Mathematical Foundations of Machine Learning – Spring 2023

Summer Project list

General comments

A. Use publicly available datasets such as the UCI machine learning repository (http://archive.ics.uci.edu/ml/index.php), Kaggle (https://www.kaggle.com/datasets), Pytorch (https://pytorch.org/vision/stable/datasets.html)
B. In all experiments try to use 5 fold cross validation (split the dataset into 5 equal parts, where each fifth serves as testing in each iteration. Then present average results over the 5 iterations).
C. For regression problems provide average error and std of error statistics (statistical significance is important!).
D. For classification problems provide accuracy (TP+TN)/(P+N), precision TP/(TP+FP) and recall TP/(TP+FN) statistics.
E. Perform hyper-parameter search and try to explain the logic of the best configuration.
F. "Debug" your results: look at confusion matrices, investigate your false positives and negatives. Try to understand where your models fail and try to fix them.
G. For ML problems - compare your results to the results using standard models from Scikit-Learn, etc.
H. For DL problems - **try to start with small datasets and small architectures and work your way from there**. Some computations such as the SparsityProbe on deep & wide networks are relatively heavy. There needs to be a certain ratio between the size of the training dataset and the dimension of the layers, so as the samples are not too sparse in the high dimensional representation.
I. Weights and Biases: In the following files, please insert your wandb credentials (wandb.login + wandb.init), or your scripts won't run:
    1. MFOML_CourseExamples/VisionSparsityProbeExperiments/train/train.py
    2. MFOML_CourseExamples/NeuralCollapse/Vision/init_loader.py
    3. MFOML_CourseExamples/NeuralCollapse/NLP/train_glue.py
    4. MFOML_CourseExamples/NLPSparsityProbeExperiments/train_glue_without_trainer.py
J. Try to come up with other ideas beyond the basic project description.
K. If you have a new research idea using the tools of this course, feel free to propose it. Please note that any project needs to have a "mathematical foundational" component to it. This means, some form of mathematical analysis of the model being built for the task at hand.

Dates:

L. Project selection & team formation deadline: 10th July 2023.
M. Submission Deadline: Friday 8th September 2023.
N. Presentation day: Sunday 10th September 2023

AWS credentials: To be provided by Ido+Yuval personally. Please keep confidential and do not(!) distribute (or mine bitcoin with :-))

Course Image: CourseImageUpdated

Applied Projects

1. Random Forest models (/home/ubuntu/projects/MFOML_CourseExamples/workshop_examples.ipynb)
    a. Find several interesting tabular datasets in UCI/Kaggle - explain why you picked them. You should find datasets that are known to be difficult in the ML world.
    b. Train using sklearn Random Forest and try different HyperParameter schemes.
    c. Try to characterize the effects of different hyper-parameters.

2. Feature importance via wavelet decomposition of RF - Reproduce the feature importance results of [1] (See Lesson 3 Slides 30-36 + /home/ubuntu/projects/MFOML_CourseExamples/workshop_examples.ipynb, **Wavelet Forest code at:** /home/ubuntu/projects/SparsityProbe/tree_models/random_forest.py)
    a. Provide summary of the wavelet-based method with emphasis on the use of the validation set to determine a threshold for wavelet norms.
    b. Test on regression & classification problems (multi-class problems).
    c. Observe differences (if any) on small/large/noisy datasets. Consider injecting random noise to the labels (additive Gaussian noise for regression, random mislabeling for % of the data in classification).
    d. Compare extensively with standard methods as in [1]

3. Compression with wavelet decomposition of RF - Reproduce and add to the results of [1] (See /home/ubuntu/projects/MFOML_CourseExamples/workshop_examples.ipynb, **Wavelet Forest code at:** /home/ubuntu/projects/SparsityProbe/tree_models/random_forest.py)

    Investigate the RF compression capabilities of wavelets through tradeoff between the number of trees and tree components versus the prediction error.

