4
Practical Optimization: Constraints and Games

4.1 Introduction to Games and Constraints

This lecture focuses on a class of techniques for managing constraints in optimization problems that arrive either in machine learning or control. The below set of techniques are ones we've found quite helpful in practice.

There are a few general classes of approach to managing constraints and the details of which is "best" depends on requirements on speed and on quality of the required constraint satisfaction and final cost. There are strong inter-relations between them, but certain ones are usually preferred. The lecture below assumes an equality constraint, but each result can be derived for inequality constraints.

4.2 Reparameterization

Arguably the "simplest" technique, when viable, is reparameterization—*i.e.* reformulate the variables so the constraint **must** be satisfied. We might call that the *physics way*: if temperature is empirically lower bounded, re-expresses in $1/T$, or if velocity is upper-bounded, re-express dynamics to enforce relativistic constraints. This is also the critical technique of the beautiful theory of Generalized Linear Models, where, *e.g.*, we express binomial probabilities as unconstrained logits pushed through a logistic function. It's also fundamental to the "shooting" methods of trajectory optimization and control like iLQR, where dynamic constraints are explicitly formulated by forwarding simulating controls (that may exist only to satisfy the reparameterization).

Reparameterization naturally allows us to express uncertainty over, e.g. controls, because we can use the Laplace approximation in the reparameterization to get meaningful uncertainty estimates. [1] For instance, in a control application, if we reparameterize a positive-only velocity components as $\text{speed} = e^{\text{log\_speed}}$, we can produce posterior confidence intervals on speed that satisfy the constraints as well by passing through the reparameterization. Reparameterization equally allows feedback gains in techniques like iLQR to continue to be useful.

Slow convergence can result as we get close (or initialize unfortunately close), as the effect of reparameterization (and a cost on that variable) becomes like a barrier function and leads to very ill-conditioned problem and

[1] B. D. Ziebart, J. Andrew Bagnell, and A. K. Dey. Modeling interaction via the principle of maximum causal entropy. In *Proceedings of the 27th International Conference on Machine Learning*, 2010

effectively cost "plateaus". We have seen this behavior as a general issue with interior point methods, although that observation is a frequent point of contention. Also, for optimal control problems, reparamertization typically buries more non-linearity in the state transition dynamics and this may be ignored by some methods (e.g. iLQR, in contrast with classical Differential Dynamic Programming).

4.3 Lagrange (Primal-Dual) Methods

We'll begin by considering the problem of minimizing $f(x)$ subject to the constraint $g(x) = 0$.

Lagrange methods convert constraints into a saddle point problem– *i.e.* a game where the "dual" (multiplier player) attempts to take advantage of any violation of constraints. We must find solutions where the dual player can't win, and if we do, we guarantee satisfying the constraints. A surprising range of practical methods for constrained optimization look like "apply some simple optimization strategy to the dual problem". [2]

[2]

Gradient/ExpGrad on multipliers

Defining the dual:

$$D(\lambda) = \min_x f(x) - \lambda^T g(x),$$

it's straightforward to take

$$\max_\lambda D(\lambda)$$

and simply compute sub-gradients (assuming, e.g. $f(x)$ is convex, otherwise we're estimating a still more general notion of derivative and an application of Danskin's theorem) of D. Note this is "easy" as $\nabla_\lambda \min_x L(x, \lambda) = \nabla_\lambda L(x^*, \lambda)$ where $x^* = \text{argmin}\, f(x) - \lambda^T g(x)$. (Where this is being evaluated at some specific λ!)

We can then simply run our favorite derivative based optimizer (heavy-ball momentum, sub-gradient, Exponentiated Gradient Descent, etc.) on this dual. We note the problem may very well *not* be strongly concave in λ: that is, we may have essentially no curvature in λ– necessitating slow step size decreases, controlled/stable optimization in general, and potentially even uphill steps. [3]

[3] We note in passing the the subgradient method is equivalent to two related, natural algorithms: a) Iteratively linearizing the objective function and providing a shrinking regularization to 0. b) Iteratively linearizing the objective function and regularizing to the previous location.

These are special cases of two different, stable, optimization strategies (a) FTRL (Follow the Regularized Leader) and (b) Mirror Descent (Proximal Descent/Trust Region) that are popular in game solving and online learning, and under linearization happen to collapse to the same classical gradient descent.

