# Robust Submodular Data Partitioning for Distributed Speech Recognition

*Jun Qi*[1], *Javier Tejedor*[2]

1. Electrical Engineering, University of Washington, Seattle, USA
2. Department of Electronics, University of Alcala, Madrid, Spain

qij13@u.washington.edu, javier.tejedor@depeca.uah.es

## Abstract

Distributed deep neural networks are commonly employed for building automatic speech recognition (ASR) systems. In this work, we employ the robust submodular partitioning approach, which aims to split the training data into small disjoint data subsets and use each of these subsets to train a particular deep neural network. Two efficient algorithms are used as robust submodular functions [1], namely 'Greedi-Max' and 'Minorization-Maximization' [2], which are guaranteed to provide tight approximations to the submodular data partition problem. Experiments on TIMIT database show that each of the distributed neural networks trained by the submodular data subset obtains better results than that trained on any subset of data partitioned in a random way.

In addition, multi-class adaboost is effectively used to fuse the outputs of the deep neural networks and provides competitive ASR results compared with the traditional ASR system. Besides, the time incurred by acoustic modeling is significantly reduced, which delivers us further benefits.

**Index Terms**: Robust Submodular Data Partitioning, Greedi-Max, Minorization-Maximization, Speech Recognition, Deep Neural Network

## 1. Introduction

The recent development of the automatic speech recognition (ASR) allows training acoustic models based on deep neural networks (DNNs) in a distributed way [3, 4]. In the stage of transcribing speech signals, all outputs of the DNNs are somehow combined to produce relatively better ASR results than traditional ASR systems [3].

However, there are several problems in the prior work [3] which hinder the distributed ASR system from using widely in practice: one is that each of the DNNs is trained by using all the training data, which demands incredibly powerful computation resources for acoustic modeling; another problem is that the prior attempt makes use of a so complicated model combination strategy that it is hard to be scalable to a large dataset. Besides, it usually takes too much time in DNN training on large datasets. Even though efficient GPUs are used, we have to spend much time in waiting for the DNN training before testing or modifying acoustic models.

Our work aims to partition a dataset into disjoint data subsets and each subset of data is used for training a particular DNN of the distributed ASR system. Generally, it is an NP-hard problem to find an optimal solution when searching for subsets of data which can maintain the performance for all DNNs. Fortunately, submodularity provides a greedily near-optimal approximation to the NP-hard problem if the objective function of the data partition problem satisfies one of the equivalent definitions for submodular functions [5].

One commonly used definition of a submodular function is based on the first order diminishing return [5]. Specifically, a set function $f$ is submodular if and only if for any two subsets $A, B$ and a ground set $V$, there are $A, B \subseteq V$ and an element $k \in V$, such that $A \subseteq B$ and $k \notin B$, and for which (1) is satisfied:

$$f(A \cup \{k\}) - f(A) \geq f(B \cup \{k\}) - f(B). \quad (1)$$

The most advantage of the submodular function is that it can deliver a close approximation to discrete optimization problems with a constant guarantee. For example, the feasible solution found by the efficient greedy algorithm for the submodular function maximization with a cardinality constraint is at least $1 - e^{-1}$ close to the optimal answer, while there is at most $50\%$ gap between the approximation and the best optimal solution when the knapsack constraint is concerned [6]. As for the prior work about submodular data selection for ASR tasks [7, 8], the related problem is formulated as the following submodular function maximization:

$$\max_{S \subseteq 2^V} f(S), s.t., |S| \leq l, \quad (2)$$

where the symbols $2^V$, $S$, and $l$ denote a set of all possible subsets of a ground set $V$, a subset from elements of $2^V$, and a constant for the budget constraint respectively. Besides, the submodular function $f$ at the empty set should be 0.

The function $f(S)$ is simply defined as a combination of a concave function $g$ and a modular function $m_u(v)$ associated with a clustered triphone feature $u$, as shown in (3):

$$f(S) = \sum_{v \in S} g(\sum_{u} m_u(v)), \quad (3)$$

where $g$ is usually defined as square root and $m_u(v)$ represents the number of frames of the feature $u$ in the utterance $v$.

