Introduction to CS & AI

Tao LIN

September 3, 2024

WESTLAKE UNIVERSITY | SCHOOL OF ENGINEERING

Table of Contents

Indeed “introduction to CS & AI” DOES NOT indicate that

we will go through all basic CS & AI materials step-by-step in this course!

If you are still unfamiliar with

[1] https://inst.eecs.berkeley.edu/~cs61a/sp22/

[2] https://sp23.datastructur.es/

[3] https://cs61c.org/fa23/

If you are still unfamiliar with

- programming, e.g., Python

[1]https://inst.eecs.berkeley.edu/~cs61a/sp22/
[2]https://sp23.datastructur.es/
[3]https://cs61c.org/fa23/

If you are still unfamiliar with

- programming, e.g., Python
- data structure, e.g., heap, minimum spanning trees

[1] https://inst.eecs.berkeley.edu/~cs61a/sp22/
[2] https://sp23.datastructur.es/
[3] https://cs61c.org/fa23/

If you are still unfamiliar with

- programming, e.g., Python
- data structure, e.g., heap, minimum spanning trees
- algorithms, like DFS, BFS, quick sort

[1] https://inst.eecs.berkeley.edu/~cs61a/sp22/
[2] https://sp23.datastructur.es/
[3] https://cs61c.org/fa23/

If you are still unfamiliar with

- programming, e.g., Python
- data structure, e.g., heap, minimum spanning trees
- algorithms, like DFS, BFS, quick sort
- principles in software engineering

[1] https://inst.eecs.berkeley.edu/~cs61a/sp22/
[2] https://sp23.datastructur.es/
[3] https://cs61c.org/fa23/

If you are still unfamiliar with

- programming, e.g., Python
- data structure, e.g., heap, minimum spanning trees
- algorithms, like DFS, BFS, quick sort
- principles in software engineering

You are encouraged to check the following awesome courses, e.g.,

[1] https://inst.eecs.berkeley.edu/~cs61a/sp22/
[2] https://sp23.datastructur.es/
[3] https://cs61c.org/fa23/

If you are still unfamiliar with

- programming, e.g., Python
- data structure, e.g., heap, minimum spanning trees
- algorithms, like DFS, BFS, quick sort
- principles in software engineering

You are encouraged to check the following awesome courses, e.g.,

- UC Berkeley, CS 61A: Structure and Interpretation of Computer Programs[1].

[1] https://inst.eecs.berkeley.edu/~cs61a/sp22/
[2] https://sp23.datastructur.es/
[3] https://cs61c.org/fa23/

If you are still unfamiliar with

- programming, e.g., Python
- data structure, e.g., heap, minimum spanning trees
- algorithms, like DFS, BFS, quick sort
- principles in software engineering

You are encouraged to check the following awesome courses, e.g.,

- UC Berkeley, CS 61A: Structure and Interpretation of Computer Programs[1].
- UC Berkeley, CS 61B: Data Structures[2].

[1] https://inst.eecs.berkeley.edu/~cs61a/sp22/
[2] https://sp23.datastructur.es/
[3] https://cs61c.org/fa23/

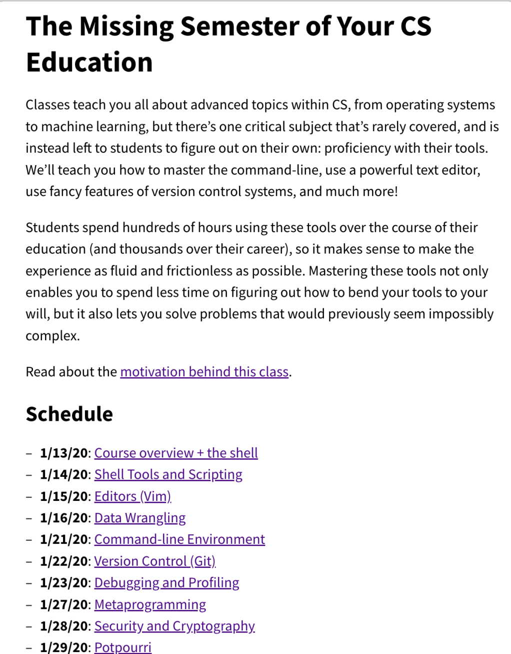
If you are still unfamiliar with

- programming, e.g., Python
- data structure, e.g., heap, minimum spanning trees
- algorithms, like DFS, BFS, quick sort
- principles in software engineering

You are encouraged to check the following awesome courses, e.g.,

- UC Berkeley, CS 61A: Structure and Interpretation of Computer Programs[1].
- UC Berkeley, CS 61B: Data Structures[2].
- (Even) UC Berkeley, CS 61C: Great Ideas in Computer Architecture (Machine Structures)[3].

[1] https://inst.eecs.berkeley.edu/~cs61a/sp22/
[2] https://sp23.datastructur.es/
[3] https://cs61c.org/fa23/



The Missing Semester of Your CS Education

Classes teach you all about advanced topics within CS, from operating systems to machine learning, but there's one critical subject that's rarely covered, and is instead left to students to figure out on their own: proficiency with their tools. We'll teach you how to master the command-line, use a powerful text editor, use fancy features of version control systems, and much more!

Students spend hundreds of hours using these tools over the course of their education (and thousands over their career), so it makes sense to make the experience as fluid and frictionless as possible. Mastering these tools not only enables you to spend less time on figuring out how to bend your tools to your will, but it also lets you solve problems that would previously seem impossibly complex.

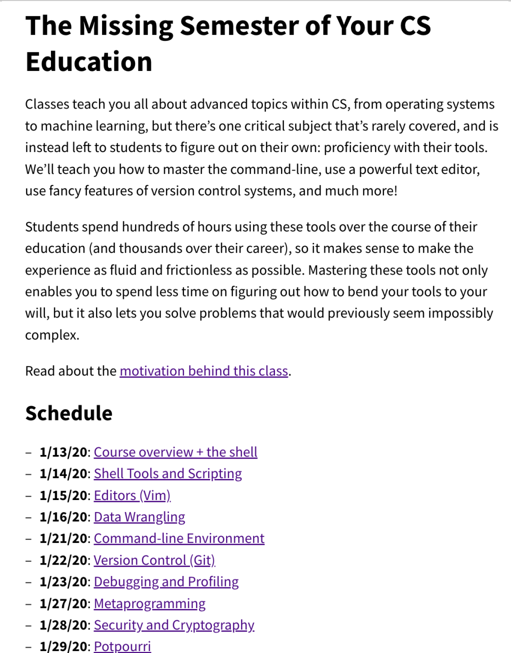
Read about the motivation behind this class.

Schedule

- **1/13/20**: Course overview + the shell
- **1/14/20**: Shell Tools and Scripting
- **1/15/20**: Editors (Vim)
- **1/16/20**: Data Wrangling
- **1/21/20**: Command-line Environment
- **1/22/20**: Version Control (Git)
- **1/23/20**: Debugging and Profiling
- **1/27/20**: Metaprogramming
- **1/28/20**: Security and Cryptography
- **1/29/20**: Potpourri

Build your research weapons and wheels

Check it out: `https://missing.csail.mit.edu/`



The Missing Semester of Your CS Education

Classes teach you all about advanced topics within CS, from operating systems to machine learning, but there's one critical subject that's rarely covered, and is instead left to students to figure out on their own: proficiency with their tools. We'll teach you how to master the command-line, use a powerful text editor, use fancy features of version control systems, and much more!

Students spend hundreds of hours using these tools over the course of their education (and thousands over their career), so it makes sense to make the experience as fluid and frictionless as possible. Mastering these tools not only enables you to spend less time on figuring out how to bend your tools to your will, but it also lets you solve problems that would previously seem impossibly complex.

Read about the motivation behind this class.

