AD Master Class: Advanced Adjoint Techniques

Computing Hessians

Viktor Mosenkis
viktor.mosenkis@nag.co.uk

Experts in numerical algorithms and HPC services

25 November 2020

AD Masterclass Schedule and Remarks

- AD Masterclass Schedule
  - 1 October 2020 | Checkpointing and external functions 1
  - 15 October 2020 | Checkpointing and external functions 2
  - 29 October 2020 | Guest lecture by Prof Uwe Naumann on Advanced AD topics in Machine Learning
  - 12 November 2020 | Monte Carlo
  - 19 November 2020 | Guest lecture by Prof Uwe Naumann on Adjoint Code Design Patterns applied to Monte Carlo
  - 25 November 2020 |Computing Hessians
- Remarks
  - Please submit your questions via the questions panel at any time during this session, these will be addressed at the end.
  - A recording of this session, along with the slides will be shared with you in a day or two.

Dialogue

We want this webinar series to be interactive (even though it's hard to do)

- We want your feedback, we want to adapt material to your feedback
- Please feel free to contact us via email to ask questions at any time
- We'd love to reach out offline, discuss what's working, what to spend more time on
- For some orgs, may make sense for us to do a few bespoke sessions

- This is an advanced course
- We assume that you are familiar with the material from the first Masterclass series
- You will get access to the materials from the first Masterclass series via email in a day or two
- Also it is not a pre-requisite we recommend to review the material from the previous series
- We will try to give references to the previous Masterclass series whenever possible

Outcomes

- Discuss four different second-order models
  - their advantages and disadvantages
  - usage of the models to compute the full Hessian
- How to reuse external function implemented for the first order adjoint code in the second order model

Second-Order Models

To compute second-order derivatives we can simply apply tangent or adjoint model to tangent/adjoint model, yielding the following four second-order models

- tangent over tangent (forward over forward)
- tangent over adjoint (forward over reverse)
- adjoint over tangent (reverse over forward)
- adjoint over adjoint (reverse over reverse)

Tangent over Tangent Model (Second-Order Tangent)

$$F : \mathbb{R}^n \to \mathbb{R}^m, \quad \boldsymbol{y} = F(\boldsymbol{x})$$

A second-order tangent code
$\dot{\tilde{F}} : \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^m \times \mathbb{R}^m \times \mathbb{R}^m \times \mathbb{R}^m$,

$$(\boldsymbol{y}, \tilde{\boldsymbol{y}}, \dot{\boldsymbol{y}}, \dot{\tilde{\boldsymbol{y}}})^T = \dot{\tilde{F}}(\boldsymbol{x}, \tilde{\boldsymbol{x}}, \dot{\boldsymbol{x}}, \dot{\tilde{\boldsymbol{x}}})$$

computes a mixture of first and second derivative information alongside with the function value as follows:

$$\begin{aligned}
\boldsymbol{y} &= F(\boldsymbol{x}) \\
\tilde{\boldsymbol{y}} &= F'(\boldsymbol{x}) \cdot \tilde{\boldsymbol{x}} \\
\dot{\boldsymbol{y}} &= F'(\boldsymbol{x}) \cdot \dot{\boldsymbol{x}} \\
\dot{\tilde{\boldsymbol{y}}} &= \dot{\boldsymbol{x}}^T \cdot F''(\boldsymbol{x}) \cdot \tilde{\boldsymbol{x}} + F'(\boldsymbol{x}) \cdot \dot{\tilde{\boldsymbol{x}}}
\end{aligned}$$

Tangent over Tangent: Accumulation of Hessian $m = 1$

$$F : \mathbb{R}^n \to \mathbb{R}, \quad \boldsymbol{y} = F(\boldsymbol{x})$$

As $m = 1$ $F''(\boldsymbol{x})$ is an n by n matrix.
In the tangent over tangent model the Hessian can be extracted from the computation of

$$\tilde{\dot{\boldsymbol{y}}} = \underbrace{\dot{\boldsymbol{x}}^T}_{\in\mathbf{R}^n} \cdot F''(\boldsymbol{x}) \cdot \underbrace{\tilde{\boldsymbol{x}}}_{\in\mathbf{R}^n} + F'(\boldsymbol{x}) \cdot \tilde{\dot{\boldsymbol{x}}}$$

