Generalized Majorization-Minimization

Sobhan Naderi [1] **Kun He** [2] **Reza Aghajani** [3] **Stan Sclaroff** [4] **Pedro Felzenszwalb** [5]

Abstract

Non-convex optimization is ubiquitous in machine learning. Majorization-Minimization (MM) is a powerful iterative procedure for optimizing non-convex functions that works by optimizing a sequence of bounds on the function. In MM, the bound at each iteration is required to *touch* the objective function at the optimizer of the previous bound. We show that this touching constraint is unnecessary and overly restrictive. We generalize MM by relaxing this constraint, and propose a new optimization framework, named Generalized Majorization-Minimization (G-MM), that is more flexible. For instance, G-MM can incorporate application-specific biases into the optimization procedure without changing the objective function. We derive G-MM algorithms for several latent variable models and show empirically that they consistently outperform their MM counterparts in optimizing non-convex objectives. In particular, G-MM algorithms appear to be less sensitive to initialization.

1. Introduction

Non-convex optimization is ubiquitous in machine learning. For example, data clustering (MacQueen, 1967; Arthur & Vassilvitskii, 2007), training classifiers with latent variables (Yu & Joachims, 2009; Felzenszwalb et al., 2010; Pirsiavash & Ramanan, 2014; Azizpour et al., 2015), and training visual object detectors from weakly labeled data (Song et al., 2014; Rastegari et al., 2015; Ries et al., 2015) all lead to non-convex optimization problems.

Majorization-Minimization (MM) (Hunter et al., 2000) is an optimization framework for designing well-behaved optimization algorithms for non-convex functions. MM algorithms work by iteratively optimizing a sequence of easy-to-optimize surrogate functions that bound the objective. Two of the most successful instances of MM algorithms are Expectation-Maximization (EM) (Dempster et al., 1977) and the Concave-Convex Procedure (CCP) (Yuille & Rangarajan, 2003). However, both have a number of drawbacks in practice, such as sensitivity to initialization and lack of uncertainty modeling for latent variables. This has been noted in (Neal & Hinton, 1998; Felzenszwalb et al., 2010; Parizi et al., 2012; Kumar et al., 2012; Ping et al., 2014).

We propose a new procedure, *Generalized Majorization-Minimization (G-MM)*, for non-convex optimization. Our approach is inspired by MM, but we generalize the bound construction process to allow for a *set* of valid bounds to be used, while still maintaining algorithmic convergence. This generalization gives us more freedom in bound selection and can be used to design better optimization algorithms.

In training latent variable models and in clustering problems, MM algorithms such as CCP and k-means are known to be sensitive to the initial values of the latent variables or cluster memberships. We refer to this problem as *stickiness* of the algorithm to the initial latent values. Our experimental results show that G-MM leads to methods that tend to be less sticky to initialization. We demonstrate the benefit of using G-MM on multiple problems, including k-means clustering and applications of Latent Structural SVMs to image classification with latent variables.

1.1. Related Work

One of the most popular and well studied iterative methods for non-convex optimization is the EM algorithm (Dempster et al., 1977). EM is best understood in the context of maximum likelihood estimation in the presence of missing data, or *latent variables*. EM is a bound optimization algorithm: in each E-step, a lower bound on the likelihood is constructed, and the M-step maximizes this bound.

Countless efforts have been made to extend the EM algorithm since its introduction. In (Neal & Hinton, 1998) it is shown that, while both steps in EM involve optimizing some functions, it is not necessary to fully optimize the functions in each step; in fact, each step only needs to "make progress". This relaxation can potentially avoid sharp local minima and even speed up convergence.

* The paper was written when Kun He was at Boston University. [1]Google Research [2]Facebook Reality Labs [3]University of California San Diego [4]Boston University [5]Brown University. Correspondence to: Sobhan Naderi <sobhan@google.com>.

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

The Majorization-Minimization (MM) framework (Hunter et al., 2000) generalizes EM by optimizing a sequence of surrogate functions (bounds) on the original objective function. The Concave-Convex Procedure (CCP) (Yuille & Rangarajan, 2003) is a widely-used instance of MM where the surrogate function is obtained by *linearizing* the concave part of the objective. Many successful learning algorithms employ CCP, *e.g.* the Latent SVM (Felzenszwalb et al., 2010). Other instances of MM algorithms include iterative scaling (Pietra et al., 1997), and non-negative matrix factorization (Lee & Seung, 1999). Another related line of research concerns the Difference-of-Convex (DC) programming (Tao, 1997), which can be shown to reduce to CCP under certain conditions. Convergence properties of such general "bound optimization" algorithms have been discussed in (Salakhutdinov et al., 2002).

Despite widespread success, MM (and CCP in particular) has a number of drawbacks, some of which have motivated our work. In practice, CCP often exhibits *stickiness* to initialization, which necessitates expensive initialization or multiple trials (Parizi et al., 2012; Song et al., 2014; Cinbis et al., 2016). In optimizing latent variable models, CCP lacks the ability to incorporate application-specific information without making modifications to the objective function, such as prior knowledge or side information (Xing et al., 2002; Yu, 2012), latent variable uncertainty (Kumar et al., 2012; Ping et al., 2014), and posterior regularization (Ganchev et al., 2010). Our framework addresses these drawbacks. Our key observation is that we can relax the constraint enforced by MM that requires the bounds to touch the objective function, and this relaxation gives us the ability to better avoid sensitivity to initialization, and to incorporate side information.

A closely related work to ours is the "pseudo-bound" optimization framework by (Tang et al., 2014). It generalizes CCP using bounds that may intersect the objective function. In contrast, our framework uses valid bounds, but only relaxes the touching requirement. Also, the pseudo-bound optimization framework is specific to binary energies in MRFs (although, it was recently generalized to multi-label energies in (Tang et al., 2019)), and it restricts the form of surrogate functions to parametric max-flow.

