CS461 Homework 3

Due: Nov. 17 11:59 pm

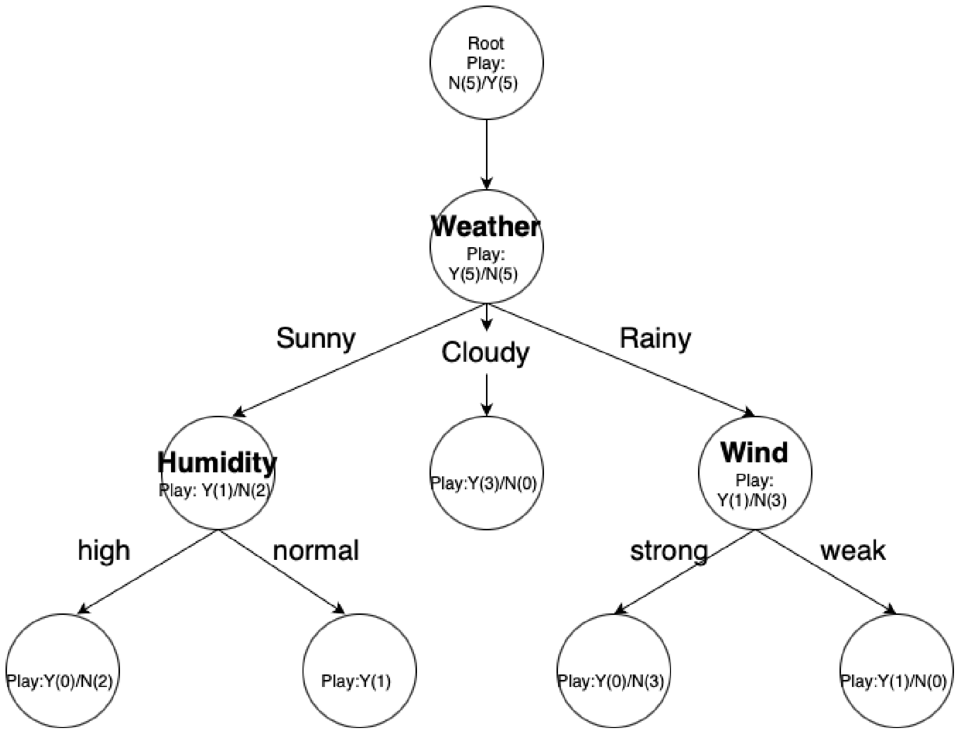
1. [Decision Tree] You will build a decision tree to determine whether or not a child goes out to play.

Day	Weather (X_0)	Temperature (X_1)	Humidity (X_2)	Wind (X_3)	Play (Y)
1	Sunny	Hot	High	Weak	No
2	Cloudy	Hot	High	Weak	Yes
3	Sunny	Mild	Normal	Strong	Yes
4	Cloudy	Mild	High	Strong	Yes
5	Rainy	Mild	High	Strong	No
6	Rainy	Cool	Normal	Strong	No
7	Rainy	Mild	High	Weak	Yes
8	Sunny	Hot	High	Strong	No
9	Cloudy	Hot	Normal	Weak	Yes
10	Rainy	Mild	High	Strong	No

