Pattern Recognition
F. Meriaudeau

1- Logistic regression-classifier (borrowed from Andrew Nd machine Learning Course)

For this exercise, suppose that a high school has a dataset representing 40 students who were admitted to college and 40 students who were not admitted. Each $(x^{(i)}, y^{(i)})$ training example contains a student's score on two standardized exams and a label of whether the student was admitted.

Your task is to build a binary classification model that estimates college admission chances based on a student's scores on two exams. In your training data,

a. The first column of your x array represents all Test 1 scores, and the second column represents all Test 2 scores.

b. The y vector uses '1' to label a student who was admitted and '0' to label a student who was not admitted.

Plot the data

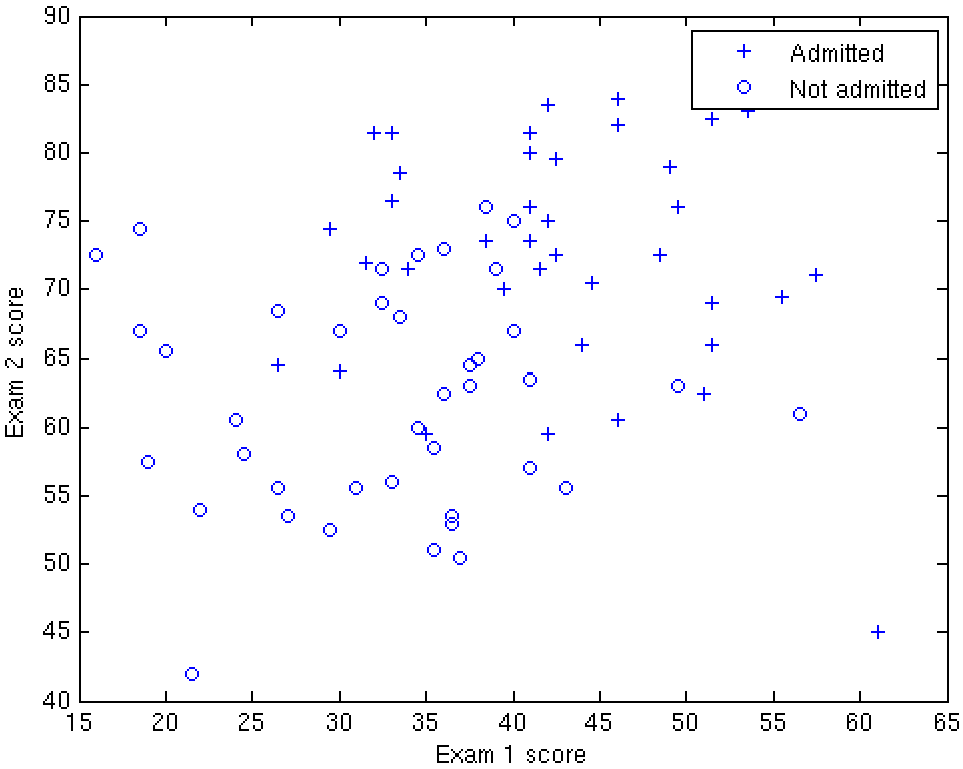
Load the data for the training examples into your program and add the $x_0 = 1$ intercept term into your x matrix.

Before beginning Newton's Method, we will first plot the data using different symbols to represent the two classes. In Matlab/Octave, you can separate the positive class and the negative class using the find command:

```
% find returns the indices of the
% rows meeting the specified condition
pos = find(y == 1); neg = find(y == 0);

% Assume the features are in the 2nd and 3rd
% columns of x
plot(x(pos, 2), x(pos,3), '+'); hold on
plot(x(neg, 2), x(neg, 3), 'o')
```

Your plot should look like the following:



Newton's Method

Recall that in logistic regression, the hypothesis function is

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1+e^{-\theta^T x}}$$

$$= P(y=1|x;\theta)$$

In our example, the hypothesis is interpreted as the probability that a driver will be accident-free, given the values of the features in x.

