Optimization

Machine Learning I, Week 9

Sibylle Mueller

Based on a lecture in the class 'Evolutionary Computation', SS 2002, by Nicol Schraudolph

Optimization -- Terminology

Global optimization problem: given $f:D \to R$, find $\arg\min_{\bar{x}} f(\bar{x})$

D discrete: integer programming, combinatorial optimization, genetic algorithms

D continuous: ($typ.\ D = R^n$): gradient methods, evol. strategies

Local optimization: find x: $(\forall \bar{y}) f(\bar{x} + \varepsilon \frac{\bar{y}}{\| \bar{y} \|}) > f(\bar{x})$ as $\varepsilon \to 0$

Constrained optimization: additionally require

(vs. unconstrained) $h(\bar{x}) = 0$ (equality constraints)

$g(\bar{x}) > 0$ (inequality constraints)

Minimization $\arg\min_{\bar{x}} f(\bar{x})$ = Maximization $\arg\max_{\bar{x}}(-f(\bar{x}))$

Optimization Methods

Simpler methods for special forms of f:

linear programming: $f(\bar{x}) = \bar{a}^T \bar{x}$

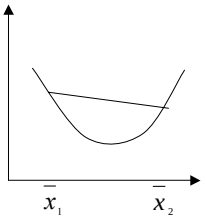
quadratic programming: $f(\bar{x}) = \bar{x}^T A\bar{x} + \bar{b}^T \bar{x}$

convex optimization: f is convex, i.e.,

$\forall \bar{x}_1, \bar{x}_2, 0 < \alpha < 1:$

$f(\alpha\bar{x}_1 + (1-\alpha)\bar{x}_2) < \alpha f(\bar{x}_1) + (1-\alpha) f(\bar{x}_2)$

$\Rightarrow$ no local minima!



nonlinear programming: everything else (the hard stuff)

No Free Lunch

Why all these different methods?

No Free Lunch Theorem (Wolpert & Macready, 1996):

"On average, any optimization algorithm is as good as any other, including random search."

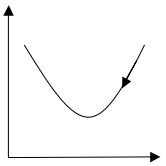
Key: this is averaged over all possible functions to optimize. Most functions have no structure at all.

$\Rightarrow$ to optimize efficiently, one must restrict oneself to some subset of functions, with structure to exploit (inductive bias).

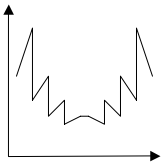
No Free Lunch

⇒ this means that the algorithm will do worse on some other functions.

Example:



but



f differentiable

⇒ use gradient

Different Optimization Methods

Thus: spectrum of optimization methods

Stochastic:	random search	simulated annealing	evolutionary algorithms	stochastic gradient
Deterministic:	simplex method	steepest descent	conjugate gradient	Levenberg--Marquardt

slow ⟷ fast

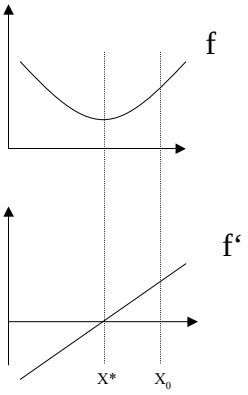
low inductive bias	high inductive bias
general, simple, robust	fragile, complex, specialized

Gradient Descent Methods

Example: $f(x) = \frac{1}{2}ax^2 + bx$

$f'(x) = ax + b$

$f''(x) = a$



To find minimum, solve $f'(x^*) = 0 \Rightarrow x^* = -\frac{b}{a}$

but: use only local information at point x_0:

$$b = -ax^* \Rightarrow f'(x_0) = a(x_0 - x^*)$$

so $$x^* = x_0 - \frac{f'(x_0)}{f''(x_0)}$$

Gradient Descent Methods

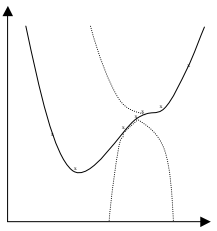
For non-quadratic functions, iterate this local quadratic approximation:

$$x_{t+1} = x_t - \frac{f'(x_t)}{f''(x_t)}$$ Newton-Raphson

Two things that can go wrong:

1) $f''(x_t) < 0 \Rightarrow$ use $|f''(x_t)|$

2) $f''(x_t) \approx 0 \Rightarrow$ use $|f''(x_t)| + \varepsilon$



Gradient Descent Methods

Multivariate generalization:

$f(\bar{x}) = \frac{1}{2}\bar{x}^T H\bar{x} + \bar{a}^T\bar{x}$ (H symmetric)

$\nabla f = H\bar{x} + \bar{a}$ gradient

$\nabla^2 f = H$ Hessian

Newton's method: $\bar{x}_{t+1} = \bar{x}_t - H^{-1}\nabla f\big|_{\bar{x}_t}$

Gradient Descent Methods

Newton's method, problems:

1) H is not positive semi-definite, i.e. $\exists\bar{x} : \bar{x}^T H\bar{x} < 0$

Solution: use an approximation of H that is guaranteed to be positive semi-definite, e.g. Gauss-Newton

2) H has very small eigenvalues, i.e. $\exists\bar{x} : \bar{x}^T H\bar{x} \approx 0$

Solution: add a small, positive diagonal matrix to H. This implements a model-trust region.

Gradient Descent Methods

Newton's method with these two safeguards added is known as Levenberg-Marquardt algorithm.

$$\bar{x}_{t+1} = \bar{x}_t - \left[G + \lambda\, diag(G)\right]^{-1} \nabla f\Big|_{\bar{x}_t}$$

(G = Gauss-Newton approximation of H)

This is the standard algorithm for unconstrained, local optimization of differentiable low-dimensional functions.

Why low-dimensional? Hessian is n x n matrix, inverting it is $0(n^3)$ Too expensive for large n!

Gradient Descent Methods

To reduce this cost to $0(n^2)$ per iteration, quasi-Newton algorithms directly update a „well-behaved" approximation of H^{-1}. E.g. Broyden-Goldfarb-Fletcher-Shanno (BFGS) algorithm.

Conjugate gradient methods manage to reduce the cost to 0(n) per iteration by using a smart way to implicitly accumulate information about H such that a quadratic function is minimized exactly in n iterations. Standard method for very large systems.