Optimization: Quick reference

vishvAs vAsuki

May 2, 2012

Contents

Part I

Themes

Aka Mathematical programming.
Notation: Vector space V, field F.

1 Solver design

Make fast solvers for specific classes of problems.
Make specification frameworks (see later section).

2 Applications of Constrained optimization

Convex optimization problems found widely in nature.
Untrained intuition is not very good at recognizing and formulating proper optimization problems: easy problems appear hard.

Part II

Problem structure

3 The problem

$\min f_0(x) : V \to F$ subject to constraints $f_i(x) \leq b_i$. Vector x is optimization variable of d dimensions. f_0 is optimization fn. b_i are limits/ bounds for constraints.

3.1 Standard (primal) form

$\min_x f_0(x) : \{f_i(x) \leq 0\}, \{h_i(x) = 0\}$: all are $V \to F$ functionals.

3.1.1 Explicit constraints

f_i and h_i are explicit constraints. Problem is unconstrained if it has no explicit constraints.

3.1.2 Implicit constraints

$x \in D = \cap dom(f_i) \cap dom(h_i)$.

3.1.3 Optimal value

$p^* = \inf \{f_0(x) : f_i(x) \leq 0, h_j(x) \leq 0 \forall i, j\}$. $p^* = \infty$ if problem is infeasible, or no x satisfies constraints. $p^* = -\infty$ if it is unbounded below.

3.2 Equivalent formulations

3.2.1 Epigraph formulation

$\min_{x,t} t : f_0(x) \leq t, \{f_i(x) \leq 0\}, \{h_i(x) = 0\}$: all are $V \to F$ functionals.

3.2.2 Linear equiality constraints

Got $\min_x f_0(x) : \{f_i(x) \leq 0\}, Ax - b = 0$. Take H which spans N(A), particular solution to Ax -b =0, x_0. Then, can rewrite as: $\min_v f_0(Hv + x_0) : \{f_i(Hv + x_0) \leq 0\}$.

3.2.3 Augmented/ Slack form

Replace inequality constraints $\{f_i(x) \leq 0\}$ with $f_i(x) + s_i \leq 0; s_i \geq 0$.

3.3 Conic (in)equalities constraints

$\min_x f_0(x) : \{f_i(x) \preceq_{K_i} 0\}, \{h_i(x) = 0\}$. Constraints are specified using conic (in)equalities.

3.4 Special cases

3.4.1 Feasibility problem

$\min 5.5$: $h(x) = 0, f(x) \leq 0$.

3.5 Perturbed problem

Replace constraints $f(x) \leq 0, h(x) = 0$ with $f(x) \leq u, h(x) = v$ to get problems parametrized by u, v; Denote optimum by $p^*(u, v)$. Now can study sensitivity of problem to perturbations.
Large u_i implies loosening constraint f_i; similarly constraints can be tightened. How will the optimum change in response?

4 The solution

4.1 Feasible region

Feasible region D satisfies all constraints.

4.1.1 Active, inactive constraints at x

Active constraints: $\{h_i(x) = 0\}$, and $\{f_i : f_i(x) = 0\}$ at x. Inactive constraints: $f_i(x) < 0$ at x.

4.1.1.1 Strict feasibility

x is strictly feasible if $x \in int(D)$: ie all ineqalities $f_i(x) < 0$ hold strictly.

4.1.2 Active constraints and their gradient

4.1.2.1 Constraint surface, gradient

The level set $h_i(x) = 0$ is a d-1 dimensional (possibly closed) surface. For any x in this constraint surface, $\nabla h_i(x) \perp$ the tangent to $\{x : h_i(x) = 0\}$.

4.1.2.2 Direction of the gradient

Consider level-set $f_i(x) = 0$. $\nabla f_i(x)$ will be oriented towards increasing $f_i(x)$, that is, away from the interior of $\{x : f_i(x) \leq 0\}$. See derivatives of functionals section in vector spaces survey.

4.1.2.3 Intersection of constraint surfaces

So, for any
$x : h_i(x) = 0 \forall i$, $\sum l_i \nabla h_i(x) \perp$ the tangent to $\{x : h_i(x) = 0 \forall i\}$.
Want $\min f_0(x)$ in this contour (constraint surface); or find $\cap$ of constraint surface with the smallest contour of f_0.

4.2 Optimality criteria

4.2.1 Optimal x

x^* is optimal if $f_0(x^*) = p^*$. Similarly, locally optimal x is also defined: optimality when x constrained to some small ball in feasible set.

4.2.2 Connection with gradient

x optimal iff it is feasible and $\nabla f_0(x)^T (y - x) \geq 0 \forall y$ feasible.

4.2.3 Unconstrained fn

4.2.3.1 Differentiable f

If f differentiable, x is a local minimum only if, $\forall y \in N_\epsilon(x) : \nabla_{(y-x)} f(x) = \langle (y - x), \nabla f(x) \rangle \geq 0$. Similar case for local maxima.
So, x is a local extreme value only if $\nabla f(x) = 0$: else, could move a little bit and decrease/ increase f(x). $x : \nabla f(x) = 0$ is a critical point.

Solving for x Conditions like $\nabla f(x) + l\frac{b+x}{\|b+x\|} = 0$ need some clever algebra. Example, try the change of variables $e = b + x$ to simplify the above.

4.2.3.2 Local extreme point

Using 2nd order expression for $f(x+u) = f(x) + u^T D^2 f(x+su)u$; so if $D^2 f(x) \succeq 0$, x is minimum; if $D^2 f(x) \preceq 0$, x is a maximum; else x is a saddle point.

4.2.3.3 Extension to convex non-differentiable f

Take subdifferential ∂f; x is an extreme point only if $0 \in \partial f$, as it implies that $f(x+d) \geq f(x)$.

4.2.4 Gradient of objective, the active constraint surface

Suppose that optimal x occurs on the active constraint surface. What is the relationship of $\nabla f_0(x)$ with $\nabla h_i, \{\nabla f_j\}$, where the latter is the set of active constraints?
$\nabla f_0(x) \perp$ to constraint surface at optimal x: else could move short distance along contour and decrease f. In other words, if you take any curve $g : R \to R^n$ in the constraint surface, passing through x, $f_0(g(t))$ would attain a minimum when $g(t) = x$; so can apply $\nabla f_0(g(t)) = \nabla f_0(x) Dg(t) = 0$ for any such curve g; so $\nabla f_0(x) \perp$ tangent to the active constraint surface at x.
So, want to achieve $\nabla f_0(x) + \sum_i m_i \nabla h_i(x) + \sum_j l_j \nabla f_j(x) = 0$.

Extension to non-differentiable f The above reasoning and optimality condition can be generalized to non differentiable $f_i(x)$: we just need to interpret $\nabla f_i \in \partial f_i(x)$. If this condition were violated, we could still move slightly along $\nabla f_0(x)$ and reduce the objective, while staying in the feasible region.

4.2.4.1 Direction in case of active ineq constraints

If x is optimal, as $f_0(x)$ is being minimized, $\nabla f_0(x)$ - the direction of increasing f_0- points towards the interior of the constraint surface; unlike $\nabla f_i(x)$.
So, $l_j \geq 0$ for active inequality constraints. ∇f_0 is not constrained in this manner by inactive inequality constraints: So, for all inactive $f_i(x)$: can use corresponding $l_i = 0$.
As earlier, this condition and the reasoning behind it can be generalized to non-differentiable convex f_i.

4.2.4.2 General condition

$\nabla f_0(x) + \sum_i m_i \nabla h_i(x) + \sum_i l_i \nabla f_i(x) = 0$, with $l_i \geq 0$, $h(x) = 0, f(x) \leq 0$.
Equivalently, $Df_0(x) + \sum_i m_i Dh_i(x) + \sum_i l_i Df_i(x) = 0$. ***This can be used when f, h are matrix functionals, as in SDP's!***

Extension to convex non-differentiable f Take subdifferential ∂f_i; x is an extreme point in the feasible region only if the general condition holds, using the notation $\nabla f_i(x) \in \partial f_i(x)$. Reasons for this were explained elsewhere.

4.2.4.3 Optimality in special cases

If problem is constrained only by linear equalities $Ax = b$, you get the condition: $\nabla f_0(x) + A^T m = 0$.

4.2.5 Sufficiency for local optima

Consider x', a local optimum. x' is a local optimum iff the optimality condition holds. So, can enumerate feasible x' which satisfy the optimality conditions, and pick the lowest to be the global optimum!

4.2.6 Conic inequality constrained problems

Some constraint qualifications, like Slater's conditions for convex programs, guarantee strong duality. So, $\exists l^* \succeq_{K^*} 0, m^* : g(l^*, m^*) = \inf_x L(x, l^*, m^*) = f_0(x^*)$. If L is differentiable, this yields the optimality criterion: $\nabla_x L(x, l, m) = 0, h(x) = 0, f(x) \prec 0$.

4.3 General strategies

For solution strategies for particular cases, like linear, quadratic, convex, non-convex programming: see elsewhere.

4.3.1 Hardness of finding global optimum

In general, there are many local minima. Even when you have an expression for f(x), listing its minima is often not possible with analytic methods. Eg: f'(x) may be a quintic polynomial, for which there is no closed form expression for the roots.
In general, cannot find a global optimum in reasonable time: So, one must be satisfied with a local optimum or search for a long time.
Only in certain special cases, you can get arbitrarily close to the global optimum in reasonable time.

4.3.2 Bisection method: using feasability solvers

Take problem in epigraph form. $\min_{x,t} t : f_0(x) \leq t, \{f_i(x) \leq 0\}, \{h_i(x) = 0\}$. For any fixed t, this becomes a feasability problem. Can use binary search to find the lowest value of t for which the feasability problem is satisfied.

