Notes on Big-n Problems

Mark Schmidt

June 19, 2012

1 Motivation

We consider problems of the form

$$\min_{\mathbf{x}} \sum_{i=1}^{n} f_i(\mathbf{x}), \tag{1}$$

where each $f_i(\mathbf{x})$ is differentiable and $\mathbf{x}$ is in $\mathbb{R}^p$. We will use the notation $f(\mathbf{x}) = \sum_{i=1}^{n} f_i(\mathbf{x})$. Two widely used examples include least squares

$$\min_{\mathbf{x}} \sum_{i=1}^{n} (1/2)(\mathbf{a}_i^T\mathbf{x} - b_i)^2,$$

and logistic regression

$$\min_{\mathbf{x}} \sum_{i=1}^{n} \log(1 + \exp(b_i\mathbf{a}_i^T\mathbf{x})).$$

Instead of decomposing the objective into single terms, another possibility is to use 'batches'. In the least squares case this would correspond to a decomposition of the form

$$\min_{\mathbf{x}} \sum_{i=1}^{n} \frac{1}{2}||\mathbf{A}_i\mathbf{x} - \mathbf{b}_i||_2^2,$$

where each $\mathbf{A}_i$ is a matrix and each $\mathbf{b}_i$ is a vector containing a subset of rows from the original problem.

A basic iterative gradient method would use steps of the form

$$\mathbf{x}^{k+1} \leftarrow \mathbf{x}^k - \alpha^k \sum_{i=1}^{n} \nabla f_i(\mathbf{x}), \tag{2}$$

where α^k is a step size. If the gradient of f is Lipschitz-continuous with constant $\mathcal{L}$, then using the step size $\alpha^k = 1/\mathcal{L}$ guarantees that $f(\mathbf{x}^{k+1}) < f(\mathbf{x}^k)$ (provided $\nabla f(\mathbf{x}^k) \neq \mathbf{0}$), and if f is convex that the distance to the optimal value $f(\mathbf{x}^*)$ as a function of k is $O(1/k)$. Nesterov [1983] proposes a simple modification, referred to as the accelerated gradient method, that improves this to $O(1/k^2)$, and under certain assumptions this is the fastest possible convergence rate.

In this work we review methods that take advantage of the problem structure to yield alternative (or complementary) methods for improving the convergence rate in cases where n is extremely large and the functions f_i are 'similar'. For example, in the degenerate case where each f_i is identical the *incremental gradient* method discussed in the next section has a convergence rate of $O(1/kn)$ in terms of k and n using a step size of $n/\mathcal{L}$. That is, the error rate is decreased n times faster than the basic gradient method. Despite being a slower convergence rate in terms of k, when n is large and only a finite number of iterations will be performed, this may yield a more practical method than an optimal $O(1/k^2)$ methods. Although in general we don't expect to achieve an n-fold speed-up since the functions f_i will not be identical, we might still expect to see computational gains in cases where the functions f_i are sufficiently similar.

2 Incremental Gradient Methods

A basic incremental gradient method takes steps of the form

$$\mathbf{x}^{k+1} \leftarrow \mathbf{x}^k - \alpha^k \sum_{i=1}^{n} \nabla f_i(\mathbf{y}^i), \tag{3}$$

where on iteration k we define $\mathbf{y}^0 = \mathbf{x}^k$, and

$$\mathbf{y}^i \leftarrow \mathbf{y}^{i-1} - \alpha^k \nabla f_i(\mathbf{y}^{i-1}).$$

That is, instead of updating $\mathbf{x}^k$ based on $\nabla f(\mathbf{x}^k)$, we update $\mathbf{x}^k$ based on the gradients with respect to a sequence of settings $\mathbf{y}^i$ that represent gradient steps with respect to individual functions $f_i(\cdot)$.

We can alternately write this as an iterative method that does not introduce the $\mathbf{y}$ variables and simply uses updates of the form

$$\mathbf{x}^{k+1} \leftarrow \mathbf{x}^k - \alpha^k \nabla f_i(\mathbf{x}^k), \tag{4}$$

where the variable i is chosen in order from 1 to n and then re-started at 1 every k iterations. We note that k iterations of (4) cost the same as one iteration of (3) or (2), while iterations of (4) that are multiplies of k give us the iterates generated by (3).

If the functions f_i are identical, then (4) is a gradient step with respect to f with a step size of α^k/n. The convergence rate of $O(1/kn)$ for (3) stated in the motivation section follows from this. Even if the functions are not identical, the incremental gradient method may still be advantageous because the individual f_i may still be sufficiently similar to allow reasonable progress.

In general $-\nabla f_i(\mathbf{x}^k)$ will not be a direction of descent on f at every iteration, but it is still possible to show convergence of the iterations to a minimizer under appropriate conditions. For example, Bertsekas and Tsitsiklis [1996, Proposition 3.8] outline sufficient conditions for convergence of the incremental gradient method when applied to non-linear least squares problems, but the proof can be can be generalized to problems with the general form (1). In the general case, provided that $f(\mathbf{x})$ is bounded below this proposition establishes that $\lim_{k\to\infty} \nabla f(\mathbf{x}^k) = \mathbf{0}$ under the following assumptions:

1. $||\nabla f_i(\mathbf{x}) - \nabla f_i(\mathbf{y})|| \leq L||\mathbf{x} - \mathbf{y}||, \forall i, \mathbf{x}, \mathbf{y}$.
2. $||\nabla f_i(\mathbf{x})|| \leq C + D||\nabla f(\mathbf{x})||, \forall i, \mathbf{x}$.
3. $\sum_{t=0}^{\infty} (\alpha^k)^2 < \infty$.
4. $\sum_{t=0}^{\infty} \alpha^k = \infty$.

In the above, L, C, and D are positive constants. The first condition is a Lipschitz-continuity assumption on the gradients of each function $f_i(\mathbf{x})$, and is a standard assumption. The second condition requires that the sizes of the individual gradient f_i' are bounded by an affine function of the magnitude of the full gradient. This condition is fairly weak and is analgous to the condition $||\mathbf{d}|| \leq C||\nabla f(\mathbf{x})||$ on a search direction $\mathbf{d}$ for non-incremental methods. The third assumption requires that the sequence of step sizes converges to zero at a certain rate. In general, without this condition the incremental gradient method will not have a limit since even if $\nabla f(\mathbf{x}^k) = \mathbf{0}$ we may have $f_i(\mathbf{x}^k) \neq \mathbf{0}$ for some i causing $\mathbf{x}^{k+1}$ to move away from the solution. The final assumption requires that the sum of the step sizes diverges. This is required so that the decreasing step sizes do not confine the iterates to a subset of $\mathbb{R}^p$ that does not contain the solution.

The conditions on the step size immediately suggest using the sequence $\alpha^k = 1/k$. However, in practice this sequence is somewhat unnappealing since the step sizes very quickly become extremely small. A generalization of this that still ensures convergence is the sequence $\alpha^k = a/(k+b)^c$, where a and b are non-negative and c is in $(0.5, 1]$. A further generalization is to keep α^k fixed for some constant number of iterations before moving to the next term in the sequence.

We can also consider variants of the incremental gradient method where a sufficiently small constant step size is used. Although in general this strategy does not lead to a convergent method, in some cases

it is possible to show that the iterates obtained using a constant step size eventually oscillate within a neighborhood of the solution, where the size of the neighborhood depends on the step size. Subsequently, the optimal solution is obtained as the limit of decreasing the step size to zero to make this neighborhood shrink to a point. For more details, see [Bertsekas and Tsitsiklis, 1996, Example 3.6 and Proposition 3.6(a)] and [Bertsekas et al., 2003, Proposition 8.2.2].

2.1 Stochastic Approximation

Recall the second form of the incremental gradien iteration:

$$\mathbf{x}^{k+1} \leftarrow \mathbf{x}^k - \alpha^k \nabla f_i(\mathbf{x}^k), \tag{5}$$

where the variable i in chosen in order from 1 to n and then re-started at 1. As discussed in Bertsekas et al. [2003, §8.2], the rate of convergence of incremental (sub-)gradient methods can be affected by this order. Further, it may be non-trivial to determine the optimal ordering. A widely-used alternative to choosing an explicit ordering is to select an i at random from the set $\{1, 2, \ldots, n\}$. The convergence properties of this randomized version of the incremental gradient method are very similar to the deterministic case, but the stochastic variant protects against the possibility that a pathological ordering of the functions will slow convergence. In the randomized variant, it is easy to see that the random selection of i gives a descent direction *on average*. This is because the expectation over i of $\nabla_i f(\mathbf{x})$ with $p(i)$ given by $1/n$ is simply $(1/n)\nabla f(\mathbf{x})$. This variant is known as *stochasctic gradient descent*, and it is a specific instance of a *stochastic approximation* algorithm.

Stochastic approximation algorithms were originally proposed by Robbins and Monro [1951] as a method to solve non-linear equations

$$g(\mathbf{x}) = \mathbf{0},$$

based on noisy measurements of $g(\mathbf{x})$. We obtain optimization problems in the special case where $g(\mathbf{x})$ is defined by $\nabla f(\mathbf{x})$ for the (convex) function $f(\mathbf{x})$ that we would like to optimize. This special case is often referred to as stochastic gradient descent, and dates back to Kiefer and Wolfowitz [1952]. In the context of incremental gradient methods, we can view $\nabla f_i(\mathbf{x})$ for a random i as a noisy measurement of the average of the full gradient $(1/n)\nabla f(\mathbf{x})$[1]. The corresponding stochastic gradient descent iteration is given by (5).

Note that since the iterations themselves are random variables, in this context *convergence* will be specified in terms of some measure of convergence of a sequence of random variables. For example, we might say that $\mathbf{x}^k$ converges *almost surely* to $\mathbf{x}^*$. This means that the random variable converges with a probability of one, since all realizations of the random variable where it doesn't converge have a probability of zero. There are variety of methods available to show that a stochastic approximation method converges, each using slightly different assumptions. For the particular case of solving optimization problems, stochastic approximation methods converge under fairly similar conditions to incremental gradient methods. For example, specializing [Bertsekas and Tsitsiklis, 1996, Proposition 4.1] to the incremental gradient case establishes that $\lim_{k\to\infty} \nabla f(\mathbf{x}^k) = \mathbf{0}$ (with probability one) for a function that is bounded below under the following assumptions:

1. $||\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})|| \leq L||\mathbf{x} - \mathbf{y}||, \mathbf{x}, \mathbf{y}$.
2. $\mathbb{E}[||\nabla f_i(\mathbf{x}^k)||^2] \leq C + D||\nabla f(\mathbf{x}^k)||^2, \forall k, \mathbf{x}$.
3. $\sum_{k=1}^{\infty} (\alpha^k)^2 < \infty$.
4. $\sum_{k=1}^{\infty} \alpha^k = \infty$.

In the second condition the expectation is taken with respect to i, and it requires that the variance of the individual f_i is bounded as a quadratic function of the magnitude of the full gradient. These assumptions are only slightly stronger than those of the previous section. [Bertsekas and Tsitsiklis, 1996, Proposition 4.1] also requires the *pseudo-gradient* property (for some $c > 0$):

[1] In this context we have introduced randomness into an otherwise deterministic method. However, stochastic approximation methods also apply in cases where we have an inherently noisy measurement of the gradient.

5. $c||\nabla f(\mathbf{x}^k)||^2 \leq -\nabla f(\mathbf{x}^k)^T \mathbb{E}[\mathbf{d}^k]$,

where $\mathbb{E}[\mathbf{d}^k]$ is the expected search direction. The pseudo-gradient property ensures that the average direction yields a descent direction, and it is trivially satisfied for the randomized incremental gradient method because we have $\mathbb{E}[\mathbf{d}^k] = \mathbb{E}[-\nabla f_i(\mathbf{x}^k)] = -(1/n)\nabla f(\mathbf{x}^k)$. However, we note it here because this condition will be important later when we consider methods that use a *biased* gradient approximation.

Convergence under these conditions is shown using known results about convergence of super-Martingales, and this type of proof technique is known as the 'smooth potential' or 'Lyaupunov' approach. A second method of proving convergence of stochastic approximation methods is discussed in [Bertsekas and Tsitsiklis, 1996, §4.3], where convergence is shown under a contraction or monotonicity assumption. This technique allows us to prove that stochastic approximation methods converge even in some cases that don't have a well-defined objective function. A third method of showing that stochastic approximation methods converge is known as the ordinary differential equation (ODE) method and is discussed in [Bertsekas and Tsitsiklis, 1996, §4.4]. This technique can be useful in cases where the noise in the gradient measurement is not necessarily independent. A notable aspect of the ODE method is that it assumes that the iterates stay within a compact set. This assumption can be guaranteed by using a projected stochastic gradient iteration

$$\mathbf{x}^{k+1} \leftarrow \Pi[\mathbf{x}^k - \alpha^k \nabla f_i(\mathbf{x}^k)],$$

where the projection operator projects the iterates onto a convex set that is known to contain the optimal solution. Such a step is also one of the available strategies for using stochastic approximation methods when we have convex constraints on the variables. Alternatively, if we do not know a compact set containing the optimal solution, we can project onto an increasing sequence of sets [Chen et al., 1987].