To see (b), note:

$$\text{argmin}_x \nabla f(x_{\text{last}})^T x + \frac{\alpha}{2}||x - x_{\text{last}}||^2$$

is solved by,

$$x = x_{\text{last}} - \frac{1}{\alpha}\nabla f(x_{\text{last}}),$$

which is gradient descent with learning rate $\frac{1}{\alpha}$.

Regularized Lagrangian Optimization / Penalty Methods

A natural way to frame the optimization problem in light of the potential instability in optimizing the dual is to consider the regularization (see side note) that is implicit in gradient descent. That is, we can consider the *Regularized Lagrangian* (ReLa):

$$max_\lambda D(\lambda) + \frac{\alpha}{2}\lambda^2$$

This form ensures compact level sets for the optimization and lets us actually solve the problem without solutions in λ running away to infinity.

I'll assume below that with regularization we can safely assume strong duality for the problem. (That's very much unnecessarily strong, as we can converge to a dual solution anyway, but it saves time/thought.)

Given strong duality, we can swap the min and max and solve the dual:

$$\min_x \max_\lambda f(x) - \lambda^T g(x) - \frac{\alpha}{2}\lambda^2$$

But note that the inner maximization is now *closed-form*. That is, if compute ∇_λ, we get back:

$$\lambda = -\frac{1}{\alpha} g(x)$$

Substituting in, we can now eliminate λ and solve the unconstrained problem:

$$\min_x f(x) + \frac{1}{2\alpha} g(x)^2$$

That's particularly interesting, as what's classically known as a *penalty method* arises as simply the optimal solution of a regularized version of the Lagrangian; penalty methods are Lagrange methods in disguise. Shrinking the Lagrange regularization is simply scheduling the penalty, and we can see that this is a sound method, and can even compute the sub-optimality of the constraints from it. Finally, it provides an estimate of the dual variables directly via the relation $\lambda = -\frac{1}{\alpha} g(x)$.

The only reason not to take regularization to 0 is that we have poor conditioning and extremely large gradients that which implies optimization methods are likely suffer in the way barrier methods do— but it's super fast and easy to implement.

Note further that we can regularize the Lagrange multipliers in many ways– for instance, by (unnormalized) entropy regularization. Each leads to a dual penalty method (in this case an exponential penalty method). Performance of these depends a great deal on the underlying optimizer as well as our prior beliefs on the multipliers λ. If the Lagrange multipliers are expected to have small L_2 norm, we would expect classical squared-norm regularization to be good. Entropy regularization, by contrast, we might expect will work well when there are many constraints, but a sparse set of multipliers are sufficient to satisfy the constraints. Adapting the literature on such dual penalty methods is likely valuable to taking best advantage of these techniques.

L_∞ ReLa / "Exact" Penalty

An interesting exercise is to consider the Regularized Lagrangian using the $\alpha \max_i |\lambda_i|$ as the regularizer. A quick calculation reveals that this leads to a *penalty method* that is solving:

$$\min_x f(x) + \frac{1}{\alpha}|g(x)|$$

This form is related to that given by the Support Vector Machine primal, where the constraint we attempt to enforce correct labelings. The classic hinge-loss here arises from an inequality exact penalty method on achieving the correct label, although this is not usually how the method is described. In practice, non-trivial regularization of the dual variables in an SVM is assumed to prevent over-fitting.

Note however, that this form, while excellent for sub-gradient based optimizers (AdaGrad, etc.) popular in machine learning, fails to be strongly convex (or smooth) and thus is poorly optimized by Newton-style methods. The advantage of this method is that a *finite value* of $\frac{1}{\alpha}$ is sufficient to achieve the constraint precisely in contrast with squared norm penalty approach.

