) = h_0 + h_1 \log k$ formunda olduğunu tahmin edelim.
- Bu tahmin geçerli iken $V(k') = h_0 + h_1 \log k'$ denklemi de geçerli olacaktır.
- Bu durumda maksimizasyon problemi şu şekilde yazılır:
-
$$V(k) = \max_{0 \leq k' \leq Ak^{\alpha}} \log(Ak^{\alpha} - k') + \beta V(k')$$
-
$$V(k) = \max_{0 \leq k' \leq Ak^{\alpha}} \log(Ak^{\alpha} - k') + \beta(h_0 + h_1 \log k') \quad (*)$$

Dinamik Programlama

Tahmin ve Doğrulama Yöntemi için 1.Aşama

- **1. aşama: Tahmin**
- 1. aşama olarak $V(k)$ fonksiyonunun $V(k) = h_0 + h_1 \log k$ formunda olduğunu tahmin edelim.
- Bu tahmin geçerli iken $V(k') = h_0 + h_1 \log k'$ denklemi de geçerli olacaktır.
- Bu durumda maksimizasyon problemi şu şekilde yazılır:
- $$V(k) = \max_{0 \leq k' \leq Ak^{\alpha}} \log(Ak^{\alpha} - k') + \beta V(k')$$
- $$V(k) = \max_{0 \leq k' \leq Ak^{\alpha}} \log(Ak^{\alpha} - k') + \beta(h_0 + h_1 \log k') \quad (*)$$
- F.O.C. k' değişkenine göre:
$$\frac{-1}{Ak^{\alpha} - k'} + \beta h_1 \frac{1}{k'} = 0$$

Dinamik Programlama

Tahmin ve Doğrulama Yöntemi için 1.Aşama

- **1. aşama: Tahmin**
- 1. aşama olarak $V(k)$ fonksiyonunun $V(k) = h_0 + h_1 \log k$ formunda olduğunu tahmin edelim.
- Bu tahmin geçerli iken $V(k') = h_0 + h_1 \log k'$ denklemi de geçerli olacaktır.
- Bu durumda maksimizasyon problemi şu şekilde yazılır:
- $$V(k) = \max_{0 \leq k' \leq Ak^{\alpha}} \log(Ak^{\alpha} - k') + \beta V(k')$$
- $$V(k) = \max_{0 \leq k' \leq Ak^{\alpha}} \log(Ak^{\alpha} - k') + \beta(h_0 + h_1 \log k') \quad (*)$$
- F.O.C. k' değişkenine göre:
$$\frac{-1}{Ak^{\alpha} - k'} + \beta h_1 \frac{1}{k'} = 0$$
- k' değişkenini çekersek:

Tahmin ve Doğrulama Yöntemi için 1.Aşama

- **1. aşama: Tahmin**
- 1. aşama olarak $V(k)$ fonksiyonunun $V(k) = h_0 + h_1 \log k$ formunda olduğunu tahmin edelim.
- Bu tahmin geçerli iken $V(k') = h_0 + h_1 \log k'$ denklemi de geçerli olacaktır.
- Bu durumda maksimizasyon problemi şu şekilde yazılır:
-
$$V(k) = \max_{0 \leq k' \leq Ak^{\alpha}} \log(Ak^{\alpha} - k') + \beta V(k')$$
-
$$V(k) = \max_{0 \leq k' \leq Ak^{\alpha}} \log(Ak^{\alpha} - k') + \beta(h_0 + h_1 \log k') \quad (*)$$
- F.O.C. k' değişkenine göre:
$$\frac{-1}{Ak^{\alpha} - k'} + \beta h_1 \frac{1}{k'} = 0$$
- k' değişkenini çekersek:
-
$$k' = \beta h_1 Ak^{\alpha} - k' \beta h_1 \Rightarrow$$

Dinamik Programlama

Tahmin ve Doğrulama Yöntemi için 1.Aşama

- **1. aşama: Tahmin**
- 1. aşama olarak $V(k)$ fonksiyonunun $V(k) = h_0 + h_1 \log k$ formunda olduğunu tahmin edelim.
- Bu tahmin geçerli iken $V(k') = h_0 + h_1 \log k'$ denklemi de geçerli olacaktır.
- Bu durumda maksimizasyon problemi şu şekilde yazılır:
- $$V(k) = \max_{0 \leq k' \leq Ak^{\alpha}} \log(Ak^{\alpha} - k') + \beta V(k')$$
- $$V(k) = \max_{0 \leq k' \leq Ak^{\alpha}} \log(Ak^{\alpha} - k') + \beta(h_0 + h_1 \log k') \quad (*)$$
- F.O.C. k' değişkenine göre:
$$\frac{-1}{Ak^{\alpha} - k'} + \beta h_1 \frac{1}{k'} = 0$$
- k' değişkenini çekersek:
- $$k' = \beta h_1 Ak^{\alpha} - k' \beta h_1 \Rightarrow$$
- $$k' = \frac{Ak^{\alpha} \beta h_1}{\beta h_1 + 1}$$

Dinamik Programlama

Tahmin ve Doğrulama Yöntemi için 1.Aşama

- **1. aşama: Tahmin**
- 1. aşama olarak $V(k)$ fonksiyonunun $V(k) = h_0 + h_1 \log k$ formunda olduğunu tahmin edelim.
- Bu tahmin geçerli iken $V(k') = h_0 + h_1 \log k'$ denklemi de geçerli olacaktır.
- Bu durumda maksimizasyon problemi şu şekilde yazılır:
- $$V(k) = \max_{0 \leq k' \leq Ak^{\alpha}} \log(Ak^{\alpha} - k') + \beta V(k')$$
- $$V(k) = \max_{0 \leq k' \leq Ak^{\alpha}} \log(Ak^{\alpha} - k') + \beta(h_0 + h_1 \log k') \quad (*)$$
- F.O.C. k' değişkenine göre:
$$\frac{-1}{Ak^{\alpha} - k'} + \beta h_1 \frac{1}{k'} = 0$$
- k' değişkenini çekersek:
- $$k' = \beta h_1 Ak^{\alpha} - k' \beta h_1 \Rightarrow$$
- $$k' = \frac{Ak^{\alpha} \beta h_1}{\beta h_1 + 1}$$
- $(*)$ denklemini, sermaye için bulduğumuz optimal sonucu ($k' = \frac{Ak^{\alpha} \beta h_1}{\beta h_1 + 1}$) ve $V(k) = h_0 + h_1 \log k$ eşitliğini kullanarak yeniden yazabiliriz:

Tahmin ve Doğrulama Yöntemi için 1.Aşama

- **1. aşama: Tahmin**
- 1. aşama olarak $V(k)$ fonksiyonunun $V(k) = h_0 + h_1 \log k$ formunda olduğunu tahmin edelim.
- Bu tahmin geçerli iken $V(k') = h_0 + h_1 \log k'$ denklemi de geçerli olacaktır.
- Bu durumda maksimizasyon problemi şu şekilde yazılır:
- $$V(k) = \max_{0 \leq k' \leq Ak^{\alpha}} \log(Ak^{\alpha} - k') + \beta V(k')$$
- $$V(k) = \max_{0 \leq k' \leq Ak^{\alpha}} \log(Ak^{\alpha} - k') + \beta(h_0 + h_1 \log k') \quad (*)$$
- F.O.C. k' değişkenine göre:
$$\frac{-1}{Ak^{\alpha} - k'} + \beta h_1 \frac{1}{k'} = 0$$
- k' değişkenini çekersek:
- $$k' = \beta h_1 Ak^{\alpha} - k' \beta h_1 \Rightarrow$$
- $$k' = \frac{Ak^{\alpha} \beta h_1}{\beta h_1 + 1}$$
- $(*)$ denklemini, sermaye için bulduğumuz optimal sonucu ($k' = \frac{Ak^{\alpha} \beta h_1}{\beta h_1 + 1}$) ve $V(k) = h_0 + h_1 \log k$ eşitliğini kullanarak yeniden yazabiliriz:
- $$h_0 + h_1 \log(k) = \log\left(Ak^{\alpha}\left(1 - \frac{\beta h_1}{\beta h_1 + 1}\right)\right) + \beta\left(h_0 + h_1 \log\left(\frac{Ak^{\alpha} \beta h_1}{\beta h_1 + 1}\right)\right)$$

Tahmin ve Doğrulama Yöntemi için 1.Aşama

Tahmin ve Doğrulama Yöntemi için 1.Aşama

- Elde edilen bu denklemde k içeren ifadeleri ve diğer tüm ifadeleri (sabitleri) ayıralım.

Tahmin ve Doğrulama Yöntemi için 1.Aşama

- Elde edilen bu denklemde k içeren ifadeleri ve diğer tüm ifadeleri (sabitleri) ayıralım.
-

$$h_0 + h_1 \log(k) = \log(A) + \alpha \log(k) + \log\left(\frac{1}{\beta h_1 + 1}\right) + \beta h_0 + \beta h_1 \log\left(\frac{A\beta h_1}{\beta h_1 + 1}\right) + \alpha\beta h_1 \log(k)$$

Tahmin ve Doğrulama Yöntemi için 1.Aşama

- Elde edilen bu denklemde k içeren ifadeleri ve diğer tüm ifadeleri (sabitleri) ayıralım.
- $$h_0 + h_1 \log(k) = \log(A) + \alpha \log(k) + \log\left(\frac{1}{\beta h_1 + 1}\right) + \beta h_0 + \beta h_1 \log\left(\frac{A\beta h_1}{\beta h_1 + 1}\right) + \alpha\beta h_1 \log(k)$$
- $$h_0 + h_1 \log(k) = \alpha \log(k) + \alpha\beta h_1 \log(k) + \underbrace{\log(A) + \log\left(\frac{1}{\beta h_1 + 1}\right) + \beta h_0 + \beta h_1 \log\left(\frac{A\beta h_1}{\beta h_1 + 1}\right)}_{M}$$

Dinamik Programlama

Tahmin ve Doğrulama Yöntemi için 1.Aşama

- Elde edilen bu denklemde k içeren ifadeleri ve diğer tüm ifadeleri (sabitleri) ayıralım.
-
$$h_0 + h_1 \log(k) = \log(A) + \alpha \log(k) + \log\left(\frac{1}{\beta h_1 + 1}\right) + \beta h_0 + \beta h_1 \log\left(\frac{A\beta h_1}{\beta h_1 + 1}\right) + \alpha\beta h_1 \log(k)$$
-
$$h_0 + h_1 \log(k) = \alpha \log(k) + \alpha\beta h_1 \log(k) + \underbrace{\log(A) + \log\left(\frac{1}{\beta h_1 + 1}\right) + \beta h_0 + \beta h_1 \log\left(\frac{A\beta h_1}{\beta h_1 + 1}\right)}_{M}$$
-
$$h_0 + h_1 \log(k) = (\beta h_1 \alpha + \alpha) \log(k) + M$$

Dinamik Programlama

Tahmin ve Doğrulama Yöntemi için 1.Aşama

- Elde edilen bu denklemde k içeren ifadeleri ve diğer tüm ifadeleri (sabitleri) ayıralım.
- $$h_0 + h_1 \log(k) = \log(A) + \alpha \log(k) + \log\left(\frac{1}{\beta h_1 + 1}\right) + \beta h_0 + \beta h_1 \log\left(\frac{A\beta h_1}{\beta h_1 + 1}\right) + \alpha\beta h_1 \log(k)$$
- $$h_0 + h_1 \log(k) = \alpha \log(k) + \alpha\beta h_1 \log(k) + \underbrace{\log(A) + \log\left(\frac{1}{\beta h_1 + 1}\right) + \beta h_0 + \beta h_1 \log\left(\frac{A\beta h_1}{\beta h_1 + 1}\right)}_{M}$$
- $$h_0 + h_1 \log(k) = (\beta h_1 \alpha + \alpha) \log(k) + M$$
- Yukarıdaki denklemde eşitliğin her iki tarafında $\log(k)$ teriminin önünde bulunan değerleri birbirine eşitlersek:

Dinamik Programlama

Tahmin ve Doğrulama Yöntemi için 1.Aşama

- Elde edilen bu denklemde k içeren ifadeleri ve diğer tüm ifadeleri (sabitleri) ayıralım.
- $$h_0 + h_1 \log(k) = \log(A) + \alpha \log(k) + \log\left(\frac{1}{\beta h_1 + 1}\right) + \beta h_0 + \beta h_1 \log\left(\frac{A\beta h_1}{\beta h_1 + 1}\right) + \alpha\beta h_1 \log(k)$$
- $$h_0 + h_1 \log(k) = \alpha \log(k) + \alpha\beta h_1 \log(k) + \underbrace{\log(A) + \log\left(\frac{1}{\beta h_1 + 1}\right) + \beta h_0 + \beta h_1 \log\left(\frac{A\beta h_1}{\beta h_1 + 1}\right)}_{M}$$
- $$h_0 + h_1 \log(k) = (\beta h_1 \alpha + \alpha) \log(k) + M$$
- Yukarıdaki denklemde eşitliğin her iki tarafında $\log(k)$ teriminin önünde bulunan değerleri birbirine eşitlersek:
- $$h_1 = \frac{\alpha}{1 - \alpha\beta}$$

Tahmin ve Doğrulama Yöntemi için 1.Aşama

Tahmin ve Doğrulama Yöntemi için 1.Aşama

- h_0 değerinin bulunması için $h_0 = M$ eşitliği kullanılır. Bu eşitlikte h_1 değerinin yerine yukarıda bulunan ifade yazılır:

