STAT 161/261: Homework 4
Due Saturday, June 4, noon

1. Color image segmentation using k-means. MATLAB, R or equivalent. In this problem, we will use k-means for color image segmentation. As we will see, this is not the best method for segmenting images – better methods use graph-based techniques. But, it will illustrate the basic principles of k-means.

   (a) Load the image `birds.jpg`. (Use the image load command or image reading command from your software. In MATLAB, you can use the `imread` command.) This should create an $n_x \times n_y \times 3$ matrix with the RGB color values over the $n_x \times n_y$ pixels. Plot the image. (In MATLAB, you can use `imshow`.)

   (b) Instead of having a three-dimensional matrix, convert the image to a matrix $\mathbf{X}$ of size $n_x n_y \times 3$ so that each of the $n_x n_y$ pixels are stored as a 3×1 vectors of colors. (In MATLAB, you will also need to convert this matrix from `uint8` to `double` using the `double` command since k-means will expect double precision data.)

   (c) Run k-means on the data matrix $\mathbf{X}$ with $n_c = 3$ clusters. In MATLAB, you can use the `kmeans` function. Otherwise, you will have to write a k-means routine yourself.

   (d) Create a "color-blocked" image, $\mathbf{Y}$, where the RGB values of each pixel are replaced by the RGB value of the cluster center that the pixel belongs to. Reshape this back to an $n_x \times n_y \times 3$ matrix. (In MATLAB, you will also need to round the values and convert back to `uint8`.) Use the `subplot` command to plot the original image with the 3 clusters. Redo this for $n_c = 5$ clusters.

   (e) In what ways were the image segmentations successful and in what ways were they not? What is the limitation of RGB-based image segmentation?

2. Non-parametric regression on synthetic data. Suppose that variables x and r are related by

$$r = f(x) + \epsilon, \quad f(x) = (1 - e^{-0.7x})e^{-0.3x}, \quad \epsilon \sim \mathcal{N}(0, \sigma^2), \tag{1}$$

   where $\sigma = 0.1$. We will see if non-parametric regression can "learn" the function $f(x)$ from training data.

   (a) *Plot the true response:* Plot the function $f(x)$ vs. x for $x \in [0, 10]$.

   (b) *Generate training and test data:* Generate 300 training samples and 300 test samples for the data (x_i, r_i) following the model (1). Generate the points x_i by

$$x_i = 10u_i^2, \quad u_i \sim U[0, 1],$$

   where u_i is uniformly distributed in $[0, 1]$. This will produce values $x_i \in [0, 10]$, but with points at a lower density for higher values of x_i.

(c) *Non-parametric fit with fixed h:* Consider the non-parametric estimate,

$$\widehat{f}(x) = \left[\sum_{j=1}^{N} K\left(\frac{x - x_j}{h}\right) \right]^{-1} \sum_{j=1}^{N} r_j K\left(\frac{x - x_j}{h}\right),$$

using the Gaussian Kernel $K(z) = \exp(-z^2/2)$. Compute and plot $\widehat{f}(x)$ for the 300 test samples using a bandwidth $h = 0.3$.

(d) *Optimal selection of h via cross-validation:* Try different values of $h \in [0.1, 2]$. For each value compute the residual sum of squares (RSS) between $\widehat{f}(x_i)$ and r_i on the test data and select the optimal h. What is the minimum RSS on the test data? Plot $\widehat{f}(x)$ with the optimal h?

(e) *Non-parametric fit with kNN:* Now try the estimator

$$\widehat{f}(x) = \left[\sum_{j=1}^{N} K\left(\frac{x - x_j}{d_k(x)}\right) \right]^{-1} \sum_{j=1}^{N} r_j K\left(\frac{x - x_j}{d_k(x)}\right),$$

where $d_k(x)$ is the distance from x to the k-th closest point in the training data. Compute and plot $\widehat{f}(x)$ for the 300 test samples using $k = 5$.

