Numerical Optimization for Machine Learning

Proximal-Gradient, Fenchel Duality, and ADMM

Mark Schmidt

University of British Columbia

Summer 2022 - Summer 2023

Last Time: Subgradient-Based Methods

- We discussed subgradients. Given a w these are vectors d satisfying

$$f(v) \geq f(w) + d^T(v - w), \quad \text{for all } v.$$

- Sub-differential $\partial f(w)$ is set of subgradients at w.
    - If differentiable at w, only contains gradient.
    - Non-empty for non-degenerate points of convex functions.
- Subgradient method uses subgradient within gradient descent.
    - Requires step size to go to zero as in SGD (though can use Polyak step size).
    - Optimal dimension-independent rates for convex and strongly-convex Lipshitz f.
    - Same rates are achieved for projected and stochastic subgradient methods.
- We discussed various ways to go faster than subgradient methods:
    - Ignore non-smoothness (particularly if smooth at solution).
    - Use a smooth approximation (if not worried about non-smooth structure at solution).
    - Cutting plane methods have faster dimension-dependent rates (but high cost).
    - Bundle methods use multiple subgradients to better approximate function.
    - Minimum-norm subgradient methods choose steepest descent subgradient.

Faster Non-Smooth Optimization by Exploiting Structure

- Last time we saw that non-smooth methods are slower than smooth methods.
  - For strongly-convex functions we need $O(1/\epsilon)$ iterations instead of $O(\log(1/\epsilon))$.
- But we typically do not optimize generic black-box non-smooth functions.
  - For example, we might only be non-smooth because of an L1-regularizer,

$$F(w) = \underbrace{\frac{1}{2}\|Xw - y\|^2}_{\text{smooth}} + \underbrace{\lambda\|w\|_1}_{\text{``simple''}}.$$

- Proximal-gradient methods apply to functions of the form

$$F(w) = \underbrace{f(w)}_{\text{smooth}} + \underbrace{r(w)}_{\text{``simple''}},$$

  and have convergence rates of gradient descent for such problems.
  - Even though the "simple" term may be non-smooth.

From Gradient Descent to Proximal Gradient

- We want to minimize a smooth function f :

$$\underset{w \in \mathbb{R}^d}{\text{argmin}}\, f(w).$$

- Iteration w^k works with a quadratic approximation to f:

$$f(v) \approx f(w^k) + \nabla f(w^k)^\top (v - w^k) + \frac{1}{2\alpha_k}\|v - w^k\|^2,$$

$$w^{k+1} \in \underset{v \in \mathbb{R}^d}{\text{argmin}} \left\{ f(w^k) + \nabla f(w^k)^\top (v - w^k) + \frac{1}{2\alpha_k}\|v - w^k\|^2 \right\}.$$

We can equivalently write this as the quadratic optimization:

$$w^{k+1} \in \underset{v \in \mathbb{R}^d}{\text{argmin}} \left\{ \frac{1}{2}\|v - (w^k - \alpha_k \nabla f(w^k))\|^2 \right\},$$

and the solution is the gradient algorithm:

$$w^{k+1} = w^k - \alpha_k \nabla f(w^k).$$

From Gradient Descent to Proximal Gradient

- We want to minimize a smooth function f plus a non-smooth convex function r:

$$\underset{w \in \mathbb{R}^d}{\operatorname{argmin}} f(w) + r(w).$$

- Iteration w^k works with a quadratic approximation to f:

$$f(v) + r(v) \approx f(w^k) + \nabla f(w^k)^\top (v - w^k) + \frac{1}{2\alpha_k}\|v - w^k\|^2 + r(v),$$

$$w^{k+1} \in \underset{v \in \mathbb{R}^d}{\operatorname{argmin}} \left\{ f(w^k) + \nabla f(w^k)^\top (v - w^k) + \frac{1}{2\alpha_k}\|v - w^k\|^2 + r(v) \right\}.$$

We can equivalently write this as the proximal optimization:

$$w^{k+1} \in \underset{v \in \mathbb{R}^d}{\operatorname{argmin}} \left\{ \frac{1}{2}\|v - (w^k - \alpha_k \nabla f(w^k))\|^2 + \alpha_k r(v) \right\},$$

and the solution is the proximal-gradient algorithm:

$$w^{k+1} = \operatorname{prox}_{\alpha_k r}[w^k - \alpha_k \nabla f(w^k)].$$

Proximal-Gradient Method

- The proximal-gradient algorithm:

$$w^{k+\frac{1}{2}} = w^k - \alpha_k \nabla f(w^k), \quad w^{k+1} = \operatorname*{argmin}_{v \in \mathbb{R}^d} \left\{ \frac{1}{2}\|v - w^{k+\frac{1}{2}}\|^2 + \alpha_k r(v) \right\}.$$

- Right side is called proximal operator with respect to a convex function $\alpha_k r$.
  - We say that r is "simple" if you can efficiently compute proximal operator.

- Very similar properties to projected-gradient when ∇f is Lipschitz-continuous:
  - Guaranteed improvement for $\alpha_k < 2/L$, practical backtracking methods work better.
  - Solution if a fixed point, $w^* = \text{prox}_r(w^* - \alpha \nabla f(w^*))$ for any $\alpha > 0$.
  - If f is strongly-convex then using $\alpha_k = 1/L$ gives

$$F(w^k) - F^* \leq \left(1 - \frac{\mu}{L}\right)^k \left[F(w^0) - F^*\right],$$

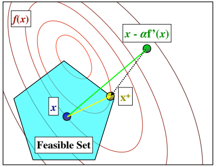
where $F(w) = f(w) + r(w)$ (while for convex f we get a $O(1/k)$ rate).

