Choisir 2 options parmi les 5 de l'UE 2 (FISE et FISA)

UE2 : 6 ECTS au total

Artificial intelligence (English)

Responsable: Alexandre Caminada, Jean Martinet

Résumé

Artificial intelligence is a field in the midst of emergence that is having an impact on the computer science through the design of programs, on the industrial, the health and the business sectors, and into our everyday life. Fundamentals are built on extracting relevant information from big data and on decision-making by autonomous systems. This course deals with algorithms and their applications for both problem families with case studies.

The second part of this course concerns Reinforcement Learning. We will study the principal approaches with and without deep learning: policy gradient, Q-learning, SARSA, and their issues. Practice will concern very simple games (atari based) allowing student to touch the difficulties and success using basic computing resources. Programming will be in python.

Objectifs

1. Be able to design and develop decision and problem solving for autonomous systems
2. Be able to integrate learning and solving algorithms in large complex systems including software and hardware components
3. Be able to understand and tune reinforcement learning solutions for autonomous systems and information extraction

Ce cours est commun Master 1 EIT Digital Data Science, Autonomous Systems, Financial technologies

Algorithmique avancée

Responsable: Johny Bond

Résumé:

Ce cours est composé de plusieurs parties indépendantes: Algorithmes de recherches de motifs et algorithmes du génome; Programmation dynamique, Algorithmes randomisés, Introduction à la calculabilité.

Prerequis:

- les acquis des cours:
  -
  - Algorithmique et Structures de données
  - Algorithmique et Complexité
  - Informatique Théorique

Objectifs

- Compléter les connaissances d'algorithmique.

Contenu

- Algorithmes de recherches de motifs, algorithmes du génome
- Programmation dynamique
- Algorithmes randomisés
- Calculabilité

References

- D. Harel : "Algorithmics : The spirit of computing", Addison Wesley
- Michael Mitzenmacher, Eli Upfal : Probability and computing - randomized algorithms and probabilistic analysis, Cambridge University Press, 2005
- Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest : Introduction to algorithms, MIT Press, 1990

Compétences

- CG1.1 Connaître et comprendre les concepts et les principes théoriques fondamentaux à la base de l'informatique. Niveau: Maîtrise

Acquis

- Connaître plusieurs paradigmes algorithmiques et leur conditions d'utilisation Niveau: Maîtrise

Préparation aux concours de programmation (SI3+SI4)

Responsable: Christophe Papazian

Résumé

Ce nouveau module est conçu pour les étudiants les plus motivés qui souhaitent s'engager dans la programmation compétitive. Ce programme d'entraînement sera destiné d'abord à la préparation aux compétitions de programmation et d'algorithmique qui ont lieu régulièrement au cours de l'année, mais nous aborderons aussi rapidement d'autres type de compétitions comme le Hashcode ou les CTF.
Les étudiants qui choisiront cette option dont **les places sont limitées**, s'engagent également à participer aux différentes compétitions de programmation pour le semestre en cours (DevBattle en mars & Hashcode en février) et si possible pour les compétitions à venir dans leurs études à Polytech.

Ce cours est un complémentaire au cours d'ASD de SI3, il est donc important pour les SI3 qui choisiront cette option d'être particulièrement attentif au contenu d'ASD et pour les SI4, de s'appuyer sur les connaissances acquises en SI3.

Ce module s'appuiera sur Python 3 : vous y apprendrez à écrire des codes courts de quelques lignes permettant de résoudre rapidement des problèmes posés.

Prerequis:

- les acquis des cours:
    - Analyse statistique des données

Programmation fonctionnelle

Responsable: Erick Gallesio

Résumé

Le module Programmation fonctionnelle dure 12 semaines. Chaque semaine comporte un cours et un TD associé:

1. Première partie – Introduction à la programmation fonctionnelle
    - Programmation fonctionnelle: Introduction
    - Listes
    - Fonctions
    - Fermetures, Types abstraits de données
    - Macros
2. Deuxième partie – Utilisations avancées
    - Implémentation d'un langage objet à base de prototypes
    - Flots & Programmation paresseuse
    - Mécanisme de continuations
    - Interprète Méta circulaire
3. Troisième partie – Introduction à la programmation logique
    - Introduction à la Programmation logique – Minikanren
    - Programmation Logique 1 (à déterminer)
    - Programmation Logique 2 (à déterminer)

Objectifs

L'objectif de ce cours est de monter quelques rudiments de programmation fonctionnelle, et par là donc, de monter que la programmation n'est pas obligatoirement impérative. Toutefois, ce cours n'est pas un cours sur Scheme (seule une toute partie du langage est montrée et utilisée), mais plutôt un cours qui essaye de montrer comment implémenter certains paradigmes de la programmation classique (structures, modules, objets, …) à l'aide d'un langage fonctionnel.

