Optimización con Python

Una introducción a CasADi

Guido Sanchez

Centro de Investigación en Señales, Sistemas e Inteligencia Computacional
Universidad Nacional del Litoral

Seminarios del sinc(i), 2015

Outline

¿Qué es CasADi?

- Es un framework open-source (LGPL) utilizado para la diferenciación algorítmica, la optimización numérica y control óptimo.
- Utiliza la sintaxis de los sistemas de álgebra (CAS) y permite construir expresiones simbólicas que pueden ser diferenciadas automáticamente de forma eficiente.
- Actualmente, es una herramienta general para optimización numérica basada en métodos de gradiente, con un fuerte enfoque hacia el control óptimo.
- Desarrollado por Joel Andersson y Joris Gillis en el Optimization in Engineering Center (OPTEC) de KU Leuven bajo supervisión de Moritz Diehl.

¿Qué *no* es CasADi?

- No es un sistema de diferenciación algorítmica convencional que permita calcular derivadas a partir de código existente sin demasiadas modificaciones. Se debe reimplementar utilizando la sintaxis de CasADi.
- No es un sistema de álgebra. A pesar de que CasADi manipula expresiones simbólicas, sus capacidades son limitadas a comparación de una herramienta CAS.
- No es un solver de problemas de control óptimo, sino que trata de proveer con los bloques que permitan al usuario construir su solver.

Componentes principales

- Un CAS minimalista (como el Symbolic Toolbox de Matlab).
- Los algoritmos de diferenciación soportan:
  - Modo hacia adelante (forward) y hacia atras (adjoint).
  - Simbólico y numérico.
  - Dos maneras de representar expresiones.
- Interfaces a Ipopt, Sundials, (KNITRO, OOQP, qpOASES, CPLEX, LAPACK, CSparse, ACADO Toolkit ...
- Front-ends para C++, Python, Octave y Matlab.
- Importa modelos de JModelica.org

Tipos de datos fundamentales

- SX – tipo simbólico escalar.
- SXMatrix y DMatrix – matrices sparse.
- FX y clases derivadas – funciones de CasADi.
- MX – tipo simbólico matricial.

Integración de ODE/DAE

- Suite de integradores open-source Sundials (ODE: CVodes / DAE: IDAS).
- Utilización: `integrator = casadi.Integrator(ode function)`.
- Casadi se encarga de armar las ecuaciones necesarias.

Optimización

$$\begin{aligned} \underset{x \in \mathbb{R}^N}{\text{minimizar}} \quad & f(x) \\ \text{sujeto a} \quad & x_{min} \leq x \leq x_{max} \\ & g_{min} \leq g(x) \leq g_{max} \end{aligned} \tag{1}$$

- Restricciones de igualdad ($x_{min} = x_{max}$) para algunos x.
- Se formula utilizando solvers NLP (por ejemplo, IPOPT).

Repaso I

Sólo restricciones de igualdad

$$\begin{aligned} \underset{x \in \mathbb{R}^N}{\text{minimizar}} \quad & f(x) \\ \text{sujeto a} \quad & g(x) = 0 \end{aligned} \tag{2}$$

Para una solución óptima x^* existen multiplicadores λ^* tal que:

$$\begin{aligned} \nabla_x \mathcal{L}(x^*, \lambda^*) &= 0 \\ g(x^*) &= 0 \end{aligned} \tag{3}$$

$\nabla_x \mathcal{L}(x, \lambda) = f(x) - \lambda^T g(x)$ es el Lagrangiano

Repaso II

Con restricciones de igualdad y desigualdad

$$\begin{aligned} \underset{x \in \mathbb{R}^N}{\text{minimizar}} \quad & f(x) \\ \text{sujeto a} \quad & g(x) = 0, h(x) \geq 0 \end{aligned} \tag{4}$$

Condiciones de Karush-Kuhn-Tucker (KKT): Para una solución óptima x^* existen multiplicadores λ^*, μ^* tal que:

$$\begin{aligned} \nabla_x \mathcal{L}(x^*, \lambda^*, \mu^*) &= 0 \\ g(x^*) &= 0 \\ h(x^*) &\geq 0 \\ \mu^* &\geq 0 \\ h(x^*)^T \mu^* &= 0 \end{aligned} \tag{5}$$

$\nabla_x \mathcal{L}(x, \lambda, \mu) = f(x) - \lambda^T g(x) - \mu^T h(x)$ es el Lagrangiano

IPOPT

- Optimizador open-source de punto interior: IPOPT[1].
- Resuelve NLP grandes (millones de variables/restricciones).
- Para resolver el sistema lineal se usa MA27, MA57, Mumps, Paradiso, ...
- Calcula $\frac{\partial g}{\partial x}$, $\nabla_x \mathcal{L}$ y $\nabla_x^2 \mathcal{L}$ utilizando algoritmos eficientes.
- Penaliza las desigualdades utilizando una funcion de barrera $\tau log(h(x))$,

$$\begin{aligned} \underset{x \in \mathbb{R}^N}{\text{minimizar}} \quad & f(x) - \tau log(h(x)) \\ \text{sujeto a} \quad & g(x) = 0 \end{aligned} \tag{6}$$

[1] www.coin-or.org/Ipopt

Notebook de iPython

- ¡Manos a la obra!

Conclusiones

- Permite formular problemas de optimización de forma simple.
- El framework simbólico permite plantear el algoritmo con formulaciones “de libro”.
- Los solver utilizados necesitan muy poco tiempo para resolver problemas de optimización no lineales.

¿Preguntas?

Más información

- http://www.casadi.org/
- https://github.com/casadi/casadi
- http://casadi.sourceforge.net/users_guide/html/casadi-users_guide.html
- Alternativa: CVXOPT - http://cvxopt.org/