1 Density Estimation

Characterizing the distributions of samples in datasets is a core problem in statistical machine learning. Methods for estimation of distribution can be categorized into parametric and non-parametric groups. The parametric methods assume known shape of the distributions and attempt to estimate the value (*Frequentists' approach*) or distribution (*Bayesian approach*) of parameters. The non-parametric methods do not assume the shape of distribution. Instead the distribution density of one sample is entirely determined by the datasets. In this assignment problem we study two examples: parametric maximum likelihood estimation (MLE) and non-parametric density estimation.

Note: please be concise and clear in your derivation

(a) (**10 points**) Let's start with maximum likelihood estimation of implementing MLE for parameters of some important distributions. Suppose we have N **i.i.d** samples $x_1, x_2, \cdots, x_n$. We will practice the maximum likelihood estimation techniques to estimate the parameters in each of the following cases:

- We assume that all samples can only take value between 0 and 1, and they are generated from the Beta distribution with parameter α unknown and $\beta = 1$. Please show how to derive the maximum likelihood estimator of α.
- We assume that all samples are generated from Normal distribution $\mathcal{N}(\boldsymbol{\theta}, \text{diag}(\boldsymbol{\theta}))$. Please show how to derive the maximum likelihood estimator of $\boldsymbol{\theta} \in \mathbb{R}^d$ where $\text{diag}(\boldsymbol{\theta})$ represents a square matrix with diagonal elements equal to vector $\boldsymbol{\theta}$ and all other elements 0.

(b) (**10 points**) MLE in linear regression: In class lectures, we studied the probabilistic interpretation of linear regression model ($P(y|\boldsymbol{x}, \boldsymbol{\theta}) := \mathcal{N}(\omega_0 + \boldsymbol{\omega}^T\boldsymbol{x})$) and used MLE to estimate its parameters. We noticed that estimating $\sigma, \omega_0, \boldsymbol{\omega}$ are decoupled.

- Prove that

$$\hat{\omega}_0 = \frac{1}{N}\sum_i y_i - \frac{1}{N}\sum_i \boldsymbol{\omega}^T\boldsymbol{x}_i = \bar{y} - \boldsymbol{\omega}^T\bar{\boldsymbol{x}} \tag{1}$$

  So $\hat{\omega}_0$ models the difference in the average output from the average predicted output. Also show that

$$\hat{\boldsymbol{\omega}} = (X_c^T X_c)^{-1} X_c^T Y_c = [\sum_{i=1}^N (\boldsymbol{x}_i - \bar{\boldsymbol{x}})(\boldsymbol{x}_i - \bar{\boldsymbol{x}})^T]^{-1} [\sum_{i=1}^N (y_i - \bar{y})(\boldsymbol{x}_i - \bar{\boldsymbol{x}})] \tag{2}$$

  where $\bar{\boldsymbol{x}}$ is empirical mean of $\{\boldsymbol{x}_i, i = 1 \cdots N\}$, X_c is the centered input matrix containing $\boldsymbol{x}_i^c = \boldsymbol{x}_i - \bar{\boldsymbol{x}}$ along its rows, and Y_c is the centered output vector.
- According to above sub-problem, we can firstly compute $\hat{\boldsymbol{\omega}}$ on the centered data, and then estimate ω_0 using equation (1). However, let's temporarily forget the task of estimating regression parameters $\boldsymbol{\omega}$ and consider the distribution of $[\boldsymbol{x}_i^c, y_i^c] \in \mathbb{R}^{(d+1)}$. Assume $\boldsymbol{x}$ and y both follow Gaussian distribution. Then by finding the MLE of $\Sigma_{X^cY^c}, \Sigma_{X^cX^c}$, and using the formula for conditional Gaussian, derive equation (2). (You may refer to conditional Gaussian Distributions)

(c) (**15 points**) Non-parametric Density Estimation:

Suppose random variables $X_1, X_2, \cdots, X_n$ are i.i.d sampled according to density function $f(x)$ and the kernel density estimation is in the form of $\hat{f}(x) = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{h} K(\frac{x-X_i}{h})$. Show the bias of the kernel density estimation method by the following steps:

- Show $\mathbf{E}_{X_1,\cdots,X_n}[\hat{f}(x)] = \frac{1}{h}\int K(\frac{x-t}{h}) f(t) dt$.
- Use Taylor's theorem around x on the density $f(x - hz)$ with $z = \frac{x-t}{h}$.
- Compute the bias $\mathbf{E}[\hat{f}(x)] - f(x)$.

