Demand-Aware Career Path Recommendations: A Reinforcement Learning Approach (Online Appendix)

Marios Kokkodis
Carroll School of Management, Boston College, Chestnut Hill, MA 02467,
kokkodis@bc.edu

Panagiotis G. Ipeirotis
Leonard N. Stern School of Business, New York University, New York, New York 10012,
panos@stern.nyu.edu

A skill's value depends on dynamic market conditions. To remain marketable, contractors need to keep reskilling themselves continuously. But choosing new skills to learn is an inherently hard task: Contractors have very little information about current and future market conditions, which often results in poor learning choices. Recommendation frameworks could reduce uncertainty in learning choices. However, conventional approaches would likely be inefficient; they would model previous (often poor) observed contractor learning behaviors to provide future career path recommendations while ignoring current market trends.

This work proposes a framework that combines reinforcement learning, Bayesian inference, and gradient boosting to provide recommendations on how contractors *should* behave when choosing new skills to learn. Compared with standard recommender systems, this framework does not learn from previous (often poor) behaviors to make future recommendations. Instead, it relies on a Markov Decision Process to operate on a graph of feasible actions and dynamically recommend profitable career paths. The framework uses market information to identify current trends and project future wages. Based on this information, it recommends feasible, relevant actions that a contractor can take to learn new, in-demand skills. Evaluation of the framework on 1.73 million job applications from an online labor market shows that its implementation could increase (1) the marketplace's revenue by up to 6%, (2) contractors' wages by 22%, and (3) the diversity of new skill acquisitions by 47%. A comparison with alternative recommender systems highlights the limitations of approaches that make recommendations based on previously observed learning behaviors.

Appendix A Mapping skillsets to numeric vectors

Multiple algorithms can decompose skills into numeric vectors. We examine three such approaches: Word embeddings (W2V, Mikolov et al. 2013), Document embeddings (D2V, Le and Mikolov 2014), and Singular Value Decomposition (SVD). The goal is to map any skillset to a numeric vector of finite dimensions.

The W2V and D2V approaches assume that any observed skillset is a document, and any skill is a term. W2V then decomposes each skill (i.e., term) into a numeric vector. Let us assume that the numeric vector has J dimensions. To get a skilset mapping $\mathbf{S} \rightarrow v(\mathbf{S})$, where $v(\mathbf{S}) = [v_1^{\mathbf{S}}, ..., v_j^{\mathbf{S}}, ..., v_J^{\mathbf{S}}]'$ we aggregate the skill mappings as follows:

$$v_j^{\mathbf{S}} = \frac{1}{|\mathbf{S}|} \sum_{k \in |\mathbf{S}|} v_j^k \,, \tag{13}$$

where v_j^k is the j^{th} dimension of the mapping of skill $k \in \mathbf{S}$.

The D2V approach does not need such an aggregation, as it directly decomposes skillsets to vectors of real numbers. Finally, SVD can also map skills into vectors of latent dimensions. The initial matrix in this scenario is a skill $\times$ user binary matrix. To get the skillset representation, we use Equation 13.

To compare the most appropriate method for estimating the learning affinity between different sets of skills, we use a comparative intrinsic evaluation (Bakarov 2018, Sayeed et al. 2016). Specifically, we manually generate two lists of skillsets: the first one (DS) consists of skillsets that are disjoint, and contextually far away from each other. The second one consists of skillsets that are contextually close to each other (JS). For each algorithm, we estimate the mean cosine similarity for both lists:

$$\text{MCS}_X = \text{AVG}\Big[cos\big(v(\mathbf{S}'), v(\mathbf{S})\big)\Big], \;\; \forall \mathbf{S}, \mathbf{S}' \in X, \;\; \mathbf{S} \neq \mathbf{S}', \;\; X \in \{DS, JS\} \,. \tag{14}$$

Table A1 shows the two lists of skills and the resulting MCS for each approach. W2V outperforms the other two approaches, in both the DS (lower similarity) and the JS (higher similarity) lists of skillsets. Hence, we choose W2V for this study.

Table A1 Comparison of different approaches for mapping skillsets

	List of disjoint skillsets (DS)	List of contextually similar skillsets (JS)
	{animation, editing, video}	{css, html, javascript}
	{css, html, javascript}	{css, html, jquery, mysql, php, wordpress}
	{arabic, english}	{css, wordpress}
	{asp, sql}	{html, php, wordpress}
	{backlinking, seo}	{joomla, wordpress}
	{video, youtube}	{ajax, css, html, javascript, jquery, mysql, php}
	{amazon, ebay}	{css, html, mysql, php}
	{logo, marketing}	{editing, wordpress}
	{proofreading, writing}	{magento, mysql, php}
Mean cosine similarity (MCS)		
W2V	-0.21 ± 0.07	0.73± 0.07
D2V	0.05 ± 0.05	0.61 ± 0.06
SVD	0.06 ± 0.03	0.46 ± 0.03

95% Confidence intervals for the MCS values. W2V yields better results in both the DS and JS lists.

Appendix B Alternative approaches for estimating learning ability

Section 4.2.2 presents a Bayesian approach for estimating the learning ability of a contractor. More sophisticated approaches can leverage a series of additional observed signals to estimate the latent learning ability of a contractor.

