Client Clustering for Hiring Modeling in Work Marketplaces

Vasilis Verroios*
Stanford University
verroios@stanford.edu

Ramesh Johari*
Stanford University
ramesh.johari@stanford.edu

Panagiotis Papadimitriou
oDesk Research
papadimitriou@elance-odesk.com

Hector Garcia-Molina
Stanford University
hector@cs.stanford.edu

ABSTRACT

We study the problem of grouping clients of an online work marketplace into clusters, such that in each cluster clients are similar with respect to their hiring criteria. Such a separation allows the marketplace to "learn" more accurately the hiring criteria in each cluster and recommend the right contractor to each client, for a successful collaboration. We provide a Maximum Likelihood definition of the "optimal" client clustering, based on a logit model. To find the optimal clustering, we propose a scalable Expectation-Maximization algorithm. We study the baseline behavior of the algorithm using synthetic data and we verify that our approach yields significant gains compared to the baseline approach of "learning" the same hiring criteria for all clients, using a real dataset of 865,000 hiring decisions provided by oDesk. In addition, we analyze the clustering results on the oDesk dataset and we find interesting differences between the hiring criteria of the different groups of clients.

1. INTRODUCTION

Online work marketplaces such as oDesk.com, Elance.com and Freelancer.com help "clients" and "contractors" across the globe to connect with each other and work for more than $1 billion in annual contractor earnings just in 2014. Typically, in such marketplaces, contractors apply to a job posted by a client and clients hire the applicant(s) that seems to be the best fit for the job posted. As these platforms grow, a fundamental problem they have to solve is the understanding of successful client hiring practices so that they can help clients make the right hiring decisions. Without such help, clients will have to deal with the friction of screening tens to hundreds of contractors to determine the ideal candidates for their jobs. The screening process is not only time-consuming, but it is also error-prone, since clients often lack the necessary knowledge to assess the qualifications of contractors (e.g., education and work experience from schools and companies that are unknown to a client).

*Work done while authors were at oDesk Research

WWW 2015 Florence, Italy

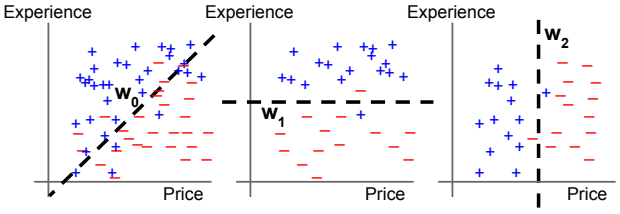


Figure 1: Right: The hiring decisions of all of the marketplace clients in a two-dimensional feature space. Middle: The hiring decisions of quality optimizers. Left: The hiring decisions of cost optimizers.

Understanding and modeling the hiring behavior of clients is challenging not only due to the variety of jobs that are posted and the diversity of contractors, but also due to the heterogeneity of client hiring criteria. For example, two different clients that have posted two seemingly similar jobs looking for "php developers" may be looking for totally different people. Say that the first client is a *quality* optimizer that is willing to pay a high hourly rate to get the most qualified contractor while the second client is a *cost* optimizer who is willing to take the risk of working with an inexperienced contractor to reduce his costs. To make recommendations that satisfy both of these clients we would ideally develop a dedicated model for each client. However, in practice, developing a model for each client is not an option, since the marketplace rarely has sufficient data points for a single client to make training possible.

Although all clients are not the same, there are usually sufficiently large groups of clients with similar hiring criteria that can provide us with data for model training. To illustrate our hypothesis, we show in Figure 1 the hiring decisions in a marketplace that is composed of quality and cost optimizers. In the first plot of the figure we illustrate the hiring decisions of all clients in a two-dimensional feature space. Each "+" point looks at an application that ended up in a hire while a "−" point looks at an application that got rejected. The point positions on the X-axis indicate the bid prices asked by the contractors and the positions on the Y-axis indicate the years of contractor experience. A linear model, such as logistic regression, trained on all of the client decisions would "learn" a linear separator $\mathbf{w_0}$ to distinguish hired from rejected applications. Such a separator would misclassify many rejected applications as hires ("-" points to the left of $\mathbf{w_0}$) and many hires as rejected applications ("+" points to the right of $\mathbf{w_0}$). However, if we could split the clients into quality (middle plot) and cost optimizers (right plot) we could then learn a different model for each client

group. The derived separators $\mathbf{w_1}$ and $\mathbf{w_2}$ would then almost perfectly separate the hires in each of the two groups.

One naive approach to find the groups of clients with similar hiring criteria is to exhaustively examining all possible partitions of clients into clusters. Nevertheless, such a brute-force approach is computationally prohibitive: for K clients and C clusters, the number of possible partitions is C^K, i.e., exponential to the number of clients. Moreover, traditional clustering methods (e.g., k-means) that would cluster clients based on attributes like age or occupation are also not suitable for our case: two clients may have the exact same characteristics (age, occupation, etc.), but very different criteria on hiring contractors. Clearly, a clustering method suitable for our problem must be based on the clients' decisions and not just attributes like age and occupation.

In this paper, we propose a simple, yet effective and scalable approach for client clustering, based on Expectation Maximization with *hard* assignments (hard EM). As input we are given a fixed number of clusters C and the positive/negative samples of all clients. Our algorithm then starts with a random assignment of clients into C clusters and for each cluster it applies sparse logistic regression to learn a linear separator in each cluster. It then re-assigns a client to the cluster that best "explains" her choices. These two steps are repeated until convergence, i.e., until no re-assignments are needed. After the last step, the algorithm has computed a partition of clients into clusters along with the criteria of clients in each cluster; given by sparse logistic regression.

In large-scale platforms like oDesk, any unnecessary model or algorithmic complexity incurs a critical cost in terms of computational overhead and software maintenance. Hence, our main focus in the paper is to explore the effect and gains of applying client clustering, rather than quantify how much better our model is, compared to other similar models proposed in the past (e.g., mixture models proposed for clusterwise regression [4, 20] or found in the marketing and economics literature [1, 9, 13, 16]). To this end, in our evaluation we focus on *a)* studying the gains compared to the simplest and most cost-effective approach of using the same hiring model for all clients and *b)* analyzing the hiring practices in the clusters of clients our algorithm produces. To the best of our knowledge, our study is the first one on the recognition and analysis of different hiring practices in online work marketplaces.

Furthermore, while similar to our model, the models proposed in the past have two critical limitations for the problem we study:

1. Most previous models apply clustering directly on the samples of the dataset. On the contrary, our clustering takes place in a different level: we cluster the clients that produced the samples.
2. As previous models were designed to solve different problems, they come with unnecessary complexity or do not capture important aspects of the problem we are trying to solve in this paper. For example, in most of the models in the marketing and economics literature, a client's decision refers to the selection of a product or brand from a set of "fixed" alternatives. On the contrary, in our model a client's decision is the acceptance or rejection of a "unique" application. (Each application is "unique" since it can only refer to a single job posting.)

Other approaches, like collaborative filtering (CF) techniques (e.g., [2, 3, 5, 12, 17, 22]), fit well in domains like electronic commerce or streaming media recommendations, but fail to deliver in online job marketplaces: each contractor cannot be considered as a "fixed" item as with movies or products. For instance, a contractor may be much more appropriate for a task involving Fortran debugging than Python debugging if she is a Fortran expert but a Python novice. That is, clients do not really "vote" for a contractor when they hire her but "vote" for her application on a specific task. Since common CF approaches rely on users "voting" on common sets of items, they cannot be directly applied in this case, where each item/application is unique.

Nevertheless, our approach is also applicable in domains other than the work marketplaces' domain. For example, in the streaming media domain, a horror movie may be much more appropriate for a specific user to watch during the night compared to watching it during the afternoon; just like a contractor may be much more appropriate for Fortran debugging compared to Python debugging. In this case, the decisions of clients can be modeled by various attributes related to the current context of a user, as, for example, the set of actors in the movies explored by the user in her current search for a movie. Note that in this movie recommendation example where the number of attributes can be extremely large, the sparsity in our model can be a key component for effectively applying our algorithm.

In our evaluation we use a large dataset of 865,000 accept/reject decisions, provided by oDesk in the period of September 2012 to March 2013. The results show that the predictive hiring models trained on the clusters yielded by our algorithm can improve the performance of a global model (i.e., a single hiring model for all clients) by 43%. Furthermore, our analysis on the clusters produced by our method reveals some very interesting differences on the hiring practices of different client groups. Finally, we perform extensive experiments on synthetic datasets in order to fully understand the behavior of our algorithm regarding the convergence to the ground truth (which is not available in a real dataset) and the algorithm's scalability.