La dernière partie du module aborde aussi la programmation logique. L'idée est de montrer qu'il existe plusieurs façons de programmer la solution d'un problème et que celle-ci ne passe pas toujours par une variante d'un langage impératif.

Evaluation

- Moyenne des contrôles intermédiaires et/ou mini-TPs (coeff 2)
- Contrôle intermédiaire (coeff 2)
- Contrôle final (coeff 3)

Programmation parallèle

Cours donné in English if needed

Responsable: Fabrice Huet

Résumé

Savoir paralléliser un algorithme et programmer plusieurs processeurs pour réaliser un traitement.

Prerequis

- les acquis des cours:
    - Gestion de la concurrence
    - Programmation Orientée Objet
    - Programmation multi-paradigmes en C++
    - Programmation procédurale

Objectifs

- Paralléliser un algorithme
- Programmer avec plusieurs CPU
- Programmer avec plusieurs machines

Contenu

- Introduction générale à la programmation parallèle
- Architectures à mémoire partagée
- Architectures à mémoire distribuée
- Architectures GPGPU et programmation GPU
- Architectures MapReduce

References

- A Grama, A. Gupt, G. Karpys, V. Kumar, Introduction to Parallel Computing (2nd edition), Addison Wesley 2003.
- Arnaud Legrand, Yves Robert. Algorithmique Parallèle, Cours et exercices corrigés, Dunod, 2003.
- M. Gengler, S. Ubeda, F. Deprez, Initiation au parallelisme - Concepts, architectures et algorithmes, Masson 1996.
- P.S. Pacheco. Parallel programming with MPI. Morgan Kaufmann. 1997.
- R. Chandra, R. Menon, L. Dagum, D. Kohr, D. Maydan, J. Mc_Donald. Parallel Programming in openMP. Morgan Kaufmann Publishers, 2000.

Compétences

- CG2.4 Maîtriser les architectures des systèmes informatiques permettant de déployer des solutions sur des plateformes hétérogènes et réparties : serveurs et postes clients, réseaux et Internet, réseaux mobiles. Niveau: Applications
- CG3.5 Maîtriser des méthodes de gestion des projets informatiques de grande échelle incluant les normes de qualité, permettant de concevoir leur architecture et leur intégration / évolution dans des systèmes préexistants : urbanisation des systèmes d'informatiques. Niveau: Applications

Acquis

- Programmation GPU Niveau: Maîtrise
- Programmation OpenMP Niveau: Maîtrise
- Effectuer un traitement avec MapReduce Niveau: Applications

Evaluation

- 1 micro-projet (20%) - 2 examens partiels (40%) - 1 examen final (40%)

Ce cours est commun avec MAM4 et Master 1 EIT Digital Data Science et Financial technologies

Choisir 4 options (3 pour les FISA) parmi les 8 de l'UE 1

UE1 : 12 ECTS (9 pour FISA) au total

Capteurs/actionneurs et intelligence artificielle embarquée / Sensors/actuators and embedded artificial intelligence

Cours donné in English if needed

Responsable: Benoît Miramond

Résumé

Les capteurs et actionneurs sont respectivement les points d'entrée et de sortie pour l'interaction entre le monde numérique et le monde réel. Cette interaction est particulièrement présente dans le contexte des systèmes embarqués comme les systèmes de contrôle commande, de l'IoT, ou des réseaux de capteurs sans fils. Le maintien d'un état stable et cohérent du système numérique face à l'évolution des processus physiques qui l'entourent est un enjeu complexe qui fait intervenir des compétences à la fois logicielles et matérielles. On s'intéressera particulièrement aux notions de systèmes temps réel et de programmation embarquée sur des systèmes autonomes contraints à la fois en énergie, en mémoire et en puissance de calcul. On étudiera plus spécifiquement comment ces notions se déclinent dans le contexte récent de l'intelligence artificielle embarquée avec le déploiement de réseaux de neurones artificiels sur systèmes à micro-contrôleurs.