Setting $\dot{\boldsymbol{x}} = e_i$ and $\tilde{\boldsymbol{x}} = e_j$, while keeping $\tilde{\dot{\boldsymbol{x}}} = 0$ yields $\tilde{\dot{\boldsymbol{y}}} = F''(\boldsymbol{x})_{i,j}$ (the i, j-th entry of the Hessian).
Hence computation of Hessian with tangent over tangent is $O(n^2)$ as we need to range over the Cartesian basis vectors of $\mathbb{R}^n$ in both $\dot{\boldsymbol{x}}$ and $\tilde{\boldsymbol{x}}$.
Hessian matrix is symmetric so we need to compute only upper or lower triangular part of the Hessian

Tangent over Tangent: Implementation

In dco/c++ tangent over tangent is implemented via

```
using DCO_BASE_M = dco::gt1s<double>;
using DCO_BASE_T = DCO_BASE_M::type;
using DCO_M = dco::gt1s<DCO_BASE_T>;
```

- $x \rightarrow$ `dco::value(dco::value(x))`
- $\dot{x} \rightarrow$ `dco::value(dco::derivative(x))`
- $\tilde{x} \rightarrow$ `dco::derivative(dco::value(x))`
- $\tilde{\dot{x}} \rightarrow$ `dco::derivative(dco::derivative(x))`

Tangent over Tangent: Driver

```
for (size_t i = 0; i < n; i++) {
  for (size_t j = 0; j < n; j++) {
    dco::value(dco::derivative(x[i])) = 1.0; //\dot{x}
    dco::derivative(dco::value(x[j])) = 1.0; //\tilde{x}

    foo(n, x, y);
    //H[i][j] = \tilde{\dot{y}}
    Hess[i][j] = dco::derivative(dco::derivative(y));

    dco::value(dco::derivative(x[i])) = 0.0; //\dot{x}
    dco::derivative(dco::value(x[j])) = 0.0; //\tilde{x}
  }
}
```

Tangent over Adjoint Model (Second-Order Adjoint)

$$F : \mathbb{R}^n \to \mathbb{R}^m, \quad \boldsymbol{y} = F(\boldsymbol{x})$$

A second-order adjoint code
$\dot{\bar{F}} : \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}^m \times \mathbb{R}^m \times \mathbb{R}^n \times \mathbb{R}^n$,

$$(\boldsymbol{y}, \dot{\boldsymbol{y}}, \bar{\boldsymbol{x}}, \dot{\bar{\boldsymbol{x}}})^T = \dot{\bar{F}}(\boldsymbol{x}, \dot{\boldsymbol{x}}, \bar{\boldsymbol{y}}, \dot{\bar{\boldsymbol{y}}})$$

computes a mixture of first and second derivative information alongside with the function value as follows:

$$\begin{aligned}
\boldsymbol{y} &= F(\boldsymbol{x}) \\
\dot{\boldsymbol{y}} &= F'(\boldsymbol{x}) \cdot \dot{\boldsymbol{x}} \\
\bar{\boldsymbol{x}} &= F'(\boldsymbol{x})^T \cdot \bar{\boldsymbol{y}} \\
\dot{\bar{\boldsymbol{x}}} &= \bar{\boldsymbol{y}}^T \cdot F''(\boldsymbol{x}) \cdot \dot{\boldsymbol{x}} + F'(\boldsymbol{x})^T \cdot \dot{\bar{\boldsymbol{y}}}
\end{aligned}$$

Tangent over Adjoint: Accumulation of Hessian $m = 1$

$$F : \mathbb{R}^n \to \mathbb{R}, \quad \boldsymbol{y} = F(\boldsymbol{x})$$

As $m = 1$ $F''(\boldsymbol{x})$ is an n by n matrix.