Projected-Gradient is Special case of Proximal-Gradient

- greProjected-gradient method is a special case of proximal-gradient:

$$r(w) = \begin{cases} 0 & \text{if } w \in \mathcal{C} \\ \infty & \text{if } w \notin \mathcal{C} \end{cases}, \quad (\text{indicator function for convex set } \mathcal{C})$$

$$w^{k+1} \in \underbrace{\operatorname*{argmin}_{v \in \mathbb{R}^d} \frac{1}{2}\|v - w^{k+\frac{1}{2}}\|^2 + \alpha_k r(v)}_{\text{proximal operator}} \equiv \underbrace{\operatorname*{argmin}_{v \in \mathcal{C}} \|v - w^{k+\frac{1}{2}}\|}_{\text{projection}}.$$



- Similar to projection, proximal operator is non-expansive:
$\|\text{prox}_r(w) - \text{prox}_r(v)\| \leq \|w - v\|$.

Proximal-Gradient for L1-Regularization

- The proximal operator for L1-regularization when using step-size α_k,

$$\text{prox}_{\alpha_k \lambda \|\cdot\|_1}[w] \in \underset{v \in \mathbb{R}^d}{\text{argmin}} \left\{ \frac{1}{2}\|v - w\|^2 + \alpha_k \lambda \|v\|_1 \right\},$$

involves solving a simple (strongly-convex) 1D problem for each variable j:

$$w_j \in \underset{v_j \in \mathbb{R}}{\text{argmin}} \left\{ \frac{1}{2}(v_j - w_j)^2 + \alpha_k \lambda |v_j| \right\}.$$

- We can find the argmin by finding the unique v_j with 0 in the sub-differential.
- The solution is given by applying the "soft-threshold" operation:
  1. If $|w_j| \leq \alpha_k \lambda$, set $w_j = 0$ ("threshold" small values of w_j).
  2. Otherwise, shrink $|w_j|$ by $\alpha_k \lambda$ (variables not thresholded move towards 0).
- So proximal-gradient takes gradient step then "shrinks" the w_j towards 0 by $\alpha_k \lambda$.
  - Unlike subgradient method, this yields iterations that are sparse (have exact zeros).

Active-Set Identification

- For L1-regularization, proximal-gradient "identifies" active set in finite time:

  (under mild assumptions)

  - For all sufficiently large k, sparsity pattern of x^k matches sparsity pattern of x^*.

$$w^0 = \begin{pmatrix} w_1^0 \\ w_2^0 \\ w_3^0 \\ w_4^0 \\ w_5^0 \end{pmatrix} \xrightarrow{\text{after finite } k \text{ iterations}} w^k = \begin{pmatrix} w_1^k \\ 0 \\ 0 \\ w_4^k \\ 0 \end{pmatrix}, \quad \text{where} \quad w^* = \begin{pmatrix} w_1^* \\ 0 \\ 0 \\ w_4^* \\ 0 \end{pmatrix}$$

- Proof under constant step-size similar to what we showed for projected-gradient.
  - Differences discussed in bonus (uses "distance to subdifferential boundary").
  - Can bound number of iterations before this happens ("active set complexity").
  - Can also be shown for backtracking along the "proximal arc".

Proximal-Gradient Linear Convergence Rate

- Simplest linear convergence proofs are based on the proximal-PL inequality,

$$\frac{1}{2}\mathcal{D}_r(w, L) \geq \mu(F(w) - F^*),$$

where $\|\nabla f(w)\|^2$ in the PL inequality is generalized to this mess:

$$\mathcal{D}_r(w, L) = -2\alpha \min_v \left[\nabla f(w)^\top (v - w) + \frac{L}{2}\|v - w\|^2 + r(v) - r(w)\right],$$

and recall that $F(w) = f(w) + r(w)$.

  - Other assumptions include KL inequality and error bounds (bonus).

- This non-intuitive property holds for some important problems:
  - Any time f is strong-convex (could add an L2-regularizer as part of f).
  - Any $f = g(Aw)$ for strongly-convex g and r being indicator for polyhedral set.
  - L1-regularized least squares.
- But it can be painful to show that functions satisfy this property.

Proximal-Gradient Convergence under Proximal-PL

- Linear convergence if ∇f is Lipschitz and F is proximal-PL:

$$\begin{aligned}
F(w_{k+1}) &= f(w^{k+1}) + r(w^{k+1}) \\
&\leq \underbrace{f(w_k) + \langle \nabla f(w_k), w_{k+1} - w_k \rangle + \frac{L}{2}||w_{k+1} - w_k||^2}_{\text{descent lemma on } f} + r(w_{k+1}) \\
&= F(w_k) + \underbrace{\langle \nabla f(w_k), w_{k+1} - w_k \rangle + \frac{L}{2}||w_{k+1} - w_k||^2 + r(w_{k+1}) - r(w_k)}_{\text{minimized by proximal-gradient, so equal to } -(1/2L) \text{ times } \mathcal{D}_r(w_k, L)} \\
&\leq F(w_k) - \frac{1}{2L}\mathcal{D}_r(w_k, L) \\
&\leq F(w_k) - \frac{\mu}{L}[F(w_k) - F^*] \quad \text{from proximal-PL,}
\end{aligned}$$

and then we can take our usual steps to show linear rate.

Outline

Application: Group L1-Regularization

- Proximal-gradient methods are often used for group L1-regularization.
  - We want sparsity in terms pre-defined groups, like sparse rows of parameter matrix,

$$W = \begin{bmatrix} -0.77 & 0.04 & -0.03 & -0.09 \\ 0 & 0 & 0 & 0 \\ 0.04 & -0.08 & 0.01 & -0.06 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

  - Group L1-regularization generalizes L1-regularization to this setting,

$$F(w) = f(w) + \lambda \sum_{g \in \mathcal{G}} \|w_g\|_2,$$

  - Applications include:
    - Variable selection using "1 of k" encodings.
    - Feature selection in multi-label or multi-class problems.
    - Graphical model structure learning.