2.2 Asymptotic Rate of Convergence

Similar to statistical estimators, the *rate* of convergence of a stochastic gradient algorithm is typically measured in terms of its asymptotic variance around the optimal parameters. Specifically, under some fairly strong assumptions (which are similar to strong convexity assumptions in optimization) we can show that the value of the random variable $\sqrt{k}(\mathbf{x}^k - \mathbf{x}^*)$ converges in distribution to a normal distribution with a mean of zero and a covariance matrix Σ [Spall, 2003, §4.4]:

$$\sqrt{k}(\mathbf{x}^k - \mathbf{x}^*) \xrightarrow{d} \mathcal{N}(\mathbf{0}, \Sigma).$$

The asymptotic covariance Σ depends on the both the choice of step sizes α^k and the Hessian $\nabla^2 f$ of the problem. This result implies that we asymptotically expect the distance between $\mathbf{x}^k$ and the optimal parameter vector $\mathbf{x}^*$ to behave like $O(1/\sqrt{k})$. Under the Lipschitz-continuity assumption on the gradients, this implies that $f(\mathbf{x}^k) - f(\mathbf{x}^*)$ asymptotically behaves like $O(1/k)$. We say that an algorithm is more *efficient* than another algorithm if it has a smaller asymptotic variance Σ (according to some matrix norm), which (roughly) means that the random variable $\mathbf{x}^k$ is distributed in a smaller neighborhood around $\mathbf{x}^*$ for a given k.

2.3 Second-Order Methods

It is well-known that Newton-like methods have a faster convergence rate than the gradient method in the deterministic scenario, so it is natural to consider stochastic versions of Newton-like algorithms. There is a substantial literature exploring this possibility, and these methods typically use iterates of the form

$$\mathbf{x}^{k+1} \leftarrow \mathbf{x}^k - \alpha^k \mathbf{B}^k \nabla f_i(\mathbf{x}^k),.$$

Given this form of iteration, we can consider designing a sequence of step sizes α^k and scaling matrices $\mathbf{B}^k$ that minimize Σ in the asymptotic covariance, in order to optimize the convergence rate. As discussed in [Spall, 2003, §4.4], the optimal choice is a sequence of step sizes that behave like $\alpha^k = 1/k$, combined

with setting the scaling matrix $\mathbf{B}^k$ to the inverse of $\nabla^2 f(\mathbf{x}^*)$. Thus, in some sense a Newton-like stochastic algorithm achieves the optimal convergence rate.

Since we do not know the inverse Hessian at the minimizer, we would like to design a sequence of values of $\mathbf{B}^k$ that converges to this value. Approaches that do this are called *adaptive*. One possible strategy is to compute the Hessian for the individual function $f_i(\mathbf{x}^k)$, and keep a running average approximation to the Hessian:

$$\mathbf{H}^k \leftarrow \frac{k}{k+1}\mathbf{H}^{k-1} + \frac{1}{k+1}\nabla^2 f_i(\mathbf{x}_k). \tag{6}$$

Subsequently, we can set $\mathbf{B}^k$ to the inverse of $\mathbf{H}^k$ (modified to be positive-definite).[2] For quadratic functions, this converges in probability to the average of the true Hessian, $(1/n)\nabla^2 f(\mathbf{x})$, by the law of large numbers. In the context of least squares problems, this is referred to as the recursive least squares (RLS) method [Spall, 2003, §3.2].[3] Under appropriate assumptions, convergence of this choice of Hessian approximation to the Hessian can also be shown for the convergent step sizes of the previous section. Several authors have also considered stochastic variants of Gauss-Newton methods for non-linear least squares problems [Bertsekas and Tsitsiklis, 1996, §3.2.6] where the scaling matrix is updated according to

$$\mathbf{H}^k \leftarrow \frac{k}{k+1}\mathbf{H}^{k-1} + \frac{1}{k+1}\nabla f_i(\mathbf{x}_k)\nabla f_i(\mathbf{x}^k)^T.$$

The advantage of Gauss-Newton methods is that the approximation $\mathbf{B}^k$ is constructed using the noisy gradient measurements rather than explicitly requiring $\nabla^2 f_i(\mathbf{x}_k)$. There has also been recent work on stochastic quasi-Newton methods that utilize variants of the L-BFGS approximation [Schraudolph et al., 2007]. Global convergence of stochastic Newton-like algorithms can be shown under the assumptions of the previous section in addition to the assumption that the eigenvalues of $\mathbf{B}^k$ are bounded between positive constants for all k [Sunehag et al., 2009].

2.4 Iterate Averaging

During the 1980s, Ruppert and Polyak independently proved the surprising result that a very simple algorithm could achieve the same optimal asymptotic convergence rate achieved by optimal stochastic Newton-like methods [Polyak and Juditsky, 1992]. This algorithm uses the basic stochastic gradient iteration discussed at the beginning of the section,

$$\mathbf{x}^{k+1} \leftarrow \mathbf{x}^k - \alpha^k \nabla f_i(\mathbf{x}^k),$$

but requires a sequence of step sizes α^k that converge to zero asymptotically slower than $O(1/k)$. More specifically, Polyak and Juditsky consider sequences satisfying $(\alpha^{k+1} - \alpha^k)/\alpha^k = o(\alpha^k)$. For example, we could set α^k to $1/k^c$ for $c \in (1/2, 1)$. These choices of step size lead to a sub-optimal convergence rate, but Ruppert and Polyak showed that the *average* value of $\mathbf{x}^k$ converges to the optimal solution at the optimal rate. Thus, if we maintain a second sequence of iterates

$$\mathbf{y}^k \leftarrow \frac{k}{k+1}\mathbf{y}^{k-1} + \frac{1}{k+1}\mathbf{x}^k,$$

then $\mathbf{y}^k$ converges to $\mathbf{x}^*$ at the optimal rate. This is a surprising result since it indicates that, if we use asymptotic normality as our measure of rate of convergence, we should not expect Newton-like stochastic methods to perform better than the average iterate of a stochastic gradient method.

[2] If the Hessians of the individual functions $f_i(\mathbf{x})$ are low-rank, then we can update a factorization of $\mathbf{H}^k$ or directly update an approximation to the inverse $\mathbf{B}^k$ using the matrix inversion lemma. We also note that some variants do not include the averaging.

[3] First-order incremental gradient methods for the least squares problem are sometimes referred as least mean squares (LMS) methods [Spall, 2003, §3.1].

2.5 Finite-Difference Methods

When applied to optimization, the stochastic approximation framework assumes that we have noisy estimates of the gradient. However, in many optimization problems it is much more realistic to expect that we have a noisy estimate of the function value. In such cases, we can consider using the noisy estimate of the function value to obtain a noisy estimate of the gradient. For example, the use of finite-difference numerical approximations within stochastic gradient methods dates back to Kiefer and Wolfowitz [1952]. In the context of incremental gradient methods, a central-differencing approximation to the derivative with respect to j would be

$$(1/n)\nabla_j f(\mathbf{x}^k) \approx \nabla_j f_i(\mathbf{x}^k) \approx \frac{f_i(\mathbf{x}^k + \epsilon^k \mathbf{e}_j) - f_i(\mathbf{x}^k - \epsilon^k \mathbf{e}_j)}{2\epsilon^k}.$$

However, note that unlike $\nabla_j f_i(\mathbf{x}^k)$, this noisy gradient estimate has a *bias* of size $O(\epsilon^2)$ [Spall, 2003, §6.4]. That is, even though $\nabla f_i(\mathbf{x}^k)$ is the limit of the central difference as ϵ^k approaches zero, for positive ϵ^k it is not an unbiased estimate of this quantity. Because of this bias, stochastic gradient methods based on a finite-difference approximation may not satisfy the *pseudo-gradient* property discussed in Section 2.1. Thus, in order to show convergence of the method we need additional assumptions. For example, that the eigenvalues of the Hessian are bounded and that the sequence ϵ^k converges to zero while $\sum_{k=1}^{\infty}(\alpha^k)^2/(\epsilon^k)^2 < \infty$ [Spall, 2003, §6.4]. From these conditions, we see that ϵ^k must go to zero *slower* than α^k. Iterations of a stochastic gradient descent method with finite-differencing satisfy asymptotic normality but with a non-zero mean due to the bias in the estimate [Spall, 2003, §6.5],

$$\sqrt{\beta}(\mathbf{x}^k - \mathbf{x}^*) \xrightarrow{d} \mathcal{N}(\mu, \Sigma).$$

The asymptotic bias μ depends on the bias in the gradient approximation as well as the step sizes, while β is a function of the step sizes. In terms of k, under the central-difference approximation the optimal asymptotic error slows from $O(1/\sqrt{k})$ to $O(1/\sqrt[3]{k})$. However, in several scenarios it can be shown that finite-difference methods do not suffer this loss of efficiency [see Spall, 2003, §6.1].

2.6 Simultaneous Perturbation

For high-dimensional problems, a major drawback of the finite-differencing method is that it requires $2p$ noisy function evaluations on each iteration. Although evaluating the noisy function value $f_i(\mathbf{x}^k)$ is relatively cheap compared to evaluating $f(\mathbf{x}^k)$, it is unnappeling to require p evaluations per iteration if p is large. The simultaneous perurbation stochastic approximation (SPSA) method was proposed to address this poor scaling with p [Spall, 2003, §7]. SPSA is a simple method that has essentially the same convergence properties as stochastic gradient descent with finite-differencing, but only requires 2 noisy function evaluations on each iteration. Rather than using a separate direction $\mathbf{e}_j$ along each coordinate to compute the finite-difference, the SPSA method perturbs all coordinates along a random direction $\mathbf{d}^k$

$$(1/n)\nabla_j f(\mathbf{x}^k) \approx \nabla_j f_i(\mathbf{x}^k) \approx \frac{f_i(\mathbf{x}^k + \epsilon^k \mathbf{d}^k) - f_i(\mathbf{x}^k - \epsilon^k \mathbf{d}^k)}{2\epsilon^k \mathbf{d}_j^k}.$$

Although there are a variety of valid ways to choose the perturbation directions $\mathbf{d}^k$, the optimal choice (in some sense) is to sample each element of $\mathbf{d}^k$ from a $\{-1, 1\}$-Bernoulli distribution [Spall, 2003, §7.7]. With this choice, the method computes a crude approximation of the effect of moving roughly half the variables in the positive direction and half the variables in the negative direction. Since it maintans the same asymptotic efficiency as the finite-difference method, the SPSA method is preferable for high-dimensional problems due to the need for a factor of p fewer evaluations of the noisy objective value.

2.7 Second-Order Simultaneous Perturbation

We can also consider implementing a stochastic-Newton method with a stochastic finite-difference approximation of the Hessian, using $O(p)$ noisy gradient evaluations, or $O(p^2)$ noisy function evaluations [Spall, 2003,

§7.8]. However, for high-dimensional problems this is unnappealing and we can again consider an SPSA-type of approximation to the Hessian by perturbing the gradient along a random direction. Specifically, consider an iteration of the form

$$\mathbf{x}^{k+1} \leftarrow \mathbf{x}^k - \alpha^k \mathbf{B}^k \nabla f_i(\mathbf{x}^k),$$

where $\mathbf{B}^k$ is given by the inverse of

$$\mathbf{H}^k \leftarrow \frac{k-1}{k}\mathbf{H}^{k-1} + \frac{1}{k}\mathbf{S},$$

and $\mathbf{S}$ is an SPSA-type approximation of the Hessian of f_i at $\mathbf{x}^k$ obtained by perturbing the gradient in a random direction. This is referred to as the *adaptive* SPSA method, and unlike a finite-difference approximation that would require $O(p)$ noisy gradient evaluations on each iteration, this method requires only three (one for computing the gradient at $\mathbf{x}^k$, and two to from $\mathbf{S}$). Further, similar to the iterate-averaging trick, it represents a simple means to achieve the optimal asymptotic efficiency without explicitly computing second-order information [Spall, 2003, §7.8].

2.8 Natural Gradient

Up to this point we have discussed several incremental gradient methods with a scaling matrix chosen to approximate the Hessian of $f(\mathbf{x})$. Amari [1998] proposes an alternative scaled incremental gradient method motivated by arguments from information geometry for cases where the parameter space defines a Riemann metric. In such cases, it is argued that the steepest descent direction in the Riemann space should be used. This direction is referred to as the *natural* gradient. Denoting $\mathcal{I}(\mathbf{x}^k)$ as the Riemann metric at $\mathbf{x}^k$, the incremental natural gradient method takes steps of the form:

$$\mathbf{x}^{k+1} \leftarrow \mathbf{x}^k - \alpha^k \mathcal{I}(\mathbf{x}^k)^{-1} \nabla f_i(\mathbf{x}^k).$$

Although this appears similar to an incremental Newton method, in general $\mathcal{I}(\mathbf{x}^k)$ will not be equal to the Hessian $\nabla^2 f(\mathbf{x}^k)$. When each $f_i(\mathbf{x})$ corresponds to a log-likelihood (as in least squares and logistic regression), the Riemann metric is given by the Fisher information matrix[4]

$$\mathcal{I}(\mathbf{x}^k) = \mathbb{E}[\nabla f(\mathbf{x}^k) \nabla f(\mathbf{x}^k)^T].$$

Here, the expectation is taken *with respect to the probabilistic model*.