In the tangent over adjoint model the Hessian can be extracted from the computation of

$$\dot{\bar{\boldsymbol{x}}} = \underbrace{\bar{\boldsymbol{y}}^T}_{\in \mathbf{R}} \cdot F''(\boldsymbol{x}) \cdot \underbrace{\dot{\boldsymbol{x}}}_{\in \mathbf{R}^n} + F'(\boldsymbol{x})^T \cdot \dot{\bar{\boldsymbol{y}}}$$

Setting $\bar{\boldsymbol{y}} = 1$ and $\dot{\boldsymbol{x}} = e_i$, while keeping $\dot{\bar{\boldsymbol{y}}} = 0$ yields $\dot{\bar{\boldsymbol{x}}} = F''(\boldsymbol{x})_i$ (the i-th row (or column) of the Hessian due to symmetry).

Hence computation of Hessian with tangent over adjoint is $O(n)$ as we need to range over the Cartesian basis vectors of $\mathbb{R}^n$ in $\dot{\boldsymbol{x}}$.

Tangent over Adjoint: Implementation

In dco/c++ tangent over adjoint is implemented via

```
using DCO_BASE_M = dco::gt1s<double>;
using DCO_BASE_T = DCO_BASE_M::type;
using DCO_M = dco::ga1s<DCO_BASE_T>;
```

- $x \rightarrow$ `dco::value(dco::value(x))`
- $\bar{x} \rightarrow$ `dco::value(dco::derivative(x))`
- $\dot{x} \rightarrow$ `dco::derivative(dco::value(x))`
- $\dot{\bar{x}} \rightarrow$ `dco::derivative(dco::derivative(x))`

Tangent over Adjoint: Driver

```
for (size_t k = 0; k < n; k++) {
  DCO_M::global_tape = DCO_TAPE_T::create(o);

  DCO_M::global_tape->register_variable(x);
  dco::derivative(dco::value(x[k])) = 1.0;//\dot{x}_i = 1

  foo(n, x, y); // Record the tape

  dco::value(dco::derivative(y)) = 1.0;//\bar{\Y}

  DCO_M::global_tape->interpret_adjoint();
  for (int i = 0; i < n; i++)
    Hess[k][i] = dco::derivative(dco::derivative(x[i]));

  dco::derivative(dco::value(x[k])) = 0.0;//\dot{x}_i = 0
  DCO_TAPE_T::remove(DCO_M::global_tape);
}
```

Adjoint over Tangent Model

$$F : \mathbb{R}^n \to \mathbb{R}^m, \quad \boldsymbol{y} = F(\boldsymbol{x})$$

A second-order adjoint over tangent model
$\bar{\dot{F}} : \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}^m \times \mathbb{R}^m \times \mathbb{R}^n \times \mathbb{R}^n$,

$$(\boldsymbol{y}, \dot{\boldsymbol{y}}, \bar{\boldsymbol{x}}, \bar{\dot{\boldsymbol{x}}})^T = \bar{\dot{F}}(\boldsymbol{x}, \dot{\boldsymbol{x}}, \bar{\boldsymbol{y}}, \bar{\dot{\boldsymbol{y}}})$$

computes a mixture of first and second derivative information alongside with the function value as follows:

$$\begin{aligned}
\boldsymbol{y} &= F(\boldsymbol{x}) \\
\dot{\boldsymbol{y}} &= F'(\boldsymbol{x}) \cdot \dot{\boldsymbol{x}} \\
\bar{\boldsymbol{x}} &= F'(\boldsymbol{x})^T \cdot \bar{\boldsymbol{y}} + \bar{\dot{\boldsymbol{y}}}^T \cdot F''(\boldsymbol{x}) \cdot \dot{\boldsymbol{x}} \\
\bar{\dot{\boldsymbol{x}}} &= F'(\boldsymbol{x})^T \cdot \bar{\dot{\boldsymbol{y}}}
\end{aligned}$$

Adjoint over Tangent: Accumulation of Hessian $m = 1$

$$F : \mathbb{R}^n \to \mathbb{R}, \quad \boldsymbol{y} = F(\boldsymbol{x})$$

As $m = 1$ $F''(\boldsymbol{x})$ is an n by n matrix.

