Advanced Computational Methods in Statistics: Lecture 2
Optimisation

Axel Gandy

Department of Mathematics
Imperial College London
http://www2.imperial.ac.uk/~agandy

London Taught Course Centre
for PhD Students in the Mathematical Sciences
Autumn 2014
Lecture 2 - Optimisation

Part 1  Deterministic Optimisation
Part 2  EM algorithm
Part 3  LASSO and LARS
Part 4  NP-complete Problems
Part 5  Stochastic Approximation
Part I

Deterministic Optimisation

Introduction

Local Search Methods

Comments

Simulation study
Introduction

- \( f : A \rightarrow \mathbb{R}, A \subset \mathbb{R}^d \).
- Goal: Find \( x^* \in \mathbb{R}^d \) such that
  \[
  f(x^*) = \min_{x \in A} f(x)
  \]
- Example: finding the maximum likelihood estimator.
- Can have side conditions:
  \( g : A \rightarrow \mathbb{R}^q \) some function. Want to
  \[
  \text{minimise}_{x \in A} f(x) \text{ subject to } g(x) = 0
  \]
- Explicit solutions: Lagrange Multipliers.
  With inequality constraints: Kuhn-Tucker conditions.
Local Search Methods - No Side Conditions

- Main idea: create a sequence \( x_0, x_1, x_2, \ldots \) approximations to \( x^* \). Hopefully \( x_n \rightarrow x^* \) as \( n \rightarrow \infty \).

- Choice of algorithm depends on how many derivatives of \( f \) are available. Some Examples:

  - no derivatives: Nelder-Mead: works with \( d + 1 \) points that move towards \( x^* \) and then contract around it.
  - gradient \( \nabla f \): Gradient descent:
    \[
    x_n = x_{n-1} - \epsilon_n \nabla f(x_{n-1})
    \]
  - other methods: conjugate gradient, ...
  - gradient \( \nabla f \) + Hessian \( H \): Newton’s Method:
    \[
    x_n = x_{n-1} - H(f, x_{n-1})^{-1} \nabla f(x_{n-1})
    \]

Typically: the more derivatives are available the better the convergence rate.

- Global convergence only guaranteed if \( f \) is convex.
- If global convergence cannot be guaranteed, the very least one should do is use several starting values.
Optimisation with Side Conditions

\[ \text{minimise}_{x \in A} f(x) \text{ subject to } g(x) = 0 \]

- \( f \) linear, \( g \) linear: “linear programming”, Simplex algorithm
- \( f \) quadratic, \( g \) linear: quadratic programming
- More general structure:
  sequential quadratic programming algorithms may work:
  idea: approximate the problem locally by a quadratic programming problem.
  (implemented e.g. in the NAG library)
- More heuristic approach: put side condition into objective function, i.e. minimise \( f(x) + \lambda (g(x))^2 \) for some large \( \lambda > 0 \).
Comments

- Optimisation (in particular with side conditions and non-convex) can be a tough problem.
- Local search algorithms are not the only algorithms - many more approaches (simulated annealing, random optimisation, genetic optimisation).
- Many solutions have been developed that work well for specific problems.
- Try to use implementation of algorithms written by experts!
- Useful resource: Decision Tree for Optimisation Software http://plato.asu.edu/guide.html
Simulation study of various optimization algorithms

Various algorithms implemented in \textit{optim()} in R:

- Nelder-Mead: a simplex-based method.
- BFGS: quasi-Newton method (Broyden-Fletcher-Goldfarb-Shanno method)
- CG: a conjugate gradient method.
- L-BFGS-B: an algorithm that would allow bounds on the parameters.
- Simulated annealing with default settings.
- Simulated annealing with more steps and slower cooling.

Applied to 3 functions.
Example 1 - quadratic function

\[ f : \mathbb{R}^2 \rightarrow \mathbb{R}, f(x, y) = x^2 + y^2 \]

(global minimum at (0,0))
Applying standard R algorithms to the quadratic function

<table>
<thead>
<tr>
<th></th>
<th>N-M</th>
<th>BFGS</th>
<th>CG</th>
<th>L-BFGS-B</th>
<th>SANN1</th>
<th>SANN2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conv</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0%</td>
<td>2.73e-09</td>
<td>7.24e-28</td>
<td>7.42e-15</td>
<td>1.53e-41</td>
<td>9e-07</td>
<td>1.01e-07</td>
</tr>
<tr>
<td>25%</td>
<td>1.14e-07</td>
<td>1.61e-24</td>
<td>2.13e-14</td>
<td>3.96e-40</td>
<td>4.19e-05</td>
<td>2.47e-06</td>
</tr>
<tr>
<td>50%</td>
<td>2.46e-07</td>
<td>1.55e-23</td>
<td>3.76e-14</td>
<td>7.69e-40</td>
<td>8.21e-05</td>
<td>7.54e-06</td>
</tr>
<tr>
<td>75%</td>
<td>5.56e-07</td>
<td>7.11e-23</td>
<td>5.27e-14</td>
<td>1.37e-39</td>
<td>0.0002</td>
<td>1.4e-05</td>
</tr>
<tr>
<td>100%</td>
<td>5.04e-06</td>
<td>1.86e-21</td>
<td>8.91e-13</td>
<td>2.72e-39</td>
<td>0.000896</td>
<td>4.84e-05</td>
</tr>
<tr>
<td>neval</td>
<td>65.8</td>
<td>9.66</td>
<td>21.8</td>
<td>4.24</td>
<td>1e+04</td>
<td>1e+05</td>
</tr>
</tbody>
</table>

