Gradient Descent for Approxmating Max-Flow

Margo Oka

May 2019

Contents

1 Introduction **2**

2 Motivation **2**

3 Necessary Background Knowledge **2**
3.1 Gradient Descent 2
3.2 Max-Flow and Min-Cut 3
3.2.1 Max-Flow 3
3.2.2 Min-Cut 3
3.2.3 Duality of Max-Flow and Min-Cut 3

4 Previous Methods **4**
4.1 Augmented Ford-Fulkerson 4
4.2 l_i Norm Minimization and the Conjugate Gradient Method . . . 4
4.3 Results 6

5 Gradient Descent for Approximating Max-Flow **7**
5.1 Benefits and Results 8

6 Conclusion **8**

1 Introduction

Determining the graph cuts is highly useful to computer vision. As it turns out we can effectively approximate graph cuts by using gradient descent. We can rewrite a maximum flow problem as a linear program and from there one can rewrite the linear program in a form that can be solved with convex optimization. Therefore, it is possible to rephrase graph cut problems in ways that can be solved and/or approximated by the conjugate gradient method and gradient descent. For many problems in computer vision rephrasing graph cut problems this way leads to great efficiencies compared to traditional methods of calculating graph cuts.

2 Motivation

Graph cuts are highly relevant to the problems of image restoration[1] and image segmentation[2].

Traditional methods of calculating graph cuts are highly serialized and do not easily lend themselves to parallelization. Rephrasing graph cut problem in terms of convex optimization allows for increased parallelization and subsequently increased speed. The popularity of GPUs for general purpose computing in the field of computer vision further motivates the search for ways to parallelize computational problems. The higher number of parallel processors in GPUs compared to CPUs makes parallelization more efficient.

Additionally, in the context of computer vision many problems that utilize maxflow lend themselves to be approximated. A "perfect" solution often is not necessary so long as it is close enough.

3 Necessary Background Knowledge

3.1 Gradient Descent

Gradient descent is an iterative optimization algorithm for determining the minimum of a function.

Gradient descent is a popular optimization strategy in machine learning and deep learning.

3.2 Max-Flow and Min-Cut

3.2.1 Max-Flow

Within a flow network you can find the maximum flow from a source s to a sink t. Finding this constitutes the max-flow problem. A **flow network** can be realized as a directed graph where the weight of an edge is its capacity. The **capacity** of an edge represents the maximum flow that can pass though that edge. The flow must respect the following two constraints:

- *capacity constraint*: the flow of an edge cannot be greater than its capacity
- *conservation of flows*: the sum of the flows entering a node must be the same as the sum of the flows exiting a node (excluding the source and sink)

The value of the flow is the sum of the the flow of all edges leaving the source or equivalently the sum of the flow of all the edges entering the sink.

The Ford-Fulkerson algorithm is a common algorithm for determining the max-flow of a flow network. The running time of the Ford-Fulkerson algorithm is $O(Mf)$ where M is the number of edges and f is the value of maximal flow.

3.2.2 Min-Cut

A **cut** of a graph is a partition of the vertices of a graph into two disjoint subsets where the subsets are connected by at least one edge. The value of a cut in this context is the sum of the weights of the edges connecting the two disjoint subsets of the graph. The **minimum cut** is the cut whose edge weight sum is minimal. Additionally, the **minimum s-t cut** of a flow network is the min-cut whose vertices s and t are in different subsets. In determining the sum of the s-t cut, only consider edge weights/capacities going from the subset containing the source to the subset containing the sink.

3.2.3 Duality of Max-Flow and Min-Cut

The maximum flow of a flow network is the same as the minimum s-t cut of the flow network's graph. Both the max-flow and min-cut problems can be realized as linear programs. It follows that the LP dual of one is the LP of the other.

4 Previous Methods

4.1 Augmented Ford-Fulkerson

An augmented version of the Ford-Fulkerson algorithm developed by Yuri Boykov and Vladimir Kolmogorov[3] has been commonly used in computer vision. In the algorithm you maintain two non-overlapping search trees S and T whose roots are at the source s and sink t, respectively.