As we mentioned in Section 3.2, a contractor might learn new skills without ever using them in the marketplace. This can happen because (1) the newly learned skills have low demand or (2) there was not enough time to observe the contractor using them. At the same time, due to poor choices, a new skillset might not be as profitable, and as a result, the contractor might choose to avoid using it.

An alternative approach that addresses some of these shortcomings is to build probabilistic models that estimate the latent ability of a contractor through a series of observed signals. Specifically:

$$d = Pr(Y = 1|\mathbf{Z}) = \Lambda(\boldsymbol{\beta}\mathbf{Z}) \,, \tag{15}$$

where Λ is the logistic sigmoid, $\boldsymbol{Z}$ is a vector of covariates that includes the (1) time after skill acquisition, (2) demand of skill in the marketplace, (3) average skill wage. Similar to the Bayesian approach, the dependent variable Y is a binary variable that is "1" if the contractor has successfully used the new skill, or if the contractor has a listed certification (top 50^{th} percentile) on the new skill. The resulting logistic model yields a ten-fold cross-validated AUC score of 0.86, indicating a superior performance than the simple Bayesian approach, which yields an AUC score of 0.69.

Appendix C Transition probabilities

Section 4.2 presents a methodology for estimating the transition probabilities between two states $\mathbf{S}$ and $\mathbf{S}'$. To estimate the coefficients of Equation 2, we create a dataset of successful and unsuccessful transitions based on the changes on the contractor's profile (similar to Section 4.2.2): A successful transition from $\mathbf{S}$ to $\mathbf{S}'$ occurs when the focal contractor either successfully completes a task that requires the new skillset $\mathbf{N} = \mathbf{S}' - \mathbf{S}$, or lists relevant skill certificates (upper 50^{th} percentile). As in Section 4.2.2, we only consider profile changes for which we have observations for more than six months.

The features of Equation 2 include (1) the learning affinity, (2) the learning ability, (3) the portion of time invested in working, and (4) the country of the contractors. We estimate the learning affinity of the new skillset $\mathbf{S}'$ according to equation 3, and the individual learning ability through the Bayesian framework described in section 4.2.2. Finally, we get a rough estimate of the fraction of time invested in learning by sampling through a normalized (max-min) distribution of the hours billed between consecutive periods of the same contractor. The final dataset includes 32,264 transitions, 19% of which were successful.

Table C2 presents the descriptive statistics of this dataset. Table C3 shows the results of this model. All the coefficients are statistically significant ($p < 0.05$). The coefficients of learning affinity and individual ability are positive, while the coefficient of λ is negative, hence verifying intuition.

To study the predictive performance of the logistic regression model, we split the focal dataset into a training (first two years) and a test set (last two years). We then build our model on the training set and evaluate on the test set. The out of sample accuracy is 82%, with an AUC score of 77%. Finally, the model predicts reasonable probabilities, ranging from 0 to 1, with a mean value of 0.2, median 0.17, and a standard deviation of 0.15.

Table C2 Features for estimating the transition probabilities between states

	Min	Mean	Median	Max	StD
Successful transition from $\mathbf{S}$ to $\mathbf{S}'$	0	0.19	0	1	0.39
Learning affinity ($L(\mathbf{N}\|\mathbf{S})$)	-0.70	0.26	0.30	0.88	0.23
Individual learning ability (d)	0.01	0.24	0.21	0.94	0.17
Time invested in working (λ)	0	0.22	0.17	1	0.17

Each country enters the model as a dummy variable.

Table C3 Logistic regression coefficients

DV: Successful transition from $\mathbf{S}$ to $\mathbf{S}'$	Logit model
Learning affinity ($L(\mathbf{N}\|\mathbf{S})$)	0.04^{**}
	(0.01)
Individual learning ability (d)	0.58^{***}
	(0.01)
Time invested in working (λ)	-0.97^{***}
	(0.02)
Country dummies	✓
Observations	32,264

The constant term is estimated but omitted from the table. $^{***}p-$value < 0.001, $^{**}p-$value < 0.01.

Appendix D Demand, supply, and wage prediction

Time-series prediction algorithms can model future demand, supply, and wage (Equations 7-9).[1] We compare four different such approaches: ARIMAX (Nau 2018), Recurrent neural networks (LSTM, Brownlee 2016), SVM Regression (Shevade et al. 2000) and Gradient boosting regression (XGBoost Chen and Guestrin 2016).

To test the performance of the four models, we split our dataset into two years of training and two years of testing. To match the simulations' processes in Section 5, we train and test the four models by aggregating observations to two-month periods. (For predictions, we use two lagged periods (L=2) as alternative values of L did not improve the models' performances. We further use the ratio between demand and supply during each period as an additional predictor.)

