Matrix Algebra and Optimization for Statistics and Machine Learning

Yiyuan She

Department of Statistics, Florida State University

- Some classical optimization methods

Gradient descent

- Consider a general framework for solving $\min_\beta f(\beta)$:

$$\beta^{t+1} = \beta^t + \alpha_t p_t, \quad t = 0, 1 \ldots$$

- $\alpha_t > 0$ ($\alpha_t \in \mathbb{R}$) is called the stepsize
- p_t gives the search direction at iteration t
- What makes a good candidate search direction?
- Descent direction: $\langle \nabla f(\beta^t), p_t \rangle < 0$

Why descent directions?

- Assume f is convex and differentiable. Hence $\boldsymbol{\Delta}_f(\beta, \beta^-) \triangleq f(\beta) - f(\beta^-) - \langle \nabla f(\beta^-), \beta - \beta^- \rangle \geq 0$
- $f(\beta^{t+1}) < f(\beta^t) \Rightarrow 0 \leq \boldsymbol{\Delta}_f(\beta^{t+1}, \beta^t) < -\langle \nabla f(\beta^t), \alpha_t p_t \rangle$
- [In general, when f is not necessarily convex but is smooth, $f(\beta^{t+1}) - f(\beta^t) = (\alpha_t + o(\alpha_t))\langle \nabla f(\beta^t), p_t \rangle$.]
- Descent direction does not guarantee a descent! (Only have $f(\beta^{t+1}) - f(\beta^t) \geq \langle \nabla f(\beta^t), \beta^{t+1} - \beta^t \rangle < 0$)
- A natural choice $p_t = -\nabla f(\beta^t)$ gives gradient descent

- Indeed, when $f(\beta+\delta)$ is smooth,

$$
\begin{aligned}
& f(\beta+\delta)-f(\beta)-\langle\nabla f(\beta),\delta\rangle \\
={}& \int_0^1 \langle\nabla f(\beta+t\delta)-\nabla f(\beta),\delta\rangle\,\mathrm{d}t \\
={}& \int_0^1 \mathrm{d}t \int_0^t \mathrm{d}s\,\delta^T\nabla^2 f(\beta)\delta = o(\|\delta\|),
\end{aligned}
$$

  due to $f(\beta+\delta)-f(\beta)=f(x+t\delta)|_0^1=\int_0^1\langle\nabla f(\beta+t\delta),\delta\rangle\,\mathrm{d}t$
- Hence $f(\beta^{t+1})-f(\beta^t)=(\alpha_t+o(\alpha_t))\langle\nabla f(\beta^t),p_t\rangle$. Setting α_t small enough, we know $\langle\nabla f(\beta^t),p_t\rangle$ must be negative.
- [MM gives a better justification of the DD requirement.]

Example: generalized linear models

- Let's consider a canonical GLM with cumulant $b(\cdot)$

$$\min_{\beta} f\beta) = -\langle y, X\beta\rangle + \langle 1, b(X\beta)\rangle,$$

  where b is applied componentwise
- $\nabla f(\beta) = -X^T y + X^T b'(X\beta), \nabla^2 f(\beta) = X^T \mathrm{diag}\{b''(X\beta)\} X$
- Regression: $b(t) = t^2/2$, $b'(t) = t$, $b''(t) = 1$
- Bernoulli: $b(t) = \log(1 + \exp(t))$, $b'(t) = \frac{\exp(t)}{1+\exp(t)}$, $b'' = \exp(t)/(1 + \exp(t))^2 \leq \frac{1}{4}$.
- Poisson: $b(t) = \exp(t) = b' = b''$ (Hessian unbounded)

Example: Gaussian graph learning

- Recall $\min_{\Omega} -\log\det\Omega + \langle \Omega, \hat{\Sigma} \rangle + \lambda \|\Omega\|_1$ s.t. $\Omega \succ 0$
- From the previous lecture, $\nabla l = \hat{\Sigma} - \Omega^{-1}$ (l: loss)
- Some issues to tackle in performing 'gradient descent'
  - The penalty (& the constraint)? Proximal/barrier
  - Unbounded Hessian? Line search
    - Of course, it's bounded in the vicinity of Ω^o
    - Matrix inverse is still too expensive to compute
  - Slow convergence rate for large p? Momentum /BCD