(d) (**5 points**) MLE for Density Estimation:

Show that MLE cannot be used to estimate optimal value of h in $\hat{f}(x) = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{h} K(\frac{x-X_i}{h})$.

2 Nearest Neighbor

(a) Suppose we have the locations (coordinates) of 10 USC students during class time, and we know their majors, as follows:
Mathematics: $\{(15, 49), (-7, 38), (-4, 47)\}$
Electrical Engineering: $\{(29, 24), (32, 36), (37, 43)\}$
Computer Science: $\{(18, 9), (40, -28), (-8, -19), (-11, 12)\}$

- (**5 points**) Normalize the data, by following the formula on K-Nearest Neighbor lecture slide, page 53.
- (**10 points**) Suppose we have a student whose coordinate is at $(9, 18)$ with unknown major. Using K-Nearest Neighbor with L_2 distance metric, predict the student's major if we are using $K = 1$ (**2 points**) and if we are using $K = 3$ (**2 points**). Similarly, what are the student's major predictions if we use L_1 distance metric with $K = 1$ (**2 points**) and with $K = 3$ (**2 points**). For $K = 3$, if there is a tie, please choose the label of the data point with closer distance. Please compare the results between these 4 different predictions (**2 points**). Do not forget to normalize/standardize the coordinate of the student with unknown major using the mean and standard deviation of the students with known major. Provide intermediate computations of how do you arrive at your predictions.

(b) Suppose now we want to derive a probabilistic K-Nearest Neighbor for classification of an unlabeled data point $\mathbf{x}$, which is D-dimensional. We have a (multi-dimensional) D-sphere with center at $\mathbf{x}$, allowing its radius to grow until it precisely contains K labeled data points, irrespective of their class. At this size, the volume of the sphere is V. Let there be a total of N labeled data points in the entire space (both inside and outside of the sphere), with N_c data points labeled as class c, such that $\sum_c N_c = N$. Also, a subset of the K data points inside of the sphere belongs to class c, there are K_c of them in total. We model estimated density associated with each class as $p(\mathbf{x} \mid Y = c) = \frac{K_c}{N_c V}$ and the class prior as $p(Y = c) = \frac{N_c}{N}$.[1]

[1] This problem is a rephrase of some materials from "Pattern Recognition and Machine Learning" book by Christopher M. Bishop. However, even without having/reading the book, this problem should be fairly easy to solve, just by following the description in this problem set.

- (**5 points**) Using the fact that $\sum_c K_c = K$, derive the formula for unconditional density $p(\mathbf{x})$.
- (**5 points**) Using Bayes rule, derive the formula for the posterior probability of class membership $p(Y = c \mid \mathbf{x})$.

3 Additive or Dropout Noising as Regularization

(**20 points**) In lecture slides, we have studied ridge regression which include a weight decay term $\lambda||\boldsymbol{\omega}||_2^2$ in the objective function. This term is useful in the case of dependent features or insufficient training samples. In this problem, we will see another way of introducing weight decay terms. Consider a linear regression model of the form:

$$y(\boldsymbol{x}, \omega) = \omega_0 + \boldsymbol{\omega}^T \boldsymbol{x} \tag{3}$$

