An Expanded Description of the TD-DO Optimization Algorithm

J. Zico Kolter
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, MA 02139
kolter@csail.mit.edu

This supplementary document describes the optimization procedure for the TD-DO algorithm in greater detail. In particular, we are interested in solving the optimization problem

$$\min_d \sum_{i=1}^m -\hat{p}_i \log d_i \quad \text{s.t. } , d_i \geq 0, \ \forall i, \ \ 1^T d = 0 \ \ \hat{F}(d) \succeq 0. \tag{1}$$

where $d \in \mathbb{R}^m$ is the optimization variable, $\hat{p}_i$ is given by the problem data,

$$\hat{F}(d) = \sum_{i=1}^m d_i \begin{bmatrix} \phi(s^{(i)})\phi(s^{(i)})^T & \phi(s^{(i)})\phi(s'^{(i)})^T \\ \phi(s'^{(i)})\phi(s^{(i)})^T & \phi(s^{(i)})\phi(s^{(i)})^T \end{bmatrix} \equiv \sum_{i=1}^m d_i \hat{F}_i \tag{2}$$

and where for simplicity we will ignore any additional constraints $d \in \mathcal{C}$. If we want to impose additional constraints on d, then the algorithm below must be modified to account for such constraints, but for many classes of constraints this can be done in a straightfoward manner. We will discuss this point in greater detail below.

We begin by forming the Lagrangian of (1), with Lagrange multiplier $Z \in \mathbb{R}^{2k \times 2k}$ for the constraint $\hat{F}(d) \succeq 0$, $\nu \in \mathbb{R}$ for the constraint $1^T d = 1$, and by noting we can drop the constraint $d \geq 0$, since $-\log d_i \to \infty$ as $d_i \to 0$

$$\mathcal{L}(d, Z, \nu) = \sum_{i=1}^m -\hat{p}_i \log d_i - \mathrm{tr}(Z^T \hat{F}(d)) + \nu(1^T d - 1). \tag{3}$$

The dual problem is them simply

$$\max_{Z \succeq 0, \nu} \min_d \mathcal{L}(d, Z, \nu). \tag{4}$$

Optimizing over ν and d

We solve the maximization over ν and minimization over d (the optimization over Z will be performed seperately) using an equally-constrained, feasible start Newton method. For fixed Z, this subproblem is given by

$$\max_\nu \min_d \sum_{i=1}^m -\hat{p}_i \log d_i + c^T d + \nu(1^T d - 1) \tag{5}$$

where $-\mathrm{tr}(Z^T \hat{F}(d)) = c^T d$ for $c_i = -\mathrm{tr}(Z^T \hat{F}_i)$. The equality-constrained Newton step Δd_{nt} for this optimization problem (see [1, pg 526] is given by a solution to the linear equation

$$\begin{bmatrix} \nabla_d^2 f(d) & 1 \\ 1^T & 0 \end{bmatrix} \begin{bmatrix} \Delta d_{\mathrm{nt}} \\ \mu \end{bmatrix} = \begin{bmatrix} -\nabla_d f(d) \\ 0 \end{bmatrix} \tag{6}$$

where μ is the Lagrange multiplier for the equality contraint in the Newton update (i.e., $1^T \Delta d_{\rm nt} = 0$, which implies $1^T(d+\alpha\Delta d_{\rm nt}) = 1$ for any $1^T d = 1$) and where $\nabla_d^2 f(d)$ and $\nabla_d f(d)$ are the Hessian and gradient w.r.t d of the objective

$$f(d) \equiv \sum_{i=1}^m -\hat{p}_i \log d_i + c^T d. \tag{7}$$

Crucially, since the objective (7) is decomposable over d_i, the Hessian term is diagonal, which allows for solving the Newton step very efficiently; in particular, the gradient and Hessian are given by

$$\left(\nabla_d f(d)\right)_i = -\frac{\hat{p}_i}{d_i} + c_i, \quad \left(\nabla_d^2 f(d)\right)_{ii} = \frac{\hat{p}_i}{d_i^2}. \tag{8}$$