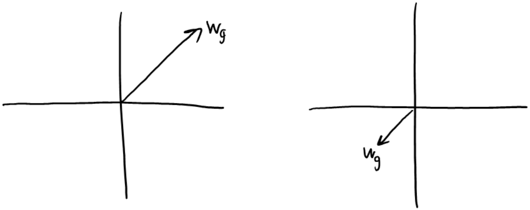
Proximal-Gradient for Group L1-Regularization

- The proximal operator for group L1-regularization,

$$\underset{v\in\mathbb{R}^d}{\text{argmin}}\left\{\frac{1}{2}\|v-w\|^2+\alpha_k\lambda\sum_{g\in G}\|v\|_2\right\},$$

applies a soft-threshold group-wise,

$$w_g \leftarrow \frac{w_g}{\|w_g\|_2}\max\{0,\|w_g\|_2-\alpha_k\lambda\}.$$



- So we can solve group L1-regularization problems as fast as smooth problems.

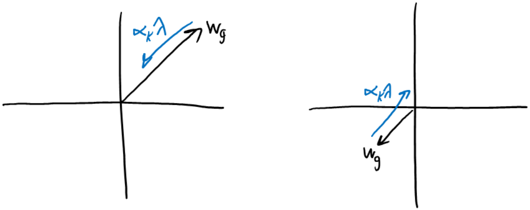
Proximal-Gradient for Group L1-Regularization

- The proximal operator for group L1-regularization,

$$\operatorname*{argmin}_{v \in \mathbb{R}^d} \left\{ \frac{1}{2}\|v - w\|^2 + \alpha_k \lambda \sum_{g \in G} \|v\|_2 \right\},$$

applies a soft-threshold group-wise,

$$w_g \leftarrow \frac{w_g}{\|w_g\|_2} \max\{0, \|w_g\|_2 - \alpha_k \lambda\}.$$



- So we can solve group L1-regularization problems as fast as smooth problems.

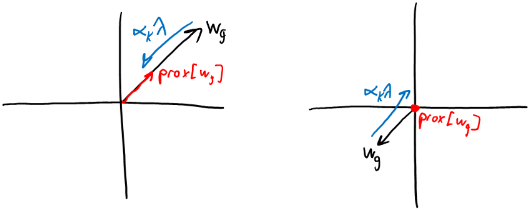
Proximal-Gradient for Group L1-Regularization

- The proximal operator for group L1-regularization,

$$\operatorname*{argmin}_{v\in\mathbb{R}^d}\left\{\frac{1}{2}\|v-w\|^2+\alpha_k\lambda\sum_{g\in G}\|v\|_2\right\},$$

applies a soft-threshold group-wise,

$$w_g \leftarrow \frac{w_g}{\|w_g\|_2}\max\{0,\|w_g\|_2-\alpha_k\lambda\}.$$



- So we can solve group L1-regularization problems as fast as smooth problems.

Structured Regularization

- There are many other patterns that regularization can encourage.
  - Total-variation regularization encourages slow/sparse changes in w.
  - Nuclear-norm regularization encourages sparsity in rank of matrices.
  - Structured sparsity encourages sparsity in variable patterns.
- Details on group L1 and strutured regularization added as note on webpage.
- Can efficiently approximate proximal operator for these problems.
- Inexact proximal-gradient methods:
  - Proximal-gradient methods with an approximation to the proximal operator.
  - If approximation error decreases fast enough, same convergence rate:
    - To get $O(\rho^t)$ rate, error must be in $o(\rho^t)$.
- A related approach is the "proximal average" for sum of "simple":
  - Replace proximal operator of sum with average of proximal operators for each term.

Alternating Direction Method of Multipliers

- ADMM is also popular for structured sparsity problems
- Alternating direction method of multipliers (ADMM) solves:

$$\min_{Aw+Bv=c} f(w) + r(v).$$

- Alternates between proximal operators with respect to f and r.
  - We usually introduce new variables and constraints to convert to this form.
- We can apply ADMM to L1-regularization with an easy prox for f using

$$\min_{w} \frac{1}{2}\|Xw - y\|^2 + \lambda\|w\|_1 \quad \Leftrightarrow \quad \min_{v=Xw} \frac{1}{2}\|v - y\|^2 + \lambda\|w\|_1,$$

- For total-variation and structured sparsity we can use

$$\min_{w} f(w) + \|Aw\|_1 \quad \Leftrightarrow \quad \min_{v=Aw} f(w) + \|v\|_1.$$

- If prox can not be computed exactly: linearized ADMM.
  - But ADMM rate depends on tuning parameter(s) and iterations are not sparse.

Coordinate-Wise and Stochastic Proximal-Gradient

- We can apply coordinate-wise proximal-gradient when g is separable,

$$F(w) = f(w) + \sum_{j=1}^{n} g_j(w_j),$$

  which includes L1-regularization methods (this is a popular/simple approach).
  - Same convergence rate as smooth randomized coordinate descent.

- We can add proximal operator to SGD,

$$w^{k+1} = \text{prox}_{\alpha_k r}[w^k - \alpha_k \nabla f(w^k)],$$

  although this is not obviously better than the subgradient method.
  - We learned earlier that SGD does not converge faster for smooth problems.
  - This method loses the active set identification property.
    - Method like regularized dual averaging that use average gradient restore this.
- Adding prximal operator for variance-reduced SGD:
  - Leads to rates of smooth setting and active set identification.

Proximal-Newton

- We can define accelerated proximal-gradient in a straightforward way.
  - Replace projection with proximal operator in accelerated projected gradient.
- We can define proximal-Newton methods using

$$w^{k+\frac{1}{2}} = w^k - \alpha_k [H_k]^{-1} \nabla f(w^k) \qquad \text{(Newton step)}$$

$$w^{k+1} = \underset{v \in \mathbb{R}^d}{\operatorname{argmin}} \left\{ \frac{1}{2} \| v - w^{k+\frac{1}{2}} \|_{H_k}^2 + \alpha_k r(v) \right\} \qquad \text{(proximal step)}$$

- Local superlinear convergence rate if f is locally nice at w^*.
- This proximal operator is expensive even for simple r like L1-regularization.
- But there are analogous tricks to projected-Newton methods:
  - Diagonal or Barzilai-Borwein or diagonal plus rank-1 Hessian approximation.
  - Inexact methods use approximate proximal operator.
    - Most useful when computing f is much more expensive than proximal operator.