Can do multi-parameter bisection similarly, when many parameters specify the objective.

5 Constraints in the objective

5.1 Constrained and unconstrained formulations: equivalence

5.1.1 Lagrangian functional

The Lagrangian functional :=$L(x, l, m) = f_0(x) + \sum m_i h_i(x) + \sum l_j f_j(x)$.

5.1.1.1 Definition motivations

you can rewrite the constrained optimization problem in the standard form as an unconstrained optimization problem. Also, you can define the dual problem.

5.1.2 Unconstrained program from constrained problem

As seen while considering the geometry of the problem, optimality criterion: $\nabla f_0 + \sum l_i' \nabla f_i + \sum m_i' \nabla h_i = 0$ for certain $l' \geq 0, m'$, where ∇f_i are represent subgradients in the case of convex non-differentiable f_i.
So, take $L(x, l', m') = f_0(x) + \sum m_i' h_i(x) + \sum l_j' f_j(x)$ for this l', m'; see that the optimality criterion is satisfied when $\nabla_x L(x, l', m') = 0$.
So, there exists an equivalent unconstrained problem $\min_x L(x, l', m')$ for every constrained optimization problem.
This analysis holds for conic (in)equality constrained problems too.

5.1.2.1 Objective - constraint tradeoff

$L(x, l, m)$ looks like the scalarized form of a vector optimization problem, with l and m representing the tradeoff between the various parts of the objective.

5.1.3 Lagrangian multipliers and tightness of constraints

l, m are called lagrangian multipliers.
Suppose that $f_i(x) \leq 0$ is changed to $f_i(x) \leq 5$: f_i now allows a greater degree of freedom, so minimizing f_i is now a less critical part of the objective. Similarly, Consider any inactive constraint f_i: the corresponding multiplier will be 0.
Also explains why $l_i \geq 0$ should hold for the problem $\min_x L(x, l, m)$ to make sense: higher values of f_i should be penalized.

5.1.4 Constrained program from unconstrained problem

Consider the problem $\min_x L(x, l, m)$ for $l \geq 0$. Considering the continuous relationship $c()$ between l_i and the tightness of constraints $f_i(x) \leq c(l_i)$, we

can conclude that there is an constrained problem equivalent for every unconstrained problem.

6 Dual problem

6.1 Dual functional

6.1.1 Definition

Take $g(m,l) = \inf_x L(x,m,l)$. For convex L(), do this by setting $\nabla_x L(x,m,l) = 0$.
g is always concave: inf of linear functions in m and l.

6.1.1.1 Uniqueness

Observe that optimization problems differing only in RHS of equality and inequality constraints have distinct lagrangian functionals, and thence distinct dual functionals.

6.1.2 Connection with conjugate of the objective

6.1.2.1 Linear constraints case

Consider $\min f_0(x) : Ax \leq b, Cx = d$. $g(m,l) = -f_0^*(-A^T l - C^T m) - b^T l - d^T m$.
This holds in many other cases too. So, simplifies derivation of dual if conjugate is known. Also, simplifies derivation of the conjugate if the dual is known.

6.1.3 As intercepts of supporting hyperplane to an image of the domain

Consider $G = \{(f(x), f_0(x)) \forall x \in dom(f_0)\}$. Then, $g(l,m) = \inf_{(u,t)\in G}(t + l^T u) = \inf_x L(x,l,m)$. $t + l^T u = k$, for fixed k is a hyperplane. So, for fixed l, m: the hyperplane $l^T u + t = g(m,l)$ is the supporting hyperplane for G; and g(m, l) is the intercept of this hyperplane on the f_0 axis. ***Interpretation fails if hyperplane is vertical!*** This is used in proof of slater's condition.
As we are interested in $f(x) < 0$, we consider only supporting hyperplanes which support G in that half-space. The convex dual problem tries to find the top intercept of all hyperplanes.
This view of dual variables is useful in proving strong duality from constraint qualifications like $x \in relint(F)$, where F is feasible set.

6.1.3.1 Epigraph view

Can consider $G' = \{(u,v) : \exists x \in dom(f_0) : f(x) \leq u, f_0(x) \leq v\}$; so $G \subseteq G'$.

6.1.4 For perturbed problem

$g'(m,l) = g(m,l) - u^T l - m^T v$.

6.1.5 Lower bounds on solution to the primal

$g(m,l) \leq L(x,m,l) \leq f_0(x)$ for all dual feasible m and l and primal feasible x; as
$\sum m_i h_i(x) = 0; \sum l_j f_j(x) \leq 0$: as $l \geq 0, f(x) \leq 0$.
So, as x^* is primal feasible, $\forall l \geq 0$: $g(m,l) \leq f_0(x^*) = p^*$.
Good way of showing if the problem is infeasible: $d^* = \infty$.
Applies to the case of conic inequalities: just use the fact that $l \succeq_{K^*} 0$ to see $l^T f_i(x) < 0$.

6.2 Convex Dual problem

$\max g(m,l) : l \geq 0$. Often, implicit constraint $m, l \in dom(g)$ made explicit.
Generalizable to primals constrained by conic inequalities: just use $l \succeq_{K^*} 0$, using dual of the cone K used to specify conic inequality constraints f().
Gives information about the sensitivity of the solution to perturbations in the input.
Solution notation: m^*, l^*, optimal value: d^*.

6.2.1 Picking the right primal

Equivalent formulations of the primal can lead to very different duals. Some duals can be hard to analyze, or useless (eg: constant); so reformulation of the primal (eg: by making implicit constraints explicit or vice versa) is a good trick.

6.2.2 As finding top intercept of supporting hyperplane

See dual function section.

6.3 Duality gap b/w primal, dual solutions

$p^* - g(l^*, m^*)$ is the duality gap. As shown while considering the properties of the dual function, this quantity is non-negative.

6.3.1 Strong duality

When this is 0, strong duality holds: Eg: Many cases in Convex optimization. Strong duality can also hold for non-convex problems.

6.3.2 Strong duality: optimality conditions

6.3.2.1 Optimality conditions using strong duality

Aka KKT conditions. Necessary conditions. Also sufficient when solving convex optimization: Certificate of optimality.

Primal and dual feasability Primal feasibility: $f(x^*) \leq 0, h(x^*) = 0$.
Dual feasiblity: $l^* \geq 0$.

Complimentary slackness $\forall j : l_j^* f_j(x^*) = 0$. Pf: $f_0(x^*) = g(l^*, m^*) \leq f_0(x^*) + \sum m_i^* h_i(x^*) + \sum l_j^* f_j(x^*) \leq f_0(x^*)$. So, as $l \geq 0, f(x) \leq 0$:
$\sum l_j^* f_j(x^*) = 0;\ \forall j : l_j^* f_j(x^*) = 0$.
If $l_i^* > 0, f_i(x^*) = 0$, if $f_i(x^*) < 0, l_i^* = 0$: one of these is 0, hence the name.
At this point, $f_0(x^*) = L(x^*, l^*, m^*) = g(l^*, m^*)$.

Optimality/ stationarity As $p^* = g(l^*, m^*)$, $x^* = \arg\min_x L(x, l^*, m^*)$: we have $\nabla_{x^*} L(x, l^*, m^*) = 0$. See geometric view of relationship between gradients of f, h, f_0 at optimal x for why this should hold for some l, m in general, when some nice conditions are met. With this condition, we are identifying l, m as being none other than the solution to the dual problem.
This can be extended to the case where $L(x, l^*, m^*)$, that is f_0 or f or h are not differentiable: in such a case, we just use some sub-gradients from the corresponding subdifferential set instead of gradients while writing the conditions.

Extension to nondifferentiable convex f Applying the optimality criterion for convex non-differentiable functions we see that the same optimality criterion applies to problems involving non-differentiable convex constraint functionals: one just needs to interpret $\nabla f_i(x) \in \partial f_i(x)$.

6.3.2.2 Primal dual solvers

They try to solve both the primal and dual problems simultaneously - maybe one of them gets solved quickly!

6.3.3 Strong duality: Meaning of dual variables

6.3.3.1 As weights in lagrangian form

Take problem in standard form, get dual function $g(l, m) = \min_x L(x, l, m)$; from strong duality, get: $\max_{l,m} g(l, m) = \max_{l,m} \min_x L(x, l, m) = p^*$. Note that $L(x, l, m) = f_0(x) + \sum_i l_i(f_i(x) - u_i) + \sum_i m_i(h_i(x) - v_i)$ if constraints are of the form $f(x) \leq u, h(x) = v$: the constants u and v are included.
This gives a way to get the 'constraints in the objective' form of the optimization problem, by solving the hard-to-solve problem of finding the saddle-point corresponding to $\max_{l,m} \min_x L(x, l, m)$ in order to get l^*, m^*.

6.3.3.2 Sensitivity analysis

Using weak duality, for perturbed problem described earlier, get:
$p^*(u,v) \geq g(l^*,v^*) - l^T u - m^T v = p^*(0,0) - l^{*T} u - m^{*T} v$, using strong duality.
So, for +ve l_i, large -ve perturbations in u_i cause p^* to increase.
Also if p^* differentiable: $\nabla_u p^*(u,v) = -l, \nabla_v p^*(u,v) = -m$.

6.3.4 Constraint qualifications to ensure strong duality

See Slater conditions in convex optimization section. Others exist. Can derive some from the geometric/ supporting hyperplanes view of the dual problem.

7 Vector optimization

The objective function is a vector to vector function. Comparison among vectors could be wrt any pointed cone.
Objective often written as $\min(f_{01}(x), f_{02}(x)..)$.

