CS 475/675 Machine Learning: Project

This semester you will complete an independent project to learn how machine learning can be applied in practice. You will work in groups of 3-4 students, and all students in the group will receive the same grade. The final project will be worth 20% of your final grade (200 points). The group policy is the same as the collaboration policy in homeworks, except that you need 3-4 students in a group.

You have a choice between two types of projects.

- **Application**: These projects will apply machine learning methods to a dataset to achieve the desired goals in an application domain. Here the desired goals may be a good predictive performance, but may also include other properties like robustness, fairness, safety, in addition to the predictive performance.
- **Methods**: These projects will conduct an analysis on certain properties of machine learning algorithms and develop novel techniques to improve the properties of state-of-the-art methods.

Additionally, each project will select a specific area: either an application domain or a subfield of ML methods. Each team will be assigned to a TA/instructor who will provide feedback and review project deliverables.

Project Structures

Your first goal will be to form a team of 3-4 students. You will share your grade with your team members, so pick people with whom you can work well.

The final project has several milestones and deliverables.
1) Project proposal (20 points): Due **Tuesday 11/1 (11:59pm)**
Your project proposal should be about 1-2 pages. Use the provided latex template here.

2) Project Update (10 points): Due **Wednesday 11/16 (11:59pm)**
Your project update should be about .5 pages. Copy your proposed list of deliverables and write a 1-2 sentence update on your progress for each deliverable. We expect that at this point you've at least processed and explored your data. Hopefully, you will have completed at least one of your "must accomplish" deliverables.

3) Final presentation (60 points): **Friday 12/9** (in recitation)
Each team will present a ~10 minute presentation on their project during recitation time. Include an update on your deliverables: are you on track to accomplish everything you proposed?

4) Project report (Project git repo and Jupyter Notebook) (110 points): **Monday 12/15 (11:59pm)**
The final deliverable for your project will be a git repository containing all of your code and a Jupyter Notebook containing the final writeup. The git repo can be stored on Github.com or Bitbucket.com, and you will provide a link to the repository. You will also provide a link to a

Jupyter notebook. The notebook can be stored directly in the git repo, or hosted on Google Colab. The notebook should be structured as a writeup, with text explanations mixed with a step by step walkthrough of the project, including summaries of the methods, goals, figures, and results. We will provide a Notebook template you should follow.

Application Project

For the application project, we provide five application domains for your reference. But you are strongly encouraged to expand beyond the provided dataset. Your goal will be to propose an application that requires machine learning using the dataset. The novelty lies in the pipeline that you develop, and you will be focusing on the various aspects of developing a complete machine learning solution to automate a real-life need. This will include, formulating a task, selecting data, feature engineering, determining the best features for your model, implementing a model, determining hyperparameters and testing your model for problematic behavior. You can use any software library for this task. The way you process your data, and the choice of models will depend on the domain that you're working on. You will be judged based on the work you put into the project beyond the available software libraries.

Each group should select a dataset from one of the following domains.

1. Medical informatics and Genetics
    - 10x Genomics https://www.10xgenomics.com/resources/datasets
    - UCSC Genome Browser https://genome.ucsc.edu/
    - Gencode https://www.gencodegenes.org/
    - Protein Classification http://scop.mrc-lmb.cam.ac.uk
2. Healthcare delivery and public health
    - COVID cases and deaths: https://ourworldindata.org/covid-deaths
    - WHO Statistics: https://apps.who.int/gho/data/node.resources
    - CDC Statistics: https://data.cdc.gov/browse
3. Textual analysis
    - Wikipedia articles: https://en.wikipedia.org/wiki/Wikipedia:Database_download
    - Sentiment prediction: http://archive.ics.uci.edu/ml/datasets/Sentiment+Labelled+Sentences
    - 20newsgroups: https://scikit-learn.org/0.19/datasets/twenty_newsgroups.html
    - Translation of Canadian Parliament: https://www.isi.edu/natural-language/download/hansard/
    - Hate speech on Twitter https://data.world/thomasrdavidson/hate-speech-and-offensive-language
4. Image analysis
    - Facial recognition: http://vision.ucsd.edu/content/yale-face-database
    - Art objects (paintings, sculptures, drawings etc) from the Rijksmuseum museum in Amsterdam: https://figshare.com/articles/Rijksmuseum_Challenge_2014/5660617

    - Satellite Imagery for Natural Disasters: https://www.digitalglobe.com/ecosystem/open-data
5. Finance and economics
    - World Bank Open Data: https://data.worldbank.org/
    - International Monetary Fund Data: https://www.imf.org/en/Data
    - Quandl: https://www.quandl.com/ (Make sure to select free data)
    - Stock prices from pandas_datareader

Methods Projects

We have learned about several algorithms and the assumptions behind them. We also commented that these observations are not valid in real data. For example, the assumption of independent and identically distributed observations does not hold for time series data. Further, it is possible that there are behaviors other than high prediction accuracy that a real world application might need. For example, you want your classifier to be fair to individuals who belong to a minority group. Real data also requires coming up with smart ways to deal with noise, missingness and unknown confounders. Additionally, you might care about other model requirements like interpretability: why did the model predict a certain outcome?

