Homework #2

Due: 30th October 2023, Monday, before 11:59 pm

Submission instructions

- Submit your solutions electronically on the course Gradescope site as PDF files.
- If you plan to typeset your solutions, please use the LaTeX solution template. If you must submit scanned handwritten solutions, please use a black pen on blank white paper and a high-quality scanner app.
- You are required to paste the snapshots of your codes with the title 'Source Codes' at the end of your solutions pdf.

Problem 1 (Decision Trees)

[15 points total]

You are checking the weather and would like to know whether a day is comfortable or not (IsComfortable). This is determined by factors such as whether a day is sunny (IsSunny), whether a day is hot (IsHot), etc. You have the following data to consider.

Example	IsSunny	IsHot	IsHumid	IsWindy	IsComfortable
A	0	0	0	0	0
B	0	0	1	0	0
C	1	1	0	1	0
D	1	0	0	1	1
E	0	1	1	0	1
F	0	0	1	1	1
G	0	0	0	1	1
H	1	0	1	1	1
X	1	1	1	1	?
Y	0	1	0	1	?
Z	1	1	0	0	?

You know whether or not a day A through H is comfortable, but you do not know about X through Z. For questions (a) - (d), consider only days A through H. For all calculations in this problem, use base 2 for the logarithms.

(a) **[1 points]** What is the entropy of IsComfortable?

(b) **[1 points]** Which attribute should you choose as the root of a decision tree?

(c) **[5 points]** What is the information gain of the attribute you chose in the previous question?

(d) **[5 points]** Using ID3 algorithm, build a decision tree to classify days as comfortable or not.

(e) **[3 points]** Classify days X, Y and Z using this decision tree as comfortable or not.

Problem 2 (Entropy and Information)

[5 points total]

The entropy of a Bernoulli (Boolean 0/1) random variable X with $p(X = 1) = q$ is given by

$$B(q) = -q \log q - (1 - q) \log(1 - q).$$

Suppose that a set S contains p positive examples and n negative examples. The entropy of S is defined as $H(S) = B\left(\frac{p}{p+n}\right)$.

Based on an attribute X_j, we split the set S into k disjoint subsets S_k, with p_k positive and n_k negative examples in each. If the ratio $\frac{p_k}{p_k+n_k}$ is the same for all k, show that the information gain of this attribute is 0.

Problem 3 (k-Nearest Neighbors)

[25 points total]

In the following questions, you will consider a k-nearest neighbor classifier using Euclidean distance metric on a binary classification task. We assign the class of the test point to be the class of the majority of the k nearest neighbors. Below is the figure for dataset. Note that the annotated data points (from 1 to 5) belong to the test set, and the rest belong to the training set.

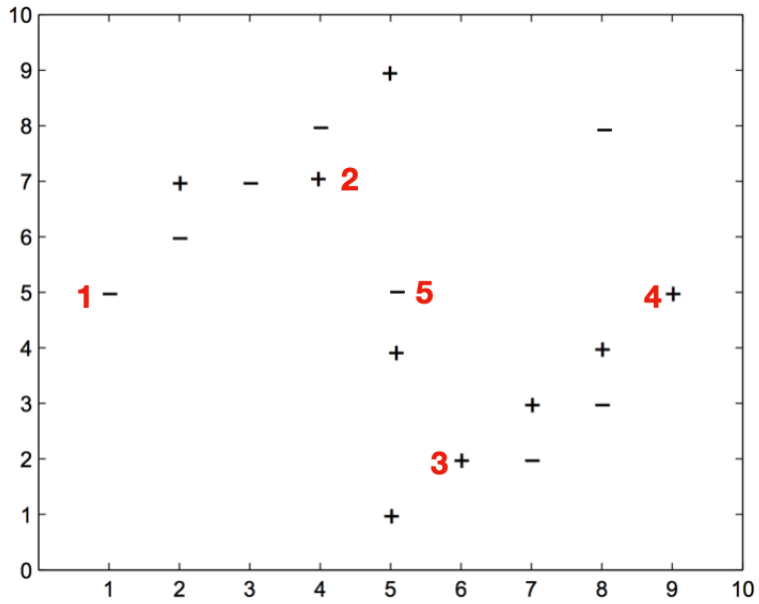

Figure 1: Dataset for kNN binary classification task.

