COMS W4721 Spring 2020 Homework 2: Linear Classifiers, Decision Trees

Instruction

Please prepare your write-up as a typeset PDF document (which can be generated using LaTex or Word). If you choose to hand-write certain portions of the assignment, make sure your handwriting is legible and the consistency in the format with other pages which are typeset (e.g. page size, page numbering). If we cannot read your handwriting, you may not receive credit for the question. This write-up contains all of your supporting materials which include plots, source code, and proofs for each of the parts in the assignment. Submit your assignment on Gradescope by clearly marking the pages for each part. On the first page of your write-up, please typeset: (1) your name and your UNI; (2) all of your collaborators whom you discussed the assignment with; (3) the parts of the assignment you had collaborated on. The solutions to the problems need to start from the second page. Please write up the solutions by yourself. The academic rules of conduct is found in the course syllabus.

Suggestions

If necessary, please define notations and explain reasoning behind the solutions as concisely as possible. Solutions without explanations when needed may receive no credit. Points can be deducted for solutions with unnecessarily long explanations for lack of clarity. Source code comment can be useful for explaining the logic behind your solutions. Please start early!

Problem 1: Linear Regression (20 points)

In our lecture, we discussed linear regression in which the data $\{(\mathbf{x}^{(i)} \in \mathbb{R}^d, y^{(i)})\}_{i=1}^N$ is generated such that $y^{(i)}$'s independent of others when conditioned on $\mathbf{x}^{(i)}$.

Let us assume that $p(y^{(i)} \mid \mathbf{x}^{(i)}; \mathbf{w}) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(\frac{-(y^{(i)} - \mathbf{w}^T\mathbf{x}^{(i)})^2}{2\sigma^2}\right)$. In this problem we will explore the Bayesian formulation of linear regression called *maximum a posteriori estimate*:

$$
\begin{aligned}
\mathbf{w}_{\mathsf{MAP}} &= \arg\max_{\mathbf{w}} p(\mathbf{w} \mid \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^N) \\
&= \arg\max_{\mathbf{w}} \frac{p(\{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^N \mid \mathbf{w}) p(\mathbf{w})}{\int\limits_{\mathbf{w}'} p(\{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^N \mid \mathbf{w}') p(\mathbf{w}') d\mathbf{w}'} \\
&= \arg\max_{\mathbf{w}} p(\{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^N \mid \mathbf{w}) p(\mathbf{w}) \\
&= \arg\max_{\mathbf{w}} p(\{\mathbf{x}^{(i)}\}_{i=1}^N \mid \mathbf{w}) p(\{y^{(i)}\}_{i=1}^N \mid \{\mathbf{x}^{(i)}\}_{i=1}^N; \mathbf{w}) p(\mathbf{w}) \\
&= \arg\max_{\mathbf{w}} \ln p(\{\mathbf{x}^{(i)}\}_{i=1}^N \mid \mathbf{w}) + \ln p(\{y^{(i)}\}_{i=1}^N \mid \{\mathbf{x}^{(i)}\}_{i=1}^N; \mathbf{w}) + \ln p(\mathbf{w}) \\
&= \arg\max_{\mathbf{w}} \ln p(\{y^{(i)}\}_{i=1}^N \mid \{\mathbf{x}^{(i)}\}_{i=1}^N; \mathbf{w}) + \ln p(\mathbf{w}) \\
&= \arg\max_{\mathbf{w}} \left(\sum_{i=1}^N \ln p(y^{(i)} \mid \mathbf{x}^{(i)}; \mathbf{w}) \right) + \ln p(\mathbf{w})
\end{aligned}
$$

where the integral in the denominator can be dropped since it has no dependence on $\mathbf{w}$ and the natural log can be taken because it is a monotonic transform. For the last equation $\ln p(\{\mathbf{x}^{(i)}\}_{i=1}^N; \mathbf{w})$ is dropped because the data has no dependence on $\mathbf{w}$. Depending on the prior function on the weight $p(\mathbf{w})$, we can get several flavors of Bayesian linear regression.

(a) (10 points) Suppose each component of $\mathbf{w}$ is selected such that $w_i \sim N(0, \tau^2)$ i.i.d. What is $\mathbf{w}_{\mathsf{MAP}}$ in this case? Have we seen this form before?

(b) (10 points) Let us assume each component of $\mathbf{w}$ is selected such that $w^{(i)} \sim \text{Laplace}(0, b)$ i.i.d. What is $\mathbf{w}_{\mathsf{MAP}}$ in this case? Have we seen this form before?