Matlab/Octave does not have a library function for the sigmoid, so you will have to define it yourself. The easiest way to do this is through an inline expression:

```
g = inline('1.0 ./ (1.0 + exp(-z))');
% Usage: To find the value of the sigmoid
% evaluated at 2, call g(2)
```

The cost function $J(\theta)$ is defined as

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

Our goal is to use Newton's method to minimize this function. Recall that the update rule for Newton's method is

$$\theta^{(t+1)} = \theta^{(t)} - H^{-1} \nabla_\theta J$$

In logistic regression, the gradient and the Hessian are

$$\nabla_\theta J = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x^{(i)}$$

$$H = \frac{1}{m} \sum_{i=1}^{m} \left[h_\theta(x^{(i)}) \left(1 - h_\theta(x^{(i)})\right) x^{(i)} \left(x^{(i)}\right)^T \right]$$

Note that the formulas presented above are the vectorized versions. Specifically, this means that $x^{(i)} \in R^{n+1}$, $x^{(i)} \left(x^{(i)}\right)^T \in R^{(n+1)\times(n+1)}$, while $h_\theta(x^{(i)})$ and $y^{(i)}$ are scalars.

Implementation

Now, implement Newton's Method in your program, starting with the initial value of

$\theta = \vec{0}$. To determine how many iterations to use, calculate J(θ) for each iteration and plot your results.

Newton's method often converges in 5-15 iterations. If you find yourself using far more iterations, you should check for errors in your implementation.

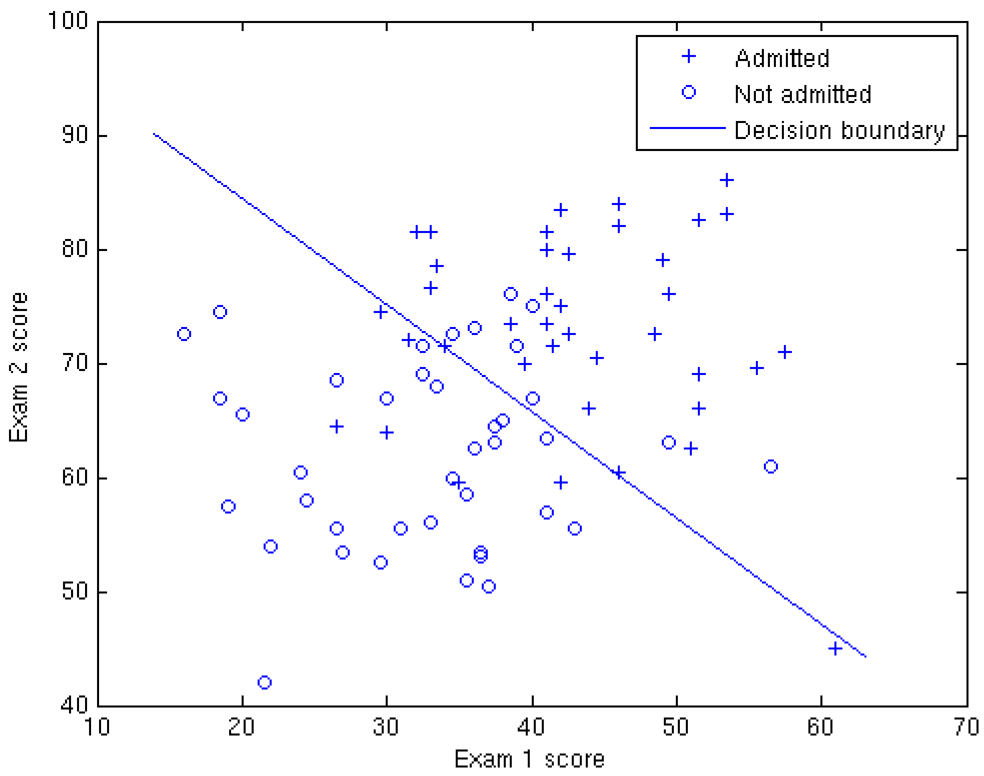
After convergence, use your values of theta to find the decision boundary in the classification problem. The decision boundary is defined as the line where

$$P(y = 1 | x; \theta) = g(\theta^T x) = 0.5$$

which corresponds to

$$\theta^T x = 0$$

Plotting the decision boundary is equivalent to plotting the $\theta^T x = 0$ line. When you are finished, your plot should appear like the figure below.



