Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)

Dual-Clustering Maximum Entropy with Application to Classification and Word Embedding

Xiaolong Wang, Jingjing Wang, Chengxiang Zhai
University of Illinois
Urbana, IL 61801
{xwang95, jwang112, czhai}@illinois.edu

Abstract

Maximum Entropy (ME), as a general-purpose machine learning model, has been successfully applied to various fields such as text mining and natural language processing. It has been used as a classification technique and recently also applied to learn word embedding. ME establishes a distribution of the exponential form over items (classes/words). When training such a model, learning efficiency is guaranteed by *globally* updating the entire set of model parameters associated with *all* items at *each* training instance. This creates a significant computational challenge when the number of items is large. To achieve learning efficiency with affordable computational cost, we propose an approach named Dual-Clustering Maximum Entropy (DCME). Exploiting the primal-dual form of ME, it conducts clustering in the dual space and approximates each dual distribution by the corresponding cluster center. This naturally enables a hybrid online-offline optimization algorithm whose time complexity per instance only scales as the product of the feature/word vector dimensionality and the cluster number. Experimental studies on text classification and word embedding learning demonstrate that DCME effectively strikes a balance between training speed and model quality, substantially outperforming state-of-the-art methods.

1 Introduction

Maximum Entropy (ME), also known by a variety of other names, including log-linear, Gibbs, exponential, softmax and multinomial logistic regression models, is one of the most widely applied machine learning techniques in various fields. As a classification method, ME has seen wide-scale applications in text mining and natural language processing, such as text classification (Nigam, Lafferty, and McCallum 1999), part-of-speech tagging (Ratnaparkhi 1996) and machine translation (Berger, Pietra, and Pietra 1996). In neural networks, ME (softmax) is the building block of network architectures to transform a vector of signals into probabilities (Collobert and Weston 2008), and has been explored to learn neural probabilistic language models (Bengio et al. 2003). In recent literature, a number of word embedding algorithms have been proposed based on ME, including skip-gram, continuous bag-of-words (CBOW) (Mikolov et al. 2013; Mikolov and Dean 2013) and log-bilinear models (Mnih and Hinton 2007), among others.

ME establishes a distribution of the exponential form over items (classes/words) (See Equation (1)). Scalability becomes a crucial challenge when the number of items is large, which occurs nowadays in many real-world problems. For example, in a text classification problem of predicting the publishing venue for research papers, the number of classes can easily exceed thousands on datasets such as ACM digital library[1]; for word embedding, commonly used training corpora, with the English Gigaword[2] as an example, typically have a vocabulary of hundreds of thousands, if not millions of words.

The main computational difficulty in ME comes from the fact that one has to enumerate all items in order to obtain either the probability of a single item or the corresponding gradient (Mnih and Teh 2012). Consequently conventional ME optimization techniques such as iterative scaling (Berger, Pietra, and Pietra 1996; Darroch and Ratcliff 1972) and gradient-based algorithms (Tsuruoka, Tsujii, and Ananiadou 2009; Gao et al. 2007) are very slow to train with large numbers of items. In practice, sampling-based methods (Gutmann and Hyvärinen 2010; Mnih and Teh 2012; Bengio and Senécal 2008) are often adopted since the complexity does not hinge on the number of items. However, one drawback they possess is the inevitable sampling variance. Furthermore, only the model parameters associated with the sampled items get updated at each training instance, while the majority of the model is left unchanged, which leads to inefficient learning.

To achieve learning efficiency with affordable computational cost, we propose a Dual-Clustering Maximum Entropy (DCME) approach. It optimizes ME in a primal-dual fashion, where the multinomial *dual distribution* for each instance is exploited. The key step of DCME is to cluster the dual distributions and to approximate each of them by the corresponding *cluster center*. The dual clustering proceeds by alternating between an online update of each instance's cluster assignment and an offline calculation[3] of

[1] http://dl.acm.org/

[2] https://catalog.ldc.upenn.edu/LDC2011T07

[3] In this paper, the term "offline" is equivalent to "batch computation".

the cluster centers. This gives rise to an efficient updating scheme which splits the computation of the model subgradient into an *online* part and an *offline* part. Our proposed DCME enjoys two desirable properties: (1) The model parameters associated with *all* items are updated at *each* training instance, which ensures learning efficiency; and (2) The computational cost per instance scales as the product of the feature/word vector dimensionality[4] and the number of clusters, which yields fast training speed.

2 Background

Maximum Entropy Framework

The general formulation of ME is simple. For a data instance t, ME establishes a distribution over N items:

$$\mathrm{P}_t(i;\boldsymbol{\Theta}) = \frac{\exp(f_t(i;\boldsymbol{\Theta}))}{\sum_{j=1}^{N}\exp(f_t(j;\boldsymbol{\Theta}))}, \quad i = 1,\ldots,N. \tag{1}$$

where $f_t(i;\boldsymbol{\Theta})$ is the scoring function with model parameters $\boldsymbol{\Theta}$, which quantifies the affinity between instance t and item i[5]. In this paper, we investigate ME in two settings, namely, multi-class classification and word embedding.

