Optimization

Classical problems:

- find x st $f(x)$ is minimized
- interesting cases

$$x \in R^n \quad \begin{cases} f \in C^2 & \text{(cont. and 1, 2 deriv cont)} \\ f \in C^1 & \\ f \in C^0, & f \text{ convex} \\ f \in C^0 & \end{cases}$$

Constrained: $x \in \{u \mid g(u) = 0\}$

Discrete: $x \in \{0, 1\}^n$

Variational:

choose g such that

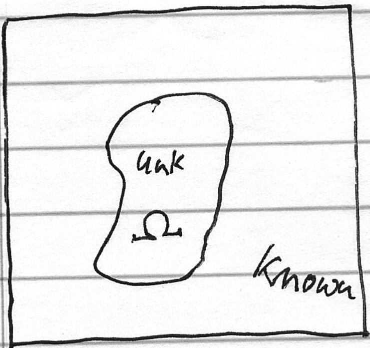
$$\int F(x, g(x))\, dx \quad \text{is minimized}$$

How do we know we are at a local min?

- Any step in any direction makes cost get bigger
- Easy test if f is ~~differentia~~ C^1

$$\nabla f = 0$$

- life is harder is f is C^0 or worse
  - local test might be difficult
  - eg.



we want to fill in Ω

reasonable criterion:

- $\min \int_{\Omega} \|\nabla f\|^2 \, dA$

<u>subject to:</u> $f = I$ on $\partial\Omega$

<u>i.e</u>
- don't create derivatives unnecessarily
- agree w/ boundary.

- How could we solve this?

a)
- discretize, work with ~~discretized~~ derivative and integral
- we are now minimizing a function of a (big!) vector

Alternative:

- What properties does f have?

→ assume that $\hat{f}$ is the soln.

→ now, for <u>ANY</u> test function φ,
such that $\phi = 0$ on $\partial\Omega$,
we have

$$\int_\Omega \| \nabla (f + \varepsilon\phi) \|^2 \, dA \geq \int \| \nabla f \|^2 \, dA$$

for small enough $\varepsilon > 0$

(i.e. if you make a small move in <u>any</u> direction, the value goes <u>UP</u>)

Now, this means

$$\frac{d}{d\varepsilon} \int_{\Omega} \| \nabla f + \varepsilon \phi \|^2 \, dA = 0 \quad \text{for } \underline{\text{any}} \ \varphi$$

now this is

$$2 \int_{\Omega} \nabla f \cdot \nabla \phi \, dA = 0 \quad (\text{doesn't seem helpful})$$

<u>But</u> recall

$$\nabla \cdot \left[g \, \underline{v} \right] = \left(\nabla g \right) \cdot v + g \left(\nabla \cdot v \right)$$

<u>i.e.</u>

$$\int_{\Omega} \nabla \cdot [\cancel{g} \phi \nabla f] \, dA = \int_{\Omega} \nabla \varphi \cdot \nabla f \, dA + \int_{\Omega} \varphi (\nabla^2 f) \, dA$$

now

$$\int_{\Omega} \nabla \cdot [\phi \nabla f] \, dA$$

$$= \int_{\partial\Omega} (\phi \nabla f) \cdot ds$$

(divergence thm – remember!)

but $\varphi = 0$ on $\partial \Omega$.

so $\int_{\Omega} \nabla\varphi \cdot \nabla f \, dA = -\int_{\Omega} \varphi \, (\nabla^2 f) \, dA = 0$

But this is true for <u>any</u> φ.

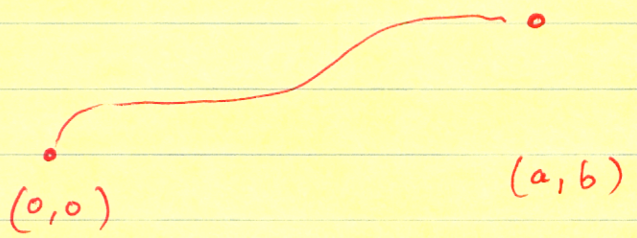
So $\nabla^2 f = 0$

and this offers other ways to solve.

- Gives a nice criterion for variational case

$$\frac{d}{d\varepsilon}\left[\int F(u, g(u)+\varepsilon\phi(u))\,du\right]\Bigg|_{\varepsilon=0} = 0$$

<u>Variational example</u>



$$\min_{g} \int_0^a \sqrt{1+\left(\frac{dg}{du}\right)^2}\,du$$

<u>Subject to:</u>

$$g(0) = 0$$
$$g(a) = b$$

① d

i.e. for any test function φ, $\varphi(0)=0$, $\varphi(a)=0$

we know g is right if

$$\frac{d}{d\varepsilon}\left[\int \sqrt{1+\left[\frac{d}{dx}(g+\varepsilon\varphi)\right]^2}\, du\right]\Bigg|_{\varepsilon=0} = 0$$

now write $\frac{dg}{du} = g'$, $\frac{d\varphi}{du} = \varphi'$

i.e

Criterion is:

$$\int \frac{g'\varphi'}{\sqrt{1+g'^2}}\, du = 0$$

Not promising; but use integration by parts

(i.e. $\int u v' dx = uv\Big|_{x=0}^{x=a} - \int v u' dx$)

to get

$$\int \frac{d}{du}\left[\frac{g'}{(1+g'^2)^{1/2}}\right]\cdot \varphi \, du = 0$$

for ~~any~~ φ $\left(\varphi(0) = \varphi(a) = 0\right)$

this gives $\frac{d}{du}\left[\frac{g'}{(1+g'^2)^{1/2}}\right] = 0$

which means

$$g''\left[\frac{1 - \frac{g'^2}{(1+g'^2)^{1/2}}}{(1+g'^2)}\right] = 0$$

and $[\ ]$ is always positive so $g'' = 0$

(The shortest distance between 2 points is a line!)

