In the first two questions, we use multinomial logistic regression models to predict one of $K$ discrete classes. Let $t_{nk} = 1$ if observation $n$ is an instance of class $k$, and $t_{nk} = 0$ otherwise. Given input variable $x_n$, let $\phi(x_n) \in \mathbb{R}^M$ be a fixed feature function. The multinomial logistic regression model then defines the conditional probability of class $k$ as follows:

$$p(t_{nk} = 1 \mid x_n, w) = \frac{\exp(w_k^T \phi(x_n))}{\sum_{\ell=1}^K \exp(w_\ell^T \phi(x_n))}.$$ 

Here $w_k \in \mathbb{R}^M$ are the feature weights associated with class $k$, and $w \in \mathbb{R}^{KM}$ is a vector concatenating the weights for all classes.

This model is not identifiable: coordinated translations of the weight vectors $w_k$ for different classes can produce identical probabilities. To avoid this ambiguity and improve robustness, we place a zero-mean, diagonal-covariance Gaussian prior on the weight vector:

$$p(w) = \mathcal{N}(w \mid 0, \alpha^{-1} I_{KM}) \propto \exp\left(-\frac{\alpha}{2} w^T w\right).$$

Here, $\alpha$ is a tunable inverse-variance parameter that controls the degree of regularization. The regularized negative log conditional likelihood, or equivalently the objective whose minimum equals the MAP estimate of the weight vector $w$, is then

$$f(w) = -\log p(w) - \sum_{n=1}^N \log p(t_n \mid x_n, w)$$

$$= \frac{\alpha}{2} w^T w - \sum_{n=1}^N \left[ t_{nk} w_k^T \phi(x_n) - \log \left( \sum_{\ell=1}^K \exp(w_\ell^T \phi(x_n)) \right) \right],$$

ignoring constants which do not depend on $w$. 
Question 1:

a) Give an equation for the partial derivative \( \frac{\partial f(w)}{\partial w_{km}} \) of the regularized negative log conditional likelihood \( f(w) \) with respect to a particular feature weight \( w_{km} \). This derivative is one entry of the gradient vector \( \nabla f(w) \).

We reexamine the gamma ray data set from homework 3, but instead apply a logistic regression model for the binary classification of star showers. We have split the previous training set so that there are now 11,413 training examples, 3,804 development examples, and 3,804 test examples. Their class labels are stored in column vectors named \texttt{trainLabels}, \texttt{devLabels}, \texttt{testLabels}. We compare the performance of three different feature mappings of the \( D = 10 \) raw inputs. In all cases, the first feature should be a constant bias or offset term, \( \phi_0(x_n) = 1 \). The three feature sets are then:

1. \( D + 1 \) linear features, the bias feature plus the raw inputs \( \phi_m(x_n) = x_{nm}, 1 \leq m \leq D \).
2. \( 2D + 1 \) diagonal quadratic features, including the \( D + 1 \) linear features from set 1, as well as \( D \) quadratic features \( \phi_{D+m}(x_n) = x_{nm}^2, 1 \leq m \leq D \).
3. \((D + 1)D/2 + D + 1\) general quadratic features, including the \( 2D + 1 \) features from set 2, as well as products of all pairs of input dimensions \( x_{nm}, x_{nm}', m \neq m' \).

Given this data and the objective from part (a), we will use the \texttt{minFunc} function from the \texttt{pmtk3} Matlab toolbox to find the weight vector \( w \) that minimizes the regularized negative log conditional likelihood \( f(w) \). You should provide the optimizer with the gradient of the regularized negative log conditional likelihood, and write your own function to compute the objective and its gradient. To get started with \texttt{minFunc}, see the sample code released with the homework, which includes suggestions for appropriate convergence tolerance parameters.

For all of the following questions, do all calculations for 10 logarithmically-spaced values of \( \alpha \) between \( 10^{-8} \) and 10, and test each of the three feature sets above.

b) Plot the negative log conditional likelihood of the \texttt{train} data as a function of \( \alpha \) when training and evaluating on \texttt{train}. Also plot the accuracy of the classifier as a function of \( \alpha \) when training and evaluating on \texttt{train}.

c) Plot the negative log conditional likelihood of the \texttt{dev} data as a function of \( \alpha \). That is, for each value of \( \alpha \) estimate the feature weight \( w \) from the \texttt{train} data, and then calculate the negative log conditional likelihood of the \texttt{dev} data for that value of \( w \). Find a value of \( \alpha \) which minimizes the negative log conditional likelihood of the \texttt{dev} data. Does the learning output seem sensitive to the value of \( \alpha \)? Provide an explanation for your observations.

d) For each of the three feature sets, and the weight vectors \( w \) corresponding to the values of \( \alpha \) identified in part (c), evaluate and report the \texttt{test} accuracy of the corresponding classifiers. How do these accuracies compare to the Gaussian naive Bayes classifier from homework 3?
Question 2:

This question uses synthetic data to compare the properties of logistic regression and linear regression for classification. Each “toy” data item has two continuous features \( x \in \mathbb{R}^2 \) and is labeled as one of either \( K = 2 \) or \( K = 3 \) classes.