7.1 Multicriterion optimization

Special case where inequalities are wrt the non-negative orthant cone.

7.2 Solutions

7.2.1 Optimality

Feasible x is pareto optimal if $f_0(x)$ is the minimal value of feasible region. If $f_0(x)$ is the minimum value, then x is optimal.

7.2.2 Scalarization

Replace the objective with $l^T f_0(x)$: defines a set of trade-offs among the various measures of goodness which form the vector f_0.

7.2.3 Visualiztion: tradeoff curve

Visualize the image of the feasible region in the range of $f_0(x)$, where the axes are various dimensions of the image of f_0.
Pareto optimal points lie in the lower border of this image; for them, no dimension of $f_0(x)$ can be reduced without hurting other dimensions of $f_0(x)$: these are the minimal points of this set.
These correspond to different scalarizations of f_0. The endpoints correspond to the case where 0 weight is assigned to some dimension(s) of $f_0(x)$.

This curve is actually helpful for the enduser to pick a good scalarization.

8 Prove properties of solution

8.1 Motivation, Notation

Suppose that one wants to solve a problem or some special case of it, Q. One often wants to claim that a solution to Q can be obtained by looking at the solutions of the optimization problem P which is often easier to solve.
One then needs to be able to rigorously show that the solution to P is also the solution to Q.

8.1.1 Example

In compressive sensing,
$Q := \arg\min \|x\|_0 : Ax = b$ and $P := \arg\min \|x\|_1 : Ax = b$.

8.1.1.1 Uniqueness

If one can either show that there is a unique solution to P, or if we can show that one can efficiently search the small number of solutions to P in order to find a solution to Q, we have now have an efficient algorithm to solve Q.

8.2 General proof technique

Specify the 'optimality' properties $C_P(x)$ (certificate/ witness) which are satisfied exactly by solutions to P. Do the same for Q and specify $C_Q(x)$. The idea is to show that $(\exists x : C_Q(x) \wedge C_P(x))$.
Construct a candidate solution x to P which simultaneously satisfies some subset of $C_Q(x) \cup C_P(x)$. Show that this candidate then satisfies the remaining properties in $C_P(x) \cup C_Q(x)$.

8.2.1 Possible computational intractability

Even though construction of the solution to P and Q described above is a valid proof technique, it often does not imply that we have an efficient algorithm to solve Q using just the conditions C_Q and C_P. This is because, to completely specify C_Q, one needs to know the solution to Q!
Eg: In the compressive sensing example, for $C_Q(x)$ to be completely specified, one needs to know not just $\|x\|_0$, but also the coordinates $\{i : x_i \neq 0\}$.

Part III

Problem formulation, solution

9 Formulating the problem

9.1 Focus on easily solved forms

Modellers of natural phenomena and engineers formulate optimization problems. They know what classes of problems are easy to solve. So, they often draw on a mental library of functions, composition rules to create, for example a convex program.

9.2 Dealing with strict inequality constraints

Replace with non-strict inequalities.

9.3 Use equivalent formulations (for convexity?)

You can always replace $f_0(x)$ or $f_i(x)$
with a $p(f_i(x))$ if p is monotonically increasing with f_i, and if you can translate between $\arg\min f_0(x)$ and $\arg\min p(y)$ or between $f_i(x) \leq 0$ and $p(y) \leq c$.
Some formulations are more easily solved than others. In fact, it can turn a non convex problem to an equivalent convex problem.
Eg: $\min \|x\|_2 + t \|x\|_1 \equiv \min \|x\|_2^2 + t \|x\|_1$.
Also, can use the dual if strong duality holds.

10 Problem parameters

10.1 Not variables

These are different from variables. For example, in $\min \|Ax - b\|^2 + l \|x\|^2$, while x is a variable, l is a parameter to the optimization problem.

10.2 Picking the best parameter

With problem parameters unspecified,
you have a variety of optimization probems. Then, pick the best problem to solve, ie pick the best parameter.

10.2.1 Qualitative judgement

Set various values for the problem parameters, solve them, and pick the parameter whose solution appears best, from the perspective of the application domain.

10.2.2 Theoretical constraints

Theoretical analysis of the problem domain sometimes connect the parameter values with desirable traits in the solutions.

11 Specification framework engineering

Specification frameworks abstract away from the
mathematical details, methodological details (which underlying solver to call), make specification easy. They lead to wide application.
It is a good idea to mimic the way people go about naturally formulating optimization problems for a certain class: eg by drawing on a library of functions known to be convex and rules of composition. Eg: cvx for disciplined convex programming.

Part IV

Linear programming

12 The problem

Minimize linear/ affine function over a polyhedron: ie subject to affine/ linear constraint functions.

12.1 Canonical form

$\vec{x} \in R^d$. $\min c^T x : Ax \leq b$.

12.1.1 Nonnegative optimization form

$\min c^T x' : Ax' \leq b$, with $x' = \begin{bmatrix} x^+ \\ x^- \end{bmatrix}$; new constraints: $x' \geq 0$, $x^+ - x^- = 0$.

12.1.2 Equality constrained form

Writable as $\min c^T x : Ax = b$;x≥ 0, using slack variables.

12.2 Constraints and the polyhedron

Every constraint is a halfspace. Intersection of halfspaces is a polyhedron; this is the **feasable region**.

12.3 The solution

Want to find a point in the feasible region making the least angle/ inner product with f.

12.3.1 Vertex in feasible region

Consider minimizing $f^T x$ over a line segment (a, b): for any point in this segment, you have: $f^T(ta+(1-t)b) \leq \min\left\{f^T a, f^T b\right\}$; so sufficient to consider end points while trying to minimize $f^T x$ over any line segment in the feasible region. By induction, sufficient to consider vertices of the feasible polyhedron.

12.3.2 Pathological cases

$\min x$, A is empty; and minx: $x \geq 50$: often ruled out by added resource constraints: x allowed to be only so small.

13 Related problems

13.1 Linear fractional program

Minimize linear fractional function over a polyhedron. Actually quasi-convex programming, but reducible to LP. Can rewrite $\min t : \frac{a^T x+b}{c^T x+d} \leq t$ using linear constraints.
Similar is the case with generalized linear fractional program:
$\min t : (\max_i \frac{a_i^T x+b_i}{c_i^T x+d_i}) \leq t$.

13.2 1, infty norm approximations

$\min_x \|Ax-b\|_\infty \equiv \min c : Ax-b \leq c1, Ax-b \geq -c1$.
$\min_x \|Ax-b\|_1 \equiv \min c : Ax-b = t_+ + t_-; t_- \leq 0; t_+ \geq 0; 1^T t_+ - 1^T t_- \leq c$.
This tends to yield sparse solutions: consider the functioning of least angles regression to see why.
Similarly, can replace 1, ∞ norm constraints in any LP.

13.3 Robust linear programming

Often, there is uncertainty in constraints like $a_i^T x \leq b_i$.

13.3.1 Deterministic model

x must be feasible $\forall a_i \in S$. If S is an ellipse, this can be handled using SOCP.

13.3.2 Stochastic model

Feasible x must satisfy $Pr(a_i^T x \leq b_i) \geq t$. If distribution is Gaussian, this can be handled using SOCP.

14 LP Algorithms

14.1 Exhaustive search alg

Find intersection of every set of d hyperplanes (constraints); see if it satisfies other constraints: $O(\binom{n}{d})$.

14.2 Simplex method

(Dantzig). Theoretically exponential, highly efficient in practice.

14.3 Interior point projective method

(Karmarkar) LP solvable in time
$poly(n, d)$.

Part V

Convex Quadratic programming

For ideas about its importance - both in solving general convex optimization problems and in other domains, look at the 'Least squares' chapter.

15 Linear Constrained QP

Aka Quadratic programming. Minimize convex quadratic functional over a polyhedron (see vector spaces survey). Generalizes linear programming, special case of QCQP.

15.1 Visualization

If optimal value occurs when a constraint is active: Think of contours of the objective hitting an edge of the polyhedron. Slightly harder than in the LP case, where the minimum always occured in a vertex.

15.2 Dual problem

This is also a quadratic program; strong duality always holds! [**Check**]

16 Quadratically constrained QP (QCQP)

Minimize convex quadratic functional, subject to convex quadratic constraints: so all equality constraints are specified by linear inequalities. Feasible set is an intersection of hyperellipes and a polyhedron.

17 Least squared error problem

17.1 Problem

Aka least squares.

17.1.1 Unregularized problem

$\min \|Aw - b\|^2$, $A \in R^{m \times n}, m > n, rank(A) = m$.

17.1.1.1 For the weighted least squares

More weight to one observation or correlated observations:
Use Weighted inner product to find projection.
$WAx = Wb; \min \|WAx - Wb\| = \min \|Ax - b\|_M$ for $M = W^*W$.

17.1.1.2 Solution

See numerical analysis survey for geometry and solution.

17.1.2 The regularized problem

Aka weight decay.
Want to balance love of sparsity or smallness with desire for a good fit. Pick a regularizer which is small for good w, and large for bad w.
Take $\min f(w) + l \|w\|_n$. As n decreases, feasible set decreases: consider progression of 1 balls from $\|.\|_\infty$ to $\|\|_0$.

17.2 Application

17.2.1 Convex Optimization

A common technique of solving convex optimization problems is to approximate the objective with a quadratic function and then solve it.

17.2.2 Other domains

See regression problem in statistics survey for motivation. Same as finding maximum likelihood solution while fitting linear model with Gaussian noise.
Interpolation of functions linear in parameters is also a specific case: see numerical analysis survey.

