Numerical Optimization

- We only consider ***unconstrained*** optimization of ***smooth*** functions $f(\boldsymbol{x})$

Global Optimization

- *e.g.* if ***all*** or ***the best*** local minima are required
- There are ***no*** good methods for ***general, high-dimensional*** functions
- It's hard to do much better than just starting many local searches from random points

— but choose the random distribution wisely . . .

- Alternative paradigms: ***simulated annealing***, ***genetic algorithms***, ***tabu search***

— all designed for discrete (combinatorial) problems with constraints, not for unconstrained continuous ones

— it's best to combine them with an *efficient* local search

Local Optimization

- Smooth function $f(\boldsymbol{x})$, gradient $\mathbf{g}(\boldsymbol{x}) = \frac{Df(\boldsymbol{x})}{D\boldsymbol{x}}$, Hessian $\mathbf{H} = \frac{D^2 f(\boldsymbol{x})}{D\boldsymbol{x}^2}$. ***Quadratic local model:***

$$\boxed{f(\boldsymbol{x} + \delta\boldsymbol{x}) \approx f(\boldsymbol{x}) + \mathbf{g} \cdot \delta\boldsymbol{x} + \frac{1}{2}\delta\boldsymbol{x}^\top \mathbf{H} \delta\boldsymbol{x}}$$

Order of Model

- ***Gradient*** (1^{st} order) methods use only $\mathbf{g}$
- ***Newton*** (2^{nd} order) methods use $\mathbf{g}$ and (approximate) $\mathbf{H}$

Search Organization

- ***Trust region:*** minimize approximate f within a restricted local region, *e.g.* a ball
- ***Line search:*** repeatedly choose a direction and move to minimum of f along it

— approximate minima suffice, but required accuracy depends on application

— many methods, efficient implementation can be delicate . . .

Gradient-like Methods

- ***Gradient descent:*** for *any* positive $\mathbf{M}$, f initially decreases along $\delta\boldsymbol{x} \propto -\mathbf{M}\mathbf{g}$
- This gives no 'natural' step size, so line search is needed

— fixed step sizes like $\delta\boldsymbol{x} = -k\mathbf{g}$ are *very* slow

— exact line search gives ***slow linear convergence:*** rate $\sim$ 1/Condition($\mathbf{M}^{-1}\mathbf{H}$)

— for narrow 'valleys', steps zig-zag *across* valley floor

— ***preconditioning:*** choose $\mathbf{M}$ to improve condition of $\mathbf{H}$

— 'momentum' terms may help, but convergence is still slow

Newton-like Methods

- ***Provided*** $\mathbf{H}$ is positive, $\delta\boldsymbol{x} = -\mathbf{H}^{-1}\mathbf{g}$ exactly minimizes the quadratic local model of f
- Each iteration must evaluate $\mathbf{H}$ & solve a linear system
- Rapid ***asymptotic convergence***

— each iteration ***doubles*** # significant digits

- Often slow / unstable when far from minimum

— naive method fails entirely for non-positive $\mathbf{H}$

— often prefer approximate $\mathbf{H}$'s that guarantee positivity

— regularize $\mathbf{H} \to \mathbf{H} + \lambda\mathbf{I}$ (but how to choose λ?)

— use line search along $\mathbf{H}^{-1}\mathbf{g}$ to guarantee decrease of f

Least squares problems

- $\min_{\boldsymbol{x}} \frac{1}{2}\|\boldsymbol{f}(\boldsymbol{x})\|^2$ where $\mathbf{H} = (\nabla\boldsymbol{f})(\nabla\boldsymbol{f})^\top + \boldsymbol{f} \cdot \nabla^2\boldsymbol{f}$
- ***Gauss-Newton*** method drops the second term, guaranteeing positive semi-definiteness
- ***Levenberg-Marquardt*** adds an additional $\lambda\mathbf{I}$ regularizer.