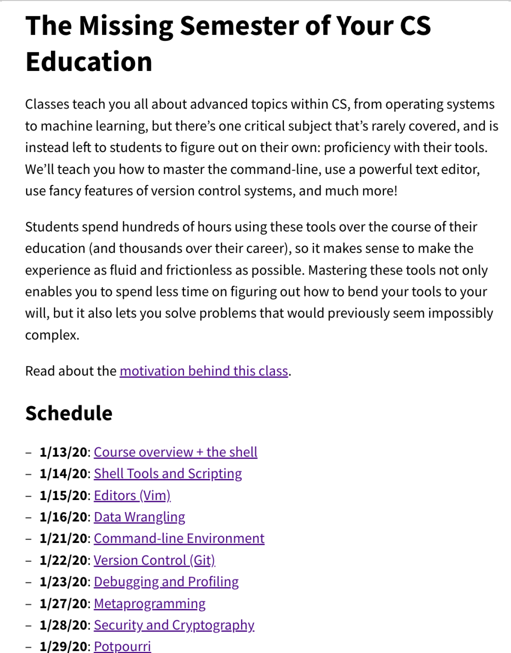
Schedule

- **1/13/20**: Course overview + the shell
- **1/14/20**: Shell Tools and Scripting
- **1/15/20**: Editors (Vim)
- **1/16/20**: Data Wrangling
- **1/21/20**: Command-line Environment
- **1/22/20**: Version Control (Git)
- **1/23/20**: Debugging and Profiling
- **1/27/20**: Metaprogramming
- **1/28/20**: Security and Cryptography
- **1/29/20**: Potpourri

Build your research weapons and wheels

Check it out: https://missing.csail.mit.edu/

E.g.,



The Missing Semester of Your CS Education

Classes teach you all about advanced topics within CS, from operating systems to machine learning, but there's one critical subject that's rarely covered, and is instead left to students to figure out on their own: proficiency with their tools. We'll teach you how to master the command-line, use a powerful text editor, use fancy features of version control systems, and much more!

Students spend hundreds of hours using these tools over the course of their education (and thousands over their career), so it makes sense to make the experience as fluid and frictionless as possible. Mastering these tools not only enables you to spend less time on figuring out how to bend your tools to your will, but it also lets you solve problems that would previously seem impossibly complex.

Read about the motivation behind this class.

Schedule

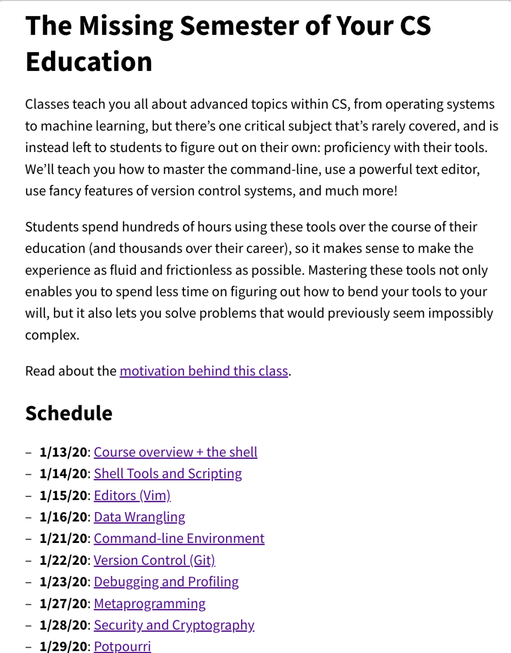
- **1/13/20**: Course overview + the shell
- **1/14/20**: Shell Tools and Scripting
- **1/15/20**: Editors (Vim)
- **1/16/20**: Data Wrangling
- **1/21/20**: Command-line Environment
- **1/22/20**: Version Control (Git)
- **1/23/20**: Debugging and Profiling
- **1/27/20**: Metaprogramming
- **1/28/20**: Security and Cryptography
- **1/29/20**: Potpourri

Build your research weapons and wheels

Check it out: `https://missing.csail.mit.edu/`

E.g.,

- Command shell



The Missing Semester of Your CS Education

Classes teach you all about advanced topics within CS, from operating systems to machine learning, but there's one critical subject that's rarely covered, and is instead left to students to figure out on their own: proficiency with their tools. We'll teach you how to master the command-line, use a powerful text editor, use fancy features of version control systems, and much more!

Students spend hundreds of hours using these tools over the course of their education (and thousands over their career), so it makes sense to make the experience as fluid and frictionless as possible. Mastering these tools not only enables you to spend less time on figuring out how to bend your tools to your will, but it also lets you solve problems that would previously seem impossibly complex.

Read about the motivation behind this class.

Schedule

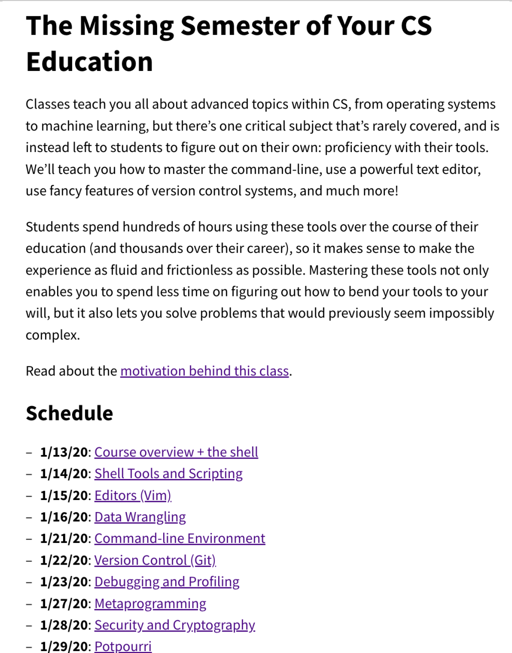
- **1/13/20**: Course overview + the shell
- **1/14/20**: Shell Tools and Scripting
- **1/15/20**: Editors (Vim)
- **1/16/20**: Data Wrangling
- **1/21/20**: Command-line Environment
- **1/22/20**: Version Control (Git)
- **1/23/20**: Debugging and Profiling
- **1/27/20**: Metaprogramming
- **1/28/20**: Security and Cryptography
- **1/29/20**: Potpourri

Build your research weapons and wheels

Check it out: `https://missing.csail.mit.edu/`

E.g.,

- Command shell
- Version control



The Missing Semester of Your CS Education

Classes teach you all about advanced topics within CS, from operating systems to machine learning, but there's one critical subject that's rarely covered, and is instead left to students to figure out on their own: proficiency with their tools. We'll teach you how to master the command-line, use a powerful text editor, use fancy features of version control systems, and much more!

Students spend hundreds of hours using these tools over the course of their education (and thousands over their career), so it makes sense to make the experience as fluid and frictionless as possible. Mastering these tools not only enables you to spend less time on figuring out how to bend your tools to your will, but it also lets you solve problems that would previously seem impossibly complex.

Read about the motivation behind this class.

Schedule

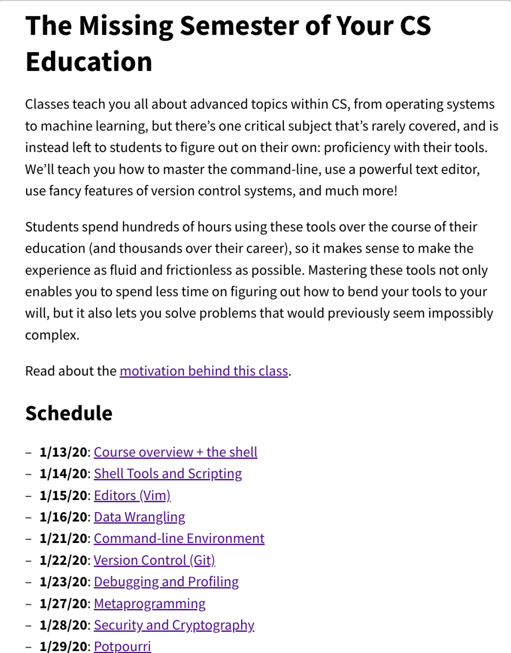
- **1/13/20**: Course overview + the shell
- **1/14/20**: Shell Tools and Scripting
- **1/15/20**: Editors (Vim)
- **1/16/20**: Data Wrangling
- **1/21/20**: Command-line Environment
- **1/22/20**: Version Control (Git)
- **1/23/20**: Debugging and Profiling
- **1/27/20**: Metaprogramming
- **1/28/20**: Security and Cryptography
- **1/29/20**: Potpourri

Build your research weapons and wheels

Check it out: `https://missing.csail.mit.edu/`

E.g.,

- Command shell
- Version control
- Text editing



The Missing Semester of Your CS Education

Classes teach you all about advanced topics within CS, from operating systems to machine learning, but there's one critical subject that's rarely covered, and is instead left to students to figure out on their own: proficiency with their tools. We'll teach you how to master the command-line, use a powerful text editor, use fancy features of version control systems, and much more!

Students spend hundreds of hours using these tools over the course of their education (and thousands over their career), so it makes sense to make the experience as fluid and frictionless as possible. Mastering these tools not only enables you to spend less time on figuring out how to bend your tools to your will, but it also lets you solve problems that would previously seem impossibly complex.

