STATS C161/C261: Homework 3
Due May 23 at 4pm (brought to class or submitted electronically on CCLE

Send questions regarding problem submission to Rosie Jia (ruoxuan@ucla.edu).

1. *Kernel density estimation (KDE).* You will use KDE on synthetic data to observe the effect of window width. Use Python or any language of your choice. An example using Python's built-in KDE is here:

   `http://scikit-learn.org/stable/auto_examples/neighbors/plot_kde_1d.html`

   (a) Generate 100 samples, x_i, from the two-component mixture distribution

   $$p(x) = \sum_{k=1}^{2} \frac{q_k}{\sqrt{2\pi}\sigma} e^{-(x-\mu_k)^2/(2\sigma^2)}, \tag{1}$$

   where

   $$q_1 = 0.6, \quad q_2 = 0.4, \quad \mu_1 = -2, \quad \mu_2 = 3, \quad \text{and} \quad \sigma = 1.5.$$

   Under the density (1), the samples x_i are from a mixture of two scalar Gaussians with the same variance.

   (b) Using the samples from part (a), compute a density estimate $\hat{p}(x)$ using the Gaussian kernel with window width $h = 0.5$. Plot $\hat{p}(x)$ and $p(x)$ together. You can use any built-in KDE estimation routine in whichever language you are using.

   (c) Re-do parts (a) and (b) with $h = 2$ and $h = 0.1$. Which estimate is over-fit and which is under-fit?

2. Suppose that a logistic regression model for a binary class label $y = 0, 1$ is given by

   $$P(y = 1|\mathbf{x}) = \frac{1}{1 + e^{-z}}, \quad z = \beta_0 + \beta_1 x_1 + \beta_2 x_2,$$

   where $\boldsymbol{\beta} = [1,\, 2,\, 3]^\mathsf{T}$. Describe the following sets:

   (a) The set of $\mathbf{x}$ such that $P(y = 1|\mathbf{x}) > P(y = 0|\mathbf{x})$.

   (b) The set of $\mathbf{x}$ such that $P(y = 1|\mathbf{x}) > 0.8$.

   (c) The set of x_1 such that $P(y = 1|\mathbf{x}) > 0.8$ and $x_2 = 0.5$.

3. *K nearest neighbors (KNN).* You will use KNN on synthetic data for classification. Use Python or the language of your choice; you may use any built-in commands that are available. An example using Python is here:

   `http://scikit-learn.org/stable/auto_examples/ensemble/plot_voting_decision_regions.html`

   The synthetic data will be of the form $(\mathbf{x}, y)$ where $\mathbf{x} \in \mathbb{R}^2$ and $y = 0, 1, 2$. So the data has three classes and $\mathbf{x}$ has dimension $d = 2$. For each class $j = 0, 1, 2$, the data $\mathbf{x}$ is Gaussian with mean $\boldsymbol{\mu}_j$ and covariance $\mathbf{S}_j$ given by

$$\boldsymbol{\mu}_0 = [0, 0], \quad \mathbf{S}_0 = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix},$$

$$\boldsymbol{\mu}_1 = [1, 2], \quad \mathbf{S}_1 = \begin{bmatrix} 0.5 & 0 \\ 0 & 1 \end{bmatrix},$$

$$\boldsymbol{\mu}_2 = [1.5, 0], \quad \mathbf{S}_2 = \begin{bmatrix} 0.5 & -0.25 \\ -0.25 & 0.5 \end{bmatrix}.$$

   (a) Generate 900 samples, 300 samples from each class. Plot the generated samples on a scatter plot with different markers for each class.
   In Python, you can use `np.random.multivariate_normal` to generate the samples in each class.

   (b) Shuffle the data and split into 60% (540) training and 40% (360) test samples. For numbers of neighbors $k = 2, 5, 15$:

   - Fit a KNN classifier on the training data. (We will ignore the test data for now.)
   - Compute the predicted class labels on a grid of points $\mathbf{x} = (x_0, x_1)$ with $x_0 \in [-3, 3]$ and $x_1 \in [-2, 3]$.
   - Use the grid of points to plot the decision regions for each class.