The generalized variants of EM proposed and analyzed by (Neal & Hinton, 1998) and (Gunawardana & Byrne, 2005) are related to our work when we restrict our attention to probabilistic models and the EM algorithm. EM can be viewed as a bound optimization procedure where the likelihood function involves both the model parameters θ and a distribution q over the latent variables, denoted by $F(\theta, q)$. Choosing q to be the posterior leads to a lower bound on F that is tight at the current estimate of θ. Generalized versions of EM, such as those given by (Neal & Hinton, 1998), use distributions other than the posterior in an alternating optimization of F. This fits into our framework, as we use the exact same objective function, and only changes the bound construction step (which amounts to picking the distribution q in EM). We propose both *stochastic* and *deterministic* strategies for bound construction, and demonstrate that they lead to higher quality solutions and less sensitivity to initialization than other EM-like methods.

2. Proposed Optimization Framework

We consider minimization of functions that are bounded from below. The extension to maximization is trivial. Let $F(w) : \mathbb{R}^d \to \mathbb{R}$ be a lower-bounded function that we wish to minimize. We propose an iterative procedure that generates a sequence of solutions $w_1, w_2, \ldots$ until it converges. The solution at iteration $t \geq 1$ is obtained by minimizing an upper bound $b_t(w)$ to the objective function *i.e.* $w_t = \operatorname{argmin}_w b_t(w)$. The bound at iteration t is chosen from a set of "valid" bounds $\mathcal{B}_t$ (see Figure 1). In practice, we take the members of $\mathcal{B}_t$ from a family $\mathcal{F}$ of functions that upper-bound F and can be optimized efficiently, such as quadratic functions, or quadratic functions with linear constraints. $\mathcal{F}$ must be rich enough so that $\mathcal{B}_t$ is never empty. Algorithm 1 gives the outline of the approach.

This general scheme is used in both MM and G-MM. However, as we shall see in the rest of this section, MM and G-MM have key differences in the way they measure progress and the way they construct new bounds.

2.1. Progress Measure

MM measures progress with respect to the objective values. To guarantee progress over time MM requires that the bound at iteration t must touch the objective function at the previous solution, leading to the following constraint:

$$\textbf{MM constraint:} \qquad b_t(w_{t-1}) = F(w_{t-1}). \tag{1}$$

This touching constraint, together with the fact that w_t minimizes b_t leads to $F(w_t) \leq F(w_{t-1})$. That is, the value of the objective function is non-increasing over time. However, it can make it hard to avoid local minima, and eliminates the possibility of using bounds that do not touch the objective function but may have other desirable properties.

In G-MM, we measure progress with respect to the bound values. It allows us to relax the touching constraint of MM, stated in (1), and require instead that,

$$\textbf{G-MM constraints:} \quad \begin{cases} b_1(w_0) = F(w_0) \\ b_t(w_{t-1}) \leq b_{t-1}(w_{t-1}). \end{cases} \tag{2}$$

Note that the G-MM constraints are weaker than MM: since b_{t-1} is an upper bound on F, (1) implies (2). While MM

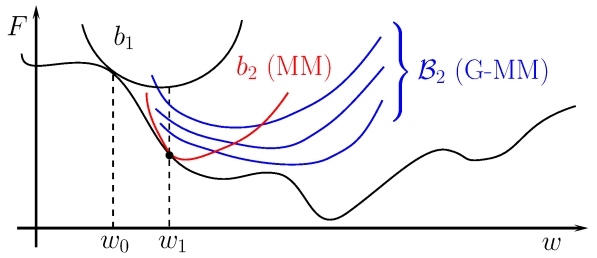


Figure 1. Optimization of F using MM (red) and G-MM (blue). In MM the bound b_2 has to touch F at w_1. In G-MM we only require that b_2 be below b_1 at w_1, leading to several choices $\mathcal{B}_2$.

Algorithm 1 G-MM optimization

input w_0, η, ϵ
1: $v_0 := F(w_0)$
2: **for** $t := 1, 2, \ldots$ **do**
3: select $b_t \in \mathcal{B}_t = \mathcal{B}(w_{t-1}, v_{t-1})$ as in (3)
4: $w_t := \operatorname{argmin}_w b_t(w)$
5: $d_t := b_t(w_t) - F(w_t)$
6: $v_t := b_t(w_t) - \eta d_t$
7: **if** $d_t < \epsilon$ **break**
8: **end for**
output w_t

constraint implies that the sequence $\{F(w_t)\}_t$ is decreasing, G-MM only requires $\{b_t(w_t)\}_t$ to be decreasing.

2.2. Bound Construction

This section describes line 3 of Algorithm 1. To construct a bound at iteration t, G-MM considers a "valid" subset of upper bounds $\mathcal{B}_t \subseteq \mathcal{F}$. To guarantee convergence, we restrict $\mathcal{B}_t$ to bounds that are below a threshold v_{t-1} at the previous solution w_{t-1}:

$$\mathcal{B}_t = \mathcal{B}(w_{t-1}, v_{t-1})$$
$$\mathcal{B}(w, v) = \{b \in \mathcal{F} \mid b(w) \leq v\}. \tag{3}$$

Initially, we set $v_0 = F(w_0)$ to ensure that the first bound *touches* F. For $t \geq 1$, we set $v_t = \eta F(w_t) + (1-\eta) b_t(w_t)$ for some hyperparameter $\eta \in (0, 1]$, which we call the *progress coefficient*. This guarantees making at least ηd_t progress, where $d_t = b_t(w_t) - F(w_t)$ is the gap between the bound and the true objective value at w_t. Small values of η allow for gradual exploratory progress while large values of η greedily select bounds that guarantee immediate progress. When $\eta = 1$ all valid bounds touch F at w_{t-1}, corresponding to the MM requirement. Note that all the bounds $b \in \mathcal{B}_t$ satisfy (2).

