CSE 251A: ML: Learning Algorithms **Winter 2026**

Programming project 1 — Prototype selection for nearest neighbor

One way to speed up nearest neighbor classification is to replace the training set by a carefully chosen subset of "prototypes" – i.e. instead of keeping the entire training set to use for nearest neighbor classification, carefully choose a small but representative subset to search for nearest neighbors in. Because this set is smaller than the training data, search will be faster and thus so will classification. However, if poor prototypes are chosen, the resulting test accuracy may be much worse. Think of a good strategy for choosing prototypes from the training set, bearing in mind that the ultimate goal is good classification performance on test data. What properties make a set of examples effective for nearest neighbor classification? Can you formalize these properties and use them to select protoypes automatically? Assume that 1-NN will be used. Then implement your algorithm, and test it on the MNIST dataset, available at:

`https://www.kaggle.com/datasets/hojjatk/mnist-dataset`

On the due date, upload (to `gradescope`) a **pdf** report containing the elements described below (each labeled clearly) and a **zip** of all code you developed for your experiments. We require that your pdf report be produced using Latex with standard conference style files. We suggest cloning the ICML Overleaf template.[1]

1. *A short, high-level description of your idea for prototype selection.*

   A few sentences should suffice. These should be crystal clear: they should communicate the key idea to the reader.

2. *Concise and unambiguous pseudocode.*

   Once again, clarity and conciseness are of the essence. Your scheme should take as input a labeled training set as well as a number M, and should return a subset of the training set of size M.

3. *Experimental results.*

   A (clearly labeled) table or graph of results showing classification performance on MNIST for a few values of M, including at the very least $M = 10000, 5000, 1000$. In each case, you should compare the performance to that of uniform-random selection (that is, picking M of the training points at random). For any strategy with randomness, you should do several experiments and give error bars – give all relevant details, including the formulas you used for computing confidence intervals.

   The pseudocode and experimental details must contain all information needed to reproduce the results.

4. *Critical evaluation.*

   Is your method a clear improvement over random selection? Is there further scope for improvement? What would you like to try next?

.

[1] `https://www.overleaf.com/latex/templates/icml2025-template/dhxrkcgkvnkt`