Example: back propagation

- Denote an L-layer feedforward neural network by

$$X \equiv \Sigma_L \xrightarrow{B_L} \Sigma_{L-1} \xrightarrow{B_{L-1}} \cdots \Sigma_1 \xrightarrow{B_1} \Sigma_0 \simeq Y,$$

  where $\Sigma_{i-1} = \sigma(\Sigma_i B_i)$, $i = 2, \ldots, L$, $\Sigma_0 = g(\Sigma_1 B_1)$
- $L = 1$: perceptron. $(X, Y) \in \mathbb{R}^{n\times p} \times \mathbb{R}^{n\times m}$: known
- $g(\cdot)$: identity for regression, softmax for classification
- Traditional sigmoid activation: $\sigma(\theta) = 1/(1 + \exp(-\theta))$
- Modern choice of activation for all **hidden** units

  ReLU: $\sigma(\theta) = \theta_+ (= \max\{0, \theta\})$ (rectified linear unit)

- We write $\Sigma_0 = g(\sigma(\cdots\sigma(\sigma(XB_L)B_{L-1})\cdots B_2)B_1)$ as $g^{B_1} \circ \cdots \circ \sigma^{B_{L-1}} \circ \sigma^{B_L} \circ X$ for short
- After choosing a proper loss function $l(\cdot; Y)$ to measure the discrepancy between Y and Σ_0, we can estimate B_i
  - Cross entropy: $l(\Sigma_0, Y) = -\sum_{i=1}^{n}\sum_{k=1}^{m} y_{ik} \log \Sigma_0[i,k]$ corresponding to **multinomial** when using soft-max g: $g(T) = [\exp(T[i,k])/\sum_k \exp(T[i,k])]$
  - ℓ_2-loss with $L = 3$: $\|Y - g^{B_1} \circ \sigma^{B_2} \circ \sigma^{B_3} \circ X\|_F^2$
- We are ready to derive a gradient update in the form

$$B^{t+1} = B^t - \alpha_t \nabla f(B^t)$$

where $f(B)$ is the loss on B and α_t is the learning rate

- The chain rule results in a simple **recursive** formula:

$$\begin{aligned}
\nabla_{B_1} f &= \Sigma_1^T \Phi_1, \ \Phi_1 = g'(\Sigma_1 B_1) \circ \nabla l(\Sigma_0) \\
\nabla_{B_2} f &= \Sigma_2^T \Phi_2, \ \Phi_2 = \sigma'(\Sigma_2 B_2) \circ (\Phi_1 B_1^T) \\
\nabla_{B_3} f &= \Sigma_3^T \Phi_3, \ \Phi_3 = \sigma'(\Sigma_3 B_3) \circ (\Phi_2 B_2^T) \\
\cdots\cdots
\end{aligned}$$

  where $\circ$ denotes elementwise product

- Φ_i: errors to **back**-propagate, for updating B_{i+1}

$$\xleftarrow{B_L} \Phi_L \xleftarrow{B_{L-1}} \Phi_{L-1} \cdots \xleftarrow{B_2} \Phi_2 \xleftarrow{B_1} \Phi_1$$

- Two passes: 1st (forward) pass for getting Σ_i, 2nd: Φ_i

- The elegant gradient update does **not** resolve all practical issues in network training
  - Vanishing gradient, learning rate, nonconvexity, topology design, overfitting, etc.
- For example, a sigmoid σ gives $\sigma'(\theta) = \frac{\exp(\theta)}{(1+\exp(\theta))^2} \leq \frac{1}{4}$
- What will happen to $\nabla_{B_i} f$ when i is large? What about ReLU?
  - Pre-training, GPU-computing
- We also need to learn how to choose an appropriate learning rate (stepsize) at iteration t

Stepsize

- How to choose the stepsize? Note that solving the problem $\min_{\alpha>0} f(\beta^t + \alpha p_t)$ exactly is often costly
- Typically, we apply *inexact* line search methods
- Yet sometimes universal choices of α_t are possible —e.g., $\alpha_t = 1/L$ under $\nabla^2 f (:= \mathrm{D}(\nabla f)) \preceq LI$
  - Regression: $\alpha = 1/\|X\|_2^2$. Logistic regression: $4/\|X\|_2^2$
- To see this, let's cast GD as an **MM** algorithm