Equations obtained like this are

$\boxed{\text{Euler Lagrange equations}}$

Variational problems can be quite delicate

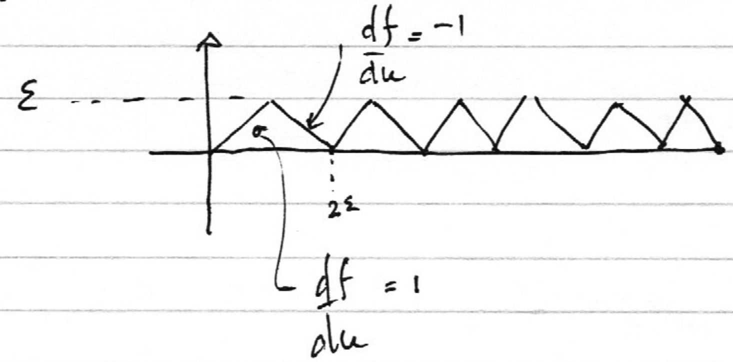
– a solution could not exist in reasonable function spaces.

example

$$\min_f \int_0^1 \cancel{\ } f(u)^2 + \left[\left(\frac{df}{du}\right)^2 - 1\right]^2 du = \mathcal{F}[f]$$

s.t. $f(0) = 0$; $f(1) = 0$.

Soln looks like



but as ε gets smaller, we have that $\mathcal{F}$ gets smaller, too

→ but there can't be a limit

So no solution.

Failure of a solution to exist occurs in practical problems

- <u>Simple issues</u>:

$$\underset{u}{\operatorname{argmin}} \quad u^2 \qquad u \in (0, 1]$$

open.

(This doesn't happen all that often, but is worth keeping in mind)

- <u>Harder issues</u>.

$$\underset{f}{\operatorname{argmin}} \quad \int_0^1 \left[\frac{1}{\sqrt{1+(f')^2}} - \frac{1}{\sqrt{2}} \right]^2 dx$$

s.t. $\quad f(0) = 0$

$\quad\quad f(1) = 0$

This sort of thing turns up in Shape from Shading problems rather often.

Notice I can get a min of the objective if $f'^2 = 1$.

→ So, if $f \in C^\infty$ no solution (there can't be a C^∞ function st $f'^2 = 1$, $f(0) = 0$, $f(1) = 0$)

→ if $f \in C^0$, too many solutions!

$\wedge$, $\wedge\wedge$, $\wedge\wedge\wedge$ etc.

~~Now assume that~~

In practice, we usually turn variational problems into continuous optimization problems by writing

$$f = \sum_i a_i g_i$$

↳ basis functions

then solving for a_i

<u>BUT</u>
- bad stuff can happen if original problem is poorly set
- The <u>reasoning</u> comes in useful later

<u>Crucial, Take Home point:</u>

You are at a minimum if every available step is uphill

Now consider:

$$\min \quad f(x) \qquad f \in C^2$$

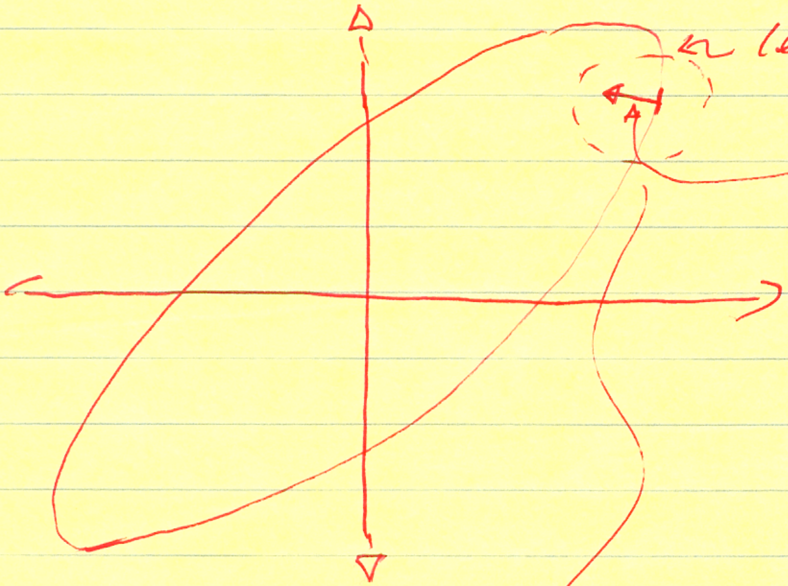
- A Descent Direction $\underline{d}$ has the property that

$$f(\underline{x}_0 + \varepsilon \underline{d}) < f(x_0) \quad \text{for}$$

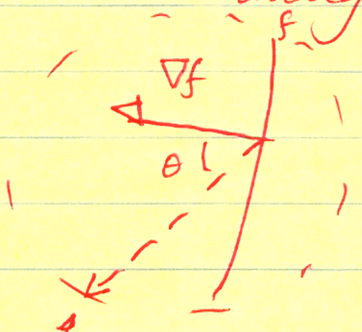
$$\varepsilon < r_d$$

Both gradient descent, coordinate descent are <u>naughty</u>:

- Local model of function as quadratic form



<u>best step</u> is <u>not</u> along gradient



<u>best step.</u>

There are numerous descent directions

* $d_g = \dfrac{-\nabla f}{\|\nabla f\|}$

this is gradient descent

issues: · how to choose ε
· perhaps interval halving
· more sophisticated machinery later

