Pareto Optimal Streaming Unsupervised Classification

Soumya Basu [1] **Steven Gutstein** [2] **Brent Lance** [2] **Sanjay Shakkottai** [1]

Abstract

We study an online and streaming unsupervised classification system. Our setting consists of a collection of classifiers (with unknown confusion matrices) each of which can classify one sample per unit time, and which are accessed by a stream of unlabeled samples. Each sample is dispatched to one or more classifiers, and depending on the labels collected from these classifiers, may be sent to other classifiers to collect additional labels. The labels are continually aggregated. Once the aggregated label has high enough accuracy (pre-specified threshold for accuracy) or the sample is sent to all the classifiers, the now labeled sample is ejected from the system. For any given pre-specified accuracy threshold, the objective is to sustain the maximum possible sample arrival rate, such that the number of samples in memory does not grow unbounded. In this paper, we characterize the Pareto-optimal region of accuracy and arrival rate, and develop an algorithm that can operate at any point within this region. Our algorithm uses queueing-based routing and scheduling approaches combined with a novel online tensor decomposition method to learn the hidden parameters, to Pareto-optimality guarantees. We finally verify our theoretical results through simulations on two ensembles formed using AlexNet, VGG, and ResNet deep image classifiers.

1. Introduction

Computational outsourcing, via the cloud, is becoming increasingly important. However, the efficacy of cloud computing agents often remains unknown (e.g. task processing on Mechanical Turk, where the human agents have unknown efficiencies). Furthermore, ground truth data needed to evaluate the agents' performance is often absent.

Such problems must be addressed through an ensemble of classifiers, operating in an unsupervised setting (i.e. Unsupervised Ensemble Learning—UEL). Over the past three decades, significant theoretical and empirical research has been performed in this area (Dawid & Skene, 1979; Li et al., 2013; Zhang et al., 2014; Jain & Oh, 2014; Jaffe et al., 2016; Shaham et al., 2016). In particular, crowdsourcing, which focuses on selecting and aggregating a large number of classifiers, has an extensive literature (Karger et al., 2011; 2014; Ok et al., 2016) culminating in optimal aggregation schemes. The key focuses of most previous works are 1) from a crowdsourcing perspective, choosing a small subset of an effectively infinite pool of classifiers for each sample, and 2) from an unsupervised learning perspective, offline/one-shot classification of a set of samples (see Section 7 for related work).

We take a somewhat different approach than the traditional crowdsourcing and UEL. Our system consists of a *fixed* ensemble of deterministic classifiers, each of which has a fixed processing speed. Such classifiers are available in many emerging fields such as cloud computation or human-in-the-loop computation. As a motivating example, consider an ensemble of both human experts and third party neural network classifiers, which are jointly tasked to label images in real time. The individual labeling characteristics of these classifiers are often unknown; for human agents because the agents have not been evaluated sufficiently in the past, and for DNNs, due to contractual or privacy reasons (e.g. the neural network was downloaded from the web where it was previously trained on an unknown data set). Importantly, these classifiers are deterministic when the DNNs have fixed weights and the humans have fixed opinions (i.e. repeatedly providing the same sample to an agent will not result in changing labels).

In our *streaming* model, samples (e.g. images) arrive online and are assigned to an ensemble of expert classifiers. Each classifier is characterized by a latent confusion matrix and can process one sample per time slot. For each sample, the labels are continually collected and aggregated as it sequentially passes through distinct classifiers. During the process of aggregation once the sample achieves a pre-

[1]The University of Texas at Austin, USA [2]Army Research Lab, USA. Correspondence to: Soumya Basu <basusoumya@utexas.edu>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

specified desired accuracy, it gets ejected from the system. Furthermore, in our fixed ensemble, repeated access for a specific classifier results in the same output. Thus, when classification by all the classifiers gets completed, a sample is removed even if desired accuracy is not achieved.

Two key features in our system are: *(i)* the sequential selection of classifiers depends on the current set of collected labels, and *(ii)* the decision to either eject a sample or continue to acquire additional labels also depends on the current set of collected labels. Sustaining a high arrival rate requires that the average number of classifiers used to label a sample be small, because each classifier has limited processing speed. Yet, obtaining a high threshold accuracy requires that the average number of classifiers used to label a sample be large, because larger ensembles are more accurate. This is a fundamental trade-off in our system.

1.1. Our Contribution

Our main contributions are as follows.

1. We define and characterize the notion of a Pareto region for sample arrival rate and threshold accuracy in the streaming, online Dawid-Skene model. We show that the posterior class distribution for a set of samples under any *history dependent* algorithm is independent of that history, given the partial label collected by the samples (Lemma 3.4). Using this conditional independence, we characterize the Pareto region as a feasible network flow (Theorem 3.6).

2. We design a scheduling algorithm, which is inspired by max-weight scheduling in stochastic networks, but uses a threshold based departure and maximum likelihood labeling. We show it achieves any point in the interior of the Pareto region (Theorem 4.1). The designed algorithm does not assume knowledge of arrival rate or classifier statistics. Using an explore-exploit strategy we learn these parameters online, with a spectral estimation technique.

3. Off-the-shelf unsupervised learning techniques are computationally prohibitive for the online learning of hidden parameters. We develop a novel online Tensor decomposition algorithm that reuses eigenvectors from the previous round. We show such reuse along with the application of the stopping criteria in (Anandkumar et al., 2015) decreases the computational complexity (Lemma 5.2).

4. Finally, we validate our theoretical results using ensembles comprised of Alexnet (Krizhevsky et al., 2012), VGG (Simonyan & Zisserman, 2014) and Resnet (He et al., 2016) deep convolutional neural nets. We perform experiments on modified Cifar-10 datasets. Various classes were merged into "super-classes", so that we work with datasets containing only 3 or 5 classes. Our results highlight the trade-off between threshold accuracy and arrival rate.

2. System Model

We consider a time-slotted online unsupervised classification problem where in each time-slot (a.k.a. round) samples (e.g. images) arrive into the system for classification. In the remaining section, we first define the description and generation of the samples. We next provide the system's description and dynamics.

2.1. Notation

We define the set $[N] = \{0, 1, \ldots, N-1\}$ for all integer $N \geq 0$. We further include the null element, denoted as $\mathbf{e}$, in this set to define $[N]_e = \{\mathbf{e}\} \cup [N]$. An ordered array is denoted as $\{a_i : i \in [N]\}$ and a ordered matrix is denoted as $\{A_{ij} : i \in [N], j \in [N']\}$. More generally an array can be indexed by an ordered set I as $\{a_i : i \in I\}$.

