2010

LEARNING-BY-DOING AND PROJECT CHOICE: A DYNAMIC STRUCTURAL MODEL OF CROWDSOURCING

Nikolay Archak
New York University, narchak@stern.nyu.edu

Anindya Ghose
New York University, aghose@stern.nyu.edu

LEARNING-BY-DOING AND PROJECT CHOICE: A DYNAMIC STRUCTURAL MODEL OF CROWDSOURCING

Research-in-Progress

Nikolay Archak
Information, Operations and Management Sciences Department,
New York University,
Leonard N. Stern School of Business
narchak@stern.nyu.edu

Anindya Ghose
Information, Operations and Management Sciences Department,
New York University,
Leonard N. Stern School of Business
aghose@stern.nyu.edu

Abstract

This paper studies determinants of project choice in online crowdsourcing contests using a unique dataset from the world's largest competitive software development portal. Particular attention is given to the strategic roles of learning and forward-looking behavior in influencing contestants' decisions. We use a structural dynamic discrete programming (DDP) model to conduct our analysis and adopt a Bayesian approach to estimation. Our preliminary results provide evidence of learning-by-doing influencing propensities of users to choose projects of different types. The value of the parameter of intertemporal substitution that we identify suggests that while users are forward-looking, the aggregate behavior is far from fully rational. We attribute that result to mix of forward-looking and myopic users in the population.

Keywords: contest, crowdsourcing, discrete choice, learning, Bayesian estimation

Introduction

The term crowdsourcing describes a new Web-based business model that harnesses creativity of a distributed network of individuals through what amounts to an open call for proposals (Howe, 2006). In other words, an individual or a firm posts a problem online, a vast number of individuals offer solutions to the problem, the winning ideas are awarded some form of a bounty, and the company mass produces the idea for its own gain (Brabham, 2008). A specific form of crowdsourcing occurs in the form of contest for software innovation and development. In fact, online contests for open innovation – seekers posting innovation projects to which solvers submit solutions – have been developed into a new online commerce model. By taking advantage of the Internet, open innovation seekers can reach large pool of potential solvers with low cost and possibly better solutions. The potential seeker could be an individual, or a firm.

The mass media has had extensive coverage of this phenomenon in an enthusiastic manner. Empirical studies of crowdsourcing sites are however lacking and hence, little is known about the success of real-world markets for expertise. This paper presents one such study. It analyzes use of a crowdsourcing site, TopCoder.com, which is the world's largest competitive software development portal, wherein software buyers offer monetary payments to software sellers for solutions to problems. More specifically, we aim to examine if there is any evidence of learning dynamics exhibited by software programmers in such crowdsourcing contests. Towards this objective, we build on the emerging literature on learning and estimate a theory-based dynamic structural model.

Models of consumer learning about product quality are ubiquitous in the empirical marketing literature. It is now widely acknowledged that consumers can be strategic and forward-looking in their choices of experience and credence goods, thus choosing the product not only based on the current period utility level, but also taking into account the informational gain obtained from learning the new product qualities (Ching and Ishihara, 2007; Crawford and Shum, 2005, Erdem et al., 2005; 2008, Mehta et al., 2008). Relatively far less attention has been paid to the strategic implications of learning for occupational choice in general and project choice in particular. This topic seems to be of increasing significance nowadays, considering the trend of ubiquity in job specialization. For example, while at a high level software engineering can be thought of as a single occupation, at the micro level it can be decomposed into numerous specializations, ranging from the web development to the database management to the mobile phone application development and programming, and so on. Each requires its own specific knowledge and skills. Given this, it is plausible to assume that an unemployed individual choosing between potential employers or an employed individual choosing between new projects to participate in (up to the extent of such choice allowed by the employer) can base the decision not only on the best match with the current set of skills, but also on the potential usefulness of the additional skills that can be obtained in the process of employment. For example, software engineers with significant prior experience with Java programming language but little experience with C++ will have higher myopic incentives to perform Java projects (due to the better expected performance and decreased costs) but also a forward-looking incentive to engage in more C++ projects (to better learn C++). Understanding the extent of such strategic considerations can have important implications in many marketing areas. Examples include project management and salesforce management, given the recent evidence that learning-by-doing plays an important role in a salesperson's performance (Lu and Voola, 2009).