4.2 l_i Norm Minimization and the Conjugate Gradient Method

Bhusnurmath and Taylor[4] were able to reformulate the problem of finding the mac-cut as an l_1 norm minimization.

Let $\mathbf{x} \in \mathbf{R}_m$ denote a vector representing the flow of the edges of the flow network. Positive values in the vector denote flow in a positive direction while negative values denote flow in a negative direction.

The optimization is to maximize the inner product $\mathbf{c}^T\mathbf{x}$ where $\mathbf{c} \in \mathbf{R}^m$ is a vector whose he entries are 1 for all edges starting from the source s and 0 otherwise. $\mathbf{c}^T\mathbf{x}$ computes the net flow out of s.

The constraints can be realized using an edge incidence matrix $A \in \mathbf{R}^{n\times m}$. The rows correspond to interior nodes and each column correspond to graph edges. For each column, there will be at most two non-zero entries, there will be a -1 entry in the row corresponding to the vertex at start of that directed edge, and there will be a 1 entry in the row corresponding to the vertex at the end of the edge. The columns corresponding to edges starting at s or ending at t will only have a single non-zero entry since A does not have rows corresponding to s and t.

$\mathbf{Ax} \in \mathbf{R}^n$ denotes the sum of the flows affecting each of the interior nodes due to the flow assignment $\mathbf{x}$. The constraint of the linear program is $\mathbf{Ax} = \mathbf{0}$. This constraint reflects the requirement that the net flow out of each interior node should be 0.

Additionally, let $\mathbf{w} \in \mathbf{R}^m$ represent the non-negative weights/capacities of each edge in the flow network. $-\mathbf{w} \leq \mathbf{x} \leq \mathbf{w}$ reflect the capacity constraints of each of the edges.

The linear program then is defined as follows:

$$\begin{aligned} \underset{\mathbf{x}}{\text{maximize}}: \quad & \mathbf{c}^T\mathbf{x} \\ \text{subject to}: \quad & \mathbf{A}\mathbf{x} = \mathbf{0} \\ & -\mathbf{w} \leq \mathbf{x} \leq \mathbf{w} \end{aligned}$$

Now find the dual of this linear program. Specifically, find the Lagrangian of the LP:

$$\begin{aligned} L(\mathbf{x}, \lambda, v) &= -\mathbf{c}^T\mathbf{x} + v^T\mathbf{A}\mathbf{x} + \lambda_+^T(\mathbf{x} - \mathbf{w}) + \lambda_-^T(-x - w) \\ &= -\mathbf{w}^T(\lambda_+ + \lambda_-) + \mathbf{x}^T(\mathbf{A}^T v - \mathbf{c} + \lambda_+ - \lambda_-) \end{aligned}$$

There are three sets of Lagrange multiplies: $\lambda_+ \in \mathbf{R}^m$, $\lambda_- \in \mathbf{R}^m$, and $v \in \mathbf{R}^n$.

The Langrangian dual function $g(\lambda, v)$ is obtained bu minimziaing the Lagrangian with respect to $\mathbf{x}$:

$$g(\lambda, v) = \inf_x L(x, \lambda, v)$$

The dual function is of the following form:

$$g(\lambda, v) = \begin{cases} -\mathbf{w}^T(\lambda_+ + \lambda_-) & \text{iff } \mathbf{A}^T v - \mathbf{c} = (\lambda_- - \lambda_+) \\ -\infty & \text{otherwise} \end{cases}$$

You can compute the optimal value of the original primal problem by maximizing the corresponding Lagrangian dual function $g(\lambda, v)$. This gives the following dual problem:

$$\begin{aligned} \underset{\lambda, v}{\text{minimize}}: \quad & \mathbf{w}^T(\lambda_+ + \lambda_-) \\ \text{subject to}: \quad & \mathbf{A}^T v - \mathbf{c} = (\lambda_- - \lambda_+) \\ & \lambda_+ \geq 0, \lambda_- \geq 0 \end{aligned}$$