2.2. Online Dawid-Skene Model

We adopt a model which is the online analogue of the generalized Dawid-Skene model (Dawid & Skene, 1979) for unsupervised classification. We call our model "Online Dawid-Skene" (OnDS) model. In our model, there are K classes and M deterministic classifiers, with $M > 1$. Each classifier $i \in [M]$ has a *confusion matrix*, $\mathbf{C}_i := \{C_i(k,l) : k, l \in [K]\}$. We assume each classifier can process *one sample per timeslot*, which can be easily generalized to non-uniform processing speeds.[1]

The samples arrive online and are represented by their unique ids. In our deterministic classification, the true label and individual classifiers' labels for each sample are generated once and fixed thereafter. Formally, the sample with id j has a *latent-tuple* $\mathbf{s}_j \in [K]^{(M+1)}$. $s_j(i)$ denotes the deterministic label received from classifier i, for all $i \in [M]$, and $s_j(M)$ denotes the true label of the sample.

• E.g., with two classifiers ($M = 2$) and two classes ($K = 2$), if sample 0 contains the latent tuple $\mathbf{s}_0 = (0, 1, 1)$, then for sample 0, (i) 0-th classifier's label $= 0$, (ii) 1^{st} classifier's label $= 1$, and (iii) true label $= 1$.

The *generation process* of $\mathbf{s}_j$, for each j, is as follows:
1) For each sample an independently and identically distributed (i.i.d.) true-label $s(M)$ is first chosen from the K classes following a *probability vector* $\mathbf{p}_g$. Thus, for all j, $\mathbb{P}[s_j(M) = k] := p_g(k)$ for $k \in [K]$.
2) Next, for all $i \in [M]$, $s_j(i)$ is chosen according to the following conditional probability; for all $k, l \in [K]$ $\mathbb{P}[s_j(i) = l | s_j(M) = k] := C_i(k, l)$.

[1] For each $i \in [M]$, if classifier i labels at most μ_i samples/round, for any finite integer $\mu_i > 1$, we can replace the i-th classifier with μ_i separate classifiers each with confusion matrix $\mathbf{C}_i$ and one sample per round processing speed.

2.3. Memory

In our time-slotted system there exists a common memory where past samples reside. At the beginning of round $\tau \geq 1$, the memory contains a set of samples, denoted as S^τ. At each time τ, a sample j in memory is associated with a *label-tuple* $\hat{\mathbf{s}}_j^\tau \in [K]_e^M$, which changes with time. Specifically, for all classifiers $i \in [M]$ and rounds $\tau \geq 1$, at the beginning of round τ:
(i) if the sample is labeled by classifier i then $\hat{s}_j^\tau(i) = s_j(i)$,
(ii) otherwise, $\hat{s}_j^\tau(i) = \mathbf{e}$.

A sample j is *partially labeled*, if any of its coordinates are unobserved (i.e. for some i, $\hat{s}_j^\tau(i) = \mathbf{e}$). Otherwise, it is *completely labeled*. For each label-tuple $\ell \in [K]_e^M$, the number of samples with that label-tuple at the beginning of round τ is denoted as $Q_\ell(\tau) = |j \in S^\tau : \hat{s}_j^\tau = \ell|$.

• E.g. with $M = 2$ and $K = 2$, let sample 0 ($\mathbf{s}_0 = (0, 1, 1)$) be labeled by classifier 0 in round $\tau = 1$, and classifier 1 in round $\tau = 3$. Then we have $\hat{\mathbf{s}}_0^1 = (\mathbf{e}, \mathbf{e})$, $\hat{\mathbf{s}}_0^2 = (0, \mathbf{e})$, $\hat{\mathbf{s}}_0^3 = (0, \mathbf{e})$, and $\hat{\mathbf{s}}_0^\tau = (0, 1)$ for all $\tau \geq 4$.

2.4. System Dynamics

Each round has four sequential *stages*: 1) scheduling, 2) labeling, 3) departure, and 4) arrival.

Scheduling: Initially, there are samples in memory which are matched with the classifiers. The system schedules a *matching* in the bipartite graph $\mathcal{G}^\tau$ with the two partites: S^τ (the samples) and $[M]$ (the classifiers). For each sample $j \in S^\tau$ and classifier $i \in [M]$, there is an edge (j, i) if and only if $\hat{s}_j^\tau(i) = \mathbf{e}$. Let $\mathfrak{A}^\tau$ denote the set of feasible matchings, and A^τ be the scheduled matching in round τ. If classifier i is matched with sample j then $A^\tau(i) = j$. Otherwise $A^\tau(i) = \mathbf{e}$.

Labeling: The labels obtained from classifiers are denoted as $Cl^\tau \in [K]_e^M$. If classifier i is not matched with any sample then $Cl^\tau(i) = \mathbf{e}$. Otherwise, for sample $j = A^\tau(i)$ the latent label $s_j(i)$ is revealed, i.e. $Cl^\tau(i) = s_j(i)$. Therefore, for the sample j the *observed-tuple* changes to $\hat{\mathbf{s}}_j^{(\tau+1)}$. So, $\hat{s}_j^{(\tau+1)}[i] = s_j[i]$, and $\hat{s}_j^{(\tau+1)}[i'] = \hat{s}_j^\tau[i']$ for all $i' \neq i \in [M]$. For all unmatched samples their respective *observed-tuples* remain unchanged.

Departure: Next, each sample either obtains a *final label* and exits the system, or reenters the memory. The set of departing samples is denoted as D^τ. Each sample $j \in D^\tau$ has an associated final label k_j^*. The set of partially labeled departing samples is denoted as $D_p^\tau \subseteq D^\tau$.

Arrival: Finally, a random number of new samples, N^τ, (generated following OnDS model) arrive into the system. Here N^τ is i.i.d., for each τ, with finite mean $\lambda_{ex} = \mathbb{E}[N^\tau]$ and bounded support $\sup_{\tau \geq 1} N^\tau \leq \lambda_{\max}$. Let the set of arriving samples be N_{ex}^τ. The sample ids are ordered according to their arrival.

Update: In $(\tau + 1)$-th time-slot the set of samples in storage changes to $S^{(\tau+1)}$, where $S^{(\tau+1)} = S^\tau \setminus D^\tau \cup \mathcal{N}_{ex}^\tau$. The number of samples that changes into the label-tuple ℓ from another and re-enters the system in round τ is denoted as $N_\ell(\tau)$. Further, the number of samples that changes from one label-tuple, ℓ, to another in round τ is denoted as $R_\ell(\tau)$. Formally, for all $\ell \in [K]_e^M$,

$$N_\ell(\tau) = |j : j \in S^{(\tau+1)}, \hat{s}_j^{(\tau+1)} = \ell, \hat{s}_j^\tau \neq \ell|,$$
$$R_\ell(\tau) = |j : j \in S^\tau, \hat{s}_j^{(\tau+1)} \neq \ell, \hat{s}_j^\tau = \ell|.$$

At time τ, label-tuple $\ell \in [K]_e^M$,

$$Q_\ell(\tau + 1) = (Q_\ell(\tau) - R_\ell(\tau))^+ + N_\ell(\tau).$$

3. Pareto Region of OnUEL

3.1. Preliminaries and Definitions

System History: The history of the system is the set of all the events that happened in the past. Formally, at various stages of round $\tau \geq 1$ the histories can be defined recursively as follows: Let the initial history of the system be H_0^1 and H_0^τ be history in the beginning of round τ.
(i) After scheduling history is $H_1^\tau := H_0^\tau \cap A^\tau$.
(ii) After labeling history is $H_2^\tau := H_1^\tau \cap Cl^\tau$.
(iii) After departure it is $H_3^\tau := H_2^\tau \cap \{D^\tau, \{k_j^* : j \in D^\tau\}\}$.
(iv) After the new arrivals it is $H_0^{(\tau+1)} := H_3^\tau \cap \{N_{ex}^\tau\}$.