For N-class classification, the dataset $\mathcal{D}$ consists of a collection of instances $\{(\mathbf{x}_t, i_t)\}$ with $\mathbf{x}_t$ being a D-dimensional feature vector and i_t a label chosen from items $1,\ldots,N$. The model $\mathbf{W} = [\mathbf{w}_1,\ldots,\mathbf{w}_N]$ is a $D\times N$ matrix which specifies the scoring function as:

$$\text{Classification:} \qquad f_t(j;\boldsymbol{\Theta}) = f_t(j;\mathbf{W}) = \mathbf{w}_j^T\mathbf{x}_t \tag{2}$$

In the word embedding setting, we focus our discussion on the continuous bag-of-words algorithm (CBOW) (Mikolov et al. 2013), but the analysis easily extends to other models as well. As a language modeling technique, it predicts the target word from a vocabulary of size N given its surrounding context. The t-th training instance contains a stream of words $w_{t,-c}, w_{t,-(c-1)},\ldots,w_{t,0},\ldots,w_{t,c-1},w_{t,c}$ with the target word $i_t = w_{t,0}$. CBOW calculates the compatibility between the j-th word in the vocabulary and the context as:

$$\text{Embedding:} \quad f_t(j;\boldsymbol{\Theta}) = f_t(j;\mathbf{V},\mathbf{H}) = f(\mathbf{v}_j,\bar{\boldsymbol{h}}_t) = \mathbf{v}_j^T\bar{\boldsymbol{h}}_t$$
$$\text{where} \quad \bar{\boldsymbol{h}}_t = \frac{1}{2c}\sum_{-c\le p\le c, p\neq 0}\mathbf{h}_{w_{t,p}} \tag{3}$$

The model parameters $\mathbf{V} = [\mathbf{v}_1,\ldots,\mathbf{v}_N]$ and $\mathbf{H} = [\mathbf{h}_1,\ldots,\mathbf{h}_N]$ are two $D\times N$ matrices of the "input" and "output" vector representations of words, respectively.

Optimization of ME

Various algorithms for ME have been studied in the literature. They approach the optimization by solving either the primal or the dual problem. The primal form maximizes the log-likelihood of the dataset. Methods of this direction, as surveyed in (Malouf 2002; Yuan, Ho, and Lin 2012), include iterative scaling algorithms (Berger, Pietra, and Pietra 1996; Darroch and Ratcliff 1972), coordinate descent (Huang et al. 2010), stochastic gradient descent (Tsuruoka, Tsujii, and Ananiadou 2009) and Quasi-Newton method (Gao et al. 2007), just to name a few. Their training complexity per instance is $\mathcal{O}(DN)$. This is a consequence of having to enumerate *all* items when computing the probability of a single item or the corresponding gradient. On the other hand, another line of research tackles the problem by maximizing the entropy of *dual distributions*. Constraint optimization techniques, such as exponentiated gradient (Collins et al. 2008) and dual coordinate descent (Yu, Huang, and Lin 2011), are investigated. Since the dimensionality of dual distributions is in fact the same as the number of items, their training complexity is still linear in N. Consequently, all these algorithms are impractical with large numbers of items due to the prohibitively expensive computational cost.

Learning with Large Item Number

Scaling algorithms for learning when the number of items N is large have become a recent research direction with focus on maintaining the training complexity sublinear in N. Among them, hierarchical approaches explore a taxonomy (of items) and convert the problem into a series of binary predictions along the tree branches, which potentially reduces the complexity from $\mathcal{O}(N)$ to $\mathcal{O}(\log N)$. Though efforts have been made in large multi-class (extreme) classification (Choromanska and Langford 2015; Choromanska, Agarwal, and Langford 2013) and word embedding (Morin and Bengio 2005; Mnih and Hinton 2009; Mikolov et al. 2013), finding balanced tree structures that provide an effective partition of items is difficult by itself, and thus their use is limited in practice. Another work of extreme classification, (Yen et al. 2016), has developed a fast active set algorithm for max-margin classifiers by exploiting the sparsity of feature vectors. The training speed-up, nevertheless, is generally insignificant for dense data representations such as word embeddings. To the best of our knowledge, the most effective approaches for training ME models with a large N are sampling-based methods, for instance (Bengio and Senécal 2008), offering a trade-off between speed and precision. In addition, as pointed out by (Mnih and Teh 2012), noise-contrastive estimation (NCE) (Gutmann and Hyvärinen 2010) is regarded as the state-of-the-art sampling algorithm which employs the idea of "learning by comparison": It reduces the N-item ME problem to a binary classification between samples from the training data and "noise" from the proposal distribution, and is guaranteed to converge to the solution of ME. Yet in practice, a slightly simpler variant, negative sampling (NS) (Mikolov and Dean 2013), is proposed to train CBOW and skip-gram though mathematically it does not solve ME. However, one drawback is that algorithms of this kind inevitably suffer from sampling variance. More crucially, the computational efficiency is gained at the expense of only updating the model parameters associated with the sampled items, while the due change of the rest is discarded. Learning efficiency is therefore sacrificed.

[4]To be precise, by taking advantage of sparsity, the complexity depends only on the number of non-zero elements in the vector.

[5]In the context of energy-based models, $-f_t(i;\boldsymbol{\Theta})$ is often referred as the energy function (Bengio et al. 2003).