17.2.3 Regularized problems

If solving a regression problem, this amounts to the least squares solution, when a prior probability distribution is imposed on w to favor smaller w values.
w could be the coefficients of polynomial used to fit data in A, b. To avoid overfitting, we want w to be small.

17.2.3.1 Penalties and priors

Different regularizers (usually norms) correspond to different priors on w. Shape of the prior distribution looks like the unit spheres corresponding to the norms used: sphere, diamond etc..
$\|\|\|_q$ penalty for $q > 2$ usually not worth the effort. Sometimes, $\|\|\|_{1.x}$ penalty used as compromize between lasso and quadratic regularizers.

17.3 Standard formulations

17.3.1 Normalizing columns of A

By solving $A'w' = b$, can solve $ADD^{-1}w \approx b$ using diagonal matrix D.

17.4 Quadratic regularizer

Aka ridge regression.
$\min f(w) + lw^T I'w = \min \|Aw - b\|_2^2 + lw^T I'w$ instead, to control size of w too.
This objective is the lagrangian fn corresponding to the problem of finding $\min f(w) = \min \|Aw - b\|_2^2$ subject to $w^T I'w < c$.

17.4.1 The solution form

$(A^T A + lI)\hat{w} = A^T b$: by setting $\nabla(f(w) + l\|w\|_2^2) = 0$. Take $A = U\Sigma V^*$, get $X\hat{w} = U\Sigma(\Sigma^2 + lI)^{-1}\Sigma U^T y = UDU^T y$, where $D_{i,i} = \sigma_i' = \frac{\sigma_i^2}{\sigma_i^2 + l}$.
So, $X\hat{w} = \sum \sigma_i' u_i^T u_i y$; observe action of $l \in [0, \infty]$ in σ_i': the shrinkage parameter; or the prior distribution on w.

17.4.2 The geometry

Consider hypersphere $\|w\|_2^2 < c$ and the paraboloid $f(w)$. The objective is to find w within the hypersphere, having the minimum $f(w)$ value. Visualize by taking contour map of $f(w)$: successively larger ellipsoids. optimal w is where the minimal contour touches the hypersphere, when the center of $f(w)$ lies outside the hypersphere.
Compare with geometry of $\min f(w)$ problem: see linear alg survey.

17.4.3 Effect

Shrinks components of w towards each other in magnitude. Also takes care of correlated features.
Does not penalize small w_i, so does not lead to sparsity.

17.5 l2 and linf regularizers

17.5.1 l2 regularizer

17.5.1.1 Problem

Solve $\min_w \|Aw - b\|^2 + \lambda \|w\|_2$, with $\lambda > 0$. An alternate formulation is $\min_w w^T H w + v^T w + \lambda \|w\|_2$, with $H \succeq 0$.

17.5.1.2 Solution

The obtimality condition involves taking the subdifferential of $\|w\|_2$. For diagonal H, this has a closed form solution. The solution technique involves some nifty variable substitution.

17.5.1.3 Relation with Quadratic regularizer

Rewriting as $\min_w \|Aw - b\|^2 : \|w\|_2 < C(\lambda)$; we see that this is equivalent to using the quadratic regularizer and solving: $\min_w \|Aw - b\|^2 : \|w\|_2^2 < C(\lambda)^2$. So, the tradeoff curve, is for practical purposes, identical.
The latter formulation has a simpler solution as subgradient calculations are not involved in specifying the optimality conditions for analytically finding a solution.
But, sometimes we are given the problem where an l_2 regularizer is used. Such problems occur in doing block coordinate descent to solve multi-task lasso, for example, while finding the search direction using a quadratic approximation of the objective. In such cases, we can either solve the specified problem directly, or we can reformulate it using the quadratic regularizer; the former is quite simple.

17.5.2 lInf regularizer

Here, we solve $\min_w \|Aw - b\|^2 + \lambda \|w\|_\infty$. Optimality condition involves taking the subdifferential of $\|w\|_\infty$. A closed form solution can be derived by using proof by cases: eg: see a paper on muti-task learning by Han Liu.

17.6 Forward stepwise regression for sparsity

Trying to solve Aw = y. Take F = active set of features used in approximating y. Start with $F = \phi$, w = 0. At step t, find residue $r = y - Aw_t$, find feature i most correlated with r; set w_i to this projection; continue. ***Observe how the residue changes at each step!***
This is a greedy algorithm, so suboptimal. But note that this is not the same as simply finding the optimal w" without any sparsity constraint, and then dropping some small elements.

17.6.1 Forward stagewise regression

A non greedy modification to forward stepwise. Find feature set B with maximum correlation with residue; for each such i: keep increasing w_B until you find another feature(s) with equal correlation with the residue; then add these feature(s) to B; repeat.
At any time, B is the set of features which form the ***least angle*** with the residue. The coefficients increase such that Aw_B increases exactly along the projection of the residue on the hyperspace spanned by features in B.
Also, in case at some time, a feature correlated with other some feature in B gets added to B, the coefficient of B.

17.6.2 Least Angle Regression

A modification of the Forward stagewise regression, so that w_B is increased at one shot, rather than gradually.

17.6.3 Lasso solving modification

Losso just specifies an objective, which may be achieved in many ways, including using forward regression.
Can turn LAR to Lasso solver: Whenever w_i for some i in B hits 0, drop it out of B.

17.6.3.1 Reason for the fix

Compare conditions you get on lasso solution by setting gradient to 0 with conditions for LAR at any time: Correlation of residue with B with features in B is equal and correlation with other features is lower. They are identical as long as $sgn(w_j)$ = sign of correlation of residue with feature j.

17.7 0 norm regularizer: Compressed sensing

17.7.1 Problem scenario

X is short and fat, Xw = y has many solutions for w, want to find w with $\|w\|_0 \leq s$, $Xw \approx y$. If X were tall and thin, the solution of a similar problem is easy: solve min $\|Xw - y\|^2$ and pick the s most important components of w. Consider the equivalent problem, where we assume columns of X are normalized.

17.7.1.1 Finding support: Combinatorial hardness

The difficulty comes from finding the support (non-zero coordinates) of w. Once support of w is found, it is easy to find the optimal linear combination of these components to get close to y: just solve min $\|X'w' = y\|$, where $m \times s$ X' is derived from X by dropping some columns.
There are many possible ways to form w with $\|w\|_0 \leq s$.

17.7.2 Finding support: Target optimization problems

Maybe want to limit w to certain number of non-zeros. Solve min $f(w)+l\|w\|_0$, where $f(w) = \|Xw - y\|_2^2$.
This is same as $\min f(w) : \|w\|_0 \leq s$. Feasible set is not convex: consider epigraph of $\|w\|_0 \leq 1$ for example.
A stricter version of the problem is: $\min \|w\|_0 : Xw = y$, as this does not allow $Xw \approx y$.

17.7.3 Solution using 1 norm minimization

Just solve the linear program $\min \|w\|_1 : Xw = y$.
Arrival at sparse solution is guaranteed when some restricted isometry and incoherence properties hold for X.

17.7.3.1 Restricted isometry constant for s

$1 - \delta_s \leq \frac{\|Xw\|^2}{\|w\|^2} \leq 1 + \delta_s, \forall w : \|w\|_0 \leq s$.

17.7.3.2 Incoherence/ almost orthogonality

$|\langle Xw, Xw' \rangle| \leq T_{s,s'}$ for w, w' such that $supp(w) \cap supp(s') = \emptyset, \|w\|_0 \leq s, \|w'\|_0 \leq s'$.

17.8 1 norm regularizer: Lasso

$\min f(w) + l\|w\|_1 = \min \|Aw - b\|_2^2 + l\|w\|_1$. Allegedly aka least absolute shrinkage and selection operator.

Same as minimizing f(w) subject to $\|w\|_1 < c$. As you increase c, you get less and less sparse solutions. When $c \geq \|\hat{w}\|_1$ where $\hat{w}$ is the unregularized least squares solution, you get least squares solution.
Want to get optimality condition $\nabla(f(w) + l\|w\|_1) = 0$, but $\|w\|_1$ not differentiable at all points; so take $l\|w\|_1 = l\sum_i sgn(w_i)w_i$. Assume that $sgn(w_i')$ around the solution w' is known.
Get conditions useful in Lasso solving modification of Least angles regression: Let $B = o : w_i' \neq 0$, the basis/ support set. $\forall j \in B : a_j^T(y - Aw) = \lambda sgn(w_j)$, $\forall j \notin B : a_j^T(y - Aw) = \lambda * sgn(w_j) \leq |\lambda|$. If A's columns have norm 1, get geometric meaning for λ: an upper bound for correlation of the residue with any feature!

17.8.1 Importance

Often yields sparse solutions: from geometry.
$\|w\|_q$ for $q \in (0, 1]$ would also yield sparse solutions: see geometric interpretation for justification. These correspond to imposing the constraint $\|w\|_q \leq c$, which corresponds to the feasible region being non-convex. 1 norm is the least q which yields besides leading to a sparse solution, is also convex.
For importance of finding sparse models, see statistics survey.

17.8.2 The geometry

$\|w\|_1 < c$ is hyper-rhombus, f(w) is a paraboloid, whose each level set is an ellipsoid. In general, w for min f(w) will lie outside the hyper-rhombus. So, the optimal w: the point where the edge of the hyper-rhombus meets the t-level set: f(w) = t, for the smallest t. Usually this happens where 2 edges meet, or where many w_i are 0: visualize in 2D; hence sparsity.
Compare with ridge regression.