Tahmin ve Doğrulama Yöntemi için 1.Aşama

- h_0 değerinin bulunması için $h_0 = M$ eşitliği kullanılır. Bu eşitlikte h_1 değerinin yerine yukarıda bulunan ifade yazılır:
- $$h_0 = \log(A) + \log\left(\frac{1}{\beta(\frac{\alpha}{1-\alpha\beta})+1}\right) + \beta h_0 + \beta\left(\frac{\alpha}{1-\alpha\beta}\right)\log\left(\frac{A\beta\left(\frac{\alpha}{1-\alpha\beta}\right)}{\beta(\frac{\alpha}{1-\alpha\beta})+1}\right)$$

Tahmin ve Doğrulama Yöntemi için 1.Aşama

- h_0 değerinin bulunması için $h_0 = M$ eşitliği kullanılır. Bu eşitlikte h_1 değerinin yerine yukarıda bulunan ifade yazılır:
- $$h_0 = \log(A) + \log\left(\frac{1}{\beta(\frac{\alpha}{1-\alpha\beta})+1}\right) + \beta h_0 + \beta\left(\frac{\alpha}{1-\alpha\beta}\right)\log\left(\frac{A\beta\left(\frac{\alpha}{1-\alpha\beta}\right)}{\beta(\frac{\alpha}{1-\alpha\beta})+1}\right)$$
- $$h_0(1-\beta) = \log(A) + \log\left(1-(\alpha\beta)\right) + \beta\left(\frac{\alpha}{1-\alpha\beta}\right)\log(A\alpha\beta)$$

Tahmin ve Doğrulama Yöntemi için 1.Aşama

- h_0 değerinin bulunması için $h_0 = M$ eşitliği kullanılır. Bu eşitlikte h_1 değerinin yerine yukarıda bulunan ifade yazılır:
- $$h_0 = \log(A) + \log\left(\frac{1}{\beta(\frac{\alpha}{1-\alpha\beta})+1}\right) + \beta h_0 + \beta\left(\frac{\alpha}{1-\alpha\beta}\right)\log\left(\frac{A\beta\left(\frac{\alpha}{1-\alpha\beta}\right)}{\beta(\frac{\alpha}{1-\alpha\beta})+1}\right)$$
- $$h_0(1-\beta) = \log(A) + \log(1-(\alpha\beta)) + \beta\left(\frac{\alpha}{1-\alpha\beta}\right)\log(A\alpha\beta)$$
- $$h_0 = \frac{1}{1-\beta}\left(\log A(1-\alpha\beta) + \frac{\beta\alpha}{1-\alpha\beta}\log(A\beta\alpha)\right)$$

Tahmin ve Doğrulama Yöntemi için 1.Aşama

Dinamik Programlama

Tahmin ve Doğrulama Yöntemi için 1.Aşama

- Bu problemde asıl ulaşılmak istenen $g(k) = k'$ yani politika fonksiyonudur (policy function). Yani herhangi bir k degeri verildiğinde gelecek dönem optimal sermaye için (k') ne seçilmeli sorusuna yanıt aranmaktadır.

Tahmin ve Doğrulama Yöntemi için 1.Aşama

- Bu problemde asıl ulaşılmak istenen $g(k) = k'$ yani politika fonksiyonudur (policy function). Yani herhangi bir k degeri verildiğinde gelecek dönem optimal sermaye için (k') ne seçilmeli sorusuna yanıt aranmaktadır.
- Politika fonksiyonunun çözümü için h_1 bilgisi yeterlidir (h_0'a gerek yoktur).

Dinamik Programlama

Tahmin ve Doğrulama Yöntemi için 1.Aşama

- Bu problemde asıl ulaşılmak istenen $g(k) = k'$ yani politika fonksiyonudur (policy function). Yani herhangi bir k degeri verildiğinde gelecek dönem optimal sermaye için (k') ne seçilmeli sorusuna yanıt aranmaktadır.
- Politika fonksiyonunun çözümü için h_1 bilgisi yeterlidir (h_0'a gerek yoktur).
- Bu sorunun cevabı $k' = \frac{Ak^{\alpha}\beta h_1}{\beta h_1 + 1}$ denkleminde h_1'in yerine ilgili parametreler yazılarak elde edilir.

Tahmin ve Doğrulama Yöntemi için 1.Aşama

- Bu problemde asıl ulaşılmak istenen $g(k) = k'$ yani politika fonksiyonudur (policy function). Yani herhangi bir k degeri verildiğinde gelecek dönem optimal sermaye için (k') ne seçilmeli sorusuna yanıt aranmaktadır.
- Politika fonksiyonunun çözümü için h_1 bilgisi yeterlidir (h_0'a gerek yoktur).
- Bu sorunun cevabı $k' = \frac{Ak^{\alpha}\beta h_1}{\beta h_1 + 1}$ denkleminde h_1'in yerine ilgili parametreler yazılarak elde edilir.
- Yani

$$k' = \frac{Ak^{\alpha}\beta\frac{\alpha}{1-\alpha\beta}}{\beta\frac{\alpha}{1-\alpha\beta}+1} = Ak^{\alpha}\beta\alpha$$

Tahmin ve Doğrulama Yöntemi için 1.Aşama

- Bu problemde asıl ulaşılmak istenen $g(k) = k'$ yani politika fonksiyonudur (policy function). Yani herhangi bir k degeri verildiğinde gelecek dönem optimal sermaye için (k') ne seçilmeli sorusuna yanıt aranmaktadır.
- Politika fonksiyonunun çözümü için h_1 bilgisi yeterlidir (h_0'a gerek yoktur).
- Bu sorunun cevabı $k' = \frac{Ak^{\alpha}\beta h_1}{\beta h_1 + 1}$ denkleminde h_1'in yerine ilgili parametreler yazılarak elde edilir.
- Yani

$$k' = \frac{Ak^{\alpha}\beta \frac{\alpha}{1-\alpha\beta}}{\beta \frac{\alpha}{1-\alpha\beta} + 1} = Ak^{\alpha}\beta\alpha$$

- Bu durumda $k' = Ak^{\alpha}\beta\alpha$ olur. k_0 ve diğer parametreler bilindigi zaman tüm k' değerleri bulunur.

Dinamik Programlama

Tahmin ve Doğrulama Yöntemi için 1.Aşama

- Bu problemde asıl ulaşılmak istenen $g(k) = k'$ yani politika fonksiyonudur (policy function). Yani herhangi bir k degeri verildiğinde gelecek dönem optimal sermaye için (k') ne seçilmeli sorusuna yanıt aranmaktadır.
- Politika fonksiyonunun çözümü için h_1 bilgisi yeterlidir (h_0'a gerek yoktur).
- Bu sorunun cevabı $k' = \frac{Ak^{\alpha}\beta h_1}{\beta h_1 + 1}$ denkleminde h_1'in yerine ilgili parametreler yazılarak elde edilir.
- Yani

$$k' = \frac{Ak^{\alpha}\beta\frac{\alpha}{1-\alpha\beta}}{\beta\frac{\alpha}{1-\alpha\beta}+1} = Ak^{\alpha}\beta\alpha$$

- Bu durumda $k' = Ak^{\alpha}\beta\alpha$ olur. k_0 ve diğer parametreler bilindigi zaman tüm k' değerleri bulunur.
- k' degerleri bilindiği zaman optimal c değerleri de bulunur:

Tahmin ve Doğrulama Yöntemi için 1.Aşama

- Bu problemde asıl ulaşılmak istenen $g(k) = k'$ yani politika fonksiyonudur (policy function). Yani herhangi bir k degeri verildiğinde gelecek dönem optimal sermaye için (k') ne seçilmeli sorusuna yanıt aranmaktadır.
- Politika fonksiyonunun çözümü için h_1 bilgisi yeterlidir (h_0'a gerek yoktur).
- Bu sorunun cevabı $k' = \frac{Ak^{\alpha}\beta h_1}{\beta h_1 + 1}$ denkleminde h_1'in yerine ilgili parametreler yazılarak elde edilir.
- Yani

$$k' = \frac{Ak^{\alpha}\beta\frac{\alpha}{1-\alpha\beta}}{\beta\frac{\alpha}{1-\alpha\beta}+1} = Ak^{\alpha}\beta\alpha$$

- Bu durumda $k' = Ak^{\alpha}\beta\alpha$ olur. k_0 ve diğer parametreler bilindigi zaman tüm k' değerleri bulunur.
- k' degerleri bilindiği zaman optimal c değerleri de bulunur:
-

$$c(k) = Ak^{\alpha} - k' = Ak^{\alpha} - Ak^{\alpha}\beta\alpha = Ak^{\alpha}(1 - \beta\alpha)$$

Dinamik Programlama

Tahmin ve Doğrulama Yöntemi için 1.Aşama

- Bu problemde asıl ulaşılmak istenen $g(k) = k'$ yani politika fonksiyonudur (policy function). Yani herhangi bir k degeri verildiğinde gelecek dönem optimal sermaye için (k') ne seçilmeli sorusuna yanıt aranmaktadır.
- Politika fonksiyonunun çözümü için h_1 bilgisi yeterlidir (h_0'a gerek yoktur).
- Bu sorunun cevabı $k' = \frac{Ak^\alpha \beta h_1}{\beta h_1 + 1}$ denkleminde h_1'in yerine ilgili parametreler yazılarak elde edilir.
- Yani

$$k' = \frac{Ak^\alpha \beta \frac{\alpha}{1-\alpha\beta}}{\beta \frac{\alpha}{1-\alpha\beta} + 1} = Ak^\alpha \beta\alpha$$

- Bu durumda $k' = Ak^\alpha \beta\alpha$ olur. k_0 ve diğer parametreler bilindigi zaman tüm k' değerleri bulunur.
- k' degerleri bilindiği zaman optimal c değerleri de bulunur:
-

$$c(k) = Ak^\alpha - k' = Ak^\alpha - Ak^\alpha \beta\alpha = Ak^\alpha(1 - \beta\alpha)$$

- **Burada sermaye değerleri k_0'dan başlayarak $k' = Ak^\alpha \beta\alpha$ denklemindeki zaman patikasını izleyecektir ve sonunda k^{ss} gibi sabit bir değere yakınsayacaktır.**

Dinamik Programlama

Tahmin ve Doğrulama Yöntemi için 1.Aşama

- Bu problemde asıl ulaşılmak istenen $g(k) = k'$ yani politika fonksiyonudur (policy function). Yani herhangi bir k degeri verildiğinde gelecek dönem optimal sermaye için (k') ne seçilmeli sorusuna yanıt aranmaktadır.
- Politika fonksiyonunun çözümü için h_1 bilgisi yeterlidir (h_0'a gerek yoktur).
- Bu sorunun cevabı $k' = \frac{Ak^\alpha \beta h_1}{\beta h_1 + 1}$ denkleminde h_1'in yerine ilgili parametreler yazılarak elde edilir.
- Yani

$$k' = \frac{Ak^\alpha \beta \frac{\alpha}{1-\alpha\beta}}{\beta \frac{\alpha}{1-\alpha\beta} + 1} = Ak^\alpha \beta\alpha$$

- Bu durumda $k' = Ak^\alpha \beta\alpha$ olur. k_0 ve diğer parametreler bilindigi zaman tüm k' değerleri bulunur.
- k' degerleri bilindiği zaman optimal c değerleri de bulunur:
-

$$c(k) = Ak^\alpha - k' = Ak^\alpha - Ak^\alpha \beta\alpha = Ak^\alpha(1 - \beta\alpha)$$

- **Burada sermaye değerleri k_0'dan başlayarak $k' = Ak^\alpha \beta\alpha$ denklemindeki zaman patikasını izleyecektir ve sonunda k^{ss} gibi sabit bir değere yakınsayacaktır.**
- **Benzer şekilde tüketim değerleri $c(k) = Ak^\alpha(1 - \beta\alpha)$ zaman patikasını izleyerek c^{ss} gibi sabit bir değere yakınsayacaktır.**

Tahmin ve Doğrulama Yöntemi için 2.Aşama

Tahmin ve Doğrulama Yöntemi için 2.Aşama

- **2. aşama: Doğrulama**

Tahmin ve Doğrulama Yöntemi için 2.Aşama

- **2. aşama: Doğrulama**
- $k' = Ak^{\alpha}\beta\alpha$ çözümünü birinci sıra koşulu olan $\frac{-1}{Ak^{\alpha}-k'} + \beta h_1 \frac{1}{k'} = 0$ denkleminde yerine koyarak kontrol edelim:

Tahmin ve Doğrulama Yöntemi için 2.Aşama

- **2. aşama: Doğrulama**
- $k' = Ak^{\alpha}\beta\alpha$ çözümünü birinci sıra koşulu olan $\frac{-1}{Ak^{\alpha}-k'} + \beta h_1 \frac{1}{k'} = 0$ denkleminde yerine koyarak kontrol edelim:
- $$\frac{-1}{Ak^{\alpha} - Ak^{\alpha}\beta\alpha} + \beta \frac{\alpha}{1-\alpha\beta} \frac{1}{Ak^{\alpha}\beta\alpha} = 0$$

  sağlandığından cevabımız doğrulanmıştır.