We compare alternative models by estimating the mean average prediction error:

$$\text{Average prediction error } (\%) = \frac{1}{N \cdot |\mathbb{S}|} \sum_{n=1}^{N} \sum_{\mathbf{S} \in \mathbb{S}} \frac{\hat{X}_n(\mathbf{S}) - X'_n(\mathbf{S})}{X'_n(\mathbf{S})}, X \in \{\text{Dem, Sup, W}\}, \tag{16}$$

In the previous, X' represents actual observations and $\hat{X}$ predictions. (For $n > 2$, we use the predicted values of the previous periods to make a prediction.) Figure D1 shows that all models perform almost on par, making balanced prediction errors (both positive and negative) that result in insignificant average errors (SVM regression's wage prediction error is the only exception).

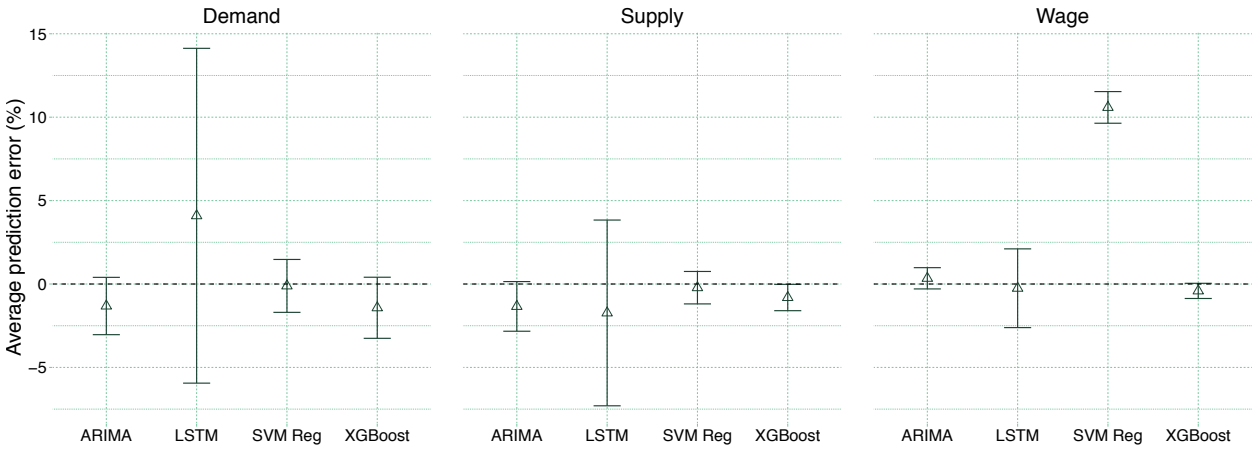
Furthermore, Figure D2 shows the size of these errors through the mean absolute prediction error:

$$\text{Average absolute prediction error } (\%) = \frac{1}{N \cdot |\mathbb{S}|} \sum_{n=1}^{N} \sum_{\mathbf{S} \in \mathbb{S}} \frac{|\hat{X}_n(\mathbf{S}) - X'_n(\mathbf{S})|}{X'_n(\mathbf{S})}, X \in \{\text{Dem, Sup, W}\}, \tag{17}$$

In terms of wage prediction, the XGBoost model seems to slightly outperform the other three models. Figure D2 further shows that, on average, the size of the wage prediction error is $\sim 3\%$. This error maps to ± 40 cents of a dollar on an average hourly wage of \$13.6 per hour. (Note that in different contexts, alternative models might perform better. For example, in longer time-series datasets LSTM will likely outperform the alternative approaches.)

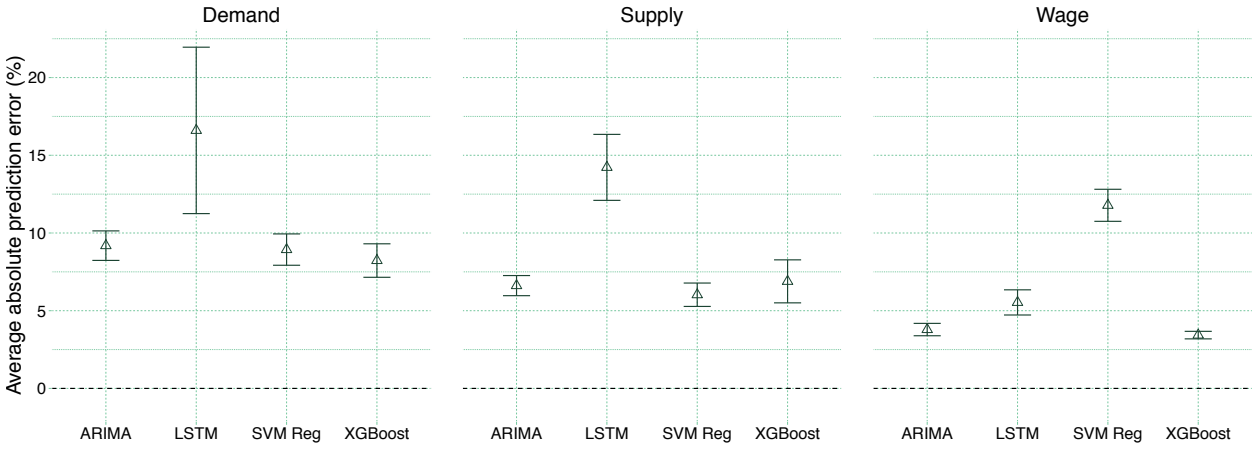
[1] Note that we capture the country component of Equation 6 by estimating the average wage per country across all observed skillsets.