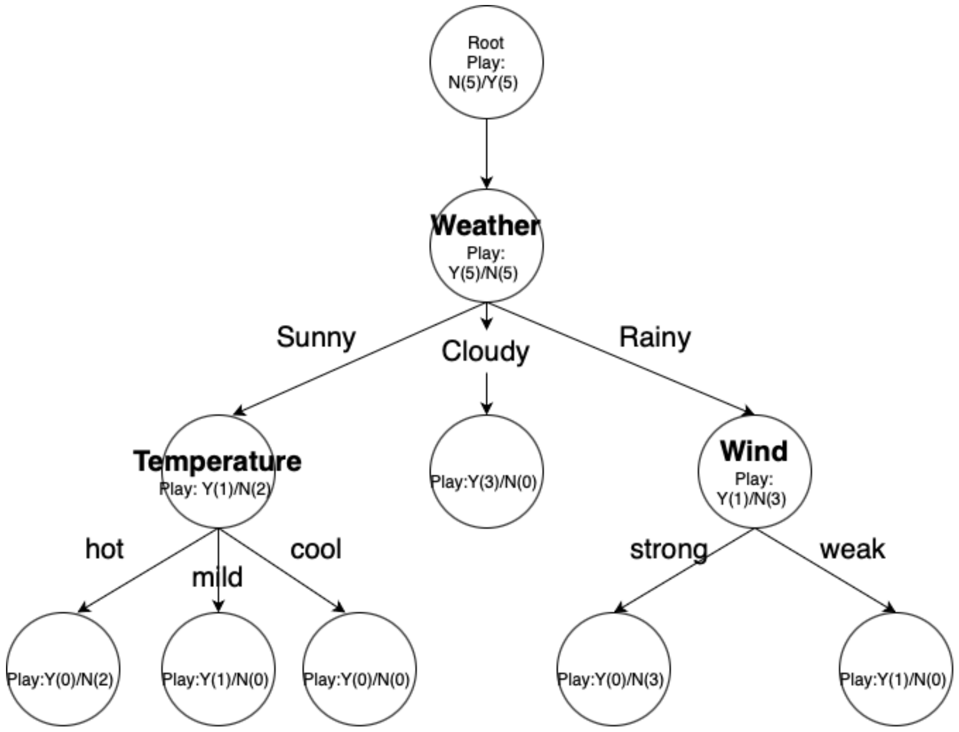
1.1 Calculate the information gain for each feature and select the feature with the highest information gain to serve as the root of the decision tree. Draw a root and split the ten training data points into some groups based on the value of the selected root feature.

1. In training, a tree node contains data points and a set of features to be chosen for split. If the node is a purity node: all yes/ no for play, then no need to split the node.
2. For each node, the feature to split k is selected based on maximal mutual information: $k* = \arg\max_k I(X_k; Y) = H(Y) - H(Y|X_k) \leftrightarrow \arg\min_k H(Y|X_k)$. The entropy and mutual information is computed using the sample points at each node.
3. Based on the different values of the feature to split, we create children nodes. Each child node will contain a subset of the sample points from parents nodes according to each feature value.

In this problem, decision tree algorithm begins at a root node containing all data points (10 samples) and the feature set to split, including weather, temperature, humidity, wind. Its descending nodes are iteratively created based on the rule described above. Two tree outcomes are possible, but the first tree has fewer leaves.



[Tree Solution - (1)]



[Tree Solution - (2)]

1.2 Repeat the two procedures: (1) selecting a feature and (2) splitting the data points until the leaf nodes of the tree achieve complete purity.

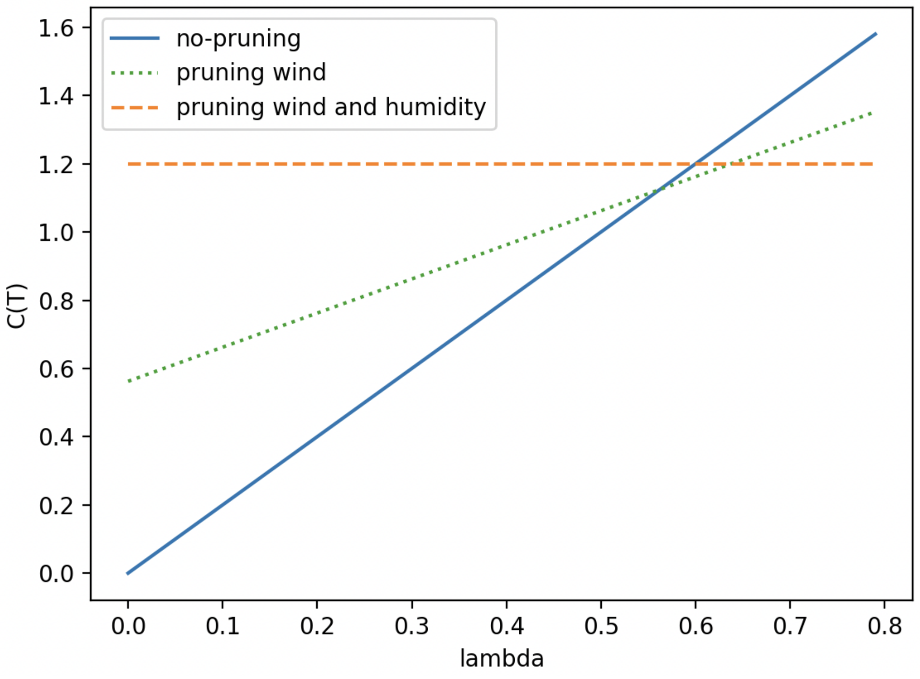
1.3 [Extra Points: 10 points] Try pruning your tree. You need to find a subtree (T') minimizing the criterion $C(T')$ below. Based on the criterion computation, do you think we need to prune the tree found in 1.2? Please show how different range of lambdas results in the different decisions on tree pruning.