Sensors and actuators are respectively the entry and exit points for the interaction between the digital and real world. This interaction is particularly present in the context of embedded systems such as control command systems, IoT, or wireless sensor networks. Maintaining a stable and coherent state of the digital system face to the constant evolution of the physical processes surrounding it is a complex issue involving both software and hardware skills. We will be particularly interested in the notions of real-time systems and embedded programming on autonomous systems constrained at the same time in energy, memory and computing power. More specifically, we will study how these notions are applied in the recent context of embedded artificial intelligence with the deployment of artificial neural networks on micro-controller systems.

Prerequis:

- programmation C, systèmes d'exploitation, architecture des ordinateur
- notions en machine learning

Objectifs

Permettre aux étudiants de comprendre les problèmes inhérents au traitement de l'information en provenance et à destination des capteurs/actionneurs.
Aborder les contraintes spécifiques de la programmation embarquée sur micro-contrôleurs.
Comprendre les enjeux du déploiement d'algorithmes d'intelligence artificielle sur des systèmes autonomes et dans des environnements contraints.
Apprentissage de données de capteurs pour applications embarquées.
Mettre en pratique les notions fondamentales sur des cibles MCU embarquées.

Contenu

- Système temps réel : Définition et Modélisation
- Notions de capteurs, actionneurs, observation, commande
- Architectures matérielles/logicielles pour des systèmes embarqués interagissant avec l'environnement physique

- Microcontrôleurs
- Etat de l'art sur les capteurs et les actionneurs
- Etat de l'art sur les cibles matérielles/logicielles d'acquisition de mesures et restitution de commandes (notions de microcontrôleurs, CAN, CNA, E/S digitales, consommation et autonomie)
- Outils de développement et cross compilation
- Projets et applications : chaque étudiant sera amené à travailler sur une application concrète sur cible embarquée pour illustrer les notions vues dans le cours. Les cibles étudiées seront des systèmes à base de microcontrôleurs telles que des réseaux de capteurs sans fil pour le traitement des mesures capteurs et l'observation d'état du système.

Références

- Embedded systems architecture, Noergaard, 2005
- uC/OS-II, the real-time kernel, J. Labrosse, Micrium, 2002
- Ordonnancement dans les systèmes temps réel, M. Chetto, ISTE edition, 2014

Compétences

- CG2.4 Maîtriser les architectures des systèmes informatiques permettant de déployer des solutions sur des plateformes hétérogènes et réparties : serveurs et postes clients, réseaux et Internet, réseaux mobiles. Niveau: Maîtrise

Acquis

- programmation embarquée sur micro-contrôleurs
- programmation par interruptions
- systèmes d'exploitation temps réel
- ordonnancement temps réel
- technologies des capteurs et actionneurs/capteurs sans fils
- déploiement de réseaux de neurones artificiels sur MCU
- intelligence artificielle embarquée

Ce cours est commun avec ELEC4 et Master 1 EIT Digital Autonomous systems

DevOps & Continuous Testing

A combiner obligatoirement avec ISA. (cours délivré en français)

Responsable: Guilhem Molines

Résumé:

Newly graduated students are generally good at writing code as well as unit-testing it. They know different programming languages, understand the concepts of software architecture, havebeen exposed to environments such as Cloud or Web Services. Yet what they often lack is a comprehensive view at the testing phase, including test that require hard-to-setup environments, as well as the ability define a build-and-test pipeline, to automate deployments, to operate deployed systems. This option will teach students all the concepts that are needed to produce a robust Software Factory, that supports the notion of Continuous Delivery, delivering robust code backed by comprehensive testing techniques. This is a very hands-on class, as students will be guided to implement the tests and factory, which will be exercised on thee complex project included in the Application Servers class.