Quasi-Newton Methods

- Iteratively build up an approximation to $\mathbf{H}$ using $\mathbf{g}$ and the ***secant equations*** $\Delta\mathbf{g} \approx \mathbf{H}\Delta\boldsymbol{x}$.
- A good approach for general unstructured problems where the Hessian can be stored but is troublesome to compute.
- Fairly inaccurate line searches are OK $(\sigma \lesssim 0.9)$.
- Various possible update formulae for $\mathbf{H}$, $\mathbf{H}^{-1}$, or Choleski factors of $\mathbf{H}$
- 'Best' update is controversial. ***BFGS*** for $\mathbf{H}$ is often used
  - — positive definite, good damping of large eigenvalues
  - — for $\mathbf{H}^{-1}$ updates, the best method is unclear

Limited Memory Quasi-Newton

- A hybrid of quasi-Newton and conjugate gradient
- Maintains ***low-rank correction*** to a ***nominal*** $\mathbf{H}$
  - — gives conjugacy w.r.t. last few steps
  - — in practice $3 \leq$ Rank ≤ 7, even for large problems
- inaccurate line searches are OK
- Reduces to Q-N for rank=n, C-G for rank=1 with exact search
- Various update formulae for $\mathbf{H}$, $\mathbf{H}^{-1}$, Choleski($\mathbf{H}$)
  - — not yet clear which will prove best
- Will probably become the preferred method for large problems
  - — low, controllable memory usage
  - — good convergence properties
  - — Q-N is still preferred for smaller problems

Conjugate Gradient Methods

- Like gradient descent, but each search direction is chosen ***conjugate*** to previous one to reduce zig-zagging.
- Needs fairly accurate line search $(\sigma \lesssim 0.1)$.
- ***Quadratic termination:*** for quadratic problems and exact search, terminates in n^2 steps.
- Various update formulae, Polack-Ribière is the usual one.
- C-G is a traditional choice for large problems
  - — uses only $\mathbf{g}$, no need to store Hessian
  - — but not as efficient as quasi-Newton when storage is available
  - — will probably be displaced by limited-memory methods

Resources for Optimization

Books

- ***R. Fletcher***, 'Practical Methods of Optimization'. The standard reference, but theory-oriented.
- ***Gill, Murray & Wright***, 'Practical Optimization'. Very good but a bit dated.
- ***Numerical Recipes***. Easily accessible introduction. But both discussion and routines are *often unreliable*.
- ***NEOS*** WWW guide, http://www.mcs.anl.gov/home/otc/Guide/.

Code

- Almost all serious numerical code is in FORTRAN and is designed for ***very large*** problems.
- ***GAMS*** index (http://gams.nist.gov/) lists many commercial and free packages.
- ***Netlib*** (http://www.netlib.org/). A large collection of numerical analysis routines, including TOMS (code for many ACM published algorithms).
- ***Statlib*** (http://lib.stat.cmu.edu/). A collection of statistical routines.
- ***Numerical Recipes*** — easy to use but buggy.
- ***Interactive packages*** are good for playing with data

— commercial: S-plus, Matlab . . .

— free: Octave, Scilab . . .

Sum over Peaks

- $\boldsymbol{p}(M, \boldsymbol{\theta}|\mathcal{T})$ and $f(\boldsymbol{x}|M, \boldsymbol{\theta})$ are often dominated by ***a few sharp peaks*** (local maxima of 'fit')
- Peaks of $\boldsymbol{p}(M, \boldsymbol{\theta}|\mathcal{T})$ are typically much narrower than those of $f(\boldsymbol{x}|M, \boldsymbol{\theta})$ (by $\mathcal{O}(1/\sqrt{|\mathcal{T}|})$)