In summary, our contributions are the following:

- We provide a Maximum-Likelihood formulation of the problem of clustering heterogeneous clients to improve the accuracy of a predictive hiring model in Section 2.
- We present a hard EM algorithm for this problem, in Section 3.
- We present the oDesk dataset and discuss our findings when applying our algorithm, in Section 4.2.
- With synthetic data experiments we illustrate our algorithm's baseline behavior and we explore other directions (which cannot be covered by the oDesk dataset) that indicate our algorithm's behavior in the general case, in Section 4.1.

1.1 Running Example

We will use a running example throughout the definition of our model and algorithm in Sections 2 and 3.

For simplicity, let us assume that there are only **three** features, experience, score, bidding, affecting a client's decision and that all information for the contractors' applications are organized in a single table:

Apps(clientID, experience, score, bidding, decision)

The semantics of the 5 columns are the following:

1. clientID: The client that opened the task to which the current application refers.
2. experience: The total number of hours the contractor(applicant) has worked in the platform, i.e., the total number of hours in the contractor's past contracts with clients.
3. score: The aggregated rating of the contractor based on the past reviews from the clients she worked for.
4. bidding: The amount asked by the contractor for performing the task.
5. decision: The decision of the client ("APPROVED"/ "REJECTED") on this application.

In practice, the three features, experience, score, bidding, are normalized so that their value range is $[0.0, 1.0]$. For example, the bidding can be normalized by the maximum amount a contractor may ask for a task; a restriction that could be enforced by the platform's provided functionality.

In our example, the dataset consists of only **four** clients, each having **ten** contractor applications approved and **ten** contractor applications rejected. In addition, we want to form **two** clusters, i.e., we want to split the four clients into two groups such that the clients in each group have "very similar" criteria regarding the experience, score, and bidding of an application. Our model in the next section quantifies the notion of "similar" by defining the *optimal* clustering.

2. PROBLEM DEFINITION

In this section, we formally define the problem of finding the optimal client partition based on the clients' hiring criteria. Intuitively, our definition requires that the reject/accept applications in each cluster of clients are as "well-separated" as possible. For example, the reject/accept applications ("−"s/"+"s) in the middle and right side of Figure 1 are "well-separated"; a different partition of clients into two clusters could result in having "+"s diffuse over the "−"s area, and the opposite. We use a logit model to quantify how "well-separated" the applications in one cluster are. Based on the cost defined by the logit model, the optimal partition of clients is the one minimizing the aggregate cost across all clusters.

We start by describing the dataset notation and the cost for a single cluster and then define the clustering optimization problem (equations (11) to (13)).

2.1 Dataset Notation

All the past applications are stored in a single table:

Apps(clientID, a1, a2, ..., aF, decision)

The features describing each application are denoted by a1, a2, ..., aF and they are normalized so that their value range is $[0.0, 1.0]$. In our running example, we use only three features: a1 ≡ experience, a2 ≡ score, a3 ≡ bidding.

We denote by $\mathbf{x_i}$ the row i in table Apps (Apps[i]), projected over columns a1, a2, ..., aF. Thus, in our running example, each $\mathbf{x_i}$ is a 3-dimensional vector, e.g., if an application involves an experience of 0.2, a score of 0.9, and a bidding of 0.8, $\mathbf{x_i} = (0.2, 0.9, 0.8)^T$.

In addition to $\mathbf{x_i}$, we use $\mathbf{u_i}$ to express the clientID of row i (Apps[i].clientID), in a 1-of-K scheme. In our running example, where we have $K = 4$ clients, if the second client approved/rejected application i, $\mathbf{u_i} = (0, 1, 0, 0)^T$.

To simplify notation we split past applications into two subsets $\{\mathcal{P}, \mathcal{N}\}$, such that:

$$\mathcal{P} = \{(\mathbf{x_i}, \mathbf{u_i}) \mid \text{Apps}[i].\text{decision} = \text{APPROVED}\} \quad (1)$$

$$\mathcal{N} = \{(\mathbf{x_i}, \mathbf{u_i}) \mid \text{Apps}[i].\text{decision} = \text{REJECTED}\} \quad (2)$$

In the running example, $|\mathcal{P}| = 40$ since each of the four clients has approved ten contractor applications, and $|\mathcal{N}| = 40$ since each of the four clients has rejected ten applications.

Moreover, we denote with:

- K: the number of clients.
- C: the number of clusters.
- F: the number of features.

2.2 Single-Cluster Cost

The single-cluster cost is based on the logistic regression model. Here, we give a brief overview of logistic regression in the context of our running example. Note that, in this section, we focus only on the applications $\{\mathcal{P}, \mathcal{N}\}$ of the clients that belong to a single cluster.

We denote by $\mathbf{w}$ the vector expressing the criteria of clients for approving/rejecting the applications. Note that all the clients of a cluster share the same $\mathbf{w}$. In the running example, a $\mathbf{w} = (1.0, 0.0, 0.0)^T$ expresses that clients prefer contractors with a lot of experience and do not care about the score and the bidding in an application. (In practice, $\mathbf{w}$ involves an additional coefficient for the general bias. That is, in our running example, an application $\mathbf{x_i} = (0.2, 0.9, 0.8)^T$ would be extended with a constant term on a fourth dimension: $\mathbf{x_i}$ would become $(0.2, 0.9, 0.8, 1.0)^T$, and $\mathbf{w}$ would become a 4-dimensional vector.)

In logistic regression, the probability of an application i being approved is given by the logistic function:

$$g(\mathbf{w^T x_i}) = \frac{1}{1 + e^{-\mathbf{w^T x_i}}} \quad (3)$$

That is,

$$P(\mathbf{x_i}\ \textit{approved}|\mathbf{w}) = g(\mathbf{w^T x_i}) \quad (4)$$

$$P(\mathbf{x_i}\ \textit{rejected}|\mathbf{w}) = 1 - g(\mathbf{w^T x_i}) \quad (5)$$

Therefore, as the value of the dot product $\mathbf{w^T x_i}$ approaches $+\infty$, $P(\mathbf{x_i}\ \textit{approved}|\mathbf{w})$ approaches 1.0, while when $\mathbf{w^T x_i}$ approaches $-\infty$, $P(\mathbf{x_i}\ \textit{rejected}|\mathbf{w})$ approaches 1.0.

The objective in logistic regression is finding the criteria $\mathbf{w}$, maximizing the likelihood:

$$P(\{\mathcal{P}, \mathcal{N}\}|\mathbf{w}) = \prod_{\mathcal{P}} g(\mathbf{w^T x_i}) \prod_{\mathcal{N}} (1 - g(\mathbf{w^T x_i})) \quad (6)$$

Taking into account regularization, the cost of a single cluster is the negative log-likelihood plus a regularization term involving a hyperparameter λ and the 1-norm of the criteria vector $\mathbf{w}$:

$$Cost(\mathbf{w}) : \lambda \|\mathbf{w}\|_1 - \sum_{\mathcal{P}} \ln(g(\mathbf{w^T x_i})) - \sum_{\mathcal{N}} \ln(1 - g(\mathbf{w^T x_i})) \quad (7)$$

2.3 Optimal Client Partitioning

In this section, we generalize the model to many clusters. Thus, our dataset $\{\mathcal{P}, \mathcal{N}\}$ refers to the applications from all clients.

We use the matrix $\mathbf{M} = [\mathbf{m_1}, \ldots, \mathbf{m_C}] \in \{0,1\}^{K \times C}$ to express the clients' membership, i.e., how the K clients are partitioned into C clusters. Column j, $\mathbf{m_j}$, gives the clients that belong to cluster j, while we denote by $\mathbf{m'_k}$ the row k of $\mathbf{M}$, which gives the cluster where client k belongs. In our running example, suppose that the first cluster contains only the third client while the second cluster contains the other three clients. In that case,

$$\mathbf{m_1} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \ \mathbf{m_2} = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \end{pmatrix}, \ \mathbf{M} = \begin{pmatrix} 0 & 1 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}$$

and

$$\mathbf{m'_1} = (0,1), \ \mathbf{m'_2} = (0,1), \ \mathbf{m'_3} = (1,0), \ \mathbf{m'_4} = (0,1)$$

Therefore, the dot product $\mathbf{u_i^T m}_j$ is 1 if the client that approved/rejected application i belongs to cluster j and 0 otherwise. For instance, if the second client approved/rejected application i and we have the same clustering as in the example above, $\mathbf{u_i^T m}_1 = (0,1,0,0)(0,0,1,0)^T = 0$, while $\mathbf{u_i^T m}_2 = (0,1,0,0)(1,1,0,1)^T = 1$.

The criteria vector for cluster j is given by $\mathbf{w_j}$. We use the matrix $\mathbf{W} = [\mathbf{w_1}, \ldots, \mathbf{w_C}] \in \mathbb{R}^{F \times C}$ to refer to the union of vectors for all clusters.