An approximate solution to the problem shown in (2) is efficiently found by using an iteratively greedy algorithm. The work in [7] suggests that a small subset with $10\%$ of the data selected by the feature-based submodular function is capable of maintaining the ASR performance.

Our work, however, tries to partition the data into several disjoint subsets for training distributed acoustic models based on deep neural networks. To do so, the robust submodular data partition problem with the most recently proposed approximation algorithms is used [2]. In addition, we make use of the phonetic knowledge when doing the conversion from triphone states to biphone states to formulate the heterogeneous submodular functions in the robust submodular data partition problem.

Other benefit for the ASR system comes from the DNN output combination. To combine the outputs of all acoustic models,

it is necessary to assign the weights to the outputs of the DNNs in a simply but self-adaptive way. In this work, a multi-class adaboost approach [9] is used, which boosts several weak classifiers into a stronger one.

In fact, the most benefit of the distributed ASR system is not only maintaining or achieving better ASR results, but the time incurred by acoustic modeling based on the DNN is expected to be significantly reduced such that we can test and modify the acoustic models more efficiently.

The rest of the paper is organized as follows: Section 2 presents the robust submodular partitioning as well as related fast approximation algorithms and how to formulate heterogeneous submodular functions. Section 3 introduces the multi-class adaboost algorithm. Experiments are reported in Section 4 and the paper is concluded in Section 5.

## 2. Robust Submodular Partitioning

### 2.1. Problem formulation

The problem of partitioning the data into subsets is formulated as a robust submodular partitioning problem [1] as shown in (4):

$$\max_{\pi\in\Pi} \min_{i} f_i(A_i^{\pi}) \tag{4}$$

$$s.t., \ \ \cup_i A_i^{\pi} = V, A_i^{\pi} \cap A_j^{\pi} = \Phi, \forall i, j,$$

where $\{f\}_i$ refers to heterogeneous submodular functions, the set $\pi = (A_1^{\pi}, A_2^{\pi}, ..., A_m^{\pi})$ is a partition of a finite set $V$, and $\Pi$ denotes the set of all partitions of $V$ into $m$ blocks. Such a robust submodular partitioning problem is also called max-min submodular fair allocation (SFA). The following will introduce two SFA approximation algorithms which are not only efficient but can also be scalable to a large dataset.

### 2.2. Greedi-Max algorithm

The first approximation algorithm for the SFA is based on Greedi-Max, which is shown in Algorithm 1. Greedi-Max aims to add one item with the maximum marginal gain to the block whose current solution is minimum. Initializing $\{A_i\}_{i=1}^m$ with the empty set, sizes of blocks are incrementally increased by greedily improving the overall objective $\min_{i=1,...,m} f_i(A_i)$ until $\{A_i\}_{i=1}^m$ forms a partition $\hat{\pi} = \{A_1^{\hat{\pi}}, A_2^{\hat{\pi}}, ..., A_m^{\hat{\pi}}\}$.

Algorithm 1: Greedi-Max

1: Input: $\{f_i\}_{i=1}^m, m, V$.
2: Let $A_1 = ... = A_m = \Phi$; $R = V$.
3: while $R \neq \Phi$ do:
4: $\quad j^* \in arg\min_j f_j(A_j)$
5: $\quad a^* \in \max_{a\in R} f_j(a|A_{j^*})$
6: $\quad A_{j^*} \leftarrow A_{j^*} \cup \{a^*\}$; $R \leftarrow R\backslash\{a^*\}$
7: end while
8: Output: $\hat{\pi}$.

Each subset partitioned by the Greedi-Max is guaranteed at least $\frac{1}{m}$ close to the optimal solution [10], as shown in (5).

$$\min_{i=1,..,m} f_i(A_i^{\hat{\pi}}) \geq \frac{1}{m} \max_{\pi\in\Pi} \min_{i=1,...,m} f_i(A_i^{\pi}), \forall i \tag{5}$$

### 2.3. Minorization-Maximization algorithm

The second approximation algorithm is the Minorization-Maximization (MMAX). As shown in Algorithm 2, the MMAX aims to iteratively maximize tight lower bounds of submodular functions $\{f_i\}_{i=1}^m$.