∴ Approximate integral as a ***weighted average*** over peaks of $\boldsymbol{p}(M, \boldsymbol{\theta}|\mathcal{T})$, of ***plug-in estimators*** $f(\boldsymbol{x}|M_i, \hat{\boldsymbol{\theta}}_i)$

$$f(\boldsymbol{x}) \approx \sum_{\text{peaks } i} f(\boldsymbol{x}|M_i \hat{\boldsymbol{\theta}}_i) \cdot \boldsymbol{p}(\text{peak}_i|\mathcal{T})$$

$$\boldsymbol{p}(\text{peak}_i|\mathcal{T}) \propto \boldsymbol{p}(M_i) \cdot \int \boldsymbol{p}(\mathcal{T}|M_i \boldsymbol{\theta}_i) \cdot \boldsymbol{p}(\boldsymbol{\theta}_i|M_i) \cdot d\boldsymbol{\theta}_i$$

- Weights are ***peak volumes***, *not* peak heights as in ML/MAP

— penalizes narrow peaks / sensitivity to parameters

- Use prior $\boldsymbol{p}(\boldsymbol{\theta}|M)$ to control unidentifiable parameters

Predictive Approach

- Hedge output $f(\boldsymbol{x})$ over ***all possible models***, weighting by ***posterior probability*** for model given training set $\mathcal{T}$

— *e.g.*, $f(\boldsymbol{x}) =$ posterior class probability $\boldsymbol{p}(k|\boldsymbol{x})$

- M labels ***type*** of model or ***region*** of parameter space
- $\boldsymbol{\theta}$ is ***parameter vector*** for model type M

$$f(\boldsymbol{x}) = \int f(\boldsymbol{x}|M, \boldsymbol{\theta}) \cdot \boldsymbol{p}(M, \boldsymbol{\theta}|\mathcal{T}) \cdot dM\, d\boldsymbol{\theta}$$

where

$$\underbrace{\boldsymbol{p}(M, \boldsymbol{\theta}|\mathcal{T})}_{\text{posterior for model } (M,\boldsymbol{\theta})} \propto \boldsymbol{p}(\mathcal{T}|M, \boldsymbol{\theta}) \cdot \underbrace{\boldsymbol{p}(\boldsymbol{\theta}|M)}_{\text{prior for } \boldsymbol{\theta} \text{ (regularization)}} \cdot \underbrace{\boldsymbol{p}(M)}_{\text{prior for model type}}$$

Saddle Point Approximation

- Gaussian approximation for $\boldsymbol{p}(\text{peak}_i|\mathcal{T})$ integral

$$\log \boldsymbol{p}(\text{peak}_i|\mathcal{T}) \approx \log \boldsymbol{p}(M_i) + \log \boldsymbol{p}(\mathcal{T}|M_i, \hat{\boldsymbol{\theta}}_i) \\ -\tfrac{1}{2} \log \det(\mathbf{I} + n\boldsymbol{\Sigma}_0 \mathbf{J})$$

- Assumes a weak Gaussian prior with covariance $\boldsymbol{\Sigma}_0$ centred not too far from MAP estimate $\hat{\boldsymbol{\theta}}_i$.
- Posterior inverse covariance is $\boldsymbol{\Sigma}_0^{-1} + n\mathbf{J}$ where $\mathbf{J} \equiv \frac{1}{n} \frac{D^2(-\log \boldsymbol{p}(\mathcal{T}|M_i, \hat{\boldsymbol{\theta}}_i))}{D\boldsymbol{\theta}^2}$ is average ***Fisher information*** (Hessian of log-likelihood) per training sample.
- $\det^{-\frac{1}{2}}(\mathbf{I} + n\boldsymbol{\Sigma}_0 \mathbf{J})$ is ***relative volume*** of posterior *vs.* prior covariance

BIC — Bayesian Information Criterion

- If $n\mathbf{J}$ dominates prior $\mathbf{\Sigma}_0^{-1}$ over $d^*(n)$ dimensions (‘identifiable parameters’):

$$-\tfrac{1}{2}\log\det(\mathbf{I}+n\mathbf{\Sigma}_0\mathbf{J}) \;\approx\; -\tfrac{1}{2}d^*(n)\cdot\log n+\ldots$$

- Corrects MAP estimates for different ***peak volumes***

— applies modulo overall normalization of posterior: only ***relative*** d^*’s are significant

- Correction depends critically on prior covariances $\mathbf{\Sigma}_0$!