3 Dual-Clustering Maximum Entropy

In this section, we present a Dual-Clustering Maximum Entropy (DCME) approach which has two advantages regarding learning and computational efficiency: (1) The model parameters associated with *all* items are updated at *each* training instance; and (2) The time complexity is independent of N.

Primal-dual ME

Different from existing approaches, DCME solves the ME problem in a primal-dual fashion. Suppose that the dataset $\mathcal{D}$ has M instances and N items where the t-th instance selects the i_t-th item. We start the derivation from the primal ME formulation which maximizes the log-likelihood:

$$\sum_{t=1}^{M} \log(\mathrm{P}_t(i_t;\boldsymbol{\Theta})) = \sum_{t=1}^{M} \big(f_t(i_t;\boldsymbol{\Theta}) - \log \sum_{j=1}^{N} \exp f_t(j;\boldsymbol{\Theta})\big) = \sum_{t=1}^{M} \big(f_t(i_t;\boldsymbol{\Theta}) - A_t(\boldsymbol{\Theta})\big) \quad (4)$$

where $A_t(\boldsymbol{\Theta})$ is referred to as the log-partition function and its conjugate dual is revealed by the following lemma (Hiriart-Urruty and Lemaréchal 1993; Wainwright and Jordan 2008):

Lemma 3.1. *Assume* $\mathrm{P}(i;\mathbf{s}) = \exp(s_i)/\sum_{j=1}^{N}\exp(s_j)$ *and* $A(\mathbf{s}) = \log\sum_{j=1}^{N}\exp(s_j)$*, the conjugate duality between the log-partition function and negative entropy states:*

$$A(\mathbf{s}) = \max_{\boldsymbol{\mu}\in\Delta_N} \{\sum_{j=1}^{N}\mu_j s_j - \sum_{j=1}^{N}\mu_j\log\mu_j\} = \max_{\boldsymbol{\mu}\in\Delta_N}\{\mathbf{E}_{\boldsymbol{\mu}}[s_j] + \mathbf{H}(\boldsymbol{\mu})\} \quad (5)$$

where the simplex set $\Delta_N = \{\mathbf{p}\in\mathbb{R}^N : p_j \geq 0, \sum_{j=1}^{N} p_j = 1\}$ *and the maximizer is attained at:*

$$\mu_j^* = \mathrm{P}(j;\mathbf{s}), \quad 1\leq j\leq N \quad (6)$$

Proof. We use the following equivalence:

$$\mathbf{E}_{\boldsymbol{\mu}}[s_j] + \mathbf{H}(\boldsymbol{\mu}) = -\sum_{j=1}^{N}\mu_j\log\frac{\mu_j}{\mathrm{P}(j;\mathbf{s})} + \log\sum_{j=1}^{N}\exp(s_j) = -D_{KL}(\boldsymbol{\mu}||\mathrm{P}) + A(\mathbf{s})$$

where $D_{KL}(\boldsymbol{\mu}||\mathrm{P})$ is the Kullback-Leibler (KL) divergence and note $D_{KL}(\boldsymbol{\mu}||P)\geq 0$ and $D_{KL}(P||P) = 0$. It follows that $\boldsymbol{\mu}^* = \arg\min_{\boldsymbol{\mu}\in\Delta_N} D_{KL}(\boldsymbol{\mu}||P) = P$. □

In view of Lemma 3.1, we arrive at the primal-dual form of ME:

$$\max_{\boldsymbol{\Theta}} \min_{\substack{\boldsymbol{\mu}_t\in\Delta_N \\ 1\leq t\leq M}} \sum_{t=1}^{M} \big(f_t(i_t;\boldsymbol{\Theta}) - \mathbf{E}_{\boldsymbol{\mu}_t}[f_t(j;\boldsymbol{\Theta})] - \mathbf{H}(\boldsymbol{\mu}_t)\big) \quad (7)$$

where $\boldsymbol{\mu}_t$ is the *dual distribution* for instance t.

Dual Distribution Clustering

Lemma 3.1 implies that $\boldsymbol{\mu}_t^*$ is determined by $f_t(j;\boldsymbol{\Theta})$. In less mathematical terms, similar instances choose similar items (in probabilities). As real-world data generally possesses a clustering structure instead of being randomly distributed, it is expected that the optimal dual distributions $\boldsymbol{\mu}_t^*$ for $t = 1,\ldots,M$ also form clusters. For the text classification example of venue prediction, if papers are grouped by topics, those in the same group should have similar chance of getting published at a particular venue; For learning word embedding, we anticipate contexts of similar semantics yield target word distributions that can be clustered together.

Therefore, it is worth exploring the cluster structure of dual distributions to reduce complexity. DCME rests on the idea of "approximation by clustering": By clustering the dual distributions into K groups, each $\boldsymbol{\mu}_t$ is assigned to a cluster $c_t\in\{1,\ldots,K\}$, and is then approximated by the corresponding *cluster center* $\boldsymbol{\alpha}_{c_t}\in\Delta_N$ which best represents the group. The optimization problem of DCME can thus be formulated as:

$$\text{DCME:} \quad \max_{\boldsymbol{\Theta}} \min_{\substack{\boldsymbol{\alpha}_k\in\Delta_N \\ 1\leq k\leq K}} \min_{\substack{1\leq c_t\leq K \\ 1\leq t\leq M}} \sum_{t=1}^{M} Q_t(\boldsymbol{\alpha}_{c_t};\boldsymbol{\Theta}) \quad (8)$$

$$\text{where} \quad Q_t(\boldsymbol{\alpha}_{c_t};\boldsymbol{\Theta}) = f_t(i_t;\boldsymbol{\Theta}) - \mathbf{E}_{\boldsymbol{\alpha}_{c_t}}[f_t(j;\boldsymbol{\Theta})] - \mathbf{H}(\boldsymbol{\alpha}_{c_t})\big)$$