Figure D1 Average prediction errors for forecasting future demand, supply, and wage



Average prediction errors are not statistically significantly different than zero. Error bars show 95% confidence intervals.

Figure D2 Average absolute prediction error for forecasting future demand, supply, and wage



XGBoost slightly outperforms alternative models. On average, the size of the wage prediction error is $\sim 3\%$. Error bars show 95% confidence intervals.

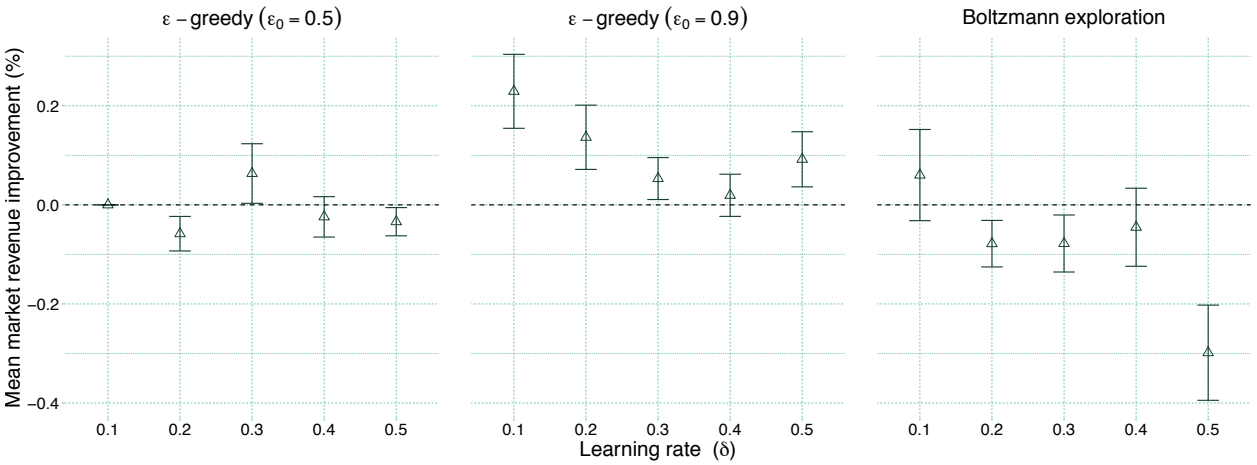
Appendix E Tuning of Algorithm 1

Algorithm 1 uses an $\epsilon-$greedy strategy and a learning rate parameter δ that require tuning. For adaptive $\epsilon-$greedy (Algorithm 1), we use two starting points: $\epsilon_0 = 0.9$ and $\epsilon_0 = 0.5$. Line 15 in Algorithm 1 adjusts the value of ϵ at the end of each step. Such an $\epsilon-$greedy exploration schedule diminishes to 0 as time increases, while it still allows many visits to potentially all state-action pairs (Singh et al. 2000). For comparison, we implement a different exploration strategy (Boltzmann exploration, Juliani 2016), which explores state-action pairs according to their softmaxed Q values.

For each $\epsilon-$greedy strategy, we explore learning rates $\delta \in [0.1, 0.2, 0.3, 0.4, 0.5]$. For each combination, we estimate a Q function according to Algorithm 1, and we run the simulations described in Section 5.2. We then compare each different combination in terms of the average market revenue that they yield (Equation 11).

Figure E3 shows the results, where we use as a baseline the $\epsilon-$greedy strategy for $\epsilon_0 = 0.5, \delta = 0.1$. The results are comparable. $\epsilon-$greedy with $\epsilon_0 = 0.9$ and $\delta = 0.1$ yields the highest market revenue; we chose these values for implementing Algorithm 1 in the evaluations of Section 5.

Figure E3 Comparison of alternative exploration strategies and learning rates



The $y-$axis shows improvements over the baseline, which is an $\epsilon-$greedy strategy with $\epsilon_0 = 0.5$ and $\delta = 0.1$. $\epsilon-$greedy with $\epsilon_0 = 0.9$ and $\delta = 0.1$ performs slightly better than alternative configurations. Error bars show 95% confidence intervals.

Appendix F Alternative algorithms for Q-function estimation

Algorithm 1 describes the most basic procedure for estimating a Q function. Alternative approaches could explore and learn potentially better policies. In this section, we discuss five alternative such approaches:

Static-rewards MDP (MDP-S): Lines 11 to 14 and line 16 of Algorithm 1 predict wages, demand, and supply for each period. Because these prediction algorithms are not perfect (Appendix D), they propagate errors. Would it be better to not update the expected rewards of each state and avoid this propagation of errors? MDP-S implements this approach by excluding lines 11 to 14 and line 16 of Algorithm 1.

Monte Carlo reinforcement learning (MC): An alternative approach to the time difference update in line 10 of Algorithm 1 is to use Monte Carlo simulations. Specifically, we use an "every visit Monte Carlo" (Choudhary 2018) approach, where, instead of estimating Equation 10 in line 10 of Algorithm 1, we average the returns of each action for every time taken at the end of each episode.