Read about the motivation behind this class.

Schedule

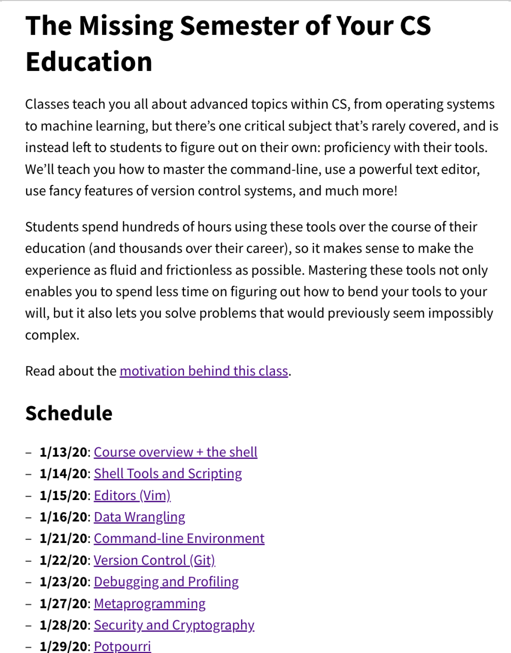
- **1/13/20**: Course overview + the shell
- **1/14/20**: Shell Tools and Scripting
- **1/15/20**: Editors (Vim)
- **1/16/20**: Data Wrangling
- **1/21/20**: Command-line Environment
- **1/22/20**: Version Control (Git)
- **1/23/20**: Debugging and Profiling
- **1/27/20**: Metaprogramming
- **1/28/20**: Security and Cryptography
- **1/29/20**: Potpourri

Build your research weapons and wheels

Check it out: `https://missing.csail.mit.edu/`

E.g.,

- Command shell
- Version control
- Text editing
- Remote machines



The Missing Semester of Your CS Education

Classes teach you all about advanced topics within CS, from operating systems to machine learning, but there's one critical subject that's rarely covered, and is instead left to students to figure out on their own: proficiency with their tools. We'll teach you how to master the command-line, use a powerful text editor, use fancy features of version control systems, and much more!

Students spend hundreds of hours using these tools over the course of their education (and thousands over their career), so it makes sense to make the experience as fluid and frictionless as possible. Mastering these tools not only enables you to spend less time on figuring out how to bend your tools to your will, but it also lets you solve problems that would previously seem impossibly complex.

Read about the motivation behind this class.

Schedule

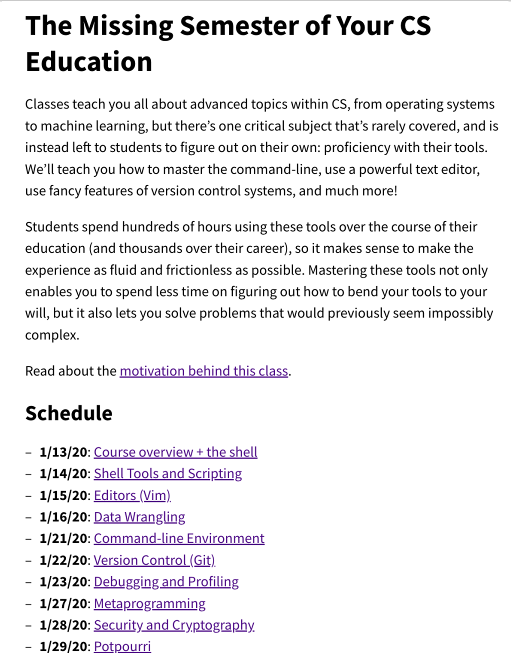
- **1/13/20**: Course overview + the shell
- **1/14/20**: Shell Tools and Scripting
- **1/15/20**: Editors (Vim)
- **1/16/20**: Data Wrangling
- **1/21/20**: Command-line Environment
- **1/22/20**: Version Control (Git)
- **1/23/20**: Debugging and Profiling
- **1/27/20**: Metaprogramming
- **1/28/20**: Security and Cryptography
- **1/29/20**: Potpourri

Build your research weapons and wheels

Check it out: `https://missing.csail.mit.edu/`

E.g.,

- Command shell
- Version control
- Text editing
- Remote machines
- Finding files



The Missing Semester of Your CS Education

Classes teach you all about advanced topics within CS, from operating systems to machine learning, but there's one critical subject that's rarely covered, and is instead left to students to figure out on their own: proficiency with their tools. We'll teach you how to master the command-line, use a powerful text editor, use fancy features of version control systems, and much more!

Students spend hundreds of hours using these tools over the course of their education (and thousands over their career), so it makes sense to make the experience as fluid and frictionless as possible. Mastering these tools not only enables you to spend less time on figuring out how to bend your tools to your will, but it also lets you solve problems that would previously seem impossibly complex.

Read about the motivation behind this class.

Schedule

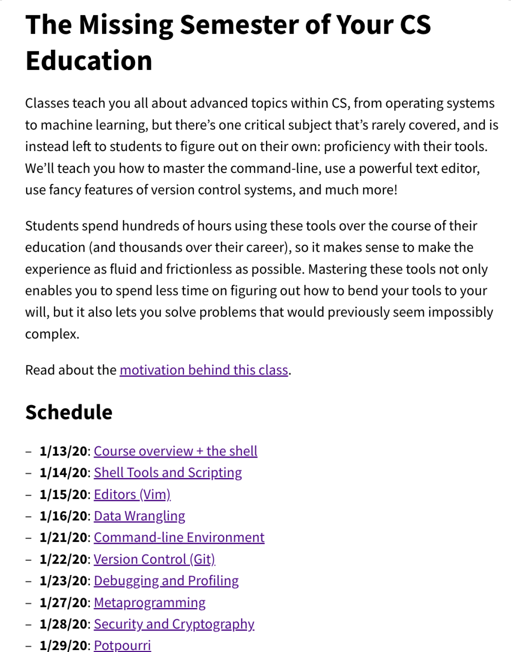
- **1/13/20**: Course overview + the shell
- **1/14/20**: Shell Tools and Scripting
- **1/15/20**: Editors (Vim)
- **1/16/20**: Data Wrangling
- **1/21/20**: Command-line Environment
- **1/22/20**: Version Control (Git)
- **1/23/20**: Debugging and Profiling
- **1/27/20**: Metaprogramming
- **1/28/20**: Security and Cryptography
- **1/29/20**: Potpourri

Build your research weapons and wheels

Check it out: `https://missing.csail.mit.edu/`

E.g.,

- Command shell
- Version control
- Text editing
- Remote machines
- Finding files
- Data wrangling



The Missing Semester of Your CS Education

Classes teach you all about advanced topics within CS, from operating systems to machine learning, but there's one critical subject that's rarely covered, and is instead left to students to figure out on their own: proficiency with their tools. We'll teach you how to master the command-line, use a powerful text editor, use fancy features of version control systems, and much more!

Students spend hundreds of hours using these tools over the course of their education (and thousands over their career), so it makes sense to make the experience as fluid and frictionless as possible. Mastering these tools not only enables you to spend less time on figuring out how to bend your tools to your will, but it also lets you solve problems that would previously seem impossibly complex.

Read about the motivation behind this class.

Schedule

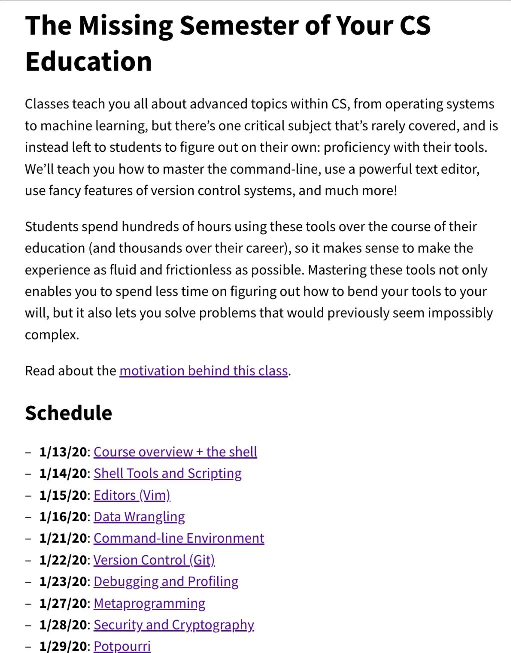
- **1/13/20**: Course overview + the shell
- **1/14/20**: Shell Tools and Scripting
- **1/15/20**: Editors (Vim)
- **1/16/20**: Data Wrangling
- **1/21/20**: Command-line Environment
- **1/22/20**: Version Control (Git)
- **1/23/20**: Debugging and Profiling
- **1/27/20**: Metaprogramming
- **1/28/20**: Security and Cryptography
- **1/29/20**: Potpourri

Build your research weapons and wheels

Check it out: `https://missing.csail.mit.edu/`

E.g.,

- Command shell
- Version control
- Text editing
- Remote machines
- Finding files
- Data wrangling
- Virtual machines



The Missing Semester of Your CS Education

Classes teach you all about advanced topics within CS, from operating systems to machine learning, but there's one critical subject that's rarely covered, and is instead left to students to figure out on their own: proficiency with their tools. We'll teach you how to master the command-line, use a powerful text editor, use fancy features of version control systems, and much more!