We suggest the following topics for the methods projects. But you are strongly encouraged to define your own desired properties of the ML algorithms, based on what you have learned in the human-centered ML. Your task will be to evaluate a method and propose/modify a method, such as creating a new loss function, constraints, data preprocessing, etc. with the goal of solving one of the tasks. You will demonstrate that your model improved on the desired task, and can be used to achieve your stated goal, on selected datasets, either simulated or real data depending on your project needs. You **must** evaluate your method on one of the simple datasets contained within sklearn.datasets, if applicable, but you are encouraged to also use a real-world dataset to explore a more interesting application. The exact evaluation will depend on your proposed method. You do not need to implement the methods from scratch; you will be allowed to use existing software libraries. However, we will judge you based on the amount you do beyond existing implementations. Additionally, you should compare your method to an existing method, such as a classifier from Scikit Learn.

6. Fairness in Machine Learning
    - A project that demonstrates fairness aspects of a model or dataset, and proposes methods to improve fairness in a measurable way
7. Interpretable Machine Learning
    - Methods for interpreting trained ML models
8. Graphical models / Structured prediction
    - Develop a new graphical model, or develop a model for a structured prediction task
9. Robust Machine Learning

    - Learning in the presence of outliers or adversarial examples. For example, see https://jerryzli.github.io/robust-ml-fall19.html
10. Private machine learning
    - Develop an algorithm that respects the privacy of users whose data appears in the training set. For example, see https://medium.com/dropoutlabs/privacy-preserving-machine-learning-2019-a-year-in-review-123733e61705

Project Examples (You are strongly recommended to explore your own ideas)

1. Music AI
   Take a dataset of songs, yours or e.g. https://www.aicrowd.com/challenges/spotify-million-playlist-dataset-challenge, read it into pytorch. Train a classifier that selects the "best" next song, given the previous song (or sequence of previous songs if you want to impress). Such a classifier could for example maximize the similarity between the frequency and "quefrency" of the next song with the beginning of the last song (using fourier and inverse fourier transforms on the logarithms of the waveforms). You could also take into account parameters describing the hooks, scale analysis of the song's scale, bpm, etc. If you know music theory feel free to improvise with respect to feature construction. It's up to you to characterize the specifics of the classifier, e.g. how you select the set of songs from which you pick the best next. In the simplest case you have a closed set that is the same size as the number of your recommender's classes. But you may want to randomize the songs that you're choosing from at each suggestion-step, to make the suggestions more interesting by using a huge overall set of songs on which you could not possibly efficiently compute softmax. You could also predict the similarity for playlists (like the spotify recommender system), or match a song to a database (as in shazam), or analyze instruments present given a waveform, e.g. is there guitar, voice, drums in the song?

2. Computational genomics and bioinformatics - ***General warning:*** for tasks related to human genetics, do not expect to obtain model performance that is comparable to other tasks such as image recognition, as human biology is extremely complicated, and biological datasets are often especially noisy. You are very welcome to pursue human genomics projects, just keep this in mind. For simpler model organisms, such as bacteria, there is stronger signal and cleaner data available, and you can identify simpler tasks which should lead to much higher performance.

   a. *Ab initio* gene prediction/ genome annotation - train a machine learning model to annotate a genome - identify gene vs. non-gene region, or go further and try to identify promoters, enhancers, and other elements of the genome (increasingly complex task from bacteria to humans, you may choose any organism you'd like as long as your project goes beyond re-implementing something you found online). Hidden Markov

Models are common, but you are welcome to consider alternative models including deep learning.

i. Sequence data is available from UCSC Genome browser, annotations for human and mouse are available from Gencode
ii. Example - http://ccb.jhu.edu/software/glimmerhmm/

b. Transcription factor motif identification - transcription factors are important proteins that bind specific DNA sequences (known as transcription factor binding sites, or motifs) and induce or repress expression of many genes. Many of the genetic mutations that are responsible for disease act by altering transcription factor binding. You can train a machine learning model to predict binding of a particular transcription factor from DNA sequence. You could even go further and check whether your model is able to predict the influence of a mutation on transcription factor binding, which could influence a person's health (you would need to find additional data for this). You could also modify an existing method designed for this task.
   i. You can find transcription factor binding data at https://www.encodeproject.org/chip-seq/transcription_factor/#outputs
   ii. A database of known transcription factor binding sites can be found at https://jaspar.genereg.net/
   iii. One approach uses expectation maximization to identify motifs in sequences that are bound by a transcription factor https://meme-suite.org/meme/doc/meme.html?man_type=web. An SVM-based approach to this which accounts for gaps in the sequence is described at https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1003711. A deep learning-based approach is described at https://www.nature.com/articles/s41588-021-00782-6