Predictive Approach and Flexible Models

- Flexible models often have several local minima
- Deep, narrow “false” minima are common

— overfitting: delicate cancellations reproduce random noise in data, not true structure $\Rightarrow$ poor generalization

— often unstable, difficult to track through perturbations, re-training, cross-validation, . . .

- Performance of point-estimator methods (*e.g.* MAP/ML) is unpredictable as deepest minimum may be “false”

— *c.f.* Ripley p 165: 3 deepest minima have low predictive weight (*i.e.* narrow)

- ***Regularization*** reduces number of “false” minima
- ***Predictive averaging*** reduces their influence
- ***Bayesian Information Criterion*** gives approximate weight corrections without explicit integration

Overfitting Bias

- As in MAP/ML case, fitted probabilities of training examples ***overestimate*** population probabilities by $\mathcal{O}(1/n)$

$\therefore$ Above formulae also need a ***peak height*** (overfitting bias) correction of

$$-p^*/2n \qquad \text{where} \qquad p^* = \mathrm{trace}(\mathbf{J}^{-1}\mathbf{K})$$

- $\mathbf{K}$ = variance of grad log-likelihood over training examples
- NB: correction is only *half* of NIC correction for MAP case

Regularization

- Penalties that discourage ‘badly behaved’ or rapidly varying functions
- Usually express as ***prior*** over ***weights*** or ***output functions***

— many possibilities: weight decay, smoothness $\left\|\frac{Df}{Dx}\right\|^p$, . . .

— fix strength heuristically or by (cross-)validation

Why Regularize?

1. Improves generalization: overfitting & variability reduced (at cost of increased bias)
2. Restricts search to relevant part of space: desired solution is usually smooth and has moderate weights
3. Reduces effect of false minima: smoother functions have fewer minima and larger basins of attraction
4. Improves numerical convergence by reducing stiffness and plateaus caused by output saturation

Weight Decay

- Penalize large weights by ***weight decay prior***

$$\boxed{-\log \boldsymbol{p}(w) \;\approx\; \frac{1}{2\sigma_w^2}\sum_{ij} w_{ij}^2}$$

- Heuristic estimate of σ_w:

— input signals lie in range $\approx [0, 1]$ and act incoherently

— limit weighted inputs to $\approx \pm 2.5$ to prevent saturation

$\therefore$ within a factor of 3 or so, weights should be in range $\sigma_w \sim \frac{2.5}{0.5} \approx 5$

- Assumed form of prior is quite strong, especially independence assumption . . .

Incremental Approach

- Adjust size incrementally by adding/deleting single edges and/or units
- Can be misleading:

— removing units or weights can drastically change covariances and even structure of minima

— model may become 'trapped' in a poor local minimum

$\therefore$ Global retraining is advisable after each change . . .

Incremental Construction

- Incrementally add units trained to 'correct' current errors, delete units that are inactive or have low total weight
- ***Cascade*** or ***pyramid*** architectures add 1 unit connected to all previous ones
- ***SN-C*** method adds groups of C units to a single hidden layer
- ***SMART*** (from projection pursuit regression) adds then deletes units to find best compromise, with retraining at each step.

Pruning Heuristics

- Reduce complexity of network for

— stability / reduced overfitting

— faster computation

— summarization / extraction of "key features"

- Successful models tend either to be small and shallow, or to have carefully designed topologies . . .
- Measure performance by cross-validation or bias-corrected fit

Weight Pruning

- Set "insignificant" weights to zero: $|w_{ij}/\sigma(w_{ij})| \lesssim 1$
- Care is needed as correlations can inflate apparent σ
- Can estimate posterior variances $\sigma^2(w_{ij})$ from Fisher information $\frac{D^2(-\log \boldsymbol{p})}{Dw^2}$

— ***Optimal Brain Damage*** just inverts diagonal of Hessian

— ***Optimal Brain Surgeon*** adds a first order correction for off-diagonal terms