Algorithm 2: Minorization-Maximization (MMAX)

1: Input: $\{f_i\}_{i=1}^m, m, V$, partition $\pi^0$.
2: Let $t = 0$.
3: repeat
4: $\quad$ for $i = 1, ..., m$ do
5: $\quad\quad$ Pick a subgradient $h_i$ at $A_i^{\pi^t}$ for $f_i$.
6: $\quad$ end for
7: $\quad \pi^{t+1} \in arg\max_{\pi\in\Pi} \min_i h_i(A_i^{\pi^t})$
8: $\quad$ t = t + 1
9: until $\pi^t = \pi^{t-1}$
10: Output: $\pi^t$.

Since submodular functions are regarded as discrete analogous to continuous convex functions, there are many connections of properties between submodular functions and convexity. Submodular subdifferential is such a case and is defined as (6):

$$\partial_f(X) = \{x \in R^n : f(Y) - x(Y) \geq f(X) - x(X), \forall Y \subseteq V\}, \tag{6}$$

where $x(Y) = \sum_{j\in Y} x(j)$ is a modular function.

A subgradient at $X$ is also denoted by $h_X \in \partial_f(X)$. The extreme points of $\partial_f(Y)$ can be computed via a greedy algorithm: Let $\sigma$ be a permutation of $V$ that assigns the elements in $Y$ to the first $|Y|$ positions ($\sigma(i) \in Y$ if and only if $i \leq |Y|$). Each permutation defines a chain with elements $S_0^{\sigma} = \Phi$, $S_i^{\sigma} = \{\sigma(1), \sigma(2), ..., \sigma(i)\}$, and $S_{|Y|}^{\sigma} = Y$. An extreme point $h_Y^{\sigma}$ of $\partial_f(Y)$ has each entry as (7):

$$h_Y^{\sigma}(\sigma(i)) = f(S_i^{\sigma}) - f(S_{i-1}^{\sigma}). \tag{7}$$

(7) implies that $h_Y^{\sigma}$ forms a lower bound of $f$, tight at $Y$, i.e., $h_Y^{\sigma}(X) = \sum_{j\in X} h_Y^{\sigma}(\sigma(j)) \leq f(X), \forall X \subseteq V$ and $h_Y^{\sigma}(Y) = f(Y)$. In fact, the idea of MMAX is to consider a modular lower bound tight at the set corresponding to each block of a partition. In theory [2], the algorithm of MMAX achieves a worst-case guarantee of

$$O(\min_i \frac{1 + (|A_i^{\hat{\pi}}| - 1)(1 - k_{f_i}(A_i^{\hat{\pi}}))}{|A_i^{\hat{\pi}}|\sqrt{m}log^3 m}),$$

where $\hat{\pi} = (A_1^{\hat{\pi}}, ..., A_m^{\hat{\pi}})$ is the partition obtained by the algorithm, and

$$k_{f_i}(A_i^{\hat{\pi}}) = 1 - \min_{v\in V} \frac{f_i(v|A_i^{\hat{\pi}}\backslash\{v\})}{f_i(v)} \in [0, 1].$$

### 2.4. Composing heterogeneous submodular functions

The previous work [7] composes a submodular function based on features of clustered triphone states, whereas our work takes biphone states as features for constructing heterogeneous submodular functions. The phonetic knowledge suggests that a triphone state can be converted into 8 broad classes of biphone states. For instance, as shown in Table 1, a triphone state sh-iy+n[2] corresponds to 8 broad classes

of biphone states (palatal-iy[2], fricative-iy[2], iy[2]+nasal, unvoiced-iy[2], continuent-iy[2], iy[2]+coronal, iy[2]+voiced, and iy[2]+alveolar).