Part VI

Convex optimization

18 The problem

18.1 Importance, efficient solvability

Superset of LP. Many problems in nature are convex optimization problems. Many non-convex problems have convex equivalents : see section on modelling/ specifying optimization problems.
(Nesterov, Nemorinsky) For any convex optimization problem, there exists self concordant barrier functions; so interior point methods and barrier methods

can be made applicable: This is a non constructive proof. So, there exists a polynomial time algorithm for every convex optimization problem, but you will find it with certainty only if you can find these self concordant barrier functions.

18.2 Standard form

$\min f_0(x) : f(x) \leq 0, Ax = b$.
All $f_i(x)$ are convex.

18.3 Convexity of feasible region X

Optimization fn is convex, constraint sets are convex sets. So, X is convex.

18.3.1 Geometry

$\nabla f_0(x^*)$, if $\neq 0$, defines supporting hyperplane for X at x^*. Imagine contours of f_0, with minimum outside X, colliding with X at some point x^*.

18.4 Identifying convex opt problems

18.4.1 Check Convexity of fesible regions

Maybe compare with known convex sets (see vector spaces survey).

18.4.1.1 Any equality constraints should be linear.

Eg: $x : h_i(x) = b$ not a convex set when $h_i(x)$ is a quadratic fn. These could be replaced by 2 inequality constraints in the standard form; both of which would need to represent convex sets.

18.5 Properties

18.5.1 Local optimum is the global optimum

By contradiction: Take radius R local optimum x; suppose there were y more than R away from x with $f_0(y) < f_0(x)$; then, can conjure a point z, a convex combination of y, x which is less than R away from x with $f_0(z) < f_0(x)$.

18.5.2 Lagrangian dual functional

Can easily get the Lagrangian dual functional: $g(l, m) = \inf_x L(x, l, m)$ by setting $\nabla_x L() = 0$ and eliminating x from L(x, l, m).

18.5.3 Certificate of optimality with strong duality

Aka KKT certificate. If feasible (x', l', m') satisfy KKT conditions, they are optimal. THis is a good way of solving convex optimization problem.
Pf: From complimentary slackness: $f_0(x') = L(x', l', m')$; from the optimality condition and convexity of f, f_0:
$g(l', m') = \inf_x L(x, l', m') = L(x', l', m') = f_0(x')$. So, x' is optimal.

18.5.4 Bound norm of solution

All sublevelsets of f_0 are convex. So, find a ball $\{x : f_0(x) \leq B > p^*\}$ which includes 0; then we can say that $\|x\| \leq 2B$.
Eg: This technique was used in bounding the deviation of the solution of a l1 regularized logistic regression problem from the actual parameters defining an ising model by pradeep etal.

18.6 Dual problem

Strong Duality usually holds. 'Constraint qualifications' can tell you whether strong duality holds.

18.6.1 Strict feasibility Constraint qualification

(Slater) If there exists some strictly (primal) feasible x, strong duality holds. ***This is very general: some strictly feasible*** x ***should exist!*** Generalizable to convex programs specificed using conic inequalities.
Actually, can relax constraint qualification to: $\exists x$:x$\in$ $relint(D)$ wrt affine plain $Ax - b = 0$.
Applies to convex conic inequality constraints too! Also, this is a way to get strong duality without using the global optimality criteria perspective.

18.6.1.1 From supporting hyperplanes view

Consider the supporting hyperplanes view of the dual problem (see dual problem section). For convex optimization problems, convexity of f, f_0 ensures that $G = \{(u, t) | f_0(x) \leq t, f(x) \leq u\}$ is convex. So, supporting hyperplane at $(0, p^*)$ for both G and
$G' = \{(u, t) | f_0(x) \leq t, f(x) \leq 0\}$ are the same.
When the primal is strictly feasible, the supporting hyperplane at $(0, p^*)$ is non vertical; so intercept with f_0 axis is well defined; and the dual problem is

feasible. Thence, slater's condition.

19 Unconstrained problems: algorithms

19.1 Problem, algorithm framework

19.1.1 Problem, Assumptions

Problem: $min_x f(x)$, f convex.

19.1.2 Algorithm framework

Produce sequence $x^{(k)} \in dom(f), k \in Z_+$: $f(x^k) \to p^*$. Input to algorithm: starting point $x^{(0)}$.
Alternate notation: x^+ for current iterate.

19.1.2.1 As Iteratively solving gradient eqns

Can be interpreted as iterative methods
for solving optimality condition $\nabla f(x) = 0$: a set of non-linear equations.

19.1.3 Common assumptions

19.1.3.1 Assumptions about f

f is twice continuously differentiable, so dom(f) is open.
p^* is attained.

Strong convexity If f is strongly convex, $f(x) - p^* \leq (2m)^{-1} \|f(x)\|_2^2$; so RHS can be used as a stopping criterion.
Also guarantees that sublevelsets are bounded: $f(y) \geq f(x) + \nabla f(x)^T (y-x) + \frac{m}{2} \|x-y\|_2^2$; this ensures faster convergence.

19.1.3.2 Initial point: Assumptions

$x^{(0)} \in dom(f)$.
Sublevel set $S = \left\{x : f(x) \leq f(x^{(0)})\right\}$ is closed. This is hard to verify usually, except when all sublevel sets are closed $\equiv$ epigraph of f is closed. This is needed, because some methods try to draw secants in the epigraph of f(x).

19.2 Descent methods

19.2.1 Algorithm

$x^{(k+1)} = x^{(k)} + t^{(k)} \Delta x^{(k)}$ with $f(x^{(k+1)}) < f(x^{(k)})$; do this repeatedly until stopping criterion is met, in each iteration: finding step or search direction Δx, the step size/ length t.

This is guaranteed to eventually come arbitrarily close to the minimum.

19.2.1.1 Descent direction

Δx is a descent direction if $\nabla f(x)^T \Delta x < 0$.

19.2.2 Find search direction

See later sections.

19.2.2.1 Search direction from optimality conditions of approximation

Aka Newton equations in some cases.
Maybe model f(x) with $\hat{f}(x)$, to find a search direction: find the optimality conditions for minimizing $\hat{f}(x)$ or f(x) and find Δx which does this exactly or approximately.
For examples, see section on 2nd order approximation descent.

19.2.3 Line search for t

19.2.3.1 Restriction to a slice

$g(t) = f(x + t\Delta x)$ is f restricted to a slice along Δx. This is convex as f is convex.

Visualization Plot $g(t)$ vs t, try to find t which minimizes this.

19.2.3.2 Exact line search

$t = \arg\min_{t>0} g(t)$. This is usually expensive.

19.2.3.3 Backtracking

Parameters: $a \in (0, 1/2), b \in (0, 1)$; b is the shrinkage parameter.
Start with t=1 (or small enough t to guarantee that $x + t\Delta$x$\in dom(f)$), repeat $t := bt$ until stopping criterion is met: $f(x+t\Delta x) < f(x)+at\nabla f(x)^T \Delta x$. With shrinkage of t, the RHS reduces. This is aka Armijo's rule.
Can be rewritten as $f(x + t\Delta x) - f(x) < ah(t)$, where h(t) is the change in f for t, according to a linear approximation of f. This form is useful when f() is composed of differentiable and non-differentiable parts.

Guaranteed reduction in objective As Δx is a descent direction, $\nabla f(x)^T \Delta x < 0$.
So, when $f(x + t\Delta x) < f(x) + at\nabla f(x)^T \Delta x$, we have $f(x + t\Delta x) < f(x)$, so it is a step size which definitely reduces f().
Such a t will always exist: you keep decreasing t until that happens.

As secant in epigraph of g(t) Take $g(t) : R \to R$: one dim fn. Consider $\nabla_t g(t)$ at t=0. $\nabla_t g(0) = \nabla_x f(x)\Delta x \leq 0$. $g(0) + at\nabla_t g(0) = f(x) + at\nabla f(x)^T \Delta x$, when a=1, is the tangent to g(t) at t=0, and therefore to f(x). $f(x) + at\nabla f(x)^T \Delta x$, for $a < 1$ is such a secant: you are making the slope less negative.
x changes along the search direction only, but you change $t \in [0, t_0]$, to get t which is close (from below) to the intersection of a secant in the epigraph of g(t) with g(t).
As $f(x) + at\nabla f(x)^T \Delta x$ is a secant, for the stopping condition should be met for some t if Δx is indeed a descent direction. This ensures that t is such that $f(x + t\Delta x) - f(x) \approx at\nabla f(x)^T \Delta x$, the improvement achieved by using a secant.

Variants Rather than ensure that t is such that
$f(x + t\Delta x) - f(x) \approx ah(t)$, the improvement achieved by using a secant; one can use second order approximations instead to define h, rather than a linear approximation: this perspective is different from the 'secant view'.

19.2.3.4 Arbitrary choice

Using backtracking rule or doing exact search during line search guarantee reduction in the objective (and therefore convergence), but they can be expensive. An alternative can be to just use 1 as the step length.
The cost of doing so is that convergence is no longer guaranteed: it is possible for example that, taking the step 1 repeatedly, one cycles between the same pair of points between which lies the optimal point. However, in practice, convergence is usually observed.
This technique is used, for example, in
'iterateively reweighted least squares', where each step involves solving a least squares problems (which may correspond to the local 2nd order approximation).

19.3 Steepest descent

Aka Gradient descent.

19.3.1 As 1st order approximation minimization

As $f(x+v) = f(x) + \langle \nabla f(x), v \rangle$, minimize $\langle \nabla f(x), v \rangle$ - Also see geometric view section.