We consider two scenarios for selecting a bound from $\mathcal{B}_t$. In the first scenario we define a bias function $g : \mathcal{B}_t \times \mathbb{R}^d \to \mathbb{R}$ that takes a bound $b \in \mathcal{B}_t$ and a current solution $w \in \mathbb{R}^d$ and returns a scalar indicating the goodness of the bound. We then select the bound with the largest bias value *i.e.* $b_t = \operatorname{argmax}_{b \in \mathcal{B}_t} g(b, w_{t-1})$. In the second scenario we propose to choose a bound from $\mathcal{B}_t$ at random. Thus, we have both a deterministic (the 1^{st} scenario) and a stochastic (the 2^{nd} scenario) bound construction mechanism.

2.3. Generalization over MM

MM algorithms, such as EM and CCP, are special cases of G-MM that use a specific bias function $g(b, w) = -b(w)$. Note that $b_t = \operatorname{argmax}_{b \in \mathcal{B}_t} -b(w_{t-1})$ touches F at w_{t-1}, assuming $\mathcal{B}_t$ includes such a bound. Also, by definition, b_t makes maximum progress with respect to the previous bound value $b_{t-1}(w_{t-1})$. By choosing bounds that maximize progress, MM algorithms tend to rapidly converge to a nearby local minimum. For instance, at iteration t, the CCP bound for latent SVMs is obtained by fixing the latent variables in the concave part of F to the maximizers of the score of the model from iteration $t-1$, making w_t attracted to w_{t-1}. Similarly, in the E-step of EM, the posterior distribution of the latent variables is computed with respect to w_{t-1} and, in the M-step, the model is updated to "match" these fixed posteriors. This explains one reason why MM algorithms are observed to be sticky to initialization.

G-MM offers a more flexible bound construction scheme than MM. In Section 5 we show empirically that picking a valid bound randomly, *i.e.* $b_t \sim U(\mathcal{B}_t)$, is less sensitive to initialization and leads to better results compared to CCP and EM. We also show that using good bias functions can further improve performance of the learned models.

3. Convergence of G-MM

We show that, under general assumptions, the sequence $\{w_t\}_t$ of bound minimizers converges, and Algorithm 1 stops after finite steps (Theorem 1). With additional assumptions, we also prove that the limit of this sequence is a stationary point of F (Theorem 2). We believe stronger convergence properties depend on the structure of the function F, the family of the bounds $\mathcal{F}$, and the bound selection strategy, and should be investigated separately for each specific problem. We prove Theorems 1 and 2 in the supplementary material.

Theorem 1. *Suppose F is a lower-bounded, continuous function with compact sublevel sets, and $\mathcal{F}$ is a family of lower-bounded and m-strongly convex functions. Then the sequence of minimizers $\{w_t\}_t$ converges* (i.e. *the limit $w^\dagger = \lim_{t\to\infty} w_t$ exists), and the gap $d_t = b_t(w_t) - F(w_t)$ converges to 0.*

Theorem 2. *Suppose the assumptions in Theorem 1 holds. In addition, let F be continuously differentiable, and $\mathcal{F}$ be a family of smooth functions such that $\forall b \in \mathcal{F}, MI \succeq \nabla^2 b(w) \succeq mI$, for some $m, M \in (0, \infty)$, where I is the identity matrix. Then $\nabla F(w^\dagger) = 0$, namely, G-MM converges to a stationary point of F.*

4. Derived Optimization Algorithms

G-MM is applicable to a variety of non-convex optimization problems, but for simplicity and ease of exposition, we primarily focus on *latent variable models* where bound construction naturally corresponds to imputing latent variables in the model. In this section we derive G-MM algorithms for two widely used families of models, namely, k-means and Latent Structural SVM. Note that the training objectives of these two problems are non-differentiable and, therefore, Theorem 2 does not apply to them. However, note that the theorem only gives a *sufficient* condition but is not necessary for convergence of the algorithms. In fact, in all our experiments we observe that G-MM converges (*e.g.* see Table 4), and these algorithms significantly outperform their MM counterparts (see Section 5).

4.1. k-means Clustering

Let $\{x_1, \ldots, x_n\}$ denote n points and $w = (\mu_1, \ldots, \mu_k)$ denote k cluster centers. We assign a cluster to each point, denoted by $z_i \in \{1, \ldots, k\}, \forall i \in \{1, \ldots, n\}$. The objective function in k-means clustering is defined as follows,

$$F(w) = \sum_{i=1}^{n} \min_{z_i \in \{1,\ldots,k\}} ||x_i - \mu_{z_i}||^2, \quad w = (\mu_1, \ldots, \mu_k). \tag{4}$$

Bound construction: We obtain a convex upper bound on F by fixing the latent variables $(z_1, \ldots, z_n)$ to certain values instead of minimizing over these variables. Such bounds are quadratic convex functions of $(\mu_1, \ldots, \mu_k)$,

$$\mathcal{F} = \left\{ \sum_{i=1}^{n} ||x_i - \mu_{z_i}||^2 \;\middle|\; \forall i,\ z_i \in \{1, \ldots, k\} \right\}. \tag{5}$$

The k-means algorithm is an instance of MM methods. The algorithm repeatedly assigns each example to its nearest center to construct a bound, and then updates the centers by optimizing the bound. We can set $g(b, w) = -b(w)$ in G-MM to obtain the k-means algorithm. We can also define g differently to obtain a G-MM algorithm that exhibits other desired properties. For instance, a common issue in clustering is cluster starvation. One can discourage starvation by defining g accordingly.

We select a bound from $\mathcal{B}_t$ uniformly at random by starting from an initial configuration $z = (z_1, \ldots, z_n)$ that corresponds to a valid bound in $\mathcal{B}_t$ (*e.g.* k-means solution). We then do a random walk on a graph whose nodes are latent configurations defining valid bounds. The neighbors of a latent configuration z are other latent configurations that can be obtained by changing the value of one of the n latent variables in z.

