note 0 views

Project Ideas

Dear All,

I have listed below some project ideas. Broadly speaking, the projects can be from one of the following types:

(1) Applications. You would try out some existing ML algorithm on some interesting task/application.

(2) Algorithmic improvements: You would pick an existing ML algorithm and propose improvements that result in better accuracies and/or faster training/test time, etc. Depending on how novel/significant your proposed improvement is, this can even be considered as a new ML algorithm.

(3) Theoretical project. You study and theoretically analyze an ML algorithm or a class of ML algorithms in terms of their strengths, limitations, etc. This is expected to be a fairly challenging project in general. The list below doesn't contain such projects but if you are interested in such a project, please let me know and I will give you some suggestions.

(4) Empirical evaluation/comparison: You would implement (or pick publicly available implementations) some ML algorithm(s) and do a rigorous empirical evaluation/comparison on some benchmark datasets.

List below are some broad topics. For each topic, there may be various possible projects.

(1) Multi-label Learning

Recall from our discussion in the class that, in multi-label learning, the output is a binary vector (denoting the presence/absence of each label). This is akin to a "tagging" problem where we wish to predict the most relevant set of tags from a set of tags. There are some interesting possibilities for doing a project on this problem:

- Look at large-scale a.k.a. "extreme" multi-label learning (XML) where the number of training examples, number of features, and number of labels could be very large. We want fast training, fast prediction at test time, and of course good prediction accuracies. KNN is a simple approach can be multi-label learning but it is slow at test time. Can you improve KNN for this problem (speed/accuracy/both)? There are several other simple models too, like one-vs-all classification that work well for XML but may be improved upon with some thought. You can look at these too. There is a nice repository that contains lots of datasets, implementations of existing algorithms, etc: http://manikvarma.org/downloads/XC/XMLRepository.html

- Semi-supervised multi-label learning: Most multi-label learning algorithms work with lots of labeled data but what if you have little amount of labeled data and possibly lots of unlabeled data? This is the semi-supervised learning setting. Can you come up with a good semi-supervised multi-label learning algorithm?

- Implement and evaluate this recent paper on extreme multi-label learning: http://proceedings.mlr.press/v80/siblini18a/siblini18a.pdf This paper is very easy to understand. Uses basic ML concepts like clustering, decision-trees. For most of the components in the paper, students should be able to find a Python library, but it should be a nice exercise for someone at an introductory level to get all the components working. Students should be able to get results for the small and medium sized datasets from the extreme multi-label classification repository.

- Think of some nice application of multi-label learning and try out some of the existing models.

(2) Multi-class learning

- Typical multi-class algorithms require training/test time that is linear in the number of classes (naively done, you need to compute the score for each class). How to we get sublinear training/test speed for multi-class learning? There is a fair bit of work on this problem, and you could (1) explore existing approaches for this problem, and (2) come up with improvements to the existing algorithms, or new algorithms for this problem. You may look at this (somewhat old now) paper and its related work section to get a good sense of prior work on this problem: https://papers.nips.cc/paper/4027-label-embedding-trees-for-large-multi-class-tasks.pdf

(3) Zero-shot Learning

How do we recognize objects from classes not present at training time? We actually looked at this problem in HW1 (programming problem) and two very simple methods to solve this problem. Can you come up with improvements to these methods, or come up with new methods for ZSL? Implementing and testing some other existing algorithms (from some recent paper) can also be a good project. Both multi-class and multi-label learning problems can be posed in this setting.

(4) One-shot/Few-shot Learning

In this setting, we have very small amount of training data per class and the goal is to learn a good classification model using this data. Both multi-class and multi-label learning problems can be posed in this setting.

(5) Learning with graph-structured data

Many datasets are naturally in form of a graph. For example, a social network, or a citation network of researchers, etc. Given such a graph, can you learn an embedding (i.e., a feature representation) of each node? This can be useful for tasks such as node classification, link prediction, etc. Here is a recent survey on this topic:
https://arxiv.org/abs/1709.05584

(6) ML for Tiny Devices

Often we wish to deploy ML models on devices that are resource-constrained (memory, computation, or both). How can we develop learning algorithms (e.g., nearest neighbors or DT or other models) that can be somehow "compressed" to work well in such settings? This can be very useful for IoT devices. Here are some recent papers on this topic:

http://proceedings.mlr.press/v70/kumar17a.html
http://proceedings.mlr.press/v70/gupta17a.html

(7) Explainable Machine Learning

Can ML algorithms explain why they predicted a certain output? This can be very useful for the goal of having "explainable" or "accountable" ML. Here is a nice recent paper on this topic where they show ANY machine learning model can be made interpretable: http://www.kdd.org/kdd2016/papers/files/rfp0573-ribeiroA.pdf You can look at this problem in this detail, understand their methodology, play with it, and think about ways to improve. This is a fairly open research problem so coming up with a nice way to yield interpretable models will be very useful.

(8) One-Class Learning and Learning from Positive and Unlabeled (PU) Data

In many ML problems, we only have positive examples (and someone also unlabeled examples) but no negative examples. For example, in building a recommender system, we collect data on users' clicking history. Here a click is a positive example but a "not clicked" is not necessarily a negative example because it could just be due to the user not being exposed to that ad. How can we build models that can learn from such data. One (and not the only) way to solve this problem is to model it as a one-class learning problem or learning from PU data problem. In this project, you can explore existing one-class or PU learning methods to solve this problem, try them on some real-world datasets, and additionally may also proposed new methods. This can in fact also be used to solve multi-label learning (which will be a very nice application of such a model).

