Structural Models/Recursive Strategic Models

Modeling Utility Functions

William O'Brochta

Summary

Key Motivations

- Disconnect between game theoretic models and their empirical tests.

Key Motivations

- Disconnect between game theoretic models and their empirical tests.
- Game theoretic models are typically tested indirectly.

Key Motivations

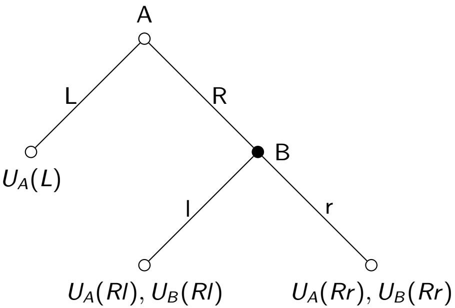
- Disconnect between game theoretic models and their empirical tests.
- Game theoretic models are typically tested indirectly.
- Previous research has developed direct tests, but implementation was complicated.

Key Motivations

- Disconnect between game theoretic models and their empirical tests.
- Game theoretic models are typically tested indirectly.
- Previous research has developed direct tests, but implementation was complicated.
- Goal: use logit models to "backward induct" estimates of utility from data.

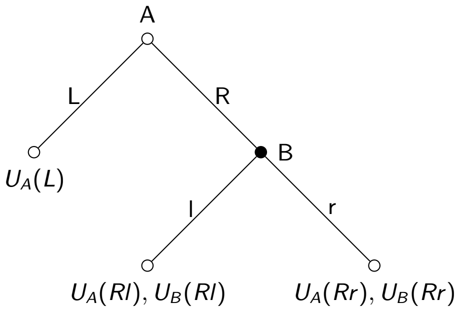
A Simple Game

Figure: Decision Tree



A Simple Game

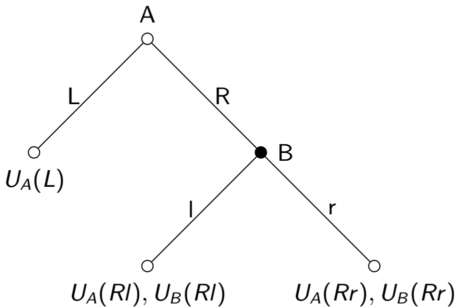
Figure: Decision Tree



- Player A chooses R or L.

A Simple Game

Figure: Decision Tree



- Player A chooses R or L.
- If Player A chooses R, Player B chooses r or l.

Solving the Game

- Assume the following:
  - Player A has preferences $Rr > L > Rl$.
  - Player B has preferences $Rr > Rl$.
- Use the technique

Solving the Game

- Assume the following:
  - Player A has preferences $Rr > L > Rl$.
  - Player B has preferences $Rr > Rl$.
- Use the technique of backward induction:
  - Player B, if given the opportunity to choose will choose

Solving the Game

- Assume the following:
  - Player A has preferences $Rr > L > Rl$.
  - Player B has preferences $Rr > Rl$.
- Use the technique of backward induction:
  - Player B, if given the opportunity to choose will choose r resulting in Rr.
  - Given the outcome Rr, player A prefers

Solving the Game

- Assume the following:
  - Player A has preferences $Rr > L > Rl$.
  - Player B has preferences $Rr > Rl$.
- Use the technique of backward induction:
  - Player B, if given the opportunity to choose will choose r resulting in Rr.
  - Given the outcome Rr, player A prefers Rr to L, so the subgame perfect equilibrium is

Solving the Game

- Assume the following:
  - Player A has preferences $Rr > L > Rl$.
  - Player B has preferences $Rr > Rl$.
- Use the technique of backward induction:
  - Player B, if given the opportunity to choose will choose r resulting in Rr.
  - Given the outcome Rr, player A prefers Rr to L, so the subgame perfect equilibrium is Rr.

Adding Uncertainty

- What if we are uncertain about each player's utility function? Can we create a probability of each player choosing a given option?

Adding Uncertainty

- What if we are uncertain about each player's utility function? Can we create a probability of each player choosing a given option?
- We can define U^* as the "true" utility and U as the observed utility. α is a random component of the utility that is unobserved, representing our uncertainty about the true utility function.