Online-Offline Optimization

We employ Gauss-Seidel coordinate descent to solve (8). Three blocks of variables, namely, the model parameters $\boldsymbol{\Theta}$, the cluster centers $\{\boldsymbol{\alpha}_k\}$, and the instances' cluster assignments $\{c_t\}$, are successively updated while keeping others constant. In particular, we devise a hybrid online-offline algorithm which breaks the computational bottleneck and leads to a time complexity that only scales as $\mathcal{O}(DK)$, as opposed to $\mathcal{O}(DN)$ in conventional ME algorithms.

Updating cluster assignments (Online) DCME approximates $\boldsymbol{\mu}_t$ by $\boldsymbol{\alpha}_{c_t}$, and the cluster assignment c_t is solved by:

$$\arg\min_{1\leq k\leq K} -\mathbf{E}_{\boldsymbol{\alpha}_k}[f_t(j;\boldsymbol{\Theta})] - \mathbf{H}(\boldsymbol{\alpha}_k) \quad (9)$$

However, a naïve computation would cost $\mathcal{O}(DN + KN)$ time. It takes $\mathcal{O}(D)$ to evaluate $f_t(j;\boldsymbol{\Theta})$ for every item $1\leq j\leq N$[6]; For each cluster, another $\mathcal{O}(N)$ is required to calculate $\mathbf{E}_{\boldsymbol{\alpha}_k}[f_t(j;\boldsymbol{\Theta})]$ and $\mathbf{H}(\boldsymbol{\alpha}_k)$ by enumeration.

Fortunately, when the scoring function is linear in the feature/context vector, the cost can be reduced to $\mathcal{O}(DK)$ per instance. To see this, from (2) and (3) we have:

$$\text{Classification:} \quad \mathbf{E}_{\boldsymbol{\alpha}_k}[f_t(j;\mathbf{W})] = (\mathbf{W}\boldsymbol{\alpha}_k)^T\mathbf{x}_t \quad (10)$$

$$\text{Embedding:} \quad \mathbf{E}_{\boldsymbol{\alpha}_k}[f_t(j;\mathbf{V},\mathbf{H})] = (\mathbf{V}\boldsymbol{\alpha}_k)^T\bar{\boldsymbol{h}}_t \quad (11)$$

The trick we apply here *trades memory for time*: By storing $\mathbf{W}\boldsymbol{\alpha}_k, \mathbf{V}\boldsymbol{\alpha}_k$ and $\mathbf{H}(\boldsymbol{\alpha}_k)$ for K clusters in the offline update,

[6] In word embedding, one can compute the scoring function in $\mathcal{O}(D)$ time. Note that the *asymptotic* complexity of computing $\bar{\boldsymbol{h}}_t$ in *every* sliding windows is $\mathcal{O}(D)$ (independent of window size) with the sum $\sum_{-c\leq p\leq c}\mathbf{h}_{w_t,p}$ maintained by adding the new word and subtracting the past word.

it is merely a D-dimensional dot product to calculate (10) and (11), and therefore the cost to online update c_t by (9) is $\mathcal{O}(DK)$.

Updating cluster centers (Offline) We update the cluster center $\boldsymbol{\alpha}_k$ as well as the cached $\mathbf{W}\boldsymbol{\alpha}_k$, $\mathbf{V}\boldsymbol{\alpha}_k$ and $\mathbf{H}(\boldsymbol{\alpha}_k)$ only in the offline computation. Let $\mathcal{I}_k$ denote the index set of instances in the k-th cluster, $\boldsymbol{\alpha}_k$ satisfies:

$$\arg\min_{\boldsymbol{\alpha}\in\Delta_N} -\mathbf{E}_{\boldsymbol{\alpha}}\left[\frac{1}{|\mathcal{I}_k|}\sum_{t\in\mathcal{I}_k} f_t(j;\boldsymbol{\Theta})\right] - \mathbf{H}(\boldsymbol{\alpha}) \tag{12}$$

Invoking Lemma 3.1 again, (12) has the following closed-form solution:

$$\alpha_{k,j} = \frac{1}{Z}\exp\big(\frac{1}{|\mathcal{I}_k|}\sum_{t\in\mathcal{I}_k} f_t(j;\boldsymbol{\Theta})\big) \tag{13}$$

where a normalization term Z is applied to keep $\sum_{j=1}^{N}\alpha_{k,j} = 1$. By the linearity of $f_t(j;\boldsymbol{\Theta})$, we express (13) as:

$$\text{Classification:}\quad \alpha_{k,j} = \frac{1}{Z}\exp\Big(\mathbf{w}_j^T\frac{1}{|\mathcal{I}_k|}\sum_{t\in\mathcal{I}_k}\mathbf{x}_t\Big) \tag{14}$$