As an example, consider the least squares problem where we assume $\mathbf{b} \sim \mathcal{N}(\mathbf{Ax}, \mathbf{I})$. In this case, the function is given by $f(\mathbf{x}^k) = (1/2)||\mathbf{Ax} - \mathbf{b}||_2^2$, the gradient by $\nabla f_i(\mathbf{x}^k) = \mathbf{A}^T(\mathbf{Ax} - \mathbf{b})$, and the covariance by $\mathbb{E}_{\mathbf{b}}[(\mathbf{Ax} - \mathbf{b})(\mathbf{Ax} - \mathbf{b})^T] = \mathbf{I}$. Thus,

$$\mathcal{I}(\mathbf{x}^k) = \mathbb{E}_{\mathbf{b}}[\mathbf{A}^T(\mathbf{Ax} - \mathbf{b})(\mathbf{Ax} - \mathbf{b})^T\mathbf{A}] = \mathbf{A}^T\mathbb{E}_{\mathbf{b}}[(\mathbf{Ax} - \mathbf{b})(\mathbf{Ax} - \mathbf{b})^T]\mathbf{A} = \mathbf{A}^T\mathbf{I}\mathbf{A} = \mathbf{A}^T\mathbf{A}.$$

That is, the Fisher information matrix and Hessian are the same. The Hessian and Fisher information matrix are also the same for logistic regression, but are not the same in general. A notable alternative interpretation of the natural gradient method is given by Le Roux et al. [2007], who take the expectation with respect to the empirical distribution over gradients. In this case, the Fisher information and Hessian are only the same if the data actually follows the model.

Similar to the Hessian, we are not typically given the Fisher information matrix but rather iteratively build an approximation $\mathbf{H}^k$ of it:

$$\mathbf{H}^k \leftarrow \frac{k}{k+1}\mathbf{H}^{k-1} + \frac{1}{k+1}\nabla f_i(\mathbf{x}_k) \nabla f_i(\mathbf{x}^k)^T.$$

[4]In statistics, Newton-like methods where the Hessian is replaced by this information matrix are sometimes referred to as Fisher scoring.

This quantity appears to be closely related to the Gauss-Newton Hessian approximation for non-linear least squares problem, but they are not the same. For example, if we consider a non-linear least squares problem $\min_{\mathbf{x}}(1/2)\sum_{i=1}^{n} f_i(\mathbf{x})^2$, then the natural gradient update becomes

$$\mathbf{H}^k \leftarrow \frac{k}{k+1}\mathbf{H}^{k-1} + \frac{1}{k+1} f_i(\mathbf{x})^2 \nabla f_i(\mathbf{x}_k) \nabla f_i(\mathbf{x}^k)^T.$$

That is, the outer-products used in the Gauss-Newton approximation are weighted by the corresponding residual squared. Further, the Fisher information matrix is defined even if the problem is not a non-linear least squares problem. In cases where the Hessian and information matrix coincide, this approximation converges to the Hessian, without the need to compute second-order information explicitly (assuming the data actually follows the model). However, similar to the Gauss-Newton method, one of the appealing aspects of using the natural gradient as opposed to the Hessian is that $\mathcal{I}(\mathbf{x}^k)$ is guaranteed to be positive semi-definite. This property holds even if the problem is not a non-linear least squares problem, and a corollary of this is that the Hessian matrix and Fisher information matrix do not coincide for non-convex objectives.

2.9 Accelerated Stochastic Approximation

In many practical problems the error in the gradient estimate can make second-order methods unstable. As opposed to the methods discussed above that build a Hessian approximation, accelerated stochastic approximation methods apply the standard stochastic gradient method but look at whether the sign of the gradient changes in order to estimate the step size. The intuition behind these methods is that the step size should remain large unless we are sufficiently close to the solution, and at this point the step size should be decreased. Kesten [1958] proposed the original one-dimensional accelerated stochastic gradient method, while Delyon and Juditsky [1993] extended this method to the multivariate scenario and also examined the asymptotic normality of the method. Formally, if we use $\mathbf{g}_k$ as the gradient approximation at iteration k, this method uses the update

$$\mathbf{x}^{k+1} \leftarrow \mathbf{x}^k - s(\gamma_k)\mathbf{g}_k,$$

where

$$\gamma_{k+1} = \gamma_k + \mathcal{I}(\mathbf{g}_k^T \mathbf{g}_{k-1} < 0),$$

and $s(n)$ is a sequence of step sizes, typically of the form $O(1/n)$.

2.10 Sub-Gradient and Proximal Methods

Incremental methods can also be used for non-differentiable problems. In particular, an incremental sub-gradient method would take iterations of the form

$$\mathbf{x}^{k+1} \leftarrow \mathbf{x}^k - \alpha^k \mathbf{g}_i^k,$$

where i is chosen in a cyclic or random way and $\mathbf{g}_i^k$ is any element of the sub-differential of f_i at $\mathbf{x}_k$. Alternately, we can consider incremental proximal methods where each iteration is given by the solution of a proximal problem for a given sample i:

$$\mathbf{x}^{k+1} \leftarrow \arg\min_{\mathbf{x}} \{f_i(\mathbf{x}) + \frac{1}{2\alpha}||\mathbf{x} - \mathbf{x}^k||_2^2\}.$$

As discussed by Bertsekas [2010], both of these methods have similar convergence properties to the incremental gradient method.

2.11 Non-Asymptotic Rates of Convergence

As opposed to the local rates of convergence discussed earlier, there has also been work on global and non-asymptotic bounds for the convergence rates of incremental gradient methods when applied to convex

problems. In particular, the error in function value for the deterministic incremental sub-gradient method is $O(1/\sqrt{k})$ for cycle k [Bertsekas et al., 2003, Proposition 8.2.3]. This is the same worst-case convergence rate as the non-incremental sub-gradient method. For the randomized incremental sub-gradient method, the *expected* worst-case error is $O(1/\sqrt{k})$ after k *updates* [Bertsekas et al., 2003, Proposition 8.2.12]. That is, the randomized method requires m times fewer iterations to achieve the same optimality level. This is a surprising result, indicating that the randomized incremental sub-gradient method may not only be more efficient than the deterministic method, but that it may be more efficient than the non-incremental sub-gradient method (despite its substantially lower iteration cost when n is large). Further, as discussed in [Agarwal et al., 2009] there is a lower bound of $\Omega(1/\sqrt{k})$ for optimizing Lipschitz-continuous functions based on a stochastic first-order oracle. Nesterov [2009] recently proposed a variant of the stochastic sub-gradient descent algorithm called dual averaging, that uses the average of previous sub-gradients as the search direction,

$$\mathbf{x}^{k+1} \leftarrow \mathbf{x}^k - \alpha^k \frac{1}{k+1} \sum_{i=0}^{k} \mathbf{g}_i^k.$$

This algorithm achieves the $O(1/\sqrt{k})$ rate but with better constants, and is related to Baher's method [see Kushner and Yin, 2003, §1.3.4] which combines gradient averaging with *online* iterate averaging and achieves the optimal convergence rate in terms of asymptotic efficiency. Incremental proximal algorithms that achieve the $O(1/\sqrt{k})$ rates in the deterministic and stochastic cases are discussed by Bertsekas [2010].

There also exist *heavy-ball* variants of incremental gradient methods that use updates of the form

$$\mathbf{x}^{k+1} \leftarrow \mathbf{x}^k - \alpha^k \nabla f_i(\mathbf{x}^k) + \beta^k(\mathbf{x}^k - \mathbf{x}^{k-1}).$$

In the stochastic gradient context, the extra term is typically referred to as the 'momentum' [Bertsekas and Tsitsiklis, 1996, §3.2]. In the non-incremental scenario, accelerated gradient methods are a variant of this iteration leads to an error in function value of $O(1/k^2)$ if the gradient is Lipschitz-continuous (but without needing strong convexity)[Nesterov, 1983]. Although there is no analogous result known for incremental methods, Ghadimi and Lan [2010] and Xiao [2010] describe methods where if we decompose the error into a deterministic and a stochastic part, then the methods achieve an $O(1/k^2)$ rate on the deterministic part while still having an $O(1/\sqrt{k})$ rate on the stochastic part.

Similar to the non-incremental case, we obtain better global convergence rates under additional assumptions. As discussed in Nemirovski et al. [2009, §2.1], for *strongly* convex objectives (with a Lipschitz-continuous gradient) the expected error in the objective value is $O(1/k)$ after k updates, and we also show that $\mathbb{E}[||\mathbf{x}^k - \mathbf{x}^*||] = O(1/\sqrt{k})$. Note that in terms of k these rates coincide with the asymptotic rates discussed in Section 2.2, and that the assumptions used to show these rates are similar. Although these sublinear rates are faster than the non-smooth case, they are slower than the linear convergence rates that can be achieved by non-incremental methods under these assumptions. Thus, in this scenario the incremental method will only be more efficient when n is sufficiently large and ϵ is not too small.

Shalev-Shwartz et al. [2007] show that when the objective function is strongly convex, but not necessarily differentiable, that the average iterate of the stochastic sub-gradient method achieves an expected error of $O(\log k/k)$. Shamir [2011] has more recently shown that using sub-gradient descent and averaging only the last iterations can achieve an expected error of $O(1/k)$. Thus, differentiability is not required to achieve an $O(1/k)$ with a stochastic sub-gradient method. As discussed by Agarwal et al. [2009], this is the optimal rate when the objective function is strongly convex and we only have a first-order oracle available.

Finally, we note that even though the convergence rate to the optimal solution may be sub-linear, the convergence to some approximate solution can be much faster. For example, Nedic and Bertsekas [2000, Proposition 2.4] show that under strong convexity assumptions that the incremental gradient method with a constant step size converges (in terms of distance to the optimal parameter vector) at a linear rate to a neighborhood containing the solution. An analgous result is given by d'Aspremont [2005]. who shows that applying an accelerated gradient method with an error in the gradient for convex (but not necessarily strongly convex) optimization achieves the optimal $O(1/k^2)$ rate up to some error tolerance.

2.12 Amplification

In the previous section, we saw that the *expected* worst-case error achieved by stochastic algorithms might be substantially smaller than the worst-case error achieved by deterministic algorithms for the same amount of computation. However, this comparison is not strictly valid, since for a single run it is possible that the error of the stochastic method will be much higher than its expectation. Using an amplification strategy where we use multiple runs of the stochastic algorithm we can protect against this possibility with high confidence. Since the error is non-negative, we can apply Markov's inequality to bound the probability that multiple runs of the algorithm deviate by more than a constant from the expected value. In particular, if we have an algorithm with an expected error of $O(1/k)$ and we want to have a confidence of $1-\delta$ that we achieve an error within a fixed constant of this, it suffices the run the stochastic algorithm $\lceil \log \delta \rceil$ times [Collins et al., 2008, §5]. Thus, we can achieve an error of $O(1/k)$ with a confidence of $1-\delta$ in a total $k\lceil \log \delta \rceil$ iterations.

2.13 Mirror Descent and Exponentiated Gradient

We can re-write the gradient iteration as a proximal iteration applied to a linearized version of the objective function. In particular, if we use the approximation

$$f_i(\mathbf{x}) \approx f_i(\mathbf{x}^k) + \nabla f_i(\mathbf{x}^k)^T(\mathbf{x}-\mathbf{x}^k),$$

we can re-write the gradient iteration with a fixed step size as the solution of the proximal problem

$$\mathbf{x}^{k+1} \leftarrow \arg\min_{\mathbf{x}} \{f_i(\mathbf{x}^k) + \nabla f_i(\mathbf{x}^k)(\mathbf{x}-\mathbf{x}^k) + \frac{1}{2\alpha}||\mathbf{x}-\mathbf{x}^k||_2^2\}.$$

We can consider a generalization of this iteration where we replace the squared Euclidean norm by a general Bregman distance:

$$\mathbf{x}^{k+1} \leftarrow \arg\min_{\mathbf{x}} \{f_i(\mathbf{x}^k) + \nabla f_i(\mathbf{x}^k)^T(\mathbf{x}-\mathbf{x}^k) + \frac{1}{\alpha}D(\mathbf{x},\mathbf{x}^k)\}.$$

This is know as the mirror descent algorithm, and in the non-incremental setting Beck and Teboulle [2003] show that it has a worst-case error of $O(1/\sqrt{k})$ (where the gradient is replaced by a sub-gradient). We obtain the gradient method if we choose the distance function $D(\mathbf{x},\mathbf{y})$ based on the squared Euclidean norm $(1/2)||\mathbf{x}-\mathbf{y}||_2^2$, while we obtain Newton's method if we choose the distance function based on a squared quadratic norm $(1/2)(\mathbf{x}-\mathbf{y})^T\nabla^2 f(\mathbf{x})(\mathbf{x}-\mathbf{y})$. Another interesting case arises if the parameters $\mathbf{x}$ are constrained to be a probability distribution, so that each x_j must be non-negative and they must sum up to one. In this case a natural Bregman distance to use is the Kullback-Leibler divergence,

$$D(\mathbf{x},\mathbf{y}) = \sum_{j=1}^{p} x_j \log\frac{x_j}{y_j}.$$

Under this distance measure, we obtain iterations of the form

$$x_j^{k+1} \leftarrow \frac{r_j^k x_j^k}{\sum_{j'=1}^{p} r_{j'}^k x_{j'}^k},$$

where the weights $\mathbf{r}_j^k$ are given by

$$\mathbf{r}_j^k = \exp(\alpha \nabla_j f(\mathbf{x}^k)).$$

Note that these udpates *implicitly* enforce the constraint that the parameters must lie on the unit simplex. Beck and Teboulle [2003] call this the entropic descent algorithm.