(a) **[5 points]** What value of k minimizes the training set error? What is the resulting training error? Assume that for the purpose of computing nearest neighbors for the training set, a point can be its own neighbor.

(b) **[10 points]** In Figure 1, the points labelled as $1, 2, 3, 4, 5$ are test points, whereas all other points are training points. Using the k you obtained from question (a), what is the test error for the data points marked as $1, 2, 3, 4, 5$? Clearly mention which points are misclassified. Note that a test point cannot be its own nearest neighbor as neighbors must belong to the training set.

In the above parts, we had considered k-nearest neighbor classifier using Euclidean distance metric. For the next part, we will choose *cosine similarity* as a distance metric. We define cosine similarity between two vectors $\mathbf{u}, \mathbf{v}$ as:

$$sim(\mathbf{u}, \mathbf{v}) = \frac{\langle \mathbf{u}, \mathbf{v} \rangle}{||\mathbf{u}||_2 ||\mathbf{v}||_2}$$

where $< \cdot, \cdot >$ is the inner product operator between two vectors, and $||.||_2$ is the L_2 norm operator.

Note that for cosine similarity, we consider the nearest neighbors to be the points in the training set with the greatest cosine similarity (i.e. most similar points) to the test point.

[*Hint*: Quoting from Wikipedia: "Cosine similarity is the cosine of the angle between the vectors; that is, it is the dot product of the vectors divided by the product of their lengths. It follows that the cosine similarity does not depend on the magnitudes of the vectors, but only on their angle."]

(c) **[10 points]** Using the k you choose from question (a), what is the test error for the data points marked as $1, 2, 3, 4, 5$ in Figure 1? Clearly mention which points are misclassified. (Assume that a point is not its own neighbor).

Problem 4 (Perceptron and Logistic Regression)

[13 points total] Suppose we have a training set with 6 samples, each sample has feature vector in $\mathbb{R}^3$, and every feature vector $\boldsymbol{x}^{(i)}$ has been augmented with a bias term such that $\boldsymbol{x}_0^{(i)} = 1$ for all i:

i	1	2	3	4	5	6
$\boldsymbol{x}^{(i)}$	[1, 4,0]	[1, 1,1]	[1, -2,1]	[1, 1,0]	[1, 5,2]	[1, 3, -1]
$y^{(i)}$	1	-1	1	1	-1	-1

We are going to use perceptron algorithm to train a linear classifier with 3 dimensional weight vector $\boldsymbol{\theta} \in \mathbb{R}^3$ (with bias term). We start with initial weight vector as the first sample in our dataset, i.e. $\boldsymbol{\theta} = \boldsymbol{x}^{(1)}$. Note: when $\boldsymbol{\theta}^T \mathbf{x} = 0$, we will assume the algorithm predicts $+1$.

(a) **[2 points]** Plot the data in Python using matplotlib with x_1 and x_2 as the two axis (ignoring the bias term $\boldsymbol{x}_0^{(i)} = 1$). Use different colors to denote $+1/-1$ labels. Paste a screenshot of your plot in the writeup.

(b) **[2 points]** Is the data linearly separable? Will our algorithm converge if we run it several times over the same sequence? Explain.

(c) **[4 points]** Regardless of whether the dataset is linearly separable or not, derive and calculate (by hand) the updates of the parameters on this dataset for one round over the entire dataset using a learning rate of $\alpha = 0.1$.