$$\text{Embedding:}\quad \alpha_{k,j} = \frac{1}{Z}\exp\Big(\mathbf{v}_j^T\frac{1}{|\mathcal{I}_k|}\sum_{t\in\mathcal{I}_k}\bar{\boldsymbol{h}}_t\Big) \tag{15}$$

which computes $\boldsymbol{\alpha}_k$ with $\mathcal{O}(D|\mathcal{I}_k| + DN)$ cost. In addition, it takes $\mathcal{O}(DN)$ and $\mathcal{O}(N)$ to update $\mathbf{W}\boldsymbol{\alpha}_k$, $\mathbf{V}\boldsymbol{\alpha}_k$ and $\mathbf{H}(\boldsymbol{\alpha}_k)$, respectively. If one only performs the offline update for cluster k until that the number of instances assigned to cluster k, $|\mathcal{I}_k| = \beta N$ for a constant β (1 for example), we can obtain an average time complexity of $\mathcal{O}(D)$ per instance.

Updating model parameters (Online/Offline) To optimize $\boldsymbol{\Theta}$ with subgradient descent:

$$\text{Classification:}\quad \frac{\partial Q_t}{\partial \mathbf{w}_j} = \underbrace{\mathbb{1}[i_t = j]\,\mathbf{x}_t}_{(a)} + \underbrace{(-\alpha_{c_t,j}\mathbf{x}_t)}_{(b)} \tag{16}$$

$$\text{Embedding:}\quad \frac{\partial Q_t}{\partial \mathbf{v}_j} = \overbrace{\mathbb{1}[i_t = j]\,\bar{\boldsymbol{h}}_t} + \overbrace{(-\alpha_{c_t,j}\bar{\boldsymbol{h}}_t)} \tag{17}$$

$$\frac{\partial Q_t}{\partial \mathbf{h}_{w_p^{(t)}}} = \frac{1}{2c}(\mathbf{v}_{i_t} - \mathbf{V}\boldsymbol{\alpha}_{c_t}) \tag{18}$$

$$\text{for all}\ -c \le p \le c,\ p \ne 0$$

where $\mathbb{1}[i_t = j]$ is the indicator function which evaluates to 1 when $i_t = j$ and 0 otherwise. In the following, we devise a hybrid *online-offline algorithm* which has an average expense of $\mathcal{O}(D)$ time per instance.

First, Term (a) in (16) and (17) only changes the model parameters associated with the correct item i_t, namely $\mathbf{w}_{i_t}$ and $\mathbf{v}_{i_t}$, and can be updated online in $\mathcal{O}(D)$ time. Similarly, for word embedding (18), $\partial Q_t/\partial\mathbf{H}$ can also be updated online with $\mathcal{O}(D)$ cost by keeping track of the sum of $\partial Q_t/\partial\mathbf{h}_w$ for all overlapping instances using a sliding window technique.

Second, Term (b) in (16) and (17) changes the model parameters of all N items. We make two crucial observations here: (1) Term (b) of different items share the *same* direction $-\mathbf{x}_t$ (or $-\bar{\boldsymbol{h}}_t$); and (2) The scale vector $\boldsymbol{\alpha}_{c_t}$ *only* depends on the cluster assignment c_t. Thus it is logical to perform offline update of Term (b). The computation for instances in $\mathcal{I}_k$ is performed together with the (offline) updating for the center of cluster k. In other words, the calculation of Term (b) for all items $1 \le j \le N$ and instances $t \in \mathcal{I}_k$ is triggered only when $|\mathcal{I}_k| = \beta N$. And such "lazy" computation yields an average $\mathcal{O}(D)$ expense per instance.

Tuning online/offline computation The overall complexity per instance is $\mathcal{O}(DK)$ time, which is appealing as it does not hinge on N. Nevertheless, an inherent limitation in learning is the delay of computing Term (b) until $|\mathcal{I}_k| \ge \beta N$ in (16) and (17) , especially for items with large values $\alpha_{k,j}$. A heuristic improvement we find effective in practice tunes the computation between online and offline updates. By sorting items (using a heap) with decreasing $\alpha_{k,j}$, Term (b) of the top Q items are updated online while the others are updated offline. The resulting average cost per instance becomes $\mathcal{O}(DK + DQ + \log Q)$. Computational efficiency is preferred with a small Q while the priority shifts to learning efficiency with a large Q.

4 DCME Algorithm

Overall Procedures

We summarize the learning procedure of DCME in Algorithm 1. DCME assigns each training instance t to a dual cluster c_t and performs the online model update. Once the size of a dual cluster reaches βN, an offline model update as well as the update of the dual cluster center are applied. Although the algorithm has a similar complexity as the sampling-based approaches such as noise contrastive estimation (NCE) and negative sampling (NS), DCME allows the *entire* model to learn from *every* training instance. In other words, the model parameters associated with all items get updated when a new training instance arrives, which yields superior performance over existing methods, as will be shown in the experimental study.