Stability: A *policy* is a sequence of matching, departure and labeling decisions $\{A^\tau, D^\tau, \{k_j^* : j \in D^\tau\} : \tau \geq 1\}$. *Time average backlog* of the system under a policy ϕ is

$$Q_{sum}^\phi := \limsup_{T\to\infty} \frac{1}{T} \sum_{\tau=1}^T \mathbb{E}^\phi \left[\sum_{\ell \in \mathcal{L}} Q_\ell(\tau) \middle| H_0^1 \right].$$

A policy ϕ *stabilizes* the system if and only if $Q_{sum}^\phi < \infty$.

Threshold Accuracy: Our objective is to ensure that each sample either achieves required confidence or is processed using best-effort. Specifically, we ensure 'on average' a sample with a final label either i) has been labeled by all classifiers, or ii) has a probability of correct labeling greater or equal to a given threshold θ_{th}. We formalize this using the notion of *threshold accuracy*.

At time τ, the accuracy of a departing sample $j \in D^\tau$ with true label $s_j[M]$ is $Acc_j(\tau) := \mathbb{P}\left[k_j^* = s_j[M] \,|H_3^\tau\right]$. The event $\{\inf_{j \in D_p^\tau} Acc_j(\tau) < \theta\}$ indicates that some *partially labeled* sample has accuracy less than θ, i.e. condition (i) above is not satisfied. Threshold accuracy of θ implies the rate of such events is zero.

Threshold accuracy under a stabilizing policy ϕ is θ_{th} if and only if for all $\theta \leq \theta_{th}$

$$\limsup_{T\to\infty} \frac{1}{T} \sum_{\tau=1}^T \mathbb{P}^\phi \left[\inf_{j \in D_p^\tau} Acc_j(\tau) < \theta \right] = 0.$$

The stabilizing policy ϕ is called θ_{th}-*accurate*.

The OnDS parameters $\Psi = (p_g, \{C_i : i \in [M]\})$, the arrival rate λ_{ex}, and the desired threshold accuracy θ_{th} describes a system completely. Therefore, we define $\mathcal{P} = (\Psi, \lambda_{ex}, \theta_{th})$ as the OnUEL instance.

Causal Policy: A policy ϕ is *causal* if and only if for each $\tau \geq 1$, the matching A^τ is a random function of history H_0^τ, the parameters $\mathcal{P}$, the departure D^τ; also, the final labeling $\{k_j^* : j \in D^\tau\}$ is a random function of history H_2^τ and parameters $\mathcal{P}$. The class of *causal policies* is $\mathcal{C}$.

Pareto Region: We now define the Pareto region of a causal policy and of the OnUEL.

Definition 3.1. *Given a OnDS model Ψ, the Pareto region of a causal policy ϕ, $\Lambda^\phi(\Psi)$, is the set of all tuple $(\lambda_{ex}, \theta_{th})$ such that the system is stable under policy ϕ for λ_{ex}, and the policy ϕ is θ_{th}-accurate.*

Definition 3.2. *Given a OnDS model Ψ, the Pareto region $\Lambda(\Psi)$, is the convex closure of the Pareto regions of the individual causal policies, i.e. $\Lambda(\Psi) = conv\left(\underline{\Lambda}(\Psi)\right)$ where,*

$$\underline{\Lambda}(\Psi) := \left\{(\lambda_{ex}, \theta_{th}) : \exists \phi \in \mathcal{C}, (\lambda_{ex}, \theta_{th}) \in \Lambda^\phi(\Psi)\right\}.$$

A policy ϕ^* is Pareto-optimal if and only if $\forall \epsilon_g > 0$, and $\forall (\lambda_{ex}, \theta_{th})(1+\epsilon_g) \in \Lambda(\Psi)$, $(\lambda_{ex}, \theta_{th}) \in \Lambda^{\phi^*}(\Psi)$.

Remark: The notions of Pareto region and Pareto optimality is similar, but not identical, to the concept of capacity region and throughput optimality, resp., in stochastic optimization literature (Tassiulas & Ephremides, 1992; Stolyar, 2005). The key difference is that throughput optimality concerns a set of feasible arrival rates, whereas Pareto optimality relates arrival rates to accuracy.

3.2. Evolution of Posterior Distribution of Samples

We consider a fixed causal policy ϕ and an OnUEL instance $\mathcal{P}$. Consider any stage $st \in \{0, 1, 2, 3\}$, in round $\tau \geq 1$. S^τ is the set of samples in memory; and for each sample $j \in S_j^\tau$ the true label is $s_j[M]$. Given a vector $\mathbf{t} = \{t_j \in [K] : j \in S^\tau\}$, we are interested in the posterior probability of the true labels being $\mathbf{t}$ given the history upto the time τ, i.e. $\mathbb{P}^\phi\left(\{s_j[M]{=}t_j : j{\in}S^\tau\}|H_{st}^\tau\right)$.

Dependence on Entire History: The history H_{st}^τ is formed by stacking up the snapshots of the system in each stage upto the stage st in round τ. What does this history include? It includes *all other possible events* that have happened in the past. A tiny part of it is the partial label of the samples present in the system. Recall, the partial labels are $\{\hat{\mathbf{s}}_j^\tau : j \in S^\tau\}$ for stages $st = 0, 1$, and $\{\hat{\mathbf{s}}_j^{(\tau+1)} : j \in S^\tau\}$ for stages $st = 2, 3$.

Let us consider a specific event: *In round τ and stage 0, sample j is in the memory with partial label $\hat{s}_j^\tau$.*

The set of events that lead to this observation are as follows.
(1) The sample j arrives at some round before τ.
(2) In each round, from arrival upto τ, it was either kept unmatched or it is matched with a specific classifier.
(3) In each round, from arrival upto τ, when matched to a classifier it obtained a specific label.
(4) In each round, from arrival upto τ, the sample is not evicted from the system.

The events in step (3) are independent of the evolution of other samples as the labels are generated independently given the true label $s_j[M]$. However, the events in step (1), (2) and (4) couple the posterior evolution with the history. In step (1), we know all the samples $j' < j$ have arrived, because the samples are numbered in the order of their arrival into the system. In step (2), the matching events impose a fixed order in which the labels are collected for this sample. Further, these matching events in step (2) and departure events in step (4) are taken collectively for all the samples present. This implies that the evolution of all the other samples which were present in the system from arrival of j upto τ, are tightly coupled with the event under consideration.