Research Setting: Simultaneous Crowdsourcing Contests on TopCoder.com

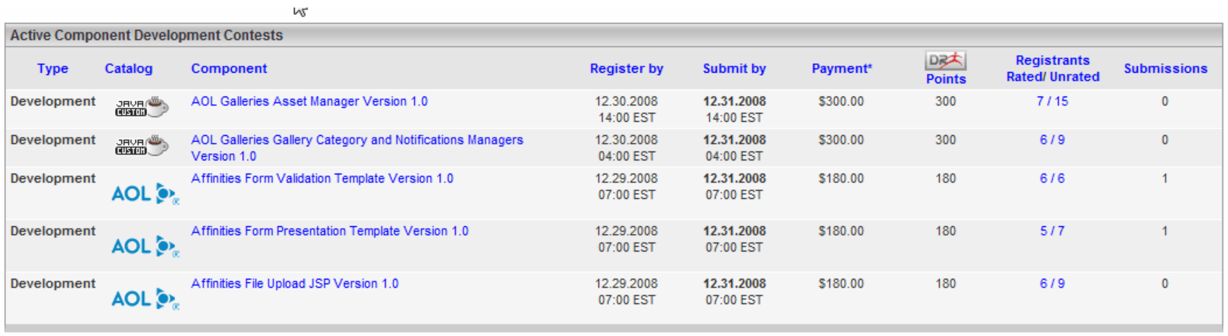
Empirical estimation of learning-by-doing models in a discrete choice setting is challenging, with the major complication being the unobservability of the full choice set of an individual. In this paper, we consider a unique setting of the online crowdsourcing contests in which this problem is largely resolved due to the constraints imposed by the environment. Our data comes from TopCoder, the world's largest competitive software development portal.

TopCoder.com is a website managed by the namesake company. The company hosts weekly online algorithm competitions as well as weekly competitions in software design and software development. The work in design and development produces useful software, which is licensed for profit by TopCoder. As of July 23, 2008 163,351 people have registered at the TopCoder website. 17.3% of those registered have participated in at least one

Algorithm competition, 0.3% in Design, 0.7% in Development[1]. We are particularly interested in Design and Development competitions as they have tangible payments to competitors.

The business model underlying software Design and Development competitions is briefly summarized below. TopCoder produces software applications for major clients. It interacts directly with the client company to establish application requirements, deadlines, budget etc. Once the application requirements are defined, the application goes to the Architecture phase, where it is split into a set of components. Each component is supposed to have a relatively small scope and precise set of technical requirements defining the expected component behavior and interface for interacting with other components. For instance, an "Address Book" component can be required to implement certain address management functionality, moreover, it should be written in Java and provide a Web service interface. The set of requirements to each component is summarized in a single document (Requirements Specification) and posted on the website as a single Design competition. Any registered member of the website satisfying minimum legal requirements can submit a UML design to any posted design competition. Winning design submission goes as input into the Development competition, which has similar structure, only the competitors are required to submit actual code implementing the provided UML design. Output from Development competitions is assembled together into a single application, which is later delivered to the customer.

Design and Development competitions are posted on TopCoder website on a weekly basis, Illustation 1 shows a sample list of weekly Development competitions. Every competition has a deadline by which all solutions must be submitted, it is usually within five to seven day interval after the competition posting date. It also has a monetary prize that is given to the competition winner and 50% of this amount is given to the first runner-up.



