

Spring School Lecture Part 1: Learning Models from Data with Non-Convex Optimization

CHANDRAJIT BAJAJ (bajaj@cs.utexas.edu) <http://www.cs.utexas.edu/~bajaj>

Problem structures that allow non-convex approaches to avoid NP-hardness results, are very similar to those that allow their convex relaxation counterparts to avoid distortions and a large relaxation gap.

However, non-convex techniques usually offer more scalable solutions !

Ref:

- [1] [Non-convex Optimization for Machine Learning](#) Prateek Jain, Purushottam Kar
- [2] [A Mathematical Primer for Computational Data Sciences](#) Chandrajit Bajaj
- [3] [SMAC: Simultaneous Mapping and Clustering](#) Chandrajit Bajaj, Tingran Gao, Qixing Huang

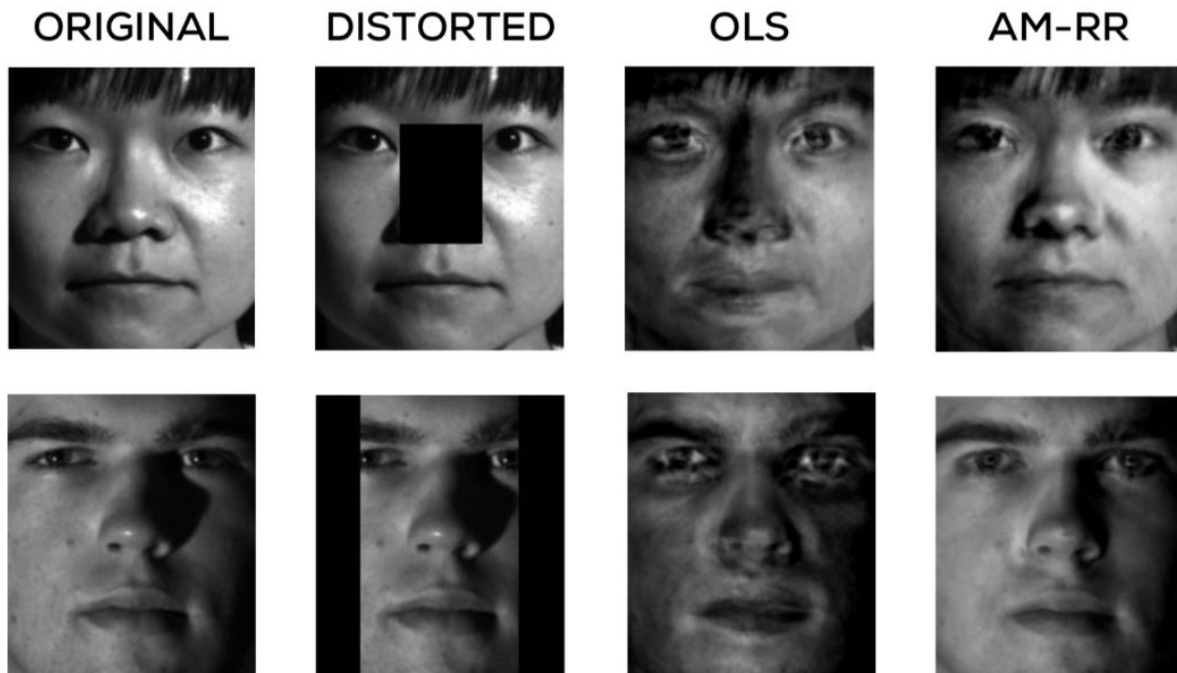
Additional References are specific to Algorithms and cited below.

Mathematical Notation ([1], [2])

- The sets of real numbers and natural numbers are denoted by \mathbb{R} and \mathbb{N} respectively.
- The cardinality of a set S is denoted by $|S|$.
- Vectors are denoted by boldface, lower case alphabets (eg \mathbf{x} , \mathbf{y}). The zero vector is denoted by $\mathbf{0}$. A vector $\mathbf{x} \in \mathbb{R}^p$ will be in column format. The transpose of a vector is denoted by \mathbf{x}^T . The i^{th} coordinate of a vector \mathbf{x} is denoted by \mathbf{x}_i .
- Matrices are denoted by upper case alphabets (eg A , B). A_i denotes the i^{th} column of the matrix A and A_j denotes its j^{th} row. A_{ij} denotes the element at the i^{th} row and j^{th} column.
- For a vector $\mathbf{x} \in \mathbb{R}^p$ and a set $S \subset [p]$, the notation \mathbf{x}_S denotes the vector $\mathbf{z} \in \mathbb{R}^p$ such that $\mathbf{z}_i = \mathbf{x}_i$ for $i \in S$, and $\mathbf{z}_i = 0$ otherwise. Similarly for matrices, A_S denotes the matrix B with $B_i = A_i$ for $i \in S$ and $B_i = 0$ for $i \notin S$. Also, A^S denotes the matrix B with $B_i = A_i$ for $i \in S$ and $B_i = 0^T$ for $i \notin S$.
- The support of a vector \mathbf{x} is denoted by $\text{supp}(\mathbf{x}) := \{i : \mathbf{x}_i \neq 0\}$. A vector \mathbf{x} is referred to as s -sparse if $|\text{supp}(\mathbf{x})| \leq s$.
- The canonical directions in \mathbb{R}^p are denoted by \mathbf{e}_i , $i = 1, \dots, p$.
- The identity matrix of order p is denoted by $I_{p \times p}$ or simply I_p .
- For a vector $\mathbf{x} \in \mathbb{R}^p$, the notation $\|\mathbf{x}\|_q = \sqrt[q]{\sum_{i=1}^p |\mathbf{x}_i|^q}$ denotes its L_q norm. As special cases we define $\|\mathbf{x}\|_\infty := \max_i |\mathbf{x}_i|$, $\|\mathbf{x}\|_{-\infty} := \min_i |\mathbf{x}_i|$, and $\|\mathbf{x}\|_0 := |\text{supp}(\mathbf{x})|$.
- Balls with respect to various norms are denoted as $B_q(r) := \{\mathbf{x} \in \mathbb{R}^p, \|\mathbf{x}\|_q \leq r\}$. Note, $B_0(s)$ is used to denote the set of s -sparse vectors.
- For a matrix $A \in \mathbb{R}^{m \times n}$, $\sigma_1(A) \geq \sigma_2(A) \geq \dots \geq \sigma_{\min\{m,n\}}(A)$ denote its singular values. The Frobenius norm of A is defined as $\|A\|_F := \sqrt{\sum_{i,j} A_{ij}^2} = \sqrt{\sum_i \sigma_i(A)^2}$. The nuclear norm of A is defined as $\|A\|_* := \sum_i \sigma_i(A)$.
- The trace of a square matrix $A \in \mathbb{R}^{m \times m}$ is defined as $\text{tr}(A) = \sum_{i=1}^m A_{ii}$.
- The spectral norm (also referred to as the operator norm) of a matrix A is defined as $\|A\|_2 := \max_i \sigma_i(A)$.

