Paper Review: Escaping from Saddle Points — Online Stochastic Gradient for Tensor Decomposition

1 Introduction

Gradient descent is a well studied, but still surprisingly versatile tool for solving optimization problems. It is the natural solution in the convex case, where the negative gradient direction is always a descent the direction, but even in the non-convex case, it can be one of the best tools we have under certain conditions. The difficulty of non-convex problems comes from the fact that critical points where $\nabla f(x) = 0$ can be local minimums, local maximums, or saddle points. Additional complexity come from the fact that the non-convex landscape can have flat regions that are not minimums and cannot be escaped with first order information alone. In the worst case, solving convex problems to even a local minimizer can be NP-hard [3]. However many real world problems have advantageous properties that allow us to avoid these issues.

In this paper, we will explore the subset of non-convex optimization problems that are so called 'strict-saddle' and where all local minimums are global. In particular, we will focus on the orthogonal tensor decomposition analysis presented in "*Escaping from Saddle Points — Online Stochastic Gradient for Tensor Decomposition*" [7]. Orthogonal tensor decomposition optimization finds a set of rank 1 tensors (which will be defined in a later section) that sum to an observed tensor. This is useful in independent component analysis [6], topic models, and community detection [2]. In order to come up with an efficient optimization method for tensor decomposition, we need to examine whether it belongs to the subset of non-convex optimization problems that are strict saddle and have only global solutions and how these properties can be used to guarantee convergence. We will also show that tweaking classic gradient descent by adding some Gaussian noise can improve the convergence rate for this problem.

2 Preliminaries

Tensors are a higher order construction than vectors or matrices and as such they are more versatile in their use. While powerful, this also means that tensor notation and definitions can be somewhat ambiguous. In this section we will clarify some of the necessary definitions we will use throughout this paper.

2.1 Tensors

A p-th order tensor is a p-dimensional array. If $T \in \mathbb{R}^{d^p}$ is a p-order tensor then the $(i_1, i_2, ..., i_p)^{th}$ element of T is $T_{i_1,i_2,...,i_p}$ where $i_j \in [1, ..., m_j]$ if m_j is the j^th dimension in the array. A simple example of a p = 3 order Tensor using python notation:

T = [[[a, b, c, d], [e, f, g, h]], [[i, j, k, l], [m, n, o, p]]]

Here, m_1 = 4, m_2= m_3 = 2, $T_{1,1,1} = a$, and $T_{1,2,3} = g$. In other words, T is two 2x4 matrices stacked along the third dimension.

Tensors can be constructed from the tensor product of vectors (outer product generalized into higher orders). The notation of this is T=$(u \otimes v)$ where T becomes a 2nd order tensor and $T_{i,j} = u_i v_j$. We can generalize this to higher orders with $(u_1 \otimes u_2 \otimes ... \otimes u_p)$. If $u_1 = u_2 = ...u_p = u \in \mathbb{R}^m$ then we can write this as: $u^{\otimes p}$. We can then create the tensor:

$$U = \left[u^{\otimes p}\right]_{i_1,i_2,...,i_p} = u_{i_1} u_{i_2} ... u_{i_p} = \prod_{j=1}^{p} u_{i_j}$$

where $m_j = m \forall j$. In this case, U can be decomposed into a single vector u and is considered a rank-1 tensor

for the rest of this paper. The goal of orthogonal tensor decomposition is to approximate a tensor as the sum of rank-1 tensors:

$$T = \sum_{i=1}^{d} a_i^{\otimes p} \tag{1}$$

where d is the rank of the T and a_i's are vectors that must have different properties depending on the mode of orthogonality. In this case, we want to look at the case were a_i's are completely orthogonal [4] and enforce $||a_i||_2 = 1$ and $a_i^T a_j = 0\ , \forall i \neq j$. These a_i's are the components of the decomposition and are unique up to sign-flips. This implies the desired property that if we find any permutation of a_i, then we have all optimal a_i's, which is one of the criteria we need for efficient optimization. Additionally, if we reorganize the $a_i s$ as the columns of an orthonormal matrix

$$A = [a_1, a_2, ..., a_d]$$

then 1 can be written as

$$T_{i_1, i_2 ..., i_p} = \sum_{k=1}^{d} \prod_{j=1}^{p} A_{i_j k} \tag{2}$$

This allows us to represent the decomposition of a tensor into an orthonormal matrix $\in \mathbb{R}^{mxd}$.