Your linear regression code from homework 4 can be adapted for classification as follows. For \( K \) classes, each response is encoded as a row vector \( t_n = [t_{n1}, \ldots, t_{nK}] \), where \( t_{nk} = 1 \) for an example of class \( k \), and zero otherwise. For \( N \) data samples we define the \( N \times K \) matrix \( T \) as a matrix of 0’s and 1’s, with each row having a single 1. We fit a linear regression model to each of the columns of \( T \) as follows:

\[
\hat{T} = \Phi(\Phi^T\Phi)^{-1}\Phi^T T
\]

Here, \( \Phi \) is the \( N \times 3 \) model matrix of corresponding to the feature function \( \phi(x_n) = [1, x_{n1}, x_{n2}]^T \), i.e. the raw 2D input data augmented by a constant bias feature. The weights corresponding to the least squares prediction above equal

\[
\hat{W} = (\Phi^T\Phi)^{-1}\Phi^T T
\]

Here, \( \hat{W} \) is a \( 3 \times K \) matrix where the \( k^{th} \) column \( \hat{w}_k \) represents the linear regression fit for class \( k \). Finally, we can use this linear regression model to classify a new observation as

\[
y(x) = \arg \max_k \phi(x)^T \hat{w}_k.
\]

The supplied function \texttt{plotClassifier} can be used to visualize decision boundaries.

We compare the performance of this linear least squares classifier to a multinomial logistic regression classifier, both using the same features. To fit multinomial logistic regression models, use your implementation from question 1, with a small but nonzero regularization constant \( \alpha = 10^{-8} \) to ensure identifiability.

a) The first dataset contains two classes which lie in well-separated clouds. Implement the least squares classifier described above. Estimate weights from training data, and plot the learned decision boundary together with the training points. If implemented correctly, your test accuracy should be 100%. Is this the case?

b) The second dataset contains three classes with means arranged in a triangular pattern. Train a least squares classifier as above, as well as a multinomial logistic regression classifier using the same features. Plot the training decision boundaries for both classifiers, and report test accuracy for each. Explain any performance differences.

c) The third dataset contains three classes with means arranged in a straight line. Train a least squares classifier as above, as well as a multinomial logistic regression classifier using the same features. Plot the training decision boundaries for both classifiers, and report test accuracy for each. Explain any performance differences.
Question 3

We now consider a binary categorization problem, where \( t_n \in \{0, 1\} \) is the output label for example \( n \), and \( x_n \in \mathbb{R}^2 \) is a two-dimensional vector of input features. Assume that the two classes are equally likely \textit{a priori}, so that \( p(t_n) = \text{Ber}(t_n \mid 0.5) \). Under the true data generation process, the features are distributed according to class-specific Gaussians:

\[
p(x_n \mid t_n = 1) = \mathcal{N}(x_n \mid \mu_1, \Sigma), \quad p(x_n \mid t_n = 0) = \mathcal{N}(x_n \mid \mu_0, \Sigma).
\]

The mean vectors \( \mu_1, \mu_0 \) are discussed below. The shared covariance matrix equals:

\[
\Sigma = \frac{4}{3} \begin{bmatrix} 1 & 1/2 \\ 1/2 & 1 \end{bmatrix} = \begin{bmatrix} 4/3 & 2/3 \\ 2/3 & 4/3 \end{bmatrix}, \quad \Sigma^{-1} = \begin{bmatrix} 1 & -1/2 \\ -1/2 & 1 \end{bmatrix}.
\]

(a) Suppose that \( \mu_0 = [0, 0]^T, \mu_1 = [2, 0]^T \). Given knowledge of the true joint distribution \( p(x_n, t_n) \), derive a classification rule \( y(x_n) \) which minimizes the probability of error. Plot the corresponding decision boundary graphically.

(b) Suppose that \( \mu_0 = [0, 0]^T, \mu_1 = [2, 2]^T \). Given knowledge of the true joint distribution \( p(x_n, t_n) \), derive a classification rule \( y(x_n) \) which minimizes the probability of error. Plot the corresponding decision boundary graphically.

Now suppose that we do not have knowledge of the true data generating process, but instead assume a naive Bayes model with Gaussian features:

\[
\begin{align*}
p(x_n \mid t_n = 1) &= \mathcal{N}(x_{n1} \mid \theta_{11}, \nu_{11}) \mathcal{N}(x_{n2} \mid \theta_{12}, \nu_{12}), \\
p(x_n \mid t_n = 0) &= \mathcal{N}(x_{n1} \mid \theta_{01}, \nu_{01}) \mathcal{N}(x_{n2} \mid \theta_{02}, \nu_{02}).
\end{align*}
\]

Consider a training dataset with \( N \) observations \((x_n, t_n)\) independently sampled from the true joint distribution \( p(x, t) \). In each question below, assume that the parameters \( \theta \) and \( \nu \) of the naive Bayes model are estimated via the maximum likelihood (ML) criterion.

(c) Suppose that \( \mu_0 = [0, 0]^T, \mu_1 = [2, 0]^T \). As \( N \to \infty \), what classification rule will the naive Bayes classifier approach? Will it be as accurate as the optimal rule from part (a)? Justify your answer and plot the corresponding decision boundary graphically.

(d) Suppose that \( \mu_0 = [0, 0]^T, \mu_1 = [2, 2]^T \). As \( N \to \infty \), what classification rule will the naive Bayes classifier approach? Will it be as accurate as the optimal rule from part (b)? Justify your answer and plot the corresponding decision boundary graphically.