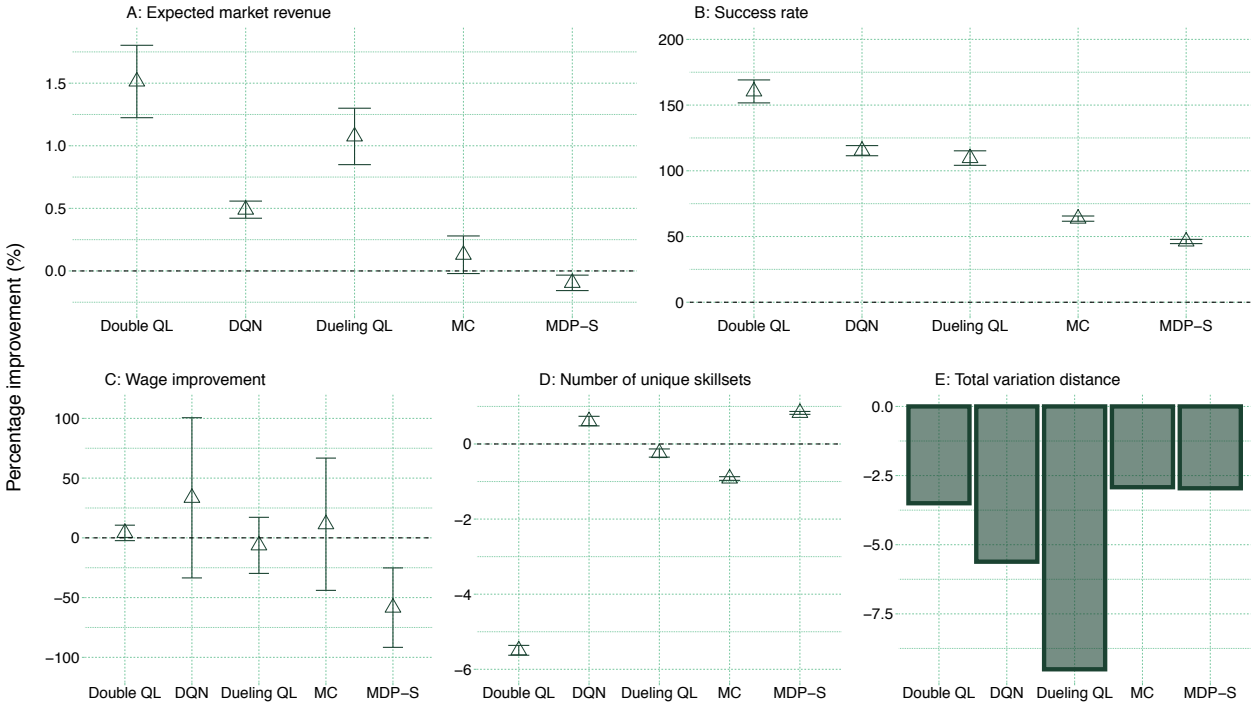
Deep Q-Learning (DQN): The objective of Equation 10 is to minimize the distance between the Temporal Difference Target (TD-Target) and the Q functions (Mnih et al. 2013, Oppermann 2018, Choudhary 2019). This distance can be expressed as a loss function that we minimize through standard gradient descent algorithms (Oppermann 2018). Deep Q-Learning estimates separately the TD-Target and Q functions through two different neural networks, such as the parameters of the Target network correspond to the parameters of the Q-Network at an earlier point in time. In other words, the Target Network is frozen in time, and gets updated after n iterations with the current parameters of the Q-Network (Oppermann 2018). For our implementation, we use networks with 3 layers, 128 and 256 hidden units with a relu activation function (Yoon 2019c). Furthermore, to increase the stability of the networks estimation, we use experience replay (Sections 3.4 and 3.5 of Oppermann 2018). We update the Target network at the end of each episode ($n = 1$ episode, Algorithm 1). Experimentation with alternative values for n did not yield significantly different results.

Double Q-Learning (Double QL): Deep Q-Learning tends to overestimate action-values, which sometimes could lead to unstable Q functions (Van Hasselt et al. 2016). Double Q-learning solves this issue by using two separate Q-value estimators, "each of which is used to update the other" (Yoon 2019). For our implementation, we follow Algorithm 1 in Yoon (2019), where we keep a model Q and a target model Q' so that we use Q' for choosing actions and Q for evaluating these actions. Similar to before, the two networks we use have 3 layers, 128 and 256 hidden units, and they use a relu activation function (Yoon 2019).

Dueling Q-Learning (Dueling QL): Dueling Q-Learning separates the representation of state values and actions, as, in many cases, it is not necessary to know the value of each action at every step (Wang et al. 2015). By doing so, Dueling-QL can learn which states are valuable without having to learn the exact value of each state-action. The networks we use for this implementation have 2 layers, 128 hidden units, and they use a relu activation function (Yoon 2019b).