together with a sum-of-squares error function of the form

$$E(\omega) = \frac{1}{2} \sum_{n=1}^{N} \{y(x_n, \omega) - t_n\}^2 \tag{4}$$

(a) Now suppose that additive Gaussian noise $\epsilon_i \sim \mathcal{N}(0, \sigma^2 I_D)$ is added *independently* to each of the input variables $\boldsymbol{x}_n$ to generate new sample $\tilde{\boldsymbol{x}}_n$ in the neighborhood of original point $\boldsymbol{x}_n$. For each sample $\boldsymbol{x}_n$ we tried M times of noising, and in this way we created a M times larger augmented dataset of $\{\tilde{x}_1^{(1)}, \tilde{\boldsymbol{x}}_1^{(2)}, \cdots \tilde{\boldsymbol{x}}_1^{(M)}, \cdots \tilde{\boldsymbol{x}}_N^{(M)}\}$. By making use of $E[\epsilon_i] = 0$ and $E[\epsilon_i \epsilon_j] = \delta_{i,j}\sigma^2$, show that minimizing E averaged over the noise distribution

$$E(\omega) = \frac{1}{2} \sum_{m=1}^{M} \sum_{n=1}^{N} \{y(\tilde{\boldsymbol{x}}_n^{(m)}, \omega) - t_n\}^2 \tag{5}$$

is equivalent to minimizing the sum-of-squares error for noise-free input variables with the addition of a *weight-decay regularization* term, in which the bias parameter w_0 is omitted from the regularizer.

(b) Then consider another way of artificial feature noising with *dropout* noise on *each dimension* of *each sample* independently: $\tilde{\boldsymbol{x}}_n := \nu(\boldsymbol{x}_n, \epsilon_n) = \boldsymbol{x}_n \odot \epsilon_n$, where $\odot$ represents the elementwise product of two vectors. Each component of $\epsilon_n \in \{0, (1-\delta)^{-1}\}^d$ is an independent draw from a saled Bernoulli$(1-\delta)$ random variables. In other words, dropout noise corresponds to setting $\tilde{x}_{n,d}$ to 0 with probability δ and to $x_{n,d}/(1-\delta)$ with probability $1-\delta$.

$$\tilde{x}_{n,d} = \begin{cases} 0 & \text{w.p. } \delta \\ x_{n,d}/(1-\delta) & \text{w.p. } 1-\delta \end{cases}$$

where $\delta \in [0, 1)$ is the probability one dimension of one sample is dropped, i.e. set to value 0.

- Show that $\mathbb{E}[\tilde{\boldsymbol{x}}_n] = \boldsymbol{x}_n$, and $\mathrm{Var}[\boldsymbol{\omega}^T \tilde{\boldsymbol{x}}_n] = \frac{\delta}{1-\delta} \sum_{d=1}^{D} x_{n,d}^2 \omega_d^2$

- By making use of above expressions of $\mathbb{E}[\tilde{\boldsymbol{x}}_n]$ and $\text{Var}[\boldsymbol{\omega}^T\tilde{\boldsymbol{x}}_n]$, show that minimizing E averaged over the noise distribution

$$E(\omega) = \frac{1}{2}\sum_{m=1}^{M}\sum_{n=1}^{N}\{y(\tilde{\boldsymbol{x}}_n^{(m)}, \omega) - t_n\}^2 \tag{6}$$

is equivalent to minimizing the sum-of-squares error for noise-free input variables with the addition of a *weight-decay regularization* term, in which the bias parameter w_0 is omitted from the regularizer.

4 Decision Tree

(a) (**2 points**) Suppose you want to grow a decision tree to predict the *Play-Tennis Feasibility* based on the following data which provides the feasibility of playing tennis in 14 observations. Which predictor variable will you choose to split in the first step to maximize the information gain?

Day	Outlook	Temperature	Humidity	Wind	Play-Tennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

(b) (**3 points**) Which attributes will you choose for the second level of the tree? You need to show all the calculations.

(c) (**10 points**) In training decision trees, the ultimate goal is to minimize the classification error. However, the classification error is not a smooth function; thus, several surrogate loss functions have been proposed. Two of the most common loss functions are the *Gini index* and *Cross-entropy*, see [MLaPP, Section 16.2.2.2] or [ESL, Section 9.2.3] for the definitions. Prove that, for any discrete probability distribution p with K classes, the value of the Gini index is less than or equal to the corresponding value of the cross-entropy. This implies that the Gini index is a better approximation of the misclassification error.