Introduction aux Systèmes et Logiciels Embarqués

Responsable: Stéphane Lavirotte

Résumé

Quel point commun y-a-t'il entre une smart watch, un aspirateur Roomba, un drone volant, un robot bipède?... Tous sont des systèmes embarqués, même s'ils reposent sur des technologies différentes. Ce cours introduit les différents aspects logiciels liés à ces systèmes embarqués omniprésents dans notre vie quotidienne. Après une présentation rapide des différents éléments d'un système embarqué (noyau et système d'exploitaion ou simple programme sur micro-contrôleur), nous étudierons les spécificités logicielles de ces systèmes: systèmes embarqués avec et sans système d'exploitation, déploiement d'un système sur une cible particulière (cross-compilation), optimisation d'un système et des applications pour tenir compte des contraintes des plateformes embarquées (consommation énergétique, emprunte mémoire, propriétés temps réel...). Nous illustrerons ces différentes problématiques à l'aide de plateformes embarquées (Arduino, Raspberry Pi, ...) et de capteurs et actionneurs.

Prerequis

- Programmation C. Programmation système (Posix).

Objectifs

- Introduire les différents types de plateformes embarquées.
- Présenter les spécificités du développement logiciel pour les systèmes embarqués.
- Concevoir et réaliser un système embarqué à base de composants logiciels.
- Choisir une architecture matérielle et logicielle pour un système embarqué.

Contenu

- Introduction aux catégories de systèmes embarqués et aux systèmes d'exploitation
- Démarrage d'un système embarqué et noyau Linux
- Cross-compilation d'un système d'exploitation pour système embarqué
- Applications pour systèmes embarqués
- Déployer son propre système embarqué GNU/Linux (Raspberry Pi)
- Optimiser les applications d'un système embarqué
- Microcontrôleurs sans OS (Arduino)
- Microcontrôleurs avec OS: RTOS
- Systèmes mixtes, micro-contrôleur et microprocesseur

References

- Pierre Ficheux, Eric Bénard, Linux embarqué. Nouvelle étude de cas - Traite d'OpenEmbedded, Eyrolles, 2012
- Chistophe Blaess, Solution temps réel sous Linux, 2ème édition, Eyrolles, nov 2015
- Donald Norris, Projets créatifs avec Raspberry Pi, fév 2014

Compétences

- CG1.2 Maîtriser les liens entre les disciplines et transposer les mêmes concepts d'un domaine à un autre, être capable de collaborer avec des spécialistes de disciplines connexes Niveau: Applications
- CG2.4 Maîtriser les architectures des systèmes informatiques permettant de déployer des solutions sur des plateformes hétérogènes et réparties : serveurs et postes clients, réseaux et Internet, réseaux mobiles. Niveau: Maîtrise
- CG3.1 Concevoir des modèles, systèmes et process en utilisant des méthodologies d'analyse, de conception et de modélisation, en connaissant leurs limites et sans perdre le sens de la réalité et du concret. Niveau: Maîtrise

Acquis

- Compréhension du fonctionnement d'un système d'exploitation Niveau: Maîtrise
- Développement en environnements hétérogènes: compilation croisée Niveau: Maîtrise
- Optimisation système et logicielle pour les plateformes embarquées Niveau: Maîtrise

- Micro-contrôleur avec et sans système d'exploitation Niveau: Maîtrise
- Systèmes Temps-Réels Niveau: Notions

Introduction à l'architecture logicielle (ISA : « Introduction to Software Architecture »)

A combiner obligatoirement avec DevOps

Responsable: Philippe Collet - Anne-Marie Dery

Résumé

Ce cours est une introduction à l'architecture logicielle qui illustre la mise en oeuvre d'une architecture n-tiers impliquant plusieurs langages. Il introduit les notions de composants logiciels, d'injection de dépendances, d'interopérabilité et de travail aux interfaces.