Similar to Section 5.2, we compare through simulations, each one of the five approaches. We use Algorithm 1 as our baseline. Figure F4 shows the results (in the same form as in Figure 12). The results are complex. In terms of market revenue and success rate, more advanced models (DQN, Dueling QL, Double QL) perform significantly ($p < 0.001$) better than the temporal difference approach of Algorithm 1 (Figures F4A and B). However, they perform similar to Algorithm 1 in terms of wage improvement (Figure F4C). At the same time, many of the alternative approaches (e.g., Double QL) provide less diverse skillset recommendations than Algorithm 1 (Figures F4D and E). Overall, the better performance of these alternative approaches shows that (1) our results are robust to different Q-function estimations, and (2) that added complexity (Dueling-QL, Double QL) could pay off.

Figure F4 Comparison of different algorithms for estimating a Q-function



The $y-$axis shows the percentage improvement of each approach over Algorithm 1. Different estimation algorithms perform better in terms of market revenue and success rate (Figures A, B), on par in terms of wage improvement (Figure C), and relatively worse in terms of recommendations diversity (Figures D, E). Error bars show 95% confidence intervals.

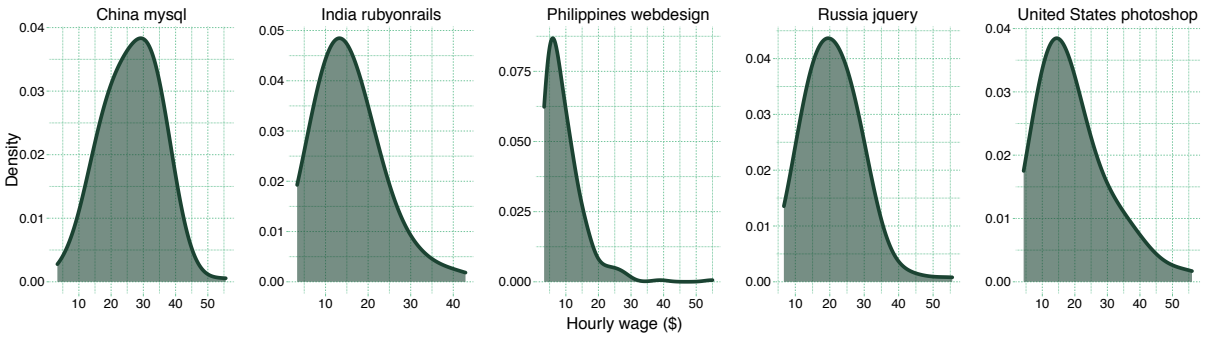
Appendix G Extension: Incorporating contractor's expertise

In the focal analysis, we have assumed that we know whether or not a contractor owns a skill. In practice, for each skill, we observe a diverse set of expertise levels. Figure G5 demonstrates the distributions of hourly wages for a randomly chosen set of five skill-country combinations. These hourly wages range from \$4 to \$55, even within the same country and for the same skill. Of course, this variance does not exist only in wages but also in a series of other characteristics, such as the number of completed jobs and certifications. This heterogeneity within skill-country combinations directly affects the rewards function of Equation 5. To account for this, we first assume that a vector $\boldsymbol{E}$ captures the levels of expertise of a set of skills $\boldsymbol{K}$. Then, we re-define the state of a contractor ($\boldsymbol{S}$) to account for vector $\boldsymbol{E}$ as follows:

$$\boldsymbol{S} =< \boldsymbol{K} .* \boldsymbol{E} > . \tag{18}$$

In the previous equation, $.*$ is the Schur product[2] (element-wise multiplication of two vectors) between the vector of available skills $\boldsymbol{K}$ and the expertise vector $\boldsymbol{E}$. Through this extended state definition, we transform the expected rewards function to account for the contractors' levels of expertise **E**.

Figure G5 The distributions of hourly wages for five different skill-country combinations



Even within the same country and for the same skill, contractors' hourly compensation wages range between 4\$ and 55\$. This variance signals the various levels of contractors' expertise.

[2] http://en.wikipedia.org/wiki/Hadamard_product_%28matrices%29

In practice, the true expertise of a contractor on a given skill is latent. In an online marketplace, however, we observe a series of characteristics that correlate with this latent expertise. First, we observe whether or not a contractor has taken a **certification** test associated with the skill at hand. Second, we observe the contractor's **past performance** on tasks that require the skill at hand. Finally, we observe (1) the number of times that contractors get **rehired** by the same employers, (2) the average **wage** that they charge, and (3) the total number of **jobs** that they complete on the platform. These signals form a vector $\boldsymbol{Z} = [$ certifications, feedback, rehires, wage, jobs $]'$.