Augmented Lagrangian Optimization / Mirror Descent

Using Mirror Descent, where we iteratively regularize around the last solution for the dual variable rather than around 0, provides a powerful generalization of penalty methods. We begin with an estimate of the Lagrange multipliers, λ_0. As with ReLa above, we consider a regularized dual of the Lagrangian:

$$\min_x \max_\lambda f(x) - \lambda^T g(x) - \frac{\alpha}{2}(\lambda - \lambda_i)^2$$

Note again that the inner optimization (with x fixed) is closed form (in terms of the next λ to choose):

$$\lambda = \lambda_i - \frac{1}{\alpha} g(x)$$

and by substitution we can eliminate λ and solve the regularized dual as a simpler optimization in x:

$$\min_x f(x) - \lambda_i^T g(x) + \frac{1}{2\alpha} g(x)^2$$

In contrast with the penalty method, we can simply iterate. We update λ with

$$\lambda_{i+1} = \lambda_i - \frac{1}{\alpha} g(x),$$

and re-solve the Augmented Lagrangian (AuLa) [4] problem above.

[4]

It's interesting to ask why we might bother with this form, when the penalty method is itself sufficient. A general theme in optimization is that it can be more efficient to phrase a problem as a saddle-point-finding exercise *rather* than as a difficult, pure optimization. AuLa takes the penalty method approach and moves it back towards a game, managing to inherit both the simplicity of each as well as better conditioning than the single optimization solved in a penalty method. The general strategy of replacing a hard optimization problem with a game-derived sequence of such is a powerful area of research and the more detailed variant of that, the strategy of regularizing the dual parameters [5] to ensure closed form solution is one that seems not fully exploited in the literature.

[5]

Inequality Variant of Mirror Descent on Lagrangian

Perhaps the classic way to handle inequalities in the Lagrange methods above is to add a slack variable with a bound constraint $z \geq 0$. However, we can actually derive variants directly using the technique above as well.

We'll begin by considering the problem of minimizing $f(x)$ subject to the constraint $g(x) \geq 0$. Following the logic above and applying duality, we can form a regularized or augmented Lagrangian as:

$$\min_x \max_{\lambda \geq 0} f(x) - \lambda^T g(x) - \frac{\alpha}{2}(\lambda - \lambda_i)^2$$

Solving for λ (note this happening component-wise in λ and I'm glossing over that) gives us,

$$\lambda = \max(0, \lambda_i - \frac{g(x)}{\alpha})$$

Eliminating λ, we have, again component-wise, but I'm too lazy to index into λ and thus I'm treating it like it's a scalar/single constraint: If $(\lambda_i - \frac{g(x)}{\alpha}) >$

0, we are minimizing

$$f(x) - \lambda_i^T g(x) + \frac{g(x)^2}{2\alpha}$$

and otherwise,

$$f(x) - \frac{\alpha \lambda_i^2}{2}$$

This result is continuous and differentiable, but it is not twice so, so **YMMV**. The primal problem can also have achieve a minima with a negative objective value, which is somewhat annoying. Again we iterate the algorithm by re-estimating

$$\lambda_{i+1} = \max(0, \lambda_i - \frac{g(x)}{\alpha})$$

and solving again for x.

4.4 Projected Gradient

A remarkably simple method, which is often very useful in machine learning, is to take a gradient step and then project (in the sense of finding the nearest location in the constraint set in Euclidean norm) onto the constraint set. If constraints are simple and convex, this method will *accelerate* convergence relative to the unconstrained problem, at least for convex losses. [6] This is because we guarantee each projection step takes us closer to the optima in the set.

This method fundamentally relies upon the projection step being simple: for instance onto a unit ball or with bound constraints. Both of these are trivially implemented as normalization and thresholding, respectively. These style of constraints are quite common on ML applications like support vector machines (SVMs) [7]. Unfortunately, the simplicity of the *Projected Gradient* approach is compromised when using methods (for instance, Gauss-Newton or AdaGrad) that rely on a *non-trivial metric*: the projection must take place using the same metric (*i.e.* the same notion of closeness) to ensure good behavior. One can convince one self of counter-examples to proper convergence using Euclidean projection combined with Newton steps.

[6] M. Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th International Conference on Machine Learning (ICML)*, 2003

[7] N. Ratliff, J. A. Bagnell, and M. Zinkevich. (Online) subgradient methods for structured prediction. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2007

4.5 Related Reading

[1] Bertsekas, D. Nonlinear Programming: 3rd Edition. Athena Scientific, 2016.

[2] Toussaint, M. *A tutorial on Newton methods for constrained trajectory optimization and relations to SLAM, Gaussian Process smoothing, optimal control, and probabilistic inference.* In Geometric and Numerical Foundations of Movements, Springer, 2017.