Active Component Development Contests

Type	Catalog	Component	Register by	Submit by	Payment*	DR Points	Registrants Rated/ Unrated	Submissions
Development	JAVA CUSTOM	AOL Galleries Asset Manager Version 1.0	12.30.2008 14:00 EST	**12.31.2008** 14:00 EST	$300.00	300	7 / 15	0
Development	JAVA CUSTOM	AOL Galleries Gallery Category and Notifications Managers Version 1.0	12.30.2008 04:00 EST	**12.31.2008** 04:00 EST	$300.00	300	6 / 9	0
Development	AOL	Affinities Form Validation Template Version 1.0	12.29.2008 07:00 EST	**12.31.2008** 07:00 EST	$180.00	180	6 / 6	1
Development	AOL	Affinities Form Presentation Template Version 1.0	12.29.2008 07:00 EST	**12.31.2008** 07:00 EST	$180.00	180	5 / 7	1
Development	AOL	Affinities File Upload JSP Version 1.0	12.29.2008 07:00 EST	**12.31.2008** 07:00 EST	$180.00	180	6 / 9	0

Illustration 1: Sample List of Weekly Contests

Important component of the Design and Development process is its scoring and review system. Once the submission deadline has passed, all submissions enter the review phase. Each submission is graded by three reviewers according to the specified scorecard on dimensions varying from technical submission correctness and clarity of documentation to flexibility and extendability of the solution.

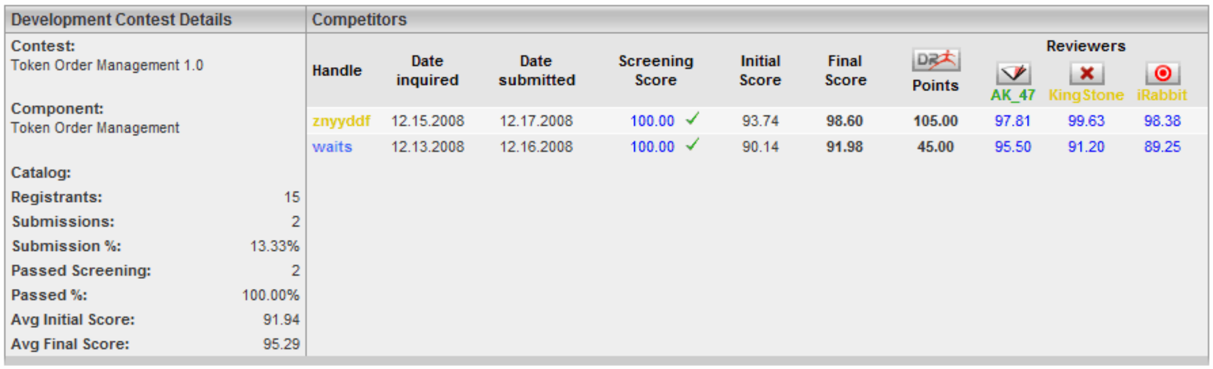
After the review process is complete, submissions enter the appeals phase where the competitors get a chance to appeal the decisions made by the reviewers. Once all appeals have been resolved, the placement is determined by the average score across all three reviewers. A sample of results is shown in Illustration 2.

TopCoder implements policy of maximum observability[2]. At first, competitors can always observe identities of their opponents, i.e., other members registered for the same contest. Moreover, for every member TopCoder tracks all prior competition history and summarizes it in a single rating number[3]. The rating is provided for members who have submitted at least one solution in some contest. It is calculated via a relatively complex formula taking into

[1]http://en.wikipedia.org/wiki/TopCoder

[2]One important exception from this rule is that contestants cannot see scores given by the reviewers to their opponents until after the appeals phase is over.

[3]Ratings are different for every competition track. Thus, an individual participating in both Design and Development competitions will have two different ratings - one for Design and one for Development.



Development Contest Details	
Contest: Token Order Management 1.0	
Component: Token Order Management	
Catalog:	
Registrants:	15
Submissions:	2
Submission %:	13.33%
Passed Screening:	2
Passed %:	100.00%
Avg Initial Score:	91.94
Avg Final Score:	95.29

Competitors

Handle	Date inquired	Date submitted	Screening Score	Initial Score	Final Score	DR Points	Reviewers AK_47	Reviewers KingStone	Reviewers iRabbit
znyyddf	12.15.2008	12.17.2008	100.00 ✓	93.74	98.60	105.00	97.81	99.63	98.38
waits	12.13.2008	12.16.2008	100.00 ✓	90.14	91.98	45.00	95.50	91.20	89.25

Illustration 2: Sample Details for a Development Contest

account all prior submission history of the contestant and relative performance compared to other contestants[4]. Fortunately, the exact formula for calculating the rating value is not important, as, in fact, even more information is available for each rated competitor, including all prior competition history. This information can be revealed by clicking on the member's handle. Thus, we will simply think of ratings as proxies for the coder's performance so far: the better the coder performed in the past, the higher the rating will be, and more recent performance will have higher effect on the current rating.