Applications I: “Learning Models from Data”

Face Recognition : In biometrics, a fundamental problem is to identify if a new face image belongs to that of a registered individual or not.



Cast as a regression problem by trying to fit various features of the new image to corresponding features of existing images of the individual in the registered database. Assume that images are represented as n -dimensional feature vectors say, using simple pixel-based features. Also assume that there already exist p images of each person in the database. Represent the new image $\mathbf{x}^t \in \mathbb{R}^n$ in terms of the database images $X = [\mathbf{x}_1, \dots, \mathbf{x}_p] \in \mathbb{R}^{n \times p}$ of that person. One solution is to perform *linear* interpolation. If the person is genuine, then there will exist a combination \mathbf{w}^* such that for all i , we have $\mathbf{x}_i^t \approx X^i \mathbf{w}^*$ i.e., all features can be faithfully reconstructed.

A popular way to recover \mathbf{w}^* is using the *least squares* formulation

$$\mathbf{w}^* = \operatorname{argmin}_{\mathbf{w} \in \mathbb{R}^p} \sum_{i=1..n} (y_i - \mathbf{x}_i^T \mathbf{w})^2.$$

Problematic, if however the new image \mathbf{x}^t has occlusions or is otherwise corrupted

$\mathbf{x}_i^t = X^i \mathbf{w}^* + \mathbf{b}_i^*$ where $\mathbf{b}_i^* = 0$ on uncorrupted pixels but can take large and unpredictable values for corrupted pixels .

Gene Expression Analysis : Given say, for n human test subjects participating in the study, the expression levels of a large number p of genes (encoded as a real vector $\mathbf{x}_i \in \mathbb{R}^p$), and the corresponding phenotypical trait $y_i \in \mathbb{R}$. For the sake of simplicity, we assume that the phenotypical response is linearly linked to the gene expression levels i.e. for some $\mathbf{w}^* \in \mathbb{R}^p$, we have $y_i = \mathbf{x}_i^T \mathbf{w}^* + \eta_i$ where η_i is some noise.

The goal then is to use gene expression data to deduce an estimate for \mathbf{w}^* . Having access to the model \mathbf{w}^* can be instrumental in discovering possible genetic bases for diseases, traits etc.

The linear regression problem as well as the least squares estimator, comes in two flavours, below :

- (a) only a few of the p features/covariates are actually relevant to the problem but do not know their identity, i.e. $n \gg p$ (Face Recognition)
- (b) working in extremely data-starved settings i.e., $n \ll p$

Issues:

Firstly, the number of genes whose expression levels are being recorded is usually very large (running into several tens of thousands), the number of samples (test subjects) is usually not nearly as large, i.e. $n \ll p$.

Standard statistical approaches require at least $n \geq p$ data points to ensure a consistent estimation of all p model parameters and are unable to offer accurate model estimates in the face of data-starvation.

Secondly, we do not expect all genes being tracked to participate in realizing the phenotype. Indeed, the whole objective of this exercise is to identify a small set of genes which most prominently influence the given phenotype. Note that this implies that the vector \mathbf{w}^* is very sparse. Traditional linear regression cannot guarantee the recovery of a sparse model.

In both cases *sparse regression/recovery* solves the twin problems as $n \geq s \log p$ (as opposed to $n \geq p$) data points are required for sparse recovery to work , drastically reducing the data requirement.

Problems can be handled by the *sparse recovery* approach, which seeks to fit a sparse model vector (i.e., a vector with say, no more than s non-zero entries) to the data. The least squares formulation, modified as a *sparse recovery* problem, is given

$$\mathbf{w}_{sp} = \underset{\mathbf{w} \in \mathbb{R}^p}{\operatorname{argmin}} \sum_{i=1..n} (y_i - \mathbf{x}_i^T \mathbf{w})^2 \quad \text{s.t. } \mathbf{w} \in B_0(s) \quad [B_0(s) := \{\mathbf{x} \in \mathbb{R}^p, \|\mathbf{x}\|_0 \leq s\}]$$

Although the **objective** function in the above formulation is **convex**, the *constraint* $\|\mathbf{w}\|_0 \leq s$ corresponds to a *non-convex constraint set*.

Application II : Learning Models from Data

Recommendation Systems:

Several internet search engines and e-commerce websites utilize recommendation systems to offer items to users that they would benefit from, or like, the most. (recommendations for songs etc, all the way to critical recommendations in *personalized medicine*).

To be able to make *accurate* recommendations, we need very good estimates of how each user likes each item (song), or would benefit from it (drug).

However, users typically rate only a handful of the hundreds of thousands of songs in any commercial catalog and it is not feasible, or even advisable, to administer every drug to a user.

Thus, for the vast majority of user-item pairs, we have no direct information.

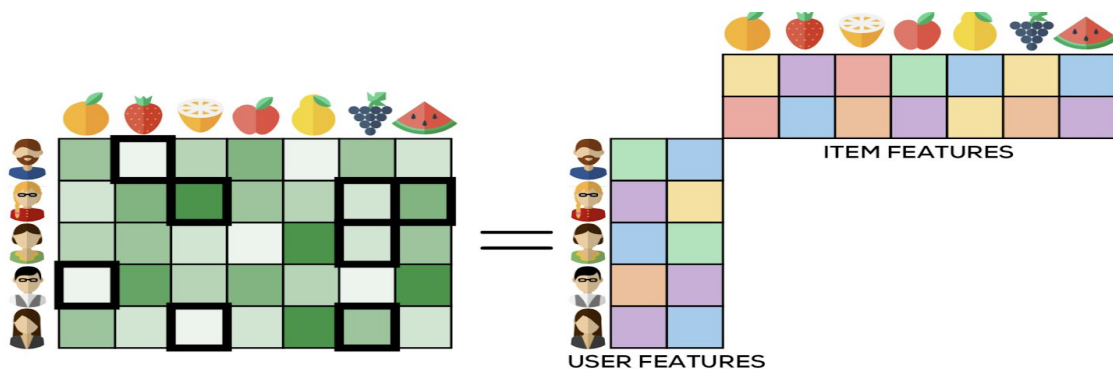


Figure 1.2: Only the entries of the ratings matrix with thick borders are observed. Notice that users rate infrequently and some items are not rated even once. Non-convex optimization techniques such as low-rank matrix completion can help recover the unobserved entries, as well as reveal hidden features that are descriptive of user and item properties, as shown on the right hand side.

It is useful to visualize this problem as a **matrix completion** problem: for a set of m users u_1, \dots, u_m and n items a_1, \dots, a_n , we have an $m \times n$ *preference matrix* $A = [A_{ij}]$ where A_{ij} encodes the preference of the i th user for the j th item.

Now, it is easy to see that *unless there exists some structure in matrix, and by extension, in the way users rate items*, there would be no relation between the unobserved entries and the observed ones. This would result in there *being no unique way to complete the matrix*. Thus, it is essential to impose some structure on the matrix.