Students spend hundreds of hours using these tools over the course of their education (and thousands over their career), so it makes sense to make the experience as fluid and frictionless as possible. Mastering these tools not only enables you to spend less time on figuring out how to bend your tools to your will, but it also lets you solve problems that would previously seem impossibly complex.

Read about the motivation behind this class.

Schedule

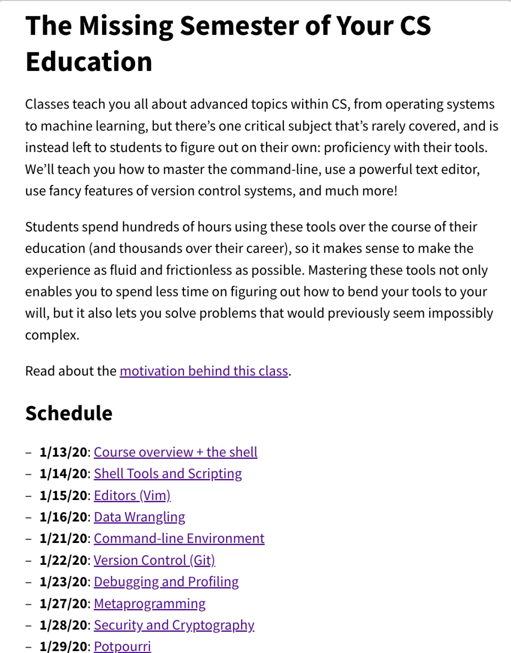
- **1/13/20**: Course overview + the shell
- **1/14/20**: Shell Tools and Scripting
- **1/15/20**: Editors (Vim)
- **1/16/20**: Data Wrangling
- **1/21/20**: Command-line Environment
- **1/22/20**: Version Control (Git)
- **1/23/20**: Debugging and Profiling
- **1/27/20**: Metaprogramming
- **1/28/20**: Security and Cryptography
- **1/29/20**: Potpourri

Build your research weapons and wheels

Check it out: `https://missing.csail.mit.edu/`

E.g.,

- Command shell
- Version control
- Text editing
- Remote machines
- Finding files
- Data wrangling
- Virtual machines
- Security



The Missing Semester of Your CS Education

Classes teach you all about advanced topics within CS, from operating systems to machine learning, but there's one critical subject that's rarely covered, and is instead left to students to figure out on their own: proficiency with their tools. We'll teach you how to master the command-line, use a powerful text editor, use fancy features of version control systems, and much more!

Students spend hundreds of hours using these tools over the course of their education (and thousands over their career), so it makes sense to make the experience as fluid and frictionless as possible. Mastering these tools not only enables you to spend less time on figuring out how to bend your tools to your will, but it also lets you solve problems that would previously seem impossibly complex.

Read about the motivation behind this class.

Schedule

- **1/13/20**: Course overview + the shell
- **1/14/20**: Shell Tools and Scripting
- **1/15/20**: Editors (Vim)
- **1/16/20**: Data Wrangling
- **1/21/20**: Command-line Environment
- **1/22/20**: Version Control (Git)
- **1/23/20**: Debugging and Profiling
- **1/27/20**: Metaprogramming
- **1/28/20**: Security and Cryptography
- **1/29/20**: Potpourri

Advanced CS: System, Theory, and AI

System

- Computer systems

Advanced CS: System, Theory, and AI

System

- Computer systems
- Distributed systems

Advanced CS: System, Theory, and AI

System

- Computer systems
- Distributed systems
- Network systems

Advanced CS: System, Theory, and AI

System

- Computer systems
- Distributed systems
- Network systems
- Embedded systems

Advanced CS: System, Theory, and AI

System

- Computer systems
- Distributed systems
- Network systems
- Embedded systems
- Compiler

Advanced CS: System, Theory, and AI

System

- Computer systems
- Distributed systems
- Network systems
- Embedded systems
- Compiler
- Software security

Advanced CS: System, Theory, and AI

System

- Computer systems
- Distributed systems
- Network systems
- Embedded systems
- Compiler
- Software security
- System-on-chip

Advanced CS: System, Theory, and AI

System

- Computer systems
- Distributed systems
- Network systems
- Embedded systems
- Compiler
- Software security
- System-on-chip
- Internet-of-Things systems

Advanced CS: System, Theory, and AI

System

- Computer systems
- Distributed systems
- Network systems
- Embedded systems
- Compiler
- Software security
- System-on-chip
- Internet-of-Things systems
- etc

Advanced CS: System, Theory, and AI

System

- Computer systems
- Distributed systems
- Network systems
- Embedded systems
- Compiler
- Software security
- System-on-chip
- Internet-of-Things systems
- etc

Theory

Advanced CS: System, Theory, and AI

System

- Computer systems
- Distributed systems
- Network systems
- Embedded systems
- Compiler
- Software security
- System-on-chip
- Internet-of-Things systems
- etc

Theory

- Distributed algorithms

Advanced CS: System, Theory, and AI

System

- Computer systems
- Distributed systems
- Network systems
- Embedded systems
- Compiler
- Software security
- System-on-chip
- Internet-of-Things systems
- etc

Theory

- Distributed algorithms
- Cryptography and security

Advanced CS: System, Theory, and AI

System

- Computer systems
- Distributed systems
- Network systems
- Embedded systems
- Compiler
- Software security
- System-on-chip
- Internet-of-Things systems
- etc

Theory

- Distributed algorithms
- Cryptography and security
- Formal verification

Advanced CS: System, Theory, and AI

System

- Computer systems
- Distributed systems
- Network systems
- Embedded systems
- Compiler
- Software security
- System-on-chip
- Internet-of-Things systems
- etc

Theory

- Distributed algorithms
- Cryptography and security
- Formal verification
- Computational complexity

Advanced CS: System, Theory, and AI

System

- Computer systems
- Distributed systems
- Network systems
- Embedded systems
- Compiler
- Software security
- System-on-chip
- Internet-of-Things systems
- etc

Theory

- Distributed algorithms
- Cryptography and security
- Formal verification
- Computational complexity
- Information theory and coding

Advanced CS: System, Theory, and AI

System

- Computer systems
- Distributed systems
- Network systems
- Embedded systems
- Compiler
- Software security
- System-on-chip
- Internet-of-Things systems
- etc

Theory

- Distributed algorithms
- Cryptography and security
- Formal verification
- Computational complexity
- Information theory and coding
- Computational linear algebra

Advanced CS: System, Theory, and AI

System

- Computer systems
- Distributed systems
- Network systems
- Embedded systems
- Compiler
- Software security
- System-on-chip
- Internet-of-Things systems
- etc

Theory

- Distributed algorithms
- Cryptography and security
- Formal verification
- Computational complexity
- Information theory and coding
- Computational linear algebra
- Statistical theory

Advanced CS: System, Theory, and AI

System

- Computer systems
- Distributed systems
- Network systems
- Embedded systems
- Compiler
- Software security
- System-on-chip
- Internet-of-Things systems
- etc

Theory

- Distributed algorithms
- Cryptography and security
- Formal verification
- Computational complexity
- Information theory and coding
- Computational linear algebra
- Statistical theory
- Quantum information theory and computation

Advanced CS: System, Theory, and AI

System

- Computer systems
- Distributed systems
- Network systems
- Embedded systems
- Compiler
- Software security
- System-on-chip
- Internet-of-Things systems
- etc

Theory

- Distributed algorithms
- Cryptography and security
- Formal verification
- Computational complexity
- Information theory and coding
- Computational linear algebra
- Statistical theory
- Quantum information theory and computation
- Dynamical system theory