Our goal is to capture how the following three factors influence project choice for software programmers in the online crowdsourcing contests: (i) monetary payment from each project, (ii) reputation rating from prior projects, and (iii) learning a particular set of skills from different projects. Two parameters are directly observable: (i) monetary payment (since we see the project payment for every project), and (ii) reputation rating. However, skills are not directly observable. Instead we assume a common multivariate normal prior on the initial skills of every coder.

Anecdotal Evidence of Forward-Looking Behavior and Learning Dynamics on TopCoder.com

Structural models of user behavior are always as good as the assumptions they rest upon. Before we proceed with describing our own structural model of project choice, we should at least demonstrate face validity of the basic assumptions that we make. In particular, we assume that (i) participation in software projects related to a particular technology, improves future users productivity in similar types of projects (learning-by-doing) and (ii) contestants are forward-looking and take into attention potential impact of new skills they learn on their future performance. The first assumption is widely adopted in the engineering community to the extent that it shifted the teaching paradigm for many engineering classes towards teaching software engineering by doing or “immersion” (Carlson and Sullivan, 1999; Riboud and Saliou, 2003). Next, we provide anecdotal evidence in favor of the second assumption. The evidence is taken from the Software Competition Discussion Roundtables (forums) run by TopCoder. These forums provide opportunity for the designers and developers to interact with each other and TopCoder representatives and discuss generic topics related to the evolution of the contest platform. Manual inspection of the forum content revealed numerous topics exposing forward-looking concerns of the contestants.

Below we give two examples that we consider illuminating. In the first example, a user interested in a particular technology (Flex), asks whether more projects of that type will soon be available.

<user>: I don't see any Flex component in upcoming project page. Is Calendar application is only Flex project? Or some applications are going to be in near future?

<admin>: There will be other Flex components coming, but I do not know of any more for at least the next 2-3 weeks.

[4]Detailed description of the rating system can be found at http://www.topcoder.com/wiki/display/tc/Component+Development+Ratings

In the second example, a newbie user asks what technologies would be useful to know in order to compete successfully in TopCoder

<user> Hi, I want to participate in TC development competitions. But it looks like one has to learn some technologies like JPA, EJB 3, Hibernate. So I have some questions: 1) If somebody can tell that which technologies like these are common and occur frequently as requirement for TC development then it would be great…

In fact, TopCoder recognized strong forward-looking aspects of user behavior in software design and development contests and have introduced the upcoming contests page listing the basic information (such as language platform and dependencies) for the contests that are likely to be posted in the nearest future. The upcoming contest page was introduced only recently and so it is not covered by the time range of our dataset and therefore it is not represented in our structural model.

Dataset

We obtained historical contest data from the TopCoder website. The dataset included 968 Java an 542 C# contests and covered a period of 226 weeks. The total number of different TopCoder members, who participated in at least one of the contests in our dataset, was 1,660. Among these members, 301 individuals participated in Design competitions only, 1,106 individuals participated in Development competitions only, and 253 participated in at least one Design and at least one Development competition. Due to limited size of this paper, we concentrate on results Software Design contests only. Descriptive statistics of projects are given in Table 1.

Table 1. Descriptive Statistics (Project Data)

	Java				C#			
	Mean	**St. Dev.**	**Min**	**Max**	**Mean**	**St. Dev.**	**Min**	**Max**
Projects Per Week	4.28	3.49	0	16	2.39	2.68	0	16
Payment (USD)	691.06	334.77	100	2900	773.25	286.95	100	3000
Number of Requirements	9.67	14.35	1	195	11.13	10.97	1	60
Specification Length (in pages)	4.79	3.22	2	43	4.55	2.48	2	21
Number of Contestants	2.79	2.16	1	26	2.48	1.54	1	11

Note that the participation rate for every particular contest is very low: the mean number of submissions in a contest is less than three and the median number of submissions is two. Nonetheless, TopCoder represents a classic example of the crowdsourcing platform as the submissions come from a diverse pool of potential contestants and the identities of individuals who will submit solutions are not fixed or known in advance.