The likelihood of the evidence $P(\{\mathcal{P}, \mathcal{N}\}|\mathbf{W}, \mathbf{M})$ becomes:

$$\prod_{j=1}^{C} \left(\prod_{\mathcal{P}} g(\mathbf{w_j^T x_i})^{\mathbf{u_i^T m_j}} \prod_{\mathcal{N}} (1 - g(\mathbf{w_j^T x_i}))^{\mathbf{u_i^T m_j}} \right) \quad (8)$$

Note that the term for a cluster j involves only the applications of clients that belong to that cluster. (For all of these applications the exponent $\mathbf{u_i^T m_j}$ is 1, while for all other applications that do not belong to the cluster j the exponent $\mathbf{u_i^T m_j}$ is 0.)

The log-likelihood, $\ell(\mathbf{W}, \mathbf{M})$, becomes:

$$\sum_{j=1}^{C} \left(\sum_{\mathcal{P}} \mathbf{u_i^T m_j} \ln(g(\mathbf{w_j^T x_i})) + \sum_{\mathcal{N}} \mathbf{u_i^T m_j} \ln(1 - g(\mathbf{w_j^T x_i})) \right) \quad (9)$$

Therefore, the cost of a client partition defined by membership and criteria matrices $\mathbf{M}$ and $\mathbf{W}$, is:

$$Cost(\mathbf{W}, \mathbf{M}) : \ \lambda \sum_{j=1}^{C} \|\mathbf{w_j}\|_1 - \ell(\mathbf{W}, \mathbf{M}) \quad (10)$$

Note that the cost involves the sum of the regularization terms for each cluster.

Our objective is to find the membership and criteria matrices that solve the following optimization problem:

$$\min_{\mathbf{W}, \mathbf{M}} \quad Cost(\mathbf{W}, \mathbf{M}) \quad (11)$$

$$\text{s.t.} \quad \|\mathbf{m'_k}\|_1 = 1, \ \forall k \in \{1, \ldots, K\} \quad (12)$$

$$\mathbf{M} \in \{0,1\}^{K \times C}, \ \mathbf{W} \in \mathbb{R}^{F \times C} \quad (13)$$

The constraint (12) expresses that each client must be part of exactly one cluster, i.e., we do not allow overlapping clusters.

3. ALGORITHM

The exhaustive approach for solving the optimization problem in equations (11)-(13) involves a $O(C^K)$ time complexity; for C clusters and K clients. Therefore, we propose a scalable algorithm based on Expectation Maximization with hard assignments. In each iteration two steps are involved:

- *E step:* compute the optimal client memberships, i.e., the optimal $\mathbf{M}$, while keeping $\mathbf{W}$ fixed.
- *M step:* compute the optimal client criteria for each cluster, i.e., the optimal $\mathbf{W}$, while keeping $\mathbf{M}$ fixed.

Algorithm 1

Input: C: the number of clusters
$\mathcal{P}$: applications approved
$\mathcal{N}$: applications rejected
Output: $\mathbf{W}$: local-optimum criteria matrix
$\mathbf{M}$: local-optimum membership matrix

```
M:= randomly assign the clients into C clusters
while M ≠ M_pre do
    M_pre := M
    W := solve problem of (11)-(13), with M fixed (M step)
    M := solve problem of (11)-(13), with W fixed (E step)
end while
```

Our algorithm is given by Algorithm 1. The input is the set of all clients' applications, $\{\mathcal{P}, \mathcal{N}\}$, along with the number of clusters C. Note that in practice there are many ways to compute the number of clusters to use as input. The simplest approach is to try several different C values and keep the one maximizing a metric like Mean Average Precision or Discounted Cumulative Gain on a testing set.

In each E step, the value of the objective function in (11) decreases or remains the same; in the worst case there are no changes in the client memberships that would decrease the value of the objective function. Likewise, in each M step the value of the objective function always decreases; or at least remains the same. Hence, the algorithm eventually converges to a minimum; when the client memberships remain the same for two consecutive iterations. Nevertheless, the minimum may be a local minimum and not a global one, since the problem of (11)-(13) is not convex. In practice, we run Algorithm 1 more than once, using different initial assignments of clients to clusters, and keep the solution that gives the lowest value for the objective function. In section 4.1, we examine how the number of runs affects the convergence of our algorithm to the global minimum using synthetic data; so that the true global optimum is known to us in advance.

One of the main advantages of our algorithm is its scalability. In the E step, a single pass over the clients is needed. (For each client we find the cluster that "explains" better her decisions on her applications, while keeping the criteria for each cluster fixed.) In the M step, we just need to solve C sparse logistic regression problems, i.e., one problem per cluster. (Solving sparse logistic regression problems is a process that has been highly optimized for large datasets, e.g., [8], [11].) In Section 4.1.7, we discuss our scalability experiments and results.

3.1 E step

In the E step, we keep criteria matrix $\mathbf{W}$ fixed and we find for each client the cluster that "explains" better her decisions. That is, if U_a is the set of applications that were approved/rejected by a client a, we compute for each cluster j the log-likelihood $\ell(\mathbf{W}; j, U_a)$:

$$\sum_{\mathcal{P} \cap U_a} \ln(g(\mathbf{w_j^T x_i})) + \sum_{\mathcal{N} \cap U_a} \ln(1 - g(\mathbf{w_j^T x_i})) \quad (14)$$

Then, we assign client a to the cluster that gives the highest $\ell(\mathbf{W}; j, U_a)$ (or, equivalently, the lowest negative log-likelihood). In our running example, if the second cluster gives the highest $\ell(\mathbf{W}; j, U_3)$ for the third client, the third row of $\mathbf{M}$, i.e., $\mathbf{m'_3}$, becomes $(0, 1)$.

At the end of E step, the assignment changes for each client will be reflected on the membership matrix $\mathbf{M}$.

3.2 M step

In the M step, we keep $\mathbf{M}$ fixed and we find for each cluster j the criteria vector $\mathbf{w_j}$ that best "explains" the clients' decisions in that cluster. That is, for a cluster c, if U is the set of applications that were approved/rejected by the clients in c, we solve the following sparse logistic regression problem:

$$\min_{\mathbf{w_c}} \left(\lambda \|\mathbf{w_c}\|_1 - \sum_{\mathcal{P} \cap U} \ln(g(\mathbf{w_c^T x_i})) - \sum_{\mathcal{N} \cap U} \ln(1 - g(\mathbf{w_c^T x_i})) \right) \quad (15)$$

The optimal solutions to the logistic regression problems form the $\mathbf{W}$ for the next E step.

4. EXPERIMENTAL RESULTS

We perform a series of experiments on synthetic and real datasets:

- In Section 4.1 we use synthetic data to study the impact of various parameters on the algorithm performance. The goal of this section is to answer questions such as "how many samples are needed to converge to the ground truth given a number of dimensions and clusters?" We use our findings from the synthetic data experiments to understand the effects taking place in the oDesk data experiments.
- In Section 4.2.1 we evaluate the performance of our algorithm on a real dataset obtained by oDesk. The results show that our clustering approach can scale to large datasets and improve the accuracy of base classifiers by more than 40%. We also observe that the improvements of our algorithm are negligible in cases where the training dataset is small relatively to the model complexity, since the clusters that are formed do not contain sufficient training data for model learning.
- Finally, in Section 4.2.2 we present a qualitative analysis of the clusters that are formed by our algorithm. The analysis shows that our algorithm not only improves the accuracy of hiring predictions, but it also reveals latent characteristics of clients that use the work marketplace in different ways.

4.1 Synthetic Data

Through our synthetic data experiments we seek the answer to the following questions: 1) how close to the ground truth do we get given a number of samples and clusters, 2) how many samples are needed to converge to the ground truth given a number of dimensions and clusters, 3) how does the number of local optima examined affects the convergence to the ground truth, 4) how does the distance between the ground truth criteria vectors affect the convergence to the ground truth, 5) how does the number of clients (sharing a fixed number of samples) affect the distance to the ground truth, 6) how does the distribution of samples to clients affect the distance to the ground truth, and 7) how does the running time increase as the number of dimensions and samples increases.

Ground Truth Criteria: We produce accepted/rejected applications using ground truth criteria vectors. Specifically, for a number of clusters C, we use C different ground truth criteria vectors $\mathbf{w}$, with each vector having components that are 1 and -1 (except for the *sparse* case we study in Section 4.1.2). In addition, we use the hamming distance to quantify how "different" the C criteria vectors are (the hamming distance between two vectors is simply the number of components in which the two vectors differ). We denote the distance between criteria vectors by HD, while we denote by D the number of dimensions.