This implies that the value of the inner product $\mathbf{w}^T(\lambda_+ + \lambda_-) = \sum_{i=1}^{m} \mathbf{w}_i(\lambda_+ + \lambda_-)_i$ will be minimized when $(\lambda_+ + \lambda_-)_i = |(\mathbf{A}^T v - \mathbf{c})_i|$. It follows that the optimization of the dual problem can be written as follows:

$$\min_v \sum_{i=1}^{m} w_i |(\mathbf{A}^T v - \mathbf{c})_i|$$

This can be rewritten as follows:

$$\min_{v} ||\text{diag}(\mathbf{w})(\mathbf{A}^T v - \mathbf{c})||_1$$

This optimization is now an *unconstrained l_1 norm minimization*. The decision variables correspond to the Lagrange multipliers $v \in \mathbf{R}^n$.

With this formulation we have a connection between graph cuts and convex optimization. We can now apply continuous optimization techniques to solve problems related to graph cuts.

However, it important to note that the optimization problem is not smooth. This problem can be solved by using the interior point method with logarithmic barrier potentials. Using this strategy we can create the a convex objective function from the original linear program.

First, note that the unconstrained l_1 norm minimization problem can be rewritten as a linear program as follows:

$$\begin{aligned} \text{minimize} &: \mathbf{w}^T \mathbf{y} \\ \text{subject to} &: \begin{bmatrix} \mathbf{A}^T & -\mathbf{I} \\ \mathbf{A}^T & -\mathbf{I} \end{bmatrix} \begin{bmatrix} v \\ \mathbf{y} \end{bmatrix} \leq \begin{bmatrix} \mathbf{c} \\ \mathbf{-c} \end{bmatrix} \end{aligned}$$

$\mathbf{y} \in \mathbf{R}^m$ is an auxiliary variable where $\mathbf{y} \geq (\mathbf{A}^T v - c)$ and $\mathbf{y} \geq -(\mathbf{A}^T v - c)$. Using the interior point method with logarithmic barrier potentials we get the the following convex objective function:

$$\phi(v, \mathbf{y}) = t(\mathbf{w}^T \mathbf{y}) - \sum_{i=1}^{m} \log(\mathbf{y}_i - \mathbf{z}_i) - \sum_{i=1}^{m} \log(\mathbf{y}_i + \mathbf{z}_i)$$

where $\mathbf{z} = (\mathbf{A}^T v - \mathbf{c})$. t is a scalar used to weight the original objective function against the barrier potentials associated with the linear constraints.

4.3 Results

With the unconstrained l_i norm minimization Bhusnurmath and Taylor and in the context of image restoration used the barrier method to solve it. The resulting l_i norm minimization problem can be reduced to the problem of solving a sequence of sparse linear systems involving the graph Laplacian. From here they used the conjugate gradients method. Linear systems of this form have been studied extensively so there exist many effective techniques for solving them.

Moreover, in context of image restoration there are useful properties that arise from this technique. The node weights v converge to binary values at the global minimum so the optimization procedure yields the node label immediately and the weights tend toward discrete values so it's easy to employ rounding as the barrier method approaches convergence. Therefore, in practice you can use single precision floating point to use this technique. Additionally, the whole procedure is performed using vector operations. This is easily parallelizable.

5 Gradient Descent for Approximating Max-Flow

Yildiz and Akgul[5] build upon the findings of Bhusnurmath and Taylor in order to use gradient descent for approximating graph cuts.

Bhusnurmath and Taylor gave the minimum cut formulation as the minimization of the following function

$$F(\mathbf{v}) = \sum_{i=1}^{m} w_i |(\mathbf{A}^T \mathbf{v} - \mathbf{c})_i|$$

Using the method described by Bhusnurmath and Taylor $\mathbf{A}$ can become very large for reasonable graphs and application of Newton's method can be complicated. Yildiz and Akgul found a method that utilizes gradient descent. Their method does not require $\mathbf{A}$ to be built and works effectively on large graphs.