An incremental version of this method was examined by Kivinen and Warmuth [1997], who called it the *exponentiated gradient* method. Kivinen and Warmuth [1997, §3] also give a simple extension of the exponentiated gradient method that allows negative variables and relaxes the assumption that the sum of the weights equals one. They divide each variable into positive and negative parts, x_i^+ and x_i^-, and enforce that

the sum of these variables is equal to some upper bound U. Thus, the exponentiated gradient method can be applied provided we have an upper bound on the ℓ_1 norm of the solution. The analysis of Kivinen and Warmuth [1997] suggests that the exponentiated gradient method may be advantageous over the gradient method when the problem contains many irrelevant variables, in the context of the online framework we discuss in the next section.

2.14 Regret Analysis

As an alternative to asymptotic normality or iteration complexity analysis of the rate of convergence, a third way to assess a form of rate of convergence is through *regret* analysis. Regret analysis is one of the main tools used in the analysis of algorithms in the *online* framework, where we receive our data in a sequential fashion and our goal is to minimize the total loss obtained over the whole data sequence. As an example, consider a variant on the linear least squares framework where at iteration k:

1. We receive a new example, $\mathbf{a}_k$.
2. We make a prediction about the new example, $\mathbf{a}_k^T\mathbf{x}^k$.
3. We receive the true target for this example, b_k.
4. We are assessed the squared error between the prediction and the target, $(\mathbf{a}_k^T\mathbf{x}^k - b_k)^2$.

In this scenario, the goal of the online prediction algorithm is to update $\mathbf{x}^k$ to try and minimize the total error across the iterations. We typically express the performance of an online prediction algorithm in terms of a bound on either its total loss or on its *regret*:

$$R(\mathbf{x}^1, \ldots, \mathbf{x}^k) = \sum_{i=1}^{k}(\mathbf{a}_k^T\mathbf{x}^k - b_k)^2 - \min_{\mathbf{x}\in\mathcal{C}}\sum_{i=1}^{k}(\mathbf{a}_k^T\mathbf{x} - b_k)^2.$$

That is, the regret is the difference between the total error achieved by the algorithm, which does not know the targets before making predictions, and the total error achieved by the best possible fixed $\mathbf{x}$ within a restricted set $\mathcal{C}$. If we have no restriction on the data and we allow $\mathcal{C}$ to simply be $\mathbb{R}^p$, then the regret for a given algorithm can be made arbitrarily large by an adversary. However, it possible to bound the regret under restrictions on the data or the comparison class $\mathcal{C}$.

One typical restriction on the data is that all examples $\mathbf{a}_k$ are bounded in norm, $||\mathbf{a}_k||_2 \leq \gamma$ for all k and some positive γ. Another restriction is that the comparison class of fixed strategies $\mathcal{C}$ is forced to be a norm-ball of fixed radius. Note that the bounds that result from these assumptions do not rely on probabilistic arguments; bounds on the regret can be obtained for *arbitrary* sequences $\mathbf{a}_k$ and b_k, that may not even follow a particular distribution. However, these bounds may become tighter as the sequences $\mathbf{a}_k$ and b_k become 'nicer'. Often these bounds are expressed in terms of the number of iterations k, and we would like the bound to grow sub-linearly with k. For example, a bound of $O(k)$ implies the vacuous result that the regret simply grows with the number of iterations. In contrast, a bound of $o(k)$ would imply that regret decreases as we see more data.

One of the simplest online prediction algorithms uses incremental gradient iterations

$$\mathbf{x}^{k+1} \leftarrow \mathbf{x}^k - \alpha^k(\mathbf{a}_k^T\mathbf{x}^k - b_k)\mathbf{a}_k,$$

where the step size $\alpha^k > 0$ is a constant. Cesa-Bianchi et al. [1993] give worst-case bounds on this method (and several extensions of it) in a variety of scenarios, while they also discuss choosing the optimal step size in certain scenarios. They also examine bounds on the so-called *normalized* loss $(\mathbf{a}_k^T\mathbf{x}^k - b_k)/||\mathbf{x}||_2^2$ using iterations of the form

$$\mathbf{x}^{k+1} \leftarrow \mathbf{x}^k - \alpha^k\frac{(\mathbf{a}_k^T\mathbf{x}^k - b_k)}{\mathbf{a}_k^T\mathbf{a}_k}\mathbf{a}_k,$$

which is the Kaczmarz algorithm discussed in Section 5.1 but augmented with a step size. As opposed to the gradient method, Kivinen and Warmuth [1997] analyze the *exponentiated gradient* method in the online setting of making real-valued predictions under the squared-error measure. The analysis of Kivinen and Warmuth [1997] suggests that in the worst case the error of the exponentiated gradient method grows only logarithmically in the number of irrelevant features. In contrast, for the gradient method it grows linearly.

Kalai and Vempala [2002] derive regret bounds in the framework of *online linear optimization*. Here, we are given a feasible convex set and a sequence of linear objective vectors $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_k$. The goal is to minimize the regret in terms on the linear objective $\sum_{i=1}^{k} \mathbf{c}_i^T \mathbf{x}^k$. In this framework, we must make a prediction before seeing each $\mathbf{x}_i$, and regret is measured in comparison to the error made by the best possible vector $\mathbf{x}$ in the convex set. Zinkevich [2003] examined a generalization of online linear optimization known as online convex optimization. Here, the parameters are again restricted to a known (compact) convex set, but the sequence of linear objective functions are replaced by a sequence of general convex objective functions. Under bounds on the norms of the gradients of the sequence of convex functions, Zinkevich [2003] showed that a projected incremental gradient method achieves a sub-linear regret bound of $O(\sqrt{k})$. Flaxman et al. [2005] examined an even more restrictive setting, where we are not given not the convex function directly on each iteration but can only evaluate it at given parameter vectors (i.e. we do not have access to gradient information). Flaxman et al. [2005] show that with a simultaneous perturbation approximation of the gradient (see Section 2.6), a regret bound of $O(k^{3/4})$ can still be achieved.

Similar to the worst-case analysis of incremental sub-gradient methods, the regret bounds can be improved under assumptions about the second derivatives of the sequence of convex functions. In particular, Hazan et al. [2007] showed that is possible to achieve a regret of $O(\log k)$ under suitable restrictions on the curvature of the sequence of convex funtions. They show logarithmic regret is possible for an incremental gradient method, an incremental Newton method (using the natural gradient Hessian approximation), and a method that uses the previously computed function values to generate a new parameter vector.

3 Convergent Methods with Non-Vanishing Step-Sizes

Because of the noisy gradient estimates used by incremental gradient methods, it is crucial that the step sizes eventually converge to zero in order for the methods to converge. Unfortunately, the convergence rates of the previous section are sub-linear, and the requirement of vanishing step-sizes seems to be in conflict with achieving linear convergence rates.[5] In this section, we examine a relatively diverse variety of methods that have been proposed for obtaining a convergent method with noisy gradient estimates but a non-vanishing step size. These methods have been proposed in very different fields for very different purposes, and superficially don't appear very similar. However, they all share essentially the same idea for achieving convergence; instead of forcing the step size to go to zero, they force the *error* in the gradient estimate to go zero. Indeed, in Friedlander and Schmidt [2011] and Schmidt et al. [2011] we analyze the gradient and accelerated-gradient methods with error, and show that linear convergence rates can be obtained provided that the errors decrease at suitable rates.

3.1 Stronger Assumption on Gradients

One way to show convergence under a non-vanishing step size is to make additional assumptions about the function. For example, consider the assumption

$$||\nabla f_i(\mathbf{x})|| \leq D||\nabla f(\mathbf{x})||, \forall i, \mathbf{x}.$$

for some constant D. This is similar to the assumption used in Section 2, but where the affine function is replaced by a linear function. However, this is a much stronger assumptions and in particular it implies that

$$\nabla f(\mathbf{x}) = 0 \rightarrow \nabla f_i(\mathbf{x}) = 0, \tag{7}$$

[5] In the literature, linear convergence rates are sometimes known as *geometric* or *exponential* convergencer rates

meaning that any minimizer of f is also a minimizer of *each* f_i. Under this assumption, Solodov [1998] shows convergence of incremental gradient methods with a sufficiently small constant step size. The reason we can use a non-vanishing step-size under this assumption is clear; by continuity of the gradients we have that each $\nabla f_i(\mathbf{x})$ approaches $\nabla f(\mathbf{x})$ as x approaches a minimizer, as they all converge to zero. Hence, the noise in the gradient measurement vanishes as we approach the minimizer. Further, under strong-convexity assumptions it is also possible to show that this assumption leads to a linear convergence rate [Schmidt and Le Roux, 2012]. However, this assumption is not satisfied in most practical applications.

3.2 Batching

The idea behind *batching* is simple; we apply an incremental gradient method but as the algorithm progresses we group together a larger number of functions f_i into batches in order to obtain a more accurate gradient estimate. This method essentially tries to interpolate between an incremental and a deterministic gradient method. In early iterations the method is purely incremental, but as we progress we become less incremental as the size of the batches increases. Once all functions are batched together, we obtain a deterministic gradient method.[6]

Since batching effectively removes the noise in the gradient over time, we can use any results available for deterministic optimization methods to show convergence even with a non-vanishing step size. For example, it is possible to show global convergence *surely* rather than *almost surely*, since even instantiations of the random variable with probability zero are captured by the deterministic global convergence theory. Further, with batching we obtain asymptotic linear convergence rates by simply using a deterministic method with a linear convergence rate. Although batching appears to be more of a 'folk' algorithm used by practioners than a strategy whose theoretical properties have been examined, it is explicitly mentioned in an informal context by some authors such as Bertsekas and Tsitsiklis [1996, page 113]. In Friedlander and Schmidt [2011] we show non-asymptotic linear convergence rates for batching methods under both deterministic and stochastic batch-selection strategies.

3.3 Weighted Incremental Gradient

Bertsekas [1997] proposes another strategy for interpolating between the incremental gradient and the deterministic gradient method. To introduce this method, recall the first form of the incremental gradient method:

$$\mathbf{x}^{k+1} \leftarrow \mathbf{x}^k - \alpha^k \sum_{i=1}^{n} \nabla f_i(\mathbf{y}^i),$$

where on iteration k we define $\mathbf{y}^0 = \mathbf{x}^k$, and

$$\mathbf{y}^i \leftarrow \mathbf{y}^{i-1} - \alpha^k \nabla f_i(\mathbf{y}^{i-1}).$$

Again, we note that the deterministic method is obtained if we set $\mathbf{y}^i$ to $\mathbf{x}^k$ for all i. The essential idea behind the method of Bertsekas [1997] is to use a different formula for $\mathbf{y}^i$ that forces all $\mathbf{y}^i$ to converge to $\mathbf{x}^k$ as the iterations proceed. Specifically, the method sets $\mathbf{y}^0 = \mathbf{x}^k$ and sets

$$\mathbf{y}^i \leftarrow \mathbf{x}^k - \alpha^k \sum_{j=1}^{i} \frac{1 + \mu + \cdots + \mu^{i-j}}{1 + \mu + \cdots + \mu^{m-j}} \nabla f_i(\mathbf{y}^{i-1}).$$

When the parameter μ is zero we obtain the incremental gradient method, while as μ^k goes to ∞ we have that each $\mathbf{y}^i$ goes to $\mathbf{x}^k$ and the method converges to deterministic gradient iterations.[7] Further, Bertsekas [1997] shows that under certain assumptions the method has a linear convergence rate for minimizing strongly-convex quadratic functions if μ is increased at a sufficient rate.

[6] We can even consider an extreme variant where we apply an incremental gradient method for several iterations, and then switch to a deterministic optimization method [Agarwal et al., 2011].

[7] The update for $\mathbf{y}^i$ can be implemented sequentially so that we do not need to re-compute the weighted sum for each i.