4. Deep Learning intermediate layer-smoothness plots - the effect of using different nonlinearities (see lesson 8, slide 45, **example at:** home/ubuntu/projects/MFOML_CourseExamples/VisionSparsityProbeExperiments/train/train.py)
    a. Use initially the MNIST-1D and MNIST datasets (then perhaps expand to more)
    b. Choose networks from the following: Conv-L-H, MLP-L-H as described in [10] (implemented for you - In folder /home/ubuntu/projects/MFOML_CourseExamples/VisionSparsityProbeExperiments/environments **see:** mnist_1D_Conv_env.py, mnist_MLP_env.py, mnist_Conv_env.py).
    c. Train networks with different non-linearities (ReLU, GELU, p-ReLU, Sigmoid, heaviside-function etc.) and try to recreate results from [10], using the Sparsity-Probe/Neural Collapse code to compute the smoothness/clustering in the matching intermediate layers.
    d. The question is if there exists correlation between the dynamics of the α smoothness/NCC across layers (computed on the training set) and the testing results (computed on testing data)?

5. Function space analysis of ResNets - Add to the research of [3,10]
    a. Use the CIFAR10 dataset and the ResNet18 (/home/ubuntu/projects/MFOML_CourseExamples/VisionSparsityProbeExperiments/environments/cifar10_resnet_env.py) network.

b. Investigate the performance and perform Besov smoothness/NCC analysis of the intermediate layers with and without the residual connections [4].
c. Create a new environment that is based on ResNet18(as done in (a)), but receives grayscale images. Run the same experiments and analysis on the MNIST dataset.
d. The question is: does there exist correlation between the dynamics of the α smoothness/NCC across layers (computed on the training set) and the testing results (computed on testing data) with/without residual connections?

6. Function space analysis of Transfer learning
   a. Train a network on the CIFAR10 dataset (For example using ResNet18 like in /home/ubuntu/projects/MFOML_CourseExamples/VisionSparsityProbeExperiments/environments/cifar10_resnet_env.py).
   b. Perform Besov smoothness analysis/NCC mismatch computation.
   c. Apply transfer learning on a 'small' set of CIFAR100 using as basis a network from (a). This implies 'freezing' some of the first layers and re-training the last layers or creating and training new last layers. Note that CIFAR10 and CIFAR100 are mutually exclusive.
   d. Perform Besov smoothness/NCC accuracy analysis of the transfer-learning architecture using the full CIFAR100 set.
   e. One can use the smaller grayscale versions of the datasets if needed (you need to convert them to grayscale).

7. SVSL loss - train vanilla vs SVSL vs non-batched SVSL (See code in /home/ubuntu/projects/MFOML_CourseExamples/NeuralCollapse/Vision and Slides 8-14 in lesson 8)
   a. Choose a network from the following: VGG19, Resnet18, ConvNet, FCNet as described in [10].
   b. Train networks on the MNIST + Fashion MNIST datasets using: the Cross Entropy loss, (Batched) SVSL loss, and non-batched SVSL loss, using different parameters for η and γ.
   c. Compare the test scores + intermediate NCC scores achieved for different η and γ, and make claims to why this may be (use the results in [11] as guidance).

8. Understanding the encoder transformer hyperparameters through intermediate NCC accuracy
   a. Choose NLP classification datasets.
   b. One option is to fine tune a pre-trained encoder on these datasets: https://huggingface.co/blog/how-to-train
   c. experiment with changes to inner transformer architectural parameters and effect on the intermediate layer NCC and testing results: token vector representation dimension, positional coding on/off, query/key/value dimensions, normalization (other then $8\sqrt{d_k}$).

9. Numerical solutions to PDEs using DL - Follow [8] to apply a DL solution to a PDE
   a. Use example 3.1.1 of [8] as a base for your experiments.
   b. Compare your results to the analytic solutions or the results of other 'off-the-shelf' solvers.
   c. Try to extend the algorithm to support a parameterized family of initial and/or boundary conditions, where the parameters are input to the neural network.

10. Numerical solutions to ill-posed PDE problems using DL Follow [14] and apply a DL approach to a toy example for inverse problems of the wave equation
    a. Create via simulations a dataset of waves with time [0,500] and different source locations. Use as domain a square with a grid of 128x128.
    b. Train a regression DL network on a dataset of images at time 500 to predict (x,y) source location.
    c. Train a regression DL network on a dataset of images at various times [250,500] to predict source location.

Advanced Projects

11. Understanding the effects of backbone architecture on intermediate NCC accuracy in SSL
    a. Choose a dataset (e.g. CIFAR100, CIFAR10, but not exclusively!)
    b. Choose an SSL algorithm (e.g. VICReg [15]), and check the intermediate layer NCC with respect to different hierarchies. Try to recreate the results of [16].
    c. How does the NCC change with respect to the arch? Check this thoroughly and explain.