Advanced CS: System, Theory, and AI

System

- Computer systems
- Distributed systems
- Network systems
- Embedded systems
- Compiler
- Software security
- System-on-chip
- Internet-of-Things systems
- etc

Theory

- Distributed algorithms
- Cryptography and security
- Formal verification
- Computational complexity
- Information theory and coding
- Computational linear algebra
- Statistical theory
- Quantum information theory and computation
- Dynamical system theory
- Advanced probability theory

Advanced CS: System, Theory, and AI

System

- Computer systems
- Distributed systems
- Network systems
- Embedded systems
- Compiler
- Software security
- System-on-chip
- Internet-of-Things systems
- etc

Theory

- Distributed algorithms
- Cryptography and security
- Formal verification
- Computational complexity
- Information theory and coding
- Computational linear algebra
- Statistical theory
- Quantum information theory and computation
- Dynamical system theory
- Advanced probability theory
- etc

Advanced CS: System, Theory, and AI

System

- Computer systems
- Distributed systems
- Network systems
- Embedded systems
- Compiler
- Software security
- System-on-chip
- Internet-of-Things systems
- etc

Theory

- Distributed algorithms
- Cryptography and security
- Formal verification
- Computational complexity
- Information theory and coding
- Computational linear algebra
- Statistical theory
- Quantum information theory and computation
- Dynamical system theory
- Advanced probability theory
- etc

AI

Advanced CS: System, Theory, and AI

System

- Computer systems
- Distributed systems
- Network systems
- Embedded systems
- Compiler
- Software security
- System-on-chip
- Internet-of-Things systems
- etc

Theory

- Distributed algorithms
- Cryptography and security
- Formal verification
- Computational complexity
- Information theory and coding
- Computational linear algebra
- Statistical theory
- Quantum information theory and computation
- Dynamical system theory
- Advanced probability theory
- etc

AI

- Statistical Machine Learning

Advanced CS: System, Theory, and AI

System

- Computer systems
- Distributed systems
- Network systems
- Embedded systems
- Compiler
- Software security
- System-on-chip
- Internet-of-Things systems
- etc

Theory

- Distributed algorithms
- Cryptography and security
- Formal verification
- Computational complexity
- Information theory and coding
- Computational linear algebra
- Statistical theory
- Quantum information theory and computation
- Dynamical system theory
- Advanced probability theory
- etc

AI

- Statistical Machine Learning
- Deep Learning

Advanced CS: System, Theory, and AI

System

- Computer systems
- Distributed systems
- Network systems
- Embedded systems
- Compiler
- Software security
- System-on-chip
- Internet-of-Things systems
- etc

Theory

- Distributed algorithms
- Cryptography and security
- Formal verification
- Computational complexity
- Information theory and coding
- Computational linear algebra
- Statistical theory
- Quantum information theory and computation
- Dynamical system theory
- Advanced probability theory
- etc

AI

- Statistical Machine Learning
- Deep Learning
- Learning theory

Advanced CS: System, Theory, and AI

System

- Computer systems
- Distributed systems
- Network systems
- Embedded systems
- Compiler
- Software security
- System-on-chip
- Internet-of-Things systems
- etc

Theory

- Distributed algorithms
- Cryptography and security
- Formal verification
- Computational complexity
- Information theory and coding
- Computational linear algebra
- Statistical theory
- Quantum information theory and computation
- Dynamical system theory
- Advanced probability theory
- etc

AI

- Statistical Machine Learning
- Deep Learning
- Learning theory
- Optimization for Machine Learning

Advanced CS: System, Theory, and AI

System

- Computer systems
- Distributed systems
- Network systems
- Embedded systems
- Compiler
- Software security
- System-on-chip
- Internet-of-Things systems
- etc

Theory

- Distributed algorithms
- Cryptography and security
- Formal verification
- Computational complexity
- Information theory and coding
- Computational linear algebra
- Statistical theory
- Quantum information theory and computation
- Dynamical system theory
- Advanced probability theory
- etc

AI

- Statistical Machine Learning
- Deep Learning
- Learning theory
- Optimization for Machine Learning
- Multi-agents

Advanced CS: System, Theory, and AI

System

- Computer systems
- Distributed systems
- Network systems
- Embedded systems
- Compiler
- Software security
- System-on-chip
- Internet-of-Things systems
- etc

Theory

- Distributed algorithms
- Cryptography and security
- Formal verification
- Computational complexity
- Information theory and coding
- Computational linear algebra
- Statistical theory
- Quantum information theory and computation
- Dynamical system theory
- Advanced probability theory
- etc

AI

- Statistical Machine Learning
- Deep Learning
- Learning theory
- Optimization for Machine Learning
- Multi-agents
- Computational photography

Advanced CS: System, Theory, and AI

System

- Computer systems
- Distributed systems
- Network systems
- Embedded systems
- Compiler
- Software security
- System-on-chip
- Internet-of-Things systems
- etc

Theory

- Distributed algorithms
- Cryptography and security
- Formal verification
- Computational complexity
- Information theory and coding
- Computational linear algebra
- Statistical theory
- Quantum information theory and computation
- Dynamical system theory
- Advanced probability theory
- etc

AI

- Statistical Machine Learning
- Deep Learning
- Learning theory
- Optimization for Machine Learning
- Multi-agents
- Computational photography
- Computer vision

Advanced CS: System, Theory, and AI

System

- Computer systems
- Distributed systems
- Network systems
- Embedded systems
- Compiler
- Software security
- System-on-chip
- Internet-of-Things systems
- etc

Theory

- Distributed algorithms
- Cryptography and security
- Formal verification
- Computational complexity
- Information theory and coding
- Computational linear algebra
- Statistical theory
- Quantum information theory and computation
- Dynamical system theory
- Advanced probability theory
- etc

AI

- Statistical Machine Learning
- Deep Learning
- Learning theory
- Optimization for Machine Learning
- Multi-agents
- Computational photography
- Computer vision
- Virtual reality

Advanced CS: System, Theory, and AI

System

- Computer systems
- Distributed systems
- Network systems
- Embedded systems
- Compiler
- Software security
- System-on-chip
- Internet-of-Things systems
- etc

Theory

- Distributed algorithms
- Cryptography and security
- Formal verification
- Computational complexity
- Information theory and coding
- Computational linear algebra
- Statistical theory
- Quantum information theory and computation
- Dynamical system theory
- Advanced probability theory
- etc

AI

- Statistical Machine Learning
- Deep Learning
- Learning theory
- Optimization for Machine Learning
- Multi-agents
- Computational photography
- Computer vision
- Virtual reality
- Computational neurosciences

Advanced CS: System, Theory, and AI

System

- Computer systems
- Distributed systems
- Network systems
- Embedded systems
- Compiler
- Software security
- System-on-chip
- Internet-of-Things systems
- etc

Theory

- Distributed algorithms
- Cryptography and security
- Formal verification
- Computational complexity
- Information theory and coding
- Computational linear algebra
- Statistical theory
- Quantum information theory and computation
- Dynamical system theory
- Advanced probability theory
- etc

AI

- Statistical Machine Learning
- Deep Learning
- Learning theory
- Optimization for Machine Learning
- Multi-agents
- Computational photography
- Computer vision
- Virtual reality
- Computational neurosciences
- Reinforcement learning

Advanced CS: System, Theory, and AI

System

- Computer systems
- Distributed systems
- Network systems
- Embedded systems
- Compiler
- Software security
- System-on-chip
- Internet-of-Things systems
- etc

Theory

- Distributed algorithms
- Cryptography and security
- Formal verification
- Computational complexity
- Information theory and coding
- Computational linear algebra
- Statistical theory
- Quantum information theory and computation
- Dynamical system theory
- Advanced probability theory
- etc

AI

- Statistical Machine Learning
- Deep Learning
- Learning theory
- Optimization for Machine Learning
- Multi-agents
- Computational photography
- Computer vision
- Virtual reality
- Computational neurosciences
- Reinforcement learning
- Online learning in games

Advanced CS: System, Theory, and AI

System

- Computer systems
- Distributed systems
- Network systems
- Embedded systems
- Compiler
- Software security
- System-on-chip
- Internet-of-Things systems
- etc

Theory

- Distributed algorithms
- Cryptography and security
- Formal verification
- Computational complexity
- Information theory and coding
- Computational linear algebra
- Statistical theory
- Quantum information theory and computation
- Dynamical system theory
- Advanced probability theory
- etc

AI

- Statistical Machine Learning
- Deep Learning
- Learning theory
- Optimization for Machine Learning
- Multi-agents
- Computational photography
- Computer vision
- Virtual reality
- Computational neurosciences
- Reinforcement learning
- Online learning in games
- Natural Language Processing

Advanced CS: System, Theory, and AI

System

- Computer systems
- Distributed systems
- Network systems
- Embedded systems
- Compiler
- Software security
- System-on-chip
- Internet-of-Things systems
- etc

Theory

- Distributed algorithms
- Cryptography and security
- Formal verification
- Computational complexity
- Information theory and coding
- Computational linear algebra
- Statistical theory
- Quantum information theory and computation
- Dynamical system theory
- Advanced probability theory
- etc

AI

- Statistical Machine Learning
- Deep Learning
- Learning theory
- Optimization for Machine Learning
- Multi-agents
- Computational photography
- Computer vision
- Virtual reality
- Computational neurosciences
- Reinforcement learning
- Online learning in games
- Natural Language Processing
- etc

Dive into AI

Table of Contents

Table of Contents

Birth of AI (1943 - 1956)

- **1943** $\rightarrow$ Warren McCulloch & Walter Pitts

  Artificial Neural Network (ANN) with on-off neurons can represent any computable function (**Universal Approximation Theorem**).