Descriptive statistics of submissions are given in Table 2 (all variables represent coder characteristics right before the coder submits a solution for the contest). The distribution of the number of contests per individual is heavily skewed as evidenced by the relationship between its mean and standard deviation; while most contestants in our sample have participated only in a couple of contests, there is also a large core of around 50 users that consistently participate in design competitions. Our results are mostly pertinent to the behavior of individuals in the core; removing the rest of the samples as a robustness check does not significantly affect the estimation results.

Table 2. Descriptive Statistics (Submission Data)

	Mean	**St. Dev.**	**Min**	**Max**

Experience in Java (number of contests)	23.61	37.5	0	211
Experience in C# (number of contests)	15.08	26.58	0	123
Rating	1255.98	623	0	2794

Model

Every week there is a new set of contests posted on the website. Users can browse through requirements documents for the posted contests and decide on the projects to choose. We will use subscript t to index weeks, subscript i to index individuals and subscript j to index projects. In a majority (more than 90%) of all individual-week pairs in our dataset, at most one project per week has been chosen. To simplify the modeling task significantly, we exclude the rest of the observations, thus reducing the model to the traditional discrete choice setting.

We also emphasize that we do not model user choice between different contest types (Algorithm, Design, Development) but only the choice of projects within a particular contest type (Design and Development). The reason we can do that is that the contests of different types on TopCoder are neither complements nor substitutes. Algorithm competitions tend to be of very different nature than design and development contests: they have a much shorter time scale of less than two hours and do not generate monetary compensation for participants. Substitution effects between Design and Development contests, although may exist, are believed to be weak due to significant difference in the style of both contests. In our sample of 1,660 contestants, only 253 ever participated in contests of both types. We performed additional robustness check by estimating the model with this set of individuals omitted; the results were not significantly affected.

Finally, we model learning as happening when the user *submits* the project. Winning is obviously not a prerequisite for learning to happen: users who deliver a solution and lose the contest are likely to learn as much as the user who won it, assuming a similar amount of effort was put in the solution. While some users may complete the component but not submit it for review, we believe that to be an extremely rare phenomenon, especially for solutions in which the user has invested significant amount of time to learn something new. Finally, submitting the solution allows user to get a useful feedback from the review board, providing detailed information on strengths and weaknesses of the submission. Such feedback should greatly contribute to the overall learning experience.

Myopic Contestants

First, consider a setting with myopic users. We model user i utility from choosing alternative j in week t as

$$u_{ijt} = \alpha x_{jt} + \beta y_{ijt} + u_{i0}\delta_{j0} + u_{i1}\delta_{j1} + \varepsilon_{ijt},$$

where x_{jt} represents project specific covariates (payment, number of requirements and specification length) for project j posted in week t, y_{ijt} represents contestant specific covariates (experience with projects of the same type, experience with projects of other type and reputation) for contestant i in week t, u_{i0} is a proxy for the individual's Java skills (or preference for Java), u_{i1} is a proxy for the individual's C# skills (or preference for C#), δ_{j0} is one if the project is in Java and zero otherwise, δ_{j1} is one if the project is in C# and zero otherwise and ε_{ijt} is idiosyncratic “love-for-variety” error term which is assumed to be i.i.d. extreme-value type I distributed. We follow a standard convention of having an outside option (not doing any project in a particular week) with the expected utility normalized to zero. Under these assumptions, the project choice probability is given by a standard multinomial logistic expression:

$$p_{ijt} = \exp(\alpha x_{jt} + \beta y_{it} + u_{i0}\delta_{j0} + u_{i1}\delta_{j1})/(1 + \Sigma_k \exp(\alpha x_{kt} + \beta y_{ikt} + u_{i0}\delta_{k0} + u_{i1}\delta_{k1})).$$

Furthermore, we assume that individual specific random effects u_{i0} and u_{i1} have a joint bivariate normal distribution with means μ_{i0} and μ_{i1} and covariance matrix Σ.