Proximal Point Algorithm

- A related method is the proximal point method for minimizing a function f,

$$w^{k+1} = \underset{v \in \mathbb{R}^d}{\text{argmin}} \left\{ f(w) + \frac{1}{2\alpha_k}\|w - w^k\|^2 \right\},$$

  where we compute proximal operator with respect to f (may be non-smooth).

- Obvious issue:
  - Computing the iteration might be as hard as solving original problem.
- However, in some settings it might be easier:
  - If f is convex, then proximal operator is strongly-convex.
  - If f is non-convex, proximal operator might be convex.
- Example usage:
  - Catalyst uses SAG/SVRG within inexact accelerated proximal point.
    - Achieves an accelerated convergence rate.

Outline

Motivation for Conjugate Functions and Duality

- We will next cover conjugate functions and duality.
- For many people, including myself, these are not particularly fun topics!
  - I failed Michael Friedlander's midterm due to jetlag and duality questions.
- To give us some motivation, here are some things you can do with duality:
  - Construct smooth approximations to non-smooth convex functions.
  - Write smooth re-formulations to non-smooth strongly-convex problems.
  - Make faster predictions with non-parametric features for some models.
    - Support vectors.
  - Use an update similar to stochastic subgradient with an optimal step size.
    - Based on progress in the dual objective.
  - Guarantee that a given iterate is within ϵ of optimal value.
    - By using the duality gap.
  - Guarantee that a variable is zero in solution.
    - Safe screening for variable selection.

Example: SVM Primal vs. Dual Problem

- Consider the support vector machine optimization problem,

$$\operatorname{argmin}_{w \in \mathbb{R}^d} C \sum_{i=i}^{n} \max\{0, 1 - y_i w^T x_i\} + \frac{1}{2}\|w\|^2,$$

where C is the regularization parameter ($\lambda = 1/C$).
  - Non-smooth but strongly-convex, and proximal operator is not simple.
  - We could use stochastic subgradient, but:
    - It converges slowly, it is hard to set step size, and deciding when to stop is annoying.

- A Fenchel dual to the SVM problem is given by

$$\operatorname{argmax}_{z \in \mathbb{R}^n \mid 0 \leq z \leq C} \sum_{i=1}^{n} z_i - \frac{1}{2}\|X^\top Y z\|^2,$$

where X has vectors x_i^T as rows and Y is diagonal with the y_i along diagonal.
  - Written in terms of n variables z_i constrained to be in $[0, C]$.

Properties of SVM Dual

- For the $d \times n$ matrix $A = X^TY$, the SVM dual problem can be written:

$$\underset{0 \leq z \leq C}{\text{argmax}}\, z^T 1 - \frac{1}{2}\|Az\|^2.$$

- Relevant properties of this constrained quadratic optimization:
  - For any dual solution z^*, the primal solution is $w^* = Az^*$.
    - We can solve the dual problem to solve the primal problem.
  - The dual is Lipschitz-smooth (L is max eigenvalue of A^TA).
    - And since the constraints are simple we can apply projected gradient.
  - The dual satisfies proximal-PL (μ is min non-zero singular values of A^TA).
    - So projected-gradient has linear convergence rate.
  - Constraints are separable and dual is friendly to random coordinate optimization.
    - So projected randomized coordinate optimization gets linear convergence rate.
  - It is simple to derive optimal step size.
    - So we do not need backtracking.
  - In the usual case where A is dense, it is friendly to greedy coordinate descent.
    - So we can greedily pick the variable to update.

Duality Gap and Safe Termination

- To summarize advantages of solving dual problem:
  - Returns same solution, but can use faster algorithm with optimal step size.
- LIBSVM is a greedy dual coordinate optimization method for fitting SVMs.
  - Probably the most-used coordinate descent method in history.
- Tracking primal vector $w^k = Az^k$ can also help decide when to stop:
  - We can show that $D(z^k) \leq f^*$ (weak duality).
    - So if $\underbrace{f(w^k) - D(z^k)}_{\text{duality gap}} \leq \epsilon$, we are guaranteed to have $f(w^k) - f^* \leq \epsilon$.
  - Further, we have $f(w^*) = D(z^*)$ (strong duality) so duality gap does converge to 0.

Support Vectors and Safe Screening

- Due to the lower bounds on dual variables z_i, solution will tend to be sparse.
  - Many z_i will be zero.
  - The non-zero values are called support vectors.
  - We know projected gradients and variants eventually identify the support vectors.
    - Many implementations try to identify support vectors to reduce cost ("shrinking").
    - This also speeds up prediction when using the kernel trick with SVMs (see bonus).

- Duality gap can be used to give a safe screening rule for removing z_i.
  - For example, if it holds for any z that

$$\nabla_i D(z) < -\sqrt{\langle a_i, a_i \rangle (f(Az) - D(z))},$$

  then we are guaranteed to have $z_i^* = 0$ in the solution (a_i is row i of A).
    - Gradient is too large compared to sub-optimality for 0 to not be the solution.
    - At this point, you can permanently remove the variable from the problem.

Outline

Digression: Supremum and Infimum

- Infimum $(\inf)$ is a generalization of $\min$ that includes limits:

$$\min_{x\in\mathbb{R}} x^2 = 0, \quad \inf_{x\in\mathbb{R}} x^2 = 0,$$

  but

$$\min_{x\in\mathbb{R}} e^x = \text{DNE}, \quad \inf_{x\in\mathbb{R}} e^x = 0.$$

- Formally, the infimum of a function f is its largest lower-bound,

$$\inf f(x) = \max_{y \,|\, y\leq f(x)} y.$$

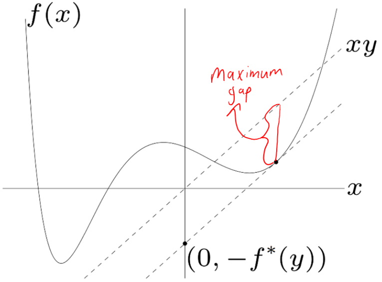
- The analogous function for $\max$ is called the supremum $(\sup)$.
  - Supremum is the smallest upper-bound on the function.

