Machine Learning 1 **Fall 24**

Homework 2

1. Suppose that Y is in the natural numbers. You decide to model Y given X with boosting. As your base model, you decide to use regression trees $\{h_j\}$ that take in covariates X and output real valued entries. Then,

   (a) You decide to make your model $P(Y|X) \sim \text{Poi}\left(\exp\left(f(X)\right)\right)$, where $f(X) = \sum_j \lambda_j h_j(X)$, with each $h_j(X)$ a (trained) regression tree.

      i. Why is Poisson a reasonable choice here?

      ii. Why do we use exp() as our link function?

   (b) Write down the loss function used for learning in terms of $f(x)$ and Y.

   (c) Give an expression for $\frac{dL}{df(x)}$.

   (d) Describe how you would choose each h_j in (first order) boosting.

2. Often, machine learning models involve so-called *hyperparameters*. These are quantities that are set by the analyst, rather than being learned by empirical risk minimization. For example, the regularization strength is a hyperparameter. One option for selecting hyperparameters is to try fitting a model with several different choices, and then using performance on a test set to select the best one.

   (a) Find (and prove) a non-trivial generalization bound for this procedure. That is, find a probabilistic bound on the gap between the risk estimated on the test set, and the true risk of the model. Give this bound in terms of (some subset of) the number of training examples n, the number of examples in the test set l, and the number of hyperparameters k.

   (b) What might go wrong if you use the same test set to select hyperparameters and to estimate the generalization error? In particular, consider the case where k is large relative to l.

   (c) Commonly, we instead split the data into a training set, a test set, and a *validation* set used to select hyperparameters.

   Use the provided dataset on white wine[1] and divide it randomly into 70% train, 10% validation, and 20% test. The last column "quality" is the target, an integer from 1-10. We'll view this as a regression problem with mean squared error as the desired loss function. Pick two model classes from the set { RandomForestRegressor, KNeighborsRegressor, MLPRegressor } and read their documentation in Scikit-learn to understand the hyperparameters of these models and what they do. Do grid search over a reasonably-large set of parameters (number of parameter sets > the number of points in the validation set): fit the model on the train set for each set of parameters, and select the "best parameters" using the score on the validation set. Finally, report the performance on the test set for each of the model classes you chose. What's the relationship between the empirical risk on the validation set and the empirical risk on the test set?

3. In class, we talked about the generalization behaviour of boosting as an increasing number of weak classifiers are included in the model fitting.

   (a) Use scikit learn to fit a gradient boosting model to dset0 (You should model the first column in **dset0.csv** given the other columns). Fit models with 1, 10, 100, 300, and 1000 trees. What do you observe about the generalization behaviour of these models?

   (b) Use scikit learn to fit a gradient boosting model to dset1 (You should model the first column in **dset1.csv** given the other columns). Fit models with 1, 10, 100, 300, and 1000 trees. What do you observe about the generalization behaviour of these models?

[1] a subset of the wine quality dataset

(c) If your observations in parts (a) and (b) differ, explain what property of the data might explain that.

4. So far in this course we have focused on fitting models $f_\mu(x)$ to approximate $\mathbb{E}[Y|X]$, the conditional mean of the target Y given the input X. Suppose we want to also incorporate a measure of uncertainty in our observations of Y, in the form of a function $f_\sigma(x) = \mathrm{Var}(Y|X)$ which models how the observation noise changes with the input X.

   (a) Say we have a neural architecture $g : \mathbb{R}^p \to \mathbb{R}^2$ that takes in a p-dimensional feature vector and returns a 2-dimensional vector of real numbers. Suggest a link function that would be appropriate for converting the output to f_μ and f_σ.

   (b) We could model data of this form as $Y|X \sim \mathcal{N}(f_\mu(X), f_\sigma(X)^2)$. Write down a loss function we could use for model training based on this model.

   (c) We have provided a dataset from taxi trips taken in NYC in `taxi_trips.parquet`. It contains 8 total columns, 7 features and 1 target (trip duration). Define a multi-layer perceptron in PyTorch or JAX that outputs a conditional mean and variance, and fit this on the taxi trip dataset. Make sure to follow best practices: define a baseline (or multiple) baseline models that you think your model should beat, split your dataset into training/validation/test sets, etc.

   (d) How would we use the model we've just fit to predict the probability that a trip will take less than 45 minutes? The normal distribution assumption is fine.

   (e) Estimate these probabilities on your held-out test set and report your accuracy.

5. For this problem, you will explore different methods for encoding categorical random variables. We have provided a dataset `retail.csv`, which contains data on retail transactions. We've cleaned up the data a bit so you only have two features: `StockCode` and `CustomerID`, both of which are categorical. (You may need to convert them to categorical data types in your code for the encoding methods to work properly.) The target is `Quantity`, which is integer-valued.

   (a) Start by exploring the cardinality of the two features. How many unique values are there for each feature? What does this suggest about how you might encode them? Is one-hot encoding feasible? To prepare for the next steps, split the data into training, validation, and test sets. The validation set will allow you to use early stopping during training for the neural network models.

   (b) Try target encoding. For each unique value of the feature, replace it with the mean of the target for that value (from the training set). You can use `TargetEncoder` from the `category_encoders` package. Using an appropriate loss function, fit a model of your choice to the data using the target-encoded features. Evaluate the model on the test set.

   (c) Try a neural network model with an embedding layer for each of the two features. Choose a reasonable embedding dimension. Fit this model to the data. Evaluate the model on the test set. How does it compare to the target encoding model?

   (d) What if we combine the two? That is, use both the target encoded features and the embeddings as input to the neural network. Fit this model to the data. Evaluate the model on the test set. How does it compare to the other two models?

   (e) Try to give some intuition for your results. What is the embedding layer doing? Why might it be useful in this context? Evaluate your findings on the test set.