Advanced Machine Learning (GR5242)
Fall 2018

Final Projects

Due: Monday 10 December at 4pm (for both sections of the class)

Project Submission: Complete one of the following projects. Please submit your completed project by publishing a notebook that cleanly displays your code, results and plots to pdf or html. You should also include a pdf file containing typeset or neatly scanned description of your project goals and results.

Project 1 (CIFAR10)

The CIFAR10 dataset is an important image classification dataset along with those that we have seen in class (MNIST, SVHN, ImageNet,...). You can learn about and download this dataset `[at this link]`.

It is a challenging classification problem with ten classes (airplanes, automobiles, birds, cats, deer, dogs, frogs, horses, ships, and trucks), but also one that is manageable in size such that you should be able to get solid performance without a large scale GPU implementation.

The project is to implement the best deep learning classifier that you can create on this data. There are many algorithms that have done well on CIFAR10; you can read about them `[at this link]`. You should implement a number of algorithms and responsibly separate your data into training, validation, and testing sets.

Notes:

- This project is about a sound implementation, understanding your implementation, explaining it well, and explaining how it fits into the broader literature. Performance is relevant, but it is not the sole objective here.
- This project does not require a GPU, though of course it can be used if interested.
- There are many implementations of CNNs to classify CIFAR10. You must create and implement your own architecture. If you source any key ideas from elsewhere (such as how to preprocess data frames), you must reference the source of those ideas.
- This project is somewhat technically and conceptually simpler than others. We thus expect you to demonstrate in depth understanding and mastery of the concepts at hand.
- Hint: pay attention to data preprocessing. Read the literature (see above links) to learn how others have done this, and be careful and empirical. Reference preprocessing ideas that you get from other sources.

Project 2 (Variational autoencoders)

A natural Bayesian representation of a large variety of real-world representations is that of a latent variable model, whereby the data we observe (e.g. the pixels of a photo) depend on some unknown latent feature (e.g. the type of object captured by a photo).

However, these latent models can be hard to describe (e.g. for a hand-written digits, one might not only consider the digit, but also the stroke, the angle, etc.), and even if we were able to do so, inference over large latent variable models can be challenging, as we have seen in the first part of the course.

Variational auto-encoders (VAE; `https://arxiv.org/pdf/1312.6114.pdf`) attempt to solve both problems simultaneously by approximating the distribution of the data using a variational family. However, as we have seen, most variational families do not approximate arbitrary distributions well, especially if we place additional assumptions (such as mean-field) to make them tractable. Variational auto-encoders parameterize the variational

family by a deep neural network, which (hopefully) ensures that the capacity of such a family is large enough to approximate the true distribution well.

In this project, you are asked to explore variational auto-encoders. Implement a variational auto-encoder to learn the SVHN dataset that we studied in class, and display images learned from and generated from the distribution. Explore how the dimension of the latent space affects the learned representation, and visualize the learned manifold for low dimensional latent representations.

Notes:

- This project does not require a GPU, though of course it can be used if interested.
- Implementations of VAE for digits are available on the internet. You must create and implement your own architecture. If you source any key ideas from elsewhere, you must reference the source of those ideas.
- You should familiarize yourself with the literature to understand how others visualize learned manifolds, and how the VAE is otherwise investigated and explored. You should implement some of those ideas to help ask and answer questions about the VAE (e.g. how does manipulating the representation in latent space change the decoded image?).
- This project combines ideas from variational inference with deep learning; it is an excellent project that combines the first and second half of the course, but if you found yourself lost with the ideas of variational inference earlier, this project will present a challenge.

Project 3 (Feature visualization)

Throughout the course we have to some extent treated neural networks as black box function approximators, not interrogating what particular features various layers represent. *Feature visualization* attempts to understand what feature maps neural networks use. Suppose you train a neural network to solve a particular classification problem (SVHN or MNIST, for example). You might then take this network and ask, given these parameters, what input image would drive a particular unit's activation the most? This is a sensible optimization problem: fix the learned parameters and treat the input image itself as an optimization variable, and then optimize for the image that maximally excites a particular unit. When you do so, interesting hierarchies and feature representations emerge.