Algorithm 1: DCME algorithm

Input: M instances, a constant β, cluster number K, and top item number Q
Output: Model $\boldsymbol{\Theta}$
Initialize K clusters $\{\boldsymbol{\alpha}_k\}$;
while $\boldsymbol{\Theta}$ *is not optimal* **do**
- Select an index t from $\{1,\ldots,M\}$
- Find the cluster assignment c_t by (9)
- Perform online update of Term (a) (and Term (b) of the top Q items) in (16) (or (17)). For embedding, also update H by (18).
- Add t to $\mathcal{I}_{c_t}$;
  if $|\mathcal{I}_{c_t}| \ge \beta N$
  - Perform offline update of Term (b) in (16) (or (17)).
  - Update cluster center α_{c_t} and empty $\mathcal{I}_{c_t}$

end

Connection with K-means

So far, readers might have already been aware of the resemblance between the dual distribution clustering and the K-means algorithm. The following theorem formally proves their connection:

Theorem 4.1. *The dual distribution clustering in DCME is a generalized K-means algorithm using KL-divergence as the distance measurement in the simplex. Moreover, it converges as fast as K-means.*

Proof. Using Lemma 3.1, the dual clustering satisfies:

$$\min_{\substack{\boldsymbol{\alpha}_k \in \Delta_N, 1 \le k \le K \\ 1 \le c_t \le K, 1 \le t \le M}} \sum_{t=1}^{M} D_{KL}(\boldsymbol{\alpha}_{c_t} || P_t) \tag{19}$$

which minimizes the *within-cluster KL-divergence* between $\boldsymbol{\alpha}_{c_t}$ and P_t. It is the same minimization objective as K-means except that DCME measures the distance in the simplex space with KL-divergence[7]. To illustrate this, notice that the dual clustering proceeds by alternating between the following two steps (See Figure 1):

- Update $c_t = \arg\min_k D_{KL}(\boldsymbol{\alpha}_k || P_t)$, and t is assigned to the cluster whose center is nearest to p_t by kl-divergence.
- Update $\boldsymbol{\alpha}_k = \arg\min_{\boldsymbol{\alpha}} \sum_{t \in \mathcal{I}_k} D_{KL}(\boldsymbol{\alpha} || P_t)$ where the cluster center is found as the point in the simplex with the least within-cluster distance.

General convergence results for the subgradient methods can be applied. Specifically, the above two-step algorithm converges to the local minimum of the problem (19) as fast as the K-means algorithm (Bottou and Bengio 1995), □

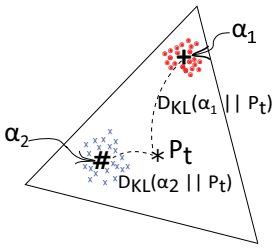


Figure 1: Dual Clustering in the Simplex with KL-divergence

It is easy to see that the least within-cluster distance (19) is also in fact the difference between the objective functions of DCME (8) and the primal ME (4) for a given Θ.

Connection with Dual ME

The DCME is reminiscent of the dual ME, and we show the following results in the classification setting:

[7]Technically, KL-divergence is not a true metric of distance.

Theorem 4.2. *The dual form of DCME in classification is:*

$$\max_{\substack{\boldsymbol{\alpha}_k \in \Delta_N, 1 \le k \le K \\ 1 \le c_t \le K, 1 \le t \le M}} \sum_{t=1}^{M} \mathbf{H}(\boldsymbol{\alpha}_{c_t}) \tag{20}$$

$$\textit{subject to} \quad \sum_{t=1}^{M} \mathbb{1}[i_t = j]\, \mathbf{x}_t = \sum_{t=1}^{M} \alpha_{c_t, j} \mathbf{x}_t, \;\; 1 \le j \le N$$

The proof is omitted because it is very similar to the derivation of dual ME. However, the dual form of DCME provides us with intuition of how DCME works: To approximate P_t, the cluster center is restricted to reproduce the observed statistics. Comparing it with the dual ME where $\boldsymbol{\mu}_t$ is in place of $\boldsymbol{\alpha}_{c_t}$, we see that the dual DCME has more restricted constraints. A limiting case that DCME becomes identical to ME is when $K = M$, i.e. each instance is a singleton cluster with the only member being itself.

5 Experiments

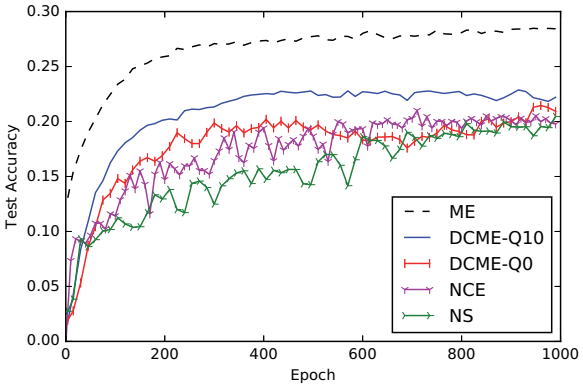
We conduct experiments on tasks of text classification and word embedding, evaluating the proposed DCME approach by examining its computational and learning efficiency. For comparison, we implement two sampling-based approaches, noise contrastive estimation (NCE) and negative sampling (NS), as well as the maximum likelihood estimation using gradient descent (GD). In order for DCME and the sampling-based approaches to have comparable training speed, we set both the cluster number K of DCME and the sampling number of NCE and NS to 20, and also control the interval between offline updates in DCME with $\beta = 1$. Two variants of DCME, DCME-Q0 and DCME-Q10, are developed, the latter of which applies the online/offline tuning with $Q = 10$. All the algorithms are run with 20 threads in parallel on a 64-bit Linux machine with the Intel Xeon 3.60GHz CPU (20 core). Our code is implemented in C and available for download at: https://github.com/dragonxlwang/dcme