Adding Uncertainty

- What if we are uncertain about each player's utility function? Can we create a probability of each player choosing a given option?
- We can define U^* as the "true" utility and U as the observed utility. α is a random component of the utility that is unobserved, representing our uncertainty about the true utility function.
  - $U^*_B(I) = U_B(I) + \alpha_I = U_B(RI) + \alpha_I$

Adding Uncertainty

- What if we are uncertain about each player's utility function? Can we create a probability of each player choosing a given option?
- We can define U^* as the "true" utility and U as the observed utility. α is a random component of the utility that is unobserved, representing our uncertainty about the true utility function.
  - $U_B^*(l) = U_B(l) + \alpha_l = U_B(Rl) + \alpha_l$
  - $U_B^*(r) = U_B(r) + \alpha_r = U_B(Rr) + \alpha_r$

Adding Uncertainty

- Assume that α has errors distributed Type I extreme-value (Gumbel). Why?

Adding Uncertainty

- Assume that α has errors distributed Type I extreme-value (Gumbel). Why?
- Why not? We have no more information about the errors so we can assume anything about their structure. If we assume normality, we can use a probit model.

Adding Uncertainty

- Assume that α has errors distributed Type I extreme-value (Gumbel). Why?
- Why not? We have no more information about the errors so we can assume anything about their structure. If we assume normality, we can use a probit model.
- If the errors are distributed via the Gumbel it has been proven that the probability is distributed via the multinomial logit.
    - The standard form of multinomial logit probabilities is $p_{nj} = \frac{e^{\lambda U_i(a_{nj})}}{\sum_{k \in A_n} e^{\lambda U_i(a_{nk})}}$.

Adding Uncertainty

- Assume that α has errors distributed Type I extreme-value (Gumbel). Why?
- Why not? We have no more information about the errors so we can assume anything about their structure. If we assume normality, we can use a probit model.
- If the errors are distributed via the Gumbel it has been proven that the probability is distributed via the multinomial logit.
    - The standard form of multinomial logit probabilities is $p_{nj} = \frac{e^{\lambda U_i(a_{nj})}}{\sum\limits_{k \in A_n} e^{\lambda U_i(a_{nk})}}$.
    - Thus, we write $p_r = \frac{e^{U_B(Rr)}}{e^{U_B(Rl)} + e^{U_B(Rr)}}$ (assuming $\lambda = 1$)
    - $p_l = \frac{e^{U_B(Rl)}}{e^{U_B(Rl)} + e^{U_B(Rr)}}$

Adding Uncertainty

- Repeat for Player A, where $U_A^*(R)$ is defined as the expected utility from Player B's choice.

Adding Uncertainty

- Repeat for Player A, where $U_A^*(R)$ is defined as the expected utility from Player B's choice.
  - $U_A^*(L) = U_A(L) + \alpha_L$.

Adding Uncertainty

- Repeat for Player A, where $U_A^*(R)$ is defined as the expected utility from Player B's choice.
  - $U_A^*(L) = U_A(L) + \alpha_L$.
  - $U_A^*(R) = E[U_A(R)] + \alpha_R = p_l U_A(Rl) + p_r U_A(Rr) + \alpha_R$.

Adding Uncertainty

- Repeat for Player A, where $U_A^*(R)$ is defined as the expected utility from Player B's choice.
  - $U_A^*(L) = U_A(L) + \alpha_L$.
  - $U_A^*(R) = E[U_A(R)] + \alpha_R = p_l U_A(Rl) + p_r U_A(Rr) + \alpha_R$.
- We can now derive the probability of R and L for Player A substituting in the previously derived probabilities for r and l.

Adding Uncertainty

- Repeat for Player A, where $U_A^*(R)$ is defined as the expected utility from Player B's choice.
  - $U_A^*(L) = U_A(L) + \alpha_L$.
  - $U_A^*(R) = E[U_A(R)] + \alpha_R = p_l U_A(Rl) + p_r U_A(Rr) + \alpha_R$.
- We can now derive the probability of R and L for Player A substituting in the previously derived probabilities for r and l.
  - $p_L = \frac{e^{U_A(L)}}{e^{U_A(L)} + e^{E[U_A(R)]}} = \frac{e^{U_A(L)}}{e^{U_A(L)} + e^{p_l U_A(Rl) + p_r U_A(Rr)}}$
  - $p_R = \frac{e^{E[U_A(R)]}}{e^{U_A(L)} + e^{E[U_A(R)]}} = \frac{e^{p_l U_A(Rl) + p_r U_A(Rr)}}{e^{U_A(L)} + e^{p_l U_A(Rl) + p_r U_A(Rr)}}$
- These probabilities are the link between player B and player A.