Accordingly, if each submodular function $f_i$ is associated with a class of biphone states, there are totally 8 heterogeneous submodular functions in ASR tasks.

| **Place of articulation** |
|---|
| 1. *Front Vowel*: iy ih eh ae aw ey y |
| 2. *Central Vowel*: ah er hh |
| 3. *Back Vowel*: aa ao uh uw ay ow oy |
| 4. *Coronal*: d l n s t z r th dh |
| 5. *Palatal*: sh zh jh ch |
| 6. *Labial*: b f m p v w |
| 7. *Velar*: g k ng |
| 8. *Silence*: sil |
| **Production manner** |
| 1. *High Vowel*: ih iy uh uw |
| 2. *Mid Vowel*: ah eh ey ow er |
| 3. *Low Vowel*: aa ae aw ay oy ao |
| 4. *Fricative*: jh ch s sh z f zh th v dh hh |
| 5. *Nasal*: m n ng |
| 6. *Stop Consonant*: b p t d k g |
| 7. *Approximant*: w y l r |
| 8. *Silence*: sil |
| **Voicedness** |
| 1. *Voiced*: iy ih eh ey ae aa aw ay ah ao oy ow uh uw er b d dh g jh l m n ng r v w y z zh |
| 2. *Unvoiced*: p f th t s sh ch k hh |
| 3. *Silence*: sil |
| **Miscellaneous** |
| 1. *Short Vowel*: eh ih uh ae ah y oy |
| 2. *Long Vowel*: iy uw aa |
| 3. *Diphthong*: ey aw ow ao |
| 4. *ay*: ay |
| 5. *Retroflex*: er r |
| 6. *Affricate*: ch jh |
| 7. *Alveolar*: s z t d n l |
| 8. *Continuent*: sh th dh hh m f ng v w zh |
| 9. *Non Continuent*: p b g k |
| 10: *Silence*: sil |

Table 1: Phonetic Knowledge from Triphones to Biphones.

## 3. Combining Outputs of Acoustic Models

The data subsets partitioned by the heterogeneous submodular functions $\{f_i\}_{i=1}^{8}$ are used to separately train acoustic models based on DNNs, which make it possible to train 8 DNNs for acoustic modeling in parallel. However, when transcribing a speech signal, it is necessary to fuse the outputs of all DNNs and provide only one decoding result. As illustrated in Figure 1, the key work of the model combination is to put weights to the different outputs of the DNNs. Note that a sequence of clustered triphone states is used for training all DNNs.

Our work attempts to weight the outputs of all DNNs in a self-adaptive way based on the multi-class adaboost through which an optimal scheme of weight assignment can be found. The multi-class adaboost directly deals with multi-class classification tasks instead of simplifying the problem into multiple two-class classification ones. The multi-class adaboost algorithm is shown in Algorithm 3, where $M$ refers to the number of DNNs, $c_i$ represents the $i$-th correct clustered triphone state, and $K$ denotes how many DNN targets are used. $I(\cdot)$ is an indicator function which returns 1 if the condition between the brackets is satisfied, otherwise returns 0. Adaboost tries to combine several weak classifiers and enables the error rates obtained by the combined system to become 0 in the training dataset.

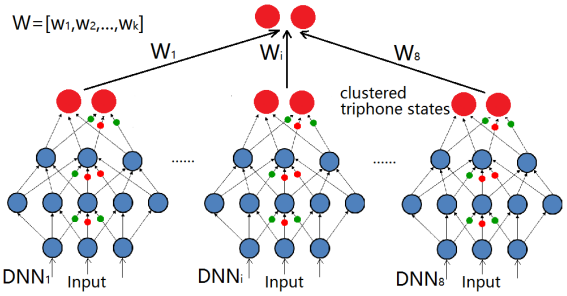


Figure 1: *The Distributed ASR System.*

When transcribing a speech signal, all clustered triphone states of DNNs should be reserved for each frame and then combined into one group of clustered triphone states by using the step 4 in the multi-class adaboost algorithm.

Algorithm 3: Multi-class Adaboost

1. Set the observation $X = (x_1, x_2, ..., x_K)$ and the test data $y$.
2. Initialize the observation weights $w_i = 1/K, i = 1, 2, ..., K$.
3. For $m = 1$ to $M$:
   (a) Fit a classifier $T^{(m)}(\cdot)$ to the observation $X$ using weights $w_i$.
   (b) Compute $err^{(m)} = \sum_{i=1}^{K} w_i \cdot I(c_i \neq T^{(m)}(x_i)) / \sum_{i=1}^{K} w_i$
   (c) Compute $\alpha^{(m)} = log\frac{1-err^{(m)}}{err^{(m)}} + log(K-1)$
   (d) Set $w_i \leftarrow w_i \cdot exp(\alpha^{(m)} \cdot I(c_i \neq T^{(m)}(x_i)))$, for $i = 1, ..., K$
4. Output $C(y) = arg\max_k \sum_{m=1}^{M} \alpha^{(m)} \cdot I(T^{(m)}(y) = k)$.