For example, for $D = 8$ dimensions, $C = 4$ clusters and a distance $HD = 4$ between each two vectors, we produce the following 4 vectors:

$$\mathbf{w_1} = (-1,-1,1,1,1,1,1,1)^T, \mathbf{w_2} = (1,1,-1,-1,1,1,1,1)^T$$
$$\mathbf{w_3} = (1,1,1,1,-1,-1,1,1)^T, \mathbf{w_4} = (1,1,1,1,1,1,-1,-1)^T$$

Each application $\mathbf{x_i}$ is a D-dimensional vector, with each component picked uniformly at random from $[0.0, 1.0)$. Once we produce an application, we assign it uniformly at random to one of the K clients (except for the experiments we discuss in Section 4.1.6). We consider a uniform distribution of clients to clusters, e.g., if we have $C = 4$ clusters and $K = 1000$ clients, the first 250 clients are assigned to the first cluster, the next 250 to the second cluster and so on.

After we assign an application to a client, and consequently to a cluster, we compute the probability, p, of the application being approved/rejected; using the logistic function (Equation 3). Then, the application becomes an approved application with probability p; or gets rejected with probability $1 - p$.

Metrics: We use the angle distance to quantify the proximity of the criteria vectors computed by the algorithm to the ground truth vectors. For example, assume that the algorithm computes the following 4 vectors in our previous example with the ground truth vectors $\mathbf{w_1}$ to $\mathbf{w_4}$:

$$\mathbf{w_a} = (-1,-1,1,1,1,1,1,0)^T, \mathbf{w_b} = (1,1,-1,-1,1,1,1,0)^T$$
$$\mathbf{w_c} = (1,1,1,1,-1,-1,1,0)^T, \mathbf{w_d} = (0,1,1,1,1,1,-1,-1)^T$$

We match each vector to the closest (in terms of the angle distance) ground truth vector (e.g., $\mathbf{w_a}$ matches with $\mathbf{w_1}$) and we compute the average angle distance across all pairs. In this example, the average angle for the 4 vector pairs is 20.7 degrees.

Besides the average angle distance between the computed and the ground truth vectors, we also use a second metric: the number of samples(applications) needed so that the angle distance between the computed and the ground truth vectors is less than 10 degrees. As expected, when the number of samples increases, the algorithm produces a more accurate outcome. This metric quantifies how many samples we need, in each case, to get an outcome that is very close (within 10 degrees) to the ground truth.

4.1.1 Number of samples

We first try to quantify how our algorithm is affected by the number of samples available. We fix the number of dimensions, D, to 8, the distance between ground truth vectors, HD, to 4, and the number of clients, K, to 1000.

Figure 2 depicts the behavior of our algorithm as we increase the number of applications (X-axis) from 8000 to

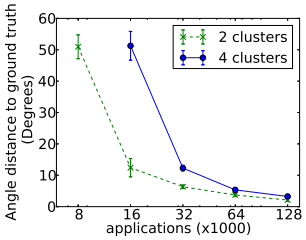
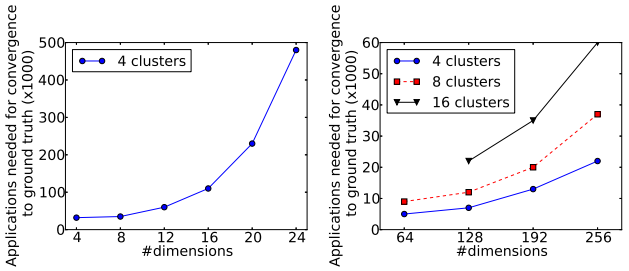


Figure 2: Angle distance to ground truth, as we increase the number of samples(applications) from 8 to 128 applications per client, for 2 and 4 clusters.



(a) *non-sparse* case (b) *sparse* case

Figure 3: In the *non-sparse* case, we plot the number of applications needed to get a 10-degree angle distance to ground truth, as we increase the number of dimensions from 4 to 24, for 4 clusters. In the *sparse* case, we increase the number of dimensions from 64 to 256, for 4, 8, and 16 clusters.

128000, i.e., from 8 applications per client, to 128 applications per client, on average. The Y-axis gives the angle distance between the vectors computed by our algorithm and the ground truth vectors. We run experiments for $C = 2$ and 4 clusters and we plot one curve for each C value.

For $C = 2$ we see a huge increase in accuracy (the distance to the ground truth drops from 50 to 12 degrees) when we go from 8 to 16 applications per client. As we keep increasing the number of applications, the accuracy further increases and approaches a zero-degree distance to the ground truth (less than 3 degrees for 128 applications per client).

For $C = 4$ the steep decrease in the distance to the ground truth, happens from 16 to 32 applications per client. In fact, if we look closely at the two curves we observe that for 4 clusters we need, roughly, twice as many applications as we need for 2 clusters, to reach the same distance to the ground truth. This can be explained by the fact that we have twice as many clusters and, hence, each cluster is assigned half the applications. For instance, for 16000 applications, for $C = 2$, each cluster has 8000 applications, while for $C = 4$, each cluster has 4000 applications. Therefore, we expect that for $C = 4$ and 16000 applications we will get an accuracy close to the accuracy for 2 clusters and 8000 applications.

4.1.2 Number of dimensions

In Figure 3(a), we plot the number of applications needed (Y-axis) to reach within a 10-degree distance to the ground truth, as we increase the number of dimensions, D, from 4 to 24 (X-axis). The number of clusters, C, is 4, the number of clients, K, is 1000, and the distance between ground truth vectors, HD, is $\frac{D}{2}$, e.g., for 20 dimensions, HD is set to 10.

As we increase D from 4 to 12, the number of applications needed for the 10-degree distance, increases slowly. For D greater than 12, there is a steep increase for the number of applications needed, reaching almost 500 per client for 24 dimensions.

We also tried a second experiment showing the behavior of our approach when the number of dimensions increase, however, this time we used *sparse* ground-truth criteria vectors. That is, out of all dimensions/criteria only a few of them play an important role in the clients' decisions. In particular, the criteria in the ground truth vectors that play an important role in clients' decisions, have a weight of 10 or -10, while all other criteria have a weight of .1. Therefore, for each cluster, we generate a ground truth vector having 4 of its components equal to 10, 4 equal to -10, and the rest equal to .1. For instance, for $D = 16$ dimensions and $C = 2$ clusters, we generate the following two vectors:

$$\mathbf{w_1} = (10, -10, 10, -10, 10, -10, 10, -10, .1, .1, .1, .1, .1, .1, .1, .1)^T$$

$$\mathbf{w_2} = (.1, .1, .1, .1, .1, .1, .1, .1, 10, -10, 10, -10, 10, -10, 10, -10)^T$$

In the experiments we run with *sparse* vectors, we use $D = 64, 128, 192$, or 256 dimensions and $C = 4$ clusters. For example, when $D = 64$, the first vector has 10 or -10 for dimensions 0 to 7 and .1 for all other 56 dimensions, the second vector has 10 or -10 for dimensions 8 to 15 and .1 for all other 56 dimensions, and so on. Note that the hamming distance between any two vectors, HD, is fixed to 16, regardless of the number of dimensions.

As in Figure 3(a), Figure 3(b) depicts the number of applications needed (Y-axis) to reach a 10-degree distance to the ground truth, as we increase the number of dimensions (X-axis), for the *sparse* case. For 4 clusters, although the number of dimensions is a lot higher than in the *non-sparse* case of Figure 3(a), the number of applications needed for the 10 degree distance, is two orders of magnitude lower; requiring around 20 applications per client for 256 dimensions.

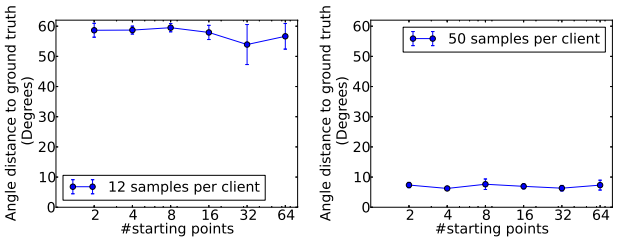
For a larger number of clusters, i.e., $C = 8$ and $C = 16$ clusters, we need considerably more applications to reach the 10-degree distance, compared to $C = 4$ clusters. Still, even for $C = 16$ and 256 dimensions, we need less than 60 applications per client, which is an order of magnitude lower than the 500 applications per client for $C = 4$ and 24 dimensions, in the *non-sparse* case of Figure 3(a).