The gradient descent algorithm to minimize $F(x)$ is given in the following algorithm:

Algorithm 1 Minimize F(x) Using Gradient Descent

1: $x =$ initial solution
2: **repeat**:
3: Calculate derivaties $\Delta x = \frac{\partial F}{\partial x}$
4: Choose a proper β parameter
5: Update $x : x = x - \beta \Delta x$
6: **until** *convergence*

It is important to not that $F(x)$ is not a smooth function because of the absolute value. This means that we cannot quite use gradient descent. This problem was solved by using the interior point method with logarithmic barrier potentials by Bhusnurmath and Taylor, but you can also solve this problem by smoothing $F(x)$ directly as follows:

$$F(\mathbf{v}) = \sum_{i}^{n} (s_i |1 - v_i| + t_i |v_i| + \sum_{j \in N(i)} w_{ij} |v_i - v_j|)$$

and approximate the absolute value function by smoothing it with some small μ:

$$|v_i| = \sqrt{v_i^2 + \mu}$$

With this approximation of the absolute value function is now differential and solvable using gradient descent.

5.1 Benefits and Results

This approach requires less memory than the previous approach outlined by Bhusnurmath and Taylor. It also allows for more parallelization as the derivative calculation is local and parallelizable. Additionally, the step factor β in Algorithm 1 is chosen at each iteration such that the update in the given direction is as large as possible while ensuring that $\mathbf{v}$ is feasible, or that $v_i \in [0, 1]$. However, $\mathbf{v}$ need not be strictly within this interval as it converges eventually. It is noteworthy however that this constraint dramatically speeds up convergence.

The time complexity of this algorithm is the number of iterations multiplied by $O(n)$ for a grid graph. The memory complexity is linear in number of vertices in the grid graph.

Using gradient descent requires less memory than other techniques and there are many well-known parallel processing techniques such as red-black ordering and the SOR that one can use. Additionally, in the context of computer vision 2D grid graphs are extremely common and using red-black ordering resulting in a dramatically decreased number of iterations.

6 Conclusion

Reformulating graph cut problems as convex optimization problems leads to increased efficiency in runtime and memory. When using Ford-Fulkerson the serialization of the process bottle necked the speed. Turning the problem into a gradient descent problem allows for greater opportunity for parallelization. With the rise of GPUs this becomes high efficient.

References

[1] Yuri Boykov, Olga Veksler, and Ramin Zabih. *Fast Approximate Energy Minimization via Graph Cuts.* PAMI, 23(11), Nov 2001.
`http://www.cs.cornell.edu/rdz/Papers/BVZ-iccv99.pdf`

[2] Y. Boykov and M. Jolly, *Interactive Graph Cuts for Optimal Boundary and Region Segmentation of Objects in N-D Images*, Proc. Eighth Int'l Conf. Computer Vision, pp. 105-112, 2001
http://www.csd.uwo.ca/ yuri/Papers/iccv01.pdf

[3] Yuri Boykov and Vladimir Kolmogorov. *An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision.* PAMI, Vol. 26, No. 9, pp. 1124-1137, Sept 2004.
http://www.csd.uwo.ca/ yuri/Papers/pami04.pdf

[4] Arvind Bhusnurmath and Camillo J. Taylor. *Solving the Graph Cut Problem vial1 Norm Minimization.* Technical Reports (CIS), January 2007.

[5] Alparslan Yildiz and Yusuf Akgul. *A Gradient Descent Approximation for Graph Cuts.* Pattern recognition. 31st DAGM symposium, Jena, Germany, September 9–11, 2009. Proceedings, 312-321, 2009.
https://link.springer.com/chapter/10.1007/978-3-642-03798-6_32