A structural assumption popularly made is that of **low rank**: we wish to fill in the missing entries of A *assuming* that A is a *low rank matrix*. This can make the problem well-posed and have a unique solution since the additional low rank structure links the entries of the matrix together. The unobserved entries can no longer take values independently of the observed values.

If we denote by $\Omega \subset [m] \times [n]$, the set of observed entries of A , then the low rank matrix completion problem can be written in the form

$$A_{lr} = \operatorname{argmin}_{x \in \mathbb{R}^{m \times n}} \sum_{(i,j) \in \Omega} (x_{ij} - A_{ij})^2 \quad \text{s.t. } \operatorname{rank}(x) \leq r,$$

where the rating given by user i to item j can now be seen to be $A_{ij} \approx U_i \cdot V_j$.

This formulation also has a **convex objective** but a **non-convex rank constraint**.

Assuming the ratings matrix to have rank at most r is equivalent to assuming that the matrix A can be written as $A = UV^T$ with the matrices $U \in \mathbb{R}^{m \times r}$ and $V \in \mathbb{R}^{n \times r}$ having at most r columns. The alternate formulation becomes

$$A_{lv} = \operatorname{argmin}_{U \in \mathbb{R}^{m \times r}, V \in \mathbb{R}^{n \times r}} \sum_{(i,j) \in \Omega} (U_i^T V_j - A_{ij})^2$$

There are no constraints in the formulation. However, the formulation requires joint optimization over a pair of variables (U, V) instead of a single variable. More importantly, it can be shown that the **objective** function is **non-convex** in (U, V) .

Recovering the rank r matrix A also gives us a bunch of *r -dimensional latent vectors* describing the users and items. These latent vectors can be extremely valuable in themselves as they can help us in understanding user behavior and item popularity, as well as be used in “*content*”-based recommendation systems which can effectively utilize item and user features.

The above examples, and several others from machine learning, such as (i) *low-rank tensor decomposition*, (ii) *training deep networks*, and (iii) training structured models, demonstrate the utility of non-convex optimization in naturally modeling learning tasks.


Application III : Learning Models from Data

Multi-Assembly Systems (SMAC: Simultaneously Mapping and Clustering)

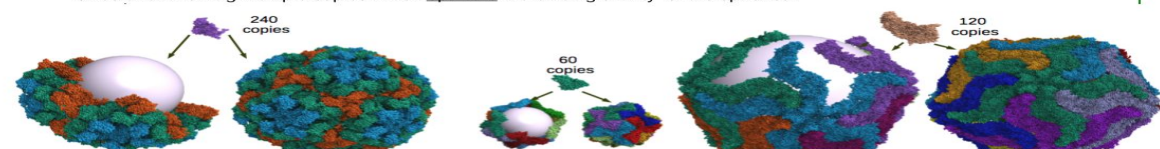
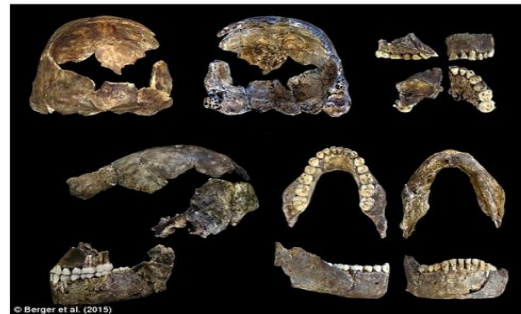
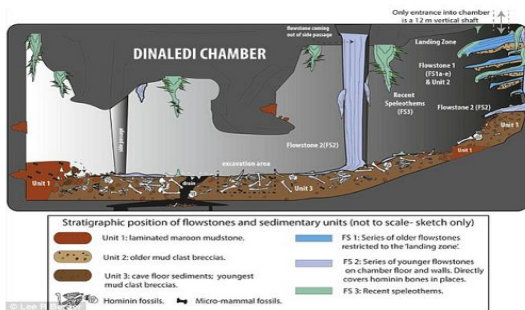
Combinatorial Geometric Optimization Problems

Multi-model Assembly Prediction :
 Given a collection of n -structural models search over the flexible configurations of each to **optimize** the binding affinity of the n -complex.

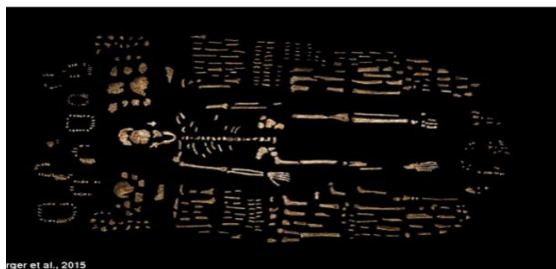
Approach: Multi-model assembly prediction can be defined as the solution of a **combinatorial geometric optimization** problem which **maximizes** a global configurational dependent binding affinity **score** of the n -model assembly



Spherical Shell Assembly Prediction:
 Given a **viral protein structure** model **predict** flexible spherical tiling arrangements at different sizes and involving multiple copies which **optimize** the binding affinity of the spherical

Homo Naledi



HOMO FEATURES

Humanlike skull
 The general shape of *H. naledi*'s skull is advanced, though the braincase is less than half of a modern human's.

Versatile hands
H. naledi's palms, wrists, and thumbs are humanlike, suggesting tool use.

Long legs
 The leg bones are long and slender and have the strong muscle attachments characteristic of a modern bipedal gait.


Humanlike feet
 Except for the slightly curved toes, *H. naledi*'s feet are nearly indistinguishable from ours, with arches that suggest an efficient, long-distance stride.

AUSTRALOPITHECINE FEATURES

Primitive shoulders
H. naledi's shoulders are positioned in a way that would have helped with climbing and hanging.

Flared pelvis
 The hip bones of *H. naledi* flare outward—a primitive trait—and are shorter front to back than those of modern humans.

Curved fingers
 Long, curved fingers, useful for climbing in trees, could be a trait retained from a more apeline ancestor.



The **objective** function is *non-convex* and in very high dimensions.

Solution Approaches #1: Convex Relaxation

Faced with the challenge of non-convexity, and the associated NP-hardness, a traditional workaround in literature has been to modify the problem formulation itself so that convex optimization tools can be readily applied. This is often done by relaxing the problem so that it becomes a convex optimization problem.

For sparse linear regression, the relaxation approach gives us the popular **LASSO** (least absolute shrinkage and selection operator) formulation.

Now, in general, such modifications change the problem drastically, and the solutions of the relaxed formulation can be poor solutions to the original problem.

However, it is known that if the problem possesses certain nice structure, then under careful relaxation, these distortions, formally referred to as a "relaxation gap", are absent, i.e., solutions to the relaxed problem would be optimal for the original non-convex problem as well.

Although a popular and successful approach, this still has limitations, the most prominent of them being scalability.

Although the relaxed convex optimization problems are solvable in polynomial time, it is often challenging to solve them efficiently for large-scale problems.

Relaxation-based methods : LASSO, various extended LASSO, Elastic Nets, SVT (Singular Value Thresholding)

Solution Approaches #2: Non-Convex

The alternative approach to solving machine learning and signal processing problems is the *non-convex optimization* approach owing to its goal of optimizing non-convex formulations directly.