Dinamik Programlama

Değer Fonksiyonu İterasyonu (Value Function Iteration)

Dinamik Programlama

Değer Fonksiyonu İterasyonu (Value Function Iteration)

- Neo-Klasik Büyüme modelini ele alalım:

Dinamik Programlama

Değer Fonksiyonu İterasyonu (Value Function Iteration)

- Neo-Klasik Büyüme modelini ele alalım:
-

$$\max_{k_{t+1}} \sum_{t=0}^{\infty} \beta^t log(Ak_t^{\alpha} - k_{t+1})$$

Dinamik Programlama

Değer Fonksiyonu İterasyonu (Value Function Iteration)

- Neo-Klasik Büyüme modelini ele alalım:
- $$\max_{k_{t+1}} \sum_{t=0}^{\infty} \beta^t log(Ak_t^{\alpha} - k_{t+1})$$
- $$k_0 > 0 \text{ veri}$$

Dinamik Programlama

Değer Fonksiyonu İterasyonu (Value Function Iteration)

- Neo-Klasik Büyüme modelini ele alalım:
- $$\max_{k_{t+1}} \sum_{t=0}^{\infty} \beta^t log(Ak_t^{\alpha} - k_{t+1})$$
- $$k_0 > 0 \text{ veri}$$
- Büyüme modeli için Bellman denklemini yazarsak:

Dinamik Programlama

Değer Fonksiyonu İterasyonu (Value Function Iteration)

- Neo-Klasik Büyüme modelini ele alalım:
- $$\max_{k_{t+1}} \sum_{t=0}^{\infty} \beta^t log(Ak_t^{\alpha} - k_{t+1})$$
- $$k_0 > 0 \text{ veri}$$
- Büyüme modeli için Bellman denklemini yazarsak:
- $$V(k) = \max_{0 \leq k' \leq Ak^{\alpha}} \underbrace{\log(Ak^{\alpha} - k')}_{\text{bugunku fayda}} + \beta \underbrace{V(k')}_{\text{gelecekteki fayda}}$$

Dinamik Programlama

Değer Fonksiyonu İterasyonu (Value Function Iteration)

- Neo-Klasik Büyüme modelini ele alalım:
- $$\max_{k_{t+1}} \sum_{t=0}^{\infty} \beta^t log(Ak_t^{\alpha} - k_{t+1})$$
- $$k_0 > 0 \text{ veri}$$
- Büyüme modeli için Bellman denklemini yazarsak:
- $$V(k) = \max_{0 \leq k' \leq Ak^{\alpha}} \underbrace{\log(Ak^{\alpha} - k')}_{\substack{\text{bugunku} \\ \text{fayda}}} + \beta \underbrace{V(k')}_{\substack{\text{gelecekteki} \\ \text{fayda}}}$$
- Algoritma aşamalarını gösterebilmek için şu şekilde yazalım:

Dinamik Programlama

Değer Fonksiyonu İterasyonu (Value Function Iteration)

- Neo-Klasik Büyüme modelini ele alalım:
- $$\max_{k_{t+1}} \sum_{t=0}^{\infty} \beta^t log(Ak_t^{\alpha} - k_{t+1})$$
- $$k_0 > 0 \text{ veri}$$
- Büyüme modeli için Bellman denklemini yazarsak:
- $$V(k) = \max_{0 \leq k' \leq Ak^{\alpha}} \underbrace{\log(Ak^{\alpha} - k')}_{\text{bugunku fayda}} + \beta \underbrace{V(k')}_{\text{gelecekteki fayda}}$$
- Algoritma aşamalarını gösterebilmek için şu şekilde yazalım:
- $$V_{n+1}(k) = \max_{0 \leq k' \leq Ak^{\alpha}} \log(Ak^{\alpha} - k') + \beta V_n(k').$$

Dinamik Programlama

Değer Fonksiyonu İterasyonu (Value Function Iteration)

Dinamik Programlama

Değer Fonksiyonu İterasyonu (Value Function Iteration)

- $$V_{n+1}(k) = \max_{0 \leq k' \leq Ak^{\alpha}} \log(Ak^{\alpha} - k') + \beta V_n(k').$$

Dinamik Programlama

Değer Fonksiyonu İterasyonu (Value Function Iteration)

- $$V_{n+1}(k) = \max_{0 \leq k' \leq Ak^{\alpha}} \log(Ak^{\alpha} - k') + \beta V_n(k').$$
- Burada $n \geq 1$ iterasyon sayısını göstermektedir.

Değer Fonksiyonu İterasyonu (Value Function Iteration)

- $$V_{n+1}(k) = \max_{0 \leq k' \leq Ak^{\alpha}} \log(Ak^{\alpha} - k') + \beta V_n(k').$$

- Burada $n \geq 1$ iterasyon sayısını göstermektedir.
- Değer fonksiyonu iterasyonunda belirli bir aşamadan sonra V_n değeri V_{n+1} değerine yakınsamalıdır.

Dinamik Programlama

Değer Fonksiyonu İterasyonu (Value Function Iteration)

- $$V_{n+1}(k) = \max_{0 \leq k' \leq Ak^{\alpha}} \log(Ak^{\alpha} - k') + \beta V_n(k').$$

- Burada $n \geq 1$ iterasyon sayısını göstermektedir.
- Değer fonksiyonu iterasyonunda belirli bir aşamadan sonra V_n değeri V_{n+1} değerine yakınsamalıdır.
- $V_1 = 0 \ \forall \ k$ gibi bir başlangıç değerinden başlanarak $V_n \rightarrow V_{n+1}$ olana kadar iterasyon devam eder.

Dinamik Programlama

Değer Fonksiyonu İterasyonu (Value Function Iteration)

- $$V_{n+1}(k) = \max_{0 \leq k' \leq Ak^{\alpha}} \log(Ak^{\alpha} - k') + \beta V_n(k').$$

- Burada $n \geq 1$ iterasyon sayısını göstermektedir.
- Değer fonksiyonu iterasyonunda belirli bir aşamadan sonra V_n değeri V_{n+1} değerine yakınsamalıdır.
- $V_1 = 0 \ \forall \ k$ gibi bir başlangıç değerinden başlanarak $V_n \to V_{n+1}$ olana kadar iterasyon devam eder.
- $V_n \to V_{n+1}$ durumunu sağlayan $g(k) = k'$ fonksiyonu problemin politika fonksiyonunu (i.e. k için zaman patikasını) oluşturur.

Dinamik Programlama

Değer Fonksiyonu İterasyonu (Value Function Iteration)

- $$V_{n+1}(k) = \max_{0 \leq k' \leq Ak^{\alpha}} \log(Ak^{\alpha} - k') + \beta V_n(k').$$

- Burada $n \geq 1$ iterasyon sayısını göstermektedir.
- Değer fonksiyonu iterasyonunda belirli bir aşamadan sonra V_n değeri V_{n+1} değerine yakınsamalıdır.
- $V_1 = 0 \ \forall \ k$ gibi bir başlangıç değerinden başlanarak $V_n \rightarrow V_{n+1}$ olana kadar iterasyon devam eder.
- $V_n \rightarrow V_{n+1}$ durumunu sağlayan $g(k) = k'$ fonksiyonu problemin politika fonksiyonunu (i.e. k için zaman patikasını) oluşturur.
- Daha sonra $g(k)$ ve bütçe kısıtı kullanılarak tüketim değerleri de kolayca bulunabilir.

Dinamik Programlama

Algoritma adımlarını şöyle sıralayabiliriz:

Dinamik Programlama

Algoritma adımlarını şöyle sıralayabiliriz:

1. Problem "recursive" formda yazılır.

Dinamik Programlama

Algoritma adımlarını şöyle sıralayabiliriz:

1. Problem "recursive" formda yazılır.
2. Başlangıç değeri $V_1(k) = 0 \ \forall \ k$ kabul edilir.

Dinamik Programlama

Algoritma adımlarını şöyle sıralayabiliriz:

1. Problem "recursive" formda yazılır.
2. Başlangıç değeri $V_1(k) = 0 \ \forall \ k$ kabul edilir.
3. $V_1(k) = 0$ iken yukarıdaki maksimizasyon problemi çözülerek $V_2(k)$ değerleri bulunur.

Dinamik Programlama

Algoritma adımlarını şöyle sıralayabiliriz:

1. Problem "recursive" formda yazılır.
2. Başlangıç değeri $V_1(k) = 0 \ \forall \ k$ kabul edilir.
3. $V_1(k) = 0$ iken yukarıdaki maksimizasyon problemi çözülerek $V_2(k)$ değerleri bulunur.
4. $V_2(k) - V_1(k) \leq \epsilon$ olup olmadığı kontrol edilir. Eğer sağlanmışsa problem çözülmüştür. Sağlanmamışsa sağlanana kadar iterasyon devam eder (ϵ herhangi bir küçük değer).

Dinamik Programlama

Algoritma adımlarını şöyle sıralayabiliriz:

1. Problem "recursive" formda yazılır.
2. Başlangıç değeri $V_1(k) = 0 \ \forall \ k$ kabul edilir.
3. $V_1(k) = 0$ iken yukarıdaki maksimizasyon problemi çözülerek $V_2(k)$ değerleri bulunur.
4. $V_2(k) - V_1(k) \leq \epsilon$ olup olmadığı kontrol edilir. Eğer sağlanmışsa problem çözülmüştür. Sağlanmamışsa sağlanana kadar iterasyon devam eder (ϵ herhangi bir küçük değer).
5. $V_{n+1}(k) - V_n(k) \leq \epsilon$ ifadesini sağlayan n. aşamadaki $g(k) = k'$ ifadesi problemin çözüm kümesidir.

Dinamik Programlama

Algoritma adımlarını şöyle sıralayabiliriz:

1. Problem "recursive" formda yazılır.
2. Başlangıç değeri $V_1(k) = 0 \ \forall \ k$ kabul edilir.
3. $V_1(k) = 0$ iken yukarıdaki maksimizasyon problemi çözülerek $V_2(k)$ değerleri bulunur.
4. $V_2(k) - V_1(k) \leq \epsilon$ olup olmadığı kontrol edilir. Eğer sağlanmışsa problem çözülmüştür. Sağlanmamışsa sağlanana kadar iterasyon devam eder (ϵ herhangi bir küçük değer).
5. $V_{n+1}(k) - V_n(k) \leq \epsilon$ ifadesini sağlayan n. aşamadaki $g(k) = k'$ ifadesi problemin çözüm kümesidir.
6. Daha sonra bütçe kısıtı ve k değerleri kullanılarak c değeri de bulunur.

Algoritma adımlarının Uygulanması

Algoritma adımlarının Uygulanması

- Bu algoritma genellikle el yordamı ile çok vakit alacağından bilgisayar yardımıyla çözülebilir. En uygun programlardan biri "MATLAB" programıdır.

Dinamik Programlama

Algoritma adımlarının Uygulanması

- Bu algoritma genellikle el yordamı ile çok vakit alacağından bilgisayar yardımıyla çözülebilir. En uygun programlardan biri "MATLAB" programıdır.
- **Örnek:**

Algoritma adımlarının Uygulanması

- Bu algoritma genellikle el yordamı ile çok vakit alacağından bilgisayar yardımıyla çözülebilir. En uygun programlardan biri "MATLAB" programıdır.
- **Örnek:**
-

$$\max_{c_t, k_{t+1}} \sum_{t=0}^{\infty} 0.6^t log(c_t)$$

Dinamik Programlama

Algoritma adımlarının Uygulanması

- Bu algoritma genellikle el yordamı ile çok vakit alacağından bilgisayar yardımıyla çözülebilir. En uygun programlardan biri "MATLAB" programıdır.
- **Örnek:**
- $$\max_{c_t, k_{t+1}} \sum_{t=0}^{\infty} 0.6^t log(c_t)$$
- s.t.
$$k_{t+1} + c_t = 20k_t^{0.3}$$

Dinamik Programlama

Algoritma adımlarının Uygulanması

- Bu algoritma genellikle el yordamı ile çok vakit alacağından bilgisayar yardımıyla çözülebilir. En uygun programlardan biri "MATLAB" programıdır.
- **Örnek:**
- $$\max_{c_t, k_{t+1}} \sum_{t=0}^{\infty} 0.6^t log(c_t)$$
- s.t. $$k_{t+1} + c_t = 20k_t^{0.3}$$
- $$k_0 > 0 \quad \text{veri}$$

Dinamik Programlama

Algoritma adımlarının Uygulanması

- Bu algoritma genellikle el yordamı ile çok vakit alacağından bilgisayar yardımıyla çözülebilir. En uygun programlardan biri "MATLAB" programıdır.
- **Örnek:**
- $$\max_{c_t, k_{t+1}} \sum_{t=0}^{\infty} 0.6^t log(c_t)$$
- s.t. $$k_{t+1} + c_t = 20k_t^{0.3}$$
- $$k_0 > 0 \quad \text{veri}$$
- $$k \in \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$$

Dinamik Programlama

Algoritma adımlarının Uygulanması

- Bu algoritma genellikle el yordamı ile çok vakit alacağından bilgisayar yardımıyla çözülebilir. En uygun programlardan biri "MATLAB" programıdır.
- **Örnek:**
- $$\max_{c_t, k_{t+1}} \sum_{t=0}^{\infty} 0.6^t log(c_t)$$
- s.t. $$k_{t+1} + c_t = 20k_t^{0.3}$$
- $$k_0 > 0 \quad \text{veri}$$
- $$k \in \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$$
- Bilgisayar ile çözeceğimiz için k değişkeninin kesikli değerler alması gerekiyor.

