**Motivation**

We saw earlier that there was an optimal smoothness parameter for each of our smoothers. We could pick the smoothness parameter to optimize the estimated prediction error. Calculating the prediction error can be time consuming. How do we do this efficiently? This is an example application for line search algorithms. Can also be used to optimize design parameters to maximize some metric of performance.

**Convexity**

- A function \( f(\alpha) \) is **convex**\([2, \text{pp. 79}]\) if \( f(\lambda \alpha_1 + (1 - \lambda) \alpha_2) \leq \lambda f(\alpha_1) + (1 - \lambda) f(\alpha_2) \) for all \( 0 \leq \lambda \leq 1 \).
- A function \( f(\alpha) \) is **quasiconvex**\([2, \text{p. 108}]\) if \( f(\lambda \alpha_1 + (1 - \lambda) \alpha_2) \leq \max \{ f(\alpha_1), f(\alpha_2) \} \) for all \( 0 \leq \lambda \leq 1 \).
- A differentiable function \( f(\alpha) \) is **pseudocentric**\([2, \text{pp. 113–114}]\) if for every \( \nabla_x f(x_1)(x_2 - x_1) \geq 0 \), we have \( f(x_2) \geq f(x_1) \).

---

**Overview of Line Search Topics**

- Problem definition
- Line search algorithms
  - Uniform search
  - Dichotomous search
  - Golden section search
  - Quadratic fit search

**Problem Definition**

- The **line search problem**: find a scalar \( \alpha \in \mathbb{R}^1 \) such that \( \alpha^* = \arg\min_{\alpha} f(\alpha) \) using as few evaluations of \( f(\alpha) \) as possible (see \([1, 7.1–7.4] \) and \([2, 8.1–8.4]\))
- The **optimization problem**: find a vector \( a \in \mathbb{R}^p \) such that \( a^* = \arg\min_{\alpha} f(a) \) using as few evaluations of \( f(a) \) as possible
  - A generalization of the line search problem to multiple dimensions
Interval Bounding Algorithm

This is one popular algorithm for finding the upper limit for a line search algorithm. The user specifies the initial step $\alpha_1$ and picks the expansion rate $c$.

- Evaluate the function at $\alpha_0 = 0$
- Evaluate the function at $\alpha_1$
- Evaluate the function at $\alpha_2 = \alpha_1 \times c$
- $k = 1$
- Until $f(\alpha_{k-1}) > f(\alpha_k)$ and $f(\alpha_k) < f(\alpha_{k+1})$
  - $k := k + 1$
  - $\alpha_{k+1} := \alpha_k \times c$
- There is a minimum between $\alpha_{k-1}$ and $\alpha_{k+1}$

Note that this algorithm may also increase the lower bound from its initial value of 0.

Interval Bounding Problem

- Many line search algorithms try to find a local minimum in the range $[0, \infty]$ as quickly as possible
- If the criteria contains multiple local minima, they find one of them quickly
- Otherwise they find the global minima
- Most of these algorithms require that the minima be constrained to a finite range: $\alpha^* \in [\alpha_{\text{min}}, \alpha_{\text{max}}]$
- By the nature of the problem, the lower limit is usually known to be zero: $\alpha_{\text{min}} = 0$
- Often the upper limit has to be found
Bounding the Line Search Interval

More generally, the line search interval can be found by “interval doubling”. Assume that $\alpha$ is constrained to being a positive number.

1. Pick initial values of $\alpha_{\text{min}}$
2. While $f(\alpha_{\text{min}}) < f(\alpha_{\text{max}})$, $\alpha_{\text{min}} = \alpha_{\text{min}}/c$
3. While $f(\alpha_{\text{min}}) > f(\alpha_{\text{max}})$, $\alpha_{\text{max}} = \alpha_{\text{max}} \times c$

Typically $c = 2$, but any $c > 1$ could be used.

Example 1: MATLAB Code

```matlab
function [] = IntervalBound;

st = 2; % Step size
x = zeros(100,1);
x(1) = 0;
x(2) = 0.1; % First step
x(3) = x(2)*st;
cat = 3;
while ¬((LSF(x(cat-2))>LSF(x(cat-1)) & LSF(x(cat-1))<LSF(x(cat)))
    cat = cat + 1;
    x(cat) = x(cat-1)*st;
end

x = x(1:cat);
s0 = x(cat-2);
s1 = x(cat);

figure;
FigureSet(1,'Slides');
h = text(x(c1)+0.01,LSF(x(c1)),num2str(c1-1));
set(h,'HorizontalAlignment','Left');
set(h,'VerticalAlignment','Middle');
endbox off;
FigureLatex;
xlabel('\alpha');
ylabel('f(\alpha)');
AxisSet(8);
print -depsc IntervalBound;
```

Uniform Search

- Pick the search points $\alpha_1, \alpha_2, \ldots, \alpha_n$ so that they are uniformly spaced over some preset range
- Then pick the best $\alpha$

$$\alpha^* = \arg\min_{\alpha \in \{\alpha_1, \alpha_2, \ldots, \alpha_n\}} f(\alpha)$$