In this project you should read some literature on feature visualization and implement from scratch a feature visualization algorithm of your own. For example, you might start by training a two layer CNN on MNIST, as we did in class. You might then optimize to find images that excited different layers of the network. You can then repeat this with SVHN or a more elaborate dataset. Recent review publications give a nice overview that can provide a starting point for your project: `https://distill.pub/2017/feature-visualization/` and `https://distill.pub/2018/building-blocks/`.

Notes:

- This project can be done without a GPU, and an advanced version of this project would involve a GPU.
- There are numerous examples of feature visualization code on the internet, though you are expected to do your own coding work for this project. If you source any key ideas from elsewhere (such as how to optimize), you must reference the source of those ideas.

Project 4 (Deep reinforcement learning with pixel features)

In this project, you will implement a deep reinforcement learner that learns directly from screen images as input to the network. Use the openai gym pong environment (`https://gym.openai.com/envs/Pong-v0/`, or a different one if you prefer). You will need to combine your knowledge of convolutional neural networks to extract features from the screen images and deep reinforcement learning to learn a good policy.

Notes:

- This project requires access to and comfort working with a GPU.
- This project is challenging, as it requires both RL and CNN. You should expect long training times (on the order of days).
- There are numerous examples of code doing similar things on the internet, though you are expected to do your own work for this project. If you source any key ideas from elsewhere (such as how to preprocess data frames), you must reference the source of those ideas.

Project 5 (Region convolutional neural network)

In the course, we have mostly explored convolutional neural networks for the problem of image classification. However, for many computer vision problems, classification may only be a small part of the problem. Another common task is that of detection, i.e. identifying a region of the image containing an element of interest (for example finding an object within an image).

Earlier works in this direction were usually done using hand-engineered features. However, the design of these features is difficult, and it is difficult to obtain very good performance from them.

Region convolutional neural networks (R-CNN; `https://arxiv.org/pdf/1506.01497.pdf`) provide an efficient way of leveraging neural networks and large classification datasets to design high performance object detection algorithms. They combine a proposal method (sometimes neural network based) which attempts to identify potential regions of interest and a CNN-based classifier to determine whether the proposed region indeed contains an image of interest.

In this project, we would like you to explore the performance of R-CNN with modern network architectures. Implement a R-CNN with proposal network backed by a pre-trained Inception classifier (trained on ImageNet; exactly as you did in homework 4), and implement the R-CNN on the MS COCO (`http://cocodataset.org/`) dataset or the Pascal VOC (`http://host.robots.ox.ac.uk/pascal/VOC/`) dataset.

Notes:

- This project will require a GPU.
- This project requires understanding the R-CNN paper in depth `https://arxiv.org/pdf/1506.01497.pdf`, which involves some advanced ideas not presented in the class.
- We are not aware of a tensorflow implementation of this algorithm, but any code design ideas you get from any other code (even in another language) must be referenced as source material.

Project 6 (Choose your own adventure)

You are also given the flexibility to design your own deep learning project, drawing on any material covered in the course (and perhaps some topics not covered).

Notes:

- This project should be understood to be harder than the other choices listed here; not only must you solve the problem, but you must identify the problem as well.
- More effort to be put into this type of project than others, and we will expect more from them when grading.
- We are interested in solutions to real problems; for example, if you have a particularly exciting dataset from your research or a job or similar, that would be relevant.
- If work with a custom dataset, we expect you to show the iterative process through which you designed you model and how you evaluated its performance at each step. Only reporting the final model and results will not suffice.

- You are strongly encouraged to contact the TAs to refine the scope of your own project. However, before contacting the TAs you must flesh out your idea for the project and acquire all relevant materials (i.e. they will not be able to allocate resources to helping you find a dataset or identify relevant literature).