Gaussian Mixture Model (GMM): We denote the latent expertise of a contractor i on a given skill s as (e_i^s). We assume for simplicity that $e_i^s \in \{$"none", "beginner", "expert""$\}$.[3] Our goal is to infer from the observed signals $(\boldsymbol{Z}_i)$ the conditional probability $\Pr(e_i^s = \text{expert}|\boldsymbol{Z}_i)$. In particular we assume that the density function of our feature vector $\boldsymbol{Z}$ is a mixture of Gaussians:

$$\Pr(\boldsymbol{Z}_i|\boldsymbol{\theta}, e_i^s) = \sum_{j \in \{\text{"beginner"},\ \text{"expert"}\}} \Pr(e_i^s = j|\boldsymbol{\theta}) \mathcal{N}(\boldsymbol{Z}_i|\mu_j, \Sigma_j) \tag{19}$$

where $\boldsymbol{\theta} = [\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j]'$ (i.e., the vector of parameters to estimate). From Bayes' rule we get:

$$\Pr(e_i^s = j|\boldsymbol{Z}_i\boldsymbol{\theta}) = \frac{\Pr(e_i^s = j|\boldsymbol{\theta}) \Pr(\boldsymbol{Z}_i|\boldsymbol{\theta}, e_i^s = j)}{\sum_j \Pr(e_i^s = j|\boldsymbol{\theta}) \Pr(\boldsymbol{Z}_i|\boldsymbol{\theta}, e_i^s = j)} \ . \tag{20}$$

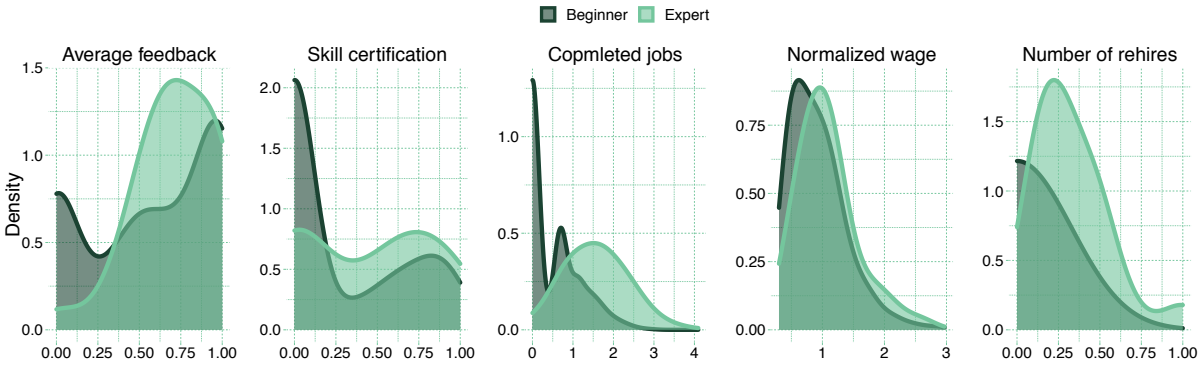
We estimate the parameter vector $\boldsymbol{\theta}$ through expectation maximization (Murphy 2012). (To avoid stacking to local maxima, we initialize the parameters at random, and we run the estimation 1,000 times. We pick the estimation that yields the lowest BIC score, Murphy (2012).) The final step is to tag the most likely level of expertise for each instance. To do so, we use the MAP estimate, i.e.:

$$e_{i*}^s = \arg\max_j \Pr(e_i^s = j|\boldsymbol{Z}_i, \boldsymbol{\theta}) \, . \tag{21}$$

Finally, note that we have simplified our estimation process by not algorithmically estimating the probability of a contractor to have the level of expertise "none." We assign this level when contractors do not list the skill at hand on their profiles.

[3] The extension to a higher number of levels of expertise is conceptually trivial.

Figure G6 "Beginner" and "expert" distributions for `wordpress`



Compared with "beginner" distributions, "expert" distributions are right-shifted across all five dimensions.

We build the GMM model using the focal dataset and present the results through a series of visualizations. Figure G6 presents the distributions of each one of the five dimensions of $\boldsymbol{Z}$ for "experts" and "beginners" for a randomly chosen skill, `wordpress`. For every dimension, the distribution of "expert" is shifted to the right compared with the distribution of "beginner." Furthermore, all "expert" distributions have statistically significant ($p < 0.0001$) greater means than the "beginner" ones. This indicates that the proposed model works as expected since it separates "experts" from "beginners" in all the observed dimensions of $\mathbf{Z}$. (Note that this is true for all the skills in our dataset.)

Figure G7 visualizes the normalized-per-country wage of a random subset of skills. The $y-$axis lists the "experts" and the $x-$axis the "beginners" average normalized wage. All skills lie above the equality line, thereby showing a positive "expert" - "beginner" difference. This positive difference verifies intuition that higher expertise correlates positively with wages.

The inclusion of the expertise estimates in the recommender framework is trivial. The different levels of expertise generate new states. Whenever a contractor chooses to learn a new skill, there is a non zero probability of becoming a "beginner" or an "expert" on that skill. Finally, contractors can improve their current skills, hence transition from a "beginner" to an "expert" state.

To summarize, this section provides a basic approach for estimating a contractor's expertise. The goal of this section is to showcase a simple way that the MDP framework can incorporate different

Figure G7 Wage difference between "beginners" and "experts"

Average normalized wage difference between "beginners" and "experts" according to the GMM predictions. All skills lie above the equality line, indicating that "experts" receive higher compensation wages than "beginners."

levels of expertise. More advanced approaches for estimating a contractor's expertise (Daltayanni et al. 2015, Christoforaki and Ipeirotis 2015, Kokkodis and Ipeirotis 2014, Kokkodis 2019, Kokkodis et al. 2019, Kokkodis and Lappas 2019, Kokkodis and Ransbotham 2020) will likely yield better results. However, implementation of these approaches is beyond the scope of this work.