Table: Started from 100 different starting points in [-10,10]x[-10,10]. Conv=Proportion of successful convergence indicated; Quantiles of f(minimizer); neval=average number of function evaluations.
Example 2 - Rosenbrock Banana function

\[ f : \mathbb{R}^2 \to \mathbb{R}, \quad f(x, y) = (1 - x)^2 + 100(y - x^2)^2 \]

(global minimum at (1,1))
Applying standard R algorithms to the Banana function

<table>
<thead>
<tr>
<th></th>
<th>N-M</th>
<th>BFGS</th>
<th>CG</th>
<th>L-BFGS-B</th>
<th>SANN1</th>
<th>SANN2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conv</td>
<td>1</td>
<td>0.85</td>
<td>0.01</td>
<td>0.99</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0%</td>
<td>4.32e-08</td>
<td>9.93e-11</td>
<td>0.000187</td>
<td>1.98e-10</td>
<td>1.76e-06</td>
<td>8.25e-07</td>
</tr>
<tr>
<td>25%</td>
<td>3.8e-05</td>
<td>2.93e-08</td>
<td>0.0765</td>
<td>3.03e-08</td>
<td>0.000194</td>
<td>9.81e-06</td>
</tr>
<tr>
<td>50%</td>
<td>0.000747</td>
<td>3.95e-08</td>
<td>0.203</td>
<td>3.99e-08</td>
<td>0.000444</td>
<td>2.18e-05</td>
</tr>
<tr>
<td>75%</td>
<td>0.0489</td>
<td>4e-08</td>
<td>3.66</td>
<td>4e-08</td>
<td>0.00105</td>
<td>4.58e-05</td>
</tr>
<tr>
<td>100%</td>
<td>1e+06</td>
<td>1e+06</td>
<td>1e+06</td>
<td>1e+06</td>
<td>2.36</td>
<td>0.000199</td>
</tr>
<tr>
<td>neval</td>
<td>129</td>
<td>111</td>
<td>253</td>
<td>54</td>
<td>1e+04</td>
<td>1e+05</td>
</tr>
</tbody>
</table>

**Table:** Started from 100 different starting points in \([-10,10] \times [-10,10]\). Conv=Proportion of successful convergence indicated; Quantiles of \( f(\text{minimizer}) \); neval=average number of function evaluations.
Example 3

$f : \mathbb{R}^2 \rightarrow \mathbb{R}, f(x, y) = (x \sin(20y) + y \sin(20x))^2 \cosh(\sin(10x)x) + (x \cos(10y) - y \sin(10x))^2 \cosh(\cos(20y)y) + |x| + |y|$

(global minimum at (0,0))
### Applying standard R algorithms to Example 3

<table>
<thead>
<tr>
<th></th>
<th>N-M</th>
<th>BFGS</th>
<th>CG</th>
<th>L-BFGS-B</th>
<th>SANN1</th>
<th>SANN2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conv</td>
<td>0.99</td>
<td>1</td>
<td>0.21</td>
<td>0.78</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0%</td>
<td>2.4e-08</td>
<td>3.59e-20</td>
<td>4.57e-10</td>
<td>1.35e-14</td>
<td>0.00125</td>
<td>0.000241</td>
</tr>
<tr>
<td>25%</td>
<td>6.69</td>
<td>3.06e-18</td>
<td>2.79</td>
<td>8.25</td>
<td>0.0123</td>
<td>0.00148</td>
</tr>
<tr>
<td>50%</td>
<td>9.74</td>
<td>6.75</td>
<td>7.99</td>
<td>11.5</td>
<td>6.01</td>
<td>0.003</td>
</tr>
<tr>
<td>75%</td>
<td>13.4</td>
<td>11.9</td>
<td>11.7</td>
<td>29.4</td>
<td>10.9</td>
<td>0.00427</td>
</tr>
<tr>
<td>100%</td>
<td>181</td>
<td>263</td>
<td>269</td>
<td>178</td>
<td>22.7</td>
<td>18.4</td>
</tr>
<tr>
<td>neval</td>
<td>103</td>
<td>65.1</td>
<td>413</td>
<td>41.2</td>
<td>1e+04</td>
<td>1e+05</td>
</tr>
</tbody>
</table>

**Table:** Started from 100 different starting points in $[-10,10] \times [-10,10]$. Conv=Proportion of successful convergence indicated; Quantiles of $f(\text{minimizer})$; neval=average number of function evaluations.
Comments

- Functions that are “nice” (no local minima, maybe even convex): standard numerical algorithms work best, the more derivatives are used the better.
- Functions with local minima: Need to add noise to avoid getting trapped (needs tuning)
- General advice:
  - Use several starting values
  - Plot function (if possible)
Part II

The EM Algorithm

Introduction

Example - Mixtures

Theoretical Properties
EM Algorithm - Introduction

- Expectation-Maximisation algorithm; two steps:
  - E-step
  - M-Step

- General-purpose algorithm for maximum likelihood estimation in incomplete data problems.

- Main reference: Dempster et al. (1977)

- According to scholar.google.com: cited > 14000 times! (narrowly beating e.g. Cox “Regression Models and Life Tables” with roughly 13500 citations) [citation count on 19/1/2009]

- Most of the material in this chapter is based on McLachlan & Krishnan (2008). An overview article is Ng et al. (2004).
Situations in which the EM algorithm is applicable

- Incomplete data situations such as
  - missing data
  - truncated distributions
  - censored or grouped observations

- Statistical models such as
  - random effects
  - mixtures
  - convolutions
  - latent class/variable structures
  - ...

