Syllabus for Using ML Concepts
Spring 2023
CSCI 3387

People

- Instructor: Prof. José Bento. **Tip:** To maximize your chances of reaching me use class time, CANVAS messaging, and office hours, where I can meet you in person, individually or in groups. I receive many emails, so using email is a bad method to grab my attention.
- TA: Jeremy Chan. Not sure how the TA likes to be contacted, but his email is **channq@bc.edu**
- Extra support: **Tip**: The Connors Family Learning Center offers great tutoring services on a variety of CS related topics. I encourage you to use them if necessary. Link

Office hours

- Prof. Bento: Wednesday 3:00pm-4:00pm. In person, or Zoom (https://bccte.zoom.us/j/2721566389), or by appointment: message me in CANVA.
    - **Tip:** I strongly recommend you maximizing your interaction with anyone involved with the course: me, TA, your colleagues, etc. The more you speak and discuss about the topics, the quicker you will learn. Coming to office hours is obviously a good way to do so.
- TA office hour: : Jeremy Chan will communicate this to you all.

Class schedule

Monday, Wednesday: 2:00pm to 2:50pm, 245 Beacon St, room 229.

During lecture time **no cellphones/computers/tablets/players/etc, and no food, are allowed,** unless I ask you to use it. You are, of course, very welcome to step outside for a snack, phone call, or a break, and them come back. Attendance is mandatory for those that willingly take my course, but remember, you are free,

you do not have to be in the lecture room, or take my course, but if you willingly want to be there, you need to respect these minimal set of rules.

Prerequisites

Being fluid in

- Calculus
- Automata
- Probability
- Algorithms

Unfortunately, having taken and forgotten the content of these courses does not fulfill the prerequisites. I strongly recommend that you to refresh your memory. I try to make all of my lectures as self-contained as possible. **Tip**: You are encouraged to ask questions about prerequisites during lectures, or in office hours. You can also use the TA, or the tutoring services aforementioned, to refresh those prerequisites. However, this class is not a class on Calculus, or Automata theory, Probability or Algorithms, which I assume you have already mastered. Unfortunately, I cannot devote most of my time to teach you these topics from scratch, at the expense of not teaching ML to the students that do have the required background, and have the right to see the class topics covered in class. To help each student self-assess if he is ready to take this class, I will give a prerequisite quiz in the first lecture. Your score in this quiz **will not** count towards your class grade. Your score will help you self-assess if you are ready to take this class this year. There is nothing wrong in delaying one year to take this class.

If you understand everything that is reviewed in class sessions and you can apply all of those skills on the homework problems, you will probably do well in the class. However, you might be asked to extrapolate the main techniques and ideas learnt in class to examples or problems that you have not seen before, just like when learning e.g., English, one learns some specific words and reads some specific texts in class, but then combine these "atoms" to produce new texts.

Reading

Books are expensive, and not everyone can afford them. There is no mandatory book for the class. I try to make all of my lectures as self-contained as possible (prerequisites assumed). However, for those that do want to read a book, there are many good books on ML from a "usage" point of view. The following list of

books is as good as any. Link. Use Google, and our library, explore a bit, and see which one you like the most.

Course topics

This class is about learning how to use machine learning concepts. The focus is on understanding enough about different methods so that we can use them properly. Some of my explanations involve different degrees of mathematics but the main focus is not the theory behind the different methods.

The level of detail with which I will cover the different class topics will vary with students' interests and time available. This, the schedule below is tentative and will most likely be revised as we move along.

During class you will help me code and test some of the concepts explained. You will have to finish some coding and testing at home.

Note that existing books might not cover everything that I teach in class, or the same way that I cover it. Many go way beyond what is covered in class. To guarantee that I cover everything in the syllabus, sometimes I might record a lecture and put it online for student viewing.

Tip: Use the list of objectives in the table below to self-assess your learning. Note: these objectives are guidelines. There is more to each topic than what can be covered in a syllabus table. A high-achieving student should feel comfortable in (i) establishing connections between the different topics, not see them in isolation, (ii) applying the main ideas in each topic beyond the examples covered in class, (iii) solving the same problem in several different ways (when possible).