Techniques frequently used in non-convex optimization approaches include simple and efficient primitives such as **projected gradient descent**, **alternating minimization**, **the expectation-maximization algorithm**, **stochastic optimization**, and **variants thereof**. These are very fast and hence scalable in practice.

Non-Convex Methods : **ProjectedGradientDescent(PGD)**, **GeneralizedProjectedGradientDescent(gPGD)**,

GeneralizedAlternatingMinimization(gAM), **AltMaxforLatentVariableModels(AM-LVM)**, **ExpectationMaximization(EM)**,

NoisyGradientDescent(NGD), **ProjectedNoisyGradientDescent(PNGD)**, **IterativeHard-thresholding(IHT)**, **SingularValueProjection(SVP)** ,

1. Convex Projected Gradient Descent

The following convex optimization problem :

$$\min_{\mathbf{x} \in \mathbb{R}^p} f(\mathbf{x}) \quad \text{s.t. } \mathbf{x} \in \mathcal{C}.$$

where $\mathcal{C} \subset \mathbb{R}^p$ is a convex constraint set and $f : \mathbb{R}^p \rightarrow \mathbb{R}$ is a convex objective function.

A fundamental step includes

Algorithm 1 Projected Gradient Descent (PGD)

Input: Convex objective f , convex constraint set \mathcal{C} , step lengths η_t

Output: A point $\hat{\mathbf{x}} \in \mathcal{C}$ with near-optimal objective value

- 1: $\mathbf{x}^1 \leftarrow \mathbf{0}$
 - 2: **for** $t = 1, 2, \dots, T$ **do**
 - 3: $\mathbf{z}^{t+1} \leftarrow \mathbf{x}^t - \eta_t \cdot \nabla f(\mathbf{x}^t)$
 - 4: $\mathbf{x}^{t+1} \leftarrow \Pi_{\mathcal{C}}(\mathbf{z}^{t+1})$
 - 5: **end for**
 - 6: (OPTION 1) **return** $\hat{\mathbf{x}}_{\text{final}} = \mathbf{x}^T$
 - 7: (OPTION 2) **return** $\hat{\mathbf{x}}_{\text{avg}} = (\sum_{t=1}^T \mathbf{x}^t) / T$
 - 8: (OPTION 3) **return** $\hat{\mathbf{x}}_{\text{best}} = \arg \min_{t \in [T]} f(\mathbf{x}^t)$
-

The projected gradient descent algorithm is given by Algorithm 1. The procedure generates iterates \mathbf{x}^t by taking steps guided by the gradient in an effort to reduce the function value locally. Finally it returns either the final iterate, the average iterate, or the best iterate.

Convergence Guarantee

Consider PGD for *objective functions* that are either

- a)** convex with bounded gradients, or **b)** strongly convex and strongly smooth.

Let $f^* = \min_{\mathbf{x} \in \mathcal{C}} f(\mathbf{x})$ be the optimal value of the optimization problem. A point $\mathbf{x} \in \mathcal{C}$ will be said to be an ϵ -optimal solution if $f(\mathbf{x}) \leq f^* + \epsilon$.

(a)Theorem 2.5. *Let f be a convex objective with bounded gradients and Algorithm 1 be executed for T time steps with step lengths $\eta_t = \eta = 1/\sqrt{T}$. Then, for any $\epsilon > 0$, if $T = O(1/\epsilon^2)$, and*

since function values on average approach f^* , $\frac{1}{T} \sum_{t=1}^T f(\mathbf{x}^t) \leq f^* + \epsilon$.

Similarly $f(\hat{\mathbf{x}}_{\text{best}}) \leq \frac{1}{T} \sum_{t=1}^T f(\mathbf{x}^t) \leq f^* + \epsilon$, since $f(\hat{\mathbf{x}}_{\text{best}}) \leq f(\mathbf{x}^t)$. Furthermore, applying Jensen's

inequality for convex functions, $f(\hat{\mathbf{x}}_{\text{avg}}) = f\left(\frac{1}{T} \sum_{t=1}^T \mathbf{x}^t\right) \leq \frac{1}{T} \sum_{t=1}^T f(\mathbf{x}^t) \leq f^* + \epsilon$.

(b) SC = Strongly Convex, SS = Strongly Smooth

Definition 2.1 (Convex Combination). A convex combination of a set of n vectors $\mathbf{x}_i \in \mathbb{R}^p$, $i = 1 \dots n$ in an arbitrary real space is a vector $\mathbf{x}_\theta := \sum_{i=1}^n \theta_i \mathbf{x}_i$ where $\theta = (\theta_1, \theta_2, \dots, \theta_n)$, $\theta_i \geq 0$ and $\sum_{i=1}^n \theta_i = 1$.

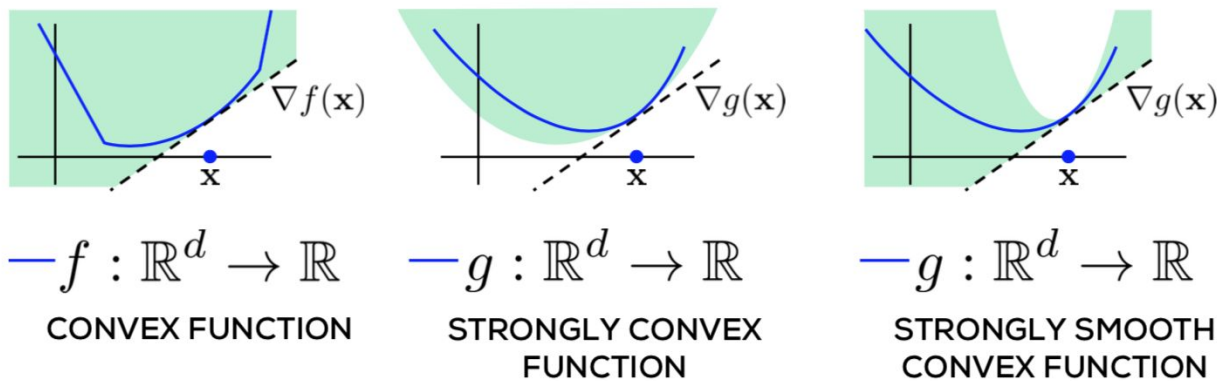


Figure 2.2: A convex function is lower bounded by its own tangent at all points. Strongly convex and smooth functions are, respectively, lower and upper bounded in the rate at which they may grow, by quadratic functions and cannot, again respectively, grow too slowly or too fast. In each figure, the shaded area describes regions the function curve is permitted to pass through.

Definition 2.2 (Convex Set). A set $\mathcal{C} \in \mathbb{R}^p$ is considered convex if, for every $\mathbf{x}, \mathbf{y} \in \mathcal{C}$ and $\lambda \in [0, 1]$, we have $(1 - \lambda) \cdot \mathbf{x} + \lambda \cdot \mathbf{y} \in \mathcal{C}$ as well.