Bertsekas [1997] mentions several heuristics that are required for the method to be effective in practice. First, it is mentioned that the weighting causes numerical issues and that batching can be used to reduce this. Second, he suggests using large step sizes in early iterations but decreasing the step size to a sufficiently small value once the method becomes sufficiently close to deterministic. Finally, he suggests several heuristics for updating μ. Although theoretically appealing, the method seems to be quite delicate in practice and the performance of the method seems to be sensitive to all of these issues.

3.4 Incremental Average Gradient

Recently, Blatt et al. [2008] presented a very different varient of the incremental gradient method that also achieves a linear convergence rate for the strongly-convex quadratic case. In their method they maintain n previous gradient vectors, one for each training example. Subsequently, on each iteration of their method the search direction is defined as the sum of these individual gradients. After each iteration, *one* of these vectors is updated in a cyclic fashion. Formally, beginning with a direction $\mathbf{d}^k$ that is the sum of n incremental gradient values at some n parameter values, the incremental average gradient (IAG) method uses iterations of the form:

$$\mathbf{x}^{k+1} \leftarrow \mathbf{x}^k - \alpha^k \mathbf{d}^k,$$

where $\mathbf{d}^k$ is updated using

$$\mathbf{d}^k \leftarrow \mathbf{d}^{k-1} - \nabla f_i(\mathbf{x}^{k-n}) + \nabla f_i(\mathbf{x}^k).$$

In the update for $\mathbf{d}^{k+1}$, the function i to update is chosen in a cyclic fashion. That is, the IAG method keeps the same gradient for each training example for k iterations, and on each iteration updates the gradient for a single training example. Since the update for $\mathbf{d}^k$ requires gradient values at previous iterates, the IAG method requires $O(np)$ storage. This is much higher than the $O(p)$ required by previous methods. However, note that the storage will be much smaller if the individual f_i are sparse.

That the IAG method has a linear convergence rate and is globally convergent with a constant step size is somewhat suprising, since unlike the weighted incremental gradient method it does *not* converge to a full-gradient algorithm. Further, it does not need to assume that each $f_i(\mathbf{x}^k)$ converges to zero as in Section 3.1. Rather, the IAG method takes advantage of the property that the sum of the gradient approximations $\mathbf{d}^k$ converges to zero as a minimizer is approached.

More recently, we considered a stochastic version of the IAG method in Le Roux et al. [2012], that we referred to as the stochastic average gradient (SAG) method. This method is identical to the IAG method, but we sample the gradient to update instead of using a cyclic order. Unlike the IAG method where the linear convergence rate depends on the number of passes through the data, the SAG method achieves a linear convergence rate that depends on the number of iterations. Further, a surprising result of this work is that when the number of training examples is sufficiently large the SAG method allows the use of a very large step size, which leads to improved theoretical and empirical performance.

3.5 Sample Average Approximation

The sample average approximation (SAA) method of Kleywegt et al. [2002] is a simple method that is closely related to batching, but was developed in the context of optimizing an inherently noisy function over a discrete set. In the SAA method, we randomly generate some number of samples m. We then solve the problem exactly with respect to these m samples. Results from probability theory are used to argue that the optimal value over the discrete set with respect to the samples converges to the true optimal value of the noisy function.

SAA is explored further in [Shapiro and Nemirovski, 2005], including the possibility of optimizing over continuous sets. An appealing property of the method is that fast deterministic algorithms can be used for the optimization over the small batch of samples. Currently, there is little guidance from the SAA literature on theoretically sound ways to chose the number of samples (or how to increase the number of samples over time). While we could examine empirical estimates of variance based on the samples, such estimates tend to be over-optimistic and may lead to premature convergence before a sufficiently large sample is chosen.

3.6 Implicit Filtering

Similar to the SAA, implicit filtering methods are designed to optimize inherently noisy functions. To avoid requiring a vanishing step size, in this literature it is assumed that the noise asymptotically vanishes as a minimizer is approached, which is similar to making a strong assumption on the gradients as in Section 3.1. The main idea behind implicit filtering methods is to use a finite-difference approximation to the gradient, but using a potentially large perturbation to avoid the effect of local noise. Subsequently, this perturbation is decreased over time. A first-order implicit filtering algorithm is discussed in Gilmore and Kelley [1995], while Choi and Kelley [2000] present a quasi-Newton implicit filtering algorithm with a local superlinear convergence rate.

4 Decomposition Methods

Recall our original problem

$$\min_{\mathbf{x}} \sum_{i=1}^{n} f_i(\mathbf{x}).$$

It would be appealing if we could minimize with respect to each $f_i(\mathbf{x})$, possibly in parallel, and then somehow efficiently combine the results to obtain the final solution. Unfortunately, since the functions $f_i(\mathbf{x})$ are not separable we can not do these minimization independently. In decomposition methods, we *introduce* constraints in order to yield a separable objective function. Subsequently, we can minimize with respect to each function individually in order to implement a block coordinate descent or gradient method.

4.1 Penalty Methods and Alternating Direction Method of Multipliers

Bertsekas and Tsitsiklis [1989, Example 4.2] suggest the following re-formulation of the problem to make the objective function separable:

$$\min_{\mathbf{z},\mathbf{x}_1,\ldots,\mathbf{x}_n} \sum_{i=1}^{n} f_i(\mathbf{x}_i), \text{ s.t. } \mathbf{x}_i = \mathbf{z}, \forall_i.$$

In this formulation we have a copy of the parameter vector $\mathbf{x}_i$ for *each* function f_i, but the constraints enforce that all of these parameter vectors must be equal. There are a variety of strategies available for solving this constrained optimization problem. One of the simplest we can imagine is to encourage feasibility by using a penalty function, resulting in the problem:

$$\min_{\mathbf{z},\mathbf{x}_1,\ldots,\mathbf{x}_n} \sum_{i=1}^{n} f_i(\mathbf{x}_i) + \frac{\rho}{2} \sum_{i=1}^{n} ||\mathbf{x}_i - \mathbf{z}||_2^2,$$

for some $\rho \geq 0$. The solution of this problem approaches the solution of the original problem as ρ increases. However, unlike the original problem we can optimize this objective by a block coordinate descent method where the update for each $\mathbf{x}_i$ has the form

$$\mathbf{x}_i \leftarrow \arg\min_{\mathbf{x}} \{f_i(\mathbf{x}) + \frac{\rho}{2}||\mathbf{x} - \mathbf{z}||_2^2\}, \tag{8}$$

and to update $\mathbf{z}$ we simply set it to the mean $(1/n)\sum_{i=1}^{n} \mathbf{x}_i$. We recognize the update with respect to $\mathbf{x}_i$ as the proximity operator for f_i computed at $\mathbf{z}$. Note that we can compute these proximity operators (8) *in parallel* for all i, since the updates all use the same proximal center $\mathbf{z}$.[8] An important property of this method is that it does not require a diminishing sequence of step sizes to converge to the solution for a fixed value of ρ, and further that coordinate descent methods have an $O(1/k)$ rate if f is convex and a linear

[8] We could also consider variants where $\mathbf{z}$ is updated more frequently, though this can affect the ability to parallelize the method.

rate if f is strongly convex [see Nesterov, 2010]. However, the problem becomes more difficult to solve as ρ increases, and the penalized formulation is only equivalent to the original problem as ρ goes to ∞.

Rather than simply using a penalty function, we could also consider an augmented Lagrangian formulation:

$$\min_{\mathbf{z},\mathbf{x}_1,\ldots,\mathbf{x}_n} \sum_{i=1}^{n} f_i(\mathbf{x}_i) + \frac{\rho}{2}\sum_{i=1}^{n} ||\mathbf{x}_i - \mathbf{z}||_2^2 + \sum_{i=1}^{n} \mathbf{p}_i^T(\mathbf{x}_i - \mathbf{z}).$$

The advantage of this formulation is that for a suitable value of the Lagrange multiplier estimates $\mathbf{p}_i$, this problem yields the same solution as the original problem for a finite value of ρ. Bertsekas and Tsitsiklis [1989, §3.4] discuss the possibility of using an alternating-direction method of multipliers (ADMM) method, where we cycle between optimizing $\mathbf{z}$, optimizing each $\mathbf{x}_i$, and then updating each $\mathbf{p}_i$ (or updating the multipliers between updates to the $\mathbf{x}_i$ variables).[9] The ADMM update for each $\mathbf{x}_i$ would take the form

$$\mathbf{x}_i \leftarrow \arg\min_{\mathbf{x}} \{f_i(\mathbf{x}) + \frac{\rho}{2}||\mathbf{x} - \mathbf{z}||_2^2 + \mathbf{p}_i^T(\mathbf{x} - \mathbf{z})\},$$

and the update for $\mathbf{z}$ would take the form

$$\mathbf{z} \leftarrow (1/n)\sum_{i=1}^{n} \mathbf{x}_i - \frac{1}{n\rho}\sum_{i=1}^{n} \mathbf{p}_i.$$

Further, we typically use a first-order update for the dual variables $\mathbf{p}_i$ of the form

$$\mathbf{p}_i \leftarrow \mathbf{p}_i + \rho(\mathbf{z} - \mathbf{x}_i).$$

For an extensive recent discussion of ADMM, see Boyd et al. [2010].

Another alternative strategy to solve the constrained optimization problem would be to solve an exact penalty problem:

$$\min_{\mathbf{z},\mathbf{x}_1,\ldots,\mathbf{x}_n} \sum_{i=1}^{n} f_i(\mathbf{x}_i) + \frac{\rho}{2}\sum_{i=1}^{n} ||\mathbf{x}_i - \mathbf{z}||_1.$$

Similar to the augmented Lagrangian formulation, this formulation has the advantage that it shares the same solution as the original problem without requiring ρ to be arbitrarily large. In this case, the update for each $f_i(\mathbf{x})$ can be formulated as a *shifted* ℓ_1-regularization problem. To update $\mathbf{z}$, we set it the median of the $\mathbf{x}_i$ variables. However, the use of this method is complicated by the non-differentiability of the non-separable exact penalty function, since this may prevent block coordinate descent strategies from converging to the solution.

An alternative to introducing an auxiliary variable $\mathbf{z}$ and introducing n equality constraints forcing each $\mathbf{x}_i$ to be equal to $\mathbf{z}$, is to simply use $(n-1)$ equalities to force the $\mathbf{x}_i$ to be equal to each other. For example, we could consider the formulation

$$\min_{\mathbf{x}_1,\ldots,\mathbf{x}_n} \sum_{i=1}^{n} f_i(\mathbf{x}_i), \text{ s.t. } \mathbf{x}_i = \mathbf{x}_{i+1}, \forall i < n-1.$$

This approach could similarly be addressed with penalty or augmented Lagrangian methods, and might be more appealing in a distributed setting since each processor would only need to communicate with its immediate neighbors (rather than requiring each processor to maintain a copy of the global variable $\mathbf{z}$). Another variant with $(n-1)$ equalities is discussed in the next section.

There are two major disadvantages of these types of methods. First, since the methods perform block coordinate descent on a potentially ill-conditioned optimization problem, they may require a large number of iterations. Second, they require $O(np)$ storage to store the n copies of the parameter vector. Indeed, even in cases where the individual gradients f_i are sparse, the parameter vectors x_i will typically be dense. Nevertheless, these methods may be appealing in a distributed setting where each processor requires only one copy of the parameter vector, and the block coordinate descent iterations can be performed in parallel.

[9] ADMM differs from a standard augmented Lagrangian method in that we minimize with respect to groups of primal variables separately, rather than jointly.

4.2 Fenchel Dual Methods

Bertsekas and Tsitsiklis [1989, Example 4.1] outline a variant of the parameter replication method where we work with a certain Fenchel dual of the original problem in the case where each $f_i(\mathbf{x})$ is strictly convex. In this method, we choose one copy of the parameter vector $\mathbf{x}_1$ and optimize subject to the constraint that all parameter vectors are equal to this one:

$$\min_{\mathbf{x},\mathbf{x}_1,\ldots,\mathbf{x}_n} f_1(\mathbf{x}) + \sum_{i=2}^{n} f_i(\mathbf{x}_i), \mathbf{x}_i = \mathbf{x}, \forall_{i>1}.$$

From this, we can derive a dual problem expressed in terms of convex conjugate functions:

$$\max_{\mathbf{p}_1,\ldots,\mathbf{p}_n} q(\mathbf{p}) = q_1(\sum_{i=2}^{n} \mathbf{p}_i) + \sum_{i=2}^{n} q_i(\mathbf{p}_i),$$

where

$$q_1(\sum_{i=2}^{n} \mathbf{p}_i) = \min_{\mathbf{x}}\{f_1(\mathbf{x}) - (\sum_{i=2}^{n} \mathbf{p}_i)^T\mathbf{x}\} \tag{9}$$

and for i greater than one we have

$$q_i(\mathbf{p}_i) = \min_{\mathbf{x}_i}\{f_i(\mathbf{x}_i) + \mathbf{p}_i^T\mathbf{x}_i\}. \tag{10}$$

The objective function in this dual is continuously differentiable with gradient

$$\frac{\partial q(\mathbf{p})}{\partial \mathbf{p}_i} = \mathbf{x}_i(\mathbf{p}) - \mathbf{x}(\mathbf{p}),$$

where $\mathbf{x}_i(\mathbf{p})$ and $\mathbf{x}(\mathbf{p})$ are the minimizers of q_i and q_1, respectively. The convex conjuage functions (9) and (10) required to compute the gradient can be computed in parallel (and have simple solutions for many objective funtions), and we can use this within a gradient-based method to optimize the dual objective.