Evaluation on Text Classification

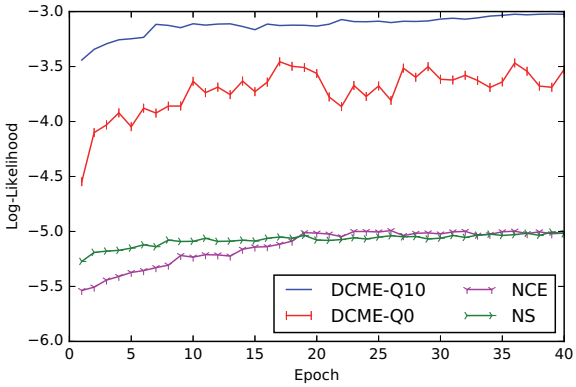
We employ the ME model to predict the publishing venue of research papers using the abstract. A public dataset ACM Digital Library is investigated. It has $162,460$ papers published at $1,236$ conferences. We hold out 10% of the documents for testing. Each paper is represented by the word count features of the top $30,000$ frequent words.

Figure 2a shows the learning curves of algorithms trained at each epoch, and Table 2c reports the training speed. It is clear that GD does not scale well to large number (thousands or more) of items. DCME is 17-20 times faster than GD while the ratio is around 26 for sampling-based approaches. But it does give an estimation about the upper-bound performance by leveraging the exact gradient information. The curve of GD converges in the least number of iterations while the test accuracy is the highest.

DCME, on the other hand, achieves a computational efficiency similar to that of the sampling-based approaches, but the accuracy is considerably higher. Particularly, Figure 2a validates that DCME benefits from tuning the computation between online and offline updates. When $Q = 10$, more



(a) Comparison of Test Accuracy for Classification Trained on ACM Digital Library Dataset



(b) Comparison of Log-Likelihood for Embedding Trained on NYT Dataset

DCME-Q10	DCME-Q0	NCE	NS	GD
9.50	7.94	6.21	6.16	165.91

(c) Time Cost (Second) per Epoch in Classification

DCME-Q10	DCME-Q0	NCE	NS
33.49	25.59	22.22	20.80

(d) Time Cost (Minute) per Epoch in Embedding

Figure 2: Performance on Text Classification and Word Embedding

model parameters are updated online and there is thus less delay than that of DCME-Q0. We also note that NCE and NS produce larger variances, which is expected due to their sampling nature.

Evaluation on Word Embedding

For the word embedding task, we explore the New York Times (NYT) corpus from the English Gigaword (Fifth Edition). It has a total of 1.35 billion words with 10.84 million unique terms. We retain the top 1 million frequent terms in the vocabulary. To assess the performance, a randomly sampled 1×10^{-4} of the text is withheld for testing. We train the word embeddings using CBOW with a context window size of 10 and embedding dimensionality of 100.

We plot the test set average log-likelihood of each epoch in Figure 2b, and report the time-per-epoch statistics in Table 2d. We do not evaluate GD in word embedding as it takes more than days to run one epoch. The time costs for other algorithms are similar. The results show that DCME remarkably outperforms NCE and NS. However, DCME-Q0 exhibits a large performance variance. One possible explanation is as follows. For N as large as 1 million, the interval between offline updates is so long that it creates two undesirable effects: (1) The delay results in a biased model which contributes to a large training error; (2) The offline computation changes the model drastically, as measured by the norm of the model difference, causing inconsistency when another thread accesses the model while the offline update is still in progress[8]. For DCME-Q10, minimal variance is observed. Indeed, it offers the best trade-off between learning and computational efficiency.

Model	Semantic	Syntactic	Overall
DCME-Q10	$\mathbf{-8.676}$	-8.648	$\mathbf{-8.654}$
DCME-Q0	-8.712	$\mathbf{-8.647}$	-8.663
NCE	-8.784	-8.782	-8.783
NS	-8.765	-8.679	-8.699

Table 1: Log-Likelihood on Semantic-Syntactic Word Relationship Dataset

To assess the quality of the trained embeddings, we use the word analogy task, which examines whether the embeddings learn the semantic/syntactic relationships of words. For instance, the question which word is similar to "small" in the same sense as "biggest" to "big" can be solved by predicting the target word with a context vector $\mathbf{h}_{biggest} - \mathbf{h}_{big} + \mathbf{h}_{small}$. We evaluate the trained word embeddings after 15 epochs. And the results on the Semantic-Syntactic Word Relationship test set (Mikolov et al. 2013) are summarized in Table 1, where the best performance is highlighted in bold. Again, it confirms that DCME achieves better model quality than sampling-based NCE and NS.

