**2023-05-03**

# 1 Related problems

The topic of today is three related issues, all of which fall under the heading of "optimization with noise" for some interpretation:

- What do we do when our objective function is uncertain (where the uncertainty may or may not be random)?
- What do we do when our objective function is expressed in terms of (statistics of) random variables?
- How can we use randomization to more efficiently solve deterministic optimization problem?

These three topics *are* closely related, but if we want to keep a full range of options open, it is worthwhile making a distinction between them.

# 2 Uncertain objectives

Suppose we want to minimize $\phi$, but only have access to an approximation $\hat{\phi}$. What can we hope to do? Can we approximate a minimizer of $\phi$ by a minimizer of $\hat{\phi}$? How close are the minimum values? The answer to all these questions, naturally enough, is "it depends." But we will see what we can say in a few common cases.

## 2.1 Bounded error

Let's warm up with a case where we know very little. Suppose both $\phi$ and $\hat{\phi}$ are continuous and $\|\phi - \hat{\phi}\|_\infty \leq \epsilon$. Let $x_*$ be a (local) minimizer of $\hat{\phi}$, and suppose that for some $\delta$, $\hat{\phi}(x) - \hat{\phi}(x_*) > 2\epsilon$ when $\|x - x_*\| = \delta$. Then we must have that

$$\phi(x) > \hat{\phi}(x) - \epsilon > \hat{\phi}(x_*) + \epsilon > \phi(x_*),$$

and so the minimum of $\phi$ over the ball $\mathcal{B}_\delta(x_*) = \{x : \|x - x_*\| \leq \delta\}$ must occur somewhere on the interior. Therefore, $\phi$ must have at least one local minimizer within $\delta$ of $x_*$.

How do we get the type of lower bound needed to pull off an argument like this? If $\hat{\phi}$ is $C^2$ and $\hat{H}_\phi(x_*)$ is positive definite, we can get an estimate using the Taylor approximation

$$\hat{\phi}(x) - \hat{\phi}(x_*) \approx \frac{1}{2}(x - x_*)^T H(x - x^*) \geq \frac{1}{2}\|x - x_*\|^2 \lambda_{\min}(H)$$

which suggests a radius of

$$\delta \approx 2\sqrt{\frac{\epsilon}{\lambda_{\min}(H)}}.$$

This estimate is pretty good when we are in a regime where the second-order Taylor expansion describes everything well. But if we want to go out further, we can be more precise when we have a Lipschitz condition like

$$\|H_{\hat{\phi}}(x) - H_{\hat{\phi}}(y)\|_2 \leq M\|x - y\|.$$

This would imply that

$$\hat{\phi}(x) - \hat{\phi}(x_*) \geq \frac{1}{2}\|x - x_*\|^2(\lambda_{\min}(H_{\hat{\phi}}(x_*)) - M\|x - x_*\|),$$

and so the argument holds whenever

$$\frac{1}{2}\delta^2(\lambda_{\min} - M\delta) > 2\epsilon,$$

for which there is a satisfactory $\delta < (2\lambda_{\min})/(3M)$ whenever $\lambda_{\min} > \sqrt{18M\epsilon}$.

## 2.2 Bounded gradient errors

Now suppose that we know that $\hat{\phi}$ and $\phi$ are both differentiable, with $\|\nabla\phi - \nabla\hat{\phi}\| \leq \epsilon$ (we'll assume this is true everywhere, but it only need hold on a large enough region of interest). Let $x_*$ again be a local minimizer of $\hat{\phi}$, and now suppose that for some $\delta$,

$$\forall \|u\| = 1, \frac{\partial\hat{\phi}}{\partial u}(x_* + \delta u) > \epsilon.$$

Then $\phi$ must have a local minimizer inside the ball of radius $\delta$, since the above condition and the bound on the gradient difference means

$$\forall \|u\| = 1, \frac{\partial\phi}{\partial u}(x_* + \delta u) > 0,$$

and so the minimum of $\phi$ on the closed ball of radius $\delta$ cannot occur on the boundary.