In the adjoint over tangent model the Hessian can be extracted from the computation of

$$\bar{\boldsymbol{x}} = F'(\boldsymbol{x})^T \cdot \bar{\boldsymbol{y}} + \underbrace{\bar{\dot{\boldsymbol{y}}}^T}_{\in \mathbf{R}} \cdot F''(\boldsymbol{x}) \cdot \underbrace{\dot{\boldsymbol{x}}}_{\in \mathbf{R}^n}$$

Setting $\bar{\dot{\boldsymbol{y}}} = 1$ and $\dot{\boldsymbol{x}} = e_i$, while keeping $\bar{\boldsymbol{y}} = 0$ yields $\bar{\boldsymbol{x}} = F''(\boldsymbol{x})_i$ (the i-th row (or column) of the Hessian due to symmetry).

Hence computation of Hessian with tangent over adjoint is $O(n)$ as we need to range over the Cartesian basis vectors of $\mathbb{R}^n$ in $\dot{\boldsymbol{x}}$.

Adjoint over Tangent: Implementation

In dco/c++ adjoint over tangent is implemented via

```
using DCO_BASE_M = dco::ga1s<double>;
using DCO_BASE_T = DCO_BASE_M::type;
using DCO_M = dco::gt1s<DCO_BASE_T>;
```

- $x \to$ `dco::value(dco::value(x))`
- $\dot{x} \to$ `dco::value(dco::derivative(x))`
- $\bar{x} \to$ `dco::derivative(dco::value(x))`
- $\dot{\bar{x}} \to$ `dco::derivative(dco::derivative(x))`

Adjoint over Tangent: Driver

```
using B_M = dco::ga1s<double>;
for (size_t k = 0; k < n; k++) {
  B_M::global_tape = DCO_TAPE_T::create(o);
  //\bar{x}
  B_M::global_tape->register_variable(dco::value(x));
  dco::value(dco::derivative(x[k])) = 1.0; //\dot{x}_i

  foo(n, x, y);
  dco::derivative(dco::derivative(y)) = 1.0;//\bar{\dot{y}}

  B_M::global_tape->interpret_adjoint();

  for (int i = 0; i < n; i++) //\bar{x}
    Hess[k][i] = dco::derivative(dco::value(x[i]));
  dco::value(dco::derivative(x[k])) = 0.0;//\dot{x}_i = 0
  DCO_TAPE_T::remove(B_M::global_tape);
}
```

Adjoint over Adjoint Model

$$F : \mathbb{R}^n \to \mathbb{R}^m, \quad \boldsymbol{y} = F(\boldsymbol{x})$$

A second-order adjoint over adjoint model
$\bar{\tilde{F}} : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^m \times \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^n,$

$$(\boldsymbol{y}, \bar{\boldsymbol{x}}, \bar{\tilde{\boldsymbol{y}}}, \tilde{\boldsymbol{x}})^T = \bar{\tilde{F}}(\boldsymbol{x}, \bar{\boldsymbol{y}}, \bar{\tilde{\boldsymbol{x}}}, \tilde{\boldsymbol{y}})$$

computes a mixture of first and second derivative information alongside with the function value as follows:

$$\begin{aligned}
\boldsymbol{y} &= F(\boldsymbol{x}) \\
\bar{\boldsymbol{x}} &= F'(\boldsymbol{x})^T \cdot \bar{\boldsymbol{y}} \\
\bar{\tilde{\boldsymbol{y}}} &= F'(\boldsymbol{x}) \cdot \bar{\tilde{\boldsymbol{x}}} \\
\tilde{\boldsymbol{x}} &= F'(\boldsymbol{x})^T \cdot \tilde{\boldsymbol{y}} + \bar{\boldsymbol{y}}^T \cdot F''(\boldsymbol{x}) \cdot \bar{\tilde{\boldsymbol{x}}}
\end{aligned}$$

Adjoint over Adjoint: Accumulation of Hessian $m = 1$

$$F : \mathbb{R}^n \rightarrow \mathbb{R}, \quad \boldsymbol{y} = F(\boldsymbol{x})$$

As $m = 1$ $F''(\boldsymbol{x})$ is an n by n matrix.