## 4. Experiments

### 4.1. Experimental setup

Our experiments are conducted on the TIMIT database. The training data consist of 3696 utterances in total. The development and test data are composed of 200 and 1200 utterances respectively. Data preprocessing includes extracting 39 dimensional Mel Frequency Cepstrum Coefficient (MFCC) features that correspond to 25.6ms speech signals. In addition, mean and variance speaker normalizations are also applied.

The acoustic models are initialized as clustered triphones modeled by 3-state left-to-right hidden Markov models (HMMs). The state emission of the HMM is modeled by the Gaussian mixture model (GMM). The DNN targets consist of approximately 3664 clustered triphone states. All sequential labels corresponding to the training data are generated by alignment based on the HMM-GMM. A 3-gram language model is used for the decoding.

The 8 subsets of data partitioned by the submodular functions are used for training 8 DNNs in parallel. The units at the input layer of each DNN correspond to a long-context feature vector that is generated by concatenating 11 consecutive

frames of the primary MFCC feature followed by a discrete cosine transformation (DCT). Thus, the dimension of the initial long-context feature is $429$ and the number is reduced to $361$ after the DCT [11]. There are $4$ hidden layers in total with a setup of $1024$-$1024$-$1024$-$1024$ for each DNN. The parameters of the hidden layers are initialized via Restricted Boltzmann Machine pre-training [12], and then fine-tuned by the Multi-layer Perceptron Back-Propagation algorithm. Besides, the feature-based maximum likelihood linear regression is applied to DNN speaker adaptation [13].

The multi-class adaboost starts to train after all DNNs have been trained. All frames of the $3696$ training utterances are used for multi-class adaboost weight adjusting. The development set is used to validate the system performance in the training process.

### 4.2. Experimental results and analysis

First, we compare the performance of the ASR system trained on the data partitioned by the two SFA approximation algorithms and by a random way. Note that data in this work refer to the utterances and that although SFA approximation algorithms do not allow sizes of all subsets to be equal, each size of random selected utterances is deliberately set to $1/8$ of the total data size. In addition, the utterances selected in the random subset do not appear in the other subsets. Table 2 shows the word error rates (WERs) on $8$ data subsets partitioned by the random way, and the Greedi-Max and MMAX algorithms respectively. Note that the results of the random partition allow $\pm 0.2$ absolute errors and that the WER of the traditional ASR baseline reaches as low as $18.5\%$ [14].

| Subset ID | Random | Greedi-Max | MMAX |
|---|---|---|---|
| 1 | 22.9 | 21.6 | 20.7 |
| 2 | 22.8 | 21.2 | 20.1 |
| 3 | 22.5 | 20.6 | 20.2 |
| 4 | 22.9 | 20.5 | 20.6 |
| 5 | 22.6 | 21.1 | 20.4 |
| 6 | 23.0 | 20.4 | 20.8 |
| 7 | 22.7 | 20.5 | 20.1 |
| 8 | 22.8 | 21.2 | 20.3 |

Table 2: WERs(%) on subsets partitioned by various ways.

As shown in Table 2, WERs corresponding to the robust submodular partition are significantly lower than those based on the random way, while the MMAX basically outperforms the Greedi-Max approximation algorithm. The best results obtained by the MMAX are attributed to the fact that more tight lower bounds of submodular functions can be iteratively found by the MMAX. By contrast, the lower bound of submodular functions is fixed in the Greedi-Max beforehand.

Table 3 compares the combination of the outputs of the DNNs with and without the use of multi-class adaboost. Without the use of multi-class adaboost means averagely putting weights to the outputs of all DNNs. As suggested in Table 3, the multi-class adaboost greatly lowers the WERs, whereas averagingly putting weights cannot achieve any improvement. Note that MMAX achieves even lower rate than the baseline system, while the WER obtained by the Greedi-Max is relatively close to the baseline result.