4.1.3 Local Optima Examined

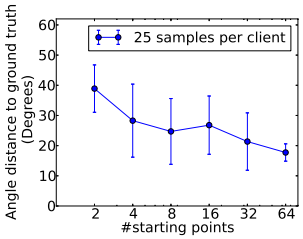
As discussed in Section 2 the optimization problem we are trying to solve is not convex. Hence, our EM algorithm may not converge to the global optimum but to a local optimum. In the experiments discussed so far, in every run of our algorithm we try 10 different starting points and, eventually, out of all the local minima, we keep the solution having the minimum objective function value (equation 11).

In the experiments of this section, we explore how the number of different starting points examined (or equivalently the local optima examined) affect the quality of the algorithm's solution. In Figure 4(c), we plot the angle distance to ground truth (Y-axis), as we increase the number of starting points examined (X-axis), from 2 to 64. We use $K = 1000$ clients, and 25000 samples(applications), i.e., 25 samples per client, on average. Moreover, we fix the number of dimensions, D, to 8, the number of clusters, C, to 4, and the distance between ground truth vectors, HD, to 4.

As we see in Figure 4(c), the distance to ground truth steadily drops from around 40 degrees for 2 starting points examined, to below 20 degrees for 64 starting points examined. In addition, for 64 starting points examined, the variance is also significantly decreased.



(a) 12 samples per client (b) 50 samples per client



(c) 25 samples per client

Figure 4: Distance to the ground truth as we increase the number of local optima we examine, for 12, 25, and 50 samples per client.

In order to confirm that an increased number of starting points always provides a significant improvement, we also tried two different settings using 12 and 50 samples per client; as opposed to the 25 samples per client, in Figure 4(c). As it appears in Figures 4(a) and 4(b), a significant improvement is not always the case, when we increase the number of starting points examined: in Figure 4(a), for 12 samples per client, the distance to ground truth remains between 50 and 60 degrees, while in Figure 4(b), for 50 samples per client, the distance to ground truth is below 10 degrees even for 2 or 4 starting points.

What Figures 4(a), 4(c) and 4(b) suggest is that when we have either a very small number of samples or a sufficiently large number of samples, the increased number of starting points examined does not help us. In other words, the space of solutions for the optimization problem of equations (11)-(13) seems to have optima that differ significantly, only when the number of samples is neither too small nor too large.

4.1.4 *Distance between ground-truth criteria vectors*

Here, we examine how many applications we need for our algorithm to reach a 10-degree distance to the ground truth, as the criteria in the ground truth clusters become more or less "different". Hence, in Figure 5, we plot the applications needed for the 10-degree distance (Y-axis), as the distance, HD, between the criteria vectors of the $C = 2$ clusters, increases from 2 to 16. The number of dimensions, D, is set to 16 and we use $K = 1000$ clients.

For very "different" ground truth criteria vectors that have a hamming distance of $HD = 16$, the algorithm needs fewer than 10 applications per client, to reach the 10-degree distance. Nevertheless, the surprising outcome has to do with the algorithm's behavior when the ground truth criteria vectors are very "close", having a hamming distance of 4 or even 2: the algorithm needs just 15 or 20 applications, respectively, per client, to reach the 10-degree distance. This means that our hard EM approach is able to distinguish between the small difference in the criteria for the two clusters,

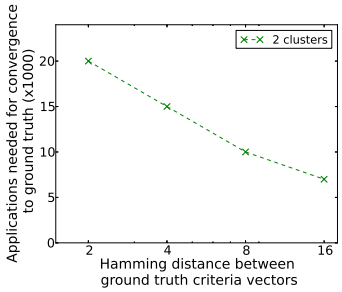
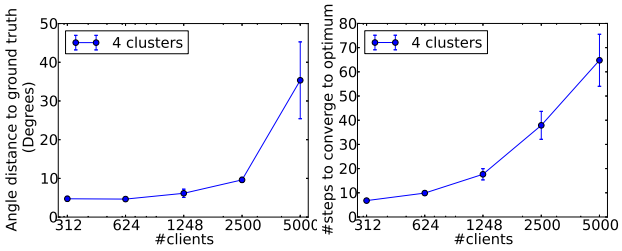


Figure 5: Number of applications needed to get a 10-degree angle distance to ground truth, as we increase the hamming distance between the 2 ground truth vectors from 2 to 16, in 16 dimensions.



(a) Distance to ground truth (b) #iterations

Figure 6: Distance to ground truth and number of iterations needed for convergence, when we increase the number of clients while we keep the number of total samples(applications) constant.

using a relatively small number of samples(applications). (It is important to mention here that the angle distance of the two ground truth vectors with an $HD = 2$, is 41.4 degrees, i.e., much larger than the 10-degree distance threshold we use in our metric.)

4.1.5 *Number of clients*

The experiments we discussed until now had a fixed number of $K = 1000$ clients. The next direction we explore is to keep the total number of applications fixed, and vary the number of clients, K, that decide upon those applications. We use 64000 applications and $D = 8$, $HD = 4$, and $C = 4$.

Figure 6(a) depicts the distance to the ground truth, as K increases from 312 to 5000. For K up to 2500, the distance to the ground truth stays below 10 degrees. However, if we increase K further, to 5000 clients, the distance steeply increases from 10 degrees to 35 degrees. Moreover, there is a significant increase in the variance for $K = 5000$.

The steep increase from $K = 2500$ to $K = 5000$ can be explained by the decrease in the number of samples per client; 12.8 per client on average for 5000 clients. Note that when having a small number of samples per client, it is "difficult" to accurately predict in which cluster a client should be assigned, in the E step. Therefore, while the total number of samples remains constant, the convergence of the algorithm to the ground truth vectors becomes more "difficult", as we keep decreasing the number of samples per client.

The above fact is confirmed by the number of iterations, until converging to an optimum, for the experiments of Figure 6(a). As we see in Figure 6(b), the number of iterations steadily increases as K increases from 312 to 5000.

Feature	Description	Tiers
Contractor's Score	A score summarizing how good the contractor is, based on the feedback received from clients that worked with the contractor.	1, 2, 3, 4
Matched Skills	Jaccard similarity between the set of skills the contractor has in her profile and the set of skills required for the task.	1, 2, 3, 4
Contractor's Total Hours	The aggregated number of hours for the contracts the contractor had in the platform.	1, 2, 3, 4
Bid Amount	The fixed amount of money the contractor asks for completing the task she applies for.	1, 2, 3, 4
Bid Rate	The hourly rate earnings the contractor asks for completing the task she applies for.	1, 2, 3, 4
Independent Contractor	Indicates if the contractor is a company or agency that runs a profile in the platform.	2, 3, 4
Contractor's Assignments	The number of tasks the contractor is assigned to and haven't been completed yet.	2, 3, 4
Contractor's Total Revenue	The aggregated earnings of the contractor in the platform.	2, 3, 4
Contractor's Tests Passed	The number of tests that the contractor has taken and successfully completed in the platform.	2, 3, 4
Contractor's Bill Rate	The hourly rate the contractor charges for the tasks he gets. Contractors usually set their bill rate in their profile to indicate the "quality" of their work.	2, 3, 4
Contractor's Experience	A score summarizing the working experience of the contractor based on her resume.	3, 4
Contractor's Portfolio Items	The number of items the contractor has in her portfolio.	3, 4
Matched Cost	The distance between the asking price and the price the client indicates she is willing to pay for the task.	3, 4
Matched Region	Indicates if the contractor's geographical region matches one of the regions the client prefers.	3, 4
Matched Task Category	Indicates if the task category is one of the categories the contractor has specified in her profile, as her areas of expertise.	3, 4
Contractor's Recent Activity	A score summarizing the contractor's recent activity, e.g., tasks recently completed or the date she completed the last task.	4
Contractor's English Skills	A score summarizing how fluently the contractor speaks english.	4
Contractor's Profile Information	A score summarizing how detailed the contractor's profile information is, e.g., full name appearing or not.	4
Contractor's Active Interviews	The number of pending applications the contractor has made, where the client has responded back to him; and are thus considered as the active interviews.	4
Contractor's Rank Percentile	For each test a contractor passes she receives a score. This feature gives in which percentile the contractor belongs, based on her test scores.	4

Table 1: Features representing each application

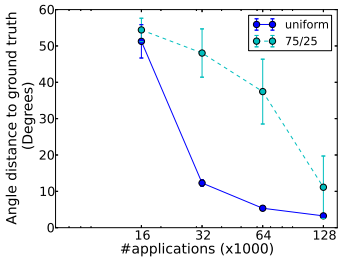
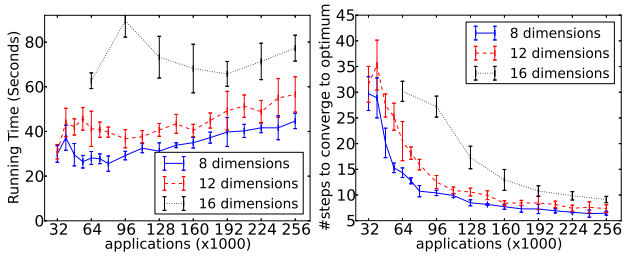


Figure 7: Angle distance to ground truth, as we increase the number of applications from 16 to 128 applications per client, for a uniform and a 75/25 distribution of clients to clusters.