Birth of AI (1943 - 1956)

- **1943** $\rightarrow$ Warren McCulloch & Walter Pitts

  Artificial Neural Network (ANN) with on-off neurons can represent any computable function (**Universal Approximation Theorem**).
- **1950** $\rightarrow$ Alan Turing

Birth of AI (1943 - 1956)

- **1943** $\rightarrow$ Warren McCulloch & Walter Pitts

  Artificial Neural Network (ANN) with on-off neurons can represent any computable function (**Universal Approximation Theorem**).
- **1950** $\rightarrow$ Alan Turing
  - **Turing Test**: a machine is said to pass the Turing test if it can convince a human judge that it is actually a human through natural language dialogue (indistinguishability).

Birth of AI (1943 - 1956)

- **1943** $\rightarrow$ Warren McCulloch & Walter Pitts

  Artificial Neural Network (ANN) with on-off neurons can represent any computable function (**Universal Approximation Theorem**).
- **1950** $\rightarrow$ Alan Turing
  - **Turing Test**: a machine is said to pass the Turing test if it can convince a human judge that it is actually a human through natural language dialogue (indistinguishability).
  - Two approaches trying to pass the Turing test:

Birth of AI (1943 - 1956)

- **1943** $\rightarrow$ Warren McCulloch & Walter Pitts

  Artificial Neural Network (ANN) with on-off neurons can represent any computable function (**Universal Approximation Theorem**).

- **1950** $\rightarrow$ Alan Turing
  - **Turing Test**: a machine is said to pass the Turing test if it can convince a human judge that it is actually a human through natural language dialogue (indistinguishability).
  - Two approaches trying to pass the Turing test:
    1. Rule-based symbolic AI

Birth of AI (1943 - 1956)

- **1943** $\rightarrow$ Warren McCulloch & Walter Pitts

  Artificial Neural Network (ANN) with on-off neurons can represent any computable function (**Universal Approximation Theorem**).

- **1950** $\rightarrow$ Alan Turing
  - **Turing Test**: a machine is said to pass the Turing test if it can convince a human judge that it is actually a human through natural language dialogue (indistinguishability).
  - Two approaches trying to pass the Turing test:
    1. Rule-based symbolic AI
    2. Training-based neural and statistical AI.

Birth of AI (1943 - 1956)

- **1943** $\rightarrow$ Warren McCulloch & Walter Pitts

  Artificial Neural Network (ANN) with on-off neurons can represent any computable function (**Universal Approximation Theorem**).
- **1950** $\rightarrow$ Alan Turing
  - **Turing Test**: a machine is said to pass the Turing test if it can convince a human judge that it is actually a human through natural language dialogue (indistinguishability).
  - Two approaches trying to pass the Turing test:
    1. Rule-based symbolic AI
    2. Training-based neural and statistical AI.
- **1956** $\rightarrow$ John McCarthy
  He organized a workshop at Dartmouth College.

Birth of AI (1943 - 1956)

- **1943** $\rightarrow$ Warren McCulloch & Walter Pitts

  Artificial Neural Network (ANN) with on-off neurons can represent any computable function (**Universal Approximation Theorem**).
- **1950** $\rightarrow$ Alan Turing
  - **Turing Test**: a machine is said to pass the Turing test if it can convince a human judge that it is actually a human through natural language dialogue (indistinguishability).
  - Two approaches trying to pass the Turing test:
    1. Rule-based symbolic AI
    2. Training-based neural and statistical AI.
- **1956** $\rightarrow$ John McCarthy
  He organized a workshop at Dartmouth College.
  1. Term “artificial intelligence” was coined.

Birth of AI (1943 - 1956)

- **1943** $\rightarrow$ Warren McCulloch & Walter Pitts

  Artificial Neural Network (ANN) with on-off neurons can represent any computable function (**Universal Approximation Theorem**).

- **1950** $\rightarrow$ Alan Turing
  - **Turing Test**: a machine is said to pass the Turing test if it can convince a human judge that it is actually a human through natural language dialogue (indistinguishability).
  - Two approaches trying to pass the Turing test:
    1. Rule-based symbolic AI
    2. Training-based neural and statistical AI.

- **1956** $\rightarrow$ John McCarthy
  He organized a workshop at Dartmouth College.
  1. Term “artificial intelligence” was coined.
  2. The participants laid out a bold proposal: to build a system that could capture every aspect of intelligence, which is called **strong AI** or **Artificial General Intelligence**.

Birth of AI (1943 - 1956)

- **1943** $\rightarrow$ Warren McCulloch & Walter Pitts

  Artificial Neural Network (ANN) with on-off neurons can represent any computable function (**Universal Approximation Theorem**).

- **1950** $\rightarrow$ Alan Turing
  - **Turing Test**: a machine is said to pass the Turing test if it can convince a human judge that it is actually a human through natural language dialogue (indistinguishability).
  - Two approaches trying to pass the Turing test:
    1. Rule-based symbolic AI
    2. Training-based neural and statistical AI.

- **1956** $\rightarrow$ John McCarthy
  He organized a workshop at Dartmouth College.
  1. Term "artificial intelligence" was coined.
  2. The participants laid out a bold proposal: to build a system that could capture every aspect of intelligence, which is called **strong AI** or **Artificial General Intelligence**.
  3. In contrast to strong AI, **weak AI** is any program that is designed to solve exactly one problem.

Early Success (1952 - 1959)

Claims: Computers can do X (weak AI)

Early Success (1952 - 1959)

Claims: Computers can do X (weak AI)

- Checkers (1952): Arthur Samuel's program played checkers games at the strong amateur level. (Wikipedia)

Early Success (1952 - 1959)

Claims: Computers can do X (weak AI)

- Checkers (1952): Arthur Samuel's program played checkers games at the strong amateur level. (Wikipedia)
- Logic Theorist: Allen Newell, Herbert A. Simon, and Cliff Shaw's computer program was deliberately engineered to perform automated reasoning. (Wikipedia)

Early Success (1952 - 1959)

Claims: Computers can do X (weak AI)

- Checkers (1952): Arthur Samuel's program played checkers games at the strong amateur level. (Wikipedia)
- Logic Theorist: Allen Newell, Herbert A. Simon, and Cliff Shaw's computer program was deliberately engineered to perform automated reasoning. (Wikipedia)
- LISP (1958): a favored programming language for AI.

Overwhelming Optimism and Underwhelming Results (1960s)

Overwhelming Optimism and Underwhelming Results (1960s)

- Herbert Simon: Machines will be capable, within twenty years, of doing any work a man can do.

Overwhelming Optimism and Underwhelming Results (1960s)

- Herbert Simon: Machines will be capable, within twenty years, of doing any work a man can do.
- Marvin Minsky: Within 10 years the problems of artificial intelligence will be substantially solved.

Overwhelming Optimism and Underwhelming Results (1960s)

- Herbert Simon: Machines will be capable, within twenty years, of doing any work a man can do.
- Marvin Minsky: Within 10 years the problems of artificial intelligence will be substantially solved.
- Claude Shannon: I visualize a time when we will be to robots what dogs are to humans, and I'm rooting for the machines.

Overwhelming Optimism and Underwhelming Results (1960s)

- Herbert Simon: Machines will be capable, within twenty years, of doing any work a man can do.
- Marvin Minsky: Within 10 years the problems of artificial intelligence will be substantially solved.
- Claude Shannon: I visualize a time when we will be to robots what dogs are to humans, and I'm rooting for the machines.