In the adjoint over adjoint model the Hessian can be extracted from the computation of

$$\tilde{\boldsymbol{x}} = F'(\boldsymbol{x})^T \cdot \tilde{\boldsymbol{y}} + \underbrace{\bar{\boldsymbol{y}}^T}_{\in \mathbf{R}} \cdot F''(\boldsymbol{x}) \cdot \underbrace{\tilde{\bar{\boldsymbol{x}}}}_{\in \mathbf{R}^n}$$

Setting $\bar{\boldsymbol{y}} = 1$ and $\tilde{\bar{\boldsymbol{x}}} = e_i$, while keeping $\tilde{\boldsymbol{y}} = 0$ yields $\tilde{\boldsymbol{x}} = F''(\boldsymbol{x})_i$ (the i-th row (or column) of the Hessian due to symmetry).

Hence computation of Hessian with tangent over adjoint is $O(n)$ as we need to range over the Cartesian basis vectors of $\mathbb{R}^n$ in $\tilde{\bar{\boldsymbol{x}}}$.

No advantage using the adjoint model for the second time

Why there is no advantage in using adjoint model twice?

The original function is

$$F : \mathbb{R}^n \to \mathbb{R}, \quad \boldsymbol{y} = F(\boldsymbol{x}) = F(x_1, \ldots, x_n)$$

The corresponding derivative function (that can be computed by one evaluation of the first order adjoint model) is

$$G : \mathbb{R}^n \to \mathbb{R}^n, \quad (y_1, \ldots, y_n)^T = \left(\frac{\partial F(\boldsymbol{x})}{\partial x_1}, \ldots, \frac{\partial F(\boldsymbol{x})}{\partial x_n}\right)$$

The second application of the adjoint model we differentiate G. The input and output dimension of G are equal, therefore it doesn't matter what models is used for the second time.

Adjoint over Adjoint: Implementation

In dco/c++ adjoint over tangent is implemented via

```
using DCO_BASE_M = dco::ga1s<double>;
using DCO_BASE_T = DCO_BASE_M::type;
using DCO_M = dco::ga1s<DCO_BASE_T>;
```

- $x \rightarrow$ `dco::value(dco::value(x))`
- $\bar{x} \rightarrow$ `dco::value(dco::derivative(x))`
- $\tilde{x} \rightarrow$ `dco::derivative(dco::value(x))`
- $\bar{\tilde{x}} \rightarrow$ `dco::derivative(dco::derivative(x))`

Adjoint over Adjoint: Driver

```
using B_M = dco::ga1s<double>;

B_M::global_tape = DCO_BASE_TAPE_T::create(o);
DCO_M::global_tape = DCO_TAPE_T::create(o);

for (size_t i = 0; i < n; ++i) {
  //need \tilde{x}
  B_M::global_tape->register_variable(dco::value(x[i]));
  //need to differentiate \bar{x} to compute the adjoints
  DCO_M::global_tape->register_variable(x[i]);
}

//record both tapes
foo(n, x, y);
```

Adjoint over Adjoint: Driver

```
for (size_t k = 0; k < n; k++) {
  dco::value(dco::derivative(y)) = 1.0;//\bar{y} = 1.0
  DCO_M::global_tape->interpret_adjoint();

  //\tilde{\bar{x}} = e_k
  dco::derivative(dco::derivative(x[k])) = 1.0;

  B_M::global_tape->interpret_adjoint();

  //Extract the derivatives
  for (int i = 0; i < n; i++)//\tilde{x}
    Hess[k][i] = dco::derivative(dco::value(x[i]));

  DCO_M::global_tape->zero_adjoints();
  B_M::global_tape->zero_adjoints();
}
```

Summary Second-Order Models ($m = 1$)

- Second-Order tangent model has
  - smallest memory requirements
  - $O(n^2)$ complexity for accumulating the Hessian
- All three adjoint models
  - are mathematically equivalent ($O(n)$ for the Hessian)
  - implementation differ
- No advantage using the adjoint model for the second time