- Even if a problem appears not to be an incomplete data problem - writing it as such a problem can sometimes simplify its analysis (by simplifying the likelihood).
The EM algorithm - Notation

\( y \) observed data, incomplete data (corresponding r.v.: \( Y \))

\( g(\cdot, \psi) \) density of \( Y \)

\( \psi \) unknown parameter vector

Likelihood \( L(\psi) := g(y, \psi) \).

Want to find the MLE, i.e. maximise \( L \).

\( z \) missing data (corresponding r.v.: \( Z \))

\( x = (y, z) \) complete data (corresponding r.v.: \( X \))

\( g_c(\cdot; \psi) \) density of \( X \)

Note: \( g(y, \psi) = E[g_c(Y, Z; \psi)|Y = y] \)
$y$ observed, $z$ missing, $x = (y, z)$, $Q(\psi, \psi^k) = E[\log g_c(X; \psi)|Y = y; \psi^k]$

g density of $y$, $g_c$ density of $x$, $k = g_c/g$ density of $z|y$

The EM-algorithm

- Let $\psi^0$ be some initial value for $\psi$.
- For $k = 0, 1, \ldots$
  
  **E-step** Calculate $Q(\psi, \psi^k)$, where

  $$Q(\psi, \psi^k) = E[\log g_c(X; \psi)|Y = y; \psi^k]$$

  **M-step**

  $\psi^{k+1} = \text{argmax}_\psi Q(\psi, \psi^k)$

- Employ some convergence criterion (e.g. based on $\log g_c(x; \psi^k)$)

Note:

$$Q(\psi, \psi^k) = \int \log g_c(y, z; \psi)k(z|y; \psi)dz,$$

where $k(z|y; \psi) = g_c(y, z; \psi)/g(y; \psi)$ is the conditional density of $z$ given $Y = y$. 
The EM algorithm

- Let $\psi^0$ be some initial value for $\psi$.
- For $k = 0, 1, \ldots$
  
  **E-step** Calculate $Q(\psi, \psi^k)$, where

  \[
  Q(\psi, \psi^k) = E[\log g_c(X; \psi) | Y = y; \psi^k]
  \]

  **M-step**

  \[
  \psi^{k+1} = \arg\max_{\psi} Q(\psi, \psi^k)
  \]

- Employ some convergence criterion (e.g. based on $\log g_c(x; \psi^k)$)

Note:

\[
Q(\psi, \psi^k) = \int \log g_c(y, z; \psi) k(z|y; \psi) dz,
\]

where $k(z|y; \psi) = g_c(y, z; \psi)/g(y; \psi)$ is the conditional density of $z$ given $Y = y$. 

\[y \text{ observed, } z \text{ missing, } x = (y, z), \quad Q(\psi, \psi^k) = E[\log g_c(X; \psi) | Y = y; \psi^k] \]

$g$ density of $y$, $g_c$ density of $x$, $k = g_c/g$ density of $z|y$
y observed, z missing, \( \mathbf{x} = (\mathbf{y}, \mathbf{z}) \), \( Q(\psi, \psi^k) = \mathbb{E}[\log g_c(\mathbf{X}; \psi)|\mathbf{Y} = \mathbf{y}; \psi^k] \)

g density of \( \mathbf{y} \), \( g_c \) density of \( \mathbf{x} \), \( k = g_c/g \) density of \( \mathbf{z}|\mathbf{y} \)

### Monotonicity of the EM Algorithm

- Then \( \log g(\mathbf{y}; \psi) = \log(g_c(\mathbf{x}; \psi)) - \log k(\mathbf{x}|\mathbf{y}; \psi) \).
- Take expectations with density \( k(\mathbf{x}|\mathbf{y}; \psi) \)

\[
\log g(\mathbf{y}; \psi) = Q(\psi, \psi^k) - \mathbb{E}[\log k(\mathbf{X}|\mathbf{y}; \psi)|\mathbf{Y} = \mathbf{y}; \psi^k] =: H(\psi, \psi^k)
\]

- Thus

\[
\log g(\mathbf{y}; \psi^{k+1}) - \log g(\mathbf{y}; \psi^k) = \\
= (Q(\psi^{k+1}, \psi^k) - Q(\psi^k, \psi^k)) + (H(\psi^k, \psi^k) - H(\psi^{k+1}, \psi^k)) \\
\geq 0 \text{ (Def EM)} + (H(\psi^k, \psi^k) - H(\psi^{k+1}, \psi^k)) \geq 0 \text{ (next slide)}
\]

- Hence, \( \log g(\mathbf{y}; \psi^k) \uparrow \) as \( k \to \infty \).
Monotonicity of the EM Algorithm

- Then \( \log g(y; \psi) = \log(g_c(x; \psi)) - \log k(x|y; \psi) \).
- Take expectations with density \( k(x|y; \psi) \)

\[
\log g(y; \psi) = Q(\psi, \psi^k) - \underbrace{\mathbb{E}[\log k(X|y; \psi)|Y = y; \psi^k]}_{=: H(\psi, \psi^k)}
\]

- Thus

\[
\log g(y; \psi^{k+1}) - \log g(y; \psi^k) =
\]

\[
= \left( Q(\psi^{k+1}, \psi^k) - Q(\psi^k, \psi^k) \right) + \left( H(\psi^k, \psi^k) - H(\psi^{k+1}, \psi^k) \right)
\]

\[
\geq 0 \text{ (Def EM)} \quad \geq 0 \text{ (next slide)}
\]

- Hence, \( \log g(y; \psi^k) \uparrow \text{ as } k \to \infty. \)
Monotonicity of the EM Algorithm (cont)

- $H(\psi, \psi^k) = \mathbb{E}[\log k(X|y; \psi)|Y = y; \psi^k]$ is maximised at $\psi = \psi^k$.