Underwhelming Results:

The report by the Automatic Language Processing Advisory Committee (ALPAC) in 1966 resulted in the government funding cut for Machine Translation (MT), causing the first AI winter.

Knowledge-base System (1970s - 1980s)

- **Expert systems**: elicit specific domain knowledge from experts in the form of rules:

  If [premises] then [conclusion]

Knowledge-base System (1970s - 1980s)

- **Expert systems**: elicit specific domain knowledge from experts in the form of rules:

  If [premises] then [conclusion]

- Example: in autonomous driving

Knowledge-base System (1970s - 1980s)

- **Expert systems**: elicit specific domain knowledge from experts in the form of rules:

  If [premises] then [conclusion]

- Example: in autonomous driving
  - If [Case 1: a pedestrian], then use [Rule 1: stop];

Knowledge-base System (1970s - 1980s)

- **Expert systems**: elicit specific domain knowledge from experts in the form of rules:

  If [premises] then [conclusion]

- Example: in autonomous driving
  - If [Case 1: a pedestrian], then use [Rule 1: stop];
  - If [Case 2: an obstacle], then use [Rule 2: get around];

Knowledge-base System (1970s - 1980s)

- **Expert systems**: elicit specific domain knowledge from experts in the form of rules:

  If [premises] then [conclusion]

- Example: in autonomous driving
  - If [Case 1: a pedestrian], then use [Rule 1: stop];
  - If [Case 2: an obstacle], then use [Rule 2: get around];
  - ...

AI Becomes an Industry (1980 - 1988)

AI Becomes an Industry (1980 - 1988)

- R1: First successful commercial expert system, configured computer systems at Digital Equipment Corporation (DEC) and saved 40 million dollars per year.

  (https://www.sciencedirect.com/science/article/abs/pii/0004370282900212)

AI Becomes an Industry (1980 - 1988)

- R1: First successful commercial expert system, configured computer systems at Digital Equipment Corporation (DEC) and saved 40 million dollars per year.

  (https://www.sciencedirect.com/science/article/abs/pii/0004370282900212)

- DEC had 40 expert systems by 1988, such as DuPont 100.

AI Becomes an Industry (1980 - 1988)

- R1: First successful commercial expert system, configured computer systems at Digital Equipment Corporation (DEC) and saved 40 million dollars per year.

  (https://www.sciencedirect.com/science/article/abs/pii/0004370282900212)
- DEC had 40 expert systems by 1988, such as DuPont 100.
- Software tools for expert systems emerged: Carnegie Group, Inference, Intellicorp and Teknowledge.

AI Becomes an Industry (1980 - 1988)

- R1: First successful commercial expert system, configured computer systems at Digital Equipment Corporation (DEC) and saved 40 million dollars per year.
  (https://www.sciencedirect.com/science/article/abs/pii/0004370282900212)
- DEC had 40 expert systems by 1988, such as DuPont 100.
- Software tools for expert systems emerged: Carnegie Group, Inference, Intellicorp and Teknowledge.
- LISP-specific hardware developed: LISP Machines Inc, TI, Symbolics and Xerox.

AI Becomes an Industry (1980 - 1988)

- R1: First successful commercial expert system, configured computer systems at Digital Equipment Corporation (DEC) and saved 40 million dollars per year.

  (https://www.sciencedirect.com/science/article/abs/pii/0004370282900212)
- DEC had 40 expert systems by 1988, such as DuPont 100.
- Software tools for expert systems emerged: Carnegie Group, Inference, Intellicorp and Teknowledge.
- LISP-specific hardware developed: LISP Machines Inc, TI, Symbolics and Xerox.
- AI industry boosts: from a few million dollars in 1980 to about two billion dollars in 1988.

Knowledge-based Systems (1980s)

- Features:

Knowledge-based Systems (1980s)

- Features:
    - Manually constructed knowledge helps, as compared to automatic learning from data.

Knowledge-based Systems (1980s)

- Features:
  - Manually constructed knowledge helps, as compared to automatic learning from data.
  - First real application that impacted the industry.

Knowledge-based Systems (1980s)

- Features:
  - Manually constructed knowledge helps, as compared to automatic learning from data.
  - First real application that impacted the industry.
- Limitations:

Knowledge-based Systems (1980s)

- Features:
  - Manually constructed knowledge helps, as compared to automatic learning from data.
  - First real application that impacted the industry.
- Limitations:
  - Deterministic rules cannot handle the uncertainty of the real world.

Knowledge-based Systems (1980s)

- Features:
  - Manually constructed knowledge helps, as compared to automatic learning from data.
  - First real application that impacted the industry.
- Limitations:
  - Deterministic rules cannot handle the uncertainty of the real world.
  - Rules quickly became too complex to create and maintain.

Knowledge-based Systems (1980s)

- Features:
  - Manually constructed knowledge helps, as compared to automatic learning from data.
  - First real application that impacted the industry.
- Limitations:
  - Deterministic rules cannot handle the uncertainty of the real world.
  - Rules quickly became too complex to create and maintain.
- 1987: collapse of LISP machines, second AI winter.

Knowledge-based Systems (1980s)

- Features:
  - Manually constructed knowledge helps, as compared to automatic learning from data.
  - First real application that impacted the industry.
- Limitations:
  - Deterministic rules cannot handle the uncertainty of the real world.
  - Rules quickly became too complex to create and maintain.
- 1987: collapse of LISP machines, second AI winter.
- End of symbolic AI domination for multiple decades.

Return of ANNs (1986 -)

Return of ANNs (1986 -)

- Disillusionment with expert systems led to the resurgence of ANNs in the mid-1980s.

Return of ANNs (1986 -)

- Disillusionment with expert systems led to the resurgence of ANNs in the mid-1980s.
- The history of neural AI dates back to 1943.

Inspiration from Neuroscience

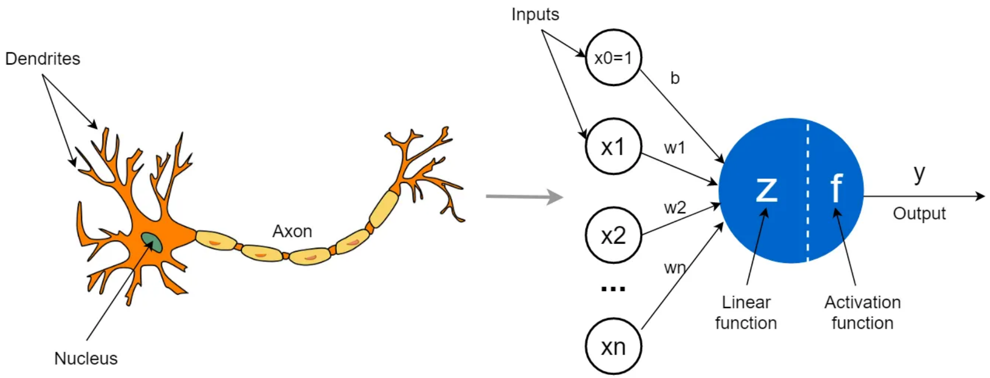


Figure: Human neurons (left) and artificial neurons in ANN (right). Image from towardsdatascience.com.

Early Research on ANNs (1943 - 1969)

Early Research on ANNs (1943 - 1969)

- Warren McCulloch & Walter Pitts (1943):

  artificial neural networks, which is related to neural circuitry and mathematical logic.

Early Research on ANNs (1943 - 1969)

- Warren McCulloch & Walter Pitts (1943):

  artificial neural networks, which is related to neural circuitry and mathematical logic.
- Donald O. Hebb (1949):

  “cells that fire together wire together”.

Early Research on ANNs (1943 - 1969)

- Warren McCulloch & Walter Pitts (1943):

  artificial neural networks, which is related to neural circuitry and mathematical logic.

- Donald O. Hebb (1949):

  “cells that fire together wire together”.

- Frank Rosenblatt (1958):

  Perceptron algorithm for the linear classifier.

Early Research on ANNs (1943 - 1969)

- Warren McCulloch & Walter Pitts (1943):

  artificial neural networks, which is related to neural circuitry and mathematical logic.
- Donald O. Hebb (1949):

  “cells that fire together wire together”.
- Frank Rosenblatt (1958):

  Perceptron algorithm for the linear classifier.
- Bernard Widrow & Ted Hoff (1959):

  adaptive linear neuron (ADALINE).