2.2 Strict Saddle

A continuously differentiable, non-convex function f(x) has critical points where gradient $\nabla f(x) = 0$ at local minimums, local maximums, and saddle points. Traditional gradient descent can get stuck in these critical points which would cause the false convergence of the algorithm at local maximums and saddle points. However, for second order methods such as the trust region approach found in [1] or newton's method, local maximums have negative curvature in every direction that can be used to find a way down. The only problematic case then is the saddle points. Strict saddle functions are those where every saddle point has at least one negative eigen value in its hessian $\nabla^2 f(x)$ and thus has a path for us to descend with second order methods. The complete definition for strict saddle problems is [7]:

> A function f:$\mathbb{R} \mapsto \mathbb{R}^n$ is $(\alpha, \beta, \gamma, \delta)$-strict saddle if $\forall x \in \mathbb{R}^n$ obeys at least on of the following:
>
> - **[Large Gradient]** $||\nabla f(x)||_2 \geq \beta$
> - **[Negative Curvature]** $\exists v \in \mathbb{S}^{n-1}$, such that $v^T \nabla^2 f(x) v \leq -\alpha$
> - **[Strong Convexity Around Minimizers]** $\exists x_*$ such that $||x - x_*||_2 \leq \delta$, and for all $y \in \mathbb{B}(x_*, 2\delta)$, we have $\nabla^2 f(y) \succcurlyeq \gamma \mathbf{I}$

These conditions ensure that every point either has a large gradient we can descend down and every non-optimal critical point has negative curvature that we can we can use to find a descent direction. If we can cast Tensor decomposition as a strict saddle problem, then we can benefit from these properties to guarantee convergence.

2.3 Stochastic Gradient Descent and its Noisy Variants

Stochastic gradient descent a modification of traditional gradient descent were the gradient is calculated from a subset of samples rather than the entire set. It is typically used in models where the amount of samples is

large and it becomes unwieldy to calculate a gradient for the entire set. In the case of non-convex optimization with the strict-saddle condition, this has the added benefit of introducing randomness into the gradient. This gives a chance to escape saddle points even in first order methods. However this may introduce some bias depending on the method of sampling, so to avoid that we can use an adjusted algorithm that adds a small amount of noise in every direction [7].

Algorithm 1 Noisy Gradient Descent

Require: Stochastic Gradient oracle SG(ω),initial point ω_0, desired accuracy κ
Ensure: ω_t that is close to some local minimum ω^*
Choose step size $\eta = min\{\tilde{O}(\kappa^2/log(1/\kappa)), \eta_{max}\}, T = \tilde{O}(1/\eta^2)$
for $t = 0$ to $T - 1$ **do**
Sample noise n uniformly from unit sphere.
$\omega_{t+1} \leftarrow \omega_t - \eta(SG(\omega) + n)$
end for

The stochastic gradient oracle SG(ω) is a sample of the gradient at point ω. If using normal first order gradient descent, SG(ω) can be replaced with the gradient of the objective function, but if using stochastic gradient descent with sample selection scheme that has non-negligible and unbiased noise, the additional noise is not needed.

This noise can be thought of as 'heat' that allows us to not get frozen near critical points. When the algorithm is not at or near saddle, the gradient of a strict-saddle function is $||\nabla f(x)||_2 \geq \beta$ and overwhelms the noise, ensuring a negative descent direction. When the function is close or at a saddle point, normal gradient descent slows down or freezes at that point. By adding heat, or random perturbations, the algorithm can quickly move away from the saddle points.