- Indeed,

$$H(\psi^k, \psi^k) - H(\psi, \psi^k) = \mathbb{E}[-\log \frac{k(X|y; \psi)}{k(X|y; \psi^k)}|Y = y; \psi^k]$$

$$\geq -\log \mathbb{E}[\frac{k(X|y; \psi)}{k(X|y; \psi^k)}|Y = y; \psi^k] \quad \text{(Jensen’s inequality)}$$

$$= -\log \int \frac{k(X|y; \psi)}{k(X|y; \psi^k)} k(X|y; \psi^k) dx$$

$$= -\log \int k(x|y; \psi) dx = -\log(1) = 0$$

- Thus $H(\psi^k, \psi^k) - H(\psi^{k+1}, \psi^k) \geq 0$. 
y observed, z missing, x = (y, z), Q(ψ, ψ^k) = E[log gc(X; ψ)|Y = y; ψ^k] 
g density of y, gc density of x, k = gc/g density of z|y

Monotonicity of the EM Algorithm (cont)

- H(ψ, ψ^k) = E[log k(X|y; ψ)|Y = y; ψ^k] is maximised at ψ = ψ^k.

- Indeed,

\[
H(ψ^k, ψ^k) - H(ψ, ψ^k) = E[- \log \frac{k(X|y; ψ)}{k(X|y; ψ^k)}|Y = y; ψ^k]
\]

\[
\geq - \log E[\frac{k(X|y; ψ)}{k(X|y; ψ^k)}|Y = y; ψ^k] \quad \text{(Jensen’s inequality)}
\]

\[
= - \log \int \frac{k(X|y; ψ)}{k(X|y; ψ^k)} k(X|y; ψ^k) dx
\]

\[
= - \log \int k(x|y; ψ) dx = - \log(1) = 0
\]

- Thus, H(ψ^k, ψ^k) − H(ψ^{k+1}, ψ^k) ≥ 0.
The inequality for $h$ is a special form of the following general inequality:
Let $X$ be a r.v. with density $g$. Let $f$ be any other density. Then

$$E[\log(f(X))] \leq E[\log(g(X))]$$

Proof: Jensen’s inequality.
Generalised EM algorithm (GEM)

- The M-step may not have a close-form solution.
- It may not be feasible to find a global maximum of $Q(\cdot, \psi^k)$.
- Replace M-step by:

  choose $\psi^{k+1}$ such that

  $$Q(\psi^{k+1}, \psi^k) \geq Q(\psi^k, \psi^k)$$
Mixture Distribution

- Consider a mixture distribution
  - $\psi_1, \ldots, \psi_d \geq 0$, mixing proportions, $\sum_{i=1}^{d} \psi_i = 1$.
  - $f_1, \ldots, f_d$ component densities.

With probability $\psi_i$ sample from $f_i$.

Resulting density:

$$f(x) = \sum_{i=1}^{d} \psi_i f_i(x)$$

- We will assume that $f_1, \ldots, f_d$ are known, but that $\psi = (\psi_1, \ldots, \psi_d)$ is unknown.
Mixture of Normals

- $d = 3$
- $f_1 = \text{pdf of } N(3, 1)$
- $f_2 = \text{pdf of } N(-3, 0.5)$
- $f_3 = \text{pdf of } N(0, 1)$
- $\psi = (0.2, 0.1, 0.7)$
Mixture Distributions (cont.)

- Let $Y_1, \ldots, Y_n$ be an iid sample from the mixture distribution.
- The likelihood of the incomplete data is

$$g(y; \psi) = \prod_{i=1}^{n} \sum_{j=1}^{d} \psi_j f_j(y_i)$$

- Missing data: $Z_{ij}$ indicator variables of chosen component
- Complete density:

$$g_c(y, z; \psi) = \prod_{i=1}^{n} \prod_{j=1}^{d} (\psi_j f_j(y_i))^{z_{ij}}$$

Hence, the log-likelihood for the full data is

$$\log g_c(y, z; \psi) = \sum_{i=1}^{n} \sum_{j=1}^{d} z_{ij} \log(\psi_j) + C,$$

where $C$ does not depend on $\psi$. 
Mixture of Normals (cont.)

A sample
Mixture Distributions (cont.)

\[ Q(\psi, \psi^k) = \mathbb{E}[\log g_c(y, z; \psi); y, \psi^k] = \sum_{i=1}^{n} \sum_{j=1}^{d} \log(\psi_j) \mathbb{E}[z_{ij}; y, \psi^k] + C, \]

where

\[ \mathbb{E}[z_{ij}; y, \psi^k] = \frac{\psi_j^k f_j(y_i)}{\sum_{\nu} \psi_{\nu}^k f_{\nu}(y_i)} =: a_{ij} \]

We want to maximise

\[ Q(\psi, \psi^k) = \sum_{j=1}^{d} \left( \sum_{i=1}^{n} a_{ij} \right) \log(\psi_j) \]

subject to \( \sum \psi_j = 1 \). Using e.g. Lagrange multipliers and \( \sum_j a_{ij} = 1 \) one can see that the optimum is at

\[ \psi_j^{k+1} = \frac{1}{n} \sum_{i=1}^{n} a_{ij}, \quad j = 1, \ldots, d \]