19.3.2 Descent direction

For a given $\|.\|$, $\Delta x = \arg\max_{v:\|v\|=1} \langle -\nabla f(x), v \rangle$.
$\langle \nabla f(x), v \rangle = -\langle \nabla f(x), -v \rangle = -\|\nabla f(x)\|_*$. ***So this is a descent direction!***

19.3.3 Stopping criterion

Use $\|\nabla x\| \leq \epsilon$ as stopping criterion.

19.3.4 Convergence

$f(x^{(k)}) - p^* \leq c^k(f(x^{(0)}) - p^*)$. [**Find proof**]

19.3.5 Geometric view

19.3.5.1 2-dim functional example

Consider contours/ level sets of $f : R^2 \to R$. The level set $\{x : f(x) = t\}$ surrounds $\{x : f(x) = t - 1\}$; etc.. At $x : f(x) = t$, the $\nabla f(x)$ is perpendicular to level set $\{x : f(x) = t\}$, pointing away from $\{x : f(x) = t - 1\}$; and so is the gradient descent direction; but it points towards $\{x : f(x) = t - 1\}$.

19.3.5.2 Goodness for circular level set case

If the level set contours are circular, then $-\nabla f(x)$ points straight towards the minimum level set. But, consider an ellipsoid. Then, $-\nabla f(x)$ is passes through a few level sets $\{x : f(x) = t' < t\}$, but it misses many smaller level sets. Eg: imagine level sets which are shaped like concentric rounded triangles.

19.3.5.3 Zig-zagness of path towards optimum

In this case, despite finding the best step size, the sequence of points produced is not a direct line towards x^*, but forms a zig-zag path.

19.3.5.4 Implications of choice of norm

If you choose a good $\|.\|$, the shape of whose unit ball approximates the shape of the level sets/ contours, Δx will be such that it points towards the minimal contour of f, while still having a large enough inner product with $-\nabla f(x)$ to guarantee that it is a descent direction. So, you can get to x^* more directly. But, if you pick a bad norm, Eg: $\|\|_2$ ***which results in a reduction to gradient descent***, the path to x^* is longer.

19.4 2nd order approximation descent

Aka Newton method (reason described below), but not Gauss-Newton method, which is a further approximation and is specific to least squares problem.

19.4.1 General search direction

Take $f_0(x + d) \approx f_0(x) + x^T \nabla f(x) + x^T H x$ for some $H \succeq 0$. If H is the 2nd order derivative, this is the Newton method described later. Often H is an easily computed approximation of $\nabla^2 f_0(x)$.

19.4.2 Search direction from Hessian

$\Delta x = -\nabla^2 f(x)^{-1} \nabla f(x)$. Minimized 2nd order approximation: $\hat{f}(x+v) = f(x) + \langle \nabla f(x), v \rangle + 2^{-1} v^T \nabla^2 f(x) v$. So, $f(x)$ approximated with a quadratic curve!
This is Aka Newton's method, as it corresponds to solving for the root of the optimality condition $\nabla f(x) = 0$.

19.4.2.1 Solving Newton equations fast

Solving the linear system of equations: $\Delta x = -\nabla^2 f(x)^{-1} \nabla f(x)$ can be expensive in general: $O(n^3)$. But, in practice, one can often exploit structure in the $\nabla^2 f(x)^{-1}$ matrix to find the (possibly approximate) search direction fast - this is very important. Can often use iterative methods to solve this.

19.4.2.2 Correctly computing gradient and Hessian

See comments from the gradient descent case.

19.4.3 Geometric view

Consider the 2-dimensional example described in the 'geometric view of steepest descent' section.

19.4.3.1 Local approximation by ellipses

Note: picking $\|x\| = (\langle x^T P x \rangle)^{1/2}$ for $P \succeq 0$ yields ellipsoid unit balls. This can be thought of taking: $\hat{f}(x+v) = f(x) + \langle \nabla f(x), v \rangle + 2^{-1} v^T P v$ and using $\inf_v \hat{f}(x+v)$ as the search direction.

19.4.3.2 2nd order approximation ellipses

Choosing $\|x\| = (\langle x^T \nabla^2 f(x) x \rangle)^{1/2}$, the local Hessian norm, results in reduction to 2nd order approximation descent. To see this, consider the 'constraints in the objective' form of $\Delta x = \arg\max_{v:\|v\|=1} \langle -\nabla f(x), v \rangle$.

19.4.4 Affine invariance

The newton method is supposed to be invariant to affine transformation of the input: can confirm by computing ∇ and ∇^2 for $f(Ax' + b)$ and $f(x)$, where $Ax' + b = x$.

19.4.5 Stopping criterion: Affine invariant

Take $\lambda(x) = (\nabla f(x)^T \nabla^2 f(x)^{-1} \nabla f(x))^{1/2} = (-\nabla f(x)^T \Delta x)$, use $\lambda(x) \leq \epsilon$.
Estimates proximity to x^*: $f(x) - \inf_y \hat{f}(y) = 2^{-1} \lambda(x)^2$: from simple substitution.
Equals length of Δx in the local Hessian norm : see steepest descent connection.

Also measures the Δx directional derivative, $\nabla f(x)^T \Delta x = -\lambda(x)^2$. So, it measures how close $\nabla f(x)$ is to being 0, as it is at x^*.
This is also affine invariant, unlike $\|x\|$. Pf: Consider x = Ax' +b. Consider $D_{x'} f(Ax' + b) = D_{Ax'+b} f(Ax' + b)A = D_x f(x)A$; thence get $\nabla_{x'} f(Ax' + b)$, and thence $\nabla^2_{x'} f(Ax' + b) = D_{x'} \nabla_{x'} f(Ax' + b)$.

19.4.6 Speed: comparison with 1st order methods

Computing the search direction makes 2nd order methods slow in general; but if some structure in $\nabla^2 f(x)$ is exploited to make this faster, it becomes very fast.
Each iteration of 1st order methods are usually much faster, but many more iterations are required.

19.4.7 Convergence: classical bounds

19.4.7.1 Assumptions

f strongly convex with constant m.
$\nabla^2 f$ is Lipschitz continuous. This measures how well f(y) can be approximated by the quadratic functional $f(x) + \nabla f(x)(y - x) + 2^{-1}(y - x)^T \nabla^2 f(x)(y - x)$.
$\exists \eta \in [0, m^2/L], \gamma > 0$.

19.4.7.2 Linear decrease phase

Aka Damped Newton Phase. $\|\nabla f(x)\| \geq \eta \implies f(x^{(k+1)}) - f(x^{(k)}) \leq -\gamma$.
Linear decrease in objective value.
This phase ends after atmost $\frac{f(x^{(0)} - p^*)}{\gamma}$ iterations.
Most iterations require backtracking search.

19.4.7.3 Quadratically convergent phase

$\left\|\nabla f(x)^{(k)}\right\| \leq \eta \implies \frac{L}{2m^2} \left\|\nabla f(x^{(k+1)})\right\| \leq (\frac{L}{2m^2} \left\|\nabla f(x^{(k)}\right\|)^2$. Quadratic decrease in gradient size.
So, for $l > k$: $\frac{L}{2m^2} \left\|\nabla f(x^{(l)})\right\| \leq 2^{-2^{(l-k)}}$.
All iterations use step size $t = 1$! [**Find proof**]
Observing these on a (maybe log) residue vs iteration and step size vs iteration plots is a good way of verifying correct implementation of gradients etc..!

19.4.7.4 Overall bounds, defects

To achieve $f(x) - p^* \leq \epsilon$: $\frac{f(x^{(0)} - p^*)}{\gamma} + \log\log(\epsilon_0/\epsilon)$ iterations needed, where $\epsilon_0 = f(m, L)$ too.
Provides qualitative insight into behaviour of 2nd order approx descent.
But, constants m, L are usually unknown: so cannot say beforehand when convergence will happen.

Bounds are not affine invariant, even though the 2nd order approx descent method is.

19.4.8 Convergence for self concordant functions

(Nesterov, Nemorinsky) Get better bounds, which don't suffer from the defects of classical analysis.

19.5 Alternating minimization

Consider f(x, y); repeat these steps:
$x := \arg\min_x f_0(x, y), y := \arg\min_y f_0(x, y)$. Guarantees that the objective is reduced in each iteration.
When minimization is done one coordinate at a time, it is called coordinate descent; if it done one coordinate-set at a time, it is called block-coordinate descent.

19.6 Diagnosing error in code

19.6.1 Incorrect gradient computation: symptoms

Often algebraic mistakes happen while computing the gradient.
To detect such a case, compare with the numerically computed gradient: See numerical analysis survey.
Or, observe that the step size found in the direction of the gradient is very small.

20 Equality constrained problems

20.1 Problem, assumptions

$\min f(x) : Ax = b$. So, optimality conditions: $\nabla f(x^*) + A^T v^* = 0$.

20.1.1 Common Assumptions

f is twice differentiable, A has full row rank: can always get an equivalent problem which satisfies this.
Optimum p^* is attained.

20.2 Solution strategies

20.2.1 Reduction to unconstrained optimization

$\min f(x) : Ax = b \equiv \min f(Hv + x')$, where $Ax' = b$, H spans null(A).

20.2.2 Search direction from optimality conditions of approximation

See 19.2.2.
For examples, see section on 2nd order approximation descent.

20.2.3 Local approximation by ellipsoid

Minimize $f(x+\Delta x) \approx \hat{f}(x+\Delta x) = f(x) + \nabla f(x)^T v + 0.5\Delta x^T P \Delta x$ subject to $A(x+\Delta x) = b$, for $P \succ 0$. From optimality conditions for minimum of this approximation, get search direction: $\left[\begin{smallmatrix} P & A^T \\ A & 0 \end{smallmatrix}\right] \left[\begin{smallmatrix} \Delta \mathrm{X} \\ w \end{smallmatrix}\right] = \left[\begin{smallmatrix} -\Delta f(x) \\ 0 \end{smallmatrix}\right]$.