3 Optimizing Tensor Decomposition

Now that we have some conditions that guarantee convergence of non-convex problems, all we need to do is find a representation of tensor decomposition that satisfies these conditions.

3.1 Simplifying Assumptions

If we are given a tensor $T \in \mathbb{R}^{d^p}$, our objective is to find a orthogonal decomposition such that

$$T = \sum_{i=1}^{d} a_i^{\otimes p}$$

$$\text{s.t } ||a_i||_2 = 1 \text{ and } a_i^T a_j = 0 \, , \forall i \neq j$$

To avoid complexity, we will assume that all of the a_i have the same dimension m and that the rank d is a fixed constant that we assume before beginning optimization. Keeping a fixed dimension for the a_i's allows us to avoid defining any formulation for adding different sized tensors. Assuming a fixed rank d means we can formulate the problem as a minimization problem with respect to the components a_i and not to both the a_i's and d. The practical implications of fixing d for ICA, for example, is that we are assuming we know the number of components tensor decomposes into, which is reasonable. One way to generalize to an unknown d is to rerun the algorithms using different values of d and using the one with the least error.

3.2 Optimization Methods

In this section, we will investigate some different methods of optimization. By comparing different formulations, we can see the relative performance of those that are strict saddle and those that are not.

3.2.1 Reconstruction Error

The simplest formulation of the problem is to minimize the reconstruction error:

$$\min_{\forall i, ||u_i||_2^2=1} ||T - \sum_{i=1}^{d} u_i^{\otimes p}||_F^2 \tag{3}$$

Where $||\cdot||_F$ is the tensor Frobenius norm

$$||T||_F^2 = \sum_{i_1,i_2,\ldots,i_p} T_{i_1,i_2,\ldots,i_p}$$

This objective function is not necessarily strict saddle and thus does not benefit from convergence guarantees. To see this, we can implement a noisy gradient descent method with 3d space p=3.

$$\min_{\forall i, ||u_i||_2^2=1} ||T - \sum_{i=1}^{d} u_i^{\otimes 3}||_F^2 \tag{4}$$

Equation 4 is a sextic polynomial which has a difficult gradient to compute. To simplify, we can break the variable into 3 different components: a_i, b_i, and c_i then solve an alternating squares gradient descent.

3.2.2 Alternating Least Squares

For third order tensor decomposition with dissimilar component vectors

$$T = \sum_{i=1}^{d} a_i \otimes b_i \otimes c_i \tag{5}$$

we want to minimize the reconstruction error while preserving the fact that the component vectors are equal to each other

$$\min_{\forall i, ||u_i||_2^2=1} ||T - \sum_{i=1}^{d} a_i \otimes b_i \otimes c_i||_F^2 \tag{6}$$

the following optimization problem

$$\min_{\forall i, ||a_i||_2^2=||b_i||_2^2=||c_i||_2^2=1} ||T - \sum_{i=1}^{d} a_i \otimes b_i \otimes c_i||_F^2 \tag{7}$$

subject to $a_i - b_i = b_i - c_i = c_i - a_i = 0, \forall i$. The augmented Lagrangian of function of 7 is

$$\begin{aligned} &\min ||T - \sum_{i=1}^{d} a_i \otimes b_i \otimes c_i||_F^2 \\ &+ < \lambda_1, a - b > + < \lambda_2, b - c > + < \lambda_3, c - a > \\ &+ \frac{\sigma}{2}(||A - B||_F^2 + ||B - C||_F^2 + ||C - A||_F^2) \end{aligned} \tag{8}$$

if we take the gradient of this function with respect to each lagragian variable and set to zero, we find that there are critical points when $\lambda_1 = \lambda_2 = \lambda_3 = 0$ [5]. We can therefore rewrite the minimization problem as

$$\min ||T - \sum_{i=1}^{d} a_i \otimes b_i \otimes c_i||_F^2 + \frac{\sigma}{2}(||A - B||_F^2 + ||B - C||_F^2 + ||C - A||_F^2) \tag{9}$$