Note: \( a_{ij} \) depends on \( \psi^k \)
Mixture Distributions (cont.)

\[ Q(\psi, \psi^k) = E[\log g_c(y, z; \psi); y, \psi^k] = \sum_{i=1}^{n} \sum_{j=1}^{d} \log(\psi_j) E[z_{ij}; y, \psi^k] + C, \]

where

\[ E[z_{ij}; y, \psi^k] = \frac{\psi_j^k f_j(y_i)}{\sum_\nu \psi_\nu^k f_\nu(y_i)} =: a_{ij} \]

We want to maximise

\[ Q(\psi, \psi^k) = \sum_j (\sum_i a_{ij}) \log(\psi_j) \]

subject to \( \sum \psi_j = 1 \). Using e.g. Lagrange multipliers and \( \sum_j a_{ij} = 1 \), one can see that the optimum is at

\[ \psi_j^{k+1} = \frac{1}{n} \sum_{i=1}^{n} a_{ij}, \quad j = 1, \ldots, d \]

Note: \( a_{ij} \) depends on \( \psi^k \)
Mixture Distributions (cont.)

\[ Q(\psi, \psi^k) = E[\log g_c(y, z; \psi); y, \psi^k] = \sum_{i=1}^{n} \sum_{j=1}^{d} \log(\psi_j) E[z_{ij}; y, \psi^k] + C, \]

where

\[ E[z_{ij}; y, \psi^k] = \frac{\psi_j^k f_j(y_i)}{\sum_{\nu} \psi_{\nu}^k f_{\nu}(y_i)} =: a_{ij} \]

We want to maximise

\[ Q(\psi, \psi^k) = \sum_{j=1}^{d} \left( \sum_{i=1}^{n} a_{ij} \right) \log(\psi_j) \]

subject to \( \sum \psi_j = 1 \). Using e.g. Lagrange multipliers and \( \sum_j a_{ij} = 1 \) one can see that the optimum is at

\[ \psi_j^{k+1} = \frac{1}{n} \sum_{i=1}^{n} a_{ij}, \quad j = 1, \ldots, d \]

Note: \( a_{ij} \) depends on \( \psi^k \)
## Mixture of Normals

### Applying the EM algorithm

<table>
<thead>
<tr>
<th>$k$</th>
<th>$\psi^k_1$</th>
<th>$\psi^k_2$</th>
<th>$\psi^k_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.333</td>
<td>0.333</td>
<td>0.333</td>
</tr>
<tr>
<td>2</td>
<td>0.261</td>
<td>0.115</td>
<td>0.624</td>
</tr>
<tr>
<td>3</td>
<td>0.225</td>
<td>0.097</td>
<td>0.678</td>
</tr>
<tr>
<td>4</td>
<td>0.216</td>
<td>0.094</td>
<td>0.69</td>
</tr>
<tr>
<td>5</td>
<td>0.214</td>
<td>0.094</td>
<td>0.692</td>
</tr>
<tr>
<td>6</td>
<td>0.213</td>
<td>0.094</td>
<td>0.693</td>
</tr>
<tr>
<td>7</td>
<td>0.213</td>
<td>0.094</td>
<td>0.693</td>
</tr>
<tr>
<td>8</td>
<td>0.213</td>
<td>0.094</td>
<td>0.693</td>
</tr>
<tr>
<td>9</td>
<td>0.213</td>
<td>0.094</td>
<td>0.693</td>
</tr>
<tr>
<td>10</td>
<td>0.213</td>
<td>0.094</td>
<td>0.693</td>
</tr>
</tbody>
</table>
Convergence Results

- We have already seen that $L(\psi^k)$ is increasing in $k$.
- Thus, if $L$ is bounded from above, $L(\psi^k)$ converges to some $L^*$.
- In almost all applications, $L^*$ is a stationary value, i.e. $L^* = L(\psi^*)$ for some $\psi^*$ such that
  \[
  \frac{\partial L(\psi)}{\partial \psi} \bigg|_{\psi=\psi^*} = 0
  \]
- Want $L^*$ to be a global maximum.
- However, general theorems will only guarantee that $L^*$ is a stationary point or a local maximum.
- There are some theorems that ensure convergence to a global maximum (assuming unimodality of $L$).
EM-Algorithm - Some Warnings

- There are (pathological?) examples, where the (Generalised) EM-algorithm does not work as expected, e.g. where there may
  - convergence to a saddle point,
  - convergence to a local MINIMUM,
  - \( L(\psi^k) \) converges, but \( \psi^k \) does not.
  (see (McLachlan & Krishnan, 2008, Section 3.6))

- Don’t trust the output of the EM result blindly! The very least you can do is try using different starting values.
Comments

- If the E-step cannot be computed analytically then Monte-Carlo techniques can be used. The resulting algorithm is often called “MCEM” algorithm. MCMC techniques (e.g. Gibbs sampling) can come into play here.

- For an overview of theoretical work concerning the convergence rate of the EM-algorithm see (McLachlan & Krishnan, 2008, Chapter 4).
Part III

LASSO and related algorithms

LASSO

Penalised Regression

LARS algorithm

Comments
Ordinary least squares (OLS)

- Linear Model:
  \[ Y = X\beta + \epsilon \]
  - \( Y \) vector of responses (n-dimensional)
  - \( X \in \mathbb{R}^{n \times p} \) matrix of covariates
  - \( \beta \in \mathbb{R}^p \) vector of regression coefficients (unknown)
  - \( \epsilon \) vector of errors, \( \mathbb{E}\epsilon = 0, \text{Cov}\epsilon = \sigma^2 I_n \)

- Classical approach (if \( n > p \)):
  \( \hat{\beta} \) is chosen as minimiser of the Sum of squares
  \[
  S(\beta) = \|Y - X\beta\|^2 = \sum_{i=1}^{n} (Y_i - (X\beta)_i)^2,
  \]
  where \( \|a\|^2 = \sum_i a_i^2 \).