Majorization-minimization principle

- Construct a surrogate function $g(\beta, \beta^-)$ satisfying

$$g(\beta, \beta^-) \geq f(\beta) \text{ and } g(\beta, \beta) = f(\beta), \ \forall \beta, \beta^-$$

- Now define $\beta^{t+1} \in \arg\min_\beta g(\beta, \beta^t)$, $\forall t \geq 0$, leading to a convergent algorithm (in terms of functional value):

$$f(\beta^{t+1}) \leq g(\beta^{t+1}, \beta^t) \leq g(\beta^t, \beta^t) = f(\beta^t)$$

- The first inequality and the last equality are due to the g-construction, the second due to the g-optimization

Linearization

- This extremely useful technique linearizes f by

$$
\begin{aligned}
g(\beta,\beta^-) &= \{f(\beta) - \mathbf{\Delta}_f(\beta,\beta^-)\} + \rho\mathbf{D}_2(\beta,\beta^-) \\
&= \{f(\beta^-) + \langle \nabla f(\beta^-), \beta - \beta^- \rangle\} + \frac{\rho}{2}\|\beta - \beta^-\|_2^2
\end{aligned}
$$

- The nonlinear problem $\min f(\beta)$ is much simplified: $\beta_{opt}(\beta^-) = \arg\min g(\beta,\beta^-) = \beta^- - (1/\rho)\nabla f(\beta^-)$
- ρ: inverse stepsize ($\alpha = 1/\rho$ or $\alpha_t = 1/\rho_t$)
- Is g a valid surrogate? It is easy to see $g(\beta,\beta) = f(\beta)$

- To ensure $g(\beta, \beta^-) \geq f(\beta)$, it suffices to have

$$f(\beta) - f(\beta^-) - \langle \nabla f(\beta^-), \beta - \beta^- \rangle \leq \frac{\rho}{2}\|\beta - \beta^-\|_2^2, \ \forall \beta, \beta^-$$

- If ∇f is L-Lipschitz continuous, $\rho \geq L$ works
- Actually, we have got a practical line search criterion:

$$g(\beta^{t+1}, \beta^t) \geq f(\beta^{t+1})$$

- $f(\beta^t) - f(\beta^{t+1}) \geq g(\beta^t, \beta^t) - g(\beta^{t+1}, \beta^t) \geq \rho_t \mathbf{D}_2(\beta^t, \beta^{t+1})$ (note the sufficient decrease from g-optimization)
- **No** differentiability is required! The condition can be relaxed further if considering the *overall* convergence.

Lipschitz gradient

- Assume ∇f is Lipschitz: $\|\nabla f(\beta) - \nabla f(\tilde{\beta})\|_* \leq \|\beta - \tilde{\beta}'\|$
- Then Δ_f is bounded:

$$
\begin{aligned}
& f(\beta) - f(\tilde{\beta}) - \langle \nabla f(\tilde{\beta}), \beta - \tilde{\beta} \rangle \\
&= \int_0^1 \langle \nabla f(\tilde{\beta} + t(\beta - \tilde{\beta})), \beta - \tilde{\beta} \rangle \, \mathrm{d}t - \int_0^1 \langle \nabla f(\tilde{\beta}), \beta - \tilde{\beta} \rangle \, \mathrm{d}t \\
&= \int_0^1 \langle \nabla f(\tilde{\beta} + t(\beta - \tilde{\beta})) - \nabla f(\tilde{\beta}), \beta - \tilde{\beta} \rangle \, \mathrm{d}t \\
&\leq \int_0^1 Lt\|\beta - \tilde{\beta}\|^2 \, \mathrm{d}t \leq \frac{L}{2}\|\beta - \tilde{\beta}\|^2.
\end{aligned}
$$

Backtracking

- An attractive line search method for GD/Newton
- Typically, choose $\alpha = 1$ (Newton), $r \in (0,1), c \in (0,1)$.
- Repeat $\alpha \leftarrow r\alpha$ until

$$\textbf{Armijo}: \quad f(\beta^t + \alpha p^t) \le f(\beta^t) + c\langle \nabla f(\beta^t), \alpha p^t \rangle$$

- The Armijo rule is one of Wolfe conditions (the other is a curvature condition). It may not work alone but we are doing back-tracking to prevent small α's