(9) Active Learning

Instead of traditional supervised ML, where *we* provide labeled data to the ML algorithm, can the algorithm *ask* for those labeled examples that it considered the most useful for learning? Basically, if the learning algorithm can identify which inputs are the hardest, it can specifically ask the annotator to provide labels for those inputs. This paradigm is known as "active learning" and here is a somewhat old (but still relevant) survey on the topic: http://burrsettles.com/pub/settles.activelearning.pdf These days, there is a renewed interest in this problem since the commonly used deep learning algorithms require lots of labeled examples, and people are interested in reducing the annotation cost. Some recent papers include

https://openreview.net/pdf?id=H1aIuk-RW and https://arxiv.org/pdf/1703.02910.pdf . If you are interested in this problem, I can provide some more pointers. Caution: If you want to do something "novel" then this is expected to be a bit more challenging project (the low-hanging fruits have been picked up); however, if you want to study existing methods, evaluate them, etc, then it should not be too hard

them, etc, then it should not be too hard.

(10) Semi-supervised Learning

Can we include unlabeled data in a supervised learning algorithm to improve the model? The answer is yes, because in general, even unlabeled data can tell us a lot about the structure of the class distributions. You might take a look at these links to get some more idea about the typical existing approaches: http://ruder.io/semi-supervised/ and https://arxiv.org/pdf/1804.09170.pdf (also this somewhat older but relevant survey: http://pages.cs.wisc.edu/~jerryzhu/pub/ssl_survey.pdf) Caution: If you want to do something "novel" then this is expected to be a bit more challenging project (the low-hanging fruits have been picked up); however, if you want to study existing methods, evaluate them, etc, then it should not be too hard.

(11) Multitask Learning

Rather than learning a single classification/regression model from a single dataset, what if we are gives multiple datasets but they are somewhat related? Instead of learning a model from each dataset separately, can we somehow pool information from these multiple learning problems (a.k.a., "tasks") to improve the overall performance? This is known as Multitask Learning (and in some cases, "transfer learning"). Here are some recent surveys on this topic: https://arxiv.org/pdf/1707.08114.pdf and https://arxiv.org/pdf/1706.05098.pdf . If you are interested in this problem, I can provide you more pointers. Caution: If you want to do something "novel" then this is expected to be a bit more challenging project (the low-hanging fruits have been picked up); however, if you want to study existing methods, evaluate them, etc, then it should not be too hard.

(12) Domain Adaptation

A common assumption made in ML algorithms is that the training and test data comes from the same probability distribution. This may not be the case in practice. For example, if you train a sentiment analysis model on movie reviews, will it work well at test time on book or gadget reviews? How can you adapt your models to work well in such setting? You can read more about the problem in these surveys: https://tinyurl.com/da-survey-cs771 (somewhat old but still relevant) and https://arxiv.org/pdf/1802.03601.pdf (this focuses more on domain adaptation for computer vision problems but most of the ideas are generic and apply to other domains as well).

(13) Empirical Evaluation/Comparision of ML Algorithms

Do a comprehensive and thorough evaluation of various ML algorithms on some benchmark datasets. A rigorous evaluation would require careful experimentation (proper tuning of parameters, looking at accuracy vs speed trade-offs, etc). This can actually be a very nice project. In fact, there have been conference papers in the past (from around 10 years go) that did such comprehensive evaluations.

https://www.cs.cornell.edu/~caruana/ctp/ct.papers/caruana.icml06.pdf
http://icml2008.cs.helsinki.fi/papers/632.pdf

I would encourage you to check out the above papers to see how to do such a study. If you can do the same on more contemporary ML algorithms and datasets, that will be a very nice and valuable project.

(14) Applications

14.1: Text representation and classification: How can we get good representation of text data that we can use for other tasks, such as text classification? There is a nice simple algorithm called Fast-text (https://fasttext.cc/ - the link has some related papers too, such as https://arxiv.org/pdf/1607.01759.pdf) that has been shown to work quite well on a wide range of text classification datasets. In this project, you can look at this method in detail, and try it on some text classification datasets (and compare with other methods). In addition, it would be nice to think of ways to improve it further.

14.2: Text "translation": Here the goal is to translate a given piece of text. The "translate" may not necessarily be translating from one language to another language but can be in more general sense, i.e., question to answer, long text to short text (summarization or paraphrasing), etc. Most of the recent successes in this domain are based on "sequence-to-sequence" models (some based on neural network based models, such as LSTM, RNN, etc).

14.3: Recommender Systems: Given the past data about "who purchased what", or "who watched which movie", etc., can you build a recommendation system? There are many challenges, such as data sparsity (e.g., though there are lots of users and items, each user buys very few items), doing it in a scalable manner for millions or users/items, and so on, handling new items/users that were not present in the training data, etc. If you are interested in this problem, I can tell you about some specific problems to look at.

14.4: Visual QA: Given an image, answer a question about the image.

14.5: Image captioning: Note that even this can be seen as a translation problem but you are translating an image into its text caption.

14.6 Scholar's Portal: Given a database containing thesis of students (say IITK PG students), build a system that helps people (1) search and retrieve the relevant theses based on specified keywords or phrases, and (2) automatically groups authors or faculty advisors based on their area of research. We actually have some real data (from IITK library) for this project if you are interested.

(15) Other Projects

You can also browse through the student projects done by students in ML classes at other universities (e.g., http://cs229.stanford.edu/projects.html). If some project sounds particularly interesting to you, you can take that up as well (the expectation would be that you would do better than what that student did :)).

project

Updated Just now by Piyush Rai

followup discussions *for lingering questions and comments*