- Many modern datasets (e.g. microarrays):
  high-dimensional covariates, even \( n \ll p \) (large \( p \) small \( n \))
  \( \Rightarrow \hat{\beta} \) is not uniquely identified!
Lasso

Lasso (’least absolute shrinkage and selection operator’) (Tibshirani, 1996)

\[ \hat{\beta} \text{ solution of} \]

\[
\begin{cases}
\| Y - X \beta \|^2_2 \to \min \\
\sum_{i=1}^{d} |\beta_i| \leq c
\end{cases}
\]

where \( c \in \mathbb{R} \) is a constant.

Remark:
Instead of side condition, can use \( L_1 \)-penalty

\[
\| Y - X \beta \|^2 + \lambda \sum_{i=1}^{d} |\beta_i| \to \min
\]

with a constant \( \lambda > 0 \).
Example: 

\[
\begin{pmatrix}
Y_1 \\
Y_2
\end{pmatrix} = \begin{pmatrix}
1 & 0 \\
0 & 1
\end{pmatrix} \begin{pmatrix}
\beta_1 \\
\beta_2
\end{pmatrix} + \epsilon.
\]

Using \( c = 1 \),

\[
\begin{cases}
(Y_1 - \beta_1)^2 + (Y_2 - \beta_2)^2 \rightarrow \min \\
|\beta_1| + |\beta_2| \leq 1
\end{cases}
\]
Example: \[
\begin{pmatrix}
Y_1 \\
Y_2
\end{pmatrix} = \begin{pmatrix}
1 & 0 \\
0 & 1
\end{pmatrix} \begin{pmatrix}
\beta_1 \\
\beta_2
\end{pmatrix} + \epsilon. \]

Using \( c = 1 \),

\[
\begin{cases}
(Y_1 - \beta_1)^2 + (Y_2 - \beta_2)^2 \rightarrow \min \\
|\beta_1| + |\beta_2| \leq 1
\end{cases}
\]
Example: \[
\begin{pmatrix}
Y_1 \\
Y_2
\end{pmatrix} = \begin{pmatrix}
1 & 0 \\
0 & 1
\end{pmatrix} \begin{pmatrix}
\beta_1 \\
\beta_2
\end{pmatrix} + \epsilon.
\]
Using \( c = 1 \),

\[
\begin{cases}
(Y_1 - \beta_1)^2 + (Y_2 - \beta_2)^2 \rightarrow \min \\
|\beta_1| + |\beta_2| \leq 1
\end{cases}
\]
Example: \[
\begin{pmatrix}
Y_1 \\
Y_2
\end{pmatrix} = \begin{pmatrix}
1 & 0 \\
0 & 1
\end{pmatrix} \begin{pmatrix}
\beta_1 \\
\beta_2
\end{pmatrix} + \epsilon. \]
Using \( c = 1 \),
\[
\begin{align*}
(Y_1 - \beta_1)^2 + (Y_2 - \beta_2)^2 & \rightarrow \min \\
|\beta_1| + |\beta_2| & \leq 1
\end{align*}
\]

\[
\hat{\beta} = (1, 0)^T
\]
Example: \[
\begin{pmatrix}
Y_1 \\
Y_2
\end{pmatrix} = \begin{pmatrix}
1 & 0 \\
0 & 1
\end{pmatrix} \begin{pmatrix}
\beta_1 \\
\beta_2
\end{pmatrix} + \epsilon.
\]
Using \( c = 1 \),
\[
\begin{cases}
(Y_1 - \beta_1)^2 + (Y_2 - \beta_2)^2 & \to \min \\
|\beta_1| + |\beta_2| & \leq 1
\end{cases}
\]

sparse solutions
Example: \(
\begin{pmatrix}
Y_1 \\
Y_2
\end{pmatrix}
= \begin{pmatrix} 1 & 0 \\
0 & 1
\end{pmatrix}
\begin{pmatrix} \beta_1 \\
\beta_2
\end{pmatrix} + \epsilon.
\)

Using \( c = 1 \),

\[
\begin{cases}
(Y_1 - \beta_1)^2 + (Y_2 - \beta_2)^2 \to \min \\
|\beta_1| + |\beta_2| \leq 1
\end{cases}
\]

\textbf{sparse solutions}
Penalised Regression

add regularity conditions on \( \beta \):

\[ p(\beta) \leq t \quad \text{for a constant } t \]

Examples:

\[ p(\beta) = \|\beta\|_0 = \# \{ i : \beta_i \neq 0 \} \quad \text{(best subset selection)} \]

\[ p(\beta) = \|\beta\|_1 = \sum_{i=1}^p |\beta_i| \quad \text{(LASSO, 'least absolute shrinkage and selection operator', see Tibshirani (1996))} \]

\[ p(\beta) = \|\beta\|_2^2 = \sum_{i=1}^p |\beta_i|^2 \quad \text{(ridge regression)} \]

Bridge Regression - families of penalties, e.g.:

\[ p_d(\beta) = \|\beta\|_d^\rho = \sum_{i=1}^p |\beta_i|^\rho \quad \text{where } 0 \leq \rho \leq 2 \]

\[ p_d(\beta) = \|\beta\|_d^\rho = \sum_{i=1}^p |\beta_i|^\rho \quad \text{elastic net} \]