* write P_c for projection to some set of coordinate axes

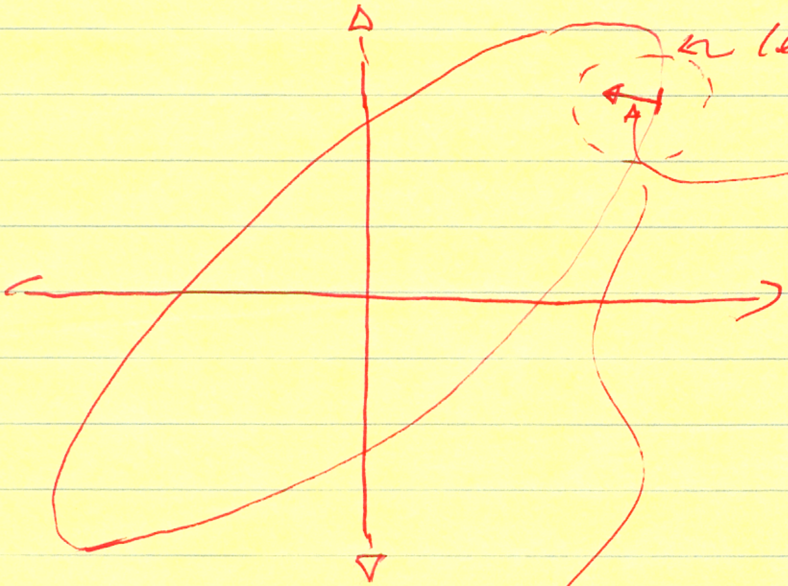
– i.e. P_c zeros some elements

$d_{cd} = -P_c d_g$

this is coordinate descent

Both gradient descent, coordinate descent are <u>naughty</u>:

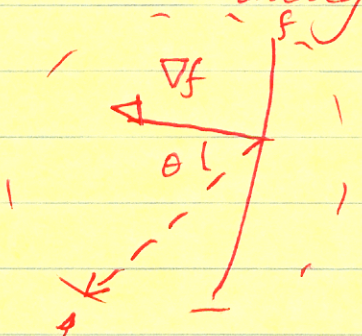
- Local model of function as quadratic form



level curve $f = a$

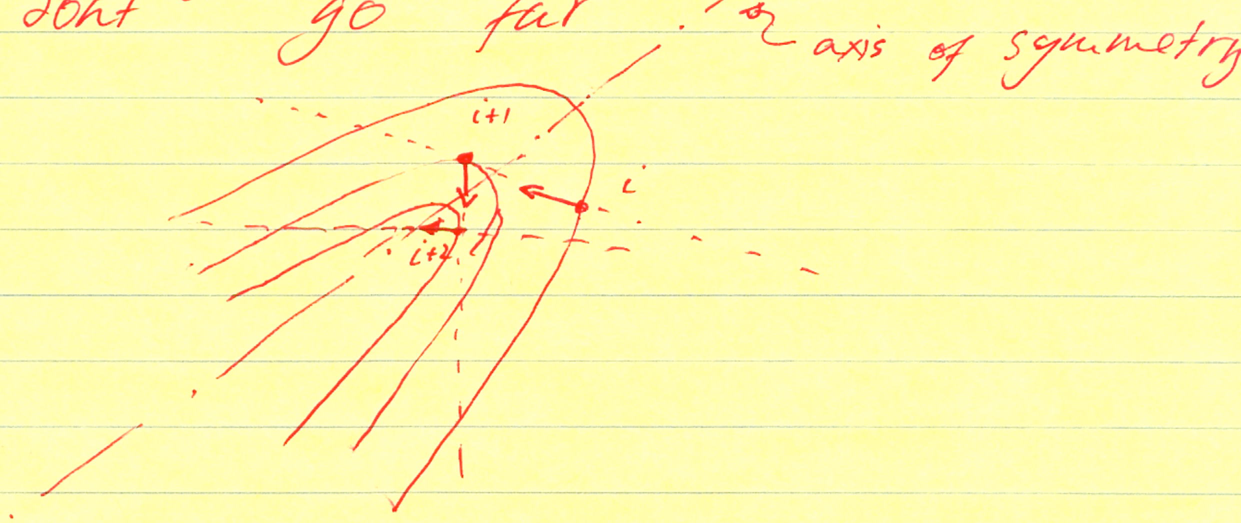
gradient, perp to level curve

<u>best step</u> is <u>not</u> along gradient



<u>best step.</u>

- if we take the best step along the gradient in this case, we don't go far



Now we zigzag slowly down the axis

Notice, best step can be at $(-90°, 90°)$ to gradient

function is:

$$x' A x \quad , A \text{ positive definite}$$

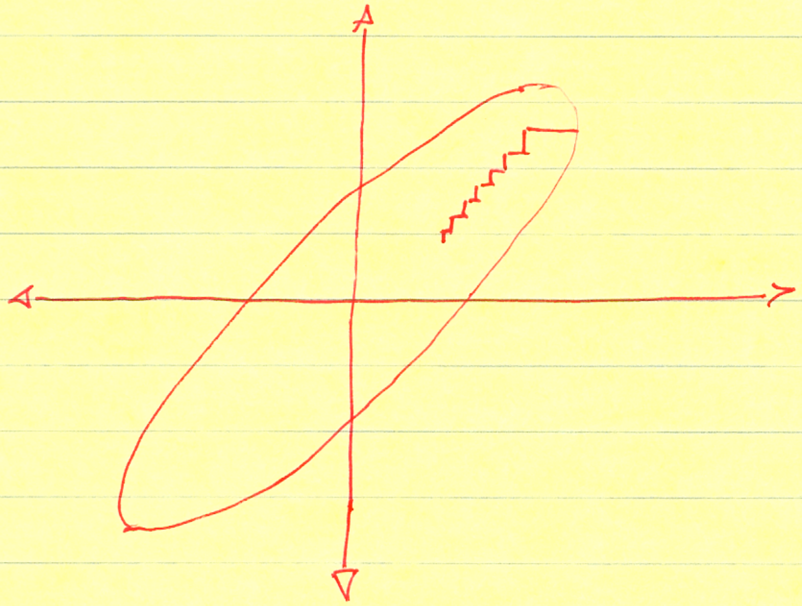
we are at u.

gradient is: $2Au$
best step dir is $-u$.