Questions

Finally, record your answers to these questions.

1. What values of θ did you get? How many iterations were required for convergence?

2. What is the probability that a student with a score of 20 on Exam 1 and a score of 80 on Exam 2 will not be admitted?

2. Dimensionality reduction.

Implement PCA and LDA in MATLAB. Apply your code to the 2D, two-class MATLAB data provided assuming equal priors. Please read carefully the last two parts of this exercise before you start.

(a) Calculate the (maximum likelihood) mean vectors and covariance matrices of each class.

(b) Use PCA to compute a 1D subspace and project the data onto it. Plot histograms of both classes on the same axes. Does this projection do a good job of separating the two classes?

(c) Use Fisher's linear discriminant (LDA) to find the vector w that optimally separates the two classes. Project the data onto this 1D subspace and plot histograms of the results for each class on the same axes. Does this projection do a good job of separating the two classes?
(d) Describe in no more than three sentences the differences between PCA and LDA.

3- Non-parametric Methods, ML estimation and KNN Classifier

Note: You will use Matlab functions provided with this tutorial.

A) Generate N=1000 data points lying in the real axis, $x_i \in R, i = 1,2, \dots \dots . N$ from the following pdf and plot p(x):

$$p(x) = \frac{1}{3}\frac{1}{\sqrt{2\pi\sigma_1^2}}\exp\left(\frac{-x^2}{2\sigma_1^2}\right) + \frac{2}{3}\frac{1}{\sqrt{2\pi\sigma_2^2}}\exp\left(\frac{-(x-2)^2}{2\sigma_2^2}\right)$$

Where $\sigma_1^2 = \sigma_2^2 = 0{,}2$

Use the Parzen windows approximation of :

$$p(x) \approx \frac{1}{N}\sum_{i=1}^{N}\frac{1}{(2\pi)^{\frac{l}{2}}h^l}\exp\left(-\frac{(x-x_i)^T(x-x_i)}{2h^2}\right)$$

with h=0,1 and **plot** the obtained estimate.

Hint: you will be using the functions generate_gauss-class() and Parzen_gauss_kernel() which are provided.

Repeat the experiment with h=0.01 , N=1000 and h=0.1, N=10000.

B) Consider the latest data from A) use the k-nearest neighbour density estimator to estimate the required pdf with k=21.
Hint: Use the function knn_density_estimate().

C) Consider a 2-dimensionnal classification problem where the data vectors stem two equiprobable classes, y_1 and y_2. The classes are modelled by Gaussian distributions with means $m_1 = [0,0]^T$ et $m_2= [1,2]^T$ and respective covariance matrices:

$$S_1 = S_2 = \begin{bmatrix} 0.8 & 0.2 \\ 0.2 & 0.8 \end{bmatrix}$$

Generate two data sets X_1 and X_2 consisting of 1000 and 5000 points, respectively. Taking X_1 as the training set, classify the points in X_2 using the k-NN classifier, with k=3 and adopting the squared Euclidian distance.

Compute the classification error.
Hint: Use the function k_nn_classifier().

D) Repeat example C) with k=1,7,15.
And compare the obtained results with the optimal Bayesian Classifier.
Hint: Use the function comp_gauss_dens_val().

E) Generate 50 2-dimensional feature vectors from a Gaussian distribution, where

m_1= $[2,-2]^T$ and $S = \begin{bmatrix} 0.9 & 0.2 \\ 0.2 & 0.3 \end{bmatrix}$
Repeat for N=500 points and N=5000 points.
Hint: Use the function Gaussian_ML_estimate().