- An equivalent condition:

$$\mathbf{\Delta}_f(\beta^+, \beta^-) \leq \frac{1-c}{c}[f(\beta^-) - f(\beta^+)],$$

which implies

$$f(\beta^t) - f(\beta^{t+1}) \geq \frac{2\rho_t}{1 + \frac{1-c}{c}}\mathbf{D}_2(\beta^t, \beta^{t+1})$$

- Compare it with the previous condition.
- $c = 0.5$? $c = 0.1$? $c = 0.9$: stringent requirement

Example: gradient boosting

- Consider an additive model (e.g., x_j as base learners)

  $$y \sim \beta_1 T(x_1, \ldots, x_p; \gamma_1) + \cdots + \beta_j T(x_1, \ldots, x_p; \gamma_j) + \cdots =: f(\beta,$$

- Loss function: $l(f(\beta, \gamma;); y) = l_0(\beta; y, \gamma)$
- We estimate f in a forward stagewise manner:

  $$\min_{\beta_j, \gamma_j} l(f^- + \beta_j T(X, \gamma_j); y)$$

  where f^- is the current estimate of the function
- With a proper loss, β_j (given γ) can often be solved in closed form, resulting in `AdaBoost`, `LogitBoost`, etc. γ_j? Often difficult in the case of a <u>non-quadratic</u> loss

- The analogy between the progressive process and gradient descent suggests (descent *direction* + stepsize)

$$\gamma_j = \arg\min_{\gamma} \|(-\nabla l(f^-)) - T(X;\gamma)\|_2^2$$

followed by $\beta_j = \arg\min_\beta l(f^- + \beta T(X, \gamma_j); y)$

- When T_j are trees (additive themselves), use only the regions (directions) from the learner-optimization step
- There exisit many other ways to update the model (e.g., a "full correction" to refit all region values, a small learning rate for **regularization**)
- Boosting belongs to a large class of **greedy** algorithms

Why using pseudo-residuals $-\nabla l(f^-)$ and the ℓ_2 loss?

- A better idea is to consider GD (1 step) on β
- Assume $T(X;\gamma_j)$ has not entered the model before, $\beta_j^- = 0$; so $\beta_j \leftarrow 0 - \alpha \nabla l_0(\beta_j^-; \beta_1^-, \ldots, \beta_{j-1}^-)|_{\beta_j^-=0}$
- Search direction: find γ_j to minimize the magnitude of

$$\nabla l_0(\beta_j; \beta_1^-, \ldots, \beta_{j-1}^-) = T(X;\gamma_j)^T \nabla l(f^-)$$

- Assuming all learners are of norm 1, this is equivalent to $\arg\min_\gamma \|\nabla l(f^-) - T(X;\gamma)\|_2^2$ (only regions matter)

Convergence analysis

- $\beta^{t+1} = \beta^t - \nabla f(\beta^t)/\rho$ with $\rho > 0$ and ∇f is $\mathrm{Lip}(L)$
- Then $f(\beta^t) - f(\beta^{t+1}) \geq (2\rho \mathbf{D}_2 - \boldsymbol{\Delta}_f)(\beta^t, \beta^{t+1})$ and so

$$\min_{t \leq T} \|\nabla f(\beta^t)\|_2^2 \leq \frac{\rho^2}{2\rho - L} \frac{f(\beta^0) - f(\beta^{T+1})}{T+1}$$

- GD has at least sublinear convergence rate $\mathcal{O}(1/T)$

- Let f be μ-strongly convex, or $\mu I \preceq \nabla^2 f \preceq LI \preceq \rho I$
- Then $\mathbf{\Delta}_f \geq (\rho/\kappa)\mathbf{D}_2$ with $\kappa := \rho/\mu$.
- Let β^o be the optimal solution. Then

$$g(\beta^{t+1}, \beta^t) + \rho \mathbf{D}_2(\beta^o, \beta^{t+1}) \leq g(\beta^o, \beta^t) \Rightarrow$$
$$(\rho \mathbf{D}_2 + \mathbf{\Delta}_f)(\beta^{t+1}, \beta^o) \leq (\rho \mathbf{D}_2 - \mathbf{\Delta}_f)(\beta^t, \beta^o) \Rightarrow$$
$$\mathbf{D}_2(\beta^{t+1}, \beta^o) \leq \frac{\kappa - 1}{\kappa + 1}\mathbf{D}_2(\beta^t, \beta^o) \leq \left(\frac{\kappa - 1}{\kappa + 1}\right)^{t+1}\mathbf{D}_2(\beta^0, \beta^o)$$