(d) **[5 points]** Now, let's adopt a probabilistic perspective to tackle this binary classification problem using logistic regression. Given a logistic regression model for binary classification, the hypothesis is represented as

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = g(\boldsymbol{\theta}^T \boldsymbol{x})$$

where $g(z) = \frac{1}{1+e^{-z}}$. Here $\boldsymbol{x}$ is the feature vector (with $x_0 = 1$ for bias) and $\boldsymbol{\theta}$ is the parameter vector. As learned in class, the loss function for logistic regression is:

$$J(\boldsymbol{\theta}) = -\frac{1}{n} \sum_{i=1}^{n} \left[y^{(i)} \log h_{\boldsymbol{\theta}} \left(\boldsymbol{x}^{(i)} \right) + \left(1 - y^{(i)} \right) \log \left(1 - h_{\boldsymbol{\theta}} \left(\boldsymbol{x}^{(i)} \right) \right) \right]$$

Suppose we are given parameter $\boldsymbol{\theta} = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}^T$,

i. Compute $h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)})$ for each sample in the training set.

ii. Compute $J(\boldsymbol{\theta})$ for the training set.

Note: When referring to the logarithm, we mean the natural logarithm. For the purpose of calculating the logistic regression loss, map the class label $y = -1$ to $y = 0$, while the class label $y = 1$ remains as $y = 1$.

Problem 5 (Programming Exercise: Applying Classification Models)

[40 points total]

In this problem, we ask you to compare the classification models we have studied till now i.e., Decision trees, K-nearest neighbors, and Logistic regression.

Introduction[1]

This data was extracted from the 1994 Census bureau database by Ronny Kohavi and Barry Becker. For computational reasons, we have already extracted a relatively clean subset of the data for this homework. The prediction task is to determine whether a person makes over $50K a year.

In this problem, we ask you to complete the analysis of what sorts of people were likely to earn more than $50K a year. In particular, we ask you to apply the tools of machine learning to predict which individuals are more likely to have high income.

Starter Files

---

code and data

- `Fall23-CS146-HW2.ipynb`. Instruction is here [2].
- `nutil.py`
- `adult_subsample.csv`

documentation

- Decision Tree Classifier:
  http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html
- K-Nearest Neighbor Classifier:
  http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html
- Logistic Regression:
  https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
- Cross-Validation:
  https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_val_score.html
  https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_validate.html
- Metrics:
  http://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html
  https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html

---

[1] This assignment is adapted from the UCI Machine learning repository, available at https://archive.ics.uci.edu/ml/datasets/adult.

[2] To run the notebook on Google Colab, check the first 3 cells in `Fall23-CS146-HW2.ipynb`; otherwise, delete the first 3 cells.

Visualization

One of the first things to do before trying any formal machine learning technique is to dive into the data. This can include looking for funny values in the data, looking for outliers, looking at the range of feature values, what features seem important, etc.

Note: We have already converted all the categorical features to numerical ones. The target column is the last one: ">50k", where 1 and 0 indicate >50k or $\leq$ 50k respectively. The feature "fnlwgt" describes the number of people the census believes the entry represents. All the other feature names should be self-explanatory. If you want to learn more about this dataset, please click here.

(a) **[5 points]** Plot and submit histograms for each feature, separating the examples by class (e.g. income greater than 50k or smaller than or equal to 50k). This should produce fourteen plots, one for each feature. Each plot should have two overlapping histograms, with the color of the histogram indicating the class.

Evaluation

Now, let us use `scikit-learn` to train a `DecisionTreeClassifier`, `KNeighborsClassifier`, and `LogisticRegression` on the data.

Using the predictive capabilities of the `scikit-learn` package is very simple. In fact, it can be carried out in three simple steps: initializing the model, fitting it to the training data, and predicting new values.[3]

(b) **[0 points]** Before trying out any classifier, it is often useful to establish a *baseline*. We have implemented one simple baseline classifier, `MajorityVoteClassifier`, that always predicts the majority class from the training set. Read through the `MajorityVoteClassifier` and its usage and make sure you understand how it works.

Your goal is to implement and evaluate another baseline classifier, `RandomClassifier`, that predicts a target class according to the distribution of classes in the training data set. For example, if 85% of the examples in the training set have `>50k = 0` and 15% have `>50k = 1`, then, when applied to a test set, `RandomClassifier` should randomly predict 85% of the examples as `>50k = 0` and 15% as `>50k = 1`.