Definition 2.3 (Convex Function). A continuously differentiable function $f : \mathbb{R}^p \rightarrow \mathbb{R}$ is considered convex if for every $\mathbf{x}, \mathbf{y} \in \mathbb{R}^p$ we have $f(\mathbf{y}) \geq f(\mathbf{x}) + \langle \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle$, where $\nabla f(\mathbf{x})$ is the gradient of f at \mathbf{x} .

Definition 2.4 (Strongly Convex/Smooth Function). A continuously differentiable function $f : \mathbb{R}^p \rightarrow \mathbb{R}$ is considered α -strongly convex (SC) and β -strongly smooth (SS) if for every $\mathbf{x}, \mathbf{y} \in \mathbb{R}^p$, we have

$$\frac{\alpha}{2} \|\mathbf{x} - \mathbf{y}\|_2^2 \leq f(\mathbf{y}) - f(\mathbf{x}) - \langle \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle \leq \frac{\beta}{2} \|\mathbf{x} - \mathbf{y}\|_2^2.$$

Theorem 2.6. Let f be an objective that satisfies the α -SC and β -SS properties. Let Algorithm 1 be executed with step lengths $\eta_t = \eta = \frac{1}{\beta}$. Then after at most $T = \mathcal{O}\left(\frac{\beta}{\alpha} \log \frac{\beta}{\epsilon}\right)$ steps, we have $f(\mathbf{x}^T) \leq f(\mathbf{x}^*) + \epsilon$.

A more general definition that extends to non-differentiable functions uses the notion of *subgradient* to replace the gradient in the above expression.

1. Non-convex Projected Gradient Descent

The algorithmic and analytic techniques used in convex problems are extensible to non-convex problems that possess nice additional structure :

- A. constraint sets that, despite being non-convex, possess additional structure so that projections onto them can be carried out efficiently.
- B. structural properties of objective functions that can aid optimization.

Problems that possess nicely structured objective functions and constraint sets, the PGD-style algorithm for non-convex problem, converges to the global optimum in *polynomial* time with a linear rate of convergence.

Projection Operators

Convex Projections

Given any closed set $C \subset \mathbb{R}^p$, the projection operator $\Pi_C(\cdot)$ is defined as

$$\Pi_C(\mathbf{z}) := \arg \min_{\mathbf{x} \in C} \|\mathbf{x} - \mathbf{z}\|_2$$

(Note, one need not use only the L2-norm in defining projections)

For instance, if $C = B_2(1)$ i.e., the unit L2 ball, then projection is equivalent to a normalization step

$$\Pi_{B_2(1)}(\mathbf{z}) = \begin{cases} \mathbf{z} / \|\mathbf{z}\|_2 & \text{if } \|\mathbf{z}\|_2 > 1 \\ \mathbf{z} & \text{otherwise} \end{cases}.$$

For the case $C = B_1(1)$, the projection step reduces to the popular *soft thresholding* operation. If $\mathbf{z} := \Pi_{B_1(1)}(\mathbf{z})$, then $\mathbf{z}_i = \max\{\mathbf{z}_i - \theta, 0\}$, where θ is a threshold that can be decided by a sorting operation on the vector .

Lemma 2.2 (Projection Property-O). For any set (convex or not) $C \subset \mathbb{R}^p$ and $\mathbf{z} \in \mathbb{R}^p$, let $\hat{\mathbf{z}} := \Pi_C(\mathbf{z})$. Then for all $\mathbf{x} \in C$, $\|\hat{\mathbf{z}} - \mathbf{z}\|_2 \leq \|\mathbf{x} - \mathbf{z}\|_2$.

Projection Property-O (Lemma 2.2), often called a zeroth order property, always holds, whether the underlying set is convex or not.

Lemma 2.3 (Projection Property-I). For any convex set $C \subset \mathbb{R}^p$ and any $\mathbf{z} \in \mathbb{R}^p$, let $\hat{\mathbf{z}} := \Pi_C(\mathbf{z})$. Then for all $\mathbf{x} \in C$, $\langle \mathbf{x} - \hat{\mathbf{z}}, \mathbf{z} - \hat{\mathbf{z}} \rangle \leq 0$.

Lemma 2.4 (Projection Property-II). For any convex set $C \subset \mathbb{R}^p$ and any $\mathbf{z} \in \mathbb{R}^p$, let $\hat{\mathbf{z}} := \Pi_C(\mathbf{z})$. Then for all $\mathbf{x} \in C$, $\|\hat{\mathbf{z}} - \mathbf{x}\|_2 \leq \|\mathbf{z} - \mathbf{x}\|_2$.

Projection Properties-I and II (Lemmas 2.3, 2.4) are *first order* properties and can be violated if the underlying set is non-convex.

Projection on Non-Convex Sets:

Executing the projected gradient descent algorithm with non-convex problems requires projections onto non-convex sets.

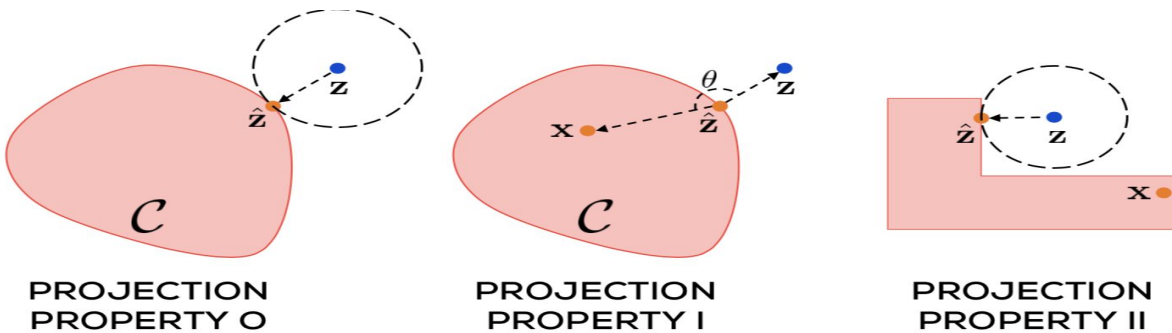


Figure 2.3: A depiction of projection operators and their properties. Projections reveal a closest point in the set being projected onto. For convex sets, projection property I ensures that the angle θ is always non-acute. Sets that satisfy projection property I also satisfy projection property II. Projection property II may be violated by non-convex sets. Projecting onto them may take the projected point \mathbf{z} closer to certain points in the set (for example, $\hat{\mathbf{z}}$) but farther from others (for example, \mathbf{x}).

A (i) Projecting onto Sparse Vectors

In the sparse linear regression problem

$$\mathbf{W}^* = \underset{\|\mathbf{w}\|_0 \leq s}{\operatorname{argmin}} \sum_{i=1..n} (y_i - \mathbf{x}_i^T \mathbf{w})^2,$$

applying projected gradient descent requires projections onto the set of s -sparse vectors i.e., $\mathcal{B}_0(s) := \{\mathbf{x} \in \mathbb{R}^p, \|\mathbf{x}\|_0 \leq s\}$. The following result shows that the projection $\Pi_{\mathcal{B}_0(s)}(\mathbf{z})$ can be carried out by simply sorting the coordinates of the vector \mathbf{z} according to magnitude and setting all except the top- s coordinates to zero.