(f) *Optimal selection of k via cross-validation:* Find the optimal value of k using cross-validation. Plot the estimate $\widehat{f}(x)$? Is this estimate better than fixed h?

3. Perceptron / logistic classification: Consider the binary logistic classification model

$$P(C_1|\mathbf{x}) = 1 - P(C_2|\mathbf{x}) = \sigma(\mathbf{w}^T\phi(\mathbf{x})),$$

where $\sigma(z) = 1/(1 - e^{-z})$ is the sigmoidal function and $\phi(\mathbf{x})$ is a feature vector. What is the log likelihood function of $\mathbf{w}$ given data $(\mathbf{x}_i, y_i)$? What are the gradient and Hessian of the log likelihood function?

4. SVM: Suppose we are given data (x_i, r_i) for two classes, $r_i = \pm 1$, where the predictors x_i are scalars (i.e. 1-dimensional). Consider a linear classifier of the form

$$\widehat{r}_i = \text{sign}\left(x_i + w_0\right),$$

for some bias w_0.

(a) When is the data linearly separable? That is, when does there exists w_0 such that $\widehat{r}_i = r_i$ for all samples i?

(b) Now suppose that the data is not linearly separable. How we would find w_0 to minimize the maximum margin

$$L(w_0) = \sum_i \max\left\{-r_i(x_i + w_0), 0\right\}.$$

5. It is argued in Section 11.6 of Alpaydin that a two-layer neural network can provide a piecewise constant approximation for any function. Given data (x_i, y_i), $i = 1, \ldots, N$, where x_i and y_i are real numbers, describe precisely how to construct a neural network that outputs y_i for each input x_i. You can assume that the input is sorted in that

$$x_1 < x_2 < \cdots < x_N.$$

Your description should provide the number of hidden units and formulas for all the weights.

6. Alpaydin provides the backpropagation equations for a neural network with one hidden layer. Write the equations for a multilayer network used for classification and write the backpropagation equations for training this network.

7. Representation of discrete functions can demonstrate the flexibilty of multilayer perceptrons. Consider the function r on (x_1, x_2) given by the following table:

x_1	x_2	$r(x_1, x_2)$
0	0	0
0	1	0
0	2	0
1	0	0
1	1	1
1	2	0

   (a) Explain why no single-layer perceptron with inputs x_1 and x_2 can match r.

   (b) Find a two-layer perceptron model that computes r. Be explicit about which nonlinearity is applied at the output of each layer.

8. Consider a multilayer perceptron as illustrated in Fig. 11.6 of Alpaydin, where the nonlinearity applied is

$$z_h = \frac{1}{1 + \exp[-\mathbf{w}_h^T \mathbf{x}]}.$$

(The output layer y is generated linearly from the hidden layer.)

   (a) Derive the update equation for the weight w_{hj}. This was discussed in lecture and an equation is given the book; explicitly account for the differentiation of the nonlinearity.

   (b) `mlp_test_data.m` (on the course webpage) generates i.i.d. samples of $\mathbf{x} \in \mathbb{R}^3$ from some distribution. (You can read the function to interpret the distribution, but this is not essential.) Implement MLP training by backpropagation to find an *autoencoder* for $\mathbf{x}$ that uses H hidden units. (You will need to be able to vary H in the next part.) You will have to choose initializations and learning rates. One possibility is to initialize the $\mathbf{W}$ and $\mathbf{V}$ weight matrices with `randn`, but experiment also with other initialization.

   (c) For $H = 2$, train your autoencoder using 10000 samples. Using the same function `mlp_test_data.m` to draw samples, evaluate the generalization error as the average of $\|\mathbf{x} - \mathbf{y}\|_2^2$ on 1000 samples. Plot the $\mathbf{x}$ samples and their approximations $\mathbf{y}$ together as point clouds.

   (d) Repeat the previous part, varying $H \in \{1, 2, \ldots, 7\}$. Comment on how the apparent fit and average squared error vary.