(a) Running Time (b) #iterations

Figure 8: Running time and number of iterations needed for convergence, when we increase the number of samples(applications) and dimensions.

4.1.6 *Distribution of applications to clients*

The final direction we examine in the synthetic data experiments, has to do with the distribution of clients to clusters. Instead of uniformly distributing the clients to clusters, we assign 75% of the clients to the 25% of the clusters and the rest 25% of the clients to the 75% of the clusters, in the ground truth. In particular, we use $C = 4$ clusters and $K = 1008$ clients. Hence, the first cluster gets assigned 756 clients, while each of the other 3 clusters gets assigned 84 clients.

Figure 7 depicts, the angle distance between our algorithm's outcome and the ground truth (Y-axis), as we increase the number of applications (X-axis). We keep the same parameters with the ones used in Section 4.1.1 for Figure 2, and we plot two curves: one for the 75/25 distribution and one for the uniform distribution(same curve as in Figure 2).

For 16000 applications, the difference between the uniform and the 75/25 distribution is negligible. Nonetheless, this is not the case for 32000 and 64000 applications: the difference between the uniform and the 75/25 distribution is more than 30 degrees. Once we further increase the number of applications from 64000 to 128000 applications, the angle distance to the ground truth for the 75/25 distribution, drops to 10 degrees and reaches close to the ground truth distance for the uniform distribution.

The difference between the curves can be explained by the number of samples(applications) that get assigned to each cluster. For the 75% of the clusters that get just the 25% of the clients, the number of applications per cluster is not enough to converge to the ground truth criteria, when the total number of applications is not more than 64000. The same observation also applies to the uniform distribution, for 16000 applications. However, when we increase the number of applications to 32000, the uniform distribution achieves a ground truth distance that is almost 10 degrees. On the other hand, in order for the 75/25 distribution to reach to a 10-degree distance, we need a lot more applications (128000).

4.1.7 *Running Time*

In this section, we study the scalability of our algorithm as we increase the number of samples and dimensions. In Figure 8(a), we plot the running time of our algorithm (Y-axis) on a 1.8 GHz Intel Core i5 processor with 4 GB of RAM. The algorithm is implemented in Python using scikit-learn, and the local optima are explored in parallel. We use

approved	*clients*	*training*	*testing*
10 apps	2281	760k apps	105k apps
20 apps	480	260k apps	32k apps
30 apps	172	120k apps	22k apps
40 apps	84	80k apps	13k apps
50 apps	48	50k apps	12k apps

Table 2: Datasets used in experiments

the same *non-sparse* setting with the one used in Figure 3(a).

The computational overhead for solving the sparse logistic regression problems increases as the number of samples increases (X-axis). Surprisingly, though, the overall time of our algorithm is not monotonically increasing: up to a point it is constant or decreasing and then slowly increases. Moreover, that point is different for each number of dimensions.

The outcome of Figure 8(a) is due to the trade-off between the computational overhead for solving the logistic regression problems and the number of iterations needed to converge to an optimum. As the number of samples increases there is more computational overhead, however, the number of iterations decreases as Figure 8(b) shows, causing the overall running time to decrease or remain constant. In addition, the point where the number of iterations essentially stops decreasing is different for 8, 12, and 16 dimensions: the more the dimensions, the more the samples needed for the number of iterations to stop decreasing. This explains why the overall time starts to slowly increase in different points for each number of dimensions, in Figure 8(a).

4.2 oDesk Dataset

We start with a brief overview of the dataset, the features, and the metric, and then we discuss our findings.

Training and Testing Datasets: We collected all the applications of contractors to tasks opened by clients in the oDesk platform, from September 1st of 2012 to December 15 of 2012. In addition, we filtered this set of applications in order to keep only the applications rejected/approved by clients who had a "sufficient" number of approved applications within this period. For example, consider a threshold of 10 approved applications, a client A that had 9 approved applications within that period, and a client B with 11 approved applications: all the applications for tasks opened by A are excluded from the training set, while all applications for tasks opened by B are included in the training set.

In particular, we generated 5 training sets using 5 different thresholds: 10, 20, 30, 40, and 50 approved applications. Note that each of these sets is a subset of the previous one. Table 2 summarizes the details for the 5 training sets: column *approved* refers to the threshold of approved applications, column *clients* refers to the number of clients satisfying the respective threshold constraint, and column *training* refers to the number of rejected/approved applications by those clients, i.e., the actual samples in the training set.

Furthermore, we generated the 5 testing sets corresponding to the 5 training sets: for each training set we kept the clients that satisfy the respective threshold constraint, and we collected the rejected/approved applications for their tasks from January 15 of 2013 to March 1st of 2013 (column *testing* in Table 2).

Features: We ran experiments with 4 different tiers of features: tier 1 includes 5 features, tier 2 includes the 5 features of tier 1 plus 5 more features and so on. Table 1 gives the names and descriptions of the features, along with the tiers that include each feature. Note that each application in our training/testing sets is represented by the values for those features the moment the application is made. We normalize all the feature values to $[0.0, 1.0]$, taking into account what is the maximum value the feature may have. For example, for the contractor's score that has a maximum of 5.0 stars, the score in each application is divided by 5.0.

Metric: We use the area-under-the-curve ratio(AUC ratio) to quantify the improvement in the prediction accuracy when using different criteria in different clusters as opposed to using the same criteria for the whole dataset, i.e., the testing dataset used in each case. Let us describe the metric using an example. Consider running our algorithm with 2 clusters as input. For each cluster i our algorithm computes, we use the criteria $\mathbf{w_i}$ learned for this cluster, and we compute the area under the ROC curve: assume an $AUC_1 = 0.8$ for cluster 1 and an $AUC_2 = 0.9$ for cluster 2. Note that in order to compute AUC_i, we use the testing-set applications of the clients that were placed in cluster i. In addition, we run our algorithm using a single cluster as input (i.e., we run the classic sparse logistic regression model) and we compute the criteria $\mathbf{w_s}$ for a single cluster. For each of the 2 clusters our algorithm produced, we use $\mathbf{w_s}$ to compute the area under the ROC curve for the testing-set applications that belong to that cluster: assume an $AUC'_1 = 0.8$ for cluster 1 and an $AUC'_2 = 0.5$ for cluster 2. Then, we compute the ratio $\frac{AUC}{AUC'}$ for each of the 2 clusters and the weighted average of the two ratios. For example, assume that in the testing set of cluster 1 there are 200 approved applications and in the testing set of cluster 2 there are 800 approved applications. The combined AUC ratio will be: $\frac{200}{1000} * \frac{AUC_1}{AUC'_1} + \frac{800}{1000} * \frac{AUC_2}{AUC'_2} = 0.2 * 1.0 + 0.8 * 1.8 = 1.64$, indicating an improvement of 64% over the baseline.

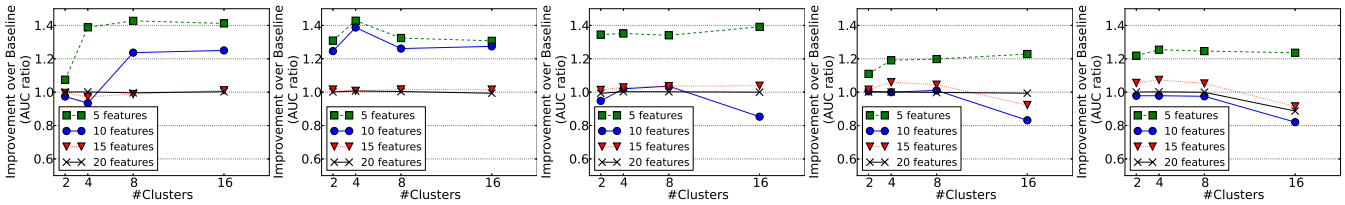
It is important to note that the AUC can not be greater than 1.0 and will not be less than 0.5; assuming a prediction accuracy not worse than random. Hence, the AUC ratio will always be less than 2.0, or in other words, the improvement over the baseline can not be more than 100%.

4.2.1 Improvement over Baseline

Figure 9 depicts the AUC ratio(Y-axis) as we increase the number of clusters(X-axis). Each curve refers to a different feature tier (see Table 1): *a)*5 features in tier 1, *b)*10 features in tier 2, *c)*15 features in tier 3, and *d)*20 features in tier 4. In Figure 9(a), we used the training/testing set for an *approved* threshold of 10, in Figure 9(b), we used the training/testing set for an *approved* threshold of 20, and so on.