$$C(T') = \sum_{\tau=1}^{T'} Q(\tau) + \lambda \cdot |\text{num of leaves in } T'|$$

$$Q(\tau) = \text{entropy of a leaf in } T' \text{ (measure of impurity)}$$

sol) The solution is based on the tree solution-1, but pruning decision for the tree solution-2 would not differ if Humidity is replaced with Temperature.

- no pruning T_1: $C(T1) = \lambda \cdot 5$
- pruning Humidity branches T_2: $C(T2) = H(1/3, 2/3) + \lambda \cdot 4$
- pruning Wind branches T_3: $C(T3) = H(1/4, 3/4) + \lambda \cdot 4$
- pruning Humidity and Wind branches T_4: $C(T4) = H(1/3, 2/3) + H(1/4, 3/4) + \lambda \cdot 3$
- The pruning decision depends on λ.
- $C(T2) > C(T3), \forall \lambda > 0$, so $T2$ is not our consideration to find a minimum.
- compare $C(T1), C(T3), C(T4)$
- for $0 \leq \lambda < H(1/4, 3/4)$, $C(T1)$ is minimum, so no pruning.
- for $H(1/4, 3/4) \leq \lambda < H(1/3, 2/3)$, $C(T3)$ is minimum, so pruning the wind branch.
- for $\lambda > H(1/3, 2/3)$, $C(T4)$ is minimum, so pruning both of humidity and wind branches.



2.[Perceptron] The Perceptron algorithm finds a decision boundary for binary classification below.

$$\begin{cases} \delta_w(x_1, x_2) = +1 & w_1x_1 + w_2x_2 > 0 \\ \delta_w(x_1, x_2) = -1 & w_1x_1 + w_2x_2 \leq 0 \end{cases} \tag{1}$$

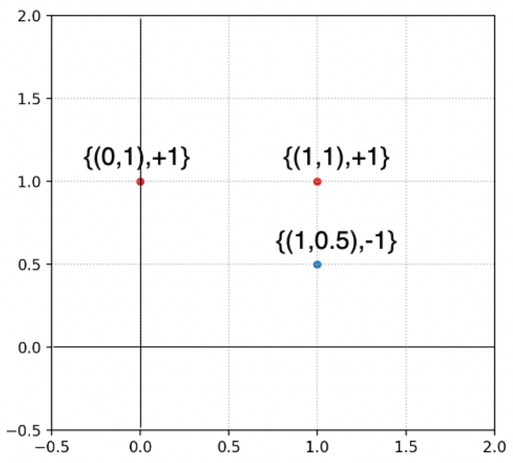
2.1 Assume a data set consists only of a single data point $\{(x_1, x_2), +1)\}$. How many iterations will be required until it finds a decision rule when the initial $w_0 = (0, 0)$ and step size $\eta = 1$?
sol)

- at $i = 0$, $w_0^t x = 0$ hence, mis-classification, so $w_1 = x$.
- at $i = 1$, $x^t x > 0$ right-classification, so one iteration will be needed.