Lemma 3.1. For any vector $\mathbf{z} \in \mathbb{R}^p$, let σ be the permutation that sorts the coordinates of \mathbf{z} in decreasing order of magnitude, i.e., $|\mathbf{z}_{\sigma(1)}| \geq |\mathbf{z}_{\sigma(2)}| \geq \dots \geq |\mathbf{z}_{\sigma(p)}|$. Then the vector $\hat{\mathbf{z}} := \Pi_{\mathcal{B}_0(s)}(\mathbf{z})$ is obtained by setting $\hat{\mathbf{z}}_i = \mathbf{z}_i$ if $\sigma(i) \leq s$ and $\hat{\mathbf{z}}_i = 0$ otherwise.

A (ii) Projecting onto Low-rank Matrices

In the recommendation systems problem, we project onto the set of low-rank matrices.

$$\hat{A}_{\text{lr}} = \arg \min_{\text{rank}(X) \leq r} \sum_{(i,j) \in \Omega} (X_{ij} - A_{ij})^2,$$

Consider matrices of a certain order, say $m \times n$ and let $C \subset \mathbb{R}^{m \times n}$ be an arbitrary set of matrices. Then, the projection operator $\Pi_C(\cdot)$ is defined as follows: for any matrix $A \in \mathbb{R}^{m \times n}$,

$$\Pi_C(A) := \arg \min_{X \in C} \|A - X\|_F,$$

where $\|\cdot\|_F$ is the Frobenius norm over matrices.

For low rank projections we require C to be the set of low rank matrices $\text{B}_{\text{rank}(r)} := \{A \in \mathbb{R}^{m \times n}, \text{rank}(A) \leq r\}$. Yet again, this projection can be done efficiently by performing a *Singular Value Decomposition* on the matrix A and retaining the top r singular values and vectors.

Theorem 3.2 (Eckart-Young-Mirsky theorem). *For any matrix $A \in \mathbb{R}^{m \times n}$, let $U \Sigma V^\top$ be the singular value decomposition of A such that $\Sigma = \text{diag}(\sigma_1, \sigma_2, \dots, \sigma_{\min(m,n)})$ where $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_{\min(m,n)}$. Then for any $r \leq \min(m, n)$, the matrix $\hat{A}_{(r)} := \Pi_{\text{B}_{\text{rank}(r)}}(A)$ can be obtained as $U_{(r)} \Sigma_{(r)} V_{(r)}^\top$ where $U_{(r)} := [U_1 U_2 \dots U_r]$, $V_{(r)} := [V_1 V_2 \dots V_r]$, and $\Sigma_{(r)} := \text{diag}(\sigma_1, \sigma_2, \dots, \sigma_r)$.*

C Generalized Projected Gradient Descent

Definition 3.1 (Restricted Convexity). *A continuously differentiable function $f : \mathbb{R}^p \rightarrow \mathbb{R}$ is said to satisfy restricted convexity over a (possibly non-convex) region $C \subseteq \mathbb{R}^p$ if for every $\mathbf{x}, \mathbf{y} \in C$ we have $f(\mathbf{y}) \geq f(\mathbf{x}) + \langle \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle$, where $\nabla f(\mathbf{x})$ is the gradient of f at \mathbf{x} .*

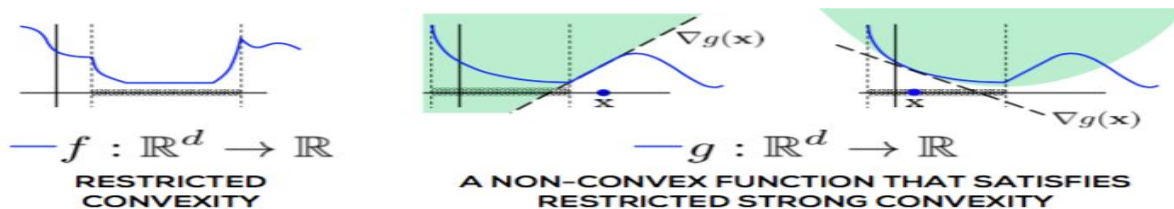


Figure 3.1: A depiction of restricted convexity properties. f is clearly non-convex over the entire real line but is convex within the cross-hatched region bounded by the dotted vertical lines. g is a non-convex function that satisfies restricted strong convexity. Outside the cross-hatched region (again bounded by the dotted vertical lines), g fails to even be convex as its curve falls below its tangent, but within the region, it actually exhibits strong convexity.

Definition 3.2 (Restricted Strong Convexity/Smoothness). *A continuously differentiable function $f : \mathbb{R}^p \rightarrow \mathbb{R}$ is said to satisfy α -restricted strong convexity (RSC) and β -restricted strong smoothness (RSS) over a (possibly non-convex) region $C \subseteq \mathbb{R}^p$ if for every $\mathbf{x}, \mathbf{y} \in C$, we have*

$$\frac{\alpha}{2} \|\mathbf{x} - \mathbf{y}\|_2^2 \leq f(\mathbf{y}) - f(\mathbf{x}) - \langle \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle \leq \frac{\beta}{2} \|\mathbf{x} - \mathbf{y}\|_2^2.$$

Algorithm 2 Generalized Projected Gradient Descent (gPGD)

Input: Objective function f , constraint set \mathcal{C} , step length η

Output: A point $\hat{\mathbf{x}} \in \mathcal{C}$ with near-optimal objective value

- 1: $\mathbf{x}^1 \leftarrow \mathbf{0}$
 - 2: **for** $t = 1, 2, \dots, T$ **do**
 - 3: $\mathbf{z}^{t+1} \leftarrow \mathbf{x}^t - \eta \cdot \nabla f(\mathbf{x}^t)$
 - 4: $\mathbf{x}^{t+1} \leftarrow \Pi_{\mathcal{C}}(\mathbf{z}^{t+1})$
 - 5: **end for**
 - 6: **return** $\hat{\mathbf{x}}_{\text{final}} = \mathbf{x}^T$
-

Theorem 3.3. *Let f be a (possibly non-convex) function satisfying the α -RSC and β -RSS properties over a (possibly non-convex) constraint set \mathcal{C} with $\beta/\alpha < 2$. Let Algorithm [2](#) be executed with a step length $\eta = \frac{1}{\beta}$. Then after at most $T = \mathcal{O}\left(\frac{\alpha}{2\alpha-\beta} \log \frac{1}{\epsilon}\right)$ steps, $f(\mathbf{x}^T) \leq f(\mathbf{x}^*) + \epsilon$.*

2. Alternating Minimization - low-rank matrix recovery, robust regression, phase retrieval

Lloyd's algorithm [Lloyd, 1982] for k-means clustering and the EM algorithm [Dempster et al., 1977] for latent variable models are problem-specific variants of the general alternating minimization principle.