Forward-Looking Contestants

A more advanced model of user behavior should take into account potential effects of contestant actions on contestant specific covariates such as experience with projects of a particular type (Java/C#) and contestant's reputation (rating). As such covariates directly enter contestant's utility function in every period, forward-looking contestants should choose projects taking into account not only direct utility gain in the current period but also change in the future expected utility due to the associated change in experience and reputation:

$$Eu^{total}_{ijt}(s_{cur}) = \max_{action} Eu^{imm}_{ijt}(action; s_{cur}) + \beta_{discount}\Sigma_{new\ state}\ p(s_{cur}\ \text{to}\ s_{new} \mid action)Eu^{total}_{ijt}(s_{new}),$$

where E represents expectation across all relevant random variables, u^{imm} stands for the immediate period utility function (as in the section for myopic users), s_{cur} and s_{new} represent the current and the new user state and p represents the transition probability between states *conditional on the action chosen by the user.* Parameter $\beta_{discount}$ is of particular importance as it captures the substitution effect for intertemporal consumption. An empirical estimate of zero would indicate fully myopic users; it is generally accepted that a rational individual should have the value of $\beta_{discount}$ above 0.9 (the actual value, of course, depends on the time scale chosen, which in our case is a single week).

We employ a structural discrete choice dynamic programming (DDP) framework to conduct our analyses for forward-looking contestants and adopt a Bayesian approach to estimation. In particular, we use the Bayesian Dynamic Programming algorithm developed in Ching et al. (2009) which significantly speeds up the estimation of the structural model by combining simulation steps of the MCMC algorithm and the fixed-point iteration steps of the DP algorithm. Due to space limitations and technical complexity of the algorithm, we avoid presenting it in the paper. Interested readers can refer to an excellent tutorial by the inventors (Ching et al., 2009) The algorithm was implemented in C++ and its correctness was validated on a simulated dataset of two competing stores with coupon reward programs from the tutorial paper. The estimation code is available upon request.

Identification in our model comes from the manner in which the three factors (payment, reputation and learning opportunities) influence the coder's decisions and how the influence changes depending on the prior history of the