Prerequis:

- les acquis des cours:
  - Programmation Orientée Objet

Objectifs

- Comprendre les principes des serveurs d'applications
- Spécifier et mettre en oeuvre des composants logiciels
- Concevoir et réaliser un système à base de composants hétérogènes
- Concevoir des applications d'entreprise en respectant les bonnes pratiques

Contenu

- Introduction à l'Architecture Logicielle
- Correspondance Objet - Relationel
- Couche de persistance
- Couche domaine
- Couche présentation
- Communications synchrones et asynchrones
- Applications hétérogènes: introduction à un nouveau langage
- Interoperabilité
- Client lourd
- Sécurité et transactions
- Retrospective

References

- Beautiful Architecture, Diomidis Spinellis & Georgios Gousios
- Martin Fowler, Patterns of enterprise application architecture
- Software architecture in practice, Bass, Clemens and Kazman
- Software System Architecture, Rozansky and woods

Compétences

- CG1.2 Maîtriser les liens entre les disciplines et transposer les mêmes concepts d'un domaine à un autre, être capable de collaborer avec des spécialistes de disciplines connexes Niveau: Maîtrise
- CG2.3 Maîtriser les différents aspects des systèmes d'information (fonctionnels, organisationnels, techniques), de leur conception à leur mise en œuvre et leur intégration tant d'un point de vue conceptuel qu'appliqué. Niveau: Maîtrise
- CG2.5 Maîtriser la sécurité des logiciels, systèmes, réseaux et des données. Niveau: Applications

Acquis

- Savoir définir l'interface d'un composant logiciel Niveau: Maîtrise
- Définir une architecture logicielle Niveau: Notions
- Mettre en oeuvre un mapping objet-relationel Niveau: Notions
- Mettre en oeuvre une application répartie hétérogène Niveau: Applications

Réalité augmentée

Responsable: Rodrigo Cabral Farias

Résumé

Ce cours a pour but d'introduire divers concepts et outils (vision 3D, calibration d'une caméra, rendu 3D) nécessaires pour la mise en œuvre de la réalité augmentée. Une introduction à la problématique de visualisation des données fait aussi partie de ce cours.

Prerequis:

Notions de base en algèbre linéaire et en optimisation.

Objectifs

- Comprendre les fondements de la vision 3D
- Comprendre les éléments de base de la chaîne de traitement du rendu 3D.
- Introduire les problématiques liées à la visualisation des données.

Contenu

- Introduction à la réalité augmentée
- Coordonnées homogènes et transformations 2D/3D
- Formation de l'image
- Calibration d'une caméra
- Stereopsis
- Pipeline du rendu 3D
- Rendus d'illumination et de texture 3D
- Historique de la visualisation de données
- Grammaire des graphiques
- Techniques de réduction de dimension pour la visualisation des données.

Compétences

- CG1.2 Maîtriser les liens entre les disciplines et transposer les mêmes concepts d'un domaine à un autre, être capable de collaborer avec des spécialistes de disciplines connexes Niveau: Applications
- CG2.2 Maîtriser les mathématiques permettant la manipulation des données informatisées sous toutes leurs formes. Niveau: Applications

Acquis

- Notions de base en vision par ordinateur Niveau: Maîtrise
- Reconnaissance de gestes Niveau: Applications

Ce cours est commun avec MAM4

Réseaux avancés et middleware

Cours donné in English if needed

Responsable : Dino Lopez Pacheco - Françoise Baude

Résumé

Distributed applications require, for their interactions,,solutions specially designed to be executed in both heterogeneous environments and systems. Such solutions, known as the *Middleware Layer* rely on a set of network protocols that must be adequately leveraged and fine tuned, to provide the required Quality of Service and performances required by the application.

This lecture consists of two parts. On the one hand, we study the mechanisms and algorithms employed at the Transport, Network and Link Layers, to identify how the network protocol stack can impact the performance of applications and even, potentially challenge the good interaction between different peers. Hence, we explore the main TCP protocol flavours (namely Reno, New Reno and CUBIC), the adjacent network overlay solutions and impact of the infrastructure on the link access protocols. Also, we provide a quick overview of NATs and firewalls.

On the other hand, our aim is to understand how a middelware, laying on top of network protocols, offers an appropriate runtime support for distributed applications. Kind of properties that are considerd are : securing communications, supporting concurrent execution of activities on one node, robust synchronous or asynchronous transmission of application-level information, interoperability and accessibility of various application parts possibly described in different languages. The typical middleware studied are Java RMI, the JMS specification and ActiveMQ implementation.