2.2 How many iterations will be required until it finds a decision rule if the initial weight vector w_0 was initialized randomly and not as the all-zero vector?
sol) Perceptron algorithm will continue running until it finds the correct classification. As assuming S number of iterations until the right classification, the updated parameter will be $w_S = w_0 + S \cdot x$.

$$\begin{aligned} w_S^t x &= (w_0 + S \cdot x)^t x > 0 \\ &= w_0^t x + S||x||^2 > 0 \\ &\leftrightarrow S||x||^2 > -w_0^t x (w_0^t x < 0) \\ &\leftrightarrow S > -w_0^t x / ||x||^2 \end{aligned}$$

2.3 Suppose you have the three data points below. Please complete the iterative updates for w_i in Perceptron algorithm. The initial $w_0 = (0, 0)$ and step size $\eta = 1$; the point $(0, 1)$ is detected as misclassification at the first iteration so w is updated by the point: $w_1 = w_0 + 1 \cdot (0, 1)$



iteration	$\vec{w}$
0	$w_0 = (0, 0)$
1	$w_1 = (0, 0) + (0, 1) = (0, 1)$
2	detect $(1, 0.5)$ as misclassification $w_2 = (0, 1) - (1.0.5) = (-1, 0.5)$
3	detect $(1, 1)$ as misclassification $w_3 = (-1, 0.5) + (1, 1) = (0, 1.5)$
4	detect $(1, 0.5)$ as misclassification $w_4 = (0, 1.5) - (1, 0.5) = (-1, 1)$
5	detect $(1, 1)$ as misclassification $w_5 = (-1, 1) + (1, 1) = (0, 2)$
6	detect $(1, 0.5)$ as misclassification $w_6 = (0, 2) - (1, 0.5) = (-1, 1.5)$, stop.

3. (GDA: Gaussian Discriminant Analysis) We will build a binary classifier by using GDA.

3.1. Suppose someone gave you a decision rule (classifier) based on GDA, as shown below. Please use "./data_1/train.npz" and write the code "train3_1.py" to estimate the four statistics in the table below. $\mathcal{N}(x|\mu,\sigma^2) = \frac{1}{\sqrt{(2\pi\sigma^2)}} \exp\{\frac{-1}{2\sigma^2}(x-\mu)^2\}$

$$\begin{cases} \delta(x) = +1 & \mathcal{N}(x|\mu_{pos},\sigma^2_{pos}) \geq \mathcal{N}(x|\mu_{neg},\sigma^2_{neg}) \\ \\ \delta(x) = -1 & \mathcal{N}(x|\mu_{pos},\sigma^2_{pos}) < \mathcal{N}(x|\mu_{neg},\sigma^2_{neg}) \end{cases} \tag{2}$$

class	mean (μ)	var (σ^2)
class +	0.07	$\sigma^2 = 1.4$
class −	0.94	$\sigma^2 = 1.94$

3.2. Write the code "test3_2.py" to predict the class for the test data "./data_1/test.npz". What is the test accuracy?
sol) test accuracy is about 0.65

3.3. How would you improve your classifier? Is the decision rule given above optimal? Write a new code "test3_3.py" modifying "test3_2.py" and report new test accuracy. (hint: recall MAP rule.)
sol) Prior probability needs to be considered and the new test accuracy is 0.855. MAP rule provides an optimal decision rule to minimize classification error. As the number of training data points are limited (in this problem, the number of positive samples are limited), we need to consider the prior $P[C_k]$ for better classification results.