Topics	Lecture	Dates
Intro + background check	0	18 Jan
Math review: vectors, matrices, probability, calculus	1	20 Jan
	2	23 Jan
	3	25 Jan
Classification algorithms: Perceptron	4	27 Jan
Adaline, (stochastic) gradient descent	5	30 Jan
Logistic regression	6	1 Feb
Support vector machines	7	3 Feb
Decision trees	8	6 Feb

k-nearest neighbors	9	8 Feb
Handling data: missing data, non-numerical data, training/test/validation data, normalization	10	10 Feb
Feature selection	11	13 Feb
Dimensionality reduction: PCA, LDA	12	15 Feb
kernel PCA, graph embedding	13	17 Feb
Model evaluation and hyper parameter tuning: cross-validation, bias/variance, overfitting	14	20 Feb
Grid search, Bayesian optimization	15	22 Feb
Performance evaluation **Ensemble learning:** majority voting, bagging	16	24 Feb
Adaboost-ing, Gradient Boosting	17	27 Feb
Regression: linear regression	18	1 Mar
Robust regression	19	3 Mar
regression with non-linear functions	20	13 Mar
Clustering Algorithms: k-means,	21	15 Mar
Selecting number of clusters	22	17 Mar
Hierarchical trees	23	20 Mar
Spectral clustering	24	22 Mar
Artificial Neural Networks: training, backpropagation	25	24 Mar
Tensor flow	26	27 Mar
Convolutional Neural Networks (CNNs)	27	29 Mar
CNNs and regularization	28	31 Mar
Recurrent Neural Networks (RNNs)	29	3 Apr
RNNs	30	5 Apr
RNNs and the Transformer model	31	12 Apr
Generative networks: GANs, autoencoders	32	14 Apr
Generator and Discriminator network	33	18 Apr
Diffusions	34	19 Apr
Reinforcement learning: theory	35	21 Apr
RL algorithms	36	24 Apr
dynamic programming,	37	26 Apr
monte carlo, temporal different learning	38	28 Apr
q-learning	39	1 May
Deep q-learning	40	3 May

Grading

- 50% project and related

- You will work in preassigned groups of 4 or 5 to teach a machine how to solve a problem via examples
- 25% one-on-one oral quiz on project knowledge and general ML knowledge (done on the 10th of May)
- 25% project report, code, and pre-recorded video presentation (due on the 5th of May)
  - The report will have to clearly state what was the specific contribution of each team member. Generically writing: "we all did the same amount of work" will make you lose points
  - The report should clearly state what came from your brain and what came from other people's brain (e.g. stuff you found on the internet)
  - All will be done using **Google colab** (https://colab.research.google.com) (get used to it asap)
  - There will be 4 milestones on the project status (dates TBD)
    - Milestone completion does not count towards your grade
    - At each mile stone I want to know who has done what
    - First stage = define problem + mock good/bad behavior + list of existing similar problems (citations)
    - Second stage = detail list of data sources + methods + project pipeline (including validation) (more citations)
    - Third stage = 1st prototype working + list of success + list of failures (lots of numerical results) + ideas to improve
    - Fourth stage = 2nd prototype working + list of success + list of failures (lots of numerical results) + ideas to improve

- 20% attendance and participation
  - Attending and not sharing/asking will not get you 20% on this component
  - During class you will help me code and test some of the concepts explained. At these moments I will call some student to use their laptop and program a bit. I will try to mostly use Python.

- 30% class completion task
  - Often time we will not have enough time to test all ideas and complete in class all examples that we started. These completions will be assigned for you to finish at home. These will be <u>unstructured</u> assignments. All I will be looking for is "Did the student play enough

with the concepts or not? “Did the student make progress to complete the task asked?”.

- All will be done using Google colab (https://colab.research.google.com) (get used to it asap)
- Completion of associated CANVAS assignment will amount to uploading a link to your colab on CANVAS.

Integrity

Do not copy your solutions to questions asked in class from online sources, your colleagues, github co-pilot, chatGPT, whatever. This is because if you get used to doing this, when the time comes for your oral quiz, you will do really bad, and have a really bad experience.

In your project you can use whatever sources you want, Github, online papers, code generators, other people, whatever, as long these sources are **cited**. If you do not cite your sources, I will assume that you are claiming that the ideas/sources are yours, and if turns out they are not, you will get into serious trouble.

Please read the following:
https://www.bc.edu/offices/stserv/academic/integrity.html

In short, the class will be lots of fun, if you just pay attention to some basic, common sense details. Cheers and welcome!