Notations for Posterior Evolution: We now setup some *notations* for describing the posterior evolution.

- We adopt the convention $C_i(k, \mathbf{e}) = 1$ for all $k \in [K]$ and $i \in [M]$.
- Let us denote $\forall \ell \in [K]_e^M, k \in [K]$,
$$P_\Psi(\ell, k) := p_g(k) \prod_{i \in [M]} C_i(k, \ell(i)). \quad (1)$$
- For all $\ell \in [K]_e^M$, $k \in [K]$, let $acc(\ell, k) := \frac{P_\Psi(\ell,k)}{\sum_{k'} P_\Psi(\ell,k)}$.
- For all $i{\in}[M], k{\in}[K]$, and for any $\ell \in [K]_e^M$ let
$$P_\Psi(k|i,\ell) := \begin{cases} 0, \text{ if } \ell[i] \neq \mathbf{e}, \\ \frac{\sum_{k' \in [K]} P_\Psi(\ell,k') C_i(k',k)}{\sum_{k',k'' \in [K]} P_\Psi(\ell,k') C_i(k',k'')}, \text{o/w.} \end{cases}.$$
- For all $i{\in}[M]$, and for any two $\ell, \ell' \in [K]_e^M$ if $\forall i' \in [M], i' \neq i, \ell(i') = \ell'(i')$ we define $P_\Psi(\ell'|i,\ell) := P_\Psi(\ell'(i)|i,\ell)$. Otherwise, $P_\Psi(\ell'|i,\ell) := 0$.
- Given a vector $\mathbf{t}{\in}[K]^{|S^\tau|}$, let $n_{k\,\ell}^{(\tau,st)}(\mathbf{t})$ denote the number of samples in memory with partial label ℓ and candidate label k, in round $\tau \geq 1$ immediately after stage $st \in \{0, 1, 2, 3\}$.

Specifically, for all $k{\in}[K], \ell{\in}[K]_e^M$,
$$n_{k\,\ell}^{(\tau,3)}(\mathbf{t}) := |\{j \in S^\tau : t_j = k, \hat{\mathbf{s}}_j^{(\tau+1)} = \ell\}|,$$
$$n_{k\,\ell}^{(\tau,st)}(\mathbf{t}) := |\{j \in S^\tau : t_j = k, \hat{\mathbf{s}}_j^\tau = \ell\}|, \forall st \neq 3.$$

Conditional Independence: We show that under a causal policy the posterior probability of the true labels of each sample is independent of the history given the collected labels for that sample.

Assumption 3.3. *The initial state is finite, $|S^1|{<}\infty$, and*

there is no labeled sample, i.e. $Q_\ell(1)=0\ \forall \ell\in[K]_e^M\setminus\{\mathbf{e}\}^M$.

The following lemma gives exact expression for the joint posterior of the samples conditioned on the history. The above assumption ensures the prior distribution of the samples present in round 1 is well behaved.

Lemma 3.4. *For a given problem instance $\mathcal{P}$ satisfying Assumption 3.3, under any causal policy ϕ, the following statements hold w.p. 1, for all $\tau \geq 1$, stage $st \in \{0,1,2,3\}$, and $\mathbf{t}\in[K]^{|S^\tau|}$,*

$$\mathbb{P}\left(\forall j\in S^\tau, s_j[M]=t_j|H_{st}^\tau\right) = \prod_{j\in S^\tau}\mathbb{P}\left(s_j[M]=t_j|H_{st}^\tau\right)$$

$$\propto \prod_{j\in S^\tau}\left(p_g(t_j)\prod_{i\in[M]}C_i(t_j,\hat{\mathbf{s}}_j^{\tau_{st}}(i))\right)\begin{bmatrix}\tau_{st}:\tau_0,\tau_1=\tau\\ \tau_2,\tau_3=(\tau+1)\end{bmatrix}$$

$$\propto \prod_{\ell\in[K]_e^M,k\in[K]} P_\Psi(\ell,k)^{n_{k\,\ell}^{(\tau,st)}(\mathbf{t})}. \tag{2}$$

Remarks on Lemma 3.4: Firstly, Lemma 3.4 shows that the posterior admits a product form across all the samples present in the system. Secondly, for each sample the probability is independent of the history given only the partial labels. Finally, this implies the posterior is independent of the *sample ids* given the counts $n_{k\,\ell}^{(\tau,st)}(\mathbf{t})$.

The following statements are simple consequences of the above lemma.

Corollary 3.5. *For a given problem instance $\mathcal{P}$ satisfying Assumption 3.3, the following statements are true for all $\tau \geq 1$ and sample j with true label $s_j[M]$.*

1) $\forall k\in[K],\ell\in[K]_e^M,\ \mathbb{P}(s_j[M]=k|\hat{\mathbf{s}}_j^\tau=\ell)=acc(\ell,k)$.

2) $\forall i\in[M],k\in[K],\ell\in\{\ell'\in[K]_e^M : \ell'[i]=\mathbf{e}\}$, $\mathbb{P}(Cl^\tau[i]=k|A^\tau[i]=j\wedge\hat{\mathbf{s}}_j^\tau=\ell)=P_\Psi(k|i,\ell)$.

The maximum likelihood estimate (MLE) of the true label for any sample with partial label ℓ is the same. We denote it as $k^*(\ell):=\arg\max_{k\in[K]}acc(\ell,k)$. Also, we denote $acc^*(\ell):=acc(\ell,k^*(\ell))$.

3.3. Characterization of the Pareto Region

Network Model: We construct a network-flow instance to characterize the Pareto region for the problem instance $(\Psi,\lambda_{ex},\theta_{th})$. The feasibility of the flow instance will imply that $(\lambda_{ex},\theta_{th})$ is in the Pareto region, as proven in Theorem 3.6 below.

The network consists of $|[K]_e^M|$ label nodes. The label nodes ℓ, for which either $acc^*(\ell)\geq\theta_{th}$ (desired accuracy is reached) or $\forall i\in[M],\ell(i)\neq\mathbf{e}$ (all labels are collected), comprise the *set of terminal* nodes $\mathcal{T}(\theta_{th})$. Thus $\mathcal{T}^c(\theta_{th}):=[K]_e^M\setminus\mathcal{T}(\theta_{th})$ The set of *non-terminal* nodes is $[K]_e^M\setminus\mathcal{T}(\theta_{th})$. The label node $\{\mathbf{e}\}^M$ is the unique source node. There are hyper-edges connecting the nodes. Each hyper-edge corresponds to the allocation of labels to classifiers. Specifically, a hyper-edge $h\in(\mathcal{T}^c(\theta_{th}))^M$ indicates the *non-terminal label* $h(i)$ is matched with classifier i, for all $i\in[M]$.

