Course Evaluations

The final course evaluations are extremely valuable. They are used in (i) improvement of the course, (ii) understanding (and improve) the course impact in students' education as voiced by students, (iii) promotion packages, (iv) performance evaluation, (v) College/University national standings.

Please feel out the course evaluations! Evaluations are due **Friday, December 10**.

- https://ufl.bluera.com/ufl/

Final Exam In-Person vs Online

The final exam for EEL3850 is scheduled for **Wednesday, December 15 @ 3:00 pm - 5:30 pm**.

The exam will be offered in-person or online. Please select which modality you prefer to take the exam on Wedesnday December 15:

- https://ufl.instructure.com/courses/435230/quizzes/1053637

Office Hours

For the rest of this week, I will be offering my usual office hours tomorrow 12/09 (10am).

- If you want to meet next week, please send me an email so we can schedule a time.

Lecture 43 - Final Exam Review

- You are allowed 1-page letter-sized front and back of formulas (handwritten or typed)
- You are allowed scientific calculator
- Closed book, closed notes

Please do not include:

- pseudo-code, solved exercises or written definitions in your formula sheet

TOTAL TIME: **2 hours + 15 minutes**

Communications between students or anyone else during the exam is considered cheating. Turn off all Slack notifications and other communications channels!

Final Exam Coverage

The midterm exam will cover all materials from Lecture 19-42. These include:

1. **Non-parametric Learning** **(Lectures 19-25)**
    - Clustering
    - K-Means Clustering
    - Cluster validity metrics
    - Probabilitic classifier
    - k-Nearest Neighbors
    - Weighted k-NN
2. **Discriminative Classification** **(Lectures 25-30)**
    - Discriminative vs Probabilistic Classification
    - Fisher's (Linear) Discriminant Analysis (LDA)
    - The Perceptron Algorithm
    - Logistic Regression
3. **Kernel Machine** **(Lectures 31-33)**
    - Kernel Machines
    - RBF kernel
        - Infitinite-dimenional feature space
    - Kernel trick
    - Lagrange optimization
    - Hard-margin Support Vector Machine (SVM)
    - Slack variables
    - Soft-Margin SVM
4. **Artificial Neural Networks****(Lectures 34-37)**
    - Multi-Layer Perceptron (MLP)
    - Universal Approximation Theorem
    - Activation functions: ReLU, leaky ReLu, sigmoid, tanh, softmax, linear
    - Backpropagation
    - Vanishing gradients
    - Exploding gradients
    - Learning curves
    - Network architecture
    - Output Encoding: integer, one-hot, binary
    - Optimization Techniques with Gradient Descent
        - Accelerated Gradient Descent strategies: momentum term
        - Adaptive Learning Rate: Adam

- Online vs Batch vs Mini-Batch learning
- Stopping Criteria
- Data scaling/normalization
- Network pruning
- Ensemble Learning: boosting, bagging
- Dropout
- Batch normalization

5. **Deep Learning (Lectures 38-39)**

   - Deep Learning (DL)
   - DL vs ML
   - Convolutional Neural Networks (CNNs)
   - Pooling layers, stride
   - Convolutional layers

6. **Dimensionality Reduction and Manifold Learning (Lectures 40-42)**

   - Curse of Dimensionality
   - Principal Component Analysis (PCA)
   - PCA vs LDA
   - Manifold Learning
   - Multi-Dimensional Scaling (MDS)
   - ISOMAP
   - Locally Linear Embedding (LLE)

How to prepare for exam

This is a suggestion only.

1. Review/read Notebooks from lectures 19-42.

2. Create your formula sheet. **Do not include pseudo-code or solutions to examples in homework assignments.** In case of doubt, confirm with me if your formula sheet meets the requirements.

3. Review/redo exercises from Part 1 of HW3, HW4 and HW5.

4. Review/redo exercises from SA4 and SA6.

5. Review discussion boards 3, 4 and 5.

6. Solve practice exam. Time yourself.

Q & A

Final Project - Deliverables

1. Code

Your code submission should include 2 main functions: training function and a test function.

