AD Model Builder introduction course

What happens internally

AD Model Builder foundation

anders@nielsensweb.org
It is all about minimizing functions

- Want to find the parameters $\theta = (\theta_1, \ldots, \theta_n)$ that makes the observations most likely.
- Equivalent to minimizing the negative log likelihood w.r.t. $\theta$
  $$\hat{\theta} = \arg\min_{\theta} \ell(y|\theta)$$
- If the dimension of $\theta$ is low (say $n$ less than 5 or 10) any method can be used (grid search, random search, finite difference approximations, ...)
- AD Model Builder is capable of handling much larger problems
- Important for fixed effects models, and even more for random effects models
- AD Model Builder uses a quasi-Newton minimizer aided by automatic differentiation
- Here we will try to explain what that is, and why that is important
Quasi-Newton minimizer

A Newton minimizer is an iterative algorithm

Each step assumes that the function $\ell(x, \theta)$ can be approximated locally by a quadratic function

It uses the first $\ell'_\theta$ and second $\ell''_\theta$ derivatives to find the minimum

Instead of calculating $\ell''_\theta$ at every step, a quasi-Newton minimizer uses successive first derivatives $\ell'_\theta$ to approximate $\ell''_\theta$.

Bottom line: We need a fast and accurate way to calculate $\ell'_\theta$
Finite difference: Simple, inaccurate, and slow

- Algorithm: The $i$'th element in $\ell'_\theta$ is calculated by
  - Add a small number $\Delta \theta_i$ to the $i$'th element of $\theta$ to get $\tilde{\theta}_i$
  - Calculate $(\ell'_\theta)_i \approx \frac{\ell(\tilde{\theta}_i, x) - \ell(\theta, x)}{\Delta \theta_i}$

- Notice: all that is required is that we can evaluate $\ell(\theta, x)$ at any point
- Notice: it is an approximation
- Notice: it will be expensive if the dimension of $\theta$ is high

Analytical: The best thing when possible

- Situations where we can find a nice analytical expression for $\ell'_\theta$ are:
  - Fast
  - Accurate
  - Extremely rare
Automatic differentiation: Fast and accurate

- We need to write a program to compute $\ell(\theta, x)$ anyway
- A computer program is a long list of simple operations: 
  `+`, `-`, `*`, `/`, `exp`, `log`, `sin`, `cos`, `tan`, `sqrt`, and so on
- We know how to derive each of these operations
- The chain rule tells us how to combine: $(f(g(x)))' = f'(g(x))g'(x)$
- So if the computer is instructed to:
  - keep track of all the simple operations used when calculating $\ell(\theta, x)$
  - use the simple derivative formulas and the chain rule
- Then once $\ell(\theta, x)$ is computed, we also have $\ell'_\theta$ with a minimum of extra calculations
- This is fast and accurate, and the difficult part is built into AD Model Builder(!)
- To get a better understanding consider the following code, which is modified from a larger example by Uffe Høgsbro Thygesen.
```cpp
#include <math.h>
#include <iostream.h>

class result {
  private: double v,d;
  public: result(){v = 0;d= 0;};
           result(double val){v = val; d = 0;};
           result(double val,double der){v = val; d = der;};
           double Value(){return v;};
           double Deriv(){return d;};
};

class parameter: public result {
  public: parameter(double pval) : result(pval,1.0) {};
          parameter() : result(0.0,1.0) {};
};

result sin(result n){
  return result(sin(n.Value()), cos(n.Value())*n.Deriv());
};

result operator*(result n1,result n2){
  return(result(n1.Value()*n2.Value(), n1.Deriv()*n2.Value() + n2.Deriv()*n1.Value()));
};

ostream& operator<<(ostream& o,result n){
  o << n.Value() << " (Derivative: " << n.Deriv() << ") " ;
  return o;
}

int main(int argc, char* argv[]){
  parameter theta(2);
  result y;
  y = sin(theta*theta);
  cout << "The result is " << y << endl;
}

The result is -0.756802 (Derivative: -2.61457)
```
Forward and reverse mode

- Forward mode is easy to understand and implement
- Not efficient when $\theta$ is high dimensional
Requires recording a stack of all operations

- Efficient in number of operations
- AD Model Builder uses reverse mode
- Except for random effects models where a combo of forward and reverse mode is used
This should be a help in understanding why ...

- we should careful about statement like:
  \[
  \text{if}(\theta<7.0)\{nll=\ldots\};\text{else}\{nll=\ldots;\}
  \]

- we can sometimes observe the memory requirements growing rather big if do a lot of iterative calculations

- a 'double' is different from a 'dvariable', a 'dvector' is different from a 'dvar_vector', ...

- we cannot do coding like:
  \[
  \text{dvariable } x=5; \ldots \text{ double } y; y=x; \ldots x=y;
  \]

- it is usually better to use the built-in functions in AD Model Builder than coding them yourself
Exercises

Exercise 1: Add the functionality to handle the plus operator, division operator and the cosine function to the program on page 6. Evaluate $f'(2)$, where:

$$f(x) = \frac{\sin(\sin(x^2) + \cos(x))}{x^2}$$

Solution:

The result is $-0.230474$ (Derivative: $-0.110843$)
Exercise 2: AD Model Builder has a facility to check the automatic derivatives by comparing them to the finite difference approximations. It can be started by pressing `ctrl-c` while a minimizer is running, or by starting the program with the flag `progname -dd 1` which will start the derivative checker after the first function evaluation. Verify the derivatives for one of the previous programs (for instance the 1D diffusion model).

Solution:

```bash
an@ch-pcb-an:~/talks/admbcourse$ ./turbot -dd 1

Initial statistics: 3 variables; iteration 0; function evaluation 0
Function value 1.3294890e+02; maximum gradient component mag -1.3054e+02
Var Value Gradient |Var Value Gradient |Var Value Gradient
1 0.00000 -2.06761e-03 | 2 6.90776 8.30058e+01 | 3 0.00000 -1.30543e+02

Enter index(1...3) of derivative to check. To check all derivatives, enter 0: To quit enter -1: 0
Checking all derivatives. Press X to terminate checking.
Enter step size (to quit derivative checker, enter 0): 1.0e-6

X Function Analytical Finite Diff; Index
1.90075e-08 1.32929e+02 -2.07065e-03 -2.07085e-03 ; 1
6.90699e+00 1.32929e+02 8.29584e+01 8.29584e+01 ; 2
1.20007e-03 1.32929e+02 -1.30223e+02 -1.30223e+02 ; 3

an@ch-pcb-an:~/talks/admbcourse$
```