we can take the gradient with respect to each component matrix using $\boxdot as the Hadamard entrywise product, \odot as the Kh Raoproduct, \otimes as the Kronecker product, and the notation \mathrm{T}_{d*dxd}$ as the flattening of T along the third dimension [6]

$$\begin{aligned}
\nabla_A &= (I_A \otimes (C^T C \boxdot B^T B))vecA^T - (I_A \otimes (C \odot B))^T vecT_{d*dxd} + 2A - B - C \\
\nabla_B &= (I_B \otimes (A^T A \boxdot C^T C))vecB^T - (I_B \otimes (A \odot C))^T vecT_{d*dxd} + 2B - A - C \\
\nabla_C &= (I_C \otimes (B^T B \boxdot A^T A))vecC^T - (I_C \otimes (B \odot A))^T vecT_{d*dxd} + 2C - B - A
\end{aligned} \tag{10}$$

and perform a noisy ADMM optimization. It is not clear whether this formulation is strict saddle, and

Algorithm 2 Noisy ADMM

Ensure: A_0,B_0,C_0,$\eta = min\{\tilde{O}(\kappa^2/log(1/\kappa)), \eta_{max}\}, T = \tilde{O}(1/\eta^2)$
  for $t = 0$ to $T - 1$ **do**
    Sample noise n uniformly from unit sphere.
    $A_{t+1} \leftarrow A_t - \eta(\nabla_A + n)$
    $B_{t+1} \leftarrow B_t - \eta(\nabla_B + n)$
    $C_{t+1} \leftarrow C_t - \eta(\nabla_C + n)$
  end for
  $A = (A_T + B_T + C_T)/3$

in practice, this algorithm tends to converge to the critical point at 0 rather than an optimal value when the starting values are near 0. This makes the formulation unpredictable and not useful for practical application.

3.2.3 Correlation Minimization

Another formulation of the tensor decomposition uses an objective function that aims to minimize the correlation between different components of a fourth order tensor $T \in \mathbb{R}^{d^4}$ [7]:

$$\min_{\forall i, ||u_i||_2^2 = 1} \sum_{i \neq j} T(u_i, u_i, u_j, u_j) \tag{11}$$

where $T(u_i, u_i, u_j, u_j) = \sum_{i=1}^{d} (z_k(i))^2 (z_l(i))^2$ and the z's are the scalar multipliers of basis expansion $u_k = \sum_{i=1}^{d} z_k(i) a_i$. This implies that 11 is minimized when u_i and u_j are orthogonal and has an optimal value of 0. It can be shown that this formulation of the problem is strict saddle with a stochastic gradient oracle (see [7] Appendix C.2):

$$\nabla u_i \theta(u, y) = \sum_{i \neq j} (< u_j, u_j > u_i + 2 < u_i, u_j > u_j - < u_j, y >^2 < u_i, y > y) \tag{12}$$

where y is a sample selected from T. If we set $y = d^{1/4} a_i$ where i is uniformly distributed on [1,d], then we can see that $\mathbb{E}[y^{\otimes 4}] = T$. Using y as samples and 12 as SG(ω), we can use algorithm 1 to compare the minimization of reconstruction error with this formulation, shown in 1 Here we can see that the reconstruction error of the correlation minimization converges faster and avoids the saddle point seen around iteration 8000 when using the reconstruction error.