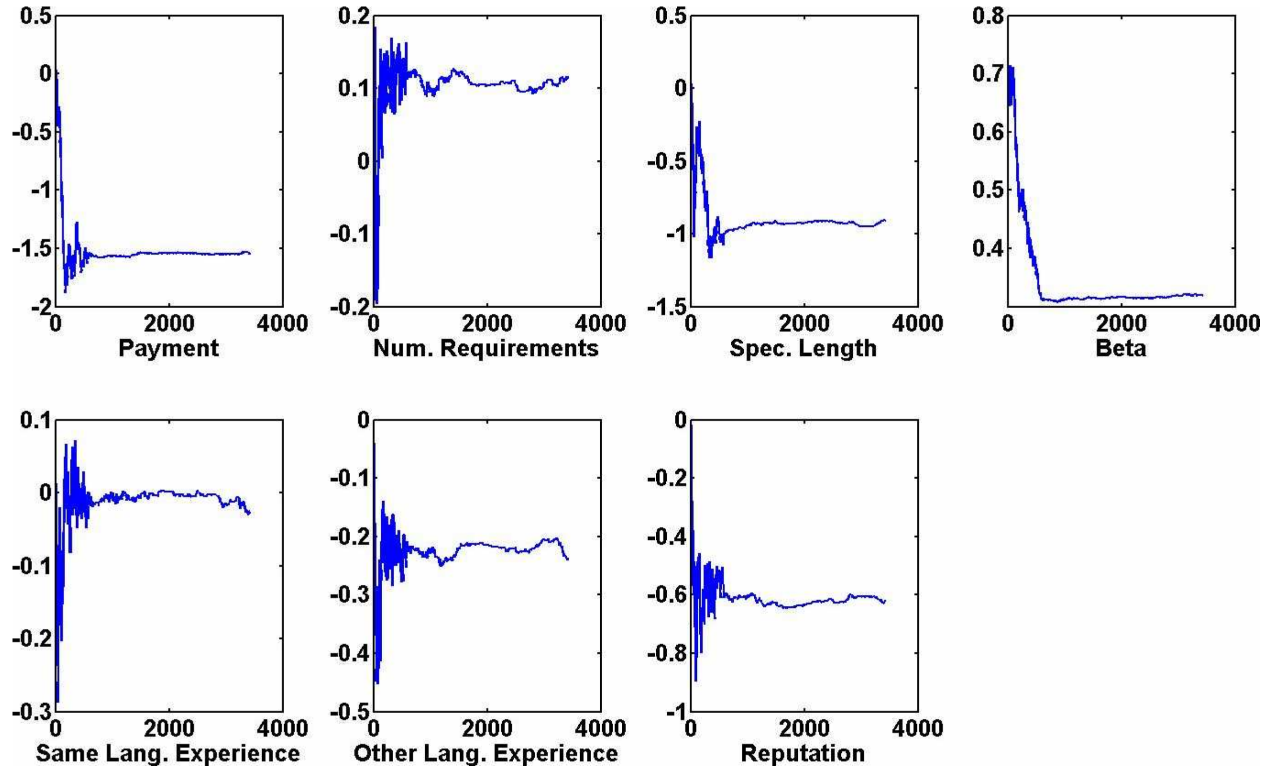


Illustration 3: MCMC Results for Major Variables

coder. While value of money should be constant over time, value of additional reputation and learning opportunities change depending on the accumulated experience and reputation. In particular, we would expect the value of learning to decrease with experience: the more we learn, the smaller is the marginal effect of the next learning experience. Our model assumes that coders are forward looking and solve a dynamic programming problem when deciding on the projects to participate in and the level of effort to choose. In the DP problem, we represent the coder's state as the combination of the current Java skills (unobservable directly but can be proxied by the number of performed Java projects and the initial coder skill which is a random effect) and the current C# skills (modeled similar to Java). Note that the reputation rating and its volatility are not currently included in the state variable due to significant increase in the size of state space associated with such inclusion. We are currently working on extension of the model and improvements to the estimation algorithm that will allow us to do that.

Preliminary Results and Discussion

We use MCMC (Markov Chain Monte Carlo) approach for estimation, in particular, we use uninformative priors and Metropolis-Hastings algorithm for sampling. The chain was run for 80,000 iterations and the last 40,000 were used as a burn-in. The sample paths for major covariates are shown in Illustration 3. As the number of iterations grows, each sample path converges to a single number representing our posterior belief about the value of the corresponding coefficient.[5]

Among project covariates, the project payment and the specification length are strong negative determinants of project choice, i.e., coders in crowdsourcing contests strongly prefer choosing simpler projects: those with shorter requirements specification documents but also with smaller payments. We are currently working on the extension of our model including unobservable project-specific random effects that will allow us to separate influence of the project payment on the project choice from the unobservable project complexity. Controlling for the requirements specification length, coders prefer contests with larger number of requirements; we suggest a plausible explanation that among two contests with similar length of the requirements document, one having more individual requirements is better documented and detailed and therefore is easier to deliver. Experience with the same language is not a significant determinant of the project choice, while experience with a different language has a strong statistically and economically significant negative effect: if a user has experience with Java, she is unlikely to pick a C# project *myopically* and vice versa. Together these two estimates provides an evidence of the potential learning-by-doing effect: as contestants accumulate experience with particular programming language, say Java, their preference for projects in that particular language only increases. Note that this effect is not due to inherent coder preference for a particular language (some coders may always prefer Java, others always prefer C#), as we control for individual coder traits using coder-specific effects. We also emphasize that these coefficients capture impact of skill on the *immediate period* user utility. Due to the way we constructed the structural model, they do not include the forward-looking component represented by β_{discount}.