Dinamik Programlama

Algoritma adımlarının Uygulanması

- Bu algoritma genellikle el yordamı ile çok vakit alacağından bilgisayar yardımıyla çözülebilir. En uygun programlardan biri "MATLAB" programıdır.
- **Örnek:**
- $$\max_{c_t, k_{t+1}} \sum_{t=0}^{\infty} 0.6^t log(c_t)$$
- s.t. $$k_{t+1} + c_t = 20k_t^{0.3}$$
- $$k_0 > 0 \quad \text{veri}$$
- $$k \in \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$$
- Bilgisayar ile çözeceğimiz için k değişkeninin kesikli değerler alması gerekiyor.
- Bu örnekte k'nın sadece $k \in \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ değerlerini alabildiğini varsayıyoruz.

Dinamik Programlama

Algoritma adımlarının Uygulanması

- Bu algoritma genellikle el yordamı ile çok vakit alacağından bilgisayar yardımıyla çözülebilir. En uygun programlardan biri "MATLAB" programıdır.
- **Örnek:**
- $$\max_{c_t, k_{t+1}} \sum_{t=0}^{\infty} 0.6^t log(c_t)$$
- s.t. $$k_{t+1} + c_t = 20k_t^{0.3}$$
- $$k_0 > 0 \quad \text{veri}$$
- $$k \in \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$$
- Bilgisayar ile çözeceğimiz için k değişkeninin kesikli değerler alması gerekiyor.
- Bu örnekte k'nın sadece $k \in \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ değerlerini alabildiğini varsayıyoruz.
- $V_n \rightarrow V_{n+1}$'yi de şu şekilde tanımlayabiliriz:

Dinamik Programlama

Algoritma adımlarının Uygulanması

- Bu algoritma genellikle el yordamı ile çok vakit alacağından bilgisayar yardımıyla çözülebilir. En uygun programlardan biri "MATLAB" programıdır.
- **Örnek:**
- $$\max_{c_t, k_{t+1}} \sum_{t=0}^{\infty} 0.6^t log(c_t)$$
- s.t. $$k_{t+1} + c_t = 20k_t^{0.3}$$
- $$k_0 > 0 \quad \text{veri}$$
- $$k \in \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$$
- Bilgisayar ile çözeceğimiz için k değişkeninin kesikli değerler alması gerekiyor.
- Bu örnekte k'nın sadece $k \in \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ değerlerini alabildiğini varsayıyoruz.
- $V_n \rightarrow V_{n+1}$'yi de şu şekilde tanımlayabiliriz:
- $V_{n+1} - V_n \leq \epsilon$. Burada epsilon herhangi bir küçük sayı. Bu örnekte $\epsilon = 0.01$ olsun.

Dinamik Programlama

Algoritma adımlarının Uygulanması

Algoritma adımlarının Uygulanması

- Bu örneği "Recursive" formda yazarsak:

Dinamik Programlama

Algoritma adımlarının Uygulanması

- Bu örneği "Recursive" formda yazarsak:
- $$V_{n+1}(k) = \max_{k \in \{1,2,3,4,5,6,7,8,9,10\}} \log(20k^{0.3} - k') + 0.6V_n(k').$$

Dinamik Programlama

Algoritma adımlarının Uygulanması

- Bu örneği "Recursive" formda yazarsak:
- $$V_{n+1}(k) = \max_{k \in \{1,2,3,4,5,6,7,8,9,10\}} \log(20k^{0.3} - k') + 0.6V_n(k').$$
- Bu durumda bulunmak istenen $g(k) = k'$ ifadesidir.

Algoritma adımlarının Uygulanması

- Bu örneği "Recursive" formda yazarsak:
- $$V_{n+1}(k) = \max_{k \in \{1,2,3,4,5,6,7,8,9,10\}} \log(20k^{0.3} - k') + 0.6V_n(k').$$
- Bu durumda bulunmak istenen $g(k) = k'$ ifadesidir.
- $V_{n+1} - V_n \leq 0.01$ sağlandığında $g(k) = k'$ ifadesini bulmuş olacağız.

Algoritma adımlarının Uygulanması

- Bu örneği "Recursive" formda yazarsak:
- $$V_{n+1}(k) = \max_{k \in \{1,2,3,4,5,6,7,8,9,10\}} \log(20k^{0.3} - k') + 0.6V_n(k').$$
- Bu durumda bulunmak istenen $g(k) = k'$ ifadesidir.
- $V_{n+1} - V_n \leq 0.01$ sağlandığında $g(k) = k'$ ifadesini bulmuş olacağız.
- Başlangıç değeri de $V_1(k) = 0 \ \forall \ k$ olsun.

Dinamik Programlama

Algoritma adımlarının Uygulanması

- Bu örneği "Recursive" formda yazarsak:
- $$V_{n+1}(k) = \max_{k \in \{1,2,3,4,5,6,7,8,9,10\}} \log(20k^{0.3} - k') + 0.6V_n(k').$$
- Bu durumda bulunmak istenen $g(k) = k'$ ifadesidir.
- $V_{n+1} - V_n \leq 0.01$ sağlandığında $g(k) = k'$ ifadesini bulmuş olacağız.
- Başlangıç değeri de $V_1(k) = 0 \ \forall \ k$ olsun.
- **Çözümün mantığını anlamak için ilk önce "MATLAB" yerine bu algoritmaları "el yordamına yakın bir şekilde excel programında" nasıl gerçekleştireceğimize Ek 2 yardımıyla bakalım.**

2. Örnek: Kek Yeme Problemi

2. Örnek: Kek Yeme Problemi

- Zaman sonsuz ve kesikli olsun $t = 0, 1, 2, \ldots$.

2. Örnek: Kek Yeme Problemi

- Zaman sonsuz ve kesikli olsun $t = 0, 1, 2, \ldots$.
- İndirgeme faktörü $0 < \beta < 1$ aralığında olsun.

2. Örnek: Kek Yeme Problemi

- Zaman sonsuz ve kesikli olsun $t = 0, 1, 2, \ldots$.
- İndirgeme faktörü $0 < \beta < 1$ aralığında olsun.
- Bir kişinin başlangıçta $W_0 > 0$ büyüklüğünde bir keke sahip olduğunu kabul edelim.

2. Örnek: Kek Yeme Problemi

- Zaman sonsuz ve kesikli olsun $t = 0, 1, 2, \ldots$.
- İndirgeme faktörü $0 < \beta < 1$ aralığında olsun.
- Bir kişinin başlangıçta $W_0 > 0$ büyüklüğünde bir keke sahip olduğunu kabul edelim.
- Tüketici her t döneminde W_t büyüklüğündeki kekin bir kısmını yemektedir (c_t) ve geri kalanını sonraki dönemler için saklamaktadır W_{t+1}.Yani $c_t + W_{t+1} \leq W_t \quad \forall t$.

2. Örnek: Kek Yeme Problemi

- Zaman sonsuz ve kesikli olsun $t = 0, 1, 2, \ldots$.
- İndirgeme faktörü $0 < \beta < 1$ aralığında olsun.
- Bir kişinin başlangıçta $W_0 > 0$ büyüklüğünde bir keke sahip olduğunu kabul edelim.
- Tüketici her t döneminde W_t büyüklüğündeki kekin bir kısmını yemektedir (c_t) ve geri kalanını sonraki dönemler için saklamaktadır W_{t+1}.Yani $c_t + W_{t+1} \leq W_t \quad \forall t$.
- Saklanan kekin bozulmadığını varsayalım.

2. Örnek: Kek Yeme Problemi

- Zaman sonsuz ve kesikli olsun $t = 0, 1, 2, \ldots$.
- İndirgeme faktörü $0 < \beta < 1$ aralığında olsun.
- Bir kişinin başlangıçta $W_0 > 0$ büyüklüğünde bir keke sahip olduğunu kabul edelim.
- Tüketici her t döneminde W_t büyüklüğündeki kekin bir kısmını yemektedir (c_t) ve geri kalanını sonraki dönemler için saklamaktadır W_{t+1}.Yani $c_t + W_{t+1} \leq W_t \ \ \forall t$.
- Saklanan kekin bozulmadığını varsayalım.
- Kişinin fayda fonksiyonu şöyle verilmiştir:

$$\max_{c_t, W_{t+1}} \sum_{t=0}^{\infty} \beta^t log(c_t)$$

2. Örnek: Kek Yeme Problemi

- Zaman sonsuz ve kesikli olsun $t = 0, 1, 2, \ldots$.
- İndirgeme faktörü $0 < \beta < 1$ aralığında olsun.
- Bir kişinin başlangıçta $W_0 > 0$ büyüklüğünde bir keke sahip olduğunu kabul edelim.
- Tüketici her t döneminde W_t büyüklüğündeki kekin bir kısmını yemektedir (c_t) ve geri kalanını sonraki dönemler için saklamaktadır W_{t+1}.Yani $c_t + W_{t+1} \leq W_t \ \ \forall t$.
- Saklanan kekin bozulmadığını varsayalım.
- Kişinin fayda fonksiyonu şöyle verilmiştir:

$$\max_{c_t, W_{t+1}} \sum_{t=0}^{\infty} \beta^t log(c_t)$$

- Bu durumda kişinin her dönem tüketmesi gereken optimal kek miktarı (c_t) ne kadardır?

2. Örnek: Kek Yeme Problemi için Sequential Form

2. Örnek: Kek Yeme Problemi için Sequential Form

$$\max_{c_t, W_{t+1}} \sum_{t=0}^{\infty} \beta^t log(c_t)$$

Dinamik Programlama

2. Örnek: Kek Yeme Problemi için Sequential Form

- $$\max_{c_t, W_{t+1}} \sum_{t=0}^{\infty} \beta^t log(c_t)$$
- s.t.
$$c_t + W_{t+1} = W_t \quad \forall t$$

Dinamik Programlama

2. Örnek: Kek Yeme Problemi için Sequential Form

- $$\max_{c_t, W_{t+1}} \sum_{t=0}^{\infty} \beta^t log(c_t)$$
- s.t.
$$c_t + W_{t+1} = W_t \quad \forall t$$
- $$W_0 \geq 0 \quad \text{(veri)}$$

Dinamik Programlama

2. Örnek: Kek Yeme Problemi için Sequential Form

- $$\max_{c_t, W_{t+1}} \sum_{t=0}^{\infty} \beta^t log(c_t)$$
- s.t.
$$c_t + W_{t+1} = W_t \quad \forall t$$
- $$W_0 \geq 0 \quad \text{(veri)}$$
- $$c_t, W_{t+1} \geq 0$$

Dinamik Programlama

2. Örnek: Kek Yeme Problemi için Sequential Form

- $$\max_{c_t, W_{t+1}} \sum_{t=0}^{\infty} \beta^t log(c_t)$$
- s.t.
$$c_t + W_{t+1} = W_t \quad \forall t$$
- $$W_0 \geq 0 \quad \text{(veri)}$$
- $$c_t, W_{t+1} \geq 0$$
- Problemi sadece W cinsinden de **Sequential Formda** yazabiliriz:

Dinamik Programlama

2. Örnek: Kek Yeme Problemi için Sequential Form

- $$\max_{c_t, W_{t+1}} \sum_{t=0}^{\infty} \beta^t log(c_t)$$
- s.t.
$$c_t + W_{t+1} = W_t \quad \forall t$$
- $$W_0 \geq 0 \quad \text{(veri)}$$
- $$c_t, W_{t+1} \geq 0$$
- Problemi sadece W cinsinden de **Sequential Formda** yazabiliriz:
- $$\max_{W_{t+1}} \sum_{t=0}^{\infty} \beta^t log(W_t - W_{t+1})$$

s.t.