Convex Conjugate

- The convex conjugate f^* of a function f is given by

$$f^*(y) = \sup_{x \in \mathcal{X}} \{y^\top x - f(x)\},$$

where $\mathcal{X}$ is values where $\sup$ is finite.



http://www.seas.ucla.edu/~vandenbe/236C/lectures/conj.pdf

- It's the maximum that the linear function $y^\top x$ can get above $f(x)$.

Convex Conjugate Examples

- If $f(x) = \frac{1}{2}\|x\|^2$ we have
  - $f^*(y) = \sup_x\{y^\top x - \frac{1}{2}\|x\|^2\}$ or equivalently (by taking derivative and setting to 0):

$$0 = y - x,$$

and pluggin in $x = y$ we get

$$f^*(y) = y^\top y - \frac{1}{2}\|y\|^2 = \frac{1}{2}\|y\|^2.$$

- If f is differentable, then $\sup$ occurs at x where $y = \nabla f(x)$.

- If $f(x) = a^\top x$ we have

$$f^*(y) = \sup_x\{y^\top x - a^\top x\} = \sup_x\{(y-a)^\top x\} = \begin{cases} 0 & y = a \\ \infty & \text{otherwise.} \end{cases}$$

Convex Conjugate Examples

- For norms, $f(x) = \|x\|$, convex conjuage is dual-norm unit ball,

$$f^*(y) = \begin{cases} 0 & \|x\|_* \leq 1 \\ \infty & \text{otherwise} \end{cases}.$$

- For logistic loss, $f(x) = \log(1 + \exp(x))$, conjugate is negative entropy,

$$f^*(y) = \begin{cases} y\log(y) + (1-y)\log(1-y) & y \in (0,1) \\ 0 & y = 0 \text{ or } y = 1 \end{cases}.$$

(the $\sup$ is unbounded when $y < 0$ or $y > 1$)

- For other examples, see Boyd & Vandenberghe's "Convex Optimization" book.

Properties of Convex Conjugate

- Properties of conve conjugate:
  - If f is differentable, then $\sup$ occurs at x where $y = \nabla f(x)$.
  - Conjugate f^* is convex, even if f is not (max over linear functions of y).
  - If f is convex and closed, then $f^{**} = f$.
  - Connection with Lipschitz-smoothness and strong-convexity:
    - If f is strongly-convex and closed, then f^* is Lipschitz smooth with $L = 1/\mu$.
    - If f is Lipschitz smooth, then f^* is strongly-convex with $\mu = 1/L$.

- The $f = f^{**}$ property gives us an alternative way to write a closed and convex f,

$$f(x) = \sup_{y \in \mathcal{Y}} \{y^T x - f^*(y)\},$$

in terms of of a "dual space" (which is space of gradients for differentiable f).
  - Get L-smooth approximation to non-smooth f by adding strongly-concave term,

$$f(x) \approx \sup_{y \in \mathcal{Y}} \{y^T x - f^*(y) - \frac{1}{2L}\|y\|^2\},$$

which (for example) gives Huber loss as approximation to L1-norm.

Fenchel Dual

- In machine learning our primal problem is usually (for convex f and r)

$$\underset{w\in\mathbb{R}^d}{\text{argmin}}\, P(w) = f(Xw) + r(w).$$

- If we introduce equality constraints,

$$\underset{v=Xw}{\text{argmin}}\, f(v) + r(w).$$

then Lagrangian dual has a special form called the Fenchel dual (see bonus).

$$\underset{z\in\mathbb{R}^n}{\text{argmax}}\, D(z) = -f^*(-z) - r^*(X^\top z),$$

where we're maximizing the (negative) convex conjugates f^* and r^*.

Fenchel Dual Properties

- Primal and dual functions:

$$P(w) = f(Xw) + r(w), D(z) = -f^*(-z) - r^*(X^\top z).$$

- Properties:
  - Number of dual variables is n instead of d.
    - Dual may be a lower-dimensional problem.
  - Weak duality is that $P(w) \geq D(z)$ for all w and z (assuming P is bounded below).
    - So any value of dual objective gives lower bound on $P(w^*)$.
  - Strong duality holds when $P(w^*) = D(z^*)$.
    - This requires an additional assumption.
    - Example: f and g convex, exists feasible w with $z = Xw$ where g continuous at z.
    - When true, can use duality gap $P(w) - D(z)$ to certify optimality of w and z.
  - Lipschitz-smoothness and strong-convexity relationship.
    - Dual is Lipschitz-smooth if primal is strongly-convex (as in SVMs).
  - Dual of loss f^* is separable if f is a finite-sum problem.
    - Allows us to use dual coordinate optimization for many problems.

Stochastic Dual Coordinate Ascent (SDCA)

- If we have an L2-regularized linear model (including SVM case discussed earlier),

$$\operatorname*{argmin}_{w\in\mathbb{R}^d} \sum_{i=1}^n f_i(w^\top x_i) + \frac{\lambda}{2}\|w\|^2,$$

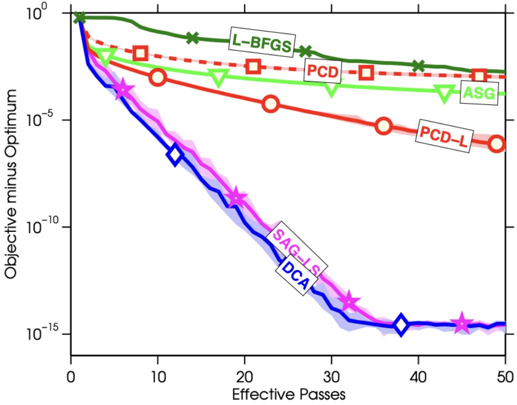
then Fenchel dual is a problem where we can apply coordinate optimization,