6 Conclusions

We propose a novel optimization method, Dual-Clustering Maximum Entropy (DCME), which solves the Maximum Entropy problem in its primal-dual form. Although it has a similar complexity as the sampling-based approaches, it allows the entire model to learn from every training instance, which we believe is the first algorithm that is efficient both in learning and computation. DCME exploits the

[8]The model parameters are shared by all threads and there is no mutex locks on writing to the model, which is a common practice for efficiency in implementations including word2vec (https://code.google.com/p/word2vec/) and ours.

dual clustering and approximates dual distributions by cluster centers. It maintains an affordable complexity using a hybrid online-offline optimization algorithm. Empirical studies demonstrate that DCME outperforms state-of-the-art algorithms such as NCE and NS in learning tasks with large numbers of items. A promising future research direction is to investigate the nonparametric mixture models for dual clustering. By taking advantages of probabilistic latent cluster assignments and learning the number of clusters from the data, we expect a better approximation for dual distributions.

Acknowledgment

This work is supported in part by the National Science Foundation under Grant Numbers CNS-1513939 and CNS-1408944.

References

Bengio, Y., and Senécal, J.-S. 2008. Adaptive importance sampling to accelerate training of a neural probabilistic language model. *IEEE Transactions on Neural Networks* 19(4):713–722.

Bengio, Y.; Ducharme, R.; Vincent, P.; and Jauvin, C. 2003. A neural probabilistic language model. *journal of machine learning research* 3(Feb):1137–1155.

Berger, A. L.; Pietra, V. J. D.; and Pietra, S. A. D. 1996. A maximum entropy approach to natural language processing. *Computational linguistics* 22(1):39–71.

Bottou, L., and Bengio, Y. 1995. Convergence properties of the k-means algorithms. In *Advances in Neural Information Processing Systems*, 585–592.

Choromanska, A.; Agarwal, A.; and Langford, J. 2013. Extreme multi class classification. In *NIPS Workshop: eXtreme Classification, submitted*.

Choromanska, A. E., and Langford, J. 2015. Logarithmic time online multiclass prediction. In *Advances in Neural Information Processing Systems*, 55–63.

Collins, M.; Globerson, A.; Koo, T.; Carreras, X.; and Bartlett, P. L. 2008. Exponentiated gradient algorithms for conditional random fields and max-margin markov networks. *Journal of Machine Learning Research* 9(Aug):1775–1822.

Collobert, R., and Weston, J. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, 160–167. ACM.

Darroch, J. N., and Ratcliff, D. 1972. Generalized iterative scaling for log-linear models. *The annals of mathematical statistics* 1470–1480.

Gao, J.; Andrew, G.; Johnson, M.; and Toutanova, K. 2007. A comparative study of parameter estimation methods for statistical natural language processing. In *ANNUAL MEETING-ASSOCIATION FOR COMPUTATIONAL LINGUISTICS*, volume 45, 824.

Gutmann, M., and Hyvärinen, A. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *AISTATS*, volume 1, 6.

Hiriart-Urruty, J.-B., and Lemaréchal, C. 1993. *Convex analysis and minimization algorithms II: Advanced theory and bundle methods*. Springer-Verlag, New York.

Huang, F.-L.; Hsieh, C.-J.; Chang, K.-W.; and Lin, C.-J. 2010. Iterative scaling and coordinate descent methods for maximum entropy models. *Journal of Machine Learning Research* 11(Feb):815–848.

Malouf, R. 2002. A comparison of algorithms for maximum entropy parameter estimation. In *proceedings of the 6th conference on Natural language learning-Volume 20*, 1–7. Association for Computational Linguistics.

Mikolov, T., and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*.

Mikolov, T.; Chen, K.; Corrado, G.; and Dean, J. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mnih, A., and Hinton, G. 2007. Three new graphical models for statistical language modelling. In *Proceedings of the 24th international conference on Machine learning*, 641–648. ACM.

Mnih, A., and Hinton, G. E. 2009. A scalable hierarchical distributed language model. In *Advances in neural information processing systems*, 1081–1088.

Mnih, A., and Teh, Y. W. 2012. A fast and simple algorithm for training neural probabilistic language models. *arXiv preprint arXiv:1206.6426*.

Morin, F., and Bengio, Y. 2005. Hierarchical probabilistic neural network language model. *AISTATS 2005* 246.

Nigam, K.; Lafferty, J.; and McCallum, A. 1999. Using maximum entropy for text classification. In *IJCAI-99 workshop on machine learning for information filtering*, volume 1, 61–67.

Ratnaparkhi, A. 1996. A maximum entropy model for part-of-speech tagging. Association for Computational Linguistics.

Tsuruoka, Y.; Tsujii, J.; and Ananiadou, S. 2009. Stochastic gradient descent training for l1-regularized log-linear models with cumulative penalty. In *Proceedings of the 47th Annual Meeting of the ACL*, 477–485. Association for Computational Linguistics.

Wainwright, M. J., and Jordan, M. I. 2008. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning* 1(1-2):1–305.

Yen, I. E.; Huang, X.; Ravikumar, P.; Zhong, K.; and Dhillon, I. S. 2016. Pd-sparse: A primal and dual sparse approach to extreme multiclass and multilabel classification. In *Proceedings of the 33nd International Conference on Machine Learning*, 3069–3077.

Yu, H.-F.; Huang, F.-L.; and Lin, C.-J. 2011. Dual coordinate descent methods for logistic regression and maximum entropy models. *Machine Learning* 85(1-2):41–75.

Yuan, G.-X.; Ho, C.-H.; and Lin, C.-J. 2012. Recent advances of large-scale linear classification. *Proceedings of the IEEE* 100(9):2584–2603.