# Classical Gradient Methods

- Note simultaneous course at AMSI (math) summer school: Nonlin. Optimization Methods
  (see http://wwwmaths.anu.edu.au/events/amsiss05/)
- Recommended textbook (Springer Verlag, 1999): Nocedal & Wright, Numerical Optimization
- Here: just quick overview, unconstrained only
- But will consider large, nonlinear problems

# Function Optimization

- Goal: given (diff'able) function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ find minimum $\vec{w}^* = \arg\min_{\vec{w}} f(\vec{w})$
- Gradient methods find only local minimum
- For convex functions, local min. = global min.

In machine learning,

- Fn. is defined over data: $f(\vec{w}) = E_{\vec{x}}[f(\vec{w}; \vec{x})]$
- May comprise loss and regularization terms

# Methods by Gradient Order

- 0th order (direct, gradient-free) methods use only the function values themselves
- 1st order gradient methods additionally use function's gradient $\vec{g} = \vec{g}(\vec{w}) = \frac{\partial f(\vec{w})}{\partial \vec{w}}$
- 2nd order gradient methods also use the function's Hessian $H = H(\vec{w}) = \frac{\partial^2 f(\vec{w})}{\partial \vec{w}\, \partial \vec{w}^T}$

# Direct (Gradient-Free) Methods

Many distinct algorithms:

- Simulated annealing, Monte Carlo optim.
- Perturbation methods, SPSA, Tabu search
- Genetic algorithms, evolutionary strategies, ant colony optimization, …

Differ in many implementation details but all share a common approach.

# Prototypical Direct Method

**Randomly initialize pool W of candidates**

**Repeat until converged:**

**pick parent(s) $w_i$ from**

**generate child(ren) $w_i'$ = perturb($w_i$)**

**compare child to parent (or entire pool):**

**$\Delta$ = f($w_i'$) - f($w_i$)**

**if $\Delta$ < 0 accept $w_i'$ into W (may replace $w_i$)**

**else if global optimization:**

**accept $w_i'$ into W with probability P($e^{-\Delta}$)**

# Direct Methods: Advantages

- No need to derive or compute gradients
  - Can solve discrete/combinatorial problems
  - Can address even non-formalized problems
- Can find (non-convex fn.'s) global optimum
- Highly and easily parallelizable
- Very fast iteration when perturbation and evaluation are both incremental, *i.e.* $O(1)$

# Direct Methods: Disadvantages

- No sense of appropriate direction or size of step to take (perturbation is random)
  - Some algorithms try to fix this heuristically
- Takes too many iterations to converge
- Global optim. requires knowing acceptance of inferior candidates $\Rightarrow$ slower still
- No strong mathematical underpinnings $\Rightarrow$ jungle of ad-hoc heuristics

# Gradient Descent

- Perturbs parameter vector in steepest downhill direction (= neg. gradient): $\vec{w}_{t+1} = \vec{w}_t - \eta \vec{g}_t$
- Step size $\eta$ can be set
  - to small positive constant: simple gradient descent
  - by line minimization: steepest descent
  - adaptively (more on this later)

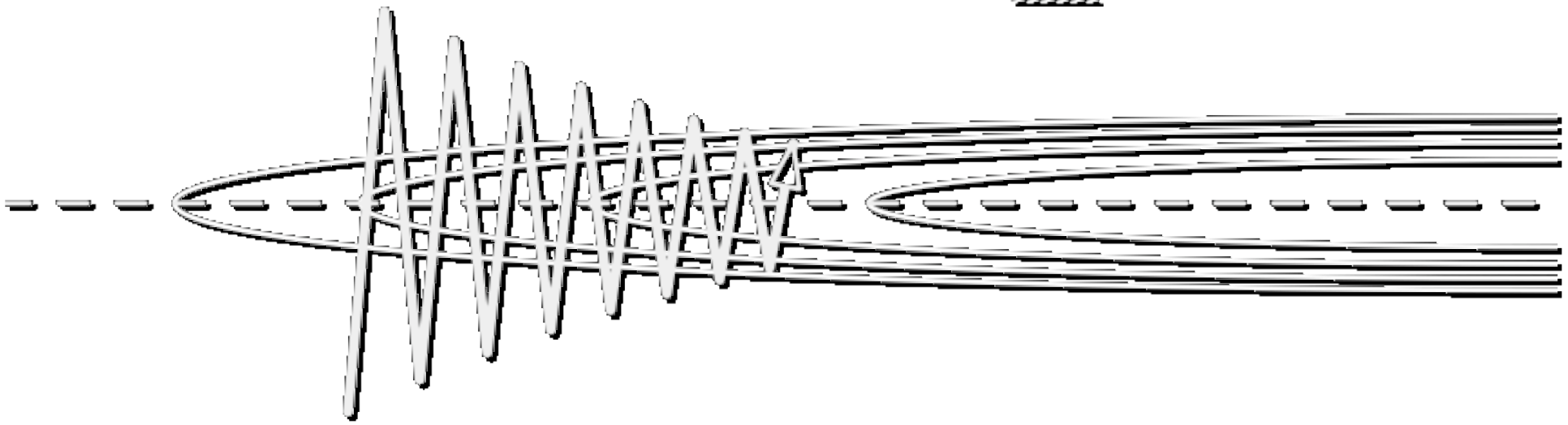
Advantage:

- Cheap to compute: iteration typically just $O(n)$

# Gradient Descent: Disadvantages

- Line minimization may be expensive
- Convergence slow for ill-conditioned problems:

  #iterations ≥ condition# $\kappa = \frac{\lambda_{max}}{\lambda_{min}}$ of Hessian



# Newton's Method

- Local quadratic model

$$f(\vec{w}) = \tfrac{1}{2}(\vec{w} - \vec{w}^*)^T H(\vec{w} - \vec{w}^*)$$

has gradient $\vec{g}(\vec{w}) = H(\vec{w} - \vec{w}^*)$

therefore let $\vec{w}_{t+1} = \vec{w}_t - H_t^{-1}\vec{g}_t$

# Newton's Method

Big advantage:

- Jumps directly to minimum of quadratic bowl (regardless of ill-conditioning)

Disadvantages:

- Hessian expensive to invert: nearly $O(n^3)$
- Hessian must be positive definite: $\lambda_{min} > 0$
- May make huge, uncontrolled steps

# Gauss-Newton Approximation

Let $f = l \circ \vec{m},\ \vec{m} : \mathbb{R}^n \rightarrow \mathbb{R}^p$ = model, $l$ = loss

Then $$H_f = \underbrace{J_{\vec{m}}^T H_l J_{\vec{m}}} + \sum_{i=1}^{p} H_{m_i}(J_l)_i$$

Gauss-Newton: $G_f$

Jacobian: $$(J_{\vec{m}})_{ij} = \frac{\partial m_i(\vec{w})}{\partial w_j}$$

- $H_l \geq 0 \Rightarrow G_f \geq 0$
- At minimum, $G_f = H_f$
- For sum-squared loss: $H_l = I$ pseudo-inverse and $G_f^{-1} g = (J_{\vec{m}}^T J_{\vec{m}})^{-1} J_{\vec{m}}^T J_l^T = J_{\vec{m}}^+ J_l^T$

# Levenberg-Marquardt

$$\vec{w}_{t+1} = \vec{w}_t - (G_t + \lambda \operatorname{diag}(G_t))^{-1} \vec{g}_t$$

- $G_t$ is Gauss-Newton approximation to $H_t$ (guaranteed positive semi-definite)
- $\lambda \geq 0$ adaptively controlled, limits step to an elliptical model-trust region
- Fixes Newton's stability issues, but still $O(n^3)$

# Quasi-Newton: BFGS

- Iteratively updates estimate $B$ of $H^{-1}$
- Guarantees $B^T = B$ and $B > 0$
- Reduces complexity to $O(n^2)$ per iteration
- Requires line minimization (direction $B_t \vec{g}_t$)
- Update formula:

$$B_{t+1} = B_t + \frac{\Delta\vec{w}\,\Delta\vec{w}^T}{\Delta\vec{w}^T\Delta\vec{g}} - \frac{B_t\Delta\vec{g}\,\Delta\vec{g}^T B_t^T}{\Delta\vec{g}^T B_t \Delta\vec{g}} + \vec{u}\,\Delta\vec{g}^T B_t \Delta\vec{g}\,\vec{u}^T$$

where $$\vec{u} \equiv \frac{\Delta\vec{w}}{\Delta\vec{w}^T\Delta\vec{g}} - \frac{B_t\Delta\vec{g}}{\Delta\vec{g}^T B_t\Delta\vec{g}}$$

# Conjugate Gradient

$\vec{w}_{t+1} = \vec{w}_t + \alpha \vec{v}_t$; $\alpha$ set by line minimization

Search directions are conjugate:

$\vec{v}_0 = -\vec{g}_0$; $\vec{v}_{t+1} = \beta \vec{v}_t - \vec{g}_t$

$i \neq j \Rightarrow \vec{v}_i^T H \vec{v}_j = 0$

$\beta = \frac{\vec{g}_t^T (\vec{g}_t - \vec{g}_{t-1})}{\vec{v}_t^T (\vec{g}_t - \vec{g}_{t-1})}$

(= orthogonal in local Mahalonobis metric)

(Hestenes-Stiefel, 1952) (a.k.a. Beale-Sørenson)

NB: other formulae for $\beta$ (Polak-Ribiere, Fletcher-Reeves) equivalent for quadratic but inferior for nonlinear fn.s!

# Conjugate Gradient: Properties

- No matrices $\Rightarrow$ each iteration costs only $O(n)$
- Minimizes quadratic fn. exactly in $n$ iterations
- Restart every $n$ iterations for nonlinear fn.s
- Optimal progress after $k < n$ iterations

An incremental 2nd-order method! Revolutionary.

- Drives nearly all large-scale optimization today.