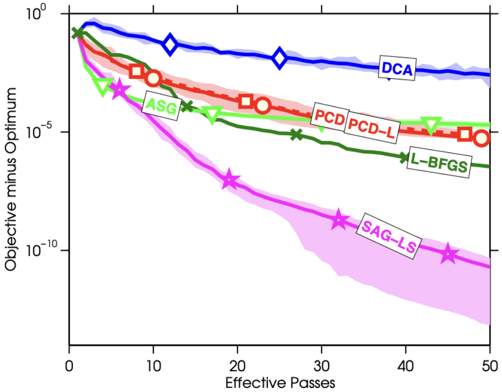
$$\operatorname*{argmax}_{z\in\mathbb{R}^n} \underbrace{-\sum_{i=1}^n f_i^*(z_i)}_{\text{separable}} - \frac{1}{2\lambda}\underbrace{\|X^\top z\|^2}_{z^\top X X^\top z}.$$

- Stochastic dual coordinate ascent (SDCA) applies dual coordinate optimization:
  - Only needs to looks at one training example on each iteration.
  - Obtains $O((L/\lambda)\log(1/\epsilon))$ rate if ∇f_i are L-Lipschitz.
    - And you can do a line-search to set the step size.
    - Performance similar to SAG for many problems, worse if $\mu >> \lambda$.
  - Obtains $O(1/\epsilon)$ rate for non-smooth f:
    - Same rate/cost as stochastic subgradient, but we can use exact/adaptive step-size.

SAG vs. SDCA (and Primal Coordinate Descent)

- We $\lambda = \mu$ on the left (RCV1) and $\lambda << \mu$ on the right (quantum).





- Bonus slides consider SDCA with a particular choice of step size:
  - SDCA is equivalent to stochastic subgradient with an adaptive step size.
  - Allows allowing SDCA based only on primal operations ("dual free").

Safe Screening Rules

- For many ML problems we want sparse solution (in primal or dual).
  - SVMs, L1-regularization, NMF, and so on.
- Safe screening rule is a rule that guarantees a variable is 0 in solution.
  - Original idea was to do the screening before you run any algorithm.
  - Later works incorporate screening as you go to continue removing variables.
- Key ingredients in safe screening rules:
  - Define a region that contains optimal solutions.
  - Bound possible function values when function is non-zero in region.
  - Screen variable if function is lower when variable is zero across region.
- An example is a gap safe spheres which use duality gap to imply variable must 0.
  - With size of spheres shrinking as duality gap shrinks.

Summary

- Proximal-gradient for sum of smooth and simple non-smooth.
  - Generalization of projected-gradient.
  - With L1-regularization as simple regularizer, performs soft-threshold.
  - Similar convergence properties to gradient descent.
  - Exist coordinate-wise, stochastic, accelerated, Newton-like, SVRG versions.
- Convex conjugates and Fenchel dual
  - Allow constructing smooth approximations and re-formulations.
  - Can lead to problems with fewer variables or more-favourable structure.
  - Allow certificates of optimality and variable pruning.
  - Fenchel dual for SVMs has the above benefits and more (like faster prediction).

- Next time: how do you optimize w^4?

Should we use projected-gradient for non-smooth problems?

- Some non-smooth problems can be written as smooth problems with simple constraints.
- But transforming might make problem harder:
  - For L1-regularization least squares,

$$\underset{w \in \mathbb{R}^d}{\text{argmin}} \frac{1}{2}\|Xw - y\|^2 + \lambda \|w\|_1,$$

we can re-write as a smooth problem with bound constraints,

$$\underset{w_+ \geq 0,\, w_- \geq 0}{\text{argmin}} \|X(w_+ - w_-) - y\|^2 + \lambda \sum_{j=1}^{d} (w_+ + w_-).$$

  - Doubles the number of variables.
  - Transformed problem is not strongly convex even if the original was.

Indicator Function for Convex Sets

- The indicator function for a convex set is

$$r(w) = \begin{cases} 0 & \text{if } w \in \mathcal{C} \\ \infty & \text{if } w \notin \mathcal{C} \end{cases}.$$

- This is a function with "extended-real-valued" output: $r : \mathbb{R}^d \to \{\mathbb{R}, \infty\}$.

- The convention for convexity of such functions:
  - The "domain" is defined as the w values where $r(w) \neq \infty$ (in this case $\mathcal{C}$).
  - We need this domain to be convex.
  - And the function should to be convex on this domain.

Example of Soft-Threshold

- An example is sof-threshold operator on absolute value with $\alpha_k \lambda = 1$:

Input	Threshold	Soft-Threshold
$\begin{bmatrix} 0.6715 \\ -1.2075 \\ 0.7172 \\ 1.6302 \\ 0.4889 \end{bmatrix}$	$\begin{bmatrix} 0 \\ -1.2075 \\ 0 \\ 1.6302 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ -0.2075 \\ 0 \\ 0.6302 \\ 0 \end{bmatrix}$

- Symbolically, the soft-threshold operation computes

$$w_j^{k+1} = \underbrace{\text{sign}(w^{k+\frac{1}{2}})}_{-1 \text{ or } +1} \max\left\{0, |w_j^{k+\frac{1}{2}}| - \alpha_k \lambda\right\},$$

Active-Set Complexity for Non-Smooth Regularizers

- Projected-gradient active set identification argument can be extended to

$$\underset{l \leq w \leq u}{\text{argmin}} \underbrace{f(w)}_{\text{smooth}} + \sum_{j=1}^{d} \underbrace{g_j(w_j)}_{\text{convex}},$$

  where "active set" is variables at a bound or non-smooth g_j value.
- Key differences:
  - The set $\mathcal{Z}$ will be variables occuring at bounds or non-smooth points.
    - For L1-regularization this is again the variables with $w_i^* = 0$.
  - The quantity δ will be the "minimum distance to the sub-differential boundary",

$$\delta = \min_{i \in \mathcal{Z}} \{\min\{-\nabla_i f(w^*) - \min\{\partial g_i(w_i^*)\}, \max\{\partial g_i(w_i^*)\} + \nabla_i f(x^*)\}\}.$$

    - For L1-regularization this is $\delta = \lambda - \max_{i \in \mathcal{Z}}\{|\nabla f_i(w^*)|\}$.
  - The non-degeneracy condition is that $\delta > 0$.
    - For L1-regularization we require $|\nabla_i f(w^*)| \neq \lambda$ for $i \in \mathcal{Z}$.
  - Proof needs to bound w_i^k from above and below based on $\partial g_i(w_i^*)$.
    - For other problems/algorithms, see "Wiggle Room Lemma".