Link to Real Data

- Now, we want to assign regressors to the utilities and estimate the parameters accompanying the regressors.

Link to Real Data

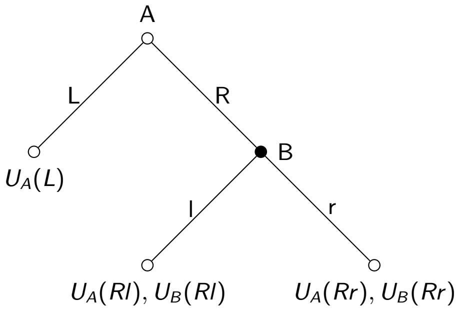
- Now, we want to assign regressors to the utilities and estimate the parameters accompanying the regressors.
- We can do this using a series of (relatively) simple logit regressions.
- This is *not* just a nested multinomial logit. Why?

Link to Real Data

- Now, we want to assign regressors to the utilities and estimate the parameters accompanying the regressors.
- We can do this using a series of (relatively) simple logit regressions.
- This is *not* just a nested multinomial logit. Why? Because different players are involved.

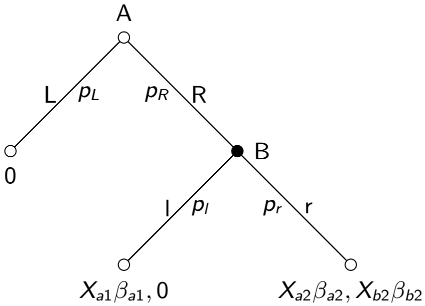
Remember This?

Figure: Decision Tree



Simplifying Utilities

Figure: Decision Tree



Recursive System of Equations

- The logit model has a binary outcome. Call that outcome y_A for Player A's choice and y_B for Player B's choice. How do we move from continuous utility to binary choices?

Recursive System of Equations

- The logit model has a binary outcome. Call that outcome y_A for Player A's choice and y_B for Player B's choice. How do we move from continuous utility to binary choices?
- Since players want to maximize utility, they compare U^*

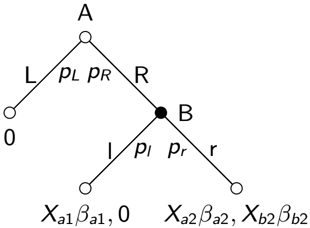
Recursive System of Equations

- The logit model has a binary outcome. Call that outcome y_A for Player A's choice and y_B for Player B's choice. How do we move from continuous utility to binary choices?
- Since players want to maximize utility, they compare U^*
  - $$y_A = \begin{cases} 1, & \text{if } U_A^*(R) \geq U_A^*(L) \\ 0, & \text{otherwise} \end{cases}$$

Recursive System of Equations

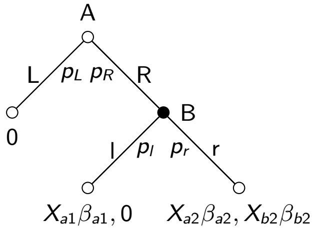
- The logit model has a binary outcome. Call that outcome y_A for Player A's choice and y_B for Player B's choice. How do we move from continuous utility to binary choices?
- Since players want to maximize utility, they compare U^*
  - $$y_A = \begin{cases} 1, & \text{if } U_A^*(R) \geq U_A^*(L) \\ 0, & \text{otherwise} \end{cases}$$
  - $$y_B = \begin{cases} 1, & \text{if } U_B^*(r) \geq U_B^*(l) \\ 0, & \text{otherwise} \end{cases}$$

Recursive System of Equations



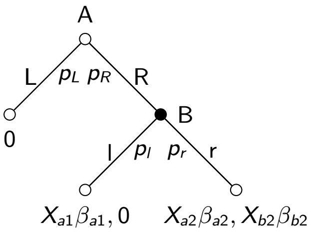
- We see that $U_A(y_A = 0) = 0$ and $U_B(y_A = 1, y_B = 0) = 0$.