Training Function

- The inputs to this function may vary, but it should at least have input variables such as training data and training labels.
- Include **all** parameters used to train the final model, including learning rate, pre-processing steps, and any other methodology/parameter choices.
- Comment your code. This will make it easier to read and in case I need to change/test any parameter.
- Your training function should save the model
    - Instructions with Keras: https://www.tensorflow.org/guide/keras/save_and_serialize
    - Instruction with PyTorch: https://pytorch.org/tutorials/beginner/saving_loading_models.html
- **Do not forget to push this file to your GitHub repository**

Test Function

- The inputs to this function may vary, at least it should include the test samples and labels (in the same format, i.e., `numpy` arrays).
- Your test function **should not** train the model from scratch -- you will loose points if this happens
- Instead, your test function will load the model you saved during training and evaluate performance in the provided test samples.
- Comment your code. This will make it easier to read and in case I need to change/test any parameter.
- If you wish, you can create a separate test function for the extra credit competition.

READ ME file

- Edit the READ-ME of your GitHub group repository.
- It should include all dependencies (libraries) that your code uses.
- It should indicate which file I should use to test your model on the test data.

- It should include a brief description of how to use your function.
    - Example for test function: step 1: load data - give example; step 2: pass it to test function - give example; step 3: describe what the expected output should look like.

Miscellaneous

- Your repository may contain other files.

2. Report

- **Format:** IEEE transactions (single spaced, double column). Use template available here.
- **Maximum number of pages:** 4
- **Sections:**
    1. **Abstract.** A summary description of the contents of the report and your findings.
    2. **Introduction.** Overview of your experiment/s and a literature review. For the literature review, include any references to any relevant papers for your experiment/s. So, whatever you decide to do, search the ACM and IEEE (or other) literature for relevant papers to read and refer to.
    3. **Implementation.** Describe and outline any specific implementation details for your project. A reader should be able to recreate your implementation and experiments from your project report. If you participate in the extra credit contest, be sure to describe the methodology on how you will identify emotional tones or other sounds that were not in the training data.
    4. **Experiments.** Carefully describe your experiments with the training data set and any data augmentation set you constructed or existing data sets. Include a description for the goal of each experiment and experimental findings. This is the bulk of what you will be graded on - if your experimental design is not sound or your experiments do not make sense, you will lose points.
    5. **Conclusions.** Describe any conclusions or things you learned from the project. Your conclusions must follow from what you did. Do not copy something out of a paper or say something that has no experimental support in the Experiments section.
    6. **References.** Listing of all references in IEEE bibliography format.
- Examples of experiments: testing whether adding more data improves performance, testing different parameter settings, testing different normalization techniques, pre-processing and feature engineering steps, data augmentation schemes, and many more.
    - The experiments section is the *bulk* of your work.
- Your conclusions should be based on the experiments section.

3. Self- & Peer-Evaluations

Do not worry about sending this form until your project is completed and submitted.

Evaluate yourself and your teammates fairly. I expect a full contributing member to obtain an average of 10.

- https://ufl.instructure.com/courses/435230/assignments/4907191
- Due: Saturday, December 11 -- day after project submission

4. Grading

1. **25% Implementation**
2. **25% Accuracy** on *easy* test set. Full points on this component will be obtained if you correctly classify 90% of the blind test data or have a classification accuracy rate greater than the average classification accuracy rate of the class (whichever is lower).
3. **40% project report**
4. **10% data collection** (already completed)

Final Grade:

$$p \cdot (0.25 \times \text{implementation} + 0.25 \times \text{performance} + 0.4 \times \text{report} + 0.1 \times \text{data collection})$$

where p is the average percentage from the self- and peer-evaluation form.

Final Project - Implementation Discussion

1. Test Data Sets

You should expect both the *easy* and *hard* test sets to have the same format as the training data. In particular, I will provide you with:

Easy Test Set: holdout samples collected from the class.

- Composed of natural images for all 10 labels
- You will be given the files: "data_test.npy" and "labels_test.npy".

Hard Test Set: contains samples collected from the class + other samples of same classes + unknown class samples.

- Contains all 10 classes plus an unknown class (label -1).

You can perform any data augmentation you'd like and add more data by either collecting new data or using publicly available data.

2. Implementations

- Before presenting data to a model (specially ANNs), shuffle it to avoid catastrophic forgetting

- Apply the **same** transformations used during training to the test set
    - Your test function should automatically do this.
    - You can save any standardization/normalization values used in training, linear transformation (e.g. with PCA)
- How to recognize a new class?
    - If your output layer is a one-hot encoding of the class labels (9-dimensional output), you can make use of softmax activation function
- Save and push trained model to your repository

Good luck on your exams

Go Gators!