Bound optimization: Optimization of $b \in \mathcal{F}$ can be done in closed form by setting μ_j to be the mean of all examples assigned to cluster j:

$$\mu_j = \frac{\sum_{i \in I_j} x_i}{|I_j|}, \quad I_j = \{1 \leq i \leq n \mid z_i = j\}. \tag{6}$$

4.2. Latent Structural SVM

A Latent Structural SVM (LS-SVM) (Yu & Joachims, 2009) defines a structured output classifier with latent variables. It extends the Structural SVM (Joachims et al., 2009) by introducing latent variables.

Let $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ denote a set of labeled examples with $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$. We assume that each example x_i has an associated latent value $z_i \in \mathcal{Z}$. Let $\phi(x, y, z) : \mathcal{X} \times \mathcal{Y} \times \mathcal{Z} \to \mathbb{R}^d$ denote a *feature map*. A vector $w \in \mathbb{R}^d$ defines a classifier $\hat{y} : \mathcal{X} \to \mathcal{Y}$,

$$\hat{y}(x) = \operatorname*{argmax}_y (\max_z w \cdot \phi(x, y, z)). \tag{7}$$

The LS-SVM training objective is defined as follows,

$$F(w) = \frac{\lambda}{2}||w||^2 + \frac{1}{n}\sum_{i=1}^{n}\Big(\max_{y,z}(w \cdot \phi(x_i, y, z) + \Delta(y, y_i)) - \max_z w \cdot \phi(x_i, y_i, z)\Big), \tag{8}$$

where λ is a hyper-parameter that controls regularization and $\Delta(y, y_i)$ is a non-negative loss function that penalizes the prediction y when the ground truth label is y_i.

Bound construction: As in the case of k-means, a convex upper bound on the LS-SVM objective can be obtained by imputing latent variables. Specifically, for each example x_i, we fix $z_i \in \mathcal{Z}$, and replace the maximization in the last term of the objective with a linear function $w \cdot \phi(x_i, y_i, z_i)$. This forms a family of convex piecewise quadratic bounds,

$$\mathcal{F} = \left\{ \frac{\lambda}{2}||w||^2 + \frac{1}{n}\sum_{i=1}^{n}\max_{y,z}(w \cdot \phi(x_i, y, z) + \Delta(y, y_i)) - w \cdot \phi(x_i, y_i, z_i) \;\middle|\; \forall i, z_i \in \mathcal{Z} \right\}. \tag{9}$$

The CCP algorithm for LS-SVM selects the bound b_t defined by $z_i^t = \operatorname{argmax}_{z_i} w_{t-1} \cdot \phi(x_i, y_i, z_i)$. This particular choice is a special case of G-MM when $g(b, w) = -b(w)$.

To generate random bounds from $\mathcal{B}_t$ we use the same approach as in the case of k-means clustering. We perform a random walk in a graph where the nodes are latent configurations leading to valid bounds, and the edges connect latent configurations that differ in a single latent variable.

Bound optimization: Optimization of $b \in \mathcal{F}$ corresponds to a convex quadratic program and can be solved using different techniques, including gradient based methods (*e.g.* SGD) and the cutting-plane method (Joachims et al., 2009). We use the cutting-plane method in our experiments.

4.3. Bias Function for Multi-fold MIL

The multi-fold MIL algorithm (Cinbis et al., 2016) was introduced for training latent SVMs for weakly supervised object localization, to deal with stickiness issues in training with CCP. It modifies how latent variables are updated during training. (Cinbis et al., 2016) divide the training set into K folds, and updates the latent variables in each fold using a model trained on the other $K-1$ folds. This algorithm does not have a formal convergence guarantee. By defining a suitable bias function, we can derive a G-MM algorithm that mimics the behavior of multi-fold MIL, and yet, is convergent.

Consider training an LS-SVM. Let $S = \{1, \dots, n\}$ and $I \subseteq S$ denote a subset of S. Also, let $z_i \in \mathcal{Z}$ denote the latent variable associated to training example (x_i, y_i), and z_I^t denote the fixed latent values for training examples indexed by I in iteration t. We denote the model trained on $\{(x_i, y_i) | i \in I\}$ with latent variables fixed to z_I^t in the last maximization of (8) by $w(I, z_I^t)$.

We assume access to a loss function $\ell(w, x, y, z)$. For example, for the binary latent SVM where $y \in \{-1, 1\}$, ℓ is the hinge loss: $\ell(w, x, y, z) = \max\{0, 1 - y\, w \cdot \phi(x, z)\}$.

We first consider the Leave-One-Out (LOO) setting, *i.e.* $K = n$, and call the algorithm of (Cinbis et al., 2016) LOO-MIL in this case. The update rule of LOO-MIL in iteration t is to set

$$z_i^t = \operatorname*{argmin}_{z \in \mathcal{Z}} \ \ell\left(w(S \backslash i, z_{S \backslash i}^{t-1}), x_i, y_i, z\right), \quad \forall i \in S. \quad (10)$$

After updating the latent values for all training examples, the model w is retrained by optimizing the resulting bound.

Now let us construct a G-MM bias function that mimics the behavior of LOO-MIL. Recall from (9) that each bound $b \in \mathcal{B}_t$ is associated with a joint latent configuration $z(b) = (z_1, \dots, z_n)$. We use the following bias function:

$$g(b, w) = -\sum_{i \in S} \ell\left(w(S \backslash i, z_{S \backslash i}^{t-1}), x_i, y_i, z_i\right). \quad (11)$$

Note that picking a bound according to (11) is equivalent to the LOO-MIL update rule of (10) except that in (11) only *valid* bounds are considered; that is bounds that make at least η-progress.

For the general multi-fold case (*i.e.* $K < n$), the bias function can be derived similarly.

5. Experiments

We evaluate G-MM and MM algorithms on k-means clustering and LS-SVM training on various datasets. Recall from (3) that the progress coefficient η defines the set of valid bounds $\mathcal{B}_t$ in each step. CCP and standard k-means bounds correspond to setting $\eta = 1$, thus taking maximally large steps towards a local minimum of the true objective.