Recursive System of Equations



- We see that $U_A(y_A = 0) = 0$ and $U_B(y_A = 1, y_B = 0) = 0$.
- We can estimate the additional utilities:
  - $U_A(y_A = 1, y_B = 0) = X_{a1}\beta_{a1}$
  - $U_A(y_A = 1, y_B = 1) = X_{a2}\beta_{a2}$
  - $U_B(y_A = 1, y_B = 1) = X_{b2}\beta_{b2}$

Recursive System of Equations



- We see that $U_A(y_A = 0) = 0$ and $U_B(y_A = 1, y_B = 0) = 0$.
- We can estimate the additional utilities:
  - $U_A(y_A = 1, y_B = 0) = X_{a1}\beta_{a1}$
  - $U_A(y_A = 1, y_B = 1) = X_{a2}\beta_{a2}$
  - $U_B(y_A = 1, y_B = 1) = X_{b2}\beta_{b2}$
- Substitute in data to our utility functions:
  - $y_A^* = p_l X_{a1}\beta_{a1} + p_r X_{a2}\beta_{a2} + \epsilon_A$, the utility for choosing R.
  - $y_B^* = X_{b2}\beta_{b2} + \epsilon_B$, the utility for choosing r.
  - where the error terms are distributed logistically by assumption.

Procedure

- First, we obtain variables that we believe influence the utility calculations of both players. We standardize so one strategy for each player is a constant.
- Then, we estimate y_B^*.

Procedure

- First, we obtain variables that we believe influence the utility calculations of both players. We standardize so one strategy for each player is a constant.
- Then, we estimate y_B^*.
- Then, we use the formula for p_r and $p_l = 1 - p_r$ to calculate $\hat{p}_r$ and $\hat{p}_l$ based on the data.

Procedure

- First, we obtain variables that we believe influence the utility calculations of both players. We standardize so one strategy for each player is a constant.
- Then, we estimate y_B^*.
- Then, we use the formula for p_r and $p_l = 1 - p_r$ to calculate $\hat{p}_r$ and $\hat{p}_l$ based on the data.
- Next, we substitute $\hat{p}_r$ and $\hat{p}_l$ into the equation for y_A^*.

Procedure

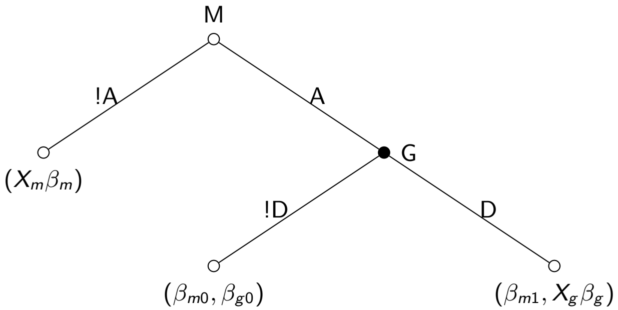
- First, we obtain variables that we believe influence the utility calculations of both players. We standardize so one strategy for each player is a constant.
- Then, we estimate y_B^*.
- Then, we use the formula for p_r and $p_l = 1 - p_r$ to calculate $\hat{p}_r$ and $\hat{p}_l$ based on the data.
- Next, we substitute $\hat{p}_r$ and $\hat{p}_l$ into the equation for y_A^*.
- Finally, we estimate y_A^*.

Procedure

- First, we obtain variables that we believe influence the utility calculations of both players. We standardize so one strategy for each player is a constant.
- Then, we estimate y_B^*.
- Then, we use the formula for p_r and $p_l = 1 - p_r$ to calculate $\hat{p}_r$ and $\hat{p}_l$ based on the data.
- Next, we substitute $\hat{p}_r$ and $\hat{p}_l$ into the equation for y_A^*.
- Finally, we estimate y_A^*.
- We can bootstrap the standard errors along the way.

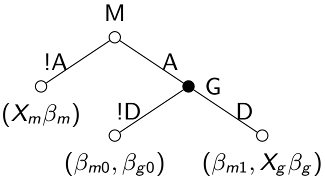
An Example

Figure: Decision Tree



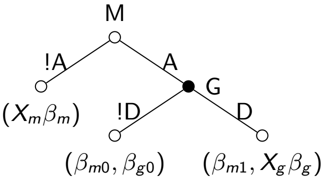
- Currency manipulation: M is market (can attack), G is government (can defend if attacked).
- If government defends, utility has two covariates.
- DV: whether an attack occurred and whether government defended.