Debugging a Proximal-Gradient Code

- In general, debugging optimization codes can be difficult.
  - The code can appear to work even if it's wrong.

- A reasonable strategy is to test things you expect to be true.
  - And keep a set of tests that should remain true if you update the code.
- For example, for proximal-gradient methods you could check:
  - Does it decrease the objective function for a small enough step-size?
  - Are the step-sizes sensible (are they much smaller than $1/L$)?
  - Is the optimality condition going to zero as you run the algorithm?
- For group L1-regularization, some other checks that you can do:
  - Set $\lambda = 0$ and see if you get the unconstrained solution.
  - Assign each variable to its own group and see if you get the L1-regularized solution.
  - Assign all variables to the same group and see if you get an L2-regularization solution (and 0 for large-enough λ).

Implicit subgradient viewpoint of proximal-gradient

- The proximal-gradient iteration is

$$w^{k+1} \in \operatorname*{argmin}_{v \in \mathbb{R}^d} \frac{1}{2}\|v - (w^k - \alpha_k \nabla f(w^k))\|^2 + \alpha_k r(v).$$

- By non-smooth optimality conditions that 0 is in subdifferential, we have that

$$0 \in (w^{k+1} - (w^k - \alpha_k \nabla f(w^k)) + \alpha_k \partial r(w^{k+1}),$$

  which we can re-write as

$$w^{k+1} = w^k - \alpha_k(\nabla f(w^k) + \partial r(w^{k+1})).$$

- So proximal-gradient is like doing a subgradient step, with
  1. Gradient of the smooth term at w^k.
  2. A particular subgradient of the non-smooth term at w^{k+1}.
     - "Implicit" subgradient.

Proximal-Gradient for L0-Regularization

- There are some results on proximal-gradient for non-convex r.
- Most common case is L0-regularization,

$$f(w) + \lambda \|w\|_0,$$

where $\|w\|_0$ is the number of non-zeroes.
  - Includes AIC and BIC from 340.
- The proximal operator for $\alpha_k \lambda \|w\|_0$ is simple:
  - Set $w_j = 0$ whenver $|w_j| \leq \alpha_k \lambda$ ("hard" thresholding).
- Analysis is complicated a bit because discontinuity of prox as function of α_k.
  - If step size is too small then you may not be able to move.

Faster Rate for Proximal-Gradient

- By analyze $\|w^k - w^*\|$ and using non-expansive, we can show a slightly faster rate for proximal-gradient using $\alpha_k = 2/(\mu + L)$:
- `http://www.cs.ubc.ca/~schmidtm/Documents/2014_Notes_ProximalGradient.pdf`

Equivalent Conditions to Proximal-PL

- When ∇f is L-Lipschitz continuous, the following 3 conditions are equivalent:
  1. Proximal-PL for some $\mu > 0$:
  $$\frac{1}{2}\mathcal{D}_r(w, L) \geq \mu(F(w) - F^*),$$
  2. Error bounds for some $c > 0$:
  $$\|w - w_p\| \leq c\left\|w - \text{prox}_{\frac{1}{L}r}\left(w - \frac{1}{L}\nabla f(w)\right)\right\|,$$
  where w_p is the projection of x onto the set of solution.
  3. Kurdyka-Lojasiewicz for some $\mu > 0$:
  $$\min_{s \in \partial F(w)} \frac{1}{2}\|s\|^2 \geq \mu(F(w) - F^*),$$
  where $\partial F(w)$ is the "local" sub-differential.

(Same as usual sub-differential for convex)

Lagrangian Function for Equality Constraints

- Consider minimizing a differentiable f with linear equality constraints,

$$\underset{Ax=b}{\operatorname{argmin}}\, f(x).$$

- The Lagrangian of this problem is defined by

$$L(x,z) = f(x) + z^\top(Ax - b),$$

  for a vector $z \in \mathbb{R}^n$ (with A being n by d).
- At a solution of the problem we must have

$$\nabla_x L(x,z) = \nabla f(x) + A^\top z = 0 \qquad \text{(gradient is orthogonal to constraints)}$$
$$\nabla_z L(x,z) = Ax - b = 0 \qquad \text{(constraints are satisfied)}$$

- So solution is stationary point of Lagrangian.

Lagrange Dual Function

- But we can't just minimize with respect to x and z.
- The solution for convex f is actually a saddle point,

$$\max_z \min_x L(x, z).$$

(in cases where the $\max$ and $\min$ have solutions)

- One way to solve this is to eliminate x,

$$\max_z D(z),$$

where $D(z) = \min_x L(x, z)$ is called the dual function.

Dual function

- Even for non-smooth convex f solution is a saddle point of the Lagrangian,

$$\max_z \inf_x \underbrace{f(x) + z^\top (Ax - b)}_{L(x,z)}.$$

(restricted to z where the $\max$ is finite)

- We can eliminate x by working with the dual function,

$$\max_z D(z),$$

  with $D(z) = \inf_x \{f(x) + z^\top (Ax - b)\}$.
  - Note that D is concave for any f, so $-D$ is convex.
    - But we may not have strong duality.
  - Many constrained qualification guarantee that strong duality holds.
    - Example is Slater's condition for convex optimization problems: exists x that satisfies equality constraints and *strictly* satisfies inequalities (x is in "relative interior" of domain).