5.1. *k*-means Clustering

We conduct experiments on four clustering datasets: Norm-25 (Arthur & Vassilvitskii, 2007), D31 (Veenman et al., 2002), Cloud (Arthur & Vassilvitskii, 2007), and GMM-200. See the references for details about the datasets. GMM-200 was created by us and has 10000 samples taken from a 2-D Gaussian mixture model with 200 mixtures (50 samples per each component). All the mixture components have unit variance and their means are placed on a 70×70 square uniformly at random, while making sure the distance between any two centers is at least 2.5.

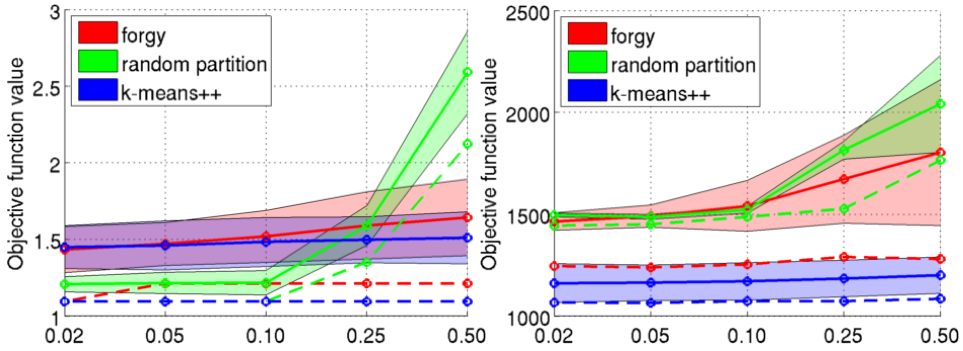
We compare results from three different initializations: **forgy** selects k training examples uniformly at random without replacement to define initial cluster centers, **random partition** assigns training samples to cluster centers randomly, and ***k*-means++** uses the algorithm in (Arthur & Vassilvitskii, 2007). In each experiment we run the algorithm 50 times and report the mean, standard deviation, and the best objective value (4). Table 1 shows the results using k-means (hard-EM) and G-MM. We note that the variance of the solutions found by G-MM is typically smaller than k-means. Moreover, the best and the average solutions found by G-MM are always better than (or the same as) those found by k-means. This trend generalizes over different initialization schemes as well as different datasets.

Although *random partition* seems to be a bad initialization for k-means on all datasets, G-MM recovers from it. In fact, on D31 and GMM-200 datasets, G-MM initialized by *random partition* performs better than when it is initialized by other methods (including k-means++). Also, the variance of the best solutions (across different initialization methods) in G-MM is smaller than that of k-means. These suggest that the G-MM optimization is less sticky to initialization than k-means.

Figure 2 shows the effect of the progress coefficient on the quality of the solution found by G-MM. Different initialization schemes are color coded. The solid line indicates the average objective over 50 iterations, the shaded area covers one standard deviation from the average, and the dashed line indicates the best solution over the 50 trials. Smaller progress coefficients allow for more extensive exploration, and hence, smaller variance in the quality of the solutions. On the other hand, when the progress coefficient is large G-MM is more sensitive to initialization (*i.e.* is more sticky) and, thus, the quality of the solutions over multiple runs is more diverse. However, despite the greater diversity, the best solution is worse when the progress coefficient is large. G-MM reduces to k-means if we set the progress coefficient to 1 (*i.e.* the largest possible value).

dataset	k	opt. method	forgy		random partition		*k*-means++	
			avg ± std	best	avg ± std	best	avg ± std	best
Norm-25	25	k-means	$1.9e5 \pm 2e5$	$7.0e4$	$5.8e5 \pm 3e5$	$2.2e5$	$5.3e3 \pm 9e3$	1.5
		G-MM	$9.7e3 \pm 1e4$	1.5	$2.0e4 \pm 0$	$2.0e4$	$4.5e3 \pm 8e3$	1.5
D31	31	k-means	1.69 ± 0.03	1.21	52.61 ± 47.06	4.00	1.55 ± 0.17	1.10
		G-MM	1.43 ± 0.15	1.10	1.21 ± 0.05	1.10	1.45 ± 0.14	1.10
Cloud	50	k-means	1929 ± 429	1293	44453 ± 88341	3026	1237 ± 92	1117
		G-MM	1465 ± 43	1246	1470 ± 8	1444	1162 ± 95	1067
GMM-200	200	k-means	2.25 ± 0.10	2.07	11.20 ± 0.63	9.77	2.12 ± 0.07	1.99
		G-MM	2.04 ± 0.09	1.90	1.85 ± 0.02	1.80	1.98 ± 0.06	1.89

Table 1. Comparison of G-MM and k-means on four clustering datasets and three initialization methods; **forgy** initializes cluster centers to random examples, **random partition** assigns each data point to a random cluster center, and ***k*-means++** implements the algorithm from (Arthur & Vassilvitskii, 2007). The mean, standard deviation, and best objective values out of 50 random trials are reported. k-means and G-MM use the exact same initialization in each trial. G-MM consistently converges to better solutions.



(a) D31 (b) Cloud

Figure 2. Effect of the progress coefficient η (x-axis) on the quality of the solutions found by G-MM (y-axis) on two clustering datasets. The quality is measured by the objective function in (4). Lower values are better. The average (solid line), the best (dashed line), and the variance (shaded area) over 50 trials are shown in the plots and different initializations are color coded.