4.3 Aside: Simple Parallelization

Bertsekas and Tsitsiklis [1989] recommend decomposition methods for settings with distributed computation, due to the ease of parallelizing the iterations for many of the above methods. However, note that it is not immediately clear that these methods are particularly advantageous compared to the 'obvious' way of parallelizing a sequential optimization algorithm for our problem. Namely, we simply split the n examples across the processors, and each processor is responsible for computing the part of the function value and gradient arising from those examples. This naive strategy appears to have a high communication cost since it requires transferring the parameter vector $\mathbf{x}^k$ to each processor on each iteration, while each processor must return some part of the gradient to a central processor. However, the cost of communication compared to evaluating each $f_i(\mathbf{x})$ will vary with the application, the number of processors, and the hardware available. Further, for most of the decomposition methods discussed above it is not immediately clear that the decomposition methods will require a substantially smaller communication cost. For example, the parameter replication of Bertsekas and Tsitsiklis [1989, Example 4.2] seems to have the same communication cost (per iteration) as the naive method, since we must communiate the $\mathbf{z}$ variable to all processors, while to update $\mathbf{z}$ we must have all the $\mathbf{x}_i$ variables. Similarly, the dual method of Bertsekas and Tsitsiklis [1989, Example 4.1] requries communicating $\mathbf{x}(\mathbf{p})$ to all processors to implement the parallel gradient update, while to compute $q_1(\mathbf{p})$ we must have all the $\mathbf{p}_i$ variables.

4.4 Fenchel Dual Methods (with fewer variables)

There are a variety of decomposition methods used in machine learning that are closely related to the dual method discussed above. However, rather than requiring $O(np)$ dual variables to enforce $O(np)$ constraints, these methods only introduce n constraints and thus only have n dual variables. These dual problems are often solved with coordinate descent methods.

One way to derive a dual problem with n variables applies when we have an ℓ_2-regularization term and the individual functions are a composition of a linear function:[10]

$$\min_{\mathbf{x}} \frac{\lambda}{2}||\mathbf{x}||_2^2 + \sum_{i=1}^{n} f_i(\mathbf{a}_i^T\mathbf{x}).$$

We introduce a new vector of variables $\mathbf{y}$ to transform this into a constrained optimization problem with a separable objective function and linear constraints:

$$\min_{\mathbf{x},\mathbf{y}} \frac{\lambda}{2}||\mathbf{x}||_2^2 + \sum_{i=1}^{n} f_i(y_i), \text{ s.t. } y_i = \mathbf{a}_i^T\mathbf{x}, \forall i$$

The dual function for this problem has a quadratic term and the convex conjugates of the individual functions. Unlike the dual method discussed previously where we have np variables, the dual function for this problem will only have n variables. The application of a coordinate descent method to this dual for the logistic regression problem was examined in [Keerthi et al., 2005].

Another formulation where we introduce n constraints to change the sum over functions into a separable form arises if we optimize a linear function over the epigraphs of the individual functions,

$$\min_{\mathbf{x},\mathbf{y}} \frac{\lambda}{2}||\mathbf{x}||_2^2 + \sum_{i=1}^{n} y_i, \text{ s.t. } y_i \geq f_i(\mathbf{x}), \forall_i. \tag{11}$$

Similar to the above, the dual function for this problem has only n variables. This construction results in bound constraints on the dual variables, but these constraints have a separable structure and thus do not prevent convergence of coordinate descent methods on the dual. Solving support vector machine optimization problems by applying coordinate descent and this dual decomposition method was popularized under the name sequential minimal optimization (SMO) by Platt [1999].[11] A more recent variant of this method was examined by Tseng and Yun [2010].

For multinomial (i.e. non-binary) generalizations of logistic regression and support vector machines, we can derive similar dual problems that have a simplex constraint on the parameters for each example. Thus, the dual problem involves optimizing a smooth objective over a product space consisting of n simplexes. Collins et al. [2008] examined solving these dual problems using a randomized (incremental) exponentiated gradient method (see Section 2.13) to implicitly address the simplex constraints. They show that their method has an error of $O(1/k)$ for the dual objective under an assumption that is similar to strong convexity, but where the squared Euclidean norm is replaced by the Kullback-Leibler divergence. Although the method only has a $O(1/\sqrt{k})$ error for the primal objective, they show that under slightly stronger assumptions that the incremental method has a linear convergence rate in both the primal and dual objectives. This rate, which is potentially much faster than the rates of primal incremental gradient methods, is possible because the incremental dual method is effectively implementing a block coordinate descent step on the dual.

5 Row-Action Methods

Row-action methods are a class of methods designed for the solution of large-scale linear feasibility problems, but that can also be extended to solve convex feasibility problems and certain convex optimization problems.

[10] More generally, we can consider a composition of an affine function

[11] Technically, they work with a slightly different dual since the ℓ_2-regularization is not applied to all terms. This results in a simple equality constraint in the dual problem, and the method optimizes over two variables at a time instead of one in order to maintain the equality.

Censor [1981] defines a a row-action method in the context of finding a solution to a consistent linear system $\mathbf{Ax} = \mathbf{b}$ as an iterative algorithm with the following properties:

1. no changes are made to the original matrix;
2. no operations are performed on the matrix as a whole;
3. in a single iterative step access is required to *only one row* of the matrix;
4. in a single iterative step, say when $\mathbf{x}^{k+1}$ is calculated, the only iterate needed is the intermediate predecessor $\mathbf{x}^k$.

These properties do not really uniquely define the class of row-action methods. For example, if we apply an incremental gradient method to minimize $||\mathbf{Ax} - \mathbf{b}||_2^2$ then it would satisfy all four properties of a row-action method, even though incremental gradient methods are not typically viewed as row-action methods. What seems to really define row-action methods is that each iteration of a row-action method consists of computing a *projection* with respect to a single constraint.[12]

Most of the literature on row-action methods focuses on the case of solving linear feasibility problems. We review this next, in addition to extensions to solving convex feasibility problems. Subsequently, we discuss using row-action methods to solve projection problems before moving on to methods for solving general (strictly) convex optimization problems. Given the various flavours of row-action methods, there are several ways that row-action methods can be applied to solve optimization problems:

1. We can convert the optimization problem into a feasibility problem.
2. We can convert the optimization problem into a projection problem.
3. We can use a row-action method that solves general optimization problems.

As an example of the first case, we could simply apply a row-action method to find a solution to

$$\sum_{i=1}^{n} \nabla f(\mathbf{x}) = \mathbf{0}.$$

In this case, row-action methods correspond to coordinate descent with an exact line search, as we would seek to satisfy the optimality conditions with respect to one variable at a time. Note that this does not separate the problem across the n functions, and so our discussion will focus on the latter two approaches.

5.1 Row-Action Methods for Feasibility

Censor [1981, §4.1] identifies the Kaczmarz algorithm (published in 1937, and sometimes referred to as the algebraic reconstruction technique) as one of the oldest row-action methods. In the most basic form of the Kaczmarz method, we seek to find a solution of the consistent linear system $\mathbf{Ax} = \mathbf{b}$ using iterations of the form

$$\mathbf{x}^{k+1} \leftarrow \mathbf{x}^k + \frac{b_i - \mathbf{a}_i^T \mathbf{x}^k}{\mathbf{a}_i^T \mathbf{a}_i} \mathbf{a}_i,$$

where we choose the row to access in a cyclic fasion. This iteration is simply the projection of $\mathbf{x}^k$ onto the hyper-plane defined by row i of the matrix, $\{\mathbf{x} | \mathbf{a}_i^T \mathbf{x} = b_i\}$, and the method is globally convergent for any choice of $\mathbf{x}^0$. Note that this can also be viewed as an iteration of the cyclic incremental gradient method for minimizing the corresponding least squares problem $||\mathbf{Ax} - \mathbf{b}||_2^2$, but where the step size is given by the row-specific $\alpha_i = 1/||\mathbf{a}_i||_2^2$.

[12] Bertsekas and Tsitsiklis [1989, §3.1] refer to a similar class of methods as component-solution methods.

It is straightforward to adapt the Kaczmarz algorithm to find a vector $\mathbf{x}$ satisfying the inequality constraints $\mathbf{Ax} \leq \mathbf{b}$. We simply replace the projections onto hyper-planes by projections onto half-spaces to obtain iterations of the form

$$\mathbf{x}^{k+1} \leftarrow \mathbf{x}^k + \min\{0, \frac{b_i - \mathbf{a}_i^T\mathbf{x}^k}{\mathbf{a}_i^T\mathbf{a}_i}\}\mathbf{a}_i.$$

According to Censor [1981, 4.2], this method was independently proposed by Agmon as well as Motzkin and Schoenberg in 1954. We can combine both types of iterations if we have both equality and inequality constraints.

We can also consider row-action methods for solving the convex feasibility problem of finding a point $\mathbf{x}$ in a convex set $\mathcal{C}$ defined as the intersection of convex sets, $\mathcal{C} \equiv \cap_i \mathcal{C}_i$. In this case we obtain the successive projection iteration

$$\mathbf{x}^{k+1} \leftarrow \mathcal{P}_{\mathcal{C}_i}[\mathbf{x}^k],$$

where $\mathcal{P}_{\mathcal{C}_i}$ denotes projection onto the convex set $\mathcal{C}_i$. This method dates back to Bregman [1965]. While it is a generalization of the case of linear equality and inequality constraints, the iterations of the more general method will only be efficient if the individual convex sets have a relatively simple structure.

When the individual sets represent subspaces, row-action methods have a linear convergence rate where the constant depends on the minimum 'angle' between the subspaces [see Galantai, 2005]. This linear convergence rate can be generalized to the case where the individual sets are polyhedral [Deutsch and Hundal, 1997]. There do not appear to be results on the convergence rate of the method for general convex sets.

Before moving on to row-action methods for optimization, we now briefly review several of the enhancements and extensions of basic row-action methods that have been proposed. Several alternatives to updating the variables in a cyclic fasion are discussed in [Censor, 1981, §3], such as *almost cyclic* sequences which only require that we have projected onto each constraint at least once for every C iterations (where C is some positive integer). It also possible to replace the Euclidean projection with a Bregman projection [Censor, 1981, §4.6], such as an entropy projection (for variables that are strictly positive) which is related to the exponentiated gradient algorithm discussed in Section 2.13. In cases where computing the projection onto constraint i is expensive, some variants consider taking a negative sub-gradient step with respect to the constraint [Censor, 1981, §5.1]. To speed the convergence of the iterations, many works consider under-relaxed and over-relaxed methods where a step size $\lambda^k \in (0, 2)$ is used, yielding the update

$$\mathbf{x}^{k+1} \leftarrow \mathbf{x}^k + \lambda^k \frac{b_i - \mathbf{a}_i^T\mathbf{x}^k}{\mathbf{a}_i^T\mathbf{a}_i}\mathbf{a}_i.$$

Yuan and Matoli [1998] give expressions for values of λ^k that optimize the convergence rates of various row-action methods. There also exist variants of the method that compute all projections in parallel rather than sequentially, such as [Bertsekas and Tsitsiklis, 1989, Example 4.3]. Recently, a randomized variant of the Kaczmarz method was examined by Strohmer and Vershynin [2009], who show that the method has a linear convergence rate in expectation. Similar to randomized incremental gradient methods, the convergence rate of the randomized algorithm does not require a full cycle through the data on each iteration. However, rather than sampling the constraints uniformly as in incremental gradient methods, they sample each constraint in proportion to $||\mathbf{a}_i||_2^2$. Taking this sampling distribution into account, we see that the expected direction of the randomized method has the form:

$$\mathbb{E}[\frac{b_i - \mathbf{a}_i^T\mathbf{x}^k}{\mathbf{a}_i^T\mathbf{a}_i}\mathbf{a}_i] = \sum_{i=1}^n p(i)\frac{b_i - \mathbf{a}_i^T\mathbf{x}^k}{\mathbf{a}_i^T\mathbf{a}_i}\mathbf{a}_i = \sum_{i=1}^n \frac{||\mathbf{a}_i||_2^2}{||\mathbf{A}||_F^2}\frac{b_i - \mathbf{a}_i^T\mathbf{x}^k}{||\mathbf{a}_i||_2^2}\mathbf{a}_i = \sum_{i=1}^n \frac{b_i - \mathbf{a}_i^T\mathbf{x}^k}{||\mathbf{A}||_F^2}\mathbf{a}_i.$$

That is, the average direction generated by the randomized Kaczmarz algorithm for solving a linear system is identical to the average direction generated by the stochstic gradient descent algorithm applied to a least squares problem, where we set the step size α^k to the constant $1/||A||_F^2$ for all k.[13]. Of course, this leaves us a bit confused since we know that in general the stochastic gradient descent algorithm does *not* converge for

[13]Note that $||A||_F^2$ is larger than $||A||_2^2$, the Lipschitz constant of the gradient in the least-squares problem.

a constant step size. However, the Kaczmarz algorithm assumes that the linear system is *consistent*. This assumption is similar to the assumption discussed in Section 3.1 that allows the incremental gradient method to converge for a constant step size. In particular, assuming that the linear system is consistent implies that property (7) holds for the corresponding least-squares problem. An enhancement of the randomized Kaczmarz method was recently examined by Eldar and Needell [2010], who considered selecting the best row to update among a selection of rows. This can lead to faster convergence but it can be computationally expensive to evaluate the rows, so Eldar and Needell [2010] considered using the fast Johnson-Lindenstrauss transform to quickly evaluate the rows.