So cos angle is :

$$\frac{u'Au}{(u'u)^{1/2}\,(u'A'Au)^{1/2}}$$

Coordinate descent is also naughty



Newton's method:

$$f(x_0 + \delta) \approx f(x_0) + \nabla f' \delta + \frac{1}{2} \delta' H_f \delta + O(\delta^3)$$

- we could minimize the quadratic part as a function of δ.

$$\min_{\delta} \quad \nabla f' \delta + \frac{1}{2} \delta' H_f \delta$$

i.e. $\quad \nabla f + H_f \delta = 0$

$$\boxed{H_f \delta = -\nabla f}$$

Q: Does Newton's method always give a descent direction?

$$f(x+d) = f(x) + \nabla f\, d + \frac{1}{2} d' H_f\, d + O(d^3)$$

$$d = -H_f^{-1} \nabla f$$

$$f(x+d) - f(x) \approx -\frac{1}{2} \nabla f\, H_f^{-1} \nabla f$$

– So we're ok if H_f is positive definite

Q: is p a descent direction?

A: if $p^T \nabla f < 0$

Notice we can obtain descent dirs from

$$p_k = -B_k^{-1} \nabla f_k$$

iteration.

$$B_k^{-1} = Id \quad \text{gradient}$$
$$B_k^{-1} = P_c \quad \text{coord}$$
$$B_k^{-1} = H_f \quad \text{Newton.}$$

but

B_k must be P.d for p to be a descent dirn.

Example: Cooordinate descent and EM

we have two parametric models

$$p_1(x/\theta_1) = e^{-g_1(x;\theta_1)}$$

and ~~a noise model~~

$$p_2(x/\theta_2) = e^{-g_2(x;\theta_2)}$$

and we observe x_i from a mixture

$$p(x/\theta) = \mu_1 P_1 + \mu_2 P_2$$

Expectation - Maximization:

- Assume we have a mixture model

$$P(x|\theta) = \mu_1 P_1(x|\theta_1) + (1-\mu_1) P(x|\theta_2)$$

- for simplicity, I'll work with a mixture of exponentials

$$P_1(x|\theta_1) = e^{-g_1(x,\theta_1)}$$

$$P_2(x|\theta_2) = e^{-g_2(x,\theta_2)}$$

- I now have

$$x_1, \dots, x_n \sim P(x|\theta)$$

→ What is $\theta = (\theta_1, \theta_2, \mu)$?

- Inference by maximum likelihood will be hard because we must find

$$\arg\max_\theta \sum_i \log\left[\mu_1 e^{-g_1(x_i,\theta_1)} + (1-\mu_1) e^{-g_2(x_i,\theta_2)}\right]$$

- This is difficult to work with, ~~an~~
- The problem can be simplified if we <u>know</u> the mixture component from which each x_i comes.

$$\delta_i = \begin{cases} 1 & \text{if from } 1 \\ 0 & \text{" " } 2 \end{cases}$$

~~This give~~

i.e. $P(X_i \mid \delta_i, \theta)$ is straightforward

Write the Complete Data Log-likelihood

$$\mathcal{L}_c(\theta) = \sum_i \log\left[P(X_i, \delta_i \mid \theta)\right]$$

$$= \sum_i \left[\log P(X_i \mid \delta_i, \theta) + \log P(\delta_i \mid \theta)\right]$$

General algorithm:

we want to estimate these

$$h_c(\theta) = F(\delta, \theta)$$

we don't know these.

assume we have an estimate $\theta^{(n)}$ of θ

then

$$Q(\theta; \theta^{(n)}) = E_{\delta | \theta^{(n)}; X}\left[F(\delta, \theta) \right]$$

function of θ that depends on $\theta^{(n)}$

E-Step:

Form $Q(\theta; \theta^{(n)})$

M-Step

~~Choo~~ Find $\theta^{(n+1)} = \underset{\theta}{\operatorname{argmax}}\, Q(\theta; \theta^{(n)})$

In our case

$$\mathcal{L}_c(\Theta) = \sum_i \left[-\delta_i g_1(x_i, \theta_1) - (1-\delta_i) g_2(x_i, \theta_2) \right]$$

$$+ \sum_i \left[\delta_i \log \mu + (1-\delta_i) \log (1-\mu) \right]$$

$E_{\delta | x, \hat{\theta}} \mathcal{L}_c(\theta)$ is straightforward, because

$\mathcal{L}_c(\theta)$ is <u>linear</u> in δ.

→ Substitute $P(\delta_i = 1 | x, \hat{\theta})$ for δ_i

$$P(\delta_i = 1 | x, \hat{\theta}) = \frac{P(x_i, \delta_i = 1 \mid \hat{\theta})}{P(x_i \mid \hat{\theta})}$$

$$= \frac{P(x_i | \delta_i = 1, \hat{\theta}) P(\delta_i = 1 | \hat{\theta})}{P(x_i | \delta_i = 1, \hat{\theta}) P(\delta_i = 1 | \hat{\theta}) + P(x_i | \delta_i = 0, \hat{\theta}) P(\delta_i = 0 | \hat{\theta})}$$

In our case

$$P(\delta_i = 1 \mid x_i, \hat{\theta}^{(i)}) = \frac{e^{-g_1(x_i, \hat{\theta}_1^{(i)})}\,\hat{\mu}}{e^{-g_1(x_i, \hat{\theta}_1^{(i)})}\,\hat{\mu} + e^{-g_2(x_i, \hat{\theta}^{(i)})}\,(1-\hat{\mu}}$$

<u>Now</u>: consider the following objective fn.

$$\mathcal{F}(\theta, S) = \mathcal{L}_c(\theta) + H(S)$$

Entropy

$$-\sum_i [\delta_i \log \delta_i + (1-\delta_i)\log(1-$$

We will do coordinate descent

- fix θ, min wrt δ
- fix δ, min wrt θ

consider

$$\nabla_\delta \mathcal{F}(\hat{\theta}^{(i)}, \delta) = 0$$

$$\frac{\partial \mathcal{F}}{\partial \delta_i} = -\left[\log \delta_i - \log(1-\delta_i)\right]$$

$$+ \left[-g_1 + \log \hat{\mu}\right]$$

$$+ \left[g_2 - \log(1-\hat{\mu})\right]$$

$$= 0$$

i.e.

$$\frac{\delta_i}{1-\delta_i} = \frac{e^{-g_1}\mu}{e^{-g_2}(1-\mu)}$$

and we substitute these in.