5.2. Latent Structural SVM for Image Classification and Object Detection

We consider the problem of training an LS-SVM classifier on the mammals dataset (Heitz et al., 2009). The dataset contains images of six mammal categories with image-level annotation. Locations of the objects in these images are not provided, and therefore, treated as latent variables in the model. Specifically, let x be an image and y be a class label ($y \in \{1, \ldots, 6\}$ in this case), and let z be the latent location of the object in the image. We define $\phi(x, y, z)$ to be a feature function with 6 blocks; one block for each category. It extracts features from location z of image x and places them in the y-th block of the output and fills the rest with zero. We use the following multi-class classification rule:

$$y(x) = \underset{y,z}{\operatorname{argmax}}\, w \cdot \phi(x, y, z), \;\; w = (w_1, \ldots, w_6). \quad (12)$$

In this experiment we use a setup similar to that in (Kumar et al., 2012): we use Histogram of Oriented Gradients (HOG) for the image feature ϕ, and the 0-1 classification loss for Δ. We set $\lambda = 0.4$ in (8). We report 5-fold cross-validation performance. Three initialization strategies are considered for the latent object locations: *image center*, *top-left corner*, and *random locations*. The first is a reasonable initialization since most objects are at the center in this dataset; the second initialization strategy is somewhat adversarial.

We try a stochastic as well as a deterministic bound construction method. For the stochastic method, in each iteration t we uniformly sample a subset of examples S_t from the training set, and update their latent variables using $z_i^t = \operatorname{argmax}_{z_i} w_{t-1} \cdot \phi(x_i, y_i, z_i)$. Other latent variables are kept the same as the previous iteration. We increase the size of S_t across iterations.

For the deterministic method, we use the bias function that we described in Section 4.3. This is inspired by the multi-fold MIL idea (Cinbis et al., 2016) and is shown to reduce stickiness to initialization, especially in high dimensions. We set the number of folds to $K = 10$ in our experiments.

Table 2 shows results on the mammals dataset. Both variants of G-MM consistently outperform CCP in terms of training objective and test error. We observed that CCP rarely updates the latent locations, under all initializations. On the other hand, both variants of G-MM significantly alter the latent locations, thereby avoiding the local minima close to the initialization. Figure 3 visualizes this for *top-left* initialization. Since objects rarely occur at the top-left corner in the mammals dataset, a good model is expected to significantly update the latent locations. Averaged over five cross-validation folds, about 90% of the latent variables were updated in G-MM after training whereas this measure was 2.4% for CCP. This is consistent with the better training objectives and test errors of G-MM.

Opt. Method	center		top-left		random	
	objective	test error	objective	test error	objective	test error
CCP	1.21 ± 0.03	22.9 ± 9.7	1.35 ± 0.03	42.5 ± 4.6	1.47 ± 0.03	31.8 ± 2.6
G-MM random	0.79 ± 0.03	17.5 ± 3.9	0.91 ± 0.02	31.4 ± 10.1	0.85 ± 0.03	19.6 ± 9.2
G-MM biased	0.64 ± 0.02	16.8 ± 3.2	0.70 ± 0.02	18.9 ± 5.0	0.65 ± 0.02	14.6 ± 5.4

Table 2. LS-SVM results on the mammals dataset (Heitz et al., 2009). We report the mean and standard deviation of the training objective (8) and test error over five folds. Three strategies for initializing latent object locations are tried: image center, top-left corner, and random location. "G-MM random" uses random bounds, and "G-MM bias" uses a bias function inspired by multi-fold MIL (Cinbis et al., 2016). Both variants consistently and significantly outperform the CCP baseline.

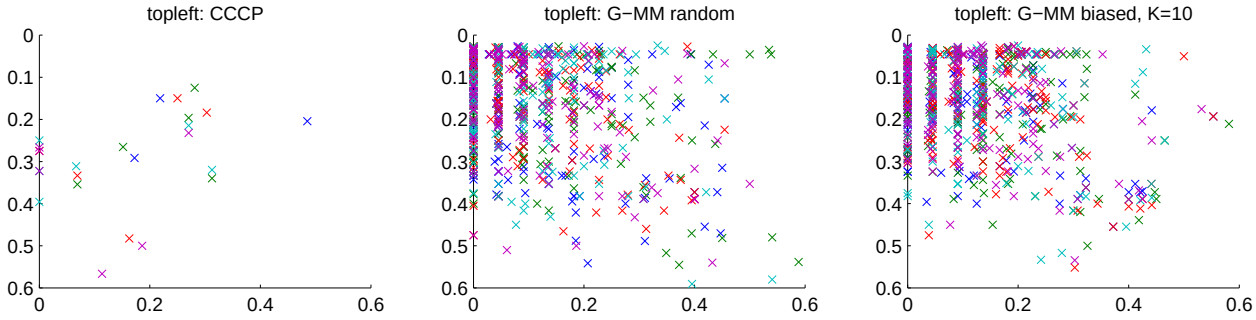


Figure 3. Latent location changes after learning, in relative image coordinates, for all five cross-validation folds, for the *top-left* initialization on the mammals dataset. Left to right: CCP, "G-MM random", "G-MM biased" (K=10). Each cross represents a training image; cross-validation folds are color coded differently. Averaged over five folds, CCP only alters 2.4% of all latent locations, leading to very bad performance. "G-MM random" and "G-MM biased" alter 86.2% and 93.6% on average, respectively, and perform much better.

5.3. Latent Structural SVM for Scene Recognition

We implement the reconfigurable model of (Parizi et al., 2012) (called RBoW) to do scene classification on MIT-Indoor dataset (Quattoni & Torralba, 2009), which has images from 67 indoor scene categories. We segment each image into a 10×10 regular grid and treat the grid cells as image regions. We train a model with 200 shared parts. Any part can be used to describe the data in a region. We use the activations of the 4096 neurons at the penultimate layer of the pre-trained hybrid ConvNet of (Zhou et al., 2014) to extract features from image regions and use PCA to reduce the dimensionality of the features to 240.

The RBoW model is an instance of LS-SVM models. The latent variables are the assignments of parts to image regions and the output structure is the multi-valued category label predictions. LS-SVMs are known to be sensitive to initialization (*a.k.a.* the stickiness issue). To cope with this issue (Parizi et al., 2012) uses a generative version of the model to initialize the training of the discriminative model. Generative models are typically less sticky but perform worse in practice. To validate the hypothesis regarding stickiness of LS-SVMs we train models with several initialization strategies.

