TTIC 31020: Intro to ML **Winter 2024**

Homework 1

Due: 6 p.m., January 9th, 2024

Note *You may discuss these problems in groups. However, you must write up your own solutions and mention the names of the people in your group. Also, please do mention any books, papers or other sources you refer to. We recommend that you typeset your solutions in LaTeX. Please submit your solutions as a PDF document on Canvas.*

Challenge and Optional Questions *Challenge questions are marked with* challenge *and Optional questions are marked with* optional *. You can get extra credit for solving* challenge *questions. You are not required to turn in* optional *questions, but we encourage you to solve them for your understanding. Course staff will help with* optional *questions but will prioritize queries on non-*optional *questions.*

Notation In all questions, we will use the following notation: let $\mathcal{X}$ and $\mathcal{Y}$ be the instance space and label space respectively. A predictor is a mapping $h : \mathcal{X} \to \mathcal{Y}$, which outputs a label $h(x)$ for data point x.

1. **A Learning Problem!**

   Think of some task that could be approached using Machine Learning. This could be a task from your research area, a task you are working on or are interested in, or a technological application from daily life. Briefly answer (a couple of sentences for each item is sufficient):

   (a) Describe the task. What should the end system be able to do. What is the metric for the system performing well?

   (b) Can the system for the task be built using expert knowledge and careful programming? What expert knowledge is required?

   (c) What data or examples would you collect for learning?

   (d) What are possible advantages and problems with using machine learning for this task?

2. **Nearest Neighbor Prediction**

   In class we discussed *memorization*. Given a training set $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$, *memorization* refers to using a h such that $h(x_i) = y_i$ for each observed instance x_i. This does not rely on any prior information, inductive bias or structure in the instance space $\mathcal{X}$, but does not define the behaviour on unseen instances. This is very limiting. In particular, in a continuous domain $\mathcal{X}$ we might *never* see the same instance twice and memorization is futile. Even in a large discrete space, e.g. discritizing images to several megapixels with a 24-bit color map, we will likely never observe the exact same image twice, and memorization will not get us very far. One way of extending memorization is by letting each observed label y_i be associated also with instances x that are close to x_i, if not identical to it, and predicting unseen instance x based on the observed instance x_i closest to them. That is, given a distance measure $\rho : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ and a training set S, the nearest neighbour learning rule outputs the predictor close to x_i, if not identical to it, and predicting unseen instance x based on the observed instance x_i closest to them. That is, given a distance measure $\rho : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ and a training set S, the nearest neighbour learning rule $\mathrm{NN}_\rho(S)$ outputs the predictor:

   The Nearest Neighbour Predictor:

   - Find the closest training point: $i = \arg\min_i \rho(x, x_i)$
   - Output the label y_i

To fully define the Nearest Neighbour learning rule we need to specify a distance measure ρ. When instances are given as vectors in $\mathbb{R}^d$, we can, for example, use the Euclidean distance $\rho(x, x') = \|x - x'\|$.

Experimentation See the accompanying Jupyter Notebook for experimentation with the Nearest Neighbour learning rule.

As we see in these experiments, Nearest Neighbor performs well on low dimensional data (2-dimensional in the example). However, we will now see that it suffers from the *curse of dimensionality*, and can be problematic in high dimensions, even in very simple situations.

Limitation of Nearest Neighbor Prediction in High Dimensions Consider $\mathcal{X} = \mathbb{R}^d$ and the hypothesis class $\mathcal{H} = \{h_i(x) = \text{sign}(x[i]) | i = 1, \ldots, d\}$, i.e. the d predictors based on individual coordinates.

(a) To establish that this setting is "easy": Suggest a "good" online learning rule for this hypothesis class and give its mistake bound M, i.e. give a learning rule that, for any sequence of samples realizable by $\mathcal{H}$, makes at most M mistakes. Your rule should be "good" in that the mistake bound should be better than that of the CONSISTENT algorithm discussed in class. (optional What is the minimum number of mistakes any online learning rule would make, and can you suggest a learning rule that achieves this optimal mistake bound?)

(b) Now consider using the Nearest Neighbor learning rule w.r.t. the standard Euclidean distance, where at each step of the online algorithm we predict using the data points seen so far. Show that the number of mistakes can be exponential in d, even if all points in the sequence have binary coordinates: $x_i \in \{\pm 1\}^d$. That is, show a sequence of labeled samples, realizable by $\mathcal{H}$, where the Nearest Neighbor rule makes $2^{\Omega(d)}$ mistakes. Compare this exponential mistake bound with your result in (a). (optional How many mistakes can you force if the points have real coordinates, i.e. $x_i \in \mathbb{R}^d$, even if d is small?)

(c) challenge optional Show a similar lower bound also in the statistical setting. Consider a data distribution $\mathcal{D}$ where $x \in \mathcal{X} = \mathbb{R}^d$ are spherical Gaussian (or if you prefer: uniform on a sphere, or uniform on $\{\pm 1\}^d$), and labels are $y = \text{sign}(x[i])$ for some fixed i, and a Nearest Neighbor predictor h_m based on a sample $S \sim \mathcal{D}^m$ of m i.i.d. samples from $\mathcal{D}$. Show that even with $m = 2^{cd}$ samples, for some constant $0 < c < 1$, $\mathbb{E}_S [L(h_m)] > 0.4$ (in fact, the $L(h_m) > 0.4$ with high probability). That is, $2^{\Omega(d)}$ samples are needed to ensure small error.