References

Bakarov, Amir. 2018. A survey of word embeddings evaluation methods. *arXiv preprint* 1801.09536.

Brownlee, Jason. 2016. Time series prediction with lstm recurrent neural networks in python with keras. https://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/. [Online; accessed: 01-December-2019].

Chen, Tianqi, Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. *International Conference on Knowledge Discovery & Data Mining*. ACM, 785–794.

Choudhary, Ankit. 2018. Reinforcement learning: Introduction to monte carlo learning using the openai gym toolkit. https://www.analyticsvidhya.com/blog/2018/11/reinforcement-learning-introduction-monte-carlo-learning-openai-gym/. [Online; accessed: 8-October-2019].

Choudhary, Ankit. 2019. A hands-on introduction to deep q-learning using openai gym in python. https://www.analyticsvidhya.com/blog/2019/04/introduction-deep-q-learning-python/. [Online; accessed: 8-October-2019].

Christoforaki, Maria, Panagiotis G Ipeirotis. 2015. A system for scalable and reliable technical-skill testing in online labor markets. *Computer Networks* **90** 110–120.

Daltayanni, Maria, Luca de Alfaro, Panagiotis Papadimitriou. 2015. Workerrank: Using employer implicit judgements to infer worker reputation. *International Conference on Web Search and Data Mining*. ACM, 263–272.

Juliani, Arthur. 2016. Simple reinforcement learning with tensorflow part 7: Action-selection strategies for exploration. https://medium.com/emergent-future/simple-reinforcement-learning-with-tensorflow-part-7-action-selection-strategies-for-exploration-d3a97b7cceaf. [Online; accessed: 08-October-2019].

Kokkodis, Marios. 2019. Reputation deflation through dynamic expertise assessment in online labor markets. *The World Wide Web Conference*. ACM, 896–905.

Kokkodis, Marios, Panagiotis G. Ipeirotis. 2014. The utility of skills in online labor markets. *International Conference on Information Systems*.

Kokkodis, Marios, Theodoros Lappas. 2019. Your hometown matters: Popularity-difference bias in online reputation platforms. *Information Systems Research (published online)* .

Kokkodis, Marios, Theodoros Lappas, Sam Ransbotham. 2019. From lurkers to workers: Predicting voluntary contribution and community welfare. *Information Systems Research (published online)* .

Kokkodis, Marios, Sam Ransbotham. 2020. Asymmetric reputation spillover from agencies on digital platforms. (Working paper).

Le, Quoc, Tomas Mikolov. 2014. Distributed representations of sentences and documents. *International Conference on Machine Learning*. 1188–1196.

Mikolov, Tomas, Kai Chen, Greg Corrado, Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint* 1301.3781.

Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *arXiv preprint* 1312.5602.

Murphy, Kevin P. 2012. *Machine learning: A probabilistic perspective*. The MIT Press.

Nau, Robert. 2018. Statistical forecasting: Notes on regression and time series analysis. https://www.prnewswire.com/news-releases/snagajob-appoints-former-upwork-ceo-to-board-of-directors-300417689.html. [Online; accessed: 11-January-2019].

Oppermann, Artem. 2018. Self learning AI-agents part II: Deep Q-Learning. https://towardsdatascience.com/self-learning-ai-agents-part-ii-deep-q-learning-b5ac60c3f47. [Online; accessed: 08-October-2019].

Sayeed, Asad, Clayton Greenberg, Vera Demberg. 2016. Thematic fit evaluation: An aspect of selectional preferences. *Workshop on Evaluating Vector-Space Representations for NLP*. 99–105.

Shevade, SK, SS Keerthi, C Bhattacharyya, KRK Murthy. 2000. Improvements to the SMO algorithm for SVM regression. *Transactions on Neural Networks* **11**(5) 1188–1193.

Singh, Satinder, Tommi Jaakkola, Michael L Littman, Csaba Szepesvári. 2000. Convergence results for single-step on-policy reinforcement-learning algorithms. *Machine Learning* **38**(3) 287–308.

Van Hasselt, Hado, Arthur Guez, David Silver. 2016. Deep reinforcement learning with double q-learning. *Conference on Artificial Intelligence*. AAAI, 2094–2100.

Wang, Ziyu, Tom Schaul, Matteo Hessel, Hado Van Hasselt, Marc Lanctot, Nando De Freitas. 2015. Dueling network architectures for deep reinforcement learning. *arXiv preprint* 1511.06581.

Yoon, Chris. 2019a. Double deep q networks: Tackling maximization bias in deep q-learning. https://towardsdatascience.com/double-deep-q-networks-905dd8325412. [Online; accessed: 08-October-2019].

Yoon, Chris. 2019b. Dueling deep q networks: Dueling network architectures for deep reinforcement learning. https://towardsdatascience.com/dueling-deep-q-networks-81ffab672751. [Online; accessed: 08-October-2019].

Yoon, Chris. 2019c. Vanilla deep q networks. https://towardsdatascience.com/dqn-part-1-vanilla-deep-q-networks-6eb4a00febfb. [Online; accessed: 08-October-2019].