Definitions: For a K-valued discrete random variable with probability mass function $p_i, i = 1, \ldots, K$ the Gini index is defined as: $\sum_{k=1}^{K} p_k(1 - p_k)$ and the cross-entropy is defined as $-\sum_{k=1}^{K} p_k \log p_k$.

5 Programming

In the programming problem, you will write a MATLAB program to train linear regression on synthetic dataset with outlier samples. You can get newest version of Matlab from http://itservices.usc.edu/matlab/.

5.1 Outliers in Linear Regression

(**25 points**) This problem provides a case study of robustness of linear regression using a synthetic toy data, i.e. how unusual sample affect learned models. It is very common to model the noise in regression models using zero-mean Gaussian distribution. In this case, MLE of linear regression parameters is equivalent to minimizing sum of squared residuals (*residual* of one sample x_i is defined as difference between true value and predicted value of this sample: $|y_i, \hat{y}_i|$). However, if we have unusual samples (e.g. outliers) in our dataset, this can result in a poor fit: squared error penalizes residuals quadratically, so points *far* from regression line have *more influence on fit* than points *near* to the line.

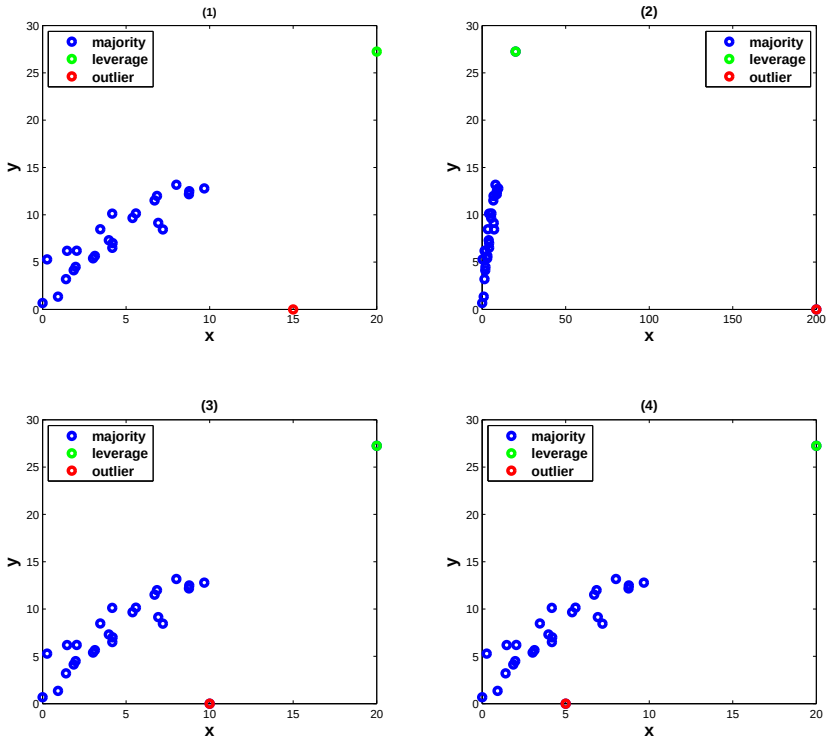

Figure 1: influential samples in learning linear regression model

In data file demoSynLR.mat are 4 synthetic dataset {data1, data2, data3,data4}, as shown in subplot (1) $\sim$ (4) in figure (1). The data points are generated from a linear model with 1-

dimensional feature x and 1-dimensional response y. In each of the subplots, we manually inserted outlier points (15,0), (200,0), (10,0) and (5,0).

