Mathematical Foundations of Machine Learning – Spring 2022

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)
B. In all experiments use 5 fold cross validation.
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 results with the results of 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, you're welcome to check with us if it is relevant. Please do so ASAP.

Dates:

L. Project selection & team formation deadline: 30th June
M. Submission Deadline: 9th September
N. Presentation day: 11th September

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

Course Image: CourseImageUpdated

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 Hyper Parameter schemes
    c. Try to characterize the effects of different HPs.

2. Feature importance via wavelet decomposition of RF - Reproduce the feature importance results of [1] (See Lesson 4 Slide 31 + /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)
    a. Investigate the RF compression capabilities of wavelets through tradeoff between the number of trees and tree components versus the prediction error.
    b. Write code that encodes efficiently a wavelet compressed RF to a file. Ensure you can decode the representation. This is done through efficient encoding of the pruned trees topology and geometry.

4. Deep Learning intermediate layer-smoothness plots - the effect of using different nonlinearities (see slide 37, **example at:** home/ubuntu/projects/MFOML_CourseExamples/VisionSparsityProbeExperiments/train/train.py)
    a. Use the MNIST-1D and MNIST datasets
    b. Choose networks from the following: Conv-L-H, MLP-L-H as described in [9] (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 code to compute the α-scores in the matching intermediate layers, with the index scores.
    d. Can we say anything about the generalization performance using the intermediate α scores? Be creative - this is not a straightforward question, and many possible conjectures can be put forward.

5. Deep Learning intermediate layer smoothness plots in a fixed Besov space - (Possible research project)
    a. In this project you will work with fixed Besov smoothness $\alpha = 1$, which for $p = 2$ implies $\tau = 2/3$.
    b. Implement the sawtooth function, and take samples from it to create a dataset
    c. Implement realization (using a Neural Network), and measure smoothness in intermediate layers using SparsityProbe code (/home/ubuntu/projects/SparsityProbe/, see example in Slide 37).

   d. Using the same network structure - train the network, how do the networks compare in terms of performance (better/worse)? Can you find some similarities between the “theoretical” network and the learnt one?
   e. Compute intermediate layer smoothness with $\alpha = 1$ for the MNIST dataset using the --use_norms option in the code ($\tau = 2/3$).

6. Function space analysis of ResNets - Add to the research of [3, 9]
   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 analysis of the intermediate layers with and without the residual connections.
   c. Compute both critical index α scores and fixed Besov smoothness at $\alpha = 1$, which for $p = 2$ implies $\tau = 2/3$.
   d. 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.

7. Function space analysis of transfer learning
   a. Train a network on the MNIST 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 Fashion-MNIST 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.
   d. Perform Besov smoothness/NCC mismatch analysis of the transfer-learning architecture using the full Fashion-MNIST set.

8. SVSL loss - train vanilla vs batch SVSL vs epoch SVSL (See code in /home/ubuntu/projects/MFOML_CourseExamples/NeuralCollapse/Vision and Slide 51)
   a. Choose a network from the following: VGG19, Resnet18, ConvNet, FCNet as described in [9].
   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).
   d. Try to enhance the SVSL loss by encouraging larger distances from the centers of the clusters of other classes (intra-class distances).

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 Apply a DL approach to an inverse problem of the wave equation

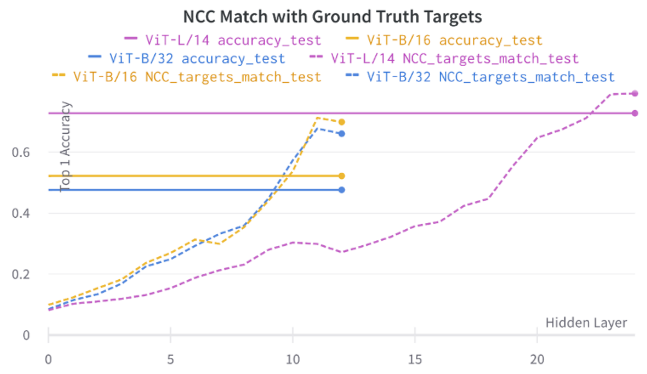
a. Create via simulations a dataset of waves with time [0,500] and different source locations. Use as domain a square with a grid of 128x128.
b. Train a regression DL network on a dataset of images at time 500 to predict source location.
c. Train a regression DL network on a dataset of images at various times [250,500] to predict source location.

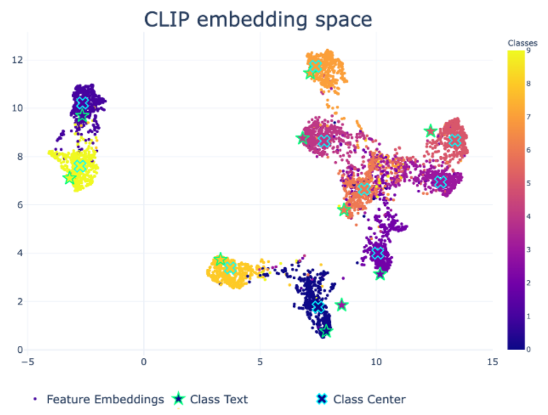
Advanced Projects

11. 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 a first network with initial layers that are fully connected and then convolutions.
    c. Try to implement a second network based on the encoder-decoder approach of [13] with the Haar wavelet transform as the encoder-decoder.

12. Neural Collapse in noisy setting (See code in /home/ubuntu/projects/MFOML_CourseExamples/NeuralCollapse/Vision and Slide 51) It has been suggested that Neural Collapse is possible even in very noisy settings (complete random labels [12]). It is of interest to see the behavior of Neural Collapse under various levels of noise (see [9] for guidance)
    a. Train different networks (without residual connections) on MNIST/F-MNIST with and without SVSL loss, on different amounts of noise levels (as done in [9] + loss from [10])
    b. Does the representation always collapse? Does the SVSL loss always help? What happens in the noisy setting?

13. Neural Collapse Analysis for Self-Supervised Self Supervised Learning (See code in /home/ubuntu/projects/MFOML_CourseExamples/NeuralCollapse/Vision and Slide 42, 51) Self Supervised Learning (SSL) has been extremely prominent in the last few years. We wish to connect the performance of an SSL model on a downstream task and NC. In this task we will only run inference, and not train any new models.
    a. Find pretrained pytorch SSL models online (3 should be enough) - some examples: BarlowTwins, SimCLR, VICReg, etc. Multimodal models can also work - e.g. CLIP. Some possible packages to use:
        i. https://github.com/TorchSSL/TorchSSL
        ii. https://pytorch-lightning-bolts.readthedocs.io/en/latest/self_supervised_utils.html
        iii. https://vissl.ai/
    b. Modify the Analyzer code in Neural collapse to compute the Ground-Truth-NCC-Mismatch with the downstream task labels, in intermediate layers.
    c. Try to link the performance of the downstream task with the NC performance





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] Galanti, T. and Galanti, L., “On the Implicit Bias Towards Minimal Depth of Deep Neural Networks”, <i>arXiv e-prints</i>, 2022.

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

[11] Ben-Shaul, I. and Dekel, S., “Nearest Class-Center Simplification through Intermediate Layers”, <i>arXiv e-prints</i>, 2022.

[12] Mixon, D. G., Parshall, H., and Pi, J., “Neural collapse with unconstrained features”, <i>arXiv e-prints</i>, 2020.

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

[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).