There are three main factors that can explain the results in Figure 9. First, as we increase the number of dimensions(features), the improvement over the baseline(same criteria for all clients) drops: note that in most cases the curves for fewer features stay above the curves for more features. This factor is aligned with our observations in Section 4.1.2, i.e., as the number of features increase, our algorithm needs more samples and at some point the number of samples in the respective training set is not sufficient to improve over the prediction accuracy of the baseline approach.

Second, if we increase the number of clusters beyond a certain point, there is no improvement over the baseline (in fact, the baseline gives a higher AUC): in Figure 9(c) for 16 clusters, the curve for 10 features falls below 1.0, in Figure 9(d) for 16 clusters, both curves for 10 and 15 features fall below 1.0, and in Figure 9(e) for 16 clusters, the curves for 10, 15, and 20 features fall below 1.0. This observation is aligned with the discussions for Figures 2 and 3(b).



(a) *approved* = 10 (b) *approved* = 20 (c) *approved* = 30 (d) *approved* = 40 (e) *approved* = 50

Figure 9: AUC ratio(Y-axis) for different numbers of clusters(X-axis). Each curve refers to a different feature tier (Table 1) and each plot to a different pair of training and testing sets (Table 2).

Figure 11: Weights of the criteria vector when a single cluster is used (traditional logistic regression).

Third, if we reduce the number of samples(applications) beyond a certain point, the improvement over the baseline diminishes: the curve for 5 features drops from a 30%-40% improvement over the baseline in Figures 9(a)-9(c) to a 20% improvement in Figures 9(d)-9(e), while the curve for 10 features does not show any improvement over the baseline (or even drops below 1.0) in Figures 9(c)-9(e). This observation is aligned with the discussion in Section 4.1.1.

Nevertheless, the three factors discussed above, fail to explain why in some cases where the number of samples decreases, the improvement over the baseline increases. For example, from Figure 9(a) to Figure 9(b), there is a large increase in the curves for 5 and 10 features, for 2 and 4 clusters. (Or, from Figure 9(b) to Figure 9(c), the improvement for 5 features increases from 30% to almost 40%, for 16 clusters.) This behavior is the outcome of an important trade-off: by increasing the *approved* threshold, the number of samples in the training set drops from 760000 to 50000, from Figure 9(a) to Figure 9(e), however, the number of applications for each client increases. Hence, we have more samples for each client in order to accurately "learn" her criteria and place her in the "right" cluster. (The positive effect of having more samples per client was also discussed in Section 4.1.5.)

4.2.2 Clients Criteria

Let us now focus on one of the most interesting outcomes of our algorithm: The 4 clusters produced for 10 features and an *approved* threshold of 20, an outcome achieving a substantial 40% improvement over the baseline, in Figure 9(b). (As we discussed in the description of the metric, the improvement can not be greater than 100%.)

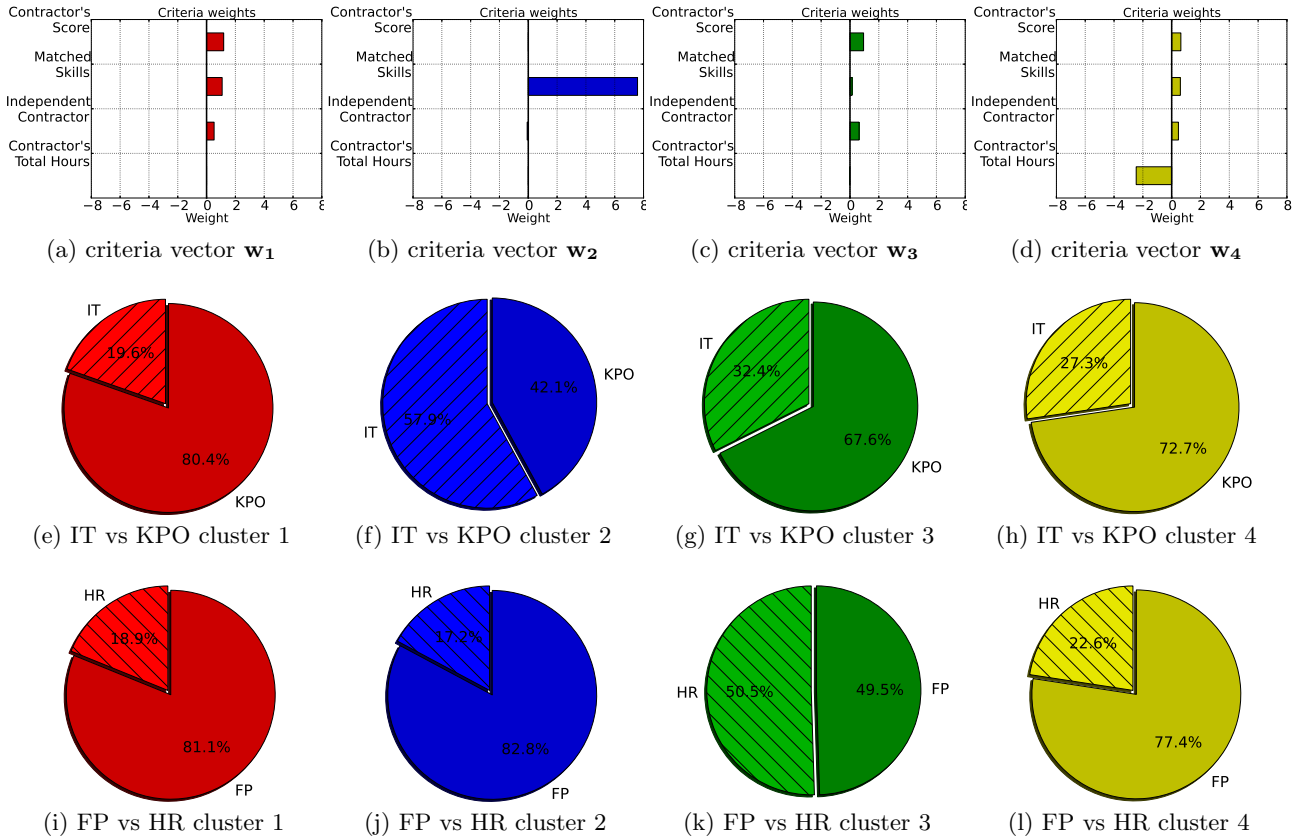
Figures 10(a) to 10(d) depict the values of the criteria vectors, $\mathbf{w_1}$ to $\mathbf{w_4}$, for the 4 clusters. Specifically, the figures show the 4, out of 10, criteria for which at least one of the vectors had a non-zero value. As these figures indicate, there are important differences in the criteria of clients for: *a)* clusters 1 and 3 (Figures 10(a) and 10(c)), *b)* cluster 2 (Figure 10(b)), and *c)* cluster 4 (Figure 10(d)).

Clients in cluster 2 have a strong tendency in selecting contractors whose skills match very well with the task they post (large positive weight for the "Matched Skills" feature), clients in cluster 4 tend to reject contractors that have completed a lot of hours working in the platform (large negative weight for "Contractor's Total Hours"), and clients in clusters 1 and 3 decide based on the contractor's score, skills, and agency independence (see Table 1 for a full description of the features).

Before discussing further the differences between client criteria in different clusters, let us compare these criteria with those learnt when using a single cluster, i.e., when learning the same criteria for all clients. Figure 11 depicts the non-zero values of the criteria vector when a single cluster is used, i.e., when we run the traditional logistic regression. There are two main inconsistencies in those values: 1) they do not capture the fact that only a specific group of clients (cluster 4) has a negative bias towards contractors with a lot of hours completed in the platform, and 2) they miss the strong positive bias of a group of clients (cluster 2) towards contractors with the appropriate skills for a task. These two inconsistencies explain why our algorithm gives this substantial 40% improvement (Figure 9(b)) over the baseline approach of learning the same criteria for all clients.

Some of the differences in the clients' criteria can be explained by observing the IT/KPO and FP/HR percentages of the applications in the different clusters. IT stands for Information Technology and refers to applications for all tasks related to software development and administration, e.g., web development, DB administration, or software for scientific experiments. Applications for all other tasks fall under the Knowledge Processing Outsourcing (KPO) category, e.g., translation, marketing, or logo design. A second categorization for tasks is the Fixed Price(FP)/Hourly Rate(HR) categorization that refers to whether a fixed amount will be given to the contractor upon the completion of the task or the client and the contractor will get into a contract with predefined hourly-rate earnings. Figures 10(e) to 10(h) depict the IT/KPO percentages, while Figures 10(i) to 10(l) depict FP/HR percentages, in the 4 clusters.