Implement the missing portions of `RandomClassifier` according to the provided specifications. Then train your `RandomClassifier` on the entire training data set, and evaluate its training error. If you implemented everything correctly, you should have an error of 0.385 or 0.374, depending on an implementation detail; both error values are correct. Paste a screenshot of your code.

[3] Note that almost all of the model techniques in `scikit-learn` share a few common named functions, once they are initialized. You can always find out more about them in the documentation for each model. These are `some-model-name.fit(...)`, `some-model-name.predict(...)`, and `some-model-name.score(...)`.

(c) **[5 points]** Now that we have a baseline, train and evaluate a `DecisionTreeClassifier` (using the `DecisionTreeClassifier` class from `scikit-learn` and referring to the documentation as needed). Make sure you initialize your classifier with the appropriate parameters; in particular, use the 'entropy' criterion discussed in class. What is the training error of this classifier? Paste a screenshot of your code.

(d) **[5 points]** Similar to the previous question, train and evaluate a `KNeighborsClassifier` (using the class from `scikit-learn` and referring to the documentation as needed). Use k=3, 5 and 7 as the number of neighbors and report the training error of this classifier respectively. Paste a screenshot of your code.

(e) **[5 points]** Similar to the previous question, train and evaluate a `LogisticRegression` (using the class from `scikit-learn` and referring to the documentation as needed). Use λ=0.1, 1 and 10 as the regularization hyperparameter and report the training error of this classifier respectively. Make sure you initialize your classifier with the appropriate parameters; `random_state`=0 and `max_iter`=1000. (*Hint*: function argument `C` is the inverse of regularization strength, therefore $\texttt{C} = 1/\lambda$.) Paste a screenshot of your code.

(f) **[10 points]** So far, we have looked only at training error, but as we learned in class, training error is a poor metric for evaluating classifiers. Let us use cross-validation instead.

Implement the missing portions of `error(...)` according to the provided specifications. You may find it helpful to use `StratifiedShuffleSplit(...)` from `scikit-learn`. To ensure that we always get the same splits across different runs (and thus can compare the classifier results), set the `random_state` parameter to be the same (e.g., 0).

Read up about F1 scores here. Next, use your `error(...)` function to evaluate the training error and (cross-validation) validation error and validation micro averaged F1 Score of the `RandomClassifier`, `DecisionTreeClassifier`,`KNeighborsClassifier`, and `LogisticRegression` models. For the `DecisionTreeClassifier`, use 'entropy' criterion, for the `KNeighborsClassifier`, use $k = 5$, for the `LogisticRegression`, use $\lambda = 1$, `random_state`=0 and `max_iter`=1000. To do this, generate a random 85/15 split of the training data, train each model on the 85% fraction, evaluate the error on both the 85% and the 15% fraction, and repeat this splitting 100 times to get an average result. What are the average training error, validation error, and validation F1 score of each of your classifiers on the adult_subsample data set? Paste a screenshot of your code.

(g) **[5 points]** One way to find out the best value of k for `KNeighborsClassifier` is n-fold cross validation. Find out the best value of k using 5-fold cross validation. You may find the `cross_val_score(...)` from `scikit-learn` helpful. Run 5-fold cross validation for all odd numbers ranging from 1 to 50 as the number of neighbors. Then plot the validation score against the number of neighbors, k. Include this plot in your writeup, and provide a 1-2 sentence description of your observations. What is the best value of k and what is the corresponding score? Paste a screenshot of your code.

(h) **[5 points]** One problem with decision trees is that they can *overfit* to training data, yielding complex classifiers that do not generalize well to new data. Let us see whether this is the case for the adult_subsample data.

One way to prevent decision trees from overfitting is to limit their depth. Repeat your cross-validation experiments but for increasing depth limits, specifically, $1, 2, \ldots, 20$. Then plot the average training error and validation error against the depth limit. You may find the `cross_validate(...)` from `scikit-learn` helpful; note that this function returns accuracy, while here, error $= 1 -$ accuracy. Include this plot in your writeup, making sure to label all axes and include a legend for your classifiers. What is the best depth limit to use for this data? Do you see overfitting? Justify your answers using the plot. Paste a screenshot of your code.