Early Research on ANNs (1943 - 1969)

- Warren McCulloch & Walter Pitts (1943):

  artificial neural networks, which is related to neural circuitry and mathematical logic.

- Donald O. Hebb (1949):

  “cells that fire together wire together”.

- Frank Rosenblatt (1958):

  Perceptron algorithm for the linear classifier.

- Bernard Widrow & Ted Hoff (1959):

  adaptive linear neuron (ADALINE).

- Marvin Minsky & Seymour Papert (1969):

  Percepron book showed that linear models could not solve XOR,
  which killed neural net research.

Revival of Connectionism (1980 -)

Revival of Connectionism (1980 -)

- Kunihiko Fukushima (1980):

  Neocognitron, a.k.a., convolutional neural networks for images.

Revival of Connectionism (1980 -)

- Kunihiko Fukushima (1980):
  Neocognitron, a.k.a., convolutional neural networks for images.
- David E. Rumelhart, Geoffrey E. Hinton & Ronald J. Williams (1986):
  backpropagation technique for training multi-layer neural networks.

Revival of Connectionism (1980 -)

- Kunihiko Fukushima (1980):
  Neocognitron, a.k.a., convolutional neural networks for images.
- David E. Rumelhart, Geoffrey E. Hinton & Ronald J. Williams (1986):
  backpropagation technique for training multi-layer neural networks.
- Yann LeCun (1989):
  applied a convolutional neural network to recognize handwritten digits for USPS.

The era of Deep Learning (2000s -)

The era of Deep Learning (2000s -)

- Geoffrey E. Hinton et. al. (2006):

  unsupervised layerwise pre-training of deep networks.

The era of Deep Learning (2000s -)

- Geoffrey E. Hinton et. al. (2006):

  unsupervised layerwise pre-training of deep networks.

- Alex Krizhevsky, Ilya Sutskever & Geoffrey E. Hinton (2012):

  AlexNet obtained huge gains in object recognition
  and transformed the computer vision community overnight.

The era of Deep Learning (2000s -)

- Geoffrey E. Hinton et. al. (2006):

  unsupervised layerwise pre-training of deep networks.

- Alex Krizhevsky, Ilya Sutskever & Geoffrey E. Hinton (2012):

  AlexNet obtained huge gains in object recognition
  and transformed the computer vision community overnight.

- David Sliver et. al. (2016):

  AlphaGo used deep reinforcement learning, and defeated the world champion in Go.

The era of Deep Learning (2000s -)

- Geoffrey E. Hinton et. al. (2006):

  unsupervised layerwise pre-training of deep networks.

- Alex Krizhevsky, Ilya Sutskever & Geoffrey E. Hinton (2012):

  AlexNet obtained huge gains in object recognition
  and transformed the computer vision community overnight.

- David Sliver et. al. (2016):

  AlphaGo used deep reinforcement learning, and defeated the world champion in Go.

- Many (2022):

  large language model and generative pre-trained transformer (GPT)
  greatly improve the performance of the generative model. e.g. ChatGPT, GPT-4.

Ideas from Outside AI: Algebra and Statistics

- Carl F. Gauss (1801): linear regression.
- Ronald Fisher (1936): linear classification.
- Richard Bellman (1953): dynamic programming, Markov decision processes.
- Judea Pearl (1985): Bayesian networks.
- Corinna Cortes & Vladimir Vapnik (1995): support vector machine (SVM).

AI is Multi-disciplinary

- Mathematics (Algebra & Statistics)
- Optimization
- Neuroscience
- Computer Software
- Computer System

Table of Contents

How to Become a Good AI Researcher

How to Become a Good AI Researcher

- Master math and CS skills

How to Become a Good AI Researcher

- Master math and CS skills

If you are not ready, please check

- the materials listed before
- `https://csdiy.wiki/`
- books e.g., Algebra, Topology, Differential Calculus, and Optimization Theory For CS and ML

How to Become a Good AI Researcher

- Master math and CS skills

If you are not ready, please check

- the materials listed before
- `https://csdiy.wiki/`
- books e.g., Algebra, Topology, Differential Calculus, and Optimization Theory For CS and ML

- Some implicit skills (what we will introduce in this lecture: **A Survival Guide to a Ph.D.**)

How to Become a Good AI Researcher

- Master math and CS skills

> If you you are not ready, please check
> - the materials listed before
> - `https://csdiy.wiki/`
> - books e.g., Algebra, Topology, Differential Calculus, and Optimization Theory For CS and ML

- Some implicit skills (what we will introduce in this lecture: **A Survival Guide to a Ph.D.**)
  - effective communication strategy

How to Become a Good AI Researcher

- Master math and CS skills

If you are not ready, please check

- the materials listed before
- `https://csdiy.wiki/`
- books e.g., Algebra, Topology, Differential Calculus, and Optimization Theory For CS and ML

- Some implicit skills (what we will introduce in this lecture: **A Survival Guide to a Ph.D.**)
  - effective communication strategy
  - how to do research

How to Become a Good AI Researcher

- Master math and CS skills

If you are not ready, please check

- the materials listed before
- `https://csdiy.wiki/`
- books e.g., Algebra, Topology, Differential Calculus, and Optimization Theory For CS and ML

- Some implicit skills (what we will introduce in this lecture: **A Survival Guide to a Ph.D.**)
  - effective communication strategy
  - how to do research
  - how to give a great research talk

How to Become a Good AI Researcher

- Master math and CS skills

If you are not ready, please check

- the materials listed before
- `https://csdiy.wiki/`
- books e.g., Algebra, Topology, Differential Calculus, and Optimization Theory For CS and ML

- Some implicit skills (what we will introduce in this lecture: **A Survival Guide to a Ph.D.**)
  - effective communication strategy
  - how to do research
  - how to give a great research talk
  - etc

How to Become a Good AI Researcher

- Master math and CS skills

If you are not ready, please check

- the materials listed before
- `https://csdiy.wiki/`
- books e.g., Algebra, Topology, Differential Calculus, and Optimization Theory For CS and ML

- Some implicit skills (what we will introduce in this lecture: **A Survival Guide to a Ph.D.**)
  - effective communication strategy
  - how to do research
  - how to give a great research talk
  - etc
- Practice these skills and make steady progress in your research

Course Schedule

Week	Date	Topics
1	2024. Sep. 03	Introduction to CS & AI
2	2024. Sep. 10	How to communicate
3	2024. Sep. 17	How to do presentation
4	2024. Sep. 24	How to be a good AI researcher (I): doing research I
5	2024. Oct. 07	How to be a good AI researcher (II): productivity and career
6	2024. Oct. 15	How to be a good AI researcher (III): academic paper writing and peer reviews
7	2024. Oct. 22	Sharing the experience of writing excellent academic papers and rebuttal
8	2024. Oct. 29	Practice course

How to communicate

How to do presentation

How to be a good AI researcher (I): doing research I

How to be a good AI researcher (II): productivity and career

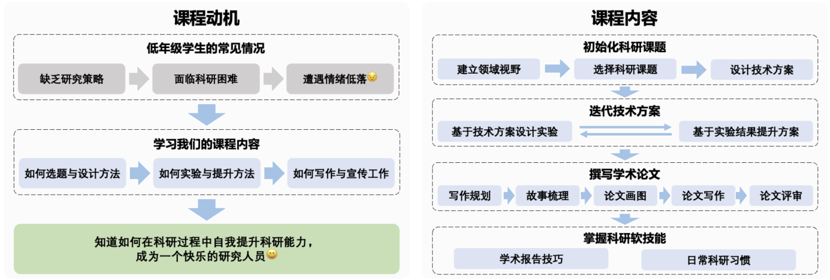
How to be a good AI researcher (III): academic paper writing and peer reviews

Parallel course

GAMES003: 科研基本素养

2024 年秋季学期（在线直播）

课程介绍



本课程为初学者展示了一条全面的学术研究路径，旨在引领大家以系统性的方法探索计算机视觉和图形学领域的科学前沿。我们将指导大家从建立领域视野，到选择科研课题、设计技术方案，再到设计实验、优化方案、管理论文投稿、设计论文图表、撰写论文、自我评审与rebuttal，以及学习做学术报告的技巧，覆盖了科研过程中的每一个关键步骤。课程中，我们将结合具体案例，分享科研经验，同时鼓励学生提出问题，以实现具象的科研素养教学。

具体课程内容请参见课程大纲。

https://pengsida.net/games003/

Thanks & Question Time!