In cluster 2 there is a large percentage of IT tasks (57.9%), as we see in Figure 10(f). This large percentage provides an explanation for the criteria in cluster 2. Most IT contractors receive good reviews by clients; although clients are not always fully satisfied with the contractor's work (e.g., software consistent with the specifications that does not do exactly what the client had in mind). As a result, IT contractors tend to have a very high score and, hence, the contractor's score stops being a powerful signal for a client to base her decision. On the other hand, the technologies where the contractor is an expert, in many cases indicate if a contractor is a good fit for an IT task. Therefore, the clients in cluster 2 have a good reason to base their decisions on the "Matched



(a) criteria vector $\mathbf{w_1}$ (b) criteria vector $\mathbf{w_2}$ (c) criteria vector $\mathbf{w_3}$ (d) criteria vector $\mathbf{w_4}$

(e) IT vs KPO cluster 1 (f) IT vs KPO cluster 2 (g) IT vs KPO cluster 3 (h) IT vs KPO cluster 4

(i) FP vs HR cluster 1 (j) FP vs HR cluster 2 (k) FP vs HR cluster 3 (l) FP vs HR cluster 4

Figure 10: Figures (a) to (d) depict the weights of the criteria vectors in each of the 4 clusters computed for 10 features and an *approved* threshold of 20. Figures (e) to (h) depict the percentage of Information Technology tasks in each cluster, while figures (i) to (l) depict the percentage of Hourly Rate tasks.

Skills" (Figure 10(b)).

Although the criteria in clusters 1 and 3 (Figures 10(a) and 10(c)) seem similar, there is an interesting difference: clients in cluster 1 give a large weight to the "Matched Skills" while the value for this feature in cluster 3 is almost zero. As Figures 10(i) and 10(k) point out, there is an important difference in the percentage of the HR tasks in the two clusters: in cluster 1 only 18.9% of the tasks are HR, while in cluster 3 50.5% of the tasks are HR. The "Matched Skills" signal is usually more useful for FP tasks that require expertise in a very specific piece of work that needs to be completed, while HR tasks mostly require a good collaboration between the client and the contractor.

As for the large negative weight on the "Contractor's Total Hours" for cluster 4 (Figure 10(d)), it appears that the IT/KPO or the HR/FP percentages can not provide any plausible explanation. Cluster 4 simply consists of clients that prefer to work with contractors that are "new" to the platform. In fact, there is a good reason for selecting such contractors: in many cases "new" contractors are eager to produce a high-quality outcome in order to build a good reputation in the system.

5. RELATED WORK

A problem similar to the one we study is addressed by collaborative filtering (CF) approaches (see [21] for a recent survey). Traditional memory-based CF approaches estimate the preference of a user U to an item I using the ratings of U for items "similar" to I or the ratings of users "similar" to U for I [5, 19]. Other CF approaches are based on latent factor models such as matrix factorization [17, 22], Latent Dirichlet allocation [2], Boltzmann machines [18], or Latent Semantic Analysis [6], while others [12] have proposed models that combine the memory and model based approaches.

The hetereogeneity of user preferences is also studied by Lenk et al [14]. The authors claim that the user preferences depend on their spatial region, therefore, they partition the users into groups based on their locations and then "learn" the user preferences in each group. User partitioning can also be based on whether a user is an "innovator" or "imitator" [7]: innovators make purchase decisions based on their own preferences, while imitators decide based on a product's stage of maturity.

The main limitation for directly applying CF methods to the problem discussed in this paper, is the fact that contractor applications to a task posted by a client are "unique". That is, while most CF methods assume that users rate, explicitly or implicitly, the same products, in our case, clients do not actually rate the same contractors but their applications to a specific task. For example, the same contractor maybe an expert for one task but not really appropriate for another task, or she may not be willing to provide the full functionality asked by a client in one task for a given price.

The effect of clients having different preferences when selecting among alternatives, is studied thoroughly in the economics and marketing literature [9, 16, 13, 1]. In particular, most of those studies focus on how a market is partitioned in segments, with each segment having clients with homo-

geneous preferences when selecting among brands. Most related to our work, are the studies using Finite Mixture Logit models [15, 9, 16], where the number of segments is finite and the preferences of clients in each segment are given by a logit model. An important difference of those studies with our work is that we don't focus on "learning" the distribution (or the parameters controlling the distribution) of market segments. On the contrary, we focus on partitioning the existing clients, with each client belonging to exactly one segment; instead of belonging to any segment with some probability (see [10] for an analysis of the tradeoffs between "hard" and "soft" assignments). Moreover, as noted earlier in the discussion for CF, the preferences of clients refer to "unique" applications in our model, as opposed to a fixed set of alternatives, e.g., brands. Our model is simpler than the Finite Mixture Logit models used in economics and marketing studies, and aims for a scalable solution to a problem met in most of the online work marketplaces.

6. CONCLUSION

We studied the problem of clustering clients of work marketplaces based on their criteria when hiring contractors. We formally defined the problem using a Maximum Likelihood formulation and we developed a simple Expectation-Maximization algorithm that can be applied effectively on large datasets. Our experimental results show that our approach can significantly improve the prediction accuracy for future hirings of clients, compared to the baseline approach that assumes all clients having the same hiring criteria.

The application of our approach to a real-world dataset provided by oDesk revealed some interesting facts about the way different groups of clients choose contractors for their tasks: some clients are positively biased to contractors that are "new" to a marketplace (probably because many new contractors are eager to build a competitive profile in the marketplace), while other clients ignore the contractor's reputation and focus on how good of a fit a contractor is to the job posted. Our approach learns such differences in client hiring criteria and can drastically improve the matching between clients and contractors in work marketplaces.

7. REFERENCES

[1] G. M. Allenby and P. E. Rossi. Marketing models of consumer heterogeneity. *J. Econometrics*, 89:57–78, 1998.

[2] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *J. Machine Learning Research*, 3:993–1022, 2003.

[3] A. S. Das, M. Datar, A. Garg, and S. Rajaram. Google news personalization: Scalable online collaborative filtering. In *WWW*, 2007.

[4] W. DeSarbo and W. Cron. A maximum likelihood methodology for clusterwise linear regression. *J. of Classification*, 5(2):249–282, 1988.

[5] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *SIGIR*, 1999.

[6] T. Hofmann. Latent semantic models for collaborative filtering. *ACM Trans. Information Systems*, 22:89–115, 2004.

[7] K. Hu, W. Hsu, and M. L. Lee. Utilizing users' tipping points in e-commerce recommender systems. In *ICDE*, 2013.

[8] S. in Lee, H. Lee, P. Abbeel, and A. Y. Ng. Efficient l1 regularized logistic regression. In *AAAI*, 2006.

[9] W. A. Kamakura and G. Russell. A probabilistic choice model for market segmentation and elasticity structure. *J. Marketing Research*, XXVI:379–390, 1989.

[10] M. J. Kearns, Y. Mansour, and A. Y. Ng. An information-theoretic analysis of hard and soft assignment methods for clustering. In *UAI*, 1997.

[11] K. Koh, S.-J. Kim, and S. Boyd. An interior-point method for large-scale l1-regularized logistic regression. *J. Machine Learning Research*, 8:1519–1555, 2007.

[12] Y. Koren. Factorization meets the neighborhood: A multifaceted collaborative filtering model. In *KDD*, 2008.

[13] P. J. Lenk, W. S. DeSarbo, P. E. Green, and M. R. Young. Hierarchical bayes conjoint analysis: Recovery of partworth heterogeneity from reduced experimental designs. *Marketing Science*, 15:173–191, 1996.

[14] J. J. Levandoski, M. Sarwat, A. Eldawy, and M. F. Mokbel. Lars: A location-aware recommender system. In *ICDE*, 2012.

[15] G. Mclachlan and D. Peel. *Finite Mixture Models*. Wiley, 2000.

[16] D. C. J. Pradeep K. Chintagunta and N. J. Vilcassim. Investigating heterogeneity in brand preferences in logit models for panel data. *J. Marketing Research*, 28:417–428, 1991.

[17] R. Salakhutdinov and A. Mnih. Probabilistic matrix factorization. In *NIPS*, 2007.

[18] R. Salakhutdinov, A. Mnih, and G. Hinton. Restricted boltzmann machines for collaborative filtering. In *ICML*, 2007.

[19] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *WWW*, 2001.

[20] H. Späth. Algorithm 39 clusterwise linear regression. *Computing*, 22(4), 1979.

[21] X. Su and T. M. Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in Artificial Intelligence*, 2009.

[22] J. Yuan, B. Kanagal, V. Josifovski, L. Garcia-Pueyo, A. Ahmed, and S. Pandey. Focused matrix factorization for audience selection in display advertising. In *ICDE*, 2013.