Problem 2: Gaussian Discriminant Analysis (15 points)

(a) (5 points) Prove that a Gaussian discriminant analysis in $\mathbb{R}^d$ induces a quadratic decision boundary. You can assume a binary classification task for the purpose of this task. What properties of the parameters will ensure a linear decision boundary?

(b) (10 points) Let's take the Gaussian discriminant analysis in $\mathbb{R}^d$ from part (a). What properties of the parameters can make the classifier a Gaussian Naive Bayes Classifier? Give a short mathematical proof to validate your answer.

Problem 3: Logistic Regression (30 points)

(a) (15 points) Logistic Regression can be used to solve the binary classification task as depicted in the figure. A simple logistic regression model is given below:

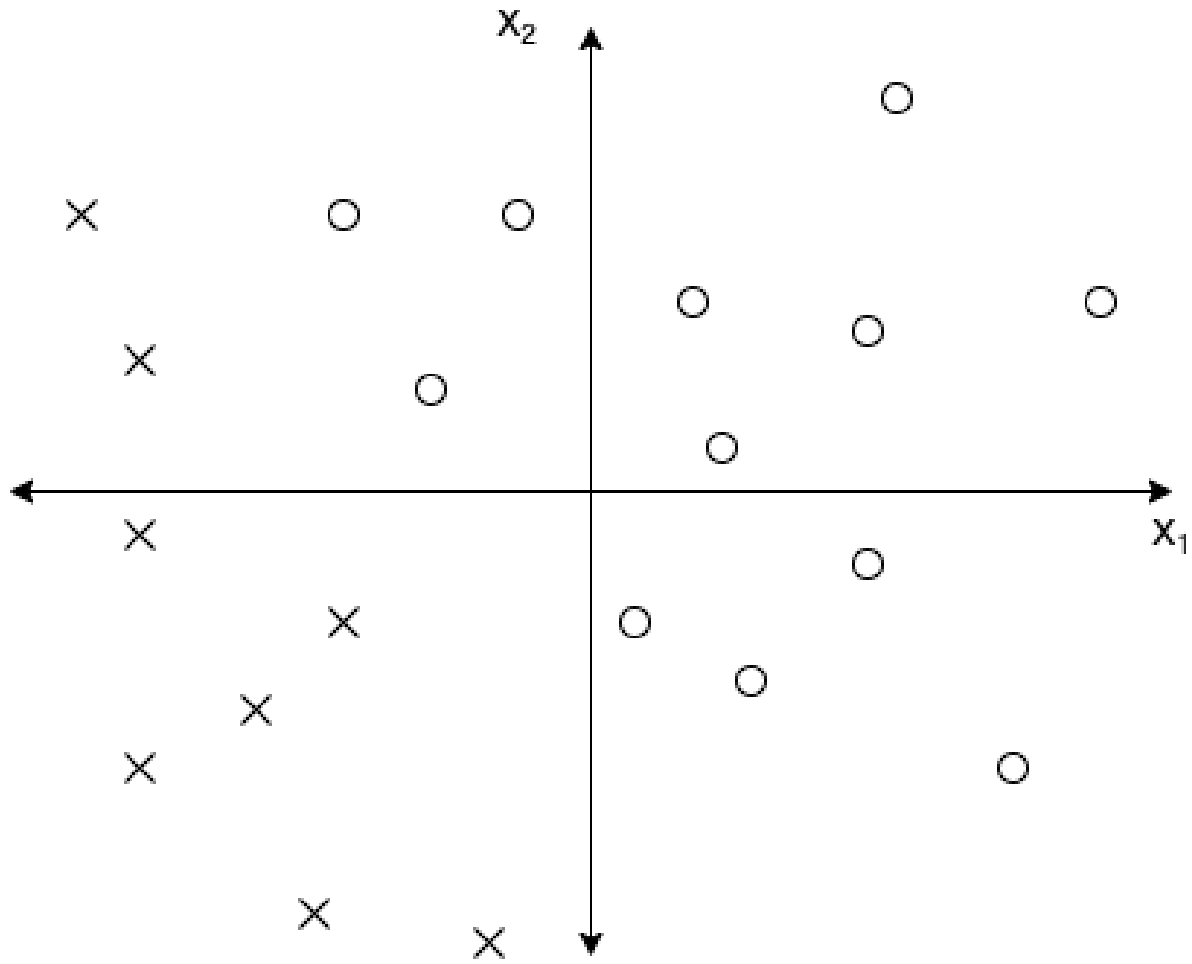

Figure 1: Binary classification dataset

$$P(y = class1 \mid \vec{x}, \vec{w}) = \frac{1}{1 + exp(-w_0 - w_1 x_1 - w_2 x_2)}$$