12. Understanding the effects of SSL algorithm (VICReg, SimCLR, DINO) on the intermediate NCC
    a. In [16], we showed that SimCLR and VICReg act quite similarly. Does this hold for a wide array of SSL algorithms? Try to incorporate as many different algorithm(also different in theme)
    b. Can you think of a better algorithm? Show us!

13. Understanding SSL Encoder models performance
    a. Follow the instructions in this blog https://huggingface.co/blog/how-to-train to train an encoder model from scratch. Use a dataset in english, which isn't too big.
    b. Apply NCC analysis and/or other clustering metrics in the intermediate layers of classes from labeled data sets.
    c. Vary the parameters (depth, width, number of heads, etc.) and see the effects on the clustering and the performance.

14. Phase Retrieval using DL - Follow [13] and apply a DL approach to solve the phase retrieval problem
    a. Use the MNIST and fashion-MNIST datasets
    b. Construct an initial simple network with shallow layers that are fully connected and then deeper layers based on convolutions.
    c. Try to implement a PR network based on the encoder-decoder approach of [13] with the Haar wavelet transform as the encoder-decoder.

15. Spectral physics aware neural networks - Follow [17] and try to use a neural network architecture inspired by spectral methods
    a. You should test linear and nonlinear equations, but you may restrict your experiments to the unit interval.
    b. Reproduce the architectures of [17] that can take as input samples from the initial condition, a point on the unit interval, a time step and output an approximation to the solution.

References

[1] O. Elisha and S. Dekel, Wavelet decompositions of Random Forests - smoothness analysis, sparse approximation and applications, JMLR 17 (2016).

[2] O. Morgan, O. Elisha and S. Dekel, Wavelet decomposition of Gradient Boosting, preprint.

[3] O. Elisha and S. Dekel, Function space analysis of deep learning representation layers, preprint.

[4] H. Kaiming, Z. Xiangyu, R. Shaoqing and SD Jian, Residual Learning for Image Recognition, proceedings of CVPR 2016.

[5] S. Mallat, Group Invariant Scattering, Comm. Pure and Applied Math 65 (2012), 1331-1398.

[6] J. Bruna and S. Mallat, Invariant Scattering Convolution Networks, IEEE Transactions on Pattern Analysis and Machine Intelligence 35 (2013), 1872 – 1886.

[7] P. Kontschieder, M. Fiterau, A. Criminisi and S. Rota Bul`o, Deep Neural Decision Forests, ICCV 2015.

[8] M. Raissi, P. Perdikaris and G.E.Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, Journal of Computational Physics 378(2019), 686–707.

[9] T. Galanti and L. Galanti, On the Implicit Bias Towards Minimal Depth of Deep Neural Networks, arXiv e-prints, 2022.

[10] I. Ben-Shaul and S. Dekel, Sparsity-Probe: Analysis tool for Deep Learning Models, arXiv e-prints, 2021.

[11] I. Ben-Shaul and S. Dekel, Nearest Class-Center Simplification through Intermediate Layers, PMLR 196 (2022).

[12] D. Mixon, H. Parshall and J. Pi, Neural collapse with unconstrained features, arXiv e-prints, 2020.

[13] S. Dekel and L. Gugel, PR-DAD: Phase retrieval using deep auto-decoders, ICFSP 2022. .

[14] S. Dekel, D. Givoli, A. Kahana and E. Turkel, Obstacle segmentation based on the wave equation and deep learning, Journal of computational physics 413 (2020).

[15] A. Bardes, J. Ponce and Y. LeCun, VICReg: Variance-Invariance-Covariance Regularization for Self-Supervised Learning, arXiv e-prints, 2021. doi:10.48550/arXiv.2105.04906.

[16] I. Ben-Shaul, R. Shwartz-Ziv, T. Galanti, S. Dekel and Y. LeCun, Reverse Engineering Self-Supervised Learning, arXiv e-prints, 2023. doi:10.48550/arXiv.2305.15614.

[17] Y. Zelig and S. Dekel, Numerical methods for PDEs over manifolds using spectral physics informed neural networks, https://arxiv.org/abs/2302.05322.