   You should see that as k increases, the decision regions become more smooth. In Python, you can use the `np.meshgrid` and `plt.contourf` commands.

   (c) Use cross-validation to find the optimal k. For values of $k = 2$ to 100, fit a model for each k on the training data and plot the test error. What k results in the minimum test error? (Note test error results may be quite "noisy," i.e. change significantly with the data shuffling. To get more consistent results, one could use K-fold validation, but you do not need to do that here.)

4. The loss function for logistic regression is the binary cross entropy defined as

$$J(\boldsymbol{\beta}) = \sum_{i=1}^{N} \ln(1 + e^{z_i}) - y_i z_i,$$

   where $z_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i}$ for two features $\mathbf{x}_1$ and $\mathbf{x}_2$.

   (a) What are the partial derivatives of z_i with respect to β_0, β_1, and β_2?

   (b) Compute the partial derivatives of $J(\boldsymbol{\beta})$ with respect to β_0, β_1, and β_2. You should use the chain rule of differentiation. (Think about whether there are closed-form solutions for the optimal parameters. We will discuss that in class.)

5. Data scientists are hired by a politician to predict who will donate money. They decide to use two predictors for each possible donor:

   - x_1 = the person's income (in thousands of dollars per year); and
   - x_2 = the number of similar politicians the person follows on Twitter.

   To train the model, they try to get donations from a random subset of people and record who donates. The following data is obtained:

Income x_{i1}	30	50	70	80	100
Twitter follows x_{i2}	0	1	1	2	1
Donate (1=yes or 0=no), y_i	0	1	0	1	1

   (a) Draw a scatter plot of the data labeling the two classes with different markers.

   (b) Find a linear classifier that makes at most one error on the training data. The classifier should be of the form,

$$\hat{y}_i = \begin{cases} 1 & \text{if } z_i > 0 \\ 0 & \text{if } z_i < 0, \end{cases} \qquad z_i = \mathbf{w}^\mathsf{T}\mathbf{x}_i + b$$

   What are the weight vector $\mathbf{w}$ and bias b in your classifier?

   (c) Consider a logistic model of the form:

$$P(y_i = 1|\mathbf{x}_i) = \frac{1}{1 + e^{-z_i}}, \qquad z_i = \mathbf{w}^\mathsf{T}\mathbf{x}_i + b.$$

   Using $\mathbf{w}$ and b from the previous part, which sample i is the *least* likely (i.e. $P(y_i|\mathbf{x}_i)$ is the smallest)? If you do the calculations correctly, you should not need a calculator.

   (d) Consider a new set of parameters:

$$\mathbf{w}' = \alpha\mathbf{w}, \quad b' = \alpha b,$$

   where $\alpha > 0$ is a positive scalar. Would using the new parameters change the values $\hat{y}$ in part (b)? Would they change the likelihoods $P(y_i|\mathbf{x}_i)$ in part (c)? If they do not change, state why. If they do change, qualitatively describe the change as a function of α.

6. *Color image segmentation using k-means.* (Note: this is not the best method for segmenting images, but it illustrates k-means.) Use Python or any language.

   (a) Load and plot the image `birds.jpg`. The image is on CCLE.

   (b) Instead of having a 3-dimensional array of size $n_x \times n_y \times 3$ for the two image dimensions and three color channels, 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 is stored as a 3×1 vector of intensities for the three color components. (In Python and MATLAB, k-means will expect double precision data. So you will need to convert this matrix from `uint8` to floating point.)

   (c) Run k-means on the data matrix $\mathbf{X}$ with $n_c = 3$ clusters. You can use any built-in command. You do not have to write the algorithm from scratch.

   (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 Python and MATLAB, you will also need to round the values and convert back to `uint8`.) Use the `subplot` command to plot the original image alongside the color-blocked version. Redo this for $n_c = 5$ clusters.

   (e) In what ways were the image segmentations successful and in what ways were they not?