5.2 Row-Action Methods for Projection

While the Kaczmarz algorithm converges to a solution of $\mathbf{Ax} = \mathbf{b}$ for any starting point, if we set $\mathbf{x}^0 = \mathbf{0}$ then it converges to the *minimum norm* solution,

$$\min_{\mathbf{x}} ||\mathbf{x}||_2, \text{ s.t. } \mathbf{Ax} = \mathbf{b},$$

see [Censor, 1981, §4.1]. A closely related method is Hildreth's algorithm, which can be used to solve problems of the form

$$\min_{\mathbf{x}} ||\mathbf{x}||_2, \text{ s.t. } \mathbf{Ax} \leq \mathbf{b}.$$

Hildreth's algorithm [Censor, 1981, §4.4] is a primal-dual algorithm that, beginning with some arbitrary non-negative $\mathbf{z}^0$, sets $\mathbf{x}^0$ to $-A^T\mathbf{z}$ and uses iterations of the form

$$\mathbf{x}^{k+1} \leftarrow \mathbf{x}^k + c\mathbf{a}_i,$$

$$\mathbf{z}^{k+1} \leftarrow \mathbf{z}^k - c\mathbf{e}_i.$$

where

$$c = \min\{z_i, \frac{b_i - \mathbf{a}_i^T\mathbf{x}^k}{||\mathbf{a}_i||_2^2}\}$$

Compared to the row-action method for solving linear feasilibity problems, these updates to $\mathbf{x}^k$ differ only in replacing 0 with z_i when computing the variable c. Note that this modification may yield a non-zero step even when constraint i is satisfied. Bregman's method [Censor, 1981, §4.7] is a generalization of Hildreth's algorithm that uses Bregman projections to minimize a Bregman function subject to linear constraints. For example, we could use Bregman's method to solve the optimization problem

$$\min_{\mathbf{x}>\mathbf{0}} \sum_{i=1}^{p} x_i \log x_i, \text{ s.t. } \mathbf{Ax} \leq \mathbf{b}.$$

An alternative method for this special case is the *multiplicative* albegraic reconstruction technique [Censor, 1981, §4.9]. The MART method begins with $x_i^0 = 1/e$ and uses multiplicative updates of the form

$$x_j^{k+1} \leftarrow \left(\frac{b_i}{\mathbf{a}_i^T\mathbf{x}^k}\right)^{(\mathbf{a}_i)_j} x_j^k,$$

for each j.

A closely related problem to findining the minimum-norm feasible solution is computing the projection of the initial point $\mathbf{x}^0$ onto the feasible set. Predating the Kaczmarz algorithm, in his 1933-34 notes von Neumann [1950] showed that the limit of cyclically projecting onto two subspaces converges to the projection of the initial point onto the subspace (this result has subsequently been generalized to an arbitrary set of subspaces). For example, to solve the projection problem

$$\min_{\mathbf{x}} ||\mathbf{x} - \mathbf{x}^0||_2, \text{ s.t. } \mathbf{Ax} = \mathbf{0},$$

we can simply apply the Kaczmarz algorithm to the system $\mathbf{A}\mathbf{x} = \mathbf{0}$ beginning from $\mathbf{x}^0$. The algorithm of Dykstra [1983] is a generalization of Hildreth's algorithm (and sometimes known as Han's algorithm) that allows us to compute the projection of $\mathbf{x}^0$ onto the intersection of arbitrary closed convex sets,

$$\min_{\mathbf{x}} ||\mathbf{x} - \mathbf{x}^0||_2, \text{ s.t. } \mathbf{x} \in \mathcal{C}_i[\mathbf{x}], \forall_i.$$

When applied to polyhedral sets, Dykstra's algorithm is equivalent to Hildreth's algorithm. When applied to non-affine sets, Dykstra's algorithm still solves the projection problem but does not satisfy all the properties of row-action methods since it requires $O(np)$ storage. Deutsch and Hundal [1994] show that Dykstra's algorithm has a linear convergence rate for polyhedral sets. Bauschke and Lewis [2000] show convergence of a variant of Dykstra's algorithm for minimizing Bregman distances. Gaffke and Mathar [1989] give an interpretation of Dykstra's algorithm as a particular dual optimization strategy.

5.3 Row-Action Methods for Optimization

A row-action method for minimizing a strictly convex function $f(\mathbf{x})$ over the intersection of convex sets,

$$\min_{\mathbf{x}} f(\mathbf{x}), \text{ s.t. } g_i(\mathbf{x}) \leq 0, \forall_i,$$

is described by Iusem and Svaiter [1994]. Each iteration of the method takes the form

$$\mathbf{x}^{k+1} \leftarrow \min_{\mathbf{x}} f(\mathbf{x}), \text{ s.t. } \mathbf{x} \in \mathcal{L}(\mathbf{x}^k) \cap \mathcal{S}_i(\mathbf{x}^k),$$

where $\mathcal{L}(\mathbf{x}^k) \triangleq \{\mathbf{x} | \nabla f(\mathbf{x}^k)^T(\mathbf{x} - \mathbf{x}^k) \geq 0\}$ and $\mathcal{S}_i(\mathbf{x}^k) \triangleq \{\mathbf{x} | \nabla \mathbf{g}(\mathbf{x}^k)^T(\mathbf{x} - \mathbf{x}^k) \geq \mathbf{g}(\mathbf{x}^k)\}$. That is, the original strictly convex function is minimized over a supporting hyper-plane of the epigraph of $f(\mathbf{x})$ and a supporting hyper-plane to *one* of the constraints. Thus, the approach is closely related to cutting-plane methods for convex optimization. This approach will only be efficient when both the constraint functions $g_i(\mathbf{x})$ and the objective function $f(\mathbf{x})$ are relatively simple. Iusem and Svaiter [1995] describe a primal-dual variant of this method, and show a linear convergence rate when the constraints are linear.

Bauschke and Combettes [2008] give a variant of Dykstra's algorithm that can compute the proximal operator

$$\min_{\mathbf{x}} \frac{1}{2}||\mathbf{x} - \mathbf{x}^0||_2^2 + \sum_{i=1}^{n} f_i(\mathbf{x}),$$

where in the iterations projection onto individual convex sets is replaced by computing the proximal operator with respect to individual functions f_i (we require that these functions are convex but not necessarily smooth). We could use this variant of Dykstra's algorithm to solve ℓ_2-regularized optimization problems of the form

$$\min_{\mathbf{x}} \frac{\lambda}{2}||\mathbf{x}||_2^2 + \sum_{i=1}^{n} f_i(\mathbf{x}).$$

Beginning with $\mathbf{x}^0 = \mathbf{0}$ and $\mathbf{y}^i = \mathbf{0}$ for $i \in \{1, \dots, n\}$, this method would take steps of the form:

$$\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}} \{f_i(\mathbf{x}) + \frac{\lambda}{2}||\mathbf{x} - (\mathbf{x}^k - \mathbf{y}^i)||_2^2\}$$

$$\mathbf{y}^i \leftarrow \mathbf{x}^{k+1} - (\mathbf{x}^k - \mathbf{y}^i),$$

where i is chosen in a cyclic fasion. We note that this is very similar to the incremental proximal method discussed in Section 2.10. However, there are two notable differences. First, the proximal variant of Dykstra's algorithm projects the point $(\mathbf{x}^k - \mathbf{y}^i)$ rather than simply $\mathbf{x}^k$. Second, the proximal version of Dykstra's algorithm *does not have a step size*, and thus does not need a diminishing sequence of step sizes to ensure convergence. This method is also similar to the penalty-based decomposition method discussed in Section 4.1, but differs in that it *does not require setting a penalty parameter* and it applies the proximal operator sequentially rather than in parallel.

6 Randomized Linear Algebra

In many cases of practical interest we have the special structure

$$\min_{\mathbf{x}} \sum_{i=1}^{n} f_i(\mathbf{a}_i^T \mathbf{x}), \tag{12}$$

and the dominant cost of the algorithm is the matrix-vector multiplications $A\mathbf{x}$ and $A^T\mathbf{r}$ for some vector r (where the rows of A are the individual a_i). Note that the matrix A only appears in the problem through multiplication by vectors, and the problem can be solved considerably faster if multiplication by A and A^T is fast. This is possible if A has a special structure, for example if A is a special matrix like a tridiagonal matrix or if we have a low-rank decomposition of A. The idea behind *randomized linear algebra* methods is to replace the original A (which may not have a nice structure) by a low-rank approximation that forms an approximate basis for the range of A. Subsequently, we can efficiently solve the problem with the approximate basis as an approximation to the solution of the original problem. The approaches described in this section are relatively new and evolving, and the exposition that follows is roughly based on the recent survey by Halko et al. [2009].

6.1 Meta-Algorithm

Randomized linear algebra methods often proceed in two stages:

1. We compute a low-rank orthogonal matrix Q such that

$$A \approx Q(Q^T A).$$

2. Giving the low-rank matrix $B = Q^T A$, we exactly solve the original problem using the approximations

$$Ax \approx Q(Bx), \quad A^T r \approx B^T(Q^T r).$$

Normally, each iteration of a standard gradient method to solve (12) would require a cost of $O(np)$ for the matrix-vector multiplications. However, if the rank of Q is k, then the second step of a randomized linear algebra technique will only require $O(nk + pk)$. If k is small, this can lead to an enormous reduction in the runtime for scenarios where both n and p are large. Note that there are a large amount of variations on this theme, see [Halko et al., 2009, §2].

6.2 Randomized Algorithm

We will only have a net gain in computation if the first stage is not too expensive. In randomized linear algebra methods, we multiply A by random vectors in order to cheaply construct the basis in the first stage. In particular, suppose that A has an approximate rank k, and we compute the products $\mathbf{y}^i = A\mathbf{z}^i$ for $k + m$ random vectors $\mathbf{z}^i$ (where m is some typically small over-sampling parameter). We can then obtain Q by constructing an orthonormal basis for the range of the $\mathbf{y}^i$ values. Under this scheme, the main computational cost of the first step is computing the matrix-vector products $A\mathbf{z}^i$. In the case of Gaussian random vectors, the total cost for these matrix-vector products will be $O(np(k+m))$. This cost will be asymptotically cheaper than running a deterministic algorithm provided that $k + m$ is lower than the number of iterations that the deterministic algorithm requires. This may be perfectly reasonable if $k + m$ is small. Further, we can reduce the computational cost by using a structured random matrix. In particular, if the subsampled random Fourier transform is used the cost of the first stage is reduced to $O(np \log(k + m))$.

Even if it leads to a lower computational cost, randomized linear algebra methods will only be useful if the approximation $Q(Q^T A)$ of A is sufficiently acccurate. Theorem 1.1 of Halko et al. [2009] gives the following bound on the expected error for Gaussian random vectors,

$$\mathbb{E}[||A - Q(Q^T A)||] \leq \left[1 + \frac{4\sqrt{k+m}}{m-1}\sqrt{\min\{n, p\}}\right] \sigma_{k+1},$$

where $k \geq 2$, $m \geq 2$, and σ_{k+1} is the $(k+1)$ singular value of A. Thus, if the spectrum of A decays quickly, we expect the approximation to be very accurate. However, the accuracy does not increase very quickly in terms of the over-sampling parameter m. Note that this bound can be improved at the expense of additional matrix-vector products by using power iterations, $\mathbf{y}^i = A(A^T(A\mathbf{z}^i))$. Further, by building the basis Q incrementally it is also possible to estimate the error of the current approximation in order to decide if the number of random vectors is large enough to achieve a desired error tolerance.

7 Core Set Methods

In some cases, problems of the form (1) have a special sparse structure in terms of i that we can take advantage of. For example, consider the case of the squared hinge loss,

$$\min_{\mathbf{x}} \sum_{i=1}^{n} [\max\{0, 1 - b_i \mathbf{a}_i^T \mathbf{x}\}]^2.$$

For this problem, we typically expect the maximum to be zero for many i in a solution x^*. Thus, these instances i have no influence on the solution x^*, in the sense that if we remove these terms from the sum the solution does not change. This type of problem structure suggests a certain algorithmic approach: we estimate a set of data instances i that will determine the final solution (the *core set*), and we solve the problem exactly with respect to the core set. If the solution has a small core set, then this strategy may give substantial computational gains.