The movement of flow is decided based on the selection of a hyper-edge. In particular, when a hyper-edge h is chosen, at most $|\{i\in[M]:h(i)=\ell\}|$ amount of flow can exit the label node ℓ. Further, the amount of flow entering label node ℓ is $\sum_{i\in[M]}P_\Psi(\ell|i,h(i))$. The all empty label node $\{\mathbf{e}\}^M$ has a incoming flow λ_{ex}. A pdf over the hyper-edges, denoted as $\mathbf{x}$, determines a flow in the network. The non-emptiness of the following polytope describes the feasibility of network flow corresponding to the tuple $(\Psi,\lambda_{ex},\theta_{th})$.

Flow Polytope, FP($\Psi,\lambda_{ex},\theta_{th}$)

- $\forall h, x_h\geq 0$ • $\sum_h x_h\leq 1$
- $\lambda_{in}(\{\mathbf{e}\}^M)=\lambda_{ex}$ • $\forall\ell,\lambda_{out}(\ell)\geq\lambda_{in}(\ell)$
- $\forall\ell\neq\{\mathbf{e}\}^M,\lambda_{in}(\ell)=\sum_h x_h\sum_{i\in[M]}P_\Psi(\ell|i,h(i))$
- $\forall\ell\notin\mathcal{T}(\theta_{th}),\lambda_{out}(\ell)=\sum_h x_h\sum_{i\in[M]}\mathbb{1}(h(i)=\ell)$

The following theorem characterizes the Pareto region of the OnUEL corresponding to OnDS Ψ.

Theorem 3.6. *Given a OnDS model Ψ satisfying Assumption 3.3, the Pareto region is:* $\Lambda(\Psi)=[0,\infty]\times[0,|\mathbf{p}_g|_\infty]\cup\{(\lambda_{ex},\theta_{th}):\theta_{th}>|\mathbf{p}_g|_\infty, FP(\Psi,\lambda_{ex},\theta_{th})\neq\emptyset\}$.

Remarks on Pareto Region: For $\theta_{th}\leq|\mathbf{p}_g|_\infty$, technically, any data rate ($\lambda_{ex}=\infty$) can be supported by giving a final label $k^*=\arg\max\{p_g(k):k\in[K]\}$. In the network flow formulation the source $\{\mathbf{e}\}^M$ itself becomes a terminal. Further, for all $\theta_{th}>|\mathbf{p}_g|_\infty$ the maximum arrival rate is less or equal to M, as each classifier processes *at most one* sample per round.

4. Learning Aided Scheduling

In this section we design dynamic matching, departure and labeling policies. Further, we do not assume the knowledge of the parameters $\mathcal{P}$ except θ_{th} (a design specification). Our algorithm tries to stabilize the system while maintaining accuracy. Therefore, different classifiers may see different true label distribution for the incoming sample. We resort to an explore-exploit learning strategy.

In the explore phase, we select an *unlabeled sample* and send it to all the classifiers.This ensures the true label distributions of the incoming exploration samples are same for all the classifiers. Using the samples classified in this phase

we gradually learn the hidden parameters with an online variant of a spectral method.

In the exploit phase, we match samples to classifiers based on the current queue length vector. For departure and labeling we use a Bayesian threshold based approach. Our algorithm resembles max-weight matching (Tassiulas & Ephremides, 1992) in network optimization.

The algorithm is structured in two separate parts. In the first part, we present the routing algorithm along with the exploration events. Here, we assume that, for all labels $\ell, \ell' \in [K]_e^M$, classifiers $i \in [M]$ and classes $k \in [K]$, $\widehat{acc}(\ell, k)$ and $\widehat{P}_\Psi(\ell'|i, \ell)$ are provided by an oracle (possibly erroneous). This oracle also has access to labels obtained during exploration events. Further, in each round the algorithm receives the queue length vector $\mathbf{Q}(\tau)$.

In the second part, deferred to Sec. 5, we construct an oracle which uses the explored samples to learn the hidden parameters, i.e. the classifiers' confusion matrices and the true class distribution. The oracle computes $\widehat{acc}(\ell, k)$ and $\widehat{P}_\Psi(\ell'|i, \ell)$ as queries arrive.

4.1. Max-weight Matching

Bipartite Graph: In any time step $\tau \geq 1$, the samples and the classifiers form the bipartite graph, $\mathcal{G}^\tau$, as given in Sec.2. We assign a weight for each edge in the graph as a function of the queue length vector $\mathbf{Q}(\tau)$. Consider an edge (i, j) for some $j \in S^\tau$ and some $i \in [M]$. For notational convenience, let sample j have the partial label ℓ (i.e. $\ell \equiv \hat{\mathbf{s}}_j^\tau$). Also, let ℓ'_k denote the unique label for which $\ell'_k(i') = \ell(i')$ for all $i' \neq i$, and $\ell'_k(i) = k$ whereas $\ell(i) = \mathbf{e}$. The weight of the edge (i, j) at time τ is given as

$$w^\tau(i, j) = \left(Q_{\ell(j)}(\tau) - \sum_{k \in [K]} \hat{P}_\Psi(\ell'_k|i, \ell(j)) Q_{\ell'_k}(\tau) \right).$$

To account for the idle machines let $w(i, \mathbf{e}) = 0$ Then, the weight of matching A is $W^\tau(A) = \sum_{i \in M} w^\tau(i, A(i))$.

Description: In each round $\tau \geq 1$ the algorithm is provided with an explore flag $Ex(\tau)$ which is a 0-1 r.v. drawn independently with mean $\epsilon_s(\tau) > 0$. Here, $\epsilon_s(\tau)$ is the *exploration rate parameter*. If exploration flag is 1, the algorithm explores by sending a sample, j, to all the classifiers. Otherwise, it creates the graph, $\mathcal{G}^\tau$, and solves the maximum bipartite matching problem to compute the matching, A^τ. Here it requires oracle access of $P_\Psi(\cdot)$.

For each classifier i, using the label $Cl^\tau(i)$ the algorithm updates the label for sample $j = A^\tau(i)$ to $\hat{\mathbf{s}}_j^{\tau+1}$. Next, the MLE accuracy $acc^*(\hat{\mathbf{s}}_j^{\tau+1})$ is obtained from the oracle. If the estimated accuracy is larger than $(\theta_{th} + \epsilon_\theta(\tau))$, the sample j departs the system with label $\widehat{k^*}(\hat{\mathbf{s}}_j^{\tau+1})$. Otherwise, it re-enters the system. The *error parameter* $\epsilon_\theta(\tau)$ accounts for the errors in oracle estimates.

Exploration: In each of the exploration rounds, a single sample is sent to all classifiers. Further, the labels acquired in this round are sent to the oracle, which may use them for learning the hidden parameters.