As before, we can estimate an appropriate radius $\delta$ using the second-order Taylor approximation, which gives

$$\frac{\partial \hat{\phi}}{\partial u}(x_* + \delta u) \approx \delta u^T H_{\hat{\phi}}(x_*) u > \delta \lambda_{\min}(H_{\hat{\phi}}(x_*)),$$

so that the condition should be satisfied a little past

$$\delta \approx \frac{\epsilon}{\lambda_{\min}(H_{\hat{\phi}}(x_*))}.$$

As before, we can pull in additional regularity constraints to get a rigorous result. For the moment, we leave this as an exercise for the ambitious reader.

The high-level point, here is that we are far better off with slightly "noisy" gradients than we are with just slightly "noisy" function values. Here we are using scare quotes because we're not using any distributional information about the difference between $\phi$ and $\hat{\phi}$ – we are just assuming something about it is bounded. But the fundamental message does not change much when we look at random perturbations.

# 3 On average

In many problems in machine learning and computational statistics, we see functions that are naturally expressed as expectations. For example, in machine learning applications, we often have loss functions that consist of a large number of independent terms (one per example):

$$\phi(x) = \frac{1}{N} \sum_{i=1}^{N} \psi_i(x),$$

which is the same as

$$\phi(x) = \mathbb{E}_i \left[ \psi_i(x) \right]$$

where $i$ is assumed uniform over the index set $1, \ldots, N$ in the latter expression. More generally, we can consider functions of the form

$$\phi(x) = \mathbb{E}_Z \left[ \psi(x, Z) \right]$$

where $Z$ is some random variable.

## 3.1 Quadrature

If $Z$ is a continuous random variable, or even a random variable over a large finite set, it may be infeasible or very expensive to compute the expected value and get $\phi$. Therefore, we need some type of approximation. For example, we might employ a *quadrature* scheme where

$$\hat{\phi}(x) = \sum_{j=1}^{m} \psi(x, z_i) w_i$$

where the $z_i$ and $w_i$ are known as the quadrature nodes and weights. When $\psi$ is smooth with respect to the $Z$ variable and $Z$ is drawn from a standard low-dimensional distribution (e.g.~uniform or Gaussian random variables in 1D or 2D), we can approximate $\phi$ to very high accuracy using quadrature techniques. In this case, we can use the regularity of $\psi$ to get error bounds on how well $\hat{\phi}$ (and $\nabla\hat{\phi}$) approximate the true expected value and its gradients.

## 3.2 Sample average approximation

When the random variable $Z$ is drawn from a high-dimensional continuous distribution, or when we do not have much smoothness of $\psi$, we might decide to instead approximate the expectation by a sample average. In the simplest case, we would sample $\{z_i\}_{i=1}^m$ according to the distribution of the random variable $Z$ and use the approximation

$$\hat{\phi}(x) = \frac{1}{m} \sum_{i=1}^{m} \psi(x, z_i).$$

In this case, we expect the error in the approximation to behave like $\sqrt{\mathrm{Var}(\psi(x, Z))/m}$.

Alternately, we could sample from a different distribution with the same support as $z$ and choose weights according to the ratio of the distributions in order to get something similar to what we have above for quadrature

$$\hat{\phi}(x) = \sum_{i=1}^{m} \psi(x, z_i) w_i.$$

Here, too, we expect a $1/\sqrt{m}$ type of approximation error.

In the *sample average approximation* (SAA) approach, we make *one* draw of the variables $z_i$ in order to get an approximation $\hat{\phi}$, then optimize $\hat{\phi}$. Like

deterministic quadrature approaches, sample average approximation has an accuracy limit that is determined by how well $\hat{\phi}$ and $\nabla\hat{\phi}$ approximate $\phi$ and $\nabla\phi$. Unlike deterministic quadrature, though, the sample average estimator of the mean typically decays like $1/\sqrt{m}$ where $m$ is the sample size. Often, this means that we need rather large samples to get acceptable accuracy, though matters may be improved somewhat is we use *variance reduction* techniques like control variates or importance sampling.

## 3.3 Stochastic gradient methods

In sample average approximation, we use *one* sample $\{z_i\}_{i=1}^m$ as the basis for computing $\hat{\phi} \approx \phi$. In stochastic gradient methods, we use a different sample *at each step*. That is, we compute

$$x^{k+1} = x^k - \alpha_k g(x^k, \xi_k)$$

where $\mathbb{E}_\xi[g(x,\xi)] = \nabla\phi(x)$ is a Monte Carlo estimator for the gradient – this can be from a single draw or from a larger sample (a so-called "mini-batch").