Initializing training entails the assignment of parts to image regions *i.e.* setting z_i's in (9) to define the first bound. To this end we first discover 200 parts that capture discriminative features in the training data. We then run graph cut on each training image to obtain part assignments to image regions. Each cell in the 10×10 image grid is a node in the graph. Two nodes in the graph are connected if their corresponding cells in the image grid are next to each other. Unary terms in the graph cut are the dot product scores between the feature vector extracted from an image region and a part filter plus the corresponding region-to-part assignment score. Pairwise terms in the graph cut implement a *Potts* model that encourages coherent labelings. Specifically, the penalty of labeling two neighboring nodes differently is λ and it is zero otherwise. λ controls the coherency of the initial assignments. We experiment using $\lambda \in \{0, 0.25, 0.5, 1\}$. We also experiment with random initialization, which corresponds to assigning z_i's randomly. This is the simplest form of initialization and does not require discovering initial part filters.

We do G-MM optimization using both random and biased bounds. For the latter we use a bias function $g(b, w)$ that measures coherence of the labeling from which the bound was constructed. Recall from (9) that each bound in $b \in B_t$ corresponds to a labeling of the image regions. We denote the labeling corresponding to the bound b by $z(b) = (z_1, \ldots, z_n)$ where $z_i = (z_{i,1}, \ldots, z_{i,100})$ specifies part assignments for all the 100 regions in the i-th image. Also, let $E(z_i)$ denote a function that measures coherence of the labeling z_i. In fact, $E(z_i)$ is the Potts energy function on a graph whose nodes are $z_{i,1}, \ldots, z_{i,100}$. The graph respects a 4-connected neighborhood system (recall that $z_{i,r}$ corresponds to the r-th cell in the 10×10 grid defined on the i-th image). If two neighboring nodes $z_{i,r}$ and $z_{i,s}$ get different labels the energy $E(z_i)$ increases by 1. For biased bounds we use the following bias function which favors

Opt. Method	Random		$\lambda = 0.00$		$\lambda = 0.25$		$\lambda = 0.50$		$\lambda = 1.00$	
	Acc.% $\pm$ std	O.F.	Acc. %	O.F.	Acc. %	O.F.	Acc. %	O.F.	Acc. %	O.F.
CCP	41.94 ± 1.1	15.20	40.88	14.81	43.99	14.77	45.60	14.72	46.62	14.70
G-MM random	47.51 ± 0.7	14.89	43.38	14.71	44.41	14.70	47.12	14.66	49.88	14.58
G-MM biased	49.34 ± 0.9	14.55	44.83	14.63	48.07	14.51	53.68	14.33	56.03	14.32

Table 3. Performance of LS-SVM trained with CCP and G-MM on MIT-Indoor dataset. We report classification accuracy (Acc.%) and the training objective value (O.F.). Columns correspond to different initialization schemes. "Random" assigns random parts to regions. λ controls the coherency of the initial part assignments: $\lambda = 1$ ($\lambda = 0$) corresponds to the most (the least) coherent case. "G-MM random" uses random bounds and "G-MM biased" uses the bias function of (13). $\eta = 0.1$ in all the experiments. Coherent initializations lead to better models in general, but, they require discovering good initial parts. "G-MM" outperforms CCP, especially with random initialization. "G-MM biased" performs the best.

bounds that correspond to more coherent labelings:

$$g(b, w) = -\sum_{i=1}^{n} E(z_i), \qquad z(b) = (z_1, \ldots, z_n). \quad (13)$$

Table 3 compares performance of models trained using CCP and G-MM with random and biased bounds. For G-MM with random bounds we repeat the experiment five times and report the average over these five trials. Also, for random initialization, we do five trials using different random seeds and report the mean and standard deviation of the results. G-MM does better than CCP under *all* initializations. It also converges to a solution with lower training objective value than CCP. Our results show that picking bounds uniformly at random from the set of valid bounds is slightly (but consistently) better than committing to the CCP bound. We get a remarkable boost in performance when we use a reasonable prior over bounds (*i.e.* the bias function of (13)). With $\lambda = 1$, CCP attains accuracy of 46.6%, whereas G-MM attains 49.9%, and 56.0% accuracy with random and biased initialization respectively. Moreover, G-MM is less sensitive to initialization.

5.4. Running Time

G-MM bounds make a fraction of the progress that can be made in each bound construction step. Therefore, we would expect G-MM to require more steps to converge when compared to MM. We report the number of iterations in MM and G-MM in Table 4. The results for G-MM depend on the value of the progress coefficient η which is set to match the experiments in the paper; $\eta = 0.02$ for the clustering experiment (Section 5.1) and $\eta = 0.10$ for the scene recognition experiment (Section 5.3).

The overhead of the bound construction step depends on the application. For example, in the scene recognition experiment, optimizing the bounds takes orders of magnitude more than sampling them (a couple of hours vs. a few seconds). In the clustering experiment, however, the optimization step is solved in closed form whereas sampling a bound involves performing a random walk on a large graph which can take a couple of minutes to run.

experiment	setup	MM	G-MM	
			random	biased
scene recognition	$\lambda = 0.0$	145	107	87
	$\lambda = 1.0$	65	69	138
data clustering (GMM-200)	forgy	35.76 ± 7.8	91.52 ± 4.4	
	rand. part.	114.98 ± 12.9	241.89 ± 2.1	
	k-means++	32.92 ± 5.8	80.78 ± 2.9	
data clustering (Cloud)	forgy	37.18 ± 12.1	87.68 ± 15.4	
	rand. part.	65.14 ± 18.7	138.64 ± 5.9	
	k-means++	21.3 ± 4.1	44.12 ± 10.7	

Table 4. Comparison of the number of iterations that MM and G-MM take to converge in the scene recognition and the data clustering experiment with different initializations. The numbers reported for the clustering experiment are the average and standard deviation over 50 trials.