Furthermore, the total training time of the distributed ASR system, which includes parallel acoustic modeling on the same $8$ GPUs and training the multi-class adaboost, is approximately $4$ times faster than the traditional ASR baseline system trained in a single GPU.

| Category | Multi-class Adaboost | Averaging |
|---|---|---|
| Random | 20.4 | 22.7 |
| Greedi-Max | **18.9** | 20.9 |
| MMAX | **18.3** | 20.1 |

Table 3: WERs(%) with and without multi-class adaboost.

## 5. Conclusions

The work tries to partition the data into disjoint subsets for training distributed acoustic models based on DNNs. We formulate the problem as a robust submodular data partition and tries to find approximated solutions by 'Greedi-Max' and 'MMAX'. The experiments on the TIMIT database show that both SFA approximation algorithms are capable of partitioning the data more efficiently than a random way. In addition, the multi-class adaboost combines the outputs of all DNNs and brings in further gains to the distributed ASR system. Besides, the final results of the distributed ASR system are not only comparable to those of the traditional one, but the time incurred in DNN training is significantly reduced if all DNNs can be trained in parallel. Future work will extend the study to larger datasets for ASR.

## 6. Acknowledgement

We would like to thank Dr. Kai Wei and Prof. Jeff Bilmes at University of Washington for sharing with us the latest algorithms for submodularity and also thank Dr. Sim Khe Chai and Prof. Mark Gales at University of Cambridge for valuable discussions in speech recognition.

## 7. References

[1] A. Krause, B. McMahan, C. Guestrin, and A. Gupta, "Robust submodular observation selection," *Journal of Machine Learning Research*, no. 9, pp. 2761–2801, 2008.

[2] K. Wei, R. K. Iyer, S. Wang, W. Bai, and J. Bilmes, "Mixed robust/average submodular partitioning: Fast algorithm, guarantees, and applications," in *Proc. of NIPS*, 2015.

[3] G. Wang and K. C. Sim, "Regression-based context-dependent modeling of deep neural networks for speech recognition," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 22, no. 11, pp. 1660–1669, 2014.

[4] K. Hacioglu and B. Pellom, "A distributed architecture for robust automatic speech recognition," in *Proc. of ICASSP*, 2003, pp. 328–331.

[5] F. Bach, "Learning with submodular functions: A convex optimization perspective," *Foundations and Trends in Machine Learning*, vol. 6, pp. 145–373, 2013.

[6] G. Nemhauser, L. Wolsey, and M. Fisher, "An analysis of the approximations for maximizing submodular set functions," *Mathematical Programming*, vol. 14, pp. 265–294, 1978.

[7] K. Wei, Y. Liu, K. Kirchhoff, C. Bartels, and J. Bilmes, "Submodular subset selection for large-scale speech training data," in *Proc. of ICASSP*, 2014, pp. 3311–3315.

[8] K. Wei, Y. Liu, K. Kirchhoff, and J. Bilmes, "Unsupervised submodular subset selection for speech data," in *Proc. of ICASSP*, 2014, pp. 4107–4111.

[9] J. Zhu, H. Zou, S. Rosset, and T. Hastie, "Multi-class adaboost," *Statistics and Its Inference*, vol. 2, pp. 349–360, 2009.

[10] R. da Ponte Barbosa, A. Ene, H. L. Nguyen, and J. Ward, "The power of randomization: Distributed submodular maximization on massive datasets," in *Proc. of ICML*, 2015, pp. 1236–1244.

[11] J. Qi, D. Wang, and J. Tejedor, "Subspace models for bottleneck features," in *Proc. of Interspeech*, 2013, pp. 1746–1750.

[12] G. Hinton and R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[13] M. Gales, "Maximum likelihood linear transformation for HMM-based speech recognition," *Computer Speech and Language*, vol. 12, no. 2, pp. 75–98, 1998.

[14] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely, "The KALDI speech recognition toolkit," in *Proc. of ASRU*, 2011.