- For each of 4 training datasets, i.e. {data1, data2, data3,data4}, train a linear regression model *with* and *without* outlier sample (the last one in each dataset). Then plot 1) the data points and 2) fitted lines in each subplot.
- For each of 4 training datasets, train linear regression models using weight decay coefficients $\lambda = \{0.1, 1, 10\}$ respectively and plot data points and 3 fitted lines in each subplot. Could weight decay on parameter significantly reduce the influence of outlier sample? Why or why not?
- If we model the noise in the regression models using zero-mean Laplace distribution, instead of Gaussian distribution, could we significantly reduce influence our outliers, i.e. achieve robustness to outliers? Why or why not? Could you write the learning objective function for linear regression in this case?

In real world problems, it is common to encounter a small amount of samples that are significantly different from majority of dataset, e.g. extreme weathers, 0.1% of most wealthy people, age of 99 in records of a few high school students. These samples may indicate experimental error and should be discarded, or may indicate some infrequent or extreme but valid events. In either case, these samples merit more attentions because with or without these samples, we learn very different model parameters. Universal criterion for selecting influential samples does not exist. Some frequently used measures include 1) Cook's distance, 2) leverage and 3) studentized residual.

- In linear regression model, the *leverage* score for the i^{th} data sample is defined as: $h_i := (H)_{ii}$, the i^{th} diagonal element of the hat matrix $H = X(X^\top X)^{-1}X^\top$. It is used to measure the unusualness of features x.
- In linear regression model, the *studentized residual* for the i^{th} data sample is defined as: $t_i = \frac{\hat{\epsilon}_i}{\hat{\sigma}\sqrt{1-h_i}}$, where $\hat{\epsilon}_i$ is the residual of i^{th} sample, and $\hat{\sigma}^2$ is variance of all residuals.
- In linear regression model, *Cook's distance* for the i^{th} data sample is defined as: $d_i = \frac{h_i}{1-h_i} \times \frac{t_i^2}{1+k}$ where k is number of fitted parameters (in our case $k = 2$, including the bias unit). It represents an overall affect of unusualness, which could result from unusual feature values or result from large residuals.
- For each data sample in 4 training datasets, calculate $\{h_i, t_i, d_i\}$ for each data point, select the samples with largest $\{h_i, t_i, d_i\}$ respectively and plot these points using 'cd', 'ms' and 'k>' shape & color.
- Could outlier sample in every dataset be picked out according to Cook's distance?

Details of files we provided:

(a) **demoSynLR.mat**: synthetic dataset, including four training datasets {data1, data2, data3,data4} and one testing dataset *dataTe*. Each dataset is a structure with attributes {x,y}. The 1-st column of x is bias unit, the 2-nd column of x is the feature value.

(b) We provided a template matlab code **lrTemplate.m**. For reference, in this file we provided two demo plot functions that 1) plot data points and 2 fitted lines 2) plot data points and highlight samples of highest H, T, C scores. We also declared multiple functions that handle the tasks in this problem. You can either use this template or write code from scratch.

5.2 Classification

(**25 points**) In this assignment, you will experiment with two classification algorithms on real-world datasets. You will use MATLAB's functions to experiment with Decision Tree, but you need to implement K-Nearest Neighbor (K-NN) algorithm by yourself. You are NOT allowed to use any related MATLAB toolbox functions for K-NN (e.g. `knnclassify`, `knnsearch`). Below, we describe the steps that you need to take to accomplish this programming assignment.

Dataset: We will use the *Car Evaluation Dataset* (for all classification algorithms) from UCI's machine learning data repository. The training/validation/test sets are provided along with the assignment in Blackboard as `hw1_train.data`, `hw1_valid.data`, and `hw1_test.data` for the *Car Evaluation Dataset*,. For description of the datasets, please refer to https://archive.ics.uci.edu/ml/datasets/Car+Evaluation.