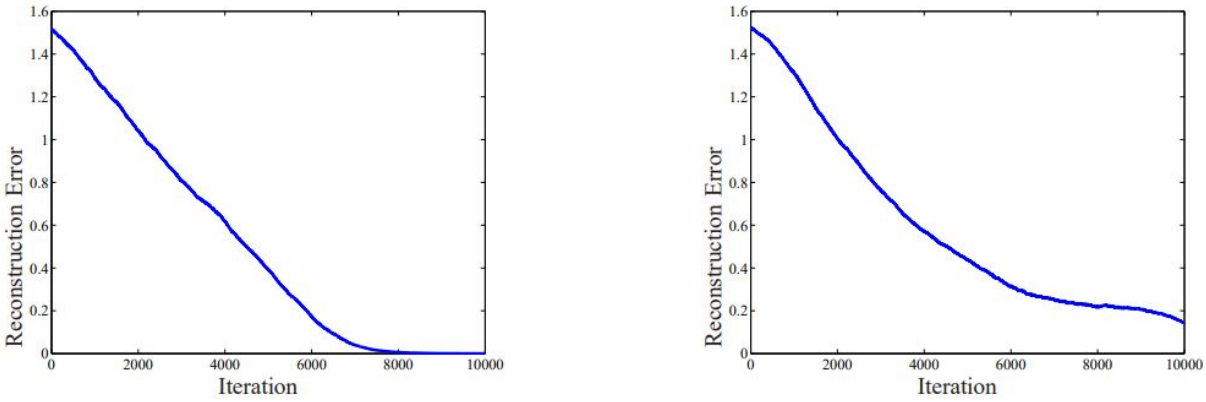

Figure 1: Comparison of Correlation Minimization (left) and Reconstruction Error (right) [7]

3.2.4 High-Order SVD (HOSVD)

Another popular approach to this problem is the Tucker decomposition [8] also known as High-Order Singular Value Decomposition. This method is a generalization of 2D SVD, were the aim is to decompose the tensor T into p orthogonal tensors Q and a diagonal weight tensor T_c [6].

$$T = (Q^{(1)}, ..., Q^{(p)}) \cdot T_c \tag{13}$$

This approach is attractive because it can be solved in closed form by itertively flattening the tensor T, taking 2D SVD to record the singular vectors U, and multiplying the hermition of U by the unflattened T. Q^i then becomes our $a_i^{\otimes p}$ tensors. This algorithm is useful for when we do not know the rank of the tensor d and also gives us the ability of rank 1 approximations of T by truncating the singular values according to Eckart-Yound Theorem. Unfortunately, we cannot actually enforce the form $a_i^{\otimes p}$ as the only requirement in the decomposition is that singular tensors are constructed from orthogonal vectors, they do not all need to be the same. Therefore the results of this method cannot be directly compared to previous methods.

4 Conclusion

From these different approaches, we can see that efficient non-convex optimization is case dependent. However, if we are able to frame our problem as a strict saddle problem, then stochastic gradient descent or noisy gradient descent is an efficient optimization method. To confirm whether this holds true in general, we can investigate the performance of noisy gradient descent in other high order problems like nueral network optimization.

References

[1] Yann Dauphin et al. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. *NIPS'14: Proceedings of the 27th International Conference on Neural Information Processing Systems*, 2:2933–2941, 2014.

[2] Niranjan U. Hakeem M. U. Anandkumar A. Huang, F. Fast detection of overlapping communities via online tensor methods. *arXiv preprint arXiv:1309.078*, 2013.

[3] S. N. Kabadi K. G. Murty. Some np-complete problems in quadratic and nonlinear programming. *Mathematical Programming*, 39(2):117–129, 1987.

[4] Tamara Kolda. Orthogonal tensor decompositions. *SIAM J. MATRIX ANAL. APPL.*, 23(1):243–255, 2001.

[5] Haixia Liu. Symmetric tensor decomposition by alternating gradient descent. *Numer Linear Algebra Appl.*, 2022.

[6] Andre L. F. de Almeida Pierre Comon, Xavier Luciani. Tensor decompositions, alternating least squares and other tales. *Journal of Chemometrics*, 23:393–405, 2009.

[7] Chi Jin Yang Yuan Rong Ge, Furong Huang. Escaping from saddle points – online stochastic gradient for tensor decomposition. *JMLR: Workshop and Conference Proceedings*, 40:1–46, 2015.

[8] L. R. Tucker. Some mathematical notes for three-mode factor analysis. *Psychometrika*, 31:279,311, 1966.