Prerequis

- IP addressing,
- Basics on Linux-based Operating Systems
- Basics on Network Routing
- Socket Programming in C or Python
- Basics on Network Topologies and Architectures (core, access and data center networks).
- Java language

Objectifs

- Understand the different protocol services at the Transport Layer (reliability, flow and congestion control, fairness).
- Understand how the physical network infrastructure drives the creation of virtual overlay networks, and the way Link Layer protocols handle network access, contention and collisions.
- Understand how the combination of the network protocol stack leads to an overhead able to hurt the applications performances.
- Understand the challenge introduced by NATs and Firewall to the interconnection of distributed applications.
- Capability to design a distributed applications, choosing the right interaction paradigm and master the mechanisms offered by the chosen middleware both at programming and at runtime levels

Contenu

- UDP vs TCP
- Lab: Exploring the flow control, congestion control and TCP performance.
- NATs and Firewall
- Lab: NATs and Firewall deployment with iptables and traversal technics.
- Introduction to overlay networks
- Lab: deployment of overlay networks and overhead analysis.
- Synchronous interaction paradigm: Remote Method invocation between Objects : Java RMI
- Lab: advanced study of the Java RMI platform
- Asynchronous interaction paradigm: Message oriented middleware
- Lab: Java Messaging system (JMS) on ActiveMQ

Acquis

- Knowledge on TCP services (reliability, congestion control, flow control, fairness) and TCP Reno, New Reno and CUBIC flavors
- Knowledge on overlay network technics (e.g. VxLAN, GRE, ...)
- Knowledge on 802.3 and 802.11 protocols.
- Design, implementation and deployment of a distributed application relying on either synchronous (RMI) or asynchronous (JMS) interaction models

Evaluation

- 1 written exam or 1 graded lab for the first part (35%)
- 1 homework for the security in networking sub part (25%)
- 1 written exam and 1 extended lab or mini project for the second part (40% in total)

Ce cours est commun Master 1 EIT Digital Financial technologies et Data Science

Service Oriented Computing /WS

Responsable: Jean-Yves Tigli

Résumé

Ce cours introduit le concept de service logiciel et les deux grandes approches pour le développement d'application orientée service (SOA, ROA). Ce cours s'appuie sur une double démarche pédagogique : - placer dans l'évolution historique du Web l'apparition des services Web - décomposer le concept abstrait de service selon un modèle unifié et présenter ses déclinaisons technologiques. Dans ce cadre les deux technologies le plus populaires du domaine sont détaillées et expliquées : - les Web Services RESTFul - les Web Services WS-SOAP du W3C Le modèle unifié présenté est ensuite illustré par l'environnement de développement de services "Windows Communication Foundation" du framework .Net. Chacune des notions, donne lieu à des Travaux Pratiques pour en assurer la maîtrise et mettre en avant l'avantage prépondérant d'une application distribuée développée avec des services : l'interopérabilité.

Prerequis:

- les acquis des cours:
  - Programmation Orientée Objet
  - Réseaux: configuration et programmation

Objectifs

- Introduction du concept de service logiciel et des deux grandes approches pour le développement d'application orientée service (SOA, ROA).
- Comprendre dans l'évolution du Web l'apparition des services Web
- Introduire un modèle unifié de service logiciel et présenter ses déclinaisons technologiques les plus populaires : - les Web Services RESTFul - les Web Services WS-SOAP du W3C
- Mis en oeuvre de l'environnement de développement de services "Windows Communication Foundation" du framework .Net basé sur le modèle unifié introduit au travers des exercices pratiques.
- Mettre en oeuvre une application multi-services sur des plateformes hétérogènes pour mettre à profit l'interopérabilité des services.

Contenu

- Les principes des architecture SOA (orientée service) et des ROA (orientée ressource)
- Illustration avec les approches les plus populaires : - les services RESTFul pour les ROA - les services WS-SOAP pour les SOA
- Modèle unifié de service logicielle. Mise en oeuvre du Modèle de programmation de services de Windows Communication Foundation .Net. Gestions des contrats, liaisons, adresses et canaux dans le développement de services.
- Introduction au Web H2M (Human to Machine): - son protocole HTTP - la gestion de pages Web statiques puis dynamiques Application aux cgi-bin, formulaires Web, et JSP.
- Passage d'une approche H2M du Web au M2M (Machine to Machine) : le browser web n'est plus le seul client web