→ <u>But this is what E step does.</u>

$$M \text{ step} \equiv \nabla_\theta \mathcal{J}(\theta, \hat{\delta}_i) = 0$$

Q: Why not do newton?

A: Not sure frankly.

H is <u>big</u> but sparse?

i.e.

$$\frac{\delta_i}{1-\delta_i} = \frac{e^{-g_1}\mu}{e^{-g_2}(1-\mu)}$$

hence: EM is coordinate ascent.

Q: why not do newtons method?

A: not sure, frankly.

H is big, but sparse

Issues:

- What to do with a descent dirn?
- How to make H behave?
  - → big
  - → not P.D.
- How bad is gradient descent?

We have p_k, and wish to choose a step length α.

consider $f(x_k + \alpha p_k)$ $\quad \alpha > 0$

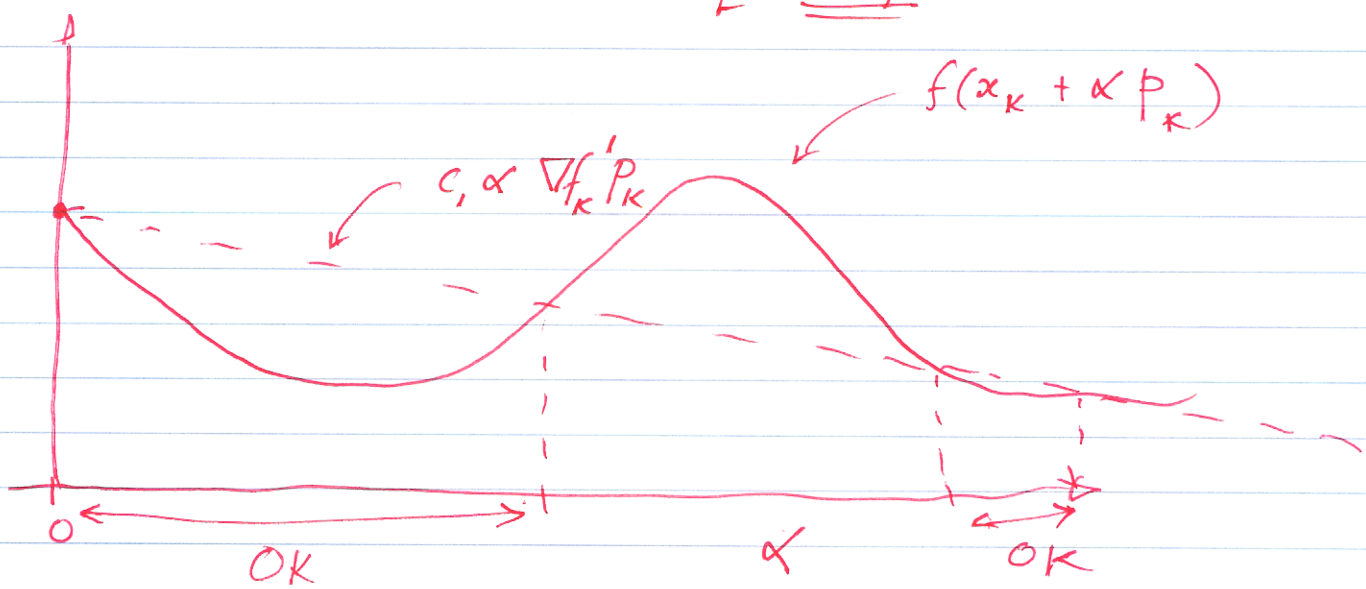
<u>what α are acceptable</u>?

- ideally, α is global minimizer
- sufficient decrease

$$f(x_k + \alpha p_k) \le f(x_k) + c_1 \alpha \nabla f_k^T p_k$$

$0 < c_1 < 1$

for some constant (typically 1e-4) [Armijo / Wolfe] condition



Sufficient Decrease is <u>not</u> enough

→ very small α are OK.