20.2.4 2nd order approximation descent

Aka Newton method. Minimize $f(x+\Delta x) \approx \hat{f}(x+\Delta x) = f(x) + \nabla f(x)^T v + 0.5\Delta x^T \nabla^2 f(x) \Delta x$ subject to $A(x+\Delta x) = b$.

20.2.4.1 Search direction

From optimality conditions for minimum of this approximation, get search direction: $\left[\begin{smallmatrix} \nabla^2 f(x) & A^T \\ A & 0 \end{smallmatrix}\right] \left[\begin{smallmatrix} \Delta \mathrm{X} \\ w \end{smallmatrix}\right] = \left[\begin{smallmatrix} -\Delta f(x) \\ 0 \end{smallmatrix}\right]$.

Solving linear equation fast Solving this linear equation can be slow in general: $O(n^3)$; but as in the unconstrained case, if you can exploit some special structure in the LHS matrix - maybe to find approximate search direction, search direction can be found fast : see unconstrained case for details.

20.2.4.2 Line search maintains feasibility

$\Delta \mathrm{x} \in null(A)$, so $t\Delta x$ found during line search remains feasible. ***Feasibility always maintained!***

20.2.4.3 Affine invariant stopping criterion

Use $\lambda(x) = (\Delta x^T \nabla^2 f(x) \Delta x)^{1/2} = (-\nabla f(x)^T \Delta x)^{1/2}$. Use last term to compare with stopping criterion for unconstrained minimization case. In general, it is not same as $\Delta x^T \nabla^2 f(x) \Delta x$.
Justification for use of $\lambda(x)^2/2$ as stopping criterion: Just as in the unconstrained case, $f(x) - p^* \approx f(x) - \inf_{Ax=b} \hat{f} = 2^{-1}\lambda(x)^2$; and $-\lambda(x)^2$ is also the directional derivative along Δx.

20.2.4.4 Analysis: Use Newton method on equiv unconstrained problem

Take $\min f(x) : Ax = b \equiv \min f(Hv + x')$ form. $x^{(t)} = Hv^{(t)} + x'$. So, analysis of convergence of unconstrained problem applies here too!

20.2.4.5 Using infeasible start

Primal, dual variable update view Take residue $r(x,m) = (\nabla f(x) + A^T m, Ax - b)$, solve the vector optimization problem: $\min_{x,m} r(x,m)$: so, minimizing the objective while maintaining constraint, but also getting close to feasibility. Try to get $r(y + \Delta y) \approx r(y) + Dr(y)\Delta y = 0$. So, use search direction from solving:
$\begin{bmatrix} \nabla^2 f(x) & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta \mathrm{x} \\ w \end{bmatrix} = -\begin{bmatrix} \Delta f(x) \\ Ax-b \end{bmatrix}$, with w =m+Δm, the new guess for the dual variable.
Compare with search direction in feasible start case, note the change in last element of RHS.
Not a descent alg: $f(x^+) \leq f(x)$ ***possible!***

Line search Backtracking line search is conducted on $\|r(y)\|$. Stopping condition for line search: $\|r(y + \Delta y)\| \leq (1 - \alpha t) \|r(y)\|$, for a certain α. So, stopping criterion becomes stricter as t increases.

Stopping criterion Ax = b and $\|r(y)\| \leq \epsilon$.

Switch to feasible start alg As soon as you get $x^{(k)} : Ax^{(k)} = b$, switch to feasible start version of the algorithm: often ensures faster descent.

21 Inequality constrained problems

21.1 Barrier methods

21.1.1 Make inequality constraints implicit

Consider constraint $f_i(x) \leq 0$. Can make this constraint implicit by including barrier functional $-\phi(x)$ in the objective $f_0'(x)$: now, the constraint is enforced by thus constraining the $dom(f_0(x))$.

21.1.1.1 Motivation using indicator fn

The best log barrier to use is actually an indicatior function $\phi(x) = I_{f_i(x) \leq 0}$ which is 0 if $f_i(x) \leq 0$, ∞ otherwise.

21.1.1.2 Logarithmic barriers

As $t \to \infty$, get $-t^{-1} \sum_i \log -f_i(x) \approx I_{f_i(x) \leq 0}$. So, $\phi(x) = \sum_i \log f_i(x)$. This is a good barrier functional, as it is convex, twice continuously differentiable.

21.1.2 Optimize both distance to log barrier and f0

Take problem in standard form. Solve the vector optimization problem: $\min(f_0(x), -\phi(x)) : Ax = b$, where $\phi(x) = \sum \log(-f_i(x))$ is the log barrier, which is convex from composition rules.

21.1.2.1 Tradeoff: minimizing f0 and repulsion from barrier

Scalarize this to get the objective $\min g(t) = \min t f_0(x) - \phi(x)$: aka the centering problem. This is an equality constrained convex optimization problem. Bigger t tends to favor minimizing $f_0(x)$, while allowing x to get closer to the barrier $f(x) = 0$.
For t large enough, letting x get closer to the barrier does not affect the minimization of $f_0(x)$: maybe minimum is acheived at a point where some constraints are inactive.

21.1.2.2 Barrier algorithm

Start with some t. Solve the centering problem specified by t. Grow t by $\mu \approx 20$. Repeat until stopping criterion is satisfied.

21.1.2.3 Interpretation using Lagrangian

Take the optimality condition :
$\nabla f_0(x^*) + t^{-1} \sum_i (-f_i(x^*))^{-1} \nabla f_i(x^*) + t^{-1} A^T m^*$.
This can be seen as the minimum of a Lagrangian-like function: $L(x, l, m) = f_0(x) + \sum_i l_i f_i(x) + m^T Ax$, with $l \geq 0$. The optimal values are:
$l_i^* = t^{-1}(-f_i(x^*))^{-1} \geq 0, A^T m^* = 0$.
So, $p^* \geq g(l^*(t), m^*(t)) = L(x^*(t), l^*(t), m^*(t)) = f_0(x^*(t)) - m/t$. m/t is the duality gap, goes to 0 as $t \to \infty$. So, $x^*(t) \to x^*$ as $t \to \infty$. ***Can use m/t as stopping criterion!***

21.1.2.4 Primal, dual points on the central path

$(x^*(t))$ are primal points on the central path. $(l^*(t), m^*(t))$ are dual points on the central path.

22 Ideas for faster solution

To reduce the time taken to solve the convex optimization problem, you reduce : a] the time taken per iteration b] the number of iterations by using a clever initialization point.

22.1 Solving using the dual function

Take primal $\min f_0(x) : \{f_i(x) \leq 0\}, \{h_i(x) = 0\}$; Get Lagrangian: $L(x, l, m)$; get $g(x) = \inf_x L(x, l, m)$; Solve $\max_{l,m} g(l, m)$; derive x^* from l^*, m^*.

Make extra inferences using KKT conditions.

22.2 Warm start

Can use the solution of a closely related optimization problem to solve the current problem. This idea is used in barrier method!
Sometimes this gives a better solution as using a bad initialization point would have returned a relatively worse solution due to the number of iterations exceeeding the limit specified in the maxiter parameter passed to the solver.

Part VII

Classes solved with convex programming

23 Quasiconvex optimization problem

f_0 is quasi-convex, $f_i(x)$ are convex. Epigraph form: $\min t : f_0(x) \leq t; f(x) \leq 0, Ax = b$.
This can have locally optimal points which are not globally optimal: there can be plateaus: from properties of quasiconvex fn.

23.1 Generality of constraints

Can replace all quasi-convex sublevel sets, which are convex sets, with sublevel sets of convex functions. Eg: Consider the case of the linear fractional programs.

23.2 Solution using bisection

Take the epigraph form, fix t. Can solve this using bisection (see another section), with each optimization problem being solved using convex programming.

24 Second order cone program (SOCP)

Minimize a linear functional : $f^T x$ subject to Fx = g and second order cone constraints.
Eg: used in robust linear programming.

24.1 Second order cone constraints

$\|A_i x - b_i\|_2 - (c_i^T x + d_i) \leq 0 : A_i \in R^{n_i \times n}$. This is obviously a convex constraint as the LHS is a sum of a convex function and a linear function. But squaring both sides is not a good way of showing convexity of the feasible set.
$(A_i \mathrm{x} - b_i, c_i^T \mathrm{x} + d_i)$ distinguishes part of a second order cone in R^{n_i+1}.

24.2 Generality

Generalizes LP: Take $n_i = 0$.
When $c_i = 0$, becomes QCQP. Does it generalize all QCQP? [**Check**]

25 Conic inequalities convex program

Consider proper cones K_i. Then, let objective f_0 remain convex, have equality constraints $Ax = b$, but specify constraints using generalized convex functions: $f_i(x) \preceq_{K_i} 0$. This is still a convex program as suvblevelsets of the above continue to be convex.
Strong duality holds if Slater's constraint qualification holds.

25.1 Conic form problem

Extends LP using conic inequalities: $Fx + g \preceq_K 0$.

25.2 Semidefinite programming (SDP)

Minimize linear fn $c^T x$ subject to linear matrix inequalities (LMI) involving symmetric matrices (see matrix algebra survey). Can collapse n LMI's into a single LMI $A_0 + \sum x_i A_i \preceq 0$ by increasing the matrix size n times.
LMI's define convex sets. Can also be viewed as a convex optimization problem specified using conic inequalities.