Thus overall:

\[ S(\beta) \rightarrow \min \text{ subject to } p(\beta) \leq t \]

Alternatively: For some constant \( \lambda \),

\[ S(\beta) + \lambda p(\beta) \rightarrow \min \]
Finding the Solution of Penalised Regression

- Best subset regression: NP hard problem
- Convex optimisation problem for e.g. LASSO, Ridge → standard optimisation techniques could be used to find a solution.
- LARS/LASSO algorithm: faster algorithm for \( p(\beta) = \sum_{j=1}^{P} |\beta_j| \).
- How to choose the threshold \( t \) (or \( \lambda \))? Use cross-validation.
Least Angle Regression

- Introduced in Efron et al. (2004).
- Efficient stepwise algorithm.
- LASSO modification of the LARS algorithm: generates LASSO solutions for all thresholds $t$. 
Assumptions

Will assume that

- response has mean 0, i.e.
  \[ \sum_{i=1}^{n} Y_i = 0 \]

- covariates have mean 0 and length 1, i.e.
  \[ \sum_{i=1}^{n} X_{ij} = 0 \text{ and } \sum_{i=1}^{n} X_{ij}^2 = 1 \text{ for } j = 1, \ldots, p \]
LARS algorithm

Least Angle Regression (Efron et al., 2004)
A rough description:
Let $x_1, \ldots, x_p$ be the predictors, i.e. the columns of $X$.

- Start with all coefficient vectors equal to 0, i.e.
  \[ \beta_1 = 0, \ldots, \beta_p = 0 \]
- Let $A$ be the set of covariates that are most correlated with the current residual (initially the residual is the response).
- Initially, $A = \{x_{j_1}\}$.
- take the largest step possible in the direction of $x_{j_1}$ until another predictor $x_{j_2}$ enters $A$
- continue in the direction equiangular between $x_{j_1}$ and $x_{j_2}$ until a third predictor $x_{j_3}$ enters $A$
- continue in the direction equiangular between $x_{j_1}, x_{j_2}, x_{j_3}$ until a fourth predictor $x_{j_4}$ enters the most correlated set
- ...
Illustration of the Algorithm for $m = 2$ Covariates

-$\tilde{Y}$ projection of $Y$ onto the plane spanned by $x_1, x_2$.
-$\hat{\mu}_j$ estimate after $j$-th step.
Illustration of the Algorithm for $m = 2$ Covariates

- $\hat{Y}$ projection of $Y$ onto the plane spanned by $x_1, x_2$.
- $\hat{\mu}_j$ estimate after $j$-th step.
Illustration of the Algorithm for $m = 2$ Covariates

- $\hat{\mu}_0$ projection of $Y$ onto the plane spanned by $x_1, x_2$.
- $\hat{\mu}_j$ estimate after j-th step.
Illustration of the Algorithm for $m = 2$ Covariates

- $\tilde{Y}$ projection of $Y$ onto the plane spanned by $x_1, x_2$.
- $\hat{\mu}_j$ estimate after $j$-th step.
Illustration of the Algorithm for $m = 2$ Covariates

- $\tilde{Y}$ projection of $Y$ onto the plane spanned by $x_1, x_2$.
- $\hat{\mu}_j$ estimate after $j$-th step.
LARS - Diabetes Data

- from Efron et al. (2004)
- 442 patients
- covariates: age, sex, BMI, blood pressure, 6 blood serum measurements
- Response: “a measure of disease progression”
LASSO Modification of the LARS Algorithm

- LARS algorithm needs to be modified to yield all LASSO solutions
- essentially a modification is needed when a $\beta_j$ crosses 0.
LASSO - Diabetes Data

Note: now 12 steps instead of 10 with the LARS algorithms
Comments

- R-package: *lars*
- A LASSO fit has no more than \( n - 1 \) (centred) predictors with nonzero coefficient
- Number of operations needed:
  \[
  \begin{align*}
  p < n & : \mathcal{O}(p^3 + np^2) \\
  p > n & : \mathcal{O}(n^3 + n^2 p)
  \end{align*}
  \]
- Other algorithm: coordinate descent
Further recent approaches

- **Group Lasso**

\[ \|Y - X\beta\|^2 + \lambda \sum_j \left( \sum_{\nu \in K_j} |\beta_{\nu}|^2 \right)^{1/2} \rightarrow \min \]

where \( K_j \) are disjoint groups of variables and \( \lambda > 0 \).

- **Fused Lasso**

\[ \|Y - X\beta\|^2 + \lambda_1 \|\beta\|_1 + \lambda_2 \sum_{(i,j) \in A} |\beta_i - \beta_j| \rightarrow \min \]

where \( A \subset \{1, \ldots, n\}^2 \) and \( \lambda_1, \lambda_2 > 0 \).

- Recent “hot” topics: compressed sensing, matrix completion, stability selection.
Part IV

NP complete problems
NP-complete Problems 1

- Concerns decision problems
  - Input: 0-1 sequence of length \( n \)
  - Output: “yes” or “no”

- \( P \) = class of all decision problems that can be solved in at most polynomial time in \( n \) (on a Turing machine)

- NP is the set of decision problems for which a solution can be verified in polynomical time with some additional input of polynomial size.
  As a consequence: all problems in NP can be solved in exponential time.