$$\nabla f(x_k + \alpha_* p_k)^T p_k \geq c_2 \nabla f_k^T p_k$$

$c_1 < c_2 < 1$

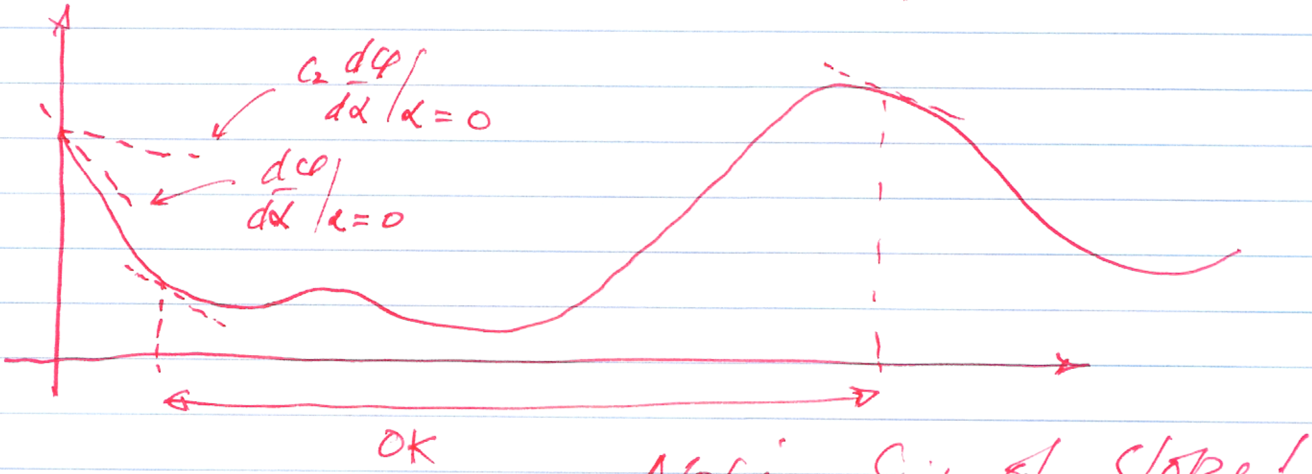
<u>notice</u>:

write $\varphi(\alpha) = f(x_k + \alpha p_k)$

then $\dfrac{d\varphi}{d\alpha} = \nabla f(x_k + \alpha p_k)^T p_k$

So condition is:

$$\frac{d\varphi}{d\alpha} \geq c_2 \nabla f_k^T p_k = c_2 \left.\frac{d\varphi}{d\alpha}\right|_{\alpha=0}$$



Notice Sign of Slope!

c_2 is usually 0.9 (Newton)
0.1 (conj. grad.)

$\boxed{\begin{array}{l} \text{Wolfe conditions} \\ f(x_k + \alpha_k p_k) \le f(x_k) + c_1 \alpha_k \nabla f_k^T p_k \\ \nabla f(x_k + \alpha_k p_k)' p_k \ge c_2 \nabla f_k' p_k \end{array}}$

Notice: for f continuously diff,
f bounded below along $x_k + \alpha p_k$, $\alpha > 0$
there exist intervals satisfying
these conds.

Alg: for $\tilde{\alpha} > 0$, $\rho \in (0,1)$
$\alpha = \tilde{\alpha}$
repeat until [sufficient descent]
$\alpha = \rho\alpha$
end

OK for newton; not as good for others.

Now we are generating a sequence $\{x_i\}$ by finding p_k, α_k and accepting

$$x_{k+1} = x_k + \alpha_k p_k$$

Q: How does this seq behave?

A:

Q: To what does it converge?

Q: How fast?

Some answers by defining

$$\cos\theta_k = \frac{-\nabla f_k^T p_k}{\|\nabla f_k\| \|p_k\|}$$

<u>Thm</u>: (Zoutendijk)

Consider iteration of given form, α_k satisfying Wolfe conds, f bounded below continuously diff in an open set N containing $L = \{x : f(x) \leq f(x_0)\}$. Assume ∇f is Lipschitz. Then

$$\sum_k \cos^2\theta_k \|\nabla f_k\|^2 < \infty$$

Rates of Convergence:

- We have, for $f \in C^2$, exact line search (i.e. best α_k), x^* the min, Gradient descent behaves like

$$f(x_{k+1}) - f(x^*) \leqslant r^2 \left[f(x_k) - f(x^*) \right]$$

related to Hessian

- For Newton, if x_0 sufficiently close to x^*

$$\| x_{k+1} - x^* \| \leqslant L \, \| x_k - x^* \|^2$$

related to Hessian

Newtons method with Hessian modification

Problem: H may not be P.d,

so

$$H p_k = -\nabla f$$ may not give

a descent direction

Strategy: modify H to be PD.

$$B_k = H_f + E_k$$

E_k chosen to make B_k PD

This will converge ~~a~~ globally if

~~k~~ ~~a~~ $\{H_f(x_k)\}$ is bounded

$$\Rightarrow \quad \|B_k\|\|B_k^{-1}\| \leq C > 0$$

Generally would like E_k small (so as to preserve Hessian info)

1: Add a multiple of Identity:

choose $\beta > 0$
if $\min_i h_{ii} > 0$

$\tau_0 = 0$

else

$\tau_0 = -\min(h_{ii}) + \beta;$

end.

for $k = \ldots$

attempt cholesky factorization of $H + \tau_k I$
if OK return factor

else

$\tau_{k+1} = \max(2\tau_k, \beta)$

end

end.

we are searching for τI to make H p.d.

Cholesky:

$$A = L L^T$$

lower triang

- works if A PD, otherwise get a $\sqrt{0}$, $\sqrt{-ve}$

Modified Cholesky

$$A = L D L^T$$

Where

L is lower triang, 1's on diag

D is diag, +ve diag

Note: if A pd, then D elements are +ve

Cholesky:

for $j = 1 \ldots n$

$$c_{jj} = a_{jj} - \sum_{s=1}^{j-1} d_s \, l_{js}^2$$

$$d_j = c_{jj}$$

for $i = j+1 \ldots n$

$$c_{ij} = a_{ij} - \sum_{s=1}^{j-1} d_s \, l_{is} \, l_{js}$$

$$l_{ij} = c_{ij} / d_j$$

end

end.

Now: d_{jj} all positive if A PD.

Modify alg so that

$$d_{jj} = \max \left(|c_{jj}|, \left(\frac{\theta_j}{\beta}\right)^2, \delta \right)$$

$$\theta_j = \max_{j < i \le n} |c_{ij}|$$

and this gives a "factorization"
where

$$d_j \geq \delta$$

$$\left| m_{ij} = l_{ij} \sqrt{d_j} \right| \leq \beta$$

Desirable for error control.

- Improvements

  - Permute rows and columns to reduce the size of the modification.

  - This will give guaranteed bounds ⇒ global convergence.