If we take a fixed step size $\alpha$ and choose independent equal-sized samples at each step, the stochastic gradient method defines a discrete-time Markov chain, and we expect it to converge to a stationary distribution that is concentrated in a neighborhood of a minimum. So for nice enough functions and a sufficiently small fixed step size $\alpha$, the expected value of the optimality gap between $x^k$ and the true minimum at $x^*$ behaves like

$$\mathbb{E}[\phi(x^k) - \phi(x^*)] \leq c_1\alpha + (1 - c_2\alpha)^{-k} \left(\phi(x^0) - \phi(x^*)\right).$$

That is, the expected optimality gap converges linearly – but not to zero! To get closer than $c_1\alpha$ to the true optimal value, we have to reduce the step size. Unfortunately, reducing the step size also reduces the rate of convergence! We can balance the two effects by taking $n_0$ steps with an initial size $\alpha_0$ to get the error down to about $c_1\alpha_0$, then $2n_0$ steps of size $2^{-1}\alpha_0$ to get the error down to about $2^{-1}c_1\alpha_0$, and so forth. This gives us a convergence rate of $O(1/k)$. More generally, we can get convergence with any (sufficiently small) schedule of step sizes such that

$$\sum_{k=1}^{\infty} \alpha_k = \infty, \quad \sum_{k=1}^{\infty} \alpha_k^2 < \infty.$$

There are a wide variety of methods for choosing the step sizes ("learning rate"), sometimes in conjunction with methods to choose a better search direction than the (approximate) steepest descent direction.

The $O(1/k)$ rate of convergence for stochastic gradient descent is quite slow compared to the rate of convergence for ordinary grdient descent. However, each step of a stochastic gradient method may be much cheaper than a full gradient computation, so there is a tradeoff.

### 3.4 Second-order, variance reduction, and beyond

Slow convergence of the stochastic gradient method comes from a combination of two effects: variance in the gradient estimate and the slow rate of gradient descent when the problem is ill-conditioned. Fortunately, there exist techniques that can sometimes help us address both these issues.

To address the variance in the gradient estimate, we need either a bigger sample or smaller variance in the sample elements. Both of these approaches are used in practice. Indeed, an alternative approach to getting SGD to converge instead of scheduling diminishing step sizes is to schedule increasing mini-batch sizes (leading to a few final steps that compute over all the data, in the extreme case). Another option is to use a variance reduction technique, with control variates as a particularly useful case.

To address the poor behavior of gradient descent even without random error, we need to incorporate second-order information. One approach is to use a Broyden (or BFGS) style updating scheme to estimate a Hessian; the main trick to this is that when we update the Hessian, we need to use the same minibatch to estimate the gradient before and after the step. Other momentum-based methods share some relationship to the types of extrapolation and acceleration techniques we have discussed earlier in this class.

There is not time in lecture to really do justice to all the variants of stochastic gradient methods that have been considered, but I can point you to a good paper by Bottou, Curtis, and Nocedal that covers many of the possibilities.

## 4 Randomized algorithms

We can see the stochastic gradient algorithm through two lenses. On the one hand, we are optimizing a function with (changing) random noise asso-

ciated with sampling error. On the other hand, we are optimizing a known deterministic function, and just happen to be using randomization to find estimates of the gradient that are good enough to apply gradient-descent types of techniques.

The idea of using randomness to help us in an optimization process of deterministic functions holds much more generally. We are not going to have time to even skim lightly over much of this, but some of the ways in which randomness can help our algorithms is:

- Random sampling of the objective to find good starting points for a locally-convergent gradient-based iteration.
- Using random probing to get a good approximate Hessian to use as a scaling matrix in a Newton-like iteration.
- Using some randomness to break out of cycling behavior in active set solvers for constrained optimization problems.
- Using random perturbations to break out of degenerate situations that can happen in continuation algorithms (e.g. pitchfork bifurcations).
- Randomly sampling in some local neighborhood in order to get a good set of points to initialize a derivative-free simplex-based gradient method
- Randomly re-ordering search directions in a pattern-search method (and then choosing the first to yield any type of improvement).