6. Conclusion

We introduced Generalized Majorization-Minimization (G-MM), an iterative bound optimization framework that generalizes Majorization-Minimization (MM). Our key observation is that MM enforces an overly-restrictive touching constraint when constructing bounds, which is inflexible and can lead to sensitivity to initialization. By adopting a different measure of progress, G-MM relaxes this constraint, allowing more freedom in bound construction. Specifically, we propose deterministic and stochastic ways of selecting bounds from a set of valid ones. This generalized bound construction process tends to be less sensitive to initialization, and enjoys the ability to directly incorporate rich application-specific priors and constraints, without modifications to the objective function. In experiments with several latent variable models, G-MM algorithms are shown to significantly outperform their MM counterparts.

Future work includes applying G-MM to a wider range of problems and theoretical analysis, such as convergence rate. We also note that, although G-MM is more conservative than MM in moving towards nearby local minima, it still requires *making progress* in every step. Another interesting research direction is to enable G-MM to occasionally pick bounds that do not make progress with respect to the solution of the previous bound, thereby making it possible to get out of local minima, while still maintaining the convergence guarantees of the method.

References

Arthur, D. and Vassilvitskii, S. K-means++: The advantages of careful seeding. In *Symposium on Discrete Algorithms (SODA)*, 2007.

Azizpour, H., Arefiyan, M., Naderi, S., and Carlsson, S. Spotlight the negatives: A generalized discriminative latent model. In *British Machine Vision Conference (BMVC)*, 2015.

Cinbis, R. G., Verbeek, J., and Schmid, C. Weakly Supervised Object Localization with Multi-fold Multiple Instance Learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, January 2016.

Dempster, A. P., Laird, N. M., and Rubin, D. B. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the royal statistical society. Series B (methodological)*, pp. 1–38, 1977.

Felzenszwalb, P., Girshick, R., McAllester, D., and Ramanan, D. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2010.

Ganchev, K., Graça, J., Gillenwater, J., and Taskar, B. Posterior regularization for structured latent variable models. *The Journal of Machine Learning Research*, 11: 2001–2049, 2010.

Gunawardana, A. and Byrne, W. Convergence theorems for generalized alternating minimization procedures. *The Journal of Machine Learning Research*, 6:2049–2073, 2005.

Heitz, G., Elidan, G., Packer, B., and Koller, D. Shape-based object localization for descriptive classification. *International journal of computer vision (IJCV)*, 2009.

Hunter, D., Lange, K., and Yang, I. Optimization transfer using surrogate objective functions. *Journal of Computational and Graphical Statistics*, 2000.

Joachims, T., Finley, T., and Yu, C.-N. J. Cutting-plane training of structural SVMs. *Machine Learning*, 2009.

Kumar, M. P., Packer, B., and Koller, D. Modeling latent variable uncertainty for loss-based learning. In *International Conference on Machine Learning (ICML)*, 2012.

Lee, D. D. and Seung, H. S. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755): 788–791, 1999.

MacQueen, J. Some methods for classification and analysis of multivariate observations. In *Berkeley Symposium on Mathematical Statistics and Probability*, 1967.

Neal, R. and Hinton, G. A view of the EM algorithm that justifies incremental, sparse, and other variants. *Learning in Graphical Models*, 1998.

Parizi, S. N., Oberlin, J., and Felzenszwalb, P. Reconfigurable models for scene recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

Pietra, S. D., Pietra, V. D., and Lafferty, J. Inducing features of random fields. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(4):380–393, 1997.

Ping, W., Liu, Q., and Ihler, A. Marginal structured SVM with hidden variables. In *International Conference on Machine Learning (ICML)*, 2014.

Pirsiavash, H. and Ramanan, D. Parsing videos of actions with segmental grammars. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

Quattoni, A. and Torralba, A. Recognizing indoor scenes. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

Rastegari, M., Hajishirzi, H., and Farhadi, A. Discriminative and consistent similarities in instance-level multiple instance learning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

Ries, C., Richter, F., and Lienhart, R. Towards automatic bounding box annotations from weakly labeled images. *Multimedia Tools and Applications*, 2015.

Salakhutdinov, R., Roweis, S., and Ghahramani, Z. On the convergence of bound optimization algorithms. In *Proceedings of the Nineteenth conference on Uncertainty in Artificial Intelligence (UAI)*, pp. 509–516. Morgan Kaufmann Publishers Inc., 2002.

Song, H. O., Girshick, R., Jegelka, S., Mairal, J., Harchaoi, Z., and Darrell, T. On learning to localize objects with minimal supervision. In *International Conference on Machine Learning (ICML)*, 2014.

Tang, M., Ayed, I. B., and Boykov, Y. Pseudo-bound optimization for binary energies. In *European Conference on Computer Vision (ECCV)*. Springer, 2014.

Tang, M., Marin, D., Ayed, I. B., and Boykov, Y. Kernel cuts: Kernel and spectral clustering meet regularization. In *International Journal of Computer Vision (IJCV)*. Springer, 2019.

Tao, P. D. Convex analysis approach to dc programming: Theory, algorithms and applications. *Acta Mathematica Vietnamica*, 22(1):289–355, 1997.

Veenman, C. J., Reinders, M., and Backer, E. A maximum variance cluster algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2002.

Xing, E. P., Jordan, M. I., Russell, S. J., and Ng, A. Y. Distance metric learning with application to clustering with side-information. In Becker, S., Thrun, S., and Obermayer, K. (eds.), *Advances in Neural Information Processing Systems (NIPS)*, pp. 521–528. MIT Press, 2002.

Yu, C.-N. Transductive learning of structural svms via prior knowledge constraints. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 1367–1376, 2012.

Yu, C.-N. J. and Joachims, T. Learning structural SVMs with latent variables. In *International Conference on Machine Learning (ICML)*. ACM, 2009.

Yuille, A. and Rangarajan, A. The concave-convex procedure. *Neural computation*, 2003.

Zhou, B., Lapedriza, A., Xiao, J., Torralba, A., and Oliva, A. Learning deep features for scene recognition using places database. In *Advances in neural information processing systems (NIPS)*, 2014.