- On strongly convex problems, the convergence is linear but depends on the condition number L/μ

$$T \geq \log(\frac{1}{\epsilon})\frac{\kappa - 1}{2} + \log \mathbf{D}_2(\beta^0, \beta^o) \Rightarrow \mathbf{D}_2(\beta^{T+1}, \beta^o) \leq \epsilon$$

- GD works well if the problem is well-conditioned, but as $\kappa \gg 1$ its zipzaging behavior is notorious
- We will see that Newton's method often has **quadratic** convergence and is condition-number free

Rates of Convergence

- Let x_k be the kth iterate, and f_k the kth function value
- **Assume** x_k and f_k converge to x^* and f^* respectively.
- Let $e_k = \|x_k - x^*\|$ or $e_k = \|f_k - f^*\|$.
- Linear convergence or geometric (!) convergence: $e_k \leq C\beta^k$ for some $C > 0$, $\beta \in (0, 1)$.
  - A sufficient condition: $\limsup e_{k+1}/e_k \leq \beta \in (0, 1)$
- Superlinear: for any $\beta \in (0, 1)$ and some $C(\beta)$. A sufficient condition: $\limsup e_{k+1}/e_k = 0$
- Quadratic: $e_k \leq C\beta^{2^k}$, for some $\beta \in (0, 1), C > 0$.
  - A sufficient condition: $\limsup e_{k+1}/e_k^2 \leq C < +\infty$

Steepest descent directions

- Given a norm $\|\cdot\|$, the normalized steepest descent direction is defined as $p_{nsd} = \arg\min_{p:\|p\|=1} \langle \nabla f(\beta), p \rangle$
- (Un-normalized) sd direction: $p_{sd} = \|\nabla f(\beta)\|_* \, p_{nsd}$
- By Holder's inequality,

$$\langle \nabla f(\beta), p_{nsd} \rangle = -\|\nabla f(\beta)\|_*, \ \langle \nabla f(\beta), p_{sd} \rangle = -\|\nabla f(\beta)\|_*^2$$

- $p_{sd} = \arg\min_p \langle \nabla f(\beta), p \rangle + (1/2)\|p\|^2$ (un-normalized)
  - $\langle \nabla f(\beta), p \rangle \geq -\|\nabla f(\beta)\|_* \|p^o\|$ & $\|p^o\| = \|\nabla f(\beta)\|_*$
- Euclidean norm: $p_{sd} = -\nabla f(\beta)$ (gradient descent)

Coordinate descent

- What are steepest descent directions under ℓ_1-norm?

$$p_{nsd} = \arg\min_{\|p\|_1 \leq 1} \langle \nabla f(\beta), p \rangle = -\mathrm{sgn}(\frac{\partial f(\beta)}{\partial \beta_j}) e_j,$$

$$p_{sd} = \|\nabla f\|_\infty p_{nsd} = -\frac{\partial f(\beta)}{\partial \beta_j} e_j,$$

where $j : |\frac{\partial f(\beta)}{\partial \beta_j}| = \|\nabla f(\beta)\|_\infty$

- This is the coordinate descent algorithm using the Gauss-Southwell (**GS**) rule for coordinate selection
- Some other popular rules: cyclic, randomized(!)

- Interestingly, we can characterize GS as an outcome of

$$g(\beta, \beta^-) = f(\beta^-) + \langle \nabla f(\beta^-), \beta - \beta^- \rangle + \frac{\rho}{2} \|\beta - \beta^-\|_1^2$$

- The surrogate perspective offers more insights. Choosing the last term flexibly gives various directions.
- Defining strong-convexity in a proper way can greatly facilitate theoretical studies

Block coordinate descent

- A straightforward extension: *block-by-block* update
- Block selection: cyclic, greedy, random
- Within-block optimization: exact vs. inexact
- CD/BCD algorithms may be very inefficient and get stuck at a non-stationary point in convex optimization
- However, BCD is simple to implement and scales well on large problems with good decomposability
- Example: graphical lasso (Friedman et al 07)

Example: lasso

- Consider $\min_\beta \|y - X\beta\|_2^2/2 + \lambda\|\beta\|_1$ with $\|x_j\|_2^2 = 1$
- Solving for β as a whole looks difficult
- Let's turn to $\min_{\beta_1} \frac{1}{2}\|y - X_{(-1)}\beta_{(-1)} - x_1\beta_1\|_2^2 + \lambda|\beta_1|$ or

  $$\frac{1}{2}\min_{\beta_1} \|x_1^T r_1 - \beta_1\|_2^2 + \lambda|\beta_1|$$

  where $r_1 = y - X_{(-1)}\beta_{(-1)}$
- No design! So $\beta_1^+ = \Theta_S(x_1^T r_1; \lambda)$ given β_j $(2 \leq j \leq p)$
- Although this CD algorithm may be slow for small λ, we can use warm starts to do *pathwise* computation.