3. Neural Machine Translation (NMT)

Train a transformer-based machine translation model. You can use existing NMT toolkits: Sockeye (https://github.com/kevinduh/sockeye-recipes3), fairseq (https://fairseq.readthedocs.io/en/latest/index.html), etc. or implement from scratch using PyTorch. You can choose one of the language pairs from the bitext dataset built based on Ted Talk subtitles: https://www.cs.jhu.edu/~kevinduh/a/multitarget-tedtalks/. Tune different hyperparameters, e.g. initial learning rate, embedding size, number of attention heads, compare the performance on the validation set to select the best model and report the result (BLEU score) on the test set using the model trained with the best hyperparameters. You can then choose one of the following topics for further exploration.

a. Language models pre-trained on large corpus are beneficial to downstream natural language processing tasks. Explore how pre-trained language models, e.g. BERT, mBART, MT5 (https://huggingface.co/models), can be applied to machine translation. Finetune your models on the Ted Talk dataset (https://www.cs.jhu.edu/~kevinduh/a/multitarget-tedtalks/). Compare the performance on the Ted Talk test set of NMT models trained with and without Ted Talk data.

b. Domain mismatch is one of the challenges when deploying NMT systems in practice. NMT systems trained on dataset within one domain (e.g. news) probably won't perform well on another domain (e.g. Twitter). Explore domain adaptation techniques in NMT and try them out. You can start with this survey paper: https://arxiv.org/pdf/1806.00258.pdf
c. To achieve good performance, NMT systems usually need millions of training examples. However, there are many low-resource language pairs and domains which have limited available training data. Explore techniques (e.g. data augmentation, back translation) designed for improving the performance of NMT systems on low-resource tasks and try them out. You can start with this survey paper: https://arxiv.org/pdf/2109.00486.pdf

4. Adversarial attack and defense
Implement a deep learning based classification, one adversarial example attack and also one defense mechanism for the attack.
Supporting materials need to read before project:

- MNIST data classify model (you may start with MNIST)
  https://github.com/pytorch/examples/tree/main/mnist
- Attack mechanism:
  Fast Gradient Sign Method (FGSM): a method to add the noise (not random noise) whose direction is the same as the gradient of the cost function with respect to the data.
  https://arxiv.org/abs/1412.6572
  https://pyimagesearch.com/2021/03/01/adversarial-attacks-with-fgsm-fast-gradient-sign-method/
- Defense mechanism:
  Defensive Distillation: https://arxiv.org/abs/1511.04508

Recommend workflow (using MNIST as an example):

- Step 1: Implement the pytorch example model for MNIST
  - Plot the training and validation loss
- Step 2: Applied the attack method on the model
  - Plot accuracy when set noise magnitude Epsilon to 0, 0.01, 0.02, 0.05, 0.1 and 0.2 (Epsilon is a parameter you will learn from the supporting materials)
  - Visualize sample adversarial examples for different Epsilon
- Step 3: Applied the defense method on the model
  - Plot accuracy when set Epsilon to 0, 0.01, 0.02, 0.05, 0.1 and 0.2
  - Visualize sample adversarial examples for different Epsilon

5. Domain adaptation/generalization using one or multiple WILDS data

WILDS is a benchmark of in-the-wild distribution shifts spanning diverse data modalities and applications, from tumor identification to wildlife monitoring to poverty mapping.

Data link:https://wilds.stanford.edu/
Paper link: https://arxiv.org/abs/2012.07421

Pick one or multiple datasets from the WILDS, implement one of the domain adaptation or generalization methods (for example, one of the methods listed here https://github.com/facebookresearch/DomainBed). Train the model in the (multiple) source distributions and test in the target distribution. Analyze the performance in different domains. Besides accuracy, evaluate the uncertainty estimates as well. Do they work well in your chosen data or domain? How can you improve them?

6. ML fairness

Using resources in https://aif360.mybluemix.net/, understand and implement existing methods in ML fairness and evaluate them in one of the benchmark datasets, critique their pros and cons. What are the potential shortcomings of the existing methods? Try to define a meaningful fairness metric in one of the domains you are interested in. Using existing approaches or proposing a new approach to achieve the desired property.