+ No assumptions about convexity or shape of $f(\alpha)$
+ Finds (nearly) a global minimum
- Relatively inefficient
Dichotomous Search

• Suppose you know that $\alpha$ is in the range of $\alpha_{min}$ to $\alpha_{max}$

1. Calculate the following evaluation points

$$b = \frac{\alpha_{min} + \alpha_{max}}{2} - \epsilon$$

$$c = \frac{\alpha_{min} + \alpha_{max}}{2} + \epsilon$$

2. If $f(b) < f(c)$, set $\alpha_{max} = c$

   Otherwise, set $\alpha_{min} = b$

3. Repeat until convergence

• If the derivative $\frac{df(\alpha)}{d\alpha}$ can be calculated, the computation can be reduced to one evaluation of the derivative per an iteration

• If the derivative is used, this is called the bisection method

  − Converges to a local minimum (global if $f(\alpha)$ is quasiconvex)
  
  + Faster than a uniform search
Example 3: Dichotomous Search

\[ f(\alpha) \]

\begin{tabular}{c c c c c c c c c c c}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\
0.25 & 0.2 & 0.15 & 0.1 & 0.05 & 0 & 0.05 & 0.1 & 0.15 & 0.2 \\
\end{tabular}

Example 3: MATLAB Code

```matlab
function [] = DichotomousSearch;
    clear all;
    close all;
    lll = 0; % Lower limit
    ul = 1; % Upper limit
    np = 10; % No. of points
    eta = 0.001; % Dither
    cnt = 0;
    x = zeros(np,1);
    f = zeros(np,1);
    for c1 = 1 : np/2 ,
        mp = (ul+ll)/2;
        b = mp-eta; fb = LSFn(b);
        c = mp+eta; fc = LSFn(c);
        if fb<fc,
            ul = c;
        else
            ll = b;
        end;
        cnt = cnt + 1;
        x(cnt) = b; f(cnt) = fb;
        cnt = cnt + 1;
        x(cnt) = c; f(cnt) = fc;
    end;
```

Example 3: MATLAB Code

```matlab
function [] = DichotomousSearch;
    clear all;
    close all;
    lll = 0; % Lower limit
    ul = 1; % Upper limit
    np = 10; % No. of points
    eta = 0.001; % Dither
    cnt = 0;
    x = zeros(np,1);
    f = zeros(np,1);
    for c1 = 1 : np/2 ,
        mp = (ul+ll)/2;
        b = mp-eta; fb = LSFn(b);
        c = mp+eta; fc = LSFn(c);
        if fb<fc,
            ul = c;
        else
            ll = b;
        end;
        cnt = cnt + 1;
        x(cnt) = b; f(cnt) = fb;
```

Example 3: MATLAB Code

```matlab
function [] = DichotomousSearch;
    clear all;
    close all;
    lll = 0; % Lower limit
    ul = 1; % Upper limit
    np = 10; % No. of points
    eta = 0.001; % Dither
    cnt = 0;
    x = zeros(np,1);
    f = zeros(np,1);
    for c1 = 1 : np/2 ,
        mp = (ul+ll)/2;
        b = mp-eta; fb = LSFn(b);
        c = mp+eta; fc = LSFn(c);
        if fb<fc,
            ul = c;
        else
            ll = b;
        end;
        cnt = cnt + 1;
        x(cnt) = b; f(cnt) = fb;
```
**Golden Section Line Search**

Suppose you know that $\alpha$ is in the range of $\alpha_{\text{min}}$ to $\alpha_{\text{max}}$

1. Evaluate $f(\alpha)$ at $\alpha_{\text{min}}, b, c,$ and $\alpha_{\text{max}}$ where
   
   \[ b_0 = \alpha_{\text{min}} + (1 - \gamma)(\alpha_{\text{max}} - \alpha_{\text{min}}) \]
   \[ c_0 = \alpha_{\text{min}} + \gamma(\alpha_{\text{max}} - \alpha_{\text{min}}) \]

2. If $f(b_k) > f(c_k)$,
   
   \[ \alpha_{\text{max},k+1} = \alpha_{\text{max},k} \]
   \[ \alpha_{\text{min},k+1} = b_k \]
   \[ b_{k+1} = c_k \]
   \[ c_{k+1} = \alpha_{\text{min},k+1} + \gamma(\alpha_{\text{max},k+1} - \alpha_{\text{min},k+1}) \]

   Else,
   
   \[ \alpha_{\text{min},k+1} = \alpha_{\text{min},k} \]
   \[ \alpha_{\text{max},k+1} = c_k \]
   \[ c_{k+1} = b_k \]
   \[ b_{k+1} = \alpha_{\text{min},k+1} + (1 - \gamma)(\alpha_{\text{max},k+1} - \alpha_{\text{min},k+1}) \]