Algorithm 1 Max-weight with Bayesian Departure

for all $\tau \geq 1$ **do**
  Obtain Explore flag $Ex(\tau)$, and queue length $\mathbf{Q}(\tau)$
  if $Ex(\tau)$ and $Q_\emptyset(\tau) > 0$ **then** ▷ Explore
    Find a j s.t. $\hat{\mathbf{s}}_j^\tau = \mathbf{e}$, set $A^\tau[i] = j, \forall i \in [M]$
  else ▷ Exploit
    Set $A^\tau = \arg\max_{A \in \mathfrak{A}^\tau} W^\tau(A)$
  for all $i \in [M]$ **do**
    Update label of j: $= A^\tau[i]$ to $\hat{\mathbf{s}}_j^{\tau+1}$ using $Cl^\tau(i)$
    if $\widehat{acc}^*(\hat{\mathbf{s}}_j^{\tau+1}) \geq \theta_{th} + \epsilon_\theta(\tau)$ **then** ▷ Departure
      Sample j departs with label $\widehat{k^*}(\hat{\mathbf{s}}_j^{\tau+1})$
    else
      Sample j re-enters the system

Query Complexity: The algorithm makes oracle queries while both computing A^τ and deciding whether a sample should depart the system. Each of the $M|\mathbf{Q}(\tau)|_1$ edges in the network, $\mathcal{G}^\tau$, requires at most K $\hat{P}_\Psi(\cdot)$-queries. Thus, computing A^τ requires $KM|\mathbf{Q}(\tau)|_1$ queries. For each matched sample, K $\widehat{acc}(\cdot)$-queries are required to take departure and final labeling decisions. This leads to an additional KM queries. Therefore, the query complexity per round is $\mathcal{O}(KM|\mathbf{Q}(\tau)|_1)$.

Time Complexity: The creation of the graph $\mathcal{G}^\tau$ requires $\mathcal{O}(KM|\mathbf{Q}(\tau)|_1)$ time. On the unbalanced graph $\mathcal{G}^\tau$ with (possibly) $|\mathbf{Q}(\tau)|_1 >> M$, solving the Max-weight matching takes another $\mathcal{O}(\max\{M^2 \log(M), M|\mathbf{Q}(\tau)|_1\})$ time. Overall, the time complexity is $O(KM|\mathbf{Q}(\tau)|_1)$ time per round. The amortized time complexity per round under the max-weight policy is given as $O(\max\{M^2 \log(M), M\mathcal{Q}_{sum}^{MW}\})$. Here, $\mathcal{Q}_{sum}^{MW}$ is the time-average backlog of Algorithm 1.

4.2. Analysis of Algorithm 1

In this section we present the guarantees of the Max-weight algorithm. Specifically, we show Algorithm 1 is Pareto-optimal when the oracle has certain asymptotic convergence properties, which we make precise shortly.

($\epsilon(\sigma), \delta(\sigma)$) Oracle: An oracle is an'(α, β) oracle' if and only if, for all (i) partial labels $\ell, \ell' \in [K]$, (ii) classifiers $i \in [M]$, and (iii) classes $k \in [K]$, the oracle with access to σ *exploration samples* satisfies,
(1) $|acc(\ell, k) - \widehat{acc}(\ell, k)| \leq \sigma^{-\alpha}$ w.p. $\geq (1 - \sigma^{-\beta})$,
(2) $|P_\Psi(\ell'|i, \ell) - \widehat{P}_\Psi(\ell'|i, \ell)| \leq \sigma^{-\alpha}$ w.p. $\geq (1 - \sigma^{-\beta})$.

The following theorem provides guarantees for Algo-

rithm 1, under the assumption that an (α, β) oracle exists with $\alpha, \beta > 0$. In Sec 5, we construct such an oracle.

Theorem 4.1. *Given an OnUEL instance $\mathcal{P}=(\Psi, \lambda_{ex}, \theta_{th})$ such that Assumption 3.3 is valid, $\theta_{th} > |\mathbf{p}_g|_\infty$, and $(\lambda_{ex}, \theta_{th})(1+\epsilon_g) \in \Lambda(\Psi)$ for some $\epsilon_g > 0$; and given an (α, β) oracle with $\alpha, \beta > 0$; under Algorithm 1 with parameters $\epsilon_s(\tau) = \frac{\log(\tau)}{\tau}$ and $\epsilon_\theta(\tau) = \frac{2}{\log(\tau)^\alpha}$; then*

$$\mathcal{Q}_{sum}^{MW} \leq O\left(\frac{\max(\lambda_{ex}^2, M^2)}{\epsilon_g \lambda_{min}}\right),$$

where $\lambda_{min} = \lambda_{ex}\left(\min_{\ell \in \mathcal{T}(\theta_{th}(1+\epsilon_g))} \sum_k P_\Psi(\ell, k) p_g(k)\right)$. *Furthermore, the threshold accuracy of Algorithm 1 is θ_{th}.*

5. Learning with Explored Samples

In this section, we describe the unsupervised learning of the confusion matrices $\{\mathbf{C}_i : i \in [M]\}$ and the true probability distribution $\mathbf{p}_g$. Specifically, our purpose is to design a (α, β)-oracle that provides the $\widehat{acc}(\ell, k)$ and $\widehat{P}_\Psi(\ell'|i, \ell)$ for all ℓ, ℓ'. We require the following assumption which is common in the literature.

Assumption 5.1. *[Assumptions on OnDS]*
(i) The entries of confusion matrix is strictly positive, i.e. $\forall i \in [M]$ and $k, k' \in [K]$, $C_i(k, k') \geq \rho > 0$.
(ii) The class distribution is also bounded from below, i.e. for all $k \in [K]$ $p_g(k) \geq \rho' > 0$.
(iii) Finally, for all classifiers $i \in [M]$, the diagonal terms in the confusion matrix dominates, i.e. $\min_{k,k' \in [K], k' \neq k} (C_i(k,k) - C_i(k,k')) \geq \kappa > 0$.

Existence of an (α, β)-Oracle: Our algorithm is closely related to the one-shot unsupervised learning in paper (Zhang et al., 2014). The algorithm proposed in (Zhang et al., 2014). ensures that for fixed constants $\delta_U < 1$ and $\epsilon_U > 0$, given $O(\frac{K^5}{\log}(K + M/\delta_U)/\epsilon_U^2)$ number of samples, with probability at least $(1 - \delta_U)$ the confusion matrices $\mathbf{C}_i$ for all $i \in [M]$, and the true class distribution $\mathbf{p}_g$ are recovered with maximum error ϵ_U (Theorem 2 in (Zhang et al., 2014)). Furthermore, as $\widehat{P}_\Psi(\ell'|i, \ell)$ and $acc(\ell, k)$ are function of the confusion matrices and the true class distribution, the error in the corresponding term is also bounded.

Specifically, given (i) $\max_{i \in [M]} |\mathbf{C}_i - \widehat{\mathbf{C}_i}|_\infty \leq \mathcal{O}(\epsilon_U)$ and (ii) $|\mathbf{p}_g - \widehat{\mathbf{p}_g}|_\infty \leq \mathcal{O}(\epsilon_U)$, under Assumption 5.1, we have $|acc(\ell, k) - \widehat{acc}(\ell, k)| = \mathcal{O}\left(\frac{\epsilon_U}{(\rho' \rho^M)^2}\right)$. This ensures that with σ samples, the error in the estimates $\widehat{P}_\Psi(\cdot)$ and $\widehat{acc}(\cdot)$ are bounded from above by $\mathcal{O}(\sigma^{-\alpha})$, for all $\alpha < 1/2$, with probability at least $1 - \sigma^{-\beta}$ for any $\beta < 1$. Therefore, an (α, β) oracle exists for $\alpha > 0$ and $\beta < 1$.