We solve the Newton update equation (6) by block elimination (see [1, pg 456, Algorithm 10.3]), giving

$$\begin{aligned} \mu &= -(1^T \nabla_d^2 f(d)^{-1} 1)^{-1} 1^T \nabla_d^2 f(d) \nabla_d f(d) \\ \Delta d_{\rm nt} &= -(\nabla_d^2 f(d))^{-1}(\nabla_d f(d) + 1\mu) \end{aligned} \tag{9}$$

which, after, simplification, gives

$$\begin{aligned} \mu &= \frac{\sum_{i=1}^m \left(d_i + \frac{c_i d_i^2}{\hat{p}_i}\right)}{\sum_{i=1}^m \frac{d_i^2}{\hat{p}_i}} \\ (\Delta d_{\rm nt})_{ii} &= \frac{\frac{\hat{p}_i}{d_i} - \mu - c_i}{\frac{\hat{p}_i}{d_i^2}}. \end{aligned} \tag{10}$$

We use a simple backtracking line search for the Newton update. Although the number of iterations of Newton's method may very, since we are solving this optimization problem repeatedly for many nearby values of Z, we can warm-start the method from the solution at previous iterations, and so in practice optimizing over x and ν also takes time $O(m)$.

Optimizing over Z

Now we return to the optimization over Z, which we write as

$$\max_{Z \succeq 0} g(Z) \equiv \max_{Z \succeq 0} \left\{ \sum_i -\hat{p}_i \log d_i^\star(Z) - {\rm tr} Z^T \hat{F}(d^\star(Z)) + \nu^\star(Z)(1^T d^\star(Z) - 1) \right\} \tag{11}$$

where $d_i^\star(Z)$ and $\nu^\star(Z)$ denote the optimal values of d_i and ν determined by the Newton procedure above (since the c term depends on Z, these optimal values will of course depend on Z as a well).

The gradient of the dual objective with respect to Z is simply

$$\nabla_Z g(Z) = -\hat{F}(d^\star(Z)); \tag{12}$$

all the $\nabla_Z d_i^\star(Z)$ and $\nabla_Z \nu^\star(Z)$ terms drop out, a fact that can be shown by considering the partial derivatives

$$\begin{aligned} \frac{\partial g(Z)}{\partial Z} &= \frac{\partial \left(\sum_i -\hat{p}_i \log d_i^\star(Z)\right)}{\partial d^\star(Z)} \frac{\partial d^\star(Z)}{\partial Z} - \hat{F}(d^\star(Z))^T - \frac{\partial {\rm tr}(Z^T \hat{F}(d^\star(Z)))}{\partial d^\star(Z)} \frac{\partial d^\star(Z)}{\partial Z} + \\ &\quad \frac{\partial \nu^\star(Z)}{\partial Z}(1^T d^\star(Z) - 1) + \frac{\partial \nu^\star(Z)(1 d^\star(Z) - 1)}{\partial d^\star(Z)} \frac{\partial d^\star(Z)}{\partial Z} \\ &= \left(\frac{\partial f(d^\star(Z))}{\partial d^\star(Z)} + 1\nu^\star(Z)\right) \frac{\partial d^\star(Z)}{\partial Z} - \hat{F}(d^\star(Z))^T + \frac{\partial \nu^\star(Z)}{\partial Z}(1^T d^\star(Z) - 1) \\ &= -\hat{F}(d^\star(Z))^T \end{aligned} \tag{13}$$

using the fact that $\nabla_d f(d^\star(Z)) + \nu^\star(Z) 1 = 0$, an optimality condition of the optimization problem (5), and that fact that $1^T d^\star(Z) = 1$.