The given data is linearly separable at this point and thus a logistic regression model can be fit to separate the data with zero training error.
Consider a regularized logistic regression model where out optimization (maximization in this case) function becomes:

$$\sum_{i=1}^{N} log(P(y^{(i)} \mid x^{(i)}; w_0, w_1, w_2)) - \lambda \cdot w_j^2$$

for a very large λ. The regularization penalties used penalize one parameter at a time, ie. either one of $\{w_0, w_1, w_2\}$ are penalized at a given time. How does the training error change with regularization of each parameter? Provide a brief mathematical justification for each of your answers.

(b) (15 points) You are given a dataset *logistic_regression.csv* which has two features x_1, x_2 and the corresponding *class* label. Please write a small program in a language of your choice to optimize the logistic regression function to

- Fit the data in the csv file without regularization
- Regularize the squared weight of w_1 associated with the feature x_1.
  For each value of $\lambda \in [10^0, 10^1, 10^2, 10^3, 10^4, 10^5]$: plot the decision boundary along with the points in the dataset. You should have 6 plots.
- Regularize the squared weight of w_2 associated with the feature x_2.
  For each value of $\lambda \in [10^3, 10^4, 10^5, 10^6, 10^7, 10^8]$: plot the decision boundary along with the points in the dataset. You should have 6 plots.

Please attach all plots for all of the three subparts with your analysis. You may use scipy.optimize.minimize for implementing your code. You do not need to include the code. You may find the Python notebook for problem 3 useful for the starter code (please see section with "Modify Me"'s).

Problem 4: Decision Tree (35 points)

Decision trees are useful for classification. They follow a tree structure by splitting the data among the feature at each level which is most descriptive (gives largest information gain). For more information on decision trees see this chapter of Mitchell's machine learning book.

We will be classifying whether a banknote is fradulent or not using this Banknotes dataset. The dataset provides signal-based features of the banknotes and we will predict whether the banknote is fradulent (1) or not (0).

We will walk you through writing some key functions of a decision tree from scratch.

(a) (3 points) In the decision tree iPython notebook, start by importing the data (numpy array or pandas dataframe both work) and splitting the dataset by train and test (usually 80% train, 20% test). If you'd like to shuffle the data before splitting, you can use `numpy.random.shuffle`

(b) (8 points) Next, you will implement some functions in the `class DecisionTree`, which uses the `Node` class to build a decision tree. For each level, we calculate the current uncertainty, then split upon the feature at the threshold which gives us the highest information gain. First, implement the function `get_uncertainty` which implements the entropy and gini index uncertainty depending on the keyword `metric` that is given.

(c) (8 points) Using the `get_uncertainty` method, implement the `getInfoGain` method which calculates the information gain when splitting the data n `node.data` on `split_index`. `node.data` stores the relevant data at each node of the decision tree (root note will have all the data, then its children will split the data in 2 parts on the `split_index`, etc.)

(d) (10 points) Lastly, in `get_feature_threshold` find the feature that gives the largest information gain and **assign** the feature number (column number) to `node.feature` while updating the threshold values (check the docstrings for more detailed information).

(e) (6 points) After you implement the above steps, the `buildTree` function will recursively build the decision tree. Then when you run `homework_evaluate` which will call `self.predict` to evaluate the accuracy on the test set. Try different values of K (depth of tree a.k.a. number of features the tree will split on) and compare the performance. Which feature gives the largest information gain? Which feature is the least useful for the decision tree? How does varying the uncertainty metric (gini versus entropy) vary the performance (and why)? (For these questions we're looking for some analysis of the resulting model and some thought. Does not have to be a long paragraph.)

Congrats! You built a decision tree! Now (optional) try it on other data for fun. We've included a breast cancer dataset with both cardinal and categorical variables for you to try. With categorical variables, we one-hot encode them (e.g. for a 3-class categorical variable, 2 is represented as [0, 1, 0]) to allow the decision tree to classify them correctly (think about why this is the case, hint: does the magnitude of the number give us useful information here?).