This shows that under the additional Assumption 5.1 Maxweight algorithm is Pareto Optimal.

Algorithm 2 We design an online spectral learner algorithm, Algorithm 2, as Algorithm 1 in (Zhang et al., 2014) is inefficient for our problem (see appendix for details).

Algorithm 2 Online Spectral Learner

1: **Initialize:** Eigenvectors $\{v_k : k \in [K]\}$
2: **for all** round $\tau \geq 1$ **do**
3: Obtain Explore flag $Ex(\tau)$ and new labels Cl^τ
4: **if** $Ex(\tau)$ is true **then** ▷ Moment Update
5: Update the moments $\hat{M}_2$ and $\hat{M}_3$.
6: Generate whitened Tensor $T_0 = \hat{T}$.
. . . . . .
7: **for all** $r \in [K]$ **do** ▷ Online RTP
8: **while** v_r does not satisfy Eq.(3) **do**
9: Randomly pick v_r on k-dim unit sphere.
10: Repeat $v_r \leftarrow \frac{T_r(I, v_r, v_r)}{\|T_r(I, v_r, v_r)\|_2}$, R_U times.
11: Repeat $v_r \leftarrow \frac{T_r(I, v_r, v_r)}{\|T_r s(I, v_r, v_r)\|_2}$, R_U times.
12: Let $T_{(r+1)} \leftarrow T_r - T_r(v_r, v_r, v_r) \cdot v_r^{\otimes 3}$
. . . . . .
13: Recover parameters using $\{v_k : k \in [K]\}$.
14: Answer $\widehat{acc}(\cdot)$ and $\widehat{P}(\cdot)$ queries.

$$|T_r(v, v, v)| \geq \max\left\{\frac{\|T_r\|_F}{2\sqrt{K-r}}, \frac{\|T_r(I, I, v)\|_F}{1.05}\right\}. \quad (3)$$

Algorithm 2, in line 8, checks if the past eigenvector v_r for all $r \in [K]$ satisfies the condition (3) in (Anandkumar et al., 2015). The rest of the steps are identical to the approach in (Zhang et al., 2014), and thus omitted for lack of space. Specifically, line 5 in Algorithm 2 uses (2a) and (2b) in (Zhang et al., 2014), and line 13 in Algorithm 2 uses (5) in (Zhang et al., 2014). We do not require the expectation maximization steps therein.

The following lemma states that Algorithm 2 is an (α, β)-oracle with constant amortized time complexity. Here, we treat dependence on M and K as constant.Further, the Lemma implies that Algorithm 1 along with Algorithm 2 is Pareto optimal.

Lemma 5.2. *Under Assumption 5.1, Algorithm 2 with $R_U = \Theta(1)$ is an (α, β)-oracle for any $\alpha < 1/2$ and any $\beta < 1$. Furthermore, the amortized time complexity of Algorithm 2 is $\mathcal{O}(1)$.*

6. Empirical Results

Experimental Setup: For our experiments, we create two datasets where the classes of Cifar-10 (Krizhevsky & Hinton, 2009) are merged into larger groups, effectively giving us fewer classes. Our ensembles consist of deep convolutional neural networks. Specifically, we consider two modifications of Cifar-10. In the first, we have three groups: (airplanes, ships, trucks, cars), (birds, frogs cats), and (dogs, deer, horses). In the second we have five groups: (airplanes, ships), (trucks, cars), (birds, frogs) (cats, dogs) and (deer, horses).

The first ensemble has 6 classifiers - three Alexnet, one VGG-19, and two ResNet-18 neural nets (Krizhevsky et al., 2012; Simonyan & Zisserman, 2014; He et al., 2016). The second ensemble has 6 - one VGG-11 , one VGG-16, two VGG-19, and two ResNet-18 neural nets. Each classifier is individually trained.

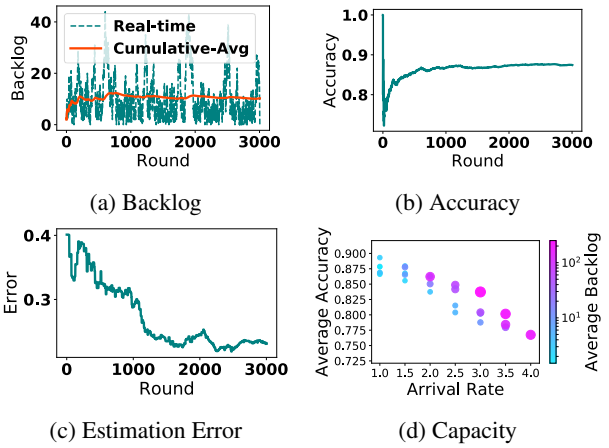
The accuracies of the various net architectures are different across the two variations of Cifar-10. The ResNet-18 neural nets out perform the other variants, whereas Alexnet neural nets perform the worst. Therefore, the performance of the first ensemble is worse than the other. The classifiers' accuracies and confusion matrices are given in the supplementary material. In the supplementary material, we show that across a broad range of arrival rates the proposed algorithm attains 2% to 5% higher average accuracy compared to a random scheduler.

Convergence Property: We now study our algorithm's temporal dynamics. The arrival rate, λ_{ex} and threshold accuracy, θ_{th} were set to 2 samples/round and 0.85, resp. In Fig. 1a, we show that inside the Pareto region the sum of queue lengths (a.k.a. backlog) fluctuates around an average of 10.8. Further, the convergence to this average value happens within a few hundred rounds. The average accuracy (i.e. departures with correct labels / total departures) of the system also converges to an average 0.87 as showed in Fig.1b. In Fig. 1c we observe that the online spectral learning effectively learns the parameters involved. The error in this figure is the total error in estimating the confusion matrices and the true class distributions. Specifically, the error is given as $\left(\|\mathbf{p}_g - \hat{\mathbf{p}}_g\|_1 + \frac{1}{M}\sum_{i\in[M]}\|\hat{\mathbf{C}}_i - \mathbf{C}_i\|_1\right)$ Similar results are noted for the other dataset in Fig. 2a-2c.

Pareto Region: We study the Pareto regions of the two ensembles in Fig. 1d and Fig. 2d. In this figure, the circles' sizes depict the logarithm of the average queue length for a given $(\lambda_{ex}, \theta_{th})$ pair. We see that for higher arrival rates, the average queue lengths start increasing for smaller thresholds, and vice versa. Furthermore, we observe that the Pareto region of the first ensemble is smaller than the second ensemble. This is consistent with more accurate ensembles supporting larger Pareto regions.