A. Marginal Convexity and Generalized Alternating Minimization (gAM): (structural properties of functions that frequently arise in alternating minimization settings)

Recall the matrix completion problem in recommendation systems from § 1 which involved two variables U and V denoting respectively, the latent factors for the users and the items. In several such cases, the optimization problem, more specifically the objective function, is not *jointly convex* in all the variables.

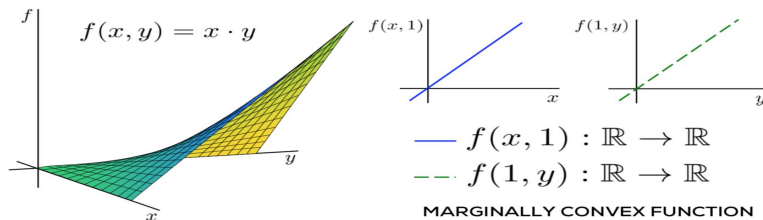


Figure 4.1: A marginally convex function is not necessarily (jointly) convex. The function $f(x, y) = x \cdot y$ is marginally linear, hence marginally convex, in both its variables, but clearly not a (jointly) convex function.

Definition 4.1 (Joint Convexity). A continuously differentiable function in two variables $f : \mathbb{R}^p \times \mathbb{R}^q \rightarrow \mathbb{R}$ is considered *jointly convex* if for every $(\mathbf{x}^1, \mathbf{y}^1), (\mathbf{x}^2, \mathbf{y}^2) \in \mathbb{R}^p \times \mathbb{R}^q$ we have

$$f(\mathbf{x}^2, \mathbf{y}^2) \geq f(\mathbf{x}^1, \mathbf{y}^1) + \langle \nabla f(\mathbf{x}^1, \mathbf{y}^1), (\mathbf{x}^2, \mathbf{y}^2) - (\mathbf{x}^1, \mathbf{y}^1) \rangle,$$

where $\nabla f(\mathbf{x}^1, \mathbf{y}^1)$ is the gradient of f at the point $(\mathbf{x}^1, \mathbf{y}^1)$.

Definition 4.2 (Marginal Convexity). *A continuously differentiable function of two variables $f : \mathbb{R}^p \times \mathbb{R}^q \rightarrow \mathbb{R}$ is considered marginally convex in its first variable if for every value of $\mathbf{y} \in \mathbb{R}^q$, the function $f(\cdot, \mathbf{y}) : \mathbb{R}^p \rightarrow \mathbb{R}$ is convex, i.e., for every $\mathbf{x}^1, \mathbf{x}^2 \in \mathbb{R}^p$, we have*

$$f(\mathbf{x}^2, \mathbf{y}) \geq f(\mathbf{x}^1, \mathbf{y}) + \langle \nabla_{\mathbf{x}} f(\mathbf{x}^1, \mathbf{y}), \mathbf{x}^2 - \mathbf{x}^1 \rangle,$$

where $\nabla_{\mathbf{x}} f(\mathbf{x}^1, \mathbf{y})$ is the partial gradient of f with respect to its first variable at the point $(\mathbf{x}^1, \mathbf{y})$. A similar condition is imposed for f to be considered marginally convex in its second variable.

The alternating minimization algorithm (gAM) is given in Algorithm 3 for an optimization problem on two variables constrained to the sets X and Y respectively. The procedure can be easily extended to functions with more variables, or have more complicated constraint sets⁴ of the form $Z \subset X \times Y$. After an initialization step, gAM alternately fixes one of the variables and optimizes over the other.

Algorithm 3 Generalized Alternating Minimization (gAM)

Input: Objective function $f : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$

Output: A point $(\hat{\mathbf{x}}, \hat{\mathbf{y}}) \in \mathcal{X} \times \mathcal{Y}$ with near-optimal objective value

- 1: $(\mathbf{x}^1, \mathbf{y}^1) \leftarrow \text{INITIALIZE}()$
 - 2: **for** $t = 1, 2, \dots, T$ **do**
 - 3: $\mathbf{x}^{t+1} \leftarrow \arg \min_{\mathbf{x} \in \mathcal{X}} f(\mathbf{x}, \mathbf{y}^t)$
 - 4: $\mathbf{y}^{t+1} \leftarrow \arg \min_{\mathbf{y} \in \mathcal{Y}} f(\mathbf{x}^{t+1}, \mathbf{y})$
 - 5: **end for**
 - 6: **return** $(\mathbf{x}^T, \mathbf{y}^T)$
-

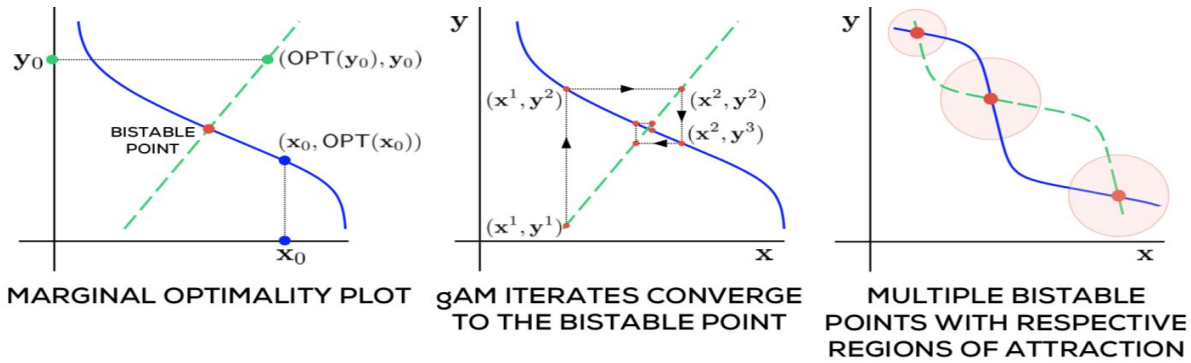
This approach of solving several intermediate *marginal* optimization problems instead of a single big problem is the key to the practical success of gAM. There also exist “descent” versions of gAM which do not completely perform marginal optimizations but take gradient steps

$$\begin{aligned} \mathbf{x}^{t+1} &\leftarrow \mathbf{x}^t - \eta_{t,1} \cdot \nabla_{\mathbf{x}} f(\mathbf{x}^t, \mathbf{y}^t) \\ \mathbf{y}^{t+1} &\leftarrow \mathbf{y}^t - \eta_{t,2} \cdot \nabla_{\mathbf{y}} f(\mathbf{x}^{t+1}, \mathbf{y}^t) \end{aligned}$$

along the variables instead.