Dinamik Programlama

2. Örnek: Kek Yeme Problemi için Sequential Form

- $$\max_{c_t, W_{t+1}} \sum_{t=0}^{\infty} \beta^t log(c_t)$$
- s.t.
$$c_t + W_{t+1} = W_t \quad \forall t$$
- $$W_0 \geq 0 \quad \text{(veri)}$$
- $$c_t, W_{t+1} \geq 0$$
- Problemi sadece W cinsinden de **Sequential Formda** yazabiliriz:
- $$\max_{W_{t+1}} \sum_{t=0}^{\infty} \beta^t log(W_t - W_{t+1})$$

  s.t.
- $$0 \leq W_{t+1} \leq W_t$$

Dinamik Programlama

2. Örnek: Kek Yeme Problemi için Sequential Form

- $$\max_{c_t, W_{t+1}} \sum_{t=0}^{\infty} \beta^t log(c_t)$$
- s.t.
$$c_t + W_{t+1} = W_t \quad \forall t$$
- $$W_0 \geq 0 \quad \text{(veri)}$$
- $$c_t, W_{t+1} \geq 0$$
- Problemi sadece W cinsinden de **Sequential Formda** yazabiliriz:
- $$\max_{W_{t+1}} \sum_{t=0}^{\infty} \beta^t log(W_t - W_{t+1})$$

  s.t.
- $$0 \leq W_{t+1} \leq W_t$$
- $$W_0 \geq 0 \quad \text{(veri)}$$

Dinamik Programlama

Kek Yeme Problemi İçin Bellman Denkleminin Yazılması ve Zarf Teoreminin Uygulanması:

Dinamik Programlama

Kek Yeme Problemi İçin Bellman Denkleminin Yazılması ve Zarf Teoreminin Uygulanması:

- $$V(W) = \max_{0 \leq W' \leq W} [\log(W - W') + \beta V(W')]$$

Dinamik Programlama

Kek Yeme Problemi İçin Bellman Denkleminin Yazılması ve Zarf Teoreminin Uygulanması:

- $$V(W) = \max_{0 \leq W' \leq W} [\log(W - W') + \beta V(W')]$$
- FOC w.r.t. W':

Dinamik Programlama

Kek Yeme Problemi İçin Bellman Denkleminin Yazılması ve Zarf Teoreminin Uygulanması:

- $$V(W) = \max_{0 \leq W' \leq W} [\log(W - W') + \beta V(W')]$$
- FOC w.r.t. W':
- $$\frac{-1}{W - W'} + \beta V'(W') = 0 \quad (*1)$$

Dinamik Programlama

Kek Yeme Problemi İçin Bellman Denkleminin Yazılması ve Zarf Teoreminin Uygulanması:

- $$V(W) = \max_{0 \leq W' \leq W} [\log(W - W') + \beta V(W')]$$
- FOC w.r.t. W':
- $$\frac{-1}{W - W'} + \beta V'(W') = 0 \quad (*1)$$
- Şimdi de $V(W)$ ifadesinin W ile nasıl değiştiğine bakalım (Zarf Teoremi):

Dinamik Programlama

Kek Yeme Problemi İçin Bellman Denkleminin Yazılması ve Zarf Teoreminin Uygulanması:

- $$V(W) = \max_{0 \leq W' \leq W} [\log(W - W') + \beta V(W')]$$
- FOC w.r.t. W':
- $$\frac{-1}{W - W'} + \beta V'(W') = 0 \quad (*1)$$
- Şimdi de $V(W)$ ifadesinin W ile nasıl değiştiğine bakalım (Zarf Teoremi):
- $$V'(W) = \frac{1}{W - W'} \Rightarrow V'(W') = \frac{1}{W' - W''} \quad (*2)$$

Dinamik Programlama

Kek Yeme Problemi İçin Bellman Denkleminin Yazılması ve Zarf Teoreminin Uygulanması:

- $$V(W) = \max_{0 \leq W' \leq W} [\log(W - W') + \beta V(W')]$$
- FOC w.r.t. W':
- $$\frac{-1}{W - W'} + \beta V'(W') = 0 \quad (*1)$$
- Şimdi de $V(W)$ ifadesinin W ile nasıl değiştiğine bakalım (Zarf Teoremi):
- $$V'(W) = \frac{1}{W - W'} \Rightarrow V'(W') = \frac{1}{W' - W''} \quad (*2)$$
- $(*1)$ ve $(*2)$ denklemlerini birleştirirsek:

Dinamik Programlama

Kek Yeme Problemi İçin Bellman Denkleminin Yazılması ve Zarf Teoreminin Uygulanması:

- $$V(W) = \max_{0 \leq W' \leq W} [\log(W - W') + \beta V(W')]$$
- FOC w.r.t. W':
- $$\frac{-1}{W - W'} + \beta V'(W') = 0 \quad (*1)$$
- Şimdi de $V(W)$ ifadesinin W ile nasıl değiştiğine bakalım (Zarf Teoremi):
- $$V'(W) = \frac{1}{W - W'} \Rightarrow V'(W') = \frac{1}{W' - W''} \quad (*2)$$
- $(*1)$ ve $(*2)$ denklemlerini birleştirirsek:
- $$\frac{1}{\beta(W - W')} = \frac{1}{W' - W''}$$

Dinamik Programlama

Kek Yeme Problemi İçin Bellman Denkleminin Yazılması ve Zarf Teoreminin Uygulanması:

- $$V(W) = \max_{0 \leq W' \leq W} [\log(W - W') + \beta V(W')]$$
- FOC w.r.t. W':
- $$\frac{-1}{W - W'} + \beta V'(W') = 0 \quad (*1)$$
- Şimdi de $V(W)$ ifadesinin W ile nasıl değiştiğine bakalım (Zarf Teoremi):
- $$V'(W) = \frac{1}{W - W'} \Rightarrow V'(W') = \frac{1}{W' - W''} \quad (*2)$$
- $(*1)$ ve $(*2)$ denklemlerini birleştirirsek:
- $$\frac{1}{\beta(W - W')} = \frac{1}{W' - W''}$$
- $$\frac{W' - W''}{W - W'} = \beta \Rightarrow \frac{c'}{c} = \beta$$

Dinamik Programlama

Kek Yeme Problemi İçin Bellman Denkleminin Yazılması ve Zarf Teoreminin Uygulanması:

- $$V(W) = \max_{0 \leq W' \leq W} [\log(W - W') + \beta V(W')]$$
- FOC w.r.t. W':
- $$\frac{-1}{W - W'} + \beta V'(W') = 0 \quad (*1)$$
- Şimdi de $V(W)$ ifadesinin W ile nasıl değiştiğine bakalım (Zarf Teoremi):
- $$V'(W) = \frac{1}{W - W'} \Rightarrow V'(W') = \frac{1}{W' - W''} \quad (*2)$$
- $(*1)$ ve $(*2)$ denklemlerini birleştirirsek:
- $$\frac{1}{\beta(W - W')} = \frac{1}{W' - W''}$$
- $$\frac{W' - W''}{W - W'} = \beta \Rightarrow \frac{c'}{c} = \beta$$
- Politika Fonksiyonu: $c' = \beta c$

Dinamik Programlama

Kek Yeme Problemi İçin Bellman Denkleminin Yazılması ve Zarf Teoreminin Uygulanması:

- $$V(W) = \max_{0 \leq W' \leq W} [\log(W - W') + \beta V(W')]$$
- FOC w.r.t. W':
- $$\frac{-1}{W - W'} + \beta V'(W') = 0 \quad (*1)$$
- Şimdi de $V(W)$ ifadesinin W ile nasıl değiştiğine bakalım (Zarf Teoremi):
- $$V'(W) = \frac{1}{W - W'} \Rightarrow V'(W') = \frac{1}{W' - W''} \quad (*2)$$
- $(*1)$ ve $(*2)$ denklemlerini birleştirirsek:
- $$\frac{1}{\beta(W - W')} = \frac{1}{W' - W''}$$
- $$\frac{W' - W''}{W - W'} = \beta \Rightarrow \frac{c'}{c} = \beta$$
- Politika Fonksiyonu: $c' = \beta c$
- $$c_0 + \beta c_0 + \beta^2 c_0 + \ldots = c_0(1 + \beta + \beta^2 + \ldots) = \frac{c_0}{1 - \beta} = W_0$$

Stokastik Süreçlerin Tanımlanması

Stokastik Süreçlerin Tanımlanması: En yaygın olarak iki tip stokastik süreç kullanılmaktadır.

Stokastik Süreçlerin Tanımlanması

Stokastik Süreçlerin Tanımlanması: En yaygın olarak iki tip stokastik süreç kullanılmaktadır.

1. Markov Süreçleri

Stokastik Süreçlerin Tanımlanması

Stokastik Süreçlerin Tanımlanması: En yaygın olarak iki tip stokastik süreç kullanılmaktadır.

1. Markov Süreçleri
2. Otoregresif (AR) Süreçleri

Stokastik Süreçlerin Tanımlanması

Markov Süreçleri

Stokastik Süreçlerin Tanımlanması

Markov Süreçleri

- N (sonlu) sayıda farklı durum (state) olsun: $1, 2, \ldots, i, j, \ldots, N$.

Stokastik Süreçlerin Tanımlanması

Markov Süreçleri

- N (sonlu) sayıda farklı durum (state) olsun: $1, 2, \ldots, i, j, \ldots, N$.
- Markov süreci cari dönemde i. durumda bulunulurken bir sonraki dönemde j. duruma geçme olasılığını yansıtır.

Stokastik Süreçlerin Tanımlanması

Markov Süreçleri

- N (sonlu) sayıda farklı durum (state) olsun: $1, 2, \ldots, i, j, \ldots, N$.
- Markov süreci cari dönemde i. durumda bulunulurken bir sonraki dönemde j. duruma geçme olasılığını yansıtır.
- Bu olasılık π_{ij} şeklinde iafde edilir.

Stokastik Süreçlerin Tanımlanması

Markov Süreçleri

- N (sonlu) sayıda farklı durum (state) olsun: $1, 2, \ldots, i, j, \ldots, N$.
- Markov süreci cari dönemde i. durumda bulunulurken bir sonraki dönemde j. duruma geçme olasılığını yansıtır.
- Bu olasılık π_{ij} şeklinde iafde edilir.
- Markov süreci için tanımlı olan matris şu şekilde ifade edilir:

$$M = \begin{pmatrix} \pi_{11} & \pi_{12} & \cdots & \pi_{1j} & \cdots & \pi_{1N} \\ \pi_{21} & \pi_{22} & \cdots & \pi_{2j} & \cdots & \pi_{2N} \\ \vdots & & & & & \vdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \pi_{i1} & \pi_{i2} & \cdots & \pi_{ij} & \cdots & \pi_{iN} \\ \vdots & & & & & \vdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \pi_{N1} & \pi_{N2} & \cdots & \pi_{Nj} & \cdots & \pi_{NN} \end{pmatrix}$$

Stokastik Süreçlerin Tanımlanması

Markov Süreçleri

- N (sonlu) sayıda farklı durum (state) olsun: $1, 2, \ldots, i, j, \ldots, N$.
- Markov süreci cari dönemde i. durumda bulunulurken bir sonraki dönemde j. duruma geçme olasılığını yansıtır.
- Bu olasılık π_{ij} şeklinde iafde edilir.
- Markov süreci için tanımlı olan matris şu şekilde ifade edilir:

$$M = \begin{pmatrix} \pi_{11} & \pi_{12} & \cdots & \pi_{1j} & \cdots & \pi_{1N} \\ \pi_{21} & \pi_{22} & \cdots & \pi_{2j} & \cdots & \pi_{2N} \\ \vdots & & & & & \vdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \pi_{i1} & \pi_{i2} & \cdots & \pi_{ij} & \cdots & \pi_{iN} \\ \vdots & & & & & \vdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \pi_{N1} & \pi_{N2} & \cdots & \pi_{Nj} & \cdots & \pi_{NN} \end{pmatrix}$$

- Örneğin π_{12} 1. durumdan 2. duruma geçme olasılığıdır.

Stokastik Süreçlerin Tanımlanması

Markov Süreçleri

Stokastik Süreçlerin Tanımlanması

Markov Süreçleri

- $t+1$ döneminde j. duruma geçme olasılığı **sadece bir dönem önceye** yani t döneminde hangi durumda bulunulduğuna bağlı olan bu tip durumlara 1. sıradan Markov süreçleri denir.

Stokastik Süreçlerin Tanımlanması

Markov Süreçleri

- $t+1$ döneminde j. duruma geçme olasılığı **sadece bir dönem önceye** yani t döneminde hangi durumda bulunulduğuna bağlı olan bu tip durumlara 1. sıradan Markov süreçleri denir.
- 1. sıradan Markov süreci için "koşullu olasılık": $P(j_{t+1}/i_t)$.

Stokastik Süreçlerin Tanımlanması

Markov Süreçleri

- $t+1$ döneminde j. duruma geçme olasılığı **sadece bir dönem önceye** yani t döneminde hangi durumda bulunulduğuna bağlı olan bu tip durumlara 1. sıradan Markov süreçleri denir.
- 1. sıradan Markov süreci için "koşullu olasılık": $P(j_{t+1}/i_t)$.
- $P(j_{t+1}/i_{t,t-1,,\ldots,t-n})$ ifadesi ise $t+1$ döneminde j. durumda bulunma olasılığının n'ye kadar olan tüm dönemlere bağlı olduğunu ifade eder ve bu süreçlere de n. sıradan Markov süreçler adı verilir.

Stokastik Süreçlerin Tanımlanması

Durağan Markov Süreçleri: 2 durum olsun.

Stokastik Süreçlerin Tanımlanması

Durağan Markov Süreçleri: 2 durum olsun.

- $$\lim_{t \to \infty} \begin{pmatrix} \pi_{11} & \pi_{12} \\ \pi_{21} & \pi_{22} \end{pmatrix}^t = \begin{pmatrix} a & b \\ a & b \end{pmatrix}$$

  durumuna durağan Markov sürecleri (stationary Markov Process) adı verilir.

Stokastik Süreçlerin Tanımlanması

Durağan Markov Süreçleri: 2 durum olsun.

- $$\lim_{t \to \infty} \begin{pmatrix} \pi_{11} & \pi_{12} \\ \pi_{21} & \pi_{22} \end{pmatrix}^t = \begin{pmatrix} a & b \\ a & b \end{pmatrix}$$

  durumuna durağan Markov sürecleri (stationary Markov Process) adı verilir.
- Hangi durumda olunursa olunsun limitte 1. duruma gitme olasılığı zaman içerisinde a'ya, 2. duruma gitme olasılığı zaman içersinde b'ye yakınsayacaktır.