25.2.1 Generality

SOCP, which includes LP, can be reduced to SDP. Can replace second order cone constraint with $\begin{bmatrix} (c_i^T x + d_i) & A_i x + b_i \\ (A_i x + b_i)^T & (c_i^T x + d_i) \end{bmatrix} \succeq 0$. [**Find proof**]
So, can flexibly specify SDP with semidefinite cone, quadratic cone, linear inequalities' constraints; so called SQLP.

25.2.2 Recognizing SDP's

First, can try manipulating objective to be a linear functional, perhaps by taking the epigraph form. Then, $A \preceq 0$ with $A_{i,j} = a_{ij}^T x + b_{ij}$ form of LMI's are common. Also, Schur's complement is often useful in forming the constraint.

25.2.3 Examples

ew minimization, matrix norm minimization.

25.2.4 Dual SDP

Lagrangian $L(x, Z) = c^T x + \langle A_0 + \sum x_i A_i, Z\rangle = c^T x + \langle A_0, Z\rangle + \sum x_i \langle A_i, Z\rangle$ with $Z \succeq 0$.
$g(Z) = \inf_x L(x, Z) = \langle A_0, Z\rangle$ if $\sum x_i \langle A_i, Z\rangle + c_i x_i = 0 \forall x$, else $g(Z) = -\infty$. So, dual problem: $\max \langle A_0, Z\rangle : Z \succeq 0, \langle A_i, Z\rangle + c_i = 0 \forall i$. (Note: used self-duality of S^n_{++} in writing $Z \succeq 0$.) ***Also an SDP!***
Dual of dual is the primal!: Use $L(Z, m, L) = \langle G, Z\rangle + \sum_i m_i \langle F_i, Z\rangle + \langle L, -Z\rangle + c^T m$.

26 Geometric programming

$\min_x f_0(x) : \{f_i(x) \leq 0\}, \{h_i(x) = 0\}$, with all f_i posynomials and h_i monomials (see vector functionals survey).
Conversion to convex form: use $y_i = \log x_i$ and restate problem.
Eg: used in finding perron-frobenius ew: $|\lambda_{max}(A)|$.

Part VIII

Non convex optimization

27 Discrete optimization problems

27.1 Difficulty

The difficulty here arises from the fact that a huge (often exponentially large in the number of variables due to combinatorial explosion) number of assignments to the discrete variables should be considered in order to find the optimum.

27.2 General Strategies

Use exhaustive search.

27.2.1 Relaxation to allow continuous values

Constraints in an Integer program can be relaxed to allow variables to take on real values.

27.3 Graph based problems

Max flow, min cut problem. See graph theory survey.

27.3.1 Resource allocation

Often modelled with graphs. Edges indicate resource constraints or conflicts.

27.3.1.1 Maximum weight matching

Find the heaviest set of disjoint edges.
For bipartite graphs: if $\exists$ a unique matching, loopy belief propogation will find it.

28 Continuous variables: general strategies

The main difficulty here arises from the presence of a large number of local optima.
Or use local optimization attempts with random choices of initial points.
Or use a relaxation.

28.1 Convexification/ smoothing

Turn it into a convex optimization problem: eg change objective fn to e^x, or maybe strong duality holds: so can solve dual problem: see section on modelling/ formulating problems too.

28.1.1 Dual of the dual

Dual of the dual is sometimes a convex relaxation to the original problem, besides being a lower bound to it.

28.1.2 Local approximation

Or approximate f_0 (maybe locally) by a convex function or atleast a smoother function. Eg: Trust region methods.

28.1.3 Smoothing

Smoothing reduces irregularity of the function output - ie it reduces the depth of local minima.
Gaussian smoothing is frequently used: $g(x_0) = \int_{x \in [-\infty, \infty]} e^{-\lambda(x-x_0)^2} f(x) dx$.

28.2 As sampling

Global optimization can be seen as sampling from the feasible set, using anything monotonic with $f_0(w) = E(w)$ as a measure of energy/ improbability - albeit with the intention of finding the minimum.
In exploring the feasible set with special attention towards finding an optimum, one would want most updates to improve the objective, while the remaining updates help get out of local minima.

28.2.1 Stochastic gradient descent

Often the objective function can be decomposed as follows: $E(w) = \sum_{i=1:n} E_i(w)$ - for example, in terms of various data-points in the case of maximum likelihood esitation.
Here, for each iteration, one chooses a data-point $E_i(w)$ at random or in a sequence, and uses $-\nabla E_i(w)$ as a descent direction.

28.2.1.1 Comparison with stochastic gradient descent

Unlike gradient descent, which, using $\nabla E(w)$ as a direction, can consistently reduce E(w) if the step size is appropriately chosen, stochastic gradient descent does not make such a guarantee. The advantage of stochastic gradient descent is in not being stuck at local optima, and in the greater speed with which $\nabla E_i(w)$ can be evaluated.

28.2.2 Damping jitters

Sampling techniques may sometimes result in excessive oscillations in the value of $E(w)$ after each variable update: $w_i = w_{i-1} + \Delta w_{i-1}$, where $\Delta w_{i-1} = \alpha t_{i-1}$ using step size α and direction t_{i-1}.
One may use a momentum parameter to dampen the abrupt changes in direction: $\Delta w_{i-1} = \Delta w_{i-2} + \alpha t_{i-1}$.
This is beneficial because in exploring the feasible set with special attention towards finding an optimum, one would want most updates to improve the objective, while the remaining updates help get out of local minima.

28.2.3 Using distribution sampling techniques

See section on sampling from a distribution in randomized algorithms survey.
Note that in sampling for optimization, algorithms may pay attention towards

finding optima.

29 Local optimization

29.1 Result

The result is often highly sensitive to the initial value of x. Also, one cannot guarantee that one will reach the minimum closest to the initial point: For example, when gradient descent is used, the angle of the gradient may lead one to a point which then leads a different well.

29.2 Techniques

Can use any convex optimization technique, like gradient descent or alternating minimization.

Part IX

Discrete and Combinatorial optimization

30 Integer programming (IP)

LP problem where variables can only take integer valued solutions. It is NP hard to find a solution: you have combinatorial hardness.
Approximate with LP; solve it; round the solutions. [**Incomplete**]

30.1 Randomized rounding

Round x to $\lfloor x \rfloor$ with prob $x - \lfloor x \rfloor$.

31 Optimal substructure problems

Aka Dynamic programming.

31.1 Applicability: Decision tree view

The problem can be cast as one of taking a sequence of decisions, and one wants to find the optimal sequence of decisions. So, essentially, one tries to find the optimal path through a decision tree. The number of decisions one needs to take is bounded by N.
Problems exhibit the 'optimal substructure' property, and also often the 'overlapping subproblems' property.

31.1.1 Optimal substructure

Optimal solutions of simpler subproblems can be compared in some way to find the overall optimal solution.
A problem corresponds to a decision tree D_l at level l. Each subproblem corresponds to finding optimal path p_i through a different decision subtree D_i one would arrive at by fixing the first decision to be e_i. One constructs the optimal decision path $p = \min_i f(p_i + e_i)$.
Eg: In case of shortest path problem: $d(s, e) = \min_{v \in \Gamma(s)}[d(s, v) + d(v, e)]$.

31.1.1.1 Remembering subproblems used

p is a sequence of decisions $d_{l,i}$ - each corresponding to making decision i at level l of the decision tree. One needs to remember the decision taken at level l - the optimal subpath augmented. Eg: in the example above, in order to reconstruct the shortest path from s to e, one needs to remember which $v \in \Gamma(s)$ was used.

31.1.2 Overlapping subproblems

The subproblems solved are repeated. This corresponds to the case where decision sub-paths to various leaves are actually identical. So it is profitable to remember solutions to subproblems.

31.2 Top down vs Bottom up

31.2.1 Top down solution

A top down solution can be easily expressed in terms of a recursive function $f(D)$ which acts on a certain decision tree and returns a] the optimal decision path p and b] its cost.
In doing this, if the 'overlapping subproblem' property holds, the algorithm memoizes : ie remembers optimal solutions to these subproblems whenever they are solved.

31.2.2 Bottom up

This solution is only applicable when the 'overlapping subproblem' property holds.

The algorithm solves decision trees of the smallest depth, records their results and builds solutions to progressively larger decision trees. So, one goes from level N and works one's way up to level 1.

31.2.2.1 Tabular view

Suppose that any node in the decision tree has at most N children. This process can be viewed by means of a $N \times M$ table or a list of N lists.
First, one constructs a list or column corresponding to the consequences of M different decisions at level N.
Then, one constructs a list corresponding to the consequences of M decisions at level $N-1$, and also a list of 'backpointers' specifying the ideal decision at level N if one were to fix decision d at level $N-1$.
One does this unductively until one covers all decisions up to level 1.

31.2.3 Time complexity

From description of the bottom up solution, it is clear that time/ space required is $M * N$ - unlike M^N in case all paths in the entire decision tree are to be considered (true in case 'overlapping subproblems' property does not hold).

31.3 Examples

Shortest path algorithm can be formulated as dynamic program - see graph theory survey.
FFT: See functional analysis ref.
Determining the most likely state sequence in the case of a HMM.

32 Branch and bound

Systematic enumeration of all candidate solutions, where large subsets of fruitless candidates are discarded en masse, by using upper and lower estimated bounds of the quantity being optimized.

33 With belief propogation

Rewrite as a problem of finding the mode of a distribution: $\max_x Pr(x)$: $Pr(x) \propto 1_{f(x)\leq 0, h(x)=0} e^{f_0(x)}$: the exponentiation is to ensure non-negativity.
This is useful when f_0,f, h are decomposable into functionals over cliques: then can take advantage of factorization.
Used in combinatorial optimization.