Our estimate of β_{discount}, which captures user patience for intertemporal substitution, is in the neighborhood of 0.3 and statistically significant. On one hand, that is evidence in favor of the forward-looking behavior of the users: they, at least partially, take into account impact of skills they can learn now on their future performance. On the other hand, the coefficient is far from what a fully rational model of behavior would suggest. Essentially, it translates \$500 a week later to just a \$150 today. One plausible explanation for the magnitude of the effect is the potential heterogeneity of our user base. It may as well be that we have about 30% of rational and 70% of myopic users in population, which theoretically can translate into a downward biased estimate for β_{discount} from the pooled model. We consider investigating this question as an interesting direction for further research.

Finally, reputation is a negative determinant of project choice: as coders accumulate more reputation, they tend to become less active. We are currently looking into potential explanations for this effect.

[5] Strictly speaking, Bayesian learning in general and MCMC in particular never give a single number as the estimate but rather the whole distribution of posterior beliefs about the parameter of interest. In our case, the dataset size is large enough so that the posterior distribution is heavily clustered around a single mode.

Conclusion

Our preliminary analyses provide evidence of learning through users switching propensities across different projects available to them. This helps us capture users' dynamic learning behavior in a project management setting. We are currently working on a number of policy simulations and counter-factual experiments based on our empirical estimates.

We would like to conclude with the observation that the strategic learning-by-doing considerations generalize far beyond the crowdsourcing platforms. In particular, advertisers in the sponsored search setting face similar exploration-exploitation trade-off: while gathering information about the click-through rates of particular slots, they may loose revenue, but the obtained information may be used to improve their future bidding strategies. Other instances of learning-by-doing occur in the management and IT consulting industry as well wherein consultants often make a choice between which projects to chose for their next assignment based on the potential learning that can be accrued from their participation in a given project. Our theoretical and empirical modeling approach will thus provide a robust and general framework that can be applied towards enhancing our understanding of learning dynamics of employees in other industries as well.

References

Carlson, Lawrence E., Jacquelyn F. Sullivan. "Hands-on Engineering: Learning by Doing in the Integrated Teaching and Learning Program," *International Journal of Engineering Education* (15:1), 1999, pp. 20-31.

Ching, Andrew, Susumu Imai, Masakazu Ishihara, Neelam Jain. "A Guide to Bayesian Estimation of Dynamic Discrete Choice Models with an Application to a Store-Level Reward Problem," *Working Paper*, 2009, Available at SSRN: http://ssrn.com/abstract=1398444.

Ching, Andrew, Masakazu Ishihara. "The effects of detailing on prescribing decisions under quality uncertainty," *Working Paper*, 2007, Available at SSRN: http://ssrn.com/abstract=1010179.

Crawford, Gregory S., Matthew Shum. "Uncertainty and Learning in Pharmaceutical Demand," *Econometrica* (73:4), 2005, pp. 1137-1173.

Erdem, Tulin, Michael Keane, T. Oncu, Judi Strebel. "Learning about computers: An analysis of information search and technology choice", *Quantitative Marketing and Economics* (3:3), 2005, pp. 207-247.

Erdem, Tulin, Michael Keane, Baohong Sun. "A Dynamic Model of Brand Choice When Price and Advertising Signal Product Quality", *Marketing Science* (27:6), 2008, pp. 1111-1125.

Howe, J. "The rise of crowdsourcing," *Wired Magazine*, 2006, Available at http://www.wired.com/wired/archive/14.06/crowds_pr.html

Huberman, B., D. Romero, F. Wu, "Crowdsourcing, attention and productivity," *Working Paper*, 2008, Available at SSRN: http://ssrn.com/abstract=1266996.

Ching, Andrew, Susumu Imai, Neelam Jain. "Bayesian Estimation of Dynamic Discrete Choice Models", *Econometrica* (77:6), 2009, pp. 1865-1899.

Lu, Q, R Voola. "Investigating salespeople's learning by doing in a Bayesian learning structural framework", *Marketing Dynamics Conference,* 2009.

Mehta, Nitin, Xinlei Chen, Om Narasimhan. "Informing, transforming and persuading: Disentangling the multiple effects of advertising on brand choice decisions", *Marketing Science* (27:3), 2008, pp. 334-355.

Ribaud, Vincent, Philippe Saliou. "Software engineering apprenticeship by immersion", *International Workshop on Patterns in Teaching Software Development,* 2003.