Dinamik Stokastik Programlama

Dinamik Stokastik Programlama ve Durağan Markov Süreçleri

Dinamik Stokastik Programlama

Dinamik Stokastik Programlama ve Durağan Markov Süreçleri

- Neo-Klasik Büyüme Modelinin Stokastik Ortamda İfade Edilmesi ve Çözümü:

Dinamik Stokastik Programlama

Dinamik Stokastik Programlama ve Durağan Markov Süreçleri

- Neo-Klasik Büyüme Modelinin Stokastik Ortamda İfade Edilmesi ve Çözümü:
- "**Stokastik** Büyüme Modelinde" üretim fonksiyonundaki $A_i k^{\alpha}$ ifadesinde yer alan A_i teriminin rassal bir değişken olduğunu varsayalım.

Dinamik Stokastik Programlama

Dinamik Stokastik Programlama ve Durağan Markov Süreçleri

- Neo-Klasik Büyüme Modelinin Stokastik Ortamda İfade Edilmesi ve Çözümü:
- "**Stokastik** Büyüme Modelinde" üretim fonksiyonundaki $A_i k^{\alpha}$ ifadesinde yer alan A_i teriminin rassal bir değişken olduğunu varsayalım.
- A terimi yoplam faktör verimliliği (total factor productivity) olarak yorumlanabilir.

Dinamik Stokastik Programlama

Dinamik Stokastik Programlama ve Durağan Markov Süreçleri

- Neo-Klasik Büyüme Modelinin Stokastik Ortamda İfade Edilmesi ve Çözümü:
- "**Stokastik** Büyüme Modelinde" üretim fonksiyonundaki $A_i k^{\alpha}$ ifadesinde yer alan A_i teriminin rassal bir değişken olduğunu varsayalım.
- A terimi yoplam faktör verimliliği (total factor productivity) olarak yorumlanabilir.
- $A_1 = 24$ ve $A_2 = 16$ değerlerini alsın yani ekonomide gerçekleşebilecek 2 farklı durum olsun: $N = 2$.

Dinamik Stokastik Programlama

Dinamik Stokastik Programlama ve Durağan Markov Süreçleri

- Neo-Klasik Büyüme Modelinin Stokastik Ortamda İfade Edilmesi ve Çözümü:
- "**Stokastik** Büyüme Modelinde" üretim fonksiyonundaki $A_i k^{\alpha}$ ifadesinde yer alan A_i teriminin rassal bir değişken olduğunu varsayalım.
- A terimi yoplam faktör verimliliği (total factor productivity) olarak yorumlanabilir.
- $A_1 = 24$ ve $A_2 = 16$ değerlerini alsın yani ekonomide gerçekleşebilecek 2 farklı durum olsun: $N = 2$.
- Bu örnekte A_1 ekonomideki iyi durumu yansıtırken, A_2 kötü durumu yansıtmaktadır.

Dinamik Stokastik Programlama

Dinamik Stokastik Programlama ve Durağan Markov Süreçleri

- Neo-Klasik Büyüme Modelinin Stokastik Ortamda İfade Edilmesi ve Çözümü:
- "**Stokastik** Büyüme Modelinde" üretim fonksiyonundaki $A_i k^{\alpha}$ ifadesinde yer alan A_i teriminin rassal bir değişken olduğunu varsayalım.
- A terimi yoplam faktör verimliliği (total factor productivity) olarak yorumlanabilir.
- $A_1 = 24$ ve $A_2 = 16$ değerlerini alsın yani ekonomide gerçekleşebilecek 2 farklı durum olsun: $N = 2$.
- Bu örnekte A_1 ekonomideki iyi durumu yansıtırken, A_2 kötü durumu yansıtmaktadır.
- Bu rassal değişkenin aşağıda belirtilen 1. sıra Markov süreci izlediği varsayılsın:

$$\begin{pmatrix} \pi_{11} & \pi_{12} \\ \pi_{21} & \pi_{22} \end{pmatrix} = \begin{pmatrix} 0.8 & 0.2 \\ 0.3 & 0.7 \end{pmatrix}$$

Dinamik Stokastik Programlama

Dinamik Stokastik Programlama ve Durağan Markov Süreçleri: Birinci durumu iyi ve ikinci durumu kötü şok olarak nitelendirirsek koşullu olasılıkları şöyle ifade edebiliriz:

Dinamik Stokastik Programlama

Dinamik Stokastik Programlama ve Durağan Markov Süreçleri: Birinci durumu iyi ve ikinci durumu kötü şok olarak nitelendirirsek koşullu olasılıkları şöyle ifade edebiliriz:

- $\pi_{11} = P(1|1) = 0.8$ (iyiyken iyi kalma olasılığı)

Dinamik Stokastik Programlama

Dinamik Stokastik Programlama ve Durağan Markov Süreçleri: Birinci durumu iyi ve ikinci durumu kötü şok olarak nitelendirirsek koşullu olasılıkları şöyle ifade edebiliriz:

- $\pi_{11} = P(1|1) = 0.8$ (iyiyken iyi kalma olasılığı)
- $\pi_{21} = P(1|2) = 0.3$ (kötüyken iyi olma olasılığı)

Dinamik Stokastik Programlama

Dinamik Stokastik Programlama ve Durağan Markov Süreçleri: Birinci durumu iyi ve ikinci durumu kötü şok olarak nitelendirirsek koşullu olasılıkları şöyle ifade edebiliriz:

- $\pi_{11} = P(1|1) = 0.8$ (iyiyken iyi kalma olasılığı)
- $\pi_{21} = P(1|2) = 0.3$ (kötüyken iyi olma olasılığı)
- $\pi_{12} = P(2|1) = 0.2$ (iyiyken kötü olma olasılığı)

Dinamik Stokastik Programlama

Dinamik Stokastik Programlama ve Durağan Markov Süreçleri: Birinci durumu iyi ve ikinci durumu kötü şok olarak nitelendirirsek koşullu olasılıkları şöyle ifade edebiliriz:

- $\pi_{11} = P(1|1) = 0.8$ (iyiyken iyi kalma olasılığı)
- $\pi_{21} = P(1|2) = 0.3$ (kötüyken iyi olma olasılığı)
- $\pi_{12} = P(2|1) = 0.2$ (iyiyken kötü olma olasılığı)
- $\pi_{22} = P(2|2) = 0.7$ (kötüyken kötü kalma olasılığı)

Dinamik Stokastik Programlama

Dinamik Stokastik Programlama ve Durağan Markov Süreçleri: Bu durumda dinamik programla problemini şu sekilde yazabiliriz:

Dinamik Stokastik Programlama

Dinamik Stokastik Programlama ve Durağan Markov Süreçleri: Bu durumda dinamik programla problemini şu sekilde yazabiliriz:

- $$V(k, A) = \max_{c,k'} \log(c) + \beta E(V(k', A'))$$

Dinamik Stokastik Programlama

Dinamik Stokastik Programlama ve Durağan Markov Süreçleri: Bu durumda dinamik programla problemini şu sekilde yazabiliriz:

- $$V(k, A) = \max_{c,k'} \log(c) + \beta E(V(k', A'))$$
- s.t.
$$c + k' \leq Ak^{\alpha}$$

Dinamik Stokastik Programlama

Dinamik Stokastik Programlama ve Durağan Markov Süreçleri: Bu durumda dinamik programla problemini şu sekilde yazabiliriz:

- $$V(k, A) = \max_{c,k'} \log(c) + \beta E(V(k', A'))$$
- s.t.
$$c + k' \leq Ak^{\alpha}$$
- $$c, k' \geq 0$$

Dinamik Stokastik Programlama

Dinamik Stokastik Programlama ve Durağan Markov Süreçleri: Bu durumda dinamik programla problemini şu sekilde yazabiliriz:

- $$V(k, A) = \max_{c,k'} \log(c) + \beta E(V(k', A'))$$
- s.t.
$$c + k' \leq Ak^{\alpha}$$
- $$c, k' \geq 0$$
- Yukarıdaki problemin deterministik ortamda belirtilen dinamik programlamadan farklarını belirtelim:

Dinamik Stokastik Programlama

Dinamik Stokastik Programlama ve Durağan Markov Süreçleri:

Dinamik Stokastik Programlama

Dinamik Stokastik Programlama ve Durağan Markov Süreçleri: Yukarıdaki problemin deterministik ortamda belirtilen dinamik programlamadan farklarını belirtelim:

Dinamik Stokastik Programlama

Dinamik Stokastik Programlama ve Durağan Markov Süreçleri: Yukarıdaki problemin deterministik ortamda belirtilen dinamik programlamadan farklarını belirtelim:

1. Artık durum değişkeni sadece k değildir. **Durum değişkenleri hem elde bulunan sermaye stoku** (k)**, hem de o an içinde bulunan durumu yansıtan** A**'dır**. Dolayısıyla bir sonraki dönemde sermaye stokunu (k') seçmek için cari dönemdeki k ile birlikte yine cari dönemdeki A değerlerinin bilgisine ihtiyaç vardır.

Dinamik Stokastik Programlama

Dinamik Stokastik Programlama ve Durağan Markov Süreçleri: Yukarıdaki problemin deterministik ortamda belirtilen dinamik programlamadan farklarını belirtelim:

1. Artık durum değişkeni sadece k değildir. **Durum değişkenleri hem elde bulunan sermaye stoku** (k)**, hem de o an içinde bulunan durumu yansıtan** A**'dır**. Dolayısıyla bir sonraki dönemde sermaye stokunu (k') seçmek için cari dönemdeki k ile birlikte yine cari dönemdeki A değerlerinin bilgisine ihtiyaç vardır.
2. Cari dönemde A bilinse de, bir sonraki ve diğer tüm dönemlerde A rassal olarak belirleneceği için gelecek dönemlerdeki değerler için **beklenen değer** ifadesi ve hesabı kullanılacaktır.

Dinamik Stokastik Programlama

Dinamik Stokastik Programlama ve Durağan Markov Süreçleri: Yukarıdaki problemin deterministik ortamda belirtilen dinamik programlamadan farklarını belirtelim:

1. Artık durum değişkeni sadece k değildir. **Durum değişkenleri hem elde bulunan sermaye stoku** (k)**, hem de o an içinde bulunan durumu yansıtan** A**'dır**. Dolayısıyla bir sonraki dönemde sermaye stokunu (k') seçmek için cari dönemdeki k ile birlikte yine cari dönemdeki A değerlerinin bilgisine ihtiyaç vardır.
2. Cari dönemde A bilinse de, bir sonraki ve diğer tüm dönemlerde A rassal olarak belirleneceği için gelecek dönemlerdeki değerler için **beklenen değer** ifadesi ve hesabı kullanılacaktır.
3. Ekonomide 2 durum olduğu için **2 politika fonksiyonu** olacak: $g(k, A_1)$,$g(k, A_2)$.

Dinamik Stokastik Programlama

Dinamik Stokastik Programlama ve Durağan Markov Süreçleri: Yukarıdaki problemin deterministik ortamda belirtilen dinamik programlamadan farklarını belirtelim:

1. Artık durum değişkeni sadece k değildir. **Durum değişkenleri hem elde bulunan sermaye stoku** (k)**, hem de o an içinde bulunan durumu yansıtan** A**'dır**. Dolayısıyla bir sonraki dönemde sermaye stokunu (k') seçmek için cari dönemdeki k ile birlikte yine cari dönemdeki A değerlerinin bilgisine ihtiyaç vardır.
2. Cari dönemde A bilinse de, bir sonraki ve diğer tüm dönemlerde A rassal olarak belirleneceği için gelecek dönemlerdeki değerler için **beklenen değer** ifadesi ve hesabı kullanılacaktır.
3. Ekonomide 2 durum olduğu için **2 politika fonksiyonu** olacak: $g(k, A_1)$,$g(k, A_2)$.

Bu problem "Tahmin ve Doğrulama yöntemiyle" analitik olarak çözülebilir. Fakat daha genel çözümlerde ise "Stokastik ortamda değer fonksiyonu iterasyonu" yöntemini (MATLAB kullanarak) kullanmamız gerekir.

Dinamik Stokastik Programlama

Dinamik Stokastik Programlama ve Durağan Markov Süreçleri:

Dinamik Stokastik Programlama

Dinamik Stokastik Programlama ve Durağan Markov Süreçleri: Yukarıda belirtilen problemi $c + k' = Ak^{\alpha}$ eşitliğini kullanarak 3 yerine 2 değişkenle ifade edilecek şekilde yazabiliriz:

Dinamik Stokastik Programlama

Dinamik Stokastik Programlama ve Durağan Markov Süreçleri: Yukarıda belirtilen problemi $c + k' = Ak^{\alpha}$ eşitliğini kullanarak 3 yerine 2 değişkenle ifade edilecek şekilde yazabiliriz:

$$V(k, A) = \max_{0 \leq k' \leq Ak^{\alpha}} \log(Ak^{\alpha} - k') + \beta E(V(k', A'))$$

Dinamik Stokastik Programlama

Dinamik Stokastik Programlama ve Durağan Markov Süreçleri: Yukarıda belirtilen problemi $c + k' = Ak^{\alpha}$ eşitliğini kullanarak 3 yerine 2 değişkenle ifade edilecek şekilde yazabiliriz:

- $$V(k, A) = \max_{0 \leq k' \leq Ak^{\alpha}} \log(Ak^{\alpha} - k') + \beta E(V(k', A'))$$

- Bu problemi 2 parçaya ayırabiliriz.