MAP rule:

$$\mathcal{K}^* = \arg\max_k P[C_k|x] \propto P[x|C_k]P[C_k]$$

3.4. Your will build a GDA classifier for 2D data. Use the data:"./data_2/train.npz" and write the code "train3_4.py" to estimate the four statistics and complete the table below.

class	mean ($\vec{\mu}$)	COV (Σ)
class +	$\begin{bmatrix} 0.01 \\ 0.06 \end{bmatrix}$	$\begin{bmatrix} 0.98 & 0.0 \\ 0.0 & 1.06 \end{bmatrix}$
class −	$\begin{bmatrix} -0.02 \\ -0.02 \end{bmatrix}$	$\begin{bmatrix} 1 & -0.01 \\ -0.01 & 4.98 \end{bmatrix}$

3.5 Write the code "test3_5.py" to predict the class for the test data: './data_2/test.npz". What is the test accuracy?
sol) Test accuracy is 0.84

3.6 [Extra Points: 10 points] The 2D data is generated by the densities specified below. Write the code "test3_6.py" based on the information and compute the new accuracy. Determine whether there is any significant change compared to the accuracy obtained in 3.5. Based on your accuracy comparison, discuss how GDA provides a reasonable framework for classification.

$$\begin{cases} x_{pos} \sim \mathcal{N}([0,0]^t, I) \\ \\ x_{neg} \sim 0.5 \times \mathcal{N}([0,2]^t, I) + 0.5 \times \mathcal{N}([0,-2]^t, I) \end{cases} \tag{3}$$

sol) Test accuracy is 0.85, no significant difference with GDA method. For complex data distribution, GDA would not be a good solution because complex decision boundaries will be required to classify the data points with high accuracy. In general, however, the pattern space for classification problem is often well-regularized (by providing a good feature space), so uni-modal Gaussian density, defined by mean and covariance, provides a reasonable density approximation enough to solve a classification problem. This homework problem serves as an example case.

4. [Logistic Regression] You will implement a spam mail detector by using logistic regression and enron data set (https://huggingface.co/datasets/SetFit/enron_spam/tree/main). The original data samples are plain texts, so a data preprocessing will be needed to convert text data to numerical values.

4.1 [Data Reprocessing] Run "preprocessing.py" to convert text data to numerical values. It will generate "spam_ham.csv". Please briefly explain the text vectorization process: tf-idf (Term Frequency-Inverse Document Frequency).

- As we represent a document into a vector form based on a vocabulary set, we can do the simple frequency counting for each of the words in vocabulary.
- However, if certain words are too common across all documents, the simple frequency/count would not provide enough discriminative information for classification. To address this, we assign weights to the terms that are frequent in the current document but rare but overall in the general collection. This concept is implemented using tf-idf.
- tf-idf weighting: $w_{i,j} = tf_{i,j} \times idf_i$ where $w_{i,j}$ denotes the vector component for document j and token i. $tf_{i,j}$ is word frequency and $idf_i = \log \frac{N}{n_i}$, where N is the total number of documents and n_i is the number of documents where the token i occurs.

4.2 [Dimensionality Reduction] Write the code "data4_2.py" to perform PCA dimensionality reduction on the 2000-D the data in "spam_ham.csv". We will reduce the data dimension to 50-D and split $4,000$ data samples into train: $3,500$ and test: 500. Please save them as "train4_2.npz" and "test4_2.npz".

4.3 [Logistic Regression] Write the code "train4_3.py" to train a logistic regression classifier using only NumPy library. Implement a gradient descent algorithm to find the global minimum of the Negative Log Likelihood (NLL) function defined below. Hint: You may use a step size about $1.0e-4$ and detect convergence when the change in NNL is less than 10, but you are free to choose your own values.

$$J(w) = -\ln P(t|w) = \sum_{n=1}^{N} -t_n \ln \sigma(w^t x_n) - (1-t_n)\ln(1-\sigma(w^t x_n))$$

4.4 [Logistic Regression] Report the train and test accuracy of your spam detector.
sol) both of train and test accuracy are about 0.96%

4.5 [Extra Points: 10 points] "mail.txt" is the text extracted from a spam email. Please test the text using your classifier and report the result whether it is classified as a spam.
sol)

1. for NLP prepossessing, "mail.txt" needs to be pre-processed together with "enron_spam_data.csv". Since tf-idf assigns a weight to each word in a document based on its term frequency and the reciprocal document frequency, a correct process can be done wihthin the context of whole data.
2. for PCA dimensionality reduction, PCA transform matrix computed using the train data must be used.
3. test 50-D vector on your trained classifier.
4. "mail.txt" is classified as a spam.