Core set methods have seen a recent rise in interest in the literature on the smooth support vector machine optimization problem, the problem of minimizing the ℓ_2-regularized squared hinge loss. While the SMO-style algorithms mentioned in 4.4 can be viewed as core set methods, modern core set methods differ from these older methods in that they seek exact solutions on the core set at each iteration, and they also seek to prove approximation guarantees given *approximate* core sets.

The first method to bear the name 'core set' for support vector optimization was described by Tsang et al. [2006]. In this work, a variant on the support vector machine optimization problem is shown to be equivalent to the minimum enclosing ball (MEB) problem in computational geometry (the problem of finding the smallest sphere that encloses a given set of points). Subsequently, a $(1+\epsilon)$ approximation algorithm for the MEB problem is used to find an approximate solution to the problem. The algorithm is very simple; at each iteration it adds the point to the core set that is furthest outside the ball based on the current core set, once no points are outside the $(1+\epsilon)$ ball the algorithm terminates. This method can be extremely fast if the size of the final approximate core set is small.

Subsequently, Har-Peled et al. [2007] proposed a core set method for support vector machines with approximation guarantees that does not involve using the equivalence with the MEB problem. This method uses a natural choice for the element to the add to the core set; it chooses the instance i where $b_i(\mathbf{a}_i^T \mathbf{x})$ is the smallest. Krishnan et al. [2008] consider choosing a random subset of the points to add to the core set, and argue that a set of $log(n)$ suffices with high probability in cases where only $log(n)$ of the $(1 - b_i(\mathbf{a}_i^T \mathbf{x}^*))$ are non-zero. Rai et al. [2009] considered a 'one-pass' approximation algorithm based on the equivalence with the MEB problem. This method goes through the data set once, and adds elements to the core set if they are outside the current ball. The method's low cost is appealing computationally, while it gives a 3/2 approximation.

8 Notes on these notes

Most of this informal document was written in the Fall of 2010, as a way for me to organize my thoughts around the enormous literature on solving 'big-N' problems. In 2011 and the start of 2012, I added Sections 6-7 and updated some of the other sections to include some more recent developements, including my own recent work [Friedlander and Schmidt, 2011, Schmidt et al., 2011, Le Roux et al., 2012, Schmidt and Le Roux, 2012]. Unfortunately, because of the scope of the literature I've attempted to survey there are inevitably some

innaccuracies in this document as well as important methods that I've probably missed, so please feel free to send me any feedback, corrections, or pointers to other interesting variations/algorithms that I've missed.

References

A. Agarwal, P. Bartlett, P. Ravikumar, and M. Wainwright. Information-theoretic lower bounds on the oracle complexity of convex optimization. *NIPS*, 1050:3, 2009.

A. Agarwal, O. Chapelle, M. Dudik, and J. Langford. A reliable effective terascale linear learning system. *Arxiv preprint arXiv:1110.4198*, 2011.

S. Amari. Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276, 1998.

H. Bauschke and P. Combettes. A Dykstra-like algorithm for two monotone operators. *Pacific Journal of Optimization*, 4(3):383–391, 2008.

H. Bauschke and A. Lewis. Dykstras algorithm with bregman projections: A convergence proof. *Optimization*, 48(4):409–427, 2000.

A. Beck and M. Teboulle. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31(3):167–175, 2003.

D. Bertsekas. A new class of incremental gradient methods for least squares problems. *SIAM Journal on Optimization*, 7(4):913–926, 1997.

D. Bertsekas. Incremental Gradient, Subgradient, and Proximal Methods for Convex Optimization: A Survey. 2010.

D. Bertsekas and J. Tsitsiklis. *Parallel and distributed computation: numerical methods*. Prentice Hall, 1989.

D. Bertsekas and J. Tsitsiklis. *Neuro-dynamic programming*. Athena Scientific, 1996.

D. Bertsekas, A. Nedic, and A. Ozdaglar. *Convex analysis and optimization*. Athena Scientific, 2003.

D. Blatt, A. Hero, and H. Gauchman. A convergent incremental gradient method with a constant step size. *SIAM Journal on Optimization*, 18(1):29–51, 2008.

S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Machine Learning*, 3(1):1–122, 2010.

L. Bregman. The method of successive projection for finding a common point of convex sets. *Doklady Akademii Nauk*, 162(3):487–490, 1965. An English translation appears in Soviet Mathematics Doklady, 6:688–692, 1965.

Y. Censor. Row-action methods for huge and sparse systems and their applications. *SIAM review*, 23(4): 444–466, 1981.

N. Cesa-Bianchi, P. Long, and M. Warmuth. Worst-case quadratic loss bounds for on-line prediction of linear functions by gradient descent. In *IEEE Transactions on Neural Networks*, 1993.

H. Chen, L. Guo, and A. Gao. Convergence and robustness of the Robbins-Monro algorithm truncated at randomly varying bounds. *Stochastic Processes and their Applications*, 27:217–231, 1987.

T. Choi and C. Kelley. Superlinear convergence and implicit filtering. *SIAM Journal on Optimization*, 10(4): 1149–1162, 2000.

M. Collins, A. Globerson, T. Koo, X. Carreras, and P. Bartlett. Exponentiated gradient algorithms for conditional random fields and max-margin markov networks. *The Journal of Machine Learning Research*, 9:1775–1822, 2008.

A. d'Aspremont. Smooth optimization with approximate gradient. *Arxiv preprint math/0512344*, 2005.

B. Delyon and A. Juditsky. Accelerated stochastic approximation. *SIAM Journal on Optimization*, 3:868, 1993.

F. Deutsch and H. Hundal. The rate of convergence of Dykstra's cyclic projections algorithm: The polyhedral case. *Numerical Functional Analysis and Optimization*, 15(5-6):537–565, 1994.

F. Deutsch and H. Hundal. The Rate of Convergence for the Method of Alternating Projections, II* 1. *Journal of Mathematical Analysis and Applications*, 205(2):381–405, 1997.

R. Dykstra. An algorithm for restricted least squares regression. *Journal of the American Statistical Association*, 78(384):837–842, 1983.

Y. Eldar and D. Needell. Acceleration of randomized kaczmarz method via the johnson–lindenstrauss lemma. *Numerical Algorithms*, pages 1–15, 2010.

A. Flaxman, A. Kalai, and H. McMahan. Online convex optimization in the bandit setting: gradient descent without a gradient. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 385–394. Society for Industrial and Applied Mathematics, 2005.

M. P. Friedlander and M. Schmidt. Hybrid deterministic-stochastic methods for data fitting. April 2011. revised September 2011.

N. Gaffke and R. Mathar. A cyclic projection algorithm via duality. *Metrika*, 36(1):29–54, 1989.

A. Galantai. On the rate of convergence of the alternating projection method in finite dimensional spaces. *Journal of Mathematical Analysis and Applications*, 310(1):30–44, 2005.

S. Ghadimi and G. Lan. Optimal stochastic approximation algorithms for strongly convex stochastic composite optimization. *Optimization Online*, 2010.

P. Gilmore and C. Kelley. An implicit filtering algorithm for optimization of functions with many local minima. *SIAM Journal on Optimization*, 5(2):269–285, 1995.

N. Halko, P. Martinsson, and J. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *Arxiv preprint arXiv:0909.4061*, 2009.

S. Har-Peled, D. Roth, and D. Zimak. Maximum margin coresets for active and noise tolerant learning. In *Proceedings of the 20th international joint conference on Artifical intelligence*, pages 836–841. Morgan Kaufmann Publishers Inc., 2007.

E. Hazan, A. Agarwal, and S. Kale. Logarithmic regret algorithms for online convex optimization. *Machine Learning*, 69(2):169–192, 2007.

A. Iusem and B. Svaiter. A row-action method for convex programming. *Mathematical Programming*, 64(1): 149–171, 1994.

A. Iusem and B. Svaiter. Primal-dual row-action method for convex programming. *Journal of Optimization Theory and Applications*, 86(1):73–112, 1995.

A. Kalai and S. Vempala. Geometric algorithms for online optimization. In *Journal of Computer and System Sciences*, 2002.

S. Keerthi, K. Duan, S. Shevade, and A. Poo. A fast dual algorithm for kernel logistic regression. *Machine Learning*, 61(1):151–165, 2005.

H. Kesten. Accelerated stochastic approximation. *The Annals of Mathematical Statistics*, pages 41–59, 1958.

J. Kiefer and J. Wolfowitz. Stochastic estimation of the maximum of a regression function. *The Annals of Mathematical Statistics*, 23(3):462–466, 1952.

J. Kivinen and M. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132(1):1–63, 1997.

A. Kleywegt, A. Shapiro, and T. Homem-de Mello. The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization*, 12(2):479–502, 2002.

S. Krishnan, C. Bhattacharyya, and R. Hariharan. A randomized algorithm for large scale support vector learning. *Proceedings of Advances in Neural Information Processing Systems, Vancouver*, pages 793–800, 2008.

H. Kushner and G. Yin. *Stochastic approximation algorithms and applications*. Springer-Verlag, second edition, 2003.

N. Le Roux, P. Manzagol, and Y. Bengio. Topmoumoute online natural gradient algorithm. In *Neural Information Processing Systems (NIPS)*. Citeseer, 2007.

N. Le Roux, M. Schmidt, and F. Bach. A stochastic gradient method with an exponential convergence rate for strongly-convex optimization with finite training sets. *Arxiv preprint arXiv:1202.6258*, 2012.

A. Nedic and D. Bertsekas. Convergence rate of incremental subgradient algorithms. *Stochastic Optimization: Algorithms and Applications*, pages 263–304, 2000.

A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on Optimization*, 19(4):1574–1609, 2009.

Y. Nesterov. A method for solving a convex programming problem with rate of convergence $O(1/k^2)$. *Soviet Math. Doklady*, 269(3):543–547, 1983.

Y. Nesterov. Primal-dual subgradient methods for convex problems. *Mathematical programming*, 120(1): 221–259, 2009.

Y. Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *Core discussion papers*, (0022010), 2010.

J. Platt. Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel Methods*, pages 185–208. MIT press, 1999.

B. Polyak and A. Juditsky. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30:838, 1992.

P. Rai, H. Daumé III, and S. Venkatasubramanian. Streamed learning: one-pass svms. In *Proceedings of the 21st international jont conference on Artifical intelligence*, pages 1211–1216. Morgan Kaufmann Publishers Inc., 2009.

H. Robbins and S. Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, pages 400–407, 1951.

M. Schmidt and N. Le Roux. Fast convergence of stochastic gradient descent under a strong growth condition. 2012.

M. Schmidt, N. Le Roux, and F. Bach. Convergence rates of inexact proximal-gradient methods for convex optimization. *Neural Information Processing Systems*, 2011.

N. Schraudolph, J. Yu, and S. Günter. A stochastic quasi-Newton method for online convex optimization. 2007.

S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal estimated sub-gradient solver for SVM. In *Proceedings of the 24th international conference on Machine learning*, page 814, 2007.

O. Shamir. Making gradient descent optimal for strongly convex stochastic optimization. *Arxiv preprint arXiv:1109.5647*, 2011.

A. Shapiro and A. Nemirovski. On complexity of stochastic programming problems. *Continuous optimization*, pages 111–146, 2005.

M. Solodov. Incremental gradient algorithms with stepsizes bounded away from zero. *Computational Optimization and Applications*, 11(1):23–35, 1998.

J. Spall. *Introduction to stochastic search and optimization: estimation, simulation, and control*. John Wiley and Sons, 2003.

T. Strohmer and R. Vershynin. A randomized Kaczmarz algorithm with exponential convergence. *Journal of Fourier Analysis and Applications*, 15(2):262–278, 2009.

P. Sunehag, J. Trumpf, A. Canberra, and S. Vishwanathan. Variable Metric Stochastic Approximation Theory. *AISTATS*, 2009.

I. Tsang, J. Kwok, and P. Cheung. Core vector machines: Fast svm training on very large data sets. *Journal of Machine Learning Research*, 6(1):363, 2006.

P. Tseng and S. Yun. A coordinate gradient descent method for linearly constrained smooth optimization and support vector machines training. *Computational Optimization and Applications*, pages 1–28, 2010.

J. von Neumann. *Functional Operators, vol. II, The Geometry of Orthogonal Spaces*, volume 22 of *Annals of Mathematical Studies*. Princeton University Press, 1950. This is a reprint of notes first distributed in 1933-34.

L. Xiao. Dual averaging methods for regularized stochastic learning and online optimization. *Journal of Machine Learning Research*, 11:2543–2596, 2010.

J. Yuan and L. Matoli. Asymptotically optimal row-action methods for generalized least squares problems. *International Journal of Computer Mathematics*, 70(1):1–18, 1998.

M. Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. 2003.