3. Loop to 2 until convergence

---

**Golden Section Line Search Comments**

- $\gamma \approx 0.618$
- This is the inverse of the “Golden ratio”: $\phi = \frac{1}{2}(1 + \sqrt{5})$
- Not a rational number
- Consider a line segment from A to C
- Place a point B such that $\frac{AB}{BC} = \frac{BC}{AC}$
- This is the golden ratio
- For more info, search the web
  - Converges to a local minimum (global if quasi-convex)
  + More efficient than dichotomous search
Example 4: MATLAB Code

```matlab
function [] = GoldenSection;
clear all;
close all;
l = 0; % Lower limit
ul = 1; % Upper limit
np = 10; % No. of points
r = 0.618; % Dither
b = l + (1-r)*(ul-l);
c = l + r*(ul-l);
fb = LSFn(b);
f = LSFn(c);
x = zeros(np,1);
f = zeros(np,1);
x(1) = b;x(2) = c;
f(1) = fb;
f(2) = fc;
cnt = 2;
for cnt = 3:np,
    if fb>fc,
        b = c;
c = b;
        fb = fc;
    end;
    else
        ul = c;
c = b;
        fc = fb;
b = ul + (1-r)*(ul-l);
        fb = LSFn(b);
    end;
end;
[fmin,id] = min(f);
xmin = x(id);
xrng = ul-l;
figure;
FigureSet(1,'Slides');
u = 0:0.01:1;
h = plot(u,LSFn(u),'r',x,f,'.',xmin,fmin,'g.);
set(h(2),'MarkerSize',8);
set(h,'LineWidth',1.2);
set(h(3),'MarkerSize',15);
for c1 = 1:length(x),
    h = text(x(c1),f(c1)+0.01,num2str(c1));
    set(h,'HorizontalAlignment','Center');
    set(h,'VerticalAlignment','Bottom');
end;
FigureLatex;
```

J. McNames  Portland State University  ECE 4/557  Line Search Algorithms  Ver. 1.14  26

Quadratic Fit Line Search

1. Find \( \alpha_1, \alpha_2, \) and \( \alpha_3 \) such that \( f(\alpha_1) \geq f(\alpha_2) \) and \( f(\alpha_2) \leq f(\alpha_3) \)
2. Find \( \alpha^* \) as the minimum to the quadratic fit of \( f(\alpha_1), f(\alpha_2), \) and \( f(\alpha_3) \)
3. If \( \alpha^* > \alpha_2, \)
   - If \( f(\alpha^*) > f(\alpha_2), \) then \( \alpha_{\text{new}} = \{\alpha_1, \alpha_2, \alpha^*\} \)
   - If \( f(\alpha^*) \leq f(\alpha_2), \) then \( \alpha_{\text{new}} = \{\alpha_2, \alpha^*, \alpha_3\} \)
Else,
   - If \( f(\alpha^*) > f(\alpha_2), \) then \( \alpha_{\text{new}} = \{\alpha^*, \alpha_2, \alpha_3\} \)
   - If \( f(\alpha^*) \leq f(\alpha_2), \) then \( \alpha_{\text{new}} = \{\alpha_1, \alpha^*, \alpha_2\} \)
4. Loop to 2 until convergence
**Quadratic Fit Line Search Comments**

- Very popular
- Often includes a safeguard technique to handle ill-conditioning effects
  - Very fast convergence
  - Can be unstable if $f(\alpha)$ is not quasi-convex and safeguard technique is not used
  - Finds a local minimum

**Example 5: Quadratic Search**

- Resolution: 0.0071

**Example 5: MATLAB Code**

```matlab
function [] = QuadraticSearch;
ll = 0; % Lower limit
ul = 1; % Upper limit
np = 10; % No. of points
x = zeros(np,1);
f = zeros(np,1);
xl = ll; fl = LSFn(xl); % Left edge
xc = (ll+ul)/2; fc = LSFn(xc); % Center
xr = ul; fr = LSFn(xr); % Right edge
x(1) = xl; f(1) = fl;
x(2) = xc; f(2) = fc;
x(3) = xr; f(3) = fr;
for cnt = 4:np,
  if fc<fl & fc<fr, % Criteria okay - do quadratic search iteration
    a = xc - xr; b = xc^2 - xr^2;
    c = xr - xl; d = xr^2 - xl^2;
    e = xl - xc; f = xl^2 - xc^2;
    x = 0.5*(b*x + c*x + d*x + e)/a;
    fn = LSFn(fn);
    x(cnt) = x;
    f(cnt) = fn;
    if fn>fc,
    end
  end
end
```
Line Search Algorithm Comments

- If the No. iterations is known in advance, there is yet another algorithm (Fibonacci search) that is more efficient than the golden section.
- If the No. iterations is not known, golden section is more efficient than Fibonacci search, dichotomous search, or uniform search.
- If the second derivative can be calculated, there is another line search algorithm called Newton's method.
- How do you know when the algorithm has converged?
- There are several different popular criteria.
- In practice, the algorithms are often stopped after:
  - Some user-specified number of iterations.
  - The interval of uncertainty is reduced to some threshold.
Line Search Algorithm Comments Continued

- All of these methods may get stuck in a local minimum
- Uniform search is the least likely to get stuck, but is also the least efficient
- Hybrid searches are often used for better convergence

References