Please follow the steps below:

(a) Data Pre-processing The first step in every data analysis experiment is to inspect the datasets and make sure that the data has the appropriate format. You will find that the features in the provided dataset are categorical. However, some of the algorithms require the features to be real-valued numbers. To convert a categorical feature with K categories to real-valued number, you can create K new binary features. The ith binary feature indicates whether the original feature belongs to the ith category or not. For example, if you have a feature called 'safety' with possible values 'low', 'medium', and 'high', then you will have three (3) binary features: if in a single data instance, you have a value of 'low', then your binary features will be 1-0-0; if value is 'medium', then your binary features will be 0-1-0; if value is 'high', then your binary features will be 0-0-1. About the training labels, if there are M labels in the dataset, then convert each of them into a number between 1,...,M. For example if there are 3 possible labels: 'positive', 'neutral', and 'negative', then 'positive' label will be converted to 1, 'neutral' label will be converted to 2, and 'negative' label will be converted to 3. So, for a training dataset (`*_train.data`) of size N, then `train_data` will be a matrix of size $N \times D$ (with D is the total number of binary features, as a result of this pre-processing) and `train_label` will be a column vector of dimension $N \times 1$. This also applies to validation dataset (`*_valid.data`) and testing dataset (`*_test.data`). This pre-processed matrices and vectors will be an input to part 5.2(b), 5.2(c), and 5.2(d).

Hint: One possible way of doing this pre-processing is by using MATLAB's Map Containers (http://www.mathworks.com/help/matlab/map-containers.html).

(b) Implement *K*-NN Please fill in the function `knn_classify` in `knn_classify.m` file (in Blackboard). The inputs of this function are training data, new data (either validation or testing data) and K. Use the L_2 norm distance in the K-NN algorithm. The function needs to output the accuracy on both training and new data (either validation or testing). Do not forget to standardize the data.

(c) Performance Comparison Compare the two algorithms (K-NN and Decision Tree) on the provided dataset.

***K*-NN:** Consider $K = 1, 3, 5, \cdots, 15$. For each K, report the training, validation and test accuracy. When computing the training accuracy of K-NN, we use leave-one-out strategy, i.e. classifying each training point using the remaining training points. Note that we use this strategy only for K-NN in this assignment.

Decision Tree: Train decision trees using function `ClassificationTree.fit` or `fitctree` in Matlab. Report the training, validation and test accuracy for different split criterions (*Gini index* and *cross-entropy*, using the SplitCriterion attribute), by setting the minimum number of leaf node observations to $1, 2, \cdots, 10$ (using the MinLeaf attribute). So basically, you need to report the results for $2 \times 10 = 20$ different cases. When training decision trees, please turn off pruning using the `Prune` attribute.

(e) Decision Boundary In this step, you need to apply K-NN on the `hw1boundary.mat` dataset (provided in Blackboard) which is a binary classification dataset with only two features. You need to run K-NN with $K = 1, 5, 15, 25$ and examine the decision boundary. A simple way to visualize the decision boundary, is to draw 10000 data points on a uniform 100×100 grid in the square $(x, y) \in [0, 1] \times [0, 1]$ and classify them using the K-NN classifier. Then, plot the data points with different markers corresponding to different classes. Repeat this process for all k and discuss the smoothness of the decision boundaries as K increases.

Submission Instruction: You need to provide the followings:

- Provide your answers to problems 1-4, 5.1, 5.2(c)(d) in PDF file, named as `CSCI567_hw1_spring16.pdf`. You need to submit the homework in both hard copy (at CS Front Desk with a box labeled as CSCI567-homework by 4 pm of the deadline date) and electronic version as pdf file on Blackboard. If you choose handwriting instead of typing all the answers, you will get 40% points deducted.
- Submit ALL the code and report via Blackboard. The only acceptable language is MATLAB. For your program, you MUST include the main function called `CSCI567_hw1_spring16.m` in the root of your folder. After running this main file, your program should be able to generate all of the results needed for this programming assignment, either as plots or console outputs. You can have multiple files (i.e your subfunctions), however, the only requirement is that once we unzip your folder and execute your main file, your program should execute correctly. Please double-check your program before submitting. You should only submit one `.zip` file. No other formats are allowed except `.zip` file. Also, please name it as `[lastname]_[firstname]_hw1_spring16.zip`.

Collaboration: You may collaborate. However, collaboration has to be limited to discussion only and you need to write your own solution and submit separately. You also need to list with whom you have discussed.