Fenchel Dual

- Lagrangian for constrained problem is

$$L(v, w, z) = f(v) + r(w) + z^\top (Xw - v),$$

so the dual function is

$$D(z) = \inf_{v,w} \{f(v) + r(w) + z^\top (Xw - v)\}$$

- For the inf wrt v we have

$$\inf_v \{f(v) - z^\top v\} = -\sup_v \{v^\top z - f(v)\} = -f^*(z).$$

- For the inf wrt w we have

$$\inf_w \{r(w) + z^\top Xw\} = -r^*(-X^\top z).$$

- This gives

$$D(z) = -f^*(z) - r^*(-X^\top z),$$

or get an alternate dual by replacing $(Xw - v)$ with $(v - Xw)$ in the Lagrangian.

Faster Predictions with Kernels

- Recall the kernel trick from 340:
  - Represent learning and prediction in terms of inner products $x_i^T x_j$.
  - Replace inner products in original feature space with kernel function $k(x_i, x_j)$.
    - Which can be an inner product in a high-dimensional feature space.
  - For details on kernels and the kernel trick, see notes on webpage.
- Writing the SVM primal problem using the kernel trick:

$$\underset{v \in \mathbb{R}^n}{\text{argmin}} \sum_{i=i}^{n} \max\{0, 1 - y_i \sum_{j=1}^{n} v_j k(x_i, x_j)\} + \frac{\lambda}{2} v^T K v,$$

  where matrix K has elements $K_{ij} = k(x_i, x_j)$ and we have n parameters v_j.
- We make predictions on a new example $\tilde{x}_i$ using

$$\hat{y}_i = \sum_{i=1}^{n} v_i k(x_i, \tilde{x}_i),$$

  which costs $O(nd)$ in typical case where the kernel function costs $O(d)$.

Faster Predictions with Kernels

- Writing our SVM dual problem using the kernel trick:

$$\underset{0 \leq z \leq 1}{\text{argmax}}\, z^T 1 - \frac{1}{2\lambda} z^T YKYz.$$

- Due to the lower bounds on the z_i, solution will tend to be sparse.
  - Many z_i will be zero. The non-zero values are called support vectors.
- We make predictions on a new example $\tilde{x}_i$ using

$$\hat{y}_i = \frac{1}{\lambda} \sum_{i=1}^{n} z_i y_i k(x_i, \tilde{x}_i).$$

- If we have m support vectors, this only costs $O(md)$.
  - We only need to use/store the support vectors to make predictions.
  - So predictions are faster when we predict with dual variables.

Stochastic Dual Coordinate Ascent vs. Stochastic Subgradient

- Consider a primal objective of the form

$$P(w) = \frac{1}{n}\sum_{i=1}^{n} f_i(w^T x_i) + \frac{\lambda}{2}\|w\|^2,$$

which includes many models such as SVMs as special cases.

- A Fenchel dual has the form

$$D(z) = \frac{1}{n}\sum_{i=1}^{n} -f_i^*(-z_i) + \frac{\lambda}{2}\|\frac{1}{\lambda n}\sum_{i=1}^{n} z_i x_i\|^2,$$

where can generate a primal variable from the duals using $w = \frac{1}{\lambda n}\sum_{i=1}^{n} z_i x_i$.

- Can show that stochastic dual coordinate ascent with special step size:
  - Corresponds to a stochastic subgradient method in terms of primal variables.
  - Does not actually need to know anything about f_i^* ("dual free").

Stochastic Dual Coordinate Ascent vs. Stochastic Subgradient

- The dual coordinate ascent update written in terms of primal and dual variables,

$$z_i^{k+1} = z_i^k + \delta_k, \quad w^{k+1} = w^k + \frac{1}{\lambda n}\delta_k x_i.$$

- Suppose that we choose a δ_k that can be written in the form

$$\delta_k = -\lambda n \alpha_k (f_i'(w^T x_i) + z_i),$$

for some "step size" α_k and f_i' in subdifferential of f evaluated at $w^T x_i$,

$$w^{k+1} = w^k - \alpha_k(f_i(w^T x_i) + z_i)x_i = w^k - \alpha_k(f_i(w^T x_i)x_i + z_i x_i).$$

- Choosing i uniformly (and assuming α_k does not depend on i) we have

$$\begin{aligned}\mathbb{E}[w^{k+1}] &= w^k - \alpha_k(\mathbb{E}[f_i(w^T x_i)] + \mathbb{E}[z_i x_i]) \\ &= w^k - \alpha_k(\frac{1}{n}\sum_{i=1}^n f_i(w^T x_i) + \lambda w) = w^k - \alpha_k g_k,\end{aligned}$$

where g_k is in subdifferential of primal P at w^k (we used $\lambda w = (1/n)\sum_{i=1}^n z_i x_i$).

Stochastic Dual Coordinate Ascent vs. Stochastic Subgradient

- So if $\delta_k = -\lambda n \alpha_k (f_i'(w^T x_i) + z_i)$ for some α_k and we choose i randomly, stochastic dual coordinate ascent is a special case of stochastic subgradient.
  - Except it can converge with a sufficiently-small step size.
    - For L-smooth f_i, converges linearly if $\alpha_k = \alpha \leq 1/(L + n\lambda)$.
    - Can show that variance of update goes to zero at solution, like SAG/SVRG.
    - For non-smooth case would need to pick a particular subgradient to cancel with z_i.
  - Allows us to implement stochastic dual coordinate ascent without using dual.
    - "Dual free" dual coordinate ascent, if you do not want to derive conjugate.

- But dual methods are not restricted to the above special case.
  - We can choose δ_k to maximize progress in dual.
    - Should make at least as much progress as stochastic subgradient.
  - We can greedily choose variable i to update
    - Which can perform much better than random choice in stochastic subgradient.