ENSE 885AU Deep Learning

Assignment A01

Regression and Perceptron

(Due date: Friday September 24th 2021, 23:59)

Instructions:

Math component

1. On separate pieces of paper or using a math input software, answer the questions for this assignment.
2. Scan or convert your drawings to Adobe PDF format.
3. Login to URCourses (https://urcourses.uregina.ca) and submit your answers to Assignment A01.

Coding component

4. Login to Snoopy (**snoopy.engg.uregina.ca** / **142.3.105.92**).
5. Create your program using any text editor of your choice (e.g. **vi / emacs**)
6. Name your files using the following convention:

Main program file	*"A"+number+username+"Q"+number+".py"* (e.g. **A01jon123Q3.py** if your username is jon123 and you are solving question 3)

7. Run your program and test that it works correctly (e.g. **python A01jon123Q3.py**)
8. Submit all your python source code files (Type **~ense885au/bin/submit A01 A01jon123Q3.py A01jon123Q4.py**)

---

1. (Math) Let D be the distribution over the data points (x, y), and let $\mathcal{H}$ be the hypothesis class, in which one would like to find a function f that has small expected loss $L(f)$ by minimizing the empirical loss $\hat{L}(f)$). A few definitions/terminologies:

- The best function among all (measurable) functions is called Bayes hypothesis:
$$f^* = \arg\inf_f L(f)$$
- The best function in the hypothesis class is denoted as
$$f_{opt} = \arg\inf_{f\in\mathcal{H}} L(f)$$
- The function that minimizes the empirical loss in the hypothesis class is denoted as
$$\hat{f}_{opt} = \arg\inf_{f\in\mathcal{H}} \hat{L}(f)$$
- The function output by the algorithm is denoted as $\hat{\mathrm{f}}$. (It can be different from $\hat{\mathrm{f}}_{\mathrm{opt}}$ since the optimization may not find the best solution.)
- The difference between the loss of f^* and $\mathrm{f}_{\mathrm{opt}}$ is called approximation error:
$$\epsilon_{app} = L(f_{opt}) - L(f^*)$$
which measures the error introduced in building the model/hypothesis class.
- The difference between the loss of $\mathrm{f}_{\mathrm{opt}}$ and $\hat{\mathrm{f}}_{\mathrm{opt}}$ is called estimation error:
$$\epsilon_{est} = L(\hat{f}_{opt}) - L(f_{opt})$$
which measures the error introduced by using finite data to approximate the distribution D.
- The difference between the loss of $\hat{\mathrm{f}}_{\mathrm{opt}}$ and $\hat{\mathrm{f}}$ is called optimization error:
$$\epsilon_{opt} = L(\hat{f}) - L(\hat{f}_{opt})$$
which measures the error introduced in optimization.
- The difference between the loss of f^* and $\hat{\mathrm{f}}$ is called excess risk:
$$\epsilon_{exc} = L(\hat{f}) - L(f^*)$$
which measures the distance from the output of the algorithm to the best solution possible.

(1) Show that $\epsilon_{exc} = \epsilon_{app} + \epsilon_{est} + \epsilon_{opt}$
Comments: This means that to get better performance, one can think of: 1) building a hypothesis class closer to the ground truth; 2) collecting more data; 3) improving the optimization.

(2) Typically, when one has enough data, the empirical loss concentrates around the expected loss: there exists $\epsilon_{con} > 0$, such that for any $f \in \mathcal{H}$, $|\hat{L}(f) - L(f)| \le \epsilon_{con}$. Show that in this case, $\epsilon_{est} \le 2\epsilon_{con}$.
Comments: This means that to get small estimation error, the number of data points should be large enough so that concentration happens. The number of data points needed to get concentration ϵ_{con} is called sample complexity, which is an important topic in learning theory and statistics.

2. (Math) Recall that the logistic regression uses the logistic sigmoid function $\sigma(a) = \frac{1}{1+\exp(a)}$ to model the conditional distribution $p(y|x)$ and then apply maximum likelihood estimation. One can use the probit function (instead of the logistic function):

$$\Phi(a) = \int_{-\infty}^{a} N(\theta|0,1) d\theta$$

where $N(\theta|0,1)$ is the standard normal distribution. Derive the negative conditional log-likelihood loss for probit regression.
Comments: No need to simplify the expression.

3. (Coding) Generate 100 synthetic data points (x, y) as follows: x is uniform over $[0,1]^{10}$ and $y = \sum_{i=1}^{10} i * x_i + 0.1 * N(0,1)$ where $N(0,1)$ is the standard normal distribution. Implement full gradient descent and stochastic gradient descent, and test them on linear regression over the synthetic data points.
Comments: The initialization, the learning rate, and the stop criterion are left for you to explore. Think about the reasons why you use a particular strategy for these; comments in the code on them are not required but will be appreciated.

4. (Coding) Implement the Perceptron algorithm and run it on the following synthetic data sets in $\mathbf{R}^{10}$: pick $w^* = [1,0,0,\cdots,0]$; generate 1000 points x by sampling uniformly at random over the unit sphere and then removing those that have margin γ smaller than 0.1; generate label $y = \text{sign}((w^*)^T x)$.

The End