- A decision problem is NP-complete if any other decision problem in NP can be reduced to it in polynomial time.
NP-complete Problems II

- There is a large number of NP-complete problems, e.g.
  - Travelling Salesman Problem
    Phrased as decision problem:
    Let $x$ be some fixed length. Is there a roundtrip for the salesman of length $\leq x$?
  - Best subset regression: (phrased as decision problem)
  - ....

(see http://en.wikipedia.org/wiki/List_of_NP-complete_problems for a long list)

- It is not clear if $P \neq NP$. This is one of the Millennium Prize Problems with a $1,000,000 prize, see http://www.claymath.org/millennium/P_vs_NP/
Part V

Stochastic Approximation

The Robbins-Monro Algorithm

Example
Stochastic Approximation

Robbins-Monro/Kiefer-Wolfowitz algorithm

- Want to minimise $z(\theta)$ over $\Theta \subset \mathbb{R}^d$
  
  e.g.: $z(\theta) = \mathbb{E}(f(X, \theta))$, where $X$ is a random vector with known distribution and $f$ is a known function.

- Iterative algorithm: successive approximations $\theta_1, \theta_2, \ldots$

- Standard approach - Gradient Descent:

$$\theta_{n+1} = \theta_n - \epsilon_n \nabla z(\theta_n)$$

for some deterministic sequence $\epsilon_n$.

- Assume that we cannot evaluate $\nabla z(\theta)$ directly.

- Available $Y_n$ “close to” $\nabla z(\theta)$.

  In the Robbins-Monro-algorithm, see Robbins & Monro (1951), one assumes

$$Y_n = \nabla z(\theta) + \epsilon$$

with $\mathbb{E}(\epsilon) = 0$.

Iteration:

$$\theta_{n+1} = \theta_n - \epsilon_n Y_{n+1},$$
How to choose $\epsilon_n$?

$$\theta_{n+1} = \theta_n - \epsilon_n Y_{n+1}$$

Requirements on $\epsilon_n$

- To be able to reach any point:
  $$\sum_{n=0}^{\infty} \epsilon_n = \infty$$
  (assuming $E Y_n$ is bounded)

- To get convergence of $\theta_n$, need
  $$\epsilon_n \to 0$$
  (assuming $\text{Var}(Y_n) \not\to 0$):

Canonical choice: $\epsilon_n = an^{-\delta}$ for some $0 < \delta \leq 1$ and some $a > 0$. 
How can one obtain $Y_n$?

Some options for $z(\theta) = E(f(X, \theta))$:

- **finite differences** (Kiefer-Wolfowitz algorithm, Kiefer & Wolfowitz (1952)): Let $M(\theta)$ be such that $E(M(\theta)) = z(\theta)$

$$Y_{n,i} = \frac{M(\theta + c_n) - M(\theta - c_n)}{2c_n}$$

- **Infinitesimal Perturbation Analysis (IPA)**

  Main Idea: often $\frac{\partial}{\partial \theta} z(\theta) = \frac{\partial}{\partial \theta} E(f(X, \theta)) = E(\frac{\partial}{\partial \theta} f(X, \theta))$.

Define $Y_n$ as Monte Carlo estimate of the RHS:

$$Y_n = \frac{1}{m} \sum_{i=1}^{m} \frac{\partial}{\partial \theta} f(X^i, \theta)$$

where $X, X^1, \ldots, X^m$ is iid.
Stochastic-Approximation - Example

based on (Asmussen & Glynn, 2007, Section VIII 5a)

- Minimise

\[ z(\theta) = E[\max(\theta X_1 + X_2, (1 - \theta)X_3)], \]

where \( X_i \sim Gamma(2, 2/i), \) \( i = 1, \ldots, 3 \) are independent. (the correct minimiser is 0.625)

- Estimate \( z'(\theta_n) \) by MC simulation:

Note \( z'(\theta) = E[g(X_1, X_2, X_3, \theta)], \) where

\[ g(x_1, x_2, x_3, \theta) = \begin{cases} x_1 & \theta x_1 + x_2 \geq (1 - \theta)x_3 \\ -x_3 & \text{otherwise} \end{cases} \]

Use the estimator

\[ Y_n = \frac{1}{m} \sum_{i=1}^{m} g(X_1^i, X_2^i, X_3^i, \theta) \]

where \( X_j^i \sim X_j, \) \( j = 1, \ldots, 3, \) \( i = 1, \ldots, m \) are independent.
Stochastic-Approximation - one run

\[ m = 10, \epsilon_n = n^{-\delta}/10, \theta_0 = 0.4 \]

\[ \delta = 0.4 \]
Stochastic-Approximation - Sensitivity to $\theta$

Same parameters as before

$\delta = 0.4$

$\delta = 0.6$

$\delta = 0.8$
Stochastic Approximation - Comments

- Very general class of algorithms - related to stochastic control.
- Several Parameters need tuning (best done on a case by case basis)
  - How many samples $m$ to take at each step? Should $m$ depend on $n$?
  - What $\epsilon_n$ to use?
- A lot of theoretical work has been concerned with establishing theoretical properties of these algorithms.

Main idea:
- Relate the sequence the sequence $\theta_n$ to the solution $\theta(t)$ of the deterministic dynamical system

$$\frac{\partial}{\partial t} \theta(t) = -\nabla z(\theta(t))$$

and use martingale theory to analyse the differences.

See e.g. Kushner & Yin (2003) for details.

- A shorter introduction can be found in e.g. Asmussen & Glynn (2007).
Part VI

Appendix
Topics in the coming lectures:

- MCMC methods
- Bootstrap
- Particle Filtering


References II