References

- Supports de cours en ligne
- Pro WCF 4: Practical Microsoft SOA Implementation, par Nishith Pathak
- RESTful Web Services Par Leonard Richardson,Sam Ruby, O'Reilly
- Web Services Essentials: Distributed Applications with XML-RPC, SOAP, UDDI ... Par Ethan Cerami, 2002, O'Reilly

Compétences

- CG2.1 Maîtriser les différents aspects du développement logiciel, qu'ils soient techniques, fonctionnels, organisationnels ou humains. Niveau: Expert
- CG2.4 Maîtriser les architectures des systèmes informatiques permettant de déployer des solutions sur des plateformes hétérogènes et réparties : serveurs et postes clients, réseaux et Internet, réseaux mobiles. Niveau: Expert
- CG3.1 Concevoir des modèles, systèmes et process en utilisant des méthodologies d'analyse, de conception et de modélisation, en connaissant leurs limites et sans perdre le sens de la réalité et du concret. Niveau: Applications

Acquis

- Architecture Logicielle orientée service et orientée ressource Niveau: Applications
- Modèle de programmation de services Windows Communication Foundation .Net Niveau: Expert
- Compréhension de l'évolution du Web des browsers aux services web Niveau: Maîtrise
- Développement Web Services WS-SOAP / W3C Niveau: Expert
- Modèles de Services Logicielles Niveau: Maîtrise
- Développement Web Services REST/RESTFul Niveau: Expert

Evaluation :

Contrôle Continu : QCMs et/ou TP à rendre et évaluation finale

Valorisation des données (English)

Responsable: Lionel Fillatre

Résumé

Ce cours est une introduction à la science des données. Il permet de découvrir les outils de base pour analyser et valoriser des données. La valorisation des données comprend plusieurs étapes : l'acquisition, l'affichage et le traitement des données. Ce cours propose un rappel progressif des outils statistiques nécessaires à la manipulation des données et à la fiabilité des traitements. Enfin, ce cours permet de se familiariser avec le langage de programmation R.

Prerequis:

- Notions de base en statistiques et probabilités.
- les acquis des cours:
  -
    - Traitement d'analyse statistiques de données
    - Machine Learning

Objectifs

- Échantillonner des données
- Analyser des données
- Visualiser des résultats d'une analyse
- Interpréter des données

Contenu

- Introduction à la science des données
- Acquisition et nettoyage de données
- Échantillonnage des données
- Visualisation des données
- Estimation ponctuelle
- Test d'hypothèses
- Test de Bayes naïf
- Régression logistique
- Analyse des correspondances
- Filtrage collaboratif
- Apprentissage par renforcement

References

- G.Govaert, "Analyse des données", Hermes, 2003.
- M. Jambu, "Exploration informatique et statistique des données", Dunod,1989.
- Matthias Kohl, "Introduction to statistical data analysis with R", London, 2015.
- Andy Field, Jeremy Miles, Zoë Field, "Discovering statistics using R", Sage, 2012.
- Trevor Hastie, Robert Tibshirani, Jerome Friedman, "The Elements of Statistical Learning: Data Mining, Inference, and Prediction", Second Edition, Springer Science & Business Media, 2009.

Compétences

- CG1.3 Maîtriser des méthodes et outils mathématiques utilisés en informatique permettant de concevoir et valider des solutions techniques et de contourner des limitations intrinsèques:théorie de l'information, statistiques, cryptographie, modèles aléatoires, calculabilité. Niveau: Applications
- CG2.2 Maîtriser les mathématiques permettant la manipulation des données informatisées sous toutes leurs formes. Niveau: Maîtrise
- CG3.1 Concevoir des modèles, systèmes et process en utilisant des méthodologies d'analyse, de conception et de modélisation, en connaissant leurs limites et sans perdre le sens de la réalité et du concret. Niveau: Notions

Acquis

- Analyser des données sous différents formats avec des outils appropriés Niveau: Applications
- Afficher les résultats d'une analyse de données de façon adaptée à un objectif métier et à l'utilisateur Niveau: Applications
- Evaluer la fiabilité d'une analyse de données

Ce cours est commun avec MAM4 et Master 1 EIT Digital Data Science, Financial technologies