7. Related Work

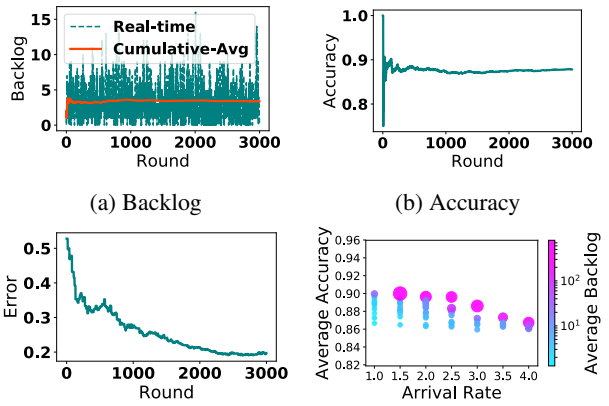
Our work belongs to the family of techniques derived from the Dawid-Skene model (Dawid & Skene, 1979). This model performs unsupervised ensemble learning (UEL), using an expectation maximization (EM) based algorithm. The need for UEL appears in fields varying from anthropology (Romney et al., 1986) to medical diagnostics (Zhou et al., 2009). Most recently, UEL has been of interest to the crowdsourcing community (Snow et al., 2008; Sheng et al., 2008; Whitehill et al., 2009; Raykar et al., 2010).



(a) Backlog (b) Accuracy

(c) Estimation Error (d) Capacity

Figure 1: Performance of Ensemble-1 on 3-Group Dataset



(a) Backlog (b) Accuracy

(c) Estimation Error (d) Capacity

Figure 2: Performance of Ensemble-2 on 5-Group Dataset

This has led to the development of various approaches, such as weighted majority voting (Li et al., 2013), message-passing (Karger et al., 2011; 2014; Ok et al., 2016), and spectral method based estimation (Zhang et al., 2014; Jain & Oh, 2014; Jaffe et al., 2016), which all have provable guarantees of classification error bounds and learning the parameters defining the ensemble and data.

We have developed an adaptive task routing technique with latent labels. In this sense, our work is most similar to (Massoulié & Xu, 2016; Shah et al., 2017), but with some important distinctions. The setting in (Massoulié & Xu, 2016) allows for resampling of classifiers, whereas our ensemble is fixed. In (Shah et al., 2017), the departure decisions are not made by the ensemble, but in our approach, both the departure and labeling decisions are made by the ensemble (thus statistically correlating the labels acquired and departures). Finally, unlike (Massoulié & Xu, 2016; Shah et al., 2017), our technique requires no *a priori* knowledge of the classifiers or incoming data.

Acknowledgments

This work is supported by ARO grant W911NF-17-1-0359 and the US DoT supported D-STOP Tier 1 University Transportation Center.

References

Anandkumar, A., Ge, R., Hsu, D., Kakade, S. M., and Telgarsky, M. Tensor decompositions for learning latent variable models (a survey for alt). In *International Conference on Algorithmic Learning Theory*, pp. 19–38. Springer, 2015.

Dawid, A. P. and Skene, A. M. Maximum likelihood estimation of observer error-rates using the em algorithm. *Applied statistics*, pp. 20–28, 1979.

Gopalan, A., Caramanis, C., and Shakkottai, S. On wireless scheduling with partial channel-state information. *IEEE Transactions on Information Theory*, 58(1):403–420, 2012.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Jaffe, A., Fetaya, E., Nadler, B., Jiang, T., and Kluger, Y. Unsupervised ensemble learning with dependent classifiers. In *Artificial Intelligence and Statistics*, pp. 351–360, 2016.

Jain, P. and Oh, S. Learning mixtures of discrete product distributions using spectral decompositions. In *Conference on Learning Theory*, pp. 824–856, 2014.

Karger, D. R., Oh, S., and Shah, D. Iterative learning for reliable crowdsourcing systems. In *Advances in neural information processing systems*, pp. 1953–1961, 2011.

Karger, D. R., Oh, S., and Shah, D. Budget-optimal task allocation for reliable crowdsourcing systems. *Operations Research*, 62(1):1–24, 2014.

Krizhevsky, A. and Hinton, G. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Li, H., Yu, B., and Zhou, D. Error rate bounds in crowdsourcing models. *arXiv preprint arXiv:1307.2674*, 2013.

Massoulié, L. and Xu, K. On the capacity of information processing systems. In *Conference on Learning Theory*, pp. 1292–1297, 2016.

Neely, M. J., Modiano, E., and Rohrs, C. E. Dynamic power allocation and routing for time-varying wireless networks. *IEEE Journal on Selected Areas in Communications*, 23(1):89–103, 2005.

Ok, J., Oh, S., Shin, J., and Yi, Y. Optimality of belief propagation for crowdsourced classification. In *International Conference on Machine Learning*, pp. 535–544, 2016.

Raykar, V. C., Yu, S., Zhao, L. H., Valadez, G. H., Florin, C., Bogoni, L., and Moy, L. Learning from crowds. *Journal of Machine Learning Research*, 11(Apr):1297–1322, 2010.

Romney, A. K., Weller, S. C., and Batchelder, W. H. Culture as consensus: A theory of culture and informant accuracy. *American anthropologist*, 88(2):313–338, 1986.

Shah, V., Gulikers, L., Massoulié, L., and Vojnović, M. Adaptive matching for expert systems with uncertain task types. In *Communication, Control, and Computing (Allerton), 2017 55th Annual Allerton Conference on*, pp. 753–760. IEEE, 2017.

Shaham, U., Cheng, X., Dror, O., Jaffe, A., Nadler, B., Chang, J., and Kluger, Y. A deep learning approach to unsupervised ensemble learning. In *International Conference on Machine Learning*, pp. 30–39, 2016.

Sheng, V. S., Provost, F., and Ipeirotis, P. G. Get another label? improving data quality and data mining using multiple, noisy labelers. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 614–622. ACM, 2008.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Snow, R., O'Connor, B., Jurafsky, D., and Ng, A. Y. Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In *Proceedings of the conference on empirical methods in natural language processing*, pp. 254–263. Association for Computational Linguistics, 2008.

Stolyar, A. L. Maximizing queueing network utility subject to stability: Greedy primal-dual algorithm. *Queueing Systems*, 50(4):401–457, 2005.

Tassiulas, L. and Ephremides, A. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. *IEEE transactions on automatic control*, 37(12):1936–1948, 1992.

Whitehill, J., Wu, T.-f., Bergsma, J., Movellan, J. R., and Ruvolo, P. L. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *Advances in neural information processing systems*, pp. 2035–2043, 2009.

Zhang, Y., Chen, X., Zhou, D., and Jordan, M. I. Spectral methods meet em: A provably optimal algorithm for crowdsourcing. In *Advances in neural information processing systems*, pp. 1260–1268, 2014.

Zhou, X.-H., McClish, D. K., and Obuchowski, N. A. *Statistical methods in diagnostic medicine*, volume 569. John Wiley & Sons, 2009.