Step length selection:

$$\varphi(\alpha) = f(x_0 + \alpha p_k)$$

Sufficient decrease is then.

$$\varphi(\alpha) \leq \varphi(0) + c_1 \alpha \varphi'(0).$$

- guess α_0
  - $\rightarrow$ OK ; stop
  - $\rightarrow$ Not OK ; there is an OK step in interval.
  - we know $\varphi(0)$, $\varphi(\alpha_0)$, $\varphi'(0)$
  - build quadratic interpolate

$$\varphi_q(\alpha) = \left(\frac{\varphi(\alpha_0) - \varphi(0) - \alpha_0 \varphi'(0)}{\alpha_0^2} \right) \alpha^2$$

$$+ \quad \varphi'(0)\, \alpha$$

$$+ \quad \varphi(0)$$

- minimize in α to get α_1

→ α_1 OK ; stop

→ else construct cubic interpolate of $\varphi(0)$ $\varphi'(0)$ $\varphi(\alpha_0)$ $\varphi(\alpha_1)$

minimize ; α_2

→ α_2 OK stop

→ else cubic with $\varphi(0)$, $\varphi'(0)$, two most recent α

It can be shown that if $x_k \to x^*$ superlinearly, then the ratio in this expression converges to 1. If we adjust the choice (3.60) by setting

$$\alpha_0 \leftarrow \min(1, 1.01\alpha_0),$$

we find that the unit step length $\alpha_0 = 1$ will eventually always be tried and accepted, and the superlinear convergence properties of Newton and quasi-Newton methods will be observed.

A LINE SEARCH ALGORITHM FOR THE WOLFE CONDITIONS

The Wolfe (or strong Wolfe) conditions are among the most widely applicable and useful termination conditions. We now describe in some detail a one-dimensional search procedure that is guaranteed to find a step length satisfying the *strong* Wolfe conditions (3.7) for any parameters c_1 and c_2 satisfying $0 < c_1 < c_2 < 1$. As before, we assume that p is a descent direction and that f is bounded below along the direction p.

The algorithm has two stages. This first stage begins with a trial estimate α_1, and keeps increasing it until it finds either an acceptable step length or an interval that brackets the desired step lengths. In the latter case, the second stage is invoked by calling a function called **zoom** (Algorithm 3.6, below), which successively decreases the size of the interval until an acceptable step length is identified.

A formal specification of the line search algorithm follows. We refer to (3.7a) as the *sufficient decrease condition* and to (3.7b) as the *curvature condition*. The parameter $\alpha_{\max}$ is a user-supplied bound on the maximum step length allowed. The line search algorithm terminates with α_* set to a step length that satisfies the strong Wolfe conditions.

Algorithm 3.5 (Line Search Algorithm).

Set $\alpha_0 \leftarrow 0$, choose $\alpha_{\max} > 0$ and $\alpha_1 \in (0, \alpha_{\max})$;
$i \leftarrow 1$;
repeat
    Evaluate $\phi(\alpha_i)$;
    if $\phi(\alpha_i) > \phi(0) + c_1\alpha_i\phi'(0)$ or $[\phi(\alpha_i) \geq \phi(\alpha_{i-1})$ and $i > 1]$
        $\alpha_* \leftarrow$**zoom**(α_{i-1}, α_i) and **stop**;
    Evaluate $\phi'(\alpha_i)$;
    if $|\phi'(\alpha_i)| \leq -c_2\phi'(0)$
        set $\alpha_* \leftarrow \alpha_i$ and **stop**;
    if $\phi'(\alpha_i) \geq 0$
        set $\alpha_* \leftarrow$**zoom**(α_i, α_{i-1}) and **stop**;
    Choose $\alpha_{i+1} \in (\alpha_i, \alpha_{\max})$;
    $i \leftarrow i + 1$;
end (repeat)

Note that the sequence of trial step lengths $\{\alpha_i\}$ is monotonically increasing, but that the order of the arguments supplied to the **zoom** function may vary. The procedure uses the knowledge that the interval (α_{i-1}, α_i) contains step lengths satisfying the strong Wolfe conditions if one of the following three conditions is satisfied:

(i) α_i violates the sufficient decrease condition;

(ii) $\phi(\alpha_i) \geq \phi(\alpha_{i-1})$;

(iii) $\phi'(\alpha_i) \geq 0$.

The last step of the algorithm performs extrapolation to find the next trial value α_{i+1}. To implement this step we can use approaches like the interpolation procedures above, or we can simply set α_{i+1} to some constant multiple of α_i. Whichever strategy we use, it is important that the successive steps increase quickly enough to reach the upper limit $\alpha_{\max}$ in a finite number of iterations.

We now specify the function **zoom**, which requires a little explanation. The order of its input arguments is such that each call has the form **zoom**$(\alpha_{\text{lo}}, \alpha_{\text{hi}})$, where

(a) the interval bounded by α_{lo} and α_{hi} contains step lengths that satisfy the strong Wolfe conditions;

(b) α_{lo} is, among all step lengths generated so far and satisfying the sufficient decrease condition, the one giving the smallest function value; and

(c) α_{hi} is chosen so that $\phi'(\alpha_{\text{lo}})(\alpha_{\text{hi}} - \alpha_{\text{lo}}) < 0$.

Each iteration of **zoom** generates an iterate α_j between α_{lo} and α_{hi}, and then replaces one of these endpoints by α_j in such a way that the properties (a), (b), and (c) continue to hold.