Dinamik Stokastik Programlama

Dinamik Stokastik Programlama ve Durağan Markov Süreçleri: Yukarıda belirtilen problemi $c + k' = Ak^{\alpha}$ eşitliğini kullanarak 3 yerine 2 değişkenle ifade edilecek şekilde yazabiliriz:

- $$V(k, A) = \max_{0 \leq k' \leq Ak^{\alpha}} \log(Ak^{\alpha} - k') + \beta E(V(k', A'))$$

- Bu problemi 2 parçaya ayırabiliriz.
- Yani genel olarak $V(k, A)$ yerine A_1 ve A_2 için ayrı ayrı 2 Bellman denklemi yazabiliriz:

Dinamik Stokastik Programlama

Dinamik Stokastik Programlama ve Durağan Markov Süreçleri: Yukarıda belirtilen problemi $c + k' = Ak^{\alpha}$ eşitliğini kullanarak 3 yerine 2 değişkenle ifade edilecek şekilde yazabiliriz:

- $$V(k, A) = \max_{0 \leq k' \leq Ak^{\alpha}} \log(Ak^{\alpha} - k') + \beta E(V(k', A'))$$

- Bu problemi 2 parçaya ayırabiliriz.
- Yani genel olarak $V(k, A)$ yerine A_1 ve A_2 için ayrı ayrı 2 Bellman denklemi yazabiliriz:
- Mevcut sermaye stoku k ve A_i için:

$$V(k, A_i) = \max_{0 \leq k' \leq A_i k^{\alpha}} \log(A_i k^{\alpha} - k') + \beta \left(\sum_{j=1}^{2} \pi_{ij} V(k', A_j) \right)$$

Dinamik Stokastik Programlama

Dinamik Stokastik Programlama ve Durağan Markov Süreçleri: Yukarıda belirtilen problemi $c + k' = Ak^{\alpha}$ eşitliğini kullanarak 3 yerine 2 değişkenle ifade edilecek şekilde yazabiliriz:

- $$V(k, A) = \max_{0 \leq k' \leq Ak^{\alpha}} \log(Ak^{\alpha} - k') + \beta E(V(k', A'))$$

- Bu problemi 2 parçaya ayırabiliriz.
- Yani genel olarak $V(k, A)$ yerine A_1 ve A_2 için ayrı ayrı 2 Bellman denklemi yazabiliriz:
- Mevcut sermaye stoku k ve A_i için:

$$V(k, A_i) = \max_{0 \leq k' \leq A_i k^{\alpha}} \log(A_i k^{\alpha} - k') + \beta \left(\sum_{j=1}^{2} \pi_{ij} V(k', A_j) \right)$$

- Bu ifadeyi her bir durum için açıkça tekrar yazarsak:

Dinamik Stokastik Programlama

Dinamik Stokastik Programlama ve Durağan Markov Süreçleri: Yukarıda belirtilen problemi $c + k' = Ak^{\alpha}$ eşitliğini kullanarak 3 yerine 2 değişkenle ifade edilecek şekilde yazabiliriz:

- $$V(k, A) = \max_{0 \leq k' \leq Ak^{\alpha}} \log(Ak^{\alpha} - k') + \beta E(V(k', A'))$$

- Bu problemi 2 parçaya ayırabiliriz.
- Yani genel olarak $V(k, A)$ yerine A_1 ve A_2 için ayrı ayrı 2 Bellman denklemi yazabiliriz:
- Mevcut sermaye stoku k ve A_i için:

$$V(k, A_i) = \max_{0 \leq k' \leq A_i k^{\alpha}} \log(A_i k^{\alpha} - k') + \beta \left(\sum_{j=1}^{2} \pi_{ij} V(k', A_j) \right)$$

- Bu ifadeyi her bir durum için açıkça tekrar yazarsak:
- Elinde k stoku olan A_1 durumundaki bir kişi için Bellman denklemi:

$$V(k, A_1) = \max_{0 \leq k' \leq A_1 k^{\alpha}} \log(A_1 k^{\alpha} - k') + \beta \left(\pi_{11} V(k', A_1) + \pi_{12} V(k', A_2) \right) \qquad (1)$$

Dinamik Stokastik Programlama

Dinamik Stokastik Programlama ve Durağan Markov Süreçleri: Yukarıda belirtilen problemi $c + k' = Ak^{\alpha}$ eşitliğini kullanarak 3 yerine 2 değişkenle ifade edilecek şekilde yazabiliriz:

- $$V(k, A) = \max_{0 \leq k' \leq Ak^{\alpha}} \log(Ak^{\alpha} - k') + \beta E(V(k', A'))$$

- Bu problemi 2 parçaya ayırabiliriz.
- Yani genel olarak $V(k, A)$ yerine A_1 ve A_2 için ayrı ayrı 2 Bellman denklemi yazabiliriz:
- Mevcut sermaye stoku k ve A_i için:

$$V(k, A_i) = \max_{0 \leq k' \leq A_i k^{\alpha}} \log(A_i k^{\alpha} - k') + \beta \left(\sum_{j=1}^{2} \pi_{ij} V(k', A_j) \right)$$

- Bu ifadeyi her bir durum için açıkça tekrar yazarsak:
- Elinde k stoku olan A_1 durumundaki bir kişi için Bellman denklemi:

$$V(k, A_1) = \max_{0 \leq k' \leq A_1 k^{\alpha}} \log(A_1 k^{\alpha} - k') + \beta \left(\pi_{11} V(k', A_1) + \pi_{12} V(k', A_2) \right) \qquad (1)$$

- Elinde k stoku olan A_2 durumundaki bir kişi için Bellman denklemi:

$$V(k, A_2) = \max_{0 \leq k' \leq A_2 k^{\alpha}} \log(A_2 k^{\alpha} - k') + \beta \left(\pi_{21} V(k', A_1) + \pi_{22} V(k', A_2) \right) \qquad (2)$$

Dinamik Stokastik Programlama

Dinamik Stokastik Programlama ve Durağan Markov Süreçleri:

Dinamik Stokastik Programlama

Dinamik Stokastik Programlama ve Durağan Markov Süreçleri: Bu durumda tahmin denklemi şu şekilde olacaktır:

Dinamik Stokastik Programlama

Dinamik Stokastik Programlama ve Durağan Markov Süreçleri: Bu durumda tahmin denklemi şu şekilde olacaktır:

- $$V(k, A_i) = h_0 + h_1 log(k) + b_{i1} log(A_1) + b_{i2} log(A_2)$$

Dinamik Stokastik Programlama

Dinamik Stokastik Programlama ve Durağan Markov Süreçleri: Bu durumda tahmin denklemi şu şekilde olacaktır:

- $$V(k, A_i) = h_0 + h_1 log(k) + b_{i1} log(A_1) + b_{i2} log(A_2)$$
- Dolayısıyla
$$V(k', A_i) = h_0 + h_1 log(k') + b_{i1} log(A_1) + b_{i2} log(A_2)$$

Dinamik Stokastik Programlama

Dinamik Stokastik Programlama ve Durağan Markov Süreçleri: Bu durumda tahmin denklemi şu şekilde olacaktır:

- $$V(k, A_i) = h_0 + h_1 log(k) + b_{i1} log(A_1) + b_{i2} log(A_2)$$
- Dolayısıyla
$$V(k', A_i) = h_0 + h_1 log(k') + b_{i1} log(A_1) + b_{i2} log(A_2)$$
- Şimdi bu ikinci tahmin denklemini kullanarak 1. Bellman denklem için optimal k' değerini bulalım:

Dinamik Stokastik Programlama

Dinamik Stokastik Programlama ve Durağan Markov Süreçleri: Bu durumda tahmin denklemi şu şekilde olacaktır:

- $$V(k, A_i) = h_0 + h_1 log(k) + b_{i1} log(A_1) + b_{i2} log(A_2)$$
- Dolayısıyla
$$V(k', A_i) = h_0 + h_1 log(k') + b_{i1} log(A_1) + b_{i2} log(A_2)$$
- Şimdi bu ikinci tahmin denklemini kullanarak 1. Bellman denklem için optimal k' değerini bulalım:
-

$$V(k, A_1) = \max_{0 \leq k' \leq A_1 k^{\alpha}} \log(A_1 k^{\alpha} - k') + \beta(\pi_{11}[h_0 + h_1 log(k') + b_{11} log(A_1) + b_{12} log(A_2)] +$$

$$\pi_{12}[h_0 + h_1 log(k') + b_{21} log(A_1) + b_{22} log(A_2)] \qquad (*)$$

Dinamik Stokastik Programlama

Dinamik Stokastik Programlama ve Durağan Markov Süreçleri: Bu durumda tahmin denklemi şu şekilde olacaktır:

- $$V(k, A_i) = h_0 + h_1 log(k) + b_{i1} log(A_1) + b_{i2} log(A_2)$$
- Dolayısıyla
$$V(k', A_i) = h_0 + h_1 log(k') + b_{i1} log(A_1) + b_{i2} log(A_2)$$
- Şimdi bu ikinci tahmin denklemini kullanarak 1. Bellman denklem için optimal k' değerini bulalım:
- $$V(k, A_1) = \max_{0 \leq k' \leq A_1 k^{\alpha}} \log(A_1 k^{\alpha} - k') + \beta(\pi_{11}[h_0 + h_1 log(k') + b_{11} log(A_1) + b_{12} log(A_2)] +$$
$$\pi_{12}[h_0 + h_1 log(k') + b_{21} log(A_1) + b_{22} log(A_2)] \qquad (*)$$
- F.O.C. k' için:
$$\frac{-1}{A_1 k^{\alpha} - k'} + \frac{\beta \pi_{11} h_1}{k'} + \frac{\beta \pi_{12} h_1}{k'} = 0$$

Dinamik Stokastik Programlama

Dinamik Stokastik Programlama ve Durağan Markov Süreçleri: Bu durumda tahmin denklemi şu şekilde olacaktır:

- $$V(k, A_i) = h_0 + h_1 log(k) + b_{i1} log(A_1) + b_{i2} log(A_2)$$
- Dolayısıyla
$$V(k', A_i) = h_0 + h_1 log(k') + b_{i1} log(A_1) + b_{i2} log(A_2)$$
- Şimdi bu ikinci tahmin denklemini kullanarak 1. Bellman denklem için optimal k' değerini bulalım:
- $$V(k, A_1) = \max_{0 \leq k' \leq A_1 k^\alpha} \log(A_1 k^\alpha - k') + \beta(\pi_{11}[h_0 + h_1 log(k') + b_{11} log(A_1) + b_{12} log(A_2)] +$$
$$\pi_{12}[h_0 + h_1 log(k') + b_{21} log(A_1) + b_{22} log(A_2)] \qquad (*)$$
- F.O.C. k' için:
$$\frac{-1}{A_1 k^\alpha - k'} + \frac{\beta \pi_{11} h_1}{k'} + \frac{\beta \pi_{12} h_1}{k'} = 0$$
- $$\frac{\beta \pi_{11} h_1 + \beta \pi_{12} h_1}{k'} = \frac{1}{A_1 k^\alpha - k'}$$

Dinamik Stokastik Programlama

Dinamik Stokastik Programlama ve Durağan Markov Süreçleri: Bu durumda tahmin denklemi şu şekilde olacaktır:

- $$V(k, A_i) = h_0 + h_1 log(k) + b_{i1} log(A_1) + b_{i2} log(A_2)$$
- Dolayısıyla
$$V(k', A_i) = h_0 + h_1 log(k') + b_{i1} log(A_1) + b_{i2} log(A_2)$$
- Şimdi bu ikinci tahmin denklemini kullanarak 1. Bellman denklem için optimal k' değerini bulalım:
- $$V(k, A_1) = \max_{0 \leq k' \leq A_1 k^\alpha} \log(A_1 k^\alpha - k') + \beta(\pi_{11}[h_0 + h_1 log(k') + b_{11} log(A_1) + b_{12} log(A_2)] +$$
$$\pi_{12}[h_0 + h_1 log(k') + b_{21} log(A_1) + b_{22} log(A_2)] \qquad (*)$$
- F.O.C. k' için:
$$\frac{-1}{A_1 k^\alpha - k'} + \frac{\beta \pi_{11} h_1}{k'} + \frac{\beta \pi_{12} h_1}{k'} = 0$$
- $$\frac{\beta \pi_{11} h_1 + \beta \pi_{12} h_1}{k'} = \frac{1}{A_1 k^\alpha - k'}$$
- $$k' = \frac{A_1 k^\alpha (\beta h_1 (\pi_{11} + \pi_{12}))}{1 + \beta h_1 (\pi_{11} + \pi_{12})} \qquad (*2)$$

Dinamik Stokastik Programlama

Dinamik Stokastik Programlama ve Durağan Markov Süreçleri:

Dinamik Stokastik Programlama

Dinamik Stokastik Programlama ve Durağan Markov Süreçleri:

- Optimal k' bilgisini (yani *2 denklemini) ve $V(k, A_1)$ tahminini kullanarak $(*)$ denklemini yeniden yazarsak:

Dinamik Stokastik Programlama

Dinamik Stokastik Programlama ve Durağan Markov Süreçleri:

- Optimal k' bilgisini (yani *2 denklemini) ve $V(k, A_1)$ tahminini kullanarak $(*)$ denklemini yeniden yazarsak:
-

$$h_0 + h_1 log(k) + b_{11} log(A_1) + b_{12} log(A_2) =$$

Dinamik Stokastik Programlama

Dinamik Stokastik Programlama ve Durağan Markov Süreçleri:

- Optimal k' bilgisini (yani *2 denklemini) ve $V(k, A_1)$ tahminini kullanarak $(*)$ denklemini yeniden yazarsak:
- $$h_0 + h_1 log(k) + b_{11} log(A_1) + b_{12} log(A_2) =$$
- $$\log(A_1 k^{\alpha} - [\frac{A_1 k^{\alpha}(\beta h_1(\pi_{11} + \pi_{12}))}{1 + \beta h_1(\pi_{11} + \pi_{12})}]) + \beta(\pi_{11}[h_0 + h_1 log([\frac{A_1 k^{\alpha}(\beta h_1(\pi_{11} + \pi_{12}))}{1 + \beta h_1(\pi_{11} + \pi_{12})}]) + b_{11} log(A_1) + b_{12}$$
$$\pi_{12}[h_0 + h_1 log([\frac{A_1 k^{\alpha}(\beta h_1(\pi_{11} + \pi_{12}))}{1 + \beta h_1(\pi_{11} + \pi_{12})}]) + b_{21} log(A_1) + b_{22} log(A_2)]$$

Dinamik Stokastik Programlama

Dinamik Stokastik Programlama ve Durağan Markov Süreçleri:

- Optimal k' bilgisini (yani *2 denklemini) ve $V(k, A_1)$ tahminini kullanarak $(*)$ denklemini yeniden yazarsak:
- $$h_0 + h_1 log(k) + b_{11} log(A_1) + b_{12} log(A_2) =$$
- $$\log(A_1 k^{\alpha} - [\frac{A_1 k^{\alpha}(\beta h_1(\pi_{11} + \pi_{12}))}{1 + \beta h_1(\pi_{11} + \pi_{12})}]) + \beta(\pi_{11}[h_0 + h_1 log([\frac{A_1 k^{\alpha}(\beta h_1(\pi_{11} + \pi_{12}))}{1 + \beta h_1(\pi_{11} + \pi_{12})}]) + b_{11} log(A_1) + b_{12}$$
$$\pi_{12}[h_0 + h_1 log([\frac{A_1 k^{\alpha}(\beta h_1(\pi_{11} + \pi_{12}))}{1 + \beta h_1(\pi_{11} + \pi_{12})}]) + b_{21} log(A_1) + b_{22} log(A_2)]$$
- Burada eşitliğin sağında ve solunda $log(k)$ önünde bulunan terimler birbirine eşitlenirse h_1 terimini elde edebiliriz:

Dinamik Stokastik Programlama

Dinamik Stokastik Programlama ve Durağan Markov Süreçleri:

- Optimal k' bilgisini (yani *2 denklemini) ve $V(k, A_1)$ tahminini kullanarak $(*)$ denklemini yeniden yazarsak:
- $$h_0 + h_1 log(k) + b_{11} log(A_1) + b_{12} log(A_2) =$$
- $$\log(A_1 k^{\alpha} - [\frac{A_1 k^{\alpha}(\beta h_1(\pi_{11} + \pi_{12}))}{1 + \beta h_1(\pi_{11} + \pi_{12})}]) + \beta(\pi_{11}[h_0 + h_1 log([\frac{A_1 k^{\alpha}(\beta h_1(\pi_{11} + \pi_{12}))}{1 + \beta h_1(\pi_{11} + \pi_{12})}]) + b_{11} log(A_1) + b_{12}$$
$$\pi_{12}[h_0 + h_1 log([\frac{A_1 k^{\alpha}(\beta h_1(\pi_{11} + \pi_{12}))}{1 + \beta h_1(\pi_{11} + \pi_{12})}]) + b_{21} log(A_1) + b_{22} log(A_2)]$$
- Burada eşitliğin sağında ve solunda $log(k)$ önünde bulunan terimler birbirine eşitlenirse h_1 terimini elde edebiliriz:
- $$h_1 = \frac{\alpha}{1 - \beta\alpha(\pi_{11} + \pi_{12})}$$

Dinamik Stokastik Programlama

Dinamik Stokastik Programlama ve Durağan Markov Süreçleri:

- Optimal k' bilgisini (yani *2 denklemini) ve $V(k, A_1)$ tahminini kullanarak $(*)$ denklemini yeniden yazarsak:
- $$h_0 + h_1 log(k) + b_{11} log(A_1) + b_{12} log(A_2) =$$
- $$\log(A_1 k^{\alpha} - [\frac{A_1 k^{\alpha}(\beta h_1(\pi_{11} + \pi_{12}))}{1 + \beta h_1(\pi_{11} + \pi_{12})}]) + \beta(\pi_{11}[h_0 + h_1 log([\frac{A_1 k^{\alpha}(\beta h_1(\pi_{11} + \pi_{12}))}{1 + \beta h_1(\pi_{11} + \pi_{12})}]) + b_{11} log(A_1) + b_{12}$$
$$\pi_{12}[h_0 + h_1 log([\frac{A_1 k^{\alpha}(\beta h_1(\pi_{11} + \pi_{12}))}{1 + \beta h_1(\pi_{11} + \pi_{12})}]) + b_{21} log(A_1) + b_{22} log(A_2)]$$
- Burada eşitliğin sağında ve solunda $log(k)$ önünde bulunan terimler birbirine eşitlenirse h_1 terimini elde edebiliriz:
- $$h_1 = \frac{\alpha}{1 - \beta\alpha(\pi_{11} + \pi_{12})}$$
- Dolayısıyla optimal k' yani politika fonksiyonunu ifade eden $(*2)$ denkleminde h_1 yerine yazılırsa:

Dinamik Stokastik Programlama

Dinamik Stokastik Programlama ve Durağan Markov Süreçleri:

- Optimal k' bilgisini (yani *2 denklemini) ve $V(k, A_1)$ tahminini kullanarak $(*)$ denklemini yeniden yazarsak:
- $$h_0 + h_1 log(k) + b_{11} log(A_1) + b_{12} log(A_2) =$$
- $$\log(A_1 k^{\alpha} - [\frac{A_1 k^{\alpha}(\beta h_1(\pi_{11} + \pi_{12}))}{1 + \beta h_1(\pi_{11} + \pi_{12})}]) + \beta(\pi_{11}[h_0 + h_1 log([\frac{A_1 k^{\alpha}(\beta h_1(\pi_{11} + \pi_{12}))}{1 + \beta h_1(\pi_{11} + \pi_{12})}]) + b_{11} log(A_1) + b_{12}$$
$$\pi_{12}[h_0 + h_1 log([\frac{A_1 k^{\alpha}(\beta h_1(\pi_{11} + \pi_{12}))}{1 + \beta h_1(\pi_{11} + \pi_{12})}]) + b_{21} log(A_1) + b_{22} log(A_2)]$$
- Burada eşitliğin sağında ve solunda $log(k)$ önünde bulunan terimler birbirine eşitlenirse h_1 terimini elde edebiliriz:
- $$h_1 = \frac{\alpha}{1 - \beta\alpha(\pi_{11} + \pi_{12})}$$
- Dolayısıyla optimal k' yani politika fonksiyonunu ifade eden $(*2)$ denkleminde h_1 yerine yazılırsa:
- $$k' = \frac{A_1 k^{\alpha}(\beta[\frac{\alpha}{1-\beta\alpha((\pi_{11}+\pi_{12}))}](\pi_{11} + \pi_{12}))}{1 + \beta[\frac{\alpha}{1-\beta\alpha((\pi_{11}+\pi_{12}))}](\pi_{11} + \pi_{12})}$$

Dinamik Stokastik Programlama

Dinamik Stokastik Programlama ve Durağan Markov Süreçleri:

- Optimal k' bilgisini (yani *2 denklemini) ve $V(k, A_1)$ tahminini kullanarak $(*)$ denklemini yeniden yazarsak:
- $$h_0 + h_1 log(k) + b_{11} log(A_1) + b_{12} log(A_2) =$$
- $$\log(A_1 k^{\alpha} - [\frac{A_1 k^{\alpha}(\beta h_1(\pi_{11} + \pi_{12}))}{1 + \beta h_1(\pi_{11} + \pi_{12})}]) + \beta(\pi_{11}[h_0 + h_1 log([\frac{A_1 k^{\alpha}(\beta h_1(\pi_{11} + \pi_{12}))}{1 + \beta h_1(\pi_{11} + \pi_{12})}]) + b_{11} log(A_1) + b_{12}$$
$$\pi_{12}[h_0 + h_1 log([\frac{A_1 k^{\alpha}(\beta h_1(\pi_{11} + \pi_{12}))}{1 + \beta h_1(\pi_{11} + \pi_{12})}]) + b_{21} log(A_1) + b_{22} log(A_2)]$$
- Burada eşitliğin sağında ve solunda $log(k)$ önünde bulunan terimler birbirine eşitlenirse h_1 terimini elde edebiliriz:
- $$h_1 = \frac{\alpha}{1 - \beta\alpha(\pi_{11} + \pi_{12})}$$
- Dolayısıyla optimal k' yani politika fonksiyonunu ifade eden $(*2)$ denkleminde h_1 yerine yazılırsa:
- $$k' = \frac{A_1 k^{\alpha}(\beta[\frac{\alpha}{1-\beta\alpha((\pi_{11}+\pi_{12}))}](\pi_{11} + \pi_{12}))}{1 + \beta[\frac{\alpha}{1-\beta\alpha((\pi_{11}+\pi_{12}))}](\pi_{11} + \pi_{12})}$$
- $\pi_{11} + \pi_{12} = 1$ olduğu hatırlanmalıdır. Ayrıca, benzer işlemler yapılarak 2. Bellman denklemi için de k' elde edilir.

Dinamik Stokastik Programlama

Dinamik Stokastik Programlama ve Durağan Markov Süreçleri:

Dinamik Stokastik Programlama

Dinamik Stokastik Programlama ve Durağan Markov Süreçleri: Sayısal Örnek

Dinamik Stokastik Programlama

Dinamik Stokastik Programlama ve Durağan Markov Süreçleri: Sayısal Örnek

- Stokastik Neo-Klasik büyüme modelinde $A_1 = 24, A_2 = 16, \beta = 0.6, \alpha = 0.3$ ve 1. sıra Markov sürecinin $\begin{pmatrix} \pi_{11} & \pi_{12} \\ \pi_{21} & \pi_{22} \end{pmatrix} = \begin{pmatrix} 0.8 & 0.2 \\ 0.3 & 0.7 \end{pmatrix}$ oldugunu varsayalım.

Dinamik Stokastik Programlama

Dinamik Stokastik Programlama ve Durağan Markov Süreçleri: Sayısal Örnek

- Stokastik Neo-Klasik büyüme modelinde $A_1 = 24, A_2 = 16, \beta = 0.6, \alpha = 0.3$ ve 1. sıra Markov sürecinin $\begin{pmatrix} \pi_{11} & \pi_{12} \\ \pi_{21} & \pi_{22} \end{pmatrix} = \begin{pmatrix} 0.8 & 0.2 \\ 0.3 & 0.7 \end{pmatrix}$ oldugunu varsayalım.
- $g(k, 24) = 4.32k^{0.3}$ (1. politika fonksiyonu) sonucuna ulasılır.

Dinamik Stokastik Programlama

Dinamik Stokastik Programlama ve Durağan Markov Süreçleri: Sayısal Örnek

- Stokastik Neo-Klasik büyüme modelinde $A_1 = 24, A_2 = 16, \beta = 0.6, \alpha = 0.3$ ve 1. sıra Markov sürecinin $\begin{pmatrix} \pi_{11} & \pi_{12} \\ \pi_{21} & \pi_{22} \end{pmatrix} = \begin{pmatrix} 0.8 & 0.2 \\ 0.3 & 0.7 \end{pmatrix}$ oldugunu varsayalım.
- $g(k, 24) = 4.32k^{0.3}$ (1. politika fonksiyonu) sonucuna ulasılır.
- Benzer bir şekilde 2. Bellman denklemi için tüm işlemler tekrar edilir ve degğrler yerine yazilirsa $g(k, 16) = 2.88k^{0.3}$ (2. politika fonksiyonu) sonucuna ulasılır.

Dinamik Stokastik Programlama

Dinamik Stokastik Programlama ve Durağan Markov Süreçleri: Sayısal Örnek

- Stokastik Neo-Klasik büyüme modelinde $A_1 = 24, A_2 = 16, \beta = 0.6, \alpha = 0.3$ ve 1. sıra Markov sürecinin $\begin{pmatrix} \pi_{11} & \pi_{12} \\ \pi_{21} & \pi_{22} \end{pmatrix} = \begin{pmatrix} 0.8 & 0.2 \\ 0.3 & 0.7 \end{pmatrix}$ oldugunu varsayalım.
- $g(k, 24) = 4.32k^{0.3}$ (1. politika fonksiyonu) sonucuna ulasılır.
- Benzer bir şekilde 2. Bellman denklemi için tüm işlemler tekrar edilir ve degğrler yerine yazilirsa $g(k, 16) = 2.88k^{0.3}$ (2. politika fonksiyonu) sonucuna ulasılır.
- Bu denklemlerin anlam elinde k sermaye stoku bulunan bir kişi eğer iyi dönemde ise gelecek dönem için olan sermaye birikimini 1. politika fonksiyonuna göre yapması iken, yine eninde k sermaye stoku bulunan bir kişi eğer kötü dönemde ise sermaye stoku tasarrufunu 2. politika fonksiyonuna gore yapacak olmasıdır.