Example: sparse PCA

- Recall that given $X = UDV^T \in \mathbb{R}^{n \times p}$, $X v_k$ $(1 \leq k \leq r)$ construct the top k PCs from p predictors
- A (nonconvex) form of rank-1 sparse PCA is defined by

$$\min_{d \in \mathbb{R}, u \in \mathbb{R}^n, v \in \mathbb{R}^p} \|X - duv^T\|_F^2 + \lambda \|v\|_0 \text{ s.t. } \|u\|_2^2 = 1, \|v\|_2^2 = 1$$

- An equivalent form after evaluating the optimal d, u

$$\max_{v \in \mathbb{R}^p} v^T (X^T X) v - \lambda \|v\|_0 \quad \text{s.t. } \|v\|_2^2 = 1$$

- Another equivalent form (by introducing $s = dv$)

$$\min_{u,s} \|X - vs^T\|_F^2 + \lambda\|s\|_0 \text{ s.t. } \|v\|_2^2 = 1$$

- Given s, it can be shown that $v_o = Xs/\|Xs\|_2$
- Given v, s can be obtained by hard thresholding
- In general, the rank-r sparse PCA can be solved as a whole (S16): $\min_{S,V} \|X - VS^T\|_F^2 + P(S)$ s.t. $V^TV = I$
- V: Procrustes rotation; S: (multivariate) thresholding

Pure Newton's method

- $p_t = -H_t^{-1} \nabla f(\beta^t)$ is a descent direction if H_t is pd
- When f is strongly convex, $H_t = \nabla^2 f(\beta^t)$ surely works
- **Pure** Newton: $\beta^{t+1} = \beta^t - H_t^{-1} \nabla f(x^t)$ $(\alpha_t \equiv 1)$, a result of using a quadratic approximation $(H := \nabla^2 f)$

$$f(\beta) \approx f(\beta^-) + \langle \nabla f(\beta^-), \beta - \beta^- \rangle + \frac{1}{2} (\beta - \beta^-)^T H(\beta^-)(\beta - \beta^-)$$

- No global convergence or performance guarantee even when f is convex and smooth!

Damped Newton

- Seen from the surrogate

  $$g_t(\beta, \beta^t) = f(\beta^t) + \langle \nabla f(\beta^t), \beta - \beta^t \rangle + \frac{\rho_t}{2} \| H(\beta^t)^{1/2} (\beta - \beta^t) \|_2^2,$$

  the local approximation alone does not guarantee the **majorization** condition $g(\beta^{t+1}, \beta^t) \geq f(\beta^{t+1})$
- We need ρ_t (line search) to get global convergence
  - Backtracking is well suited for Newton
- Another idea is to modify the weighting matrix to be **pd**, e.g., $H \to H + \delta I$ with a proper δ (δ large: GD)

Condition-number free

- Consider $\min f(X\beta)$ (e.g., GLM) with X nonsingular (square) but ill-conditioned (i.e., $\kappa = \frac{\sigma_{\max}(X)}{\sigma_{\min}(X)}$ large)
- Let $\gamma = X\beta$ ('best' change of coordinates). Ignoring the stepsize, the Newton method on β gives

$$
\begin{aligned}
\beta^+ - \beta^- &= -(X^T \nabla^2 f(\gamma^-) X)^{-1} X^T \nabla f(\gamma^-) \\
&= X^{-1} (\nabla^2 f(\gamma^-))^{-1} \nabla f(\gamma^-), \text{ or} \\
\gamma^+ - \gamma^- &= (\nabla^2 f(\gamma^-))^{-1} \nabla f(\gamma^-)
\end{aligned}
$$

- The same path as applying Newton on γ (κ-free)!
- The affine **equivariant** property is in contrast to GD

- Consider an example in logistic regression (b'' bounded)
- Let $X = \mathrm{diag}\{100, 1\}$ (poor scaling)
- Let's compare β-updatings in the γ-domain:

$$\gamma^+ - \gamma^- = \{XX^T/\|X\|_2^2\}(1/\max b'')(y - b'(\gamma^-)) \text{ (GD)}$$
$$\gamma^+ - \gamma^- = \mathrm{diag}\{1/b''(\gamma^-)\}(y - b'(\gamma^-)) \text{ (Newton)}$$

- Note the condition-number free convergence of Newton
- Newton's method is the routine for fitting GLMs; in the canonical case, it is identical to Fisher scoring.