At this point, we use use a projected gradient method to optimize the dual objective; however, projecting onto the positive semidefinite cone requires $O(k^3)$ operations at each point to compute the eigenvalue decomposition of the currente Z iterate. Instead, we use a recent technique from semidefinite optimization [2], factor Z as $Z = YY^T$, and optimize the (now non-convex) objective $g(YY^T)$. The gradient of the objective with respect to Y is simply

$$\nabla_Y g(YY^T) = \nabla_Z g(YY^T) = -2\hat{F}(d^\star(YY^T))Y \tag{14}$$

and note that we can now omit the semidefinite constraint, since a Z of this form will always be positive semidefinite. Thus, this is just an unconstrained optimization problem, solvable by off-the-shelf methods such as LBFGS. Crucially, although the optimization problem is non-convex, it has been shown that local optimization will stll converge to the global solution of the original problem provided Y is chosen to be sufficient rank to represent the true solution $Z^\star$ [2]. Thus, choosing $Y \in \mathbb{R}^{2k\times 2k}$ will guarantee that we find the optimal solution; of course, the complexity of this update would still be $O(k^3)$ owning to the matrix multiplication, even though it may be faster due to the fact that matrix multiplication is typically faster than an eigenvalue decomposition.

The real benefit of this approach comes from the fact that in practice we often find that the optimal Z is *low-rank*: the KKT conditions of semidefinite programming imply that $\hat{F}(d^\star)$ and $Z^\star$ will have complementary ranks, and we often expect $\hat{F}(d^\star)$ to be nearly full rank in practice. This is not a mathematically precise statement, since the rank of $\hat{F}(d^\star)$ will of course depend on the MDP and value function features. However, in practice we expect $F(d^\star)$ to be close to full rank: recall that when states are sampled on-policy, $\hat{F}(d^\star) \succeq 0$ (and indeed, $\hat{F}(d^\star) \succ 0$ except on a subspace for constant-valued features). Empirically, $\hat{F}(d)$ tends to have only a few negative eigenvalues for when computed for an off-policy distribution d, and thus we expect Z to have small rank, just enough to "force" these eigenvalues to be zero. We can thus use $Y \in \mathbb{R}^{2k\times p}$ with $p \ll 2k$, which lets us compute $\nabla_Z g(YY^T)$ in time $O(mkp)$.

Although it is difficult to bound the rank p a priori, we have found that very small values $p = 2$ were sufficient in all our experiements. Importantly, however, we can always check a solution to see whether the chosen p was sufficient: if we choose p to be one larger than the presumed rank of $Z^\star$, we can check to ensure that the resulting $Y^\star$ is rank-deficient (to within the numerical precision of the optimization); if this is the case, then the chosen p was sufficient and the solution is optimal.

Additional constraints on d

The above discussion assumed no addition constraints $d \in \mathcal{C}$. As mentioned in the text, we do need to ensure that $|d_i - d_j| \to 0$ as $\|\phi(s_i) - \phi(s_j)\| \to 0$. This can be done (as we do in the experimental sections on sampling) by clustering states so that multiple states must be assigned the same probability d_i; this effectively reduces the number of d_i variables we need to consider, and always us to perform the optimization without any additional constraints on the d_i's. However, if additional constraints *are* desired, such as requiring that $|d_i - d_j|$ be bounded by some function of $\|\phi(s_i) - \phi(s_j)\|$, then this constraint $d \in \mathcal{C}$ would need to be included in the optimization of the subproblem (5). Fortunately, the number of such constraints woudl typically be small, because only small values of $\|\phi(s_i) - \phi(s_j)\|$ need to be constrained in this manner. Thus, the additional constraints are the d terms would typically be sparse, and so could be incorporated into the Newton update with little additional difficult. For the finite sampling results in the paper, we use clustering to define a small number of variables d_i that each correspond to a large number of samples; in this case we can solve the optimization problem directly, and do not need to worry about any additional constraints on d.

References

[1] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[2] M. Journee, F. Bach, P.A. Absil, and R. Sepulchre. Low-rank optimization on the cone of positive semidefinite matrices. *SIAM Journal on Optimization*, 20(5):2327–2351, 2010.