Algorithm 3.6 (zoom).

repeat
  Interpolate (using quadratic, cubic, or bisection) to find
    a trial step length α_j between α_{lo} and α_{hi};
  Evaluate $\phi(\alpha_j)$;
  if $\phi(\alpha_j) > \phi(0) + c_1\alpha_j\phi'(0)$ or $\phi(\alpha_j) \geq \phi(\alpha_{\text{lo}})$
    $\alpha_{\text{hi}} \leftarrow \alpha_j$;
  else
    Evaluate $\phi'(\alpha_j)$;
    if $|\phi'(\alpha_j)| \leq -c_2\phi'(0)$
      Set $\alpha_* \leftarrow \alpha_j$ and **stop**;
    if $\phi'(\alpha_j)(\alpha_{\text{hi}} - \alpha_{\text{lo}}) \geq 0$
      $\alpha_{\text{hi}} \leftarrow \alpha_{\text{lo}}$;
    $\alpha_{\text{lo}} \leftarrow \alpha_j$;
end (repeat)

If the new estimate α_j happens to satisfy the strong Wolfe conditions, then **zoom** has served its purpose of identifying such a point, so it terminates with $\alpha_* = \alpha_j$. Otherwise, if α_j satisfies the sufficient decrease condition and has a lower function value than x_{lo}, then we set $\alpha_{\text{lo}} \leftarrow \alpha_j$ to maintain condition (b). If this setting results in a violation of condition (c), we remedy the situation by setting α_{hi} to the old value of α_{lo}. Readers should sketch some graphs to see for themselves how **zoom** works!

As mentioned earlier, the interpolation step that determines α_j should be safeguarded to ensure that the new step length is not too close to the endpoints of the interval. Practical line search algorithms also make use of the properties of the interpolating polynomials to make educated guesses of where the next step length should lie; see [39, 216]. A problem that can arise is that as the optimization algorithm approaches the solution, two consecutive function values $f(x_k)$ and $f(x_{k-1})$ may be indistinguishable in finite-precision arithmetic. Therefore, the line search must include a stopping test if it cannot attain a lower function value after a certain number (typically, ten) of trial step lengths. Some procedures also stop if the relative change in x is close to machine precision, or to some user-specified threshold.

A line search algorithm that incorporates all these features is difficult to code. We advocate the use of one of the several good software implementations available in the public domain. See Dennis and Schnabel [92], Lemaréchal [189], Fletcher [101], Moré and Thuente [216] (in particular), and Hager and Zhang [161].

One may ask how much more expensive it is to require the strong Wolfe conditions instead of the regular Wolfe conditions. Our experience suggests that for a "loose" line search (with parameters such as $c_1 = 10^{-4}$ and $c_2 = 0.9$), both strategies require a similar amount of work. The strong Wolfe conditions have the advantage that by decreasing c_2 we can directly control the quality of the search, by forcing the accepted value of α to lie closer to a local minimum. This feature is important in steepest descent or nonlinear conjugate gradient methods, and therefore a step selection routine that enforces the strong Wolfe conditions has wide applicability.

NOTES AND REFERENCES

For an extensive discussion of line search termination conditions see Ortega and Rheinboldt [230]. Akaike [2] presents a probabilistic analysis of the steepest descent method with exact line searches on quadratic functions. He shows that when $n > 2$, the worst-case bound (3.29) can be expected to hold for most starting points. The case $n = 2$ can be studied in closed form; see Bazaraa, Sherali, and Shetty [14]. Theorem 3.6 is due to Dennis and Moré.

Some line search methods (see Goldfarb [132] and Moré and Sorensen [213]) compute a direction of negative curvature, whenever it exists, to prevent the iteration from converging to nonminimizing stationary points. A direction of negative curvature p_- is one that satisfies $p_-^T \nabla^2 f(x_k) p_- < 0$. These algorithms generate a search direction by combining p_- with the steepest descent direction $-\nabla f_k$, often performing a curvilinear backtracking line search.

It is difficult to determine the relative contributions of the steepest descent and negative curvature directions. Because of this fact, the approach fell out of favor after the introduction of trust-region methods.

For a more thorough treatment of the modified Cholesky factorization see Gill, Murray, and Wright [130] or Dennis and Schnabel [92]. A modified Cholesky factorization based on Gershgorin disk estimates is described in Schnabel and Eskow [276]. The modified indefinite factorization is from Cheng and Higham [58].

Another strategy for implementing a line search Newton method when the Hessian contains negative eigenvalues is to compute a direction of negative curvature and use it to define the search direction (see Moré and Sorensen [213] and Goldfarb [132]).

Derivative-free line search algorithms include golden section and Fibonacci search. They share some of the features with the line search method given in this chapter. They typically store three trial points that determine an interval containing a one-dimensional minimizer. Golden section and Fibonacci differ in the way in which the trial step lengths are generated; see, for example, [79, 39].

Our discussion of interpolation follows Dennis and Schnabel [92], and the algorithm for finding a step length satisfying the strong Wolfe conditions can be found in Fletcher [101].

EXERCISES

3.1 Program the steepest descent and Newton algorithms using the backtracking line search, Algorithm 3.1. Use them to minimize the Rosenbrock function (2.22). Set the initial step length $\alpha_0 = 1$ and print the step length used by each method at each iteration. First try the initial point $x_0 = (1.2, 1.2)^T$ and then the more difficult starting point $x_0 = (-1.2, 1)^T$.

3.2 Show that if $0 < c_2 < c_1 < 1$, there may be no step lengths that satisfy the Wolfe conditions.

3.3 Show that the one-dimensional minimizer of a strongly convex quadratic function is given by (3.55).

3.4 Show that the one-dimensional minimizer of a strongly convex quadratic function always satisfies the Goldstein conditions (3.11).

3.5 Prove that $\|Bx\| \geq \|x\|/\|B^{-1}\|$ for any nonsingular matrix B. Use this fact to establish (3.19).

3.6 Consider the steepest descent method with exact line searches applied to the convex quadratic function (3.24). Using the properties given in this chapter, show that if the initial point is such that $x_0 - x^*$ is parallel to an eigenvector of Q, then the steepest descent method will find the solution in one step.