Definition 4.4 (Marginally Optimum Coordinate). *Let f be a function of two variables constrained to be in the sets \mathcal{X}, \mathcal{Y} respectively. For any point $\mathbf{y} \in \mathcal{Y}$, we say that $\tilde{\mathbf{x}}$ is a marginally optimal coordinate with respect to \mathbf{y} , and use the shorthand $\tilde{\mathbf{x}} \in \text{mOPT}_f(\mathbf{y})$, if $f(\tilde{\mathbf{x}}, \mathbf{y}) \leq f(\mathbf{x}, \mathbf{y})$ for all $\mathbf{x} \in \mathcal{X}$. Similarly for any $\mathbf{x} \in \mathcal{X}$, we say $\tilde{\mathbf{y}} \in \text{mOPT}_f(\mathbf{x})$ if $\tilde{\mathbf{y}}$ is a marginally optimal coordinate with respect to \mathbf{x} .*

Definition 4.5 (Bistable Point). *Given a function f over two variables constrained within the sets \mathcal{X}, \mathcal{Y} respectively, a point $(\mathbf{x}, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$ is considered a bistable point if $\mathbf{y} \in \text{mOPT}_f(\mathbf{x})$ and $\mathbf{x} \in \text{mOPT}_f(\mathbf{y})$ i.e., both coordinates are marginally optimal with respect to each other.*



In the above figures, the bold solid curve plots the function $g : X \rightarrow X \times Y$ with $g(x) = (x, \text{mOPT}_f(x))$. The bold dashed curve similarly plots $h : Y \rightarrow X \times Y$ with $h(y) = (\text{mOPT}_f(y), y)$. The figure (left and middle) shows that this may happen even if the marginally optimal coordinates are unique i.e., for every \mathbf{x} there is a unique \mathbf{y} such that $\mathbf{y} = \text{mOPT}_f(\mathbf{x})$ and vice versa. *In case a function taking bounded values possesses multiple bistable points, the bistable point to which gAM eventually converges depends on where the procedure was initialized.*

gAM-style algorithms for learning latent variable models, matrix completion and phase retrieval need to pay special attention to initialize the procedure “close” to the optimum. An exception is for robust regression where the problem structure ensures a unique bistable point and so, a careful initialization is not required.

Convergence Guarantee for gAM (convex)

Theorem 4.1. *Let $f : \mathbb{R}^p \times \mathbb{R}^q \rightarrow \mathbb{R}$ be jointly convex, continuously differentiable, satisfy β -MSS in both its variables, and $f^* = \min_{\mathbf{x}, \mathbf{y}} f(\mathbf{x}, \mathbf{y}) > -\infty$. Let the region $S_0 = \{\mathbf{x}, \mathbf{y} : f(\mathbf{x}, \mathbf{y}) \leq f(\mathbf{0}, \mathbf{0})\} \subset \mathbb{R}^{p+q}$ be bounded, i.e., satisfy $S_0 \subseteq \mathcal{B}_2((\mathbf{0}, \mathbf{0}), R)$ for some $R > 0$. Let Algorithm 3 be executed with the initialization $(\mathbf{x}^1, \mathbf{y}^1) = (\mathbf{0}, \mathbf{0})$. Then after at most $T = \mathcal{O}(\frac{1}{\epsilon})$ steps, we have $f(\mathbf{x}^T, \mathbf{y}^T) \leq f^* + \epsilon$.*

Lemma 4.2. *A point (\mathbf{x}, \mathbf{y}) is bistable with respect to a continuously differentiable function $f : \mathbb{R}^p \times \mathbb{R}^q$ that is marginally convex in both its variables iff $\nabla f(\mathbf{x}, \mathbf{y}) = \mathbf{0}$.*

Definition 4.6 (Robust Bistability Property). *A function $f : \mathbb{R}^p \times \mathbb{R}^q \rightarrow \mathbb{R}$ satisfies the C -robust bistability property if for some $C > 0$, for every $(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^p \times \mathbb{R}^q$, $\tilde{\mathbf{y}} \in \text{mOPT}_f(\mathbf{x})$ and $\tilde{\mathbf{x}} \in \text{mOPT}_f(\mathbf{y})$, we have*

$$f(\mathbf{x}, \mathbf{y}^*) + f(\mathbf{x}^*, \mathbf{y}) - 2f^* \leq C \cdot (2f(\mathbf{x}, \mathbf{y}) - f(\mathbf{x}, \tilde{\mathbf{y}}) - f(\tilde{\mathbf{x}}, \mathbf{y})).$$

Convergence Guarantee for gAM (non-convex)

Theorem 4.3. *Let $f : \mathbb{R}^p \times \mathbb{R}^q \rightarrow \mathbb{R}$ be a continuously differentiable (but possibly non-convex) function that, within the region $S_0 = \{\mathbf{x}, \mathbf{y} : f(\mathbf{x}, \mathbf{y}) \leq f(\mathbf{0}, \mathbf{0})\} \subset \mathbb{R}^{p+q}$, satisfies the properties of α -MSC, β -MSS in both its variables, and C -robust bistability. Let Algorithm [3](#) be executed with the initialization $(\mathbf{x}^1, \mathbf{y}^1) = (\mathbf{0}, \mathbf{0})$. Then after at most $T = \mathcal{O}\left(\log \frac{1}{\epsilon}\right)$ steps, we have $f(\mathbf{x}^T, \mathbf{y}^T) \leq f^* + \epsilon$.*

Lemma 4.4. *Let f satisfy the properties mentioned in Theorem [4.3](#). Then for any $(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^p \times \mathbb{R}^q$, $\tilde{\mathbf{y}} \in mOPT_f(\mathbf{x})$ and $\tilde{\mathbf{x}} \in mOPT_f(\mathbf{y})$,*

$$\|\mathbf{x} - \mathbf{x}^*\|_2^2 + \|\mathbf{y} - \mathbf{y}^*\|_2^2 \leq \frac{C\beta}{\alpha} \left(\|\mathbf{x} - \tilde{\mathbf{x}}\|_2^2 + \|\mathbf{y} - \tilde{\mathbf{y}}\|_2^2 \right)$$

Software List:

Iterative Hard Thresholding:

https://www.researchgate.net/publication/280301598_Matlab_Code_for_Iterative_Hard_Thresholding_Algorithm_Based_on_Backtracking

<http://www.personal.soton.ac.uk/tb1m08/sparsify/sparsify.html>

Orthogonal Matching Pursuit:

https://www.mathworks.com/matlabcentral/fileexchange/50584-orthogonal-matching-pursuit-algorithm--omp-?s_tid=gn_loc_drop

Compressive Sampling Matching Pursuit (CoSaMP):

<https://www.mathworks.com/matlabcentral/fileexchange/32402-cosamp-and-omp-for-sparse-recovery>

Forward-backward (FoBa):

<https://github.com/pxchen95/overlappingspikes/blob/master/notimeshifts/FoBa.m>

Projected Gradient Descent (PGD):

<https://github.com/hiroyuki-kasai/NMFLibrary>

Expectation Maximization (EM):

<https://www.mathworks.com/matlabcentral/fileexchange/47889-expectation-maximization-algorithm-zip>

Singular Value Projection (SVP):

<http://www.cs.utexas.edu/~pjain/svp/>

Gerchberg-Saxton Alternating Minimization:

<https://www.mathworks.com/matlabcentral/answers/245368-gerchberg-saxton-algorithm>

Wirtinger's Flow for Phase Retrieval:

http://www-bcf.usc.edu/~soltanol/matlab/WF_StanfordQuad.m