Example: concentration matrix estimation

- $g(\Omega) = \nabla l(\Omega) = \hat{\Sigma} - \Omega^{-1}$. From $\mathrm{d}g = \Omega^{-1}\,\mathrm{d}\Omega\,\Omega^{-1}$, the 2nd-order approximation of $l(\Omega + \mathrm{d}\Omega)$ is

$$l(\Omega) + \langle \hat{\Sigma} - \Omega^{-1}, \mathrm{d}\Omega \rangle + \frac{1}{2} tr\{\mathrm{d}\Omega\,\Omega^{-1}\,\mathrm{d}\Omega\,\Omega^{-1}\}$$

- Ignoring the pd constraint (or any penalty), from $\hat{\Sigma} - \Omega^{-1} + \Omega^{-1}\,\mathrm{d}\Omega\,\Omega^{-1} = 0$, we get $\mathrm{d}\Omega = \Omega - \Omega\hat{\Sigma}\Omega$ or

$$\Omega^{t+1} = 2\Omega^t - \Omega^t\hat{\Sigma}\Omega^t = \Omega^t(2I - \hat{\Sigma}\Omega^t) \text{ (pure)}$$

- Very different from GD. No inverse. Local convergence.
- The learning problem is tricker: ℓ_1, psd, & stepsize.

- A reasonable idea to get the Newton direction:

$$l(\Omega)+\langle\hat{\Sigma}-\Omega^{-1},\, \mathrm{d}\Omega\rangle+\frac{1}{2}tr\{\,\mathrm{d}\Omega\,\Omega^{-1}\,\mathrm{d}\Omega\,\Omega^{-1}\}+\lambda\|\Omega+\mathrm{d}\Omega\|_1$$

- Vectorization gives a lasso-type problem with design $(\Omega^{-1}\otimes\Omega^{-1})^{1/2}$, which can be solved by **BCD**

$$\langle\hat{\Sigma}-\Omega^{-1},\, \mathrm{d}\Omega\rangle+\frac{1}{2}(\,\mathrm{vec}\,\mathrm{d}\Omega)^T(\Omega^{-1}\otimes\Omega^{-1})\,\mathrm{vec}\,\mathrm{d}\Omega+\lambda\|\Omega+\,\mathrm{d}\Omega\|_1$$

- The structure of the Gram matrix helps reduce the cost. We also need a proper line search method.
- It leads to a proximal Newton method (Hsieh et al, 11)

Quasi-Newton

- Newton is often more expensive than GD *per iteration*
  - Hessian inverse: $\mathcal{O}(p^3)$, memory: $\mathcal{O}(p^2)$
- Main idea: approximate H_t or H_t^{-1} or p_t!
- Introduce B_t in place of the Hessian used in Newton: $\beta^{t+1} = \beta^t - \alpha_t p_t$ with $p_t = -B_t^{-1} \nabla f(\beta^t)$
- An efficient way is to update $B_t/B_t^{-1}/p_t$ sequentially
  - Typical cost: $\mathcal{O}(p^2)$

- With β^{t+1} available, how to update B_{t+1}? It should satisfy the secant equation: $B_{t+1}s_t = y_t$ where $s_t = \beta^{t+1} - \beta^t$, $y_t = \nabla f(\beta^{t+1}) - \nabla f(\beta^t)$
- Add other properties (symmetry, pd(?), min-change, min-norm, rank-1 or rank-2 modification) to the under-determined system to get various solutions
- Alternatively, we may update B_{t+1}^{-1}, or $B_t^{-1}p_t$ using m pairs of (s_i, y_i) with $m \ll p$ (storage cost: $\mathcal{O}(mp)$)
- Examples: SR1, L-BFGS, etc.