Summary Second-Order Adjoint Models ($m = 1$)

- tangent over adjoint
  - the preferred second-order adjoint model
  - smallest tape size from all adjoint models
  - external functions implementation from the first order model can be reused
- adjoint over tangent
  - higher tape size compared to tangent over adjoint
- adjoint over adjoint
  - highest memory requirements for the tape
  - complicated drivers due to two tapes
  - no re-recording of the tapes needed for computing Hessian projections (can be faster compared second order models involving tangent computation in some cases)

Second-Order Models $(m > 1)$

For $m > 1$ the Hessian $F''(\boldsymbol{x})$ is a $m \times n \times n$ tensor.

- tangent over tangent $\tilde{\dot{\boldsymbol{y}}} = \underbrace{\dot{\boldsymbol{x}}^T}_{\in \mathbf{R}^n} \cdot F''(\boldsymbol{x}) \cdot \underbrace{\tilde{\boldsymbol{x}}}_{\in \mathbf{R}^n} + F'(\boldsymbol{x}) \cdot \tilde{\dot{\boldsymbol{x}}}$. For each $\dot{\boldsymbol{x}} = e_i$, $\tilde{\boldsymbol{x}} = e_j$ we compute i, j-th entry in all $n \times n$ sub-matrices of F'' tensor. Hence $O(n^2)$ complexity.
- tangent over adjoint $\dot{\bar{\boldsymbol{x}}} = \underbrace{\bar{\boldsymbol{y}}^T}_{\in \mathbf{R}^m} \cdot F''(\boldsymbol{x}) \cdot \underbrace{\dot{\boldsymbol{x}}}_{\in \mathbf{R}^n} + F'(\boldsymbol{x})^T \cdot \dot{\bar{\boldsymbol{y}}}$ For each $\dot{\boldsymbol{x}} = e_j$, $\bar{\boldsymbol{y}} = e_i$ we compute i, j-th entry in all $m \times n$ sub-matrices of F'' tensor. Hence $O(m \cdot n)$ complexity.

nag

Second-Order Models $(m > 1)$

For $m > 1$ the Hessian $F''(\boldsymbol{x})$ is a $m \times n \times n$ tensor.

- adjoint over tangent $\bar{\boldsymbol{x}} = F'(\boldsymbol{x})^T \cdot \bar{\boldsymbol{y}} + \underbrace{\dot{\bar{\boldsymbol{y}}}^T}_{\in \mathbf{R}^m} \cdot F''(\boldsymbol{x}) \cdot \underbrace{\dot{\boldsymbol{x}}}_{\in \mathbf{R}^n}$ For each $\dot{\boldsymbol{x}} = e_j$, $\dot{\bar{\boldsymbol{y}}} = e_i$ we compute i, j-th entry in all $m \times n$ sub-matrices of F'' tensor. Hence $O(m \cdot n)$ complexity.

- adjoint over adjoint $\tilde{\boldsymbol{x}} = F'(\boldsymbol{x})^T \cdot \tilde{\boldsymbol{y}} + \underbrace{\bar{\boldsymbol{y}}^T}_{\in \mathbf{R}^m} \cdot F''(\boldsymbol{x}) \cdot \underbrace{\tilde{\bar{\boldsymbol{x}}}}_{\in \mathbf{R}^n}$ For each $\tilde{\bar{\boldsymbol{x}}} = e_j$, $\bar{\boldsymbol{y}} = e_i$ we compute i, j-th entry in all $m \times n$ sub-matrices of F'' tensor. Hence $O(m \cdot n)$ complexity.

Reusing first order adjoint code for external functions

Summary

In this Masterclass we

- learned the four different second-order models
- learned how to use the second-order models to compute the Hessian
- discussed the complexity of computing the Hessian with different models
- learned how to reuse external function implementation from the first-order adjoint model in the second-order adjoint model

You will see a survey on your screen after exiting from this session.
We would appreciate your feedback.

We are now moving on the Q&A Session