Solving the Game



	M	G
Constants		
β_{g0}		0.20 (0.78)
β_{m0}	-4.05* (0.47)	
β_{m1}	-3.41* (0.50)	
Variables	β_m	β_g
β_{g1}		-0.07 (0.45)
β_{gm1}	-0.46* (0.18)	
β_{gm2}	0.29* (0.07)	0.59* (0.21)

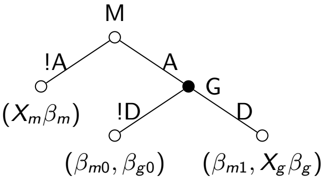
Solving the Game



	M	G
Constants		
β_{g0}		0.20 (0.78)
β_{m0}	-4.05* (0.47)	
β_{m1}	-3.41* (0.50)	
Variables	β_m	β_g
β_{g1}		-0.07 (0.45)
β_{gm1}	-0.46* (0.18)	
β_{gm2}	0.29* (0.07)	0.59* (0.21)

- Now, we use backward induction to solve the game and estimate the utilities for each player. *G* plays first and compares $\beta_{g0} = 0.20$ to $X_g \beta_g = \beta_{g1} + \beta_{gm2} = -0.07 + 0.59 = 0.52$.

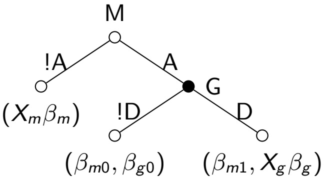
Solving the Game



	M	G
Constants		
β_{g0}		0.20 (0.78)
β_{m0}	-4.05* (0.47)	
β_{m1}	-3.41* (0.50)	
Variables	β_m	β_g
β_{g1}		-0.07 (0.45)
β_{gm1}	-0.46* (0.18)	
β_{gm2}	0.29* (0.07)	0.59* (0.21)

- Now, we use backward induction to solve the game and estimate the utilities for each player. G plays first and compares $\beta_{g0} = 0.20$ to $X_g\beta_g = \beta_{g1} + \beta_{gm2} = -0.07 + 0.59 = 0.52$.
- Calculate $p_{!D} = \frac{e^{0.20}}{e^{0.20}+e^{0.52}} = 0.57$ and $p_D = 0.43$; substitute to y_A^*.

Solving the Game



	M	G
Constants		
β_{g0}		0.20 (0.78)
β_{m0}	-4.05* (0.47)	
β_{m1}	-3.41* (0.50)	
Variables	β_m	β_g
β_{g1}		-0.07 (0.45)
β_{gm1}	-0.46* (0.18)	
β_{gm2}	0.29* (0.07)	0.59* (0.21)

- Now, we use backward induction to solve the game and estimate the utilities for each player. G plays first and compares $\beta_{g0} = 0.20$ to $X_g\beta_g = \beta_{g1} + \beta_{gm2} = -0.07 + 0.59 = 0.52$.
- Calculate $p_{!D} = \frac{e^{0.20}}{e^{0.20}+e^{0.52}} = 0.57$ and $p_D = 0.43$; substitute to y_A^*.
- Knowing this, M chooses between a payoff of $-3.41(0.43) - 4.05(0.57) = -3.78$ or $X_m\beta_m = -0.46 + 0.29 = -0.17$. Thus, the equilibrium is $!A$ with payoffs $(-0.17, 0)$.

Code?

- You can estimate the two logit models in any statistical package and also calculate bootstrapped standard errors.

Code?

- You can estimate the two logit models in any statistical package and also calculate bootstrapped standard errors.
- The sequential method described here generally means that the user has to calculate the probabilities by hand.

Code?

- You can estimate the two logit models in any statistical package and also calculate bootstrapped standard errors.
- The sequential method described here generally means that the user has to calculate the probabilities by hand.
- There are ways to estimate all this at once, but this occurs less often since a separate piece of software (STRAT) is needed.

Summary and Extensions

- Way to estimate utilities in a simple game framework.

Summary and Extensions

- Way to estimate utilities in a simple game framework.
- Can extend to more complex game trees and models with more possible variables contributing to the utility. Gets complicated quickly.

Summary and Extensions

- Way to estimate utilities in a simple game framework.
- Can extend to more complex game trees and models with more possible variables contributing to the utility. Gets complicated quickly.
- Only works if you are confident that your covariates really measure players utility.

Summary and Extensions

- Way to estimate utilities in a simple game framework.
- Can extend to more complex game trees and models with more possible variables contributing to the utility. Gets complicated quickly.
- Only works if you are confident that your covariates really measure players utility.
- Has frequently been used in IR, especially by David Carter (if you're looking for research ideas).