Assignment 2: Polynomial Fitting and Logistic Regression

October 16, 2017

1 Least Squares Polynomial Fitting (0.9 points)

1.1 Introduction

As discussed during Week 2, one can use linear regression to fit a nonlinear function to data by employing a nonlinear embedding function $\phi : \Re^D \to \Re^K$. One can then express a nonlinear function in terms of an inner product as follows:

$$f(\mathbf{x}) = \langle \mathbf{w}, \phi(\mathbf{x}) \rangle$$

In this assignment we will consider the case of an embedding function that takes a point $x \in R$ to a K-dimensional space using the following expression:

$$\phi_K(x) = \begin{bmatrix} 1 \\ x \\ x^2 \\ \vdots \\ x^K \end{bmatrix}$$

We will be treating K as a hyper-parameter, i.e. we will not try to estimate K, but will rather explore the effect of changing its values.

Using this embedding function allows us to express a K-th order polynomial in terms of an inner product:

$$f_K(x) = \langle \mathbf{w}_K^T \phi_K(x) \rangle = w_0 + w_1 x + w_2 x^2 + \ldots + x_K x^K$$

If the quality of the approximation on a set of points is measured in terms of the quadratic loss:

$$E(f, \mathcal{S}) = \sum_i (y_i - f_K(x_i))^2 \tag{1}$$

we can then find the optimal parameters $\mathbf{w}_K$ of f_K using the least squares formula derived in class.

1.2 Assignments

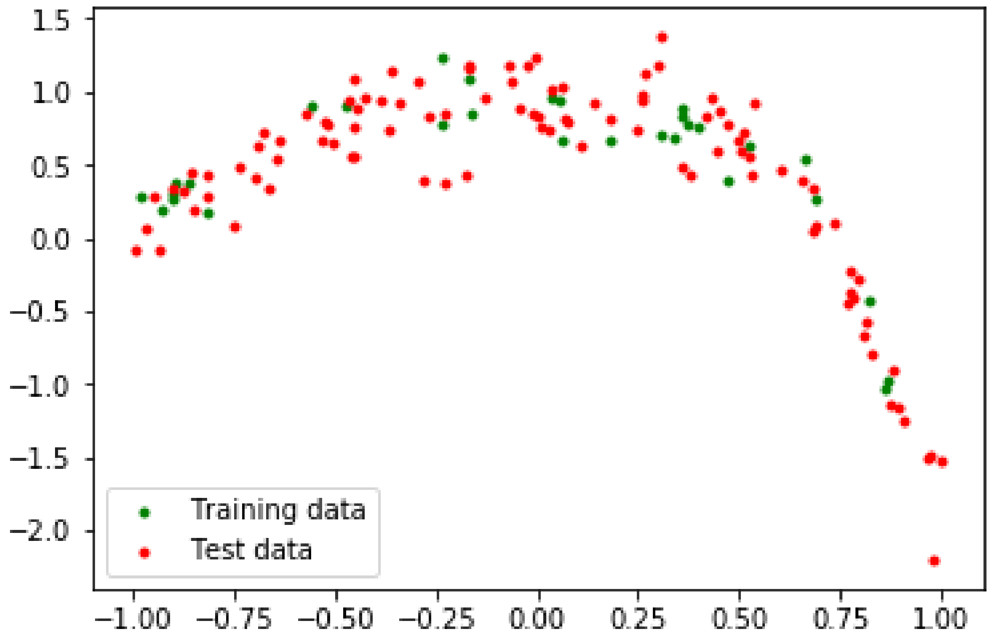
The function `pickle.load` loads the data set from the file `data_pol_fit.pkl`, we then display the training and test sets.

This notably provides you with a training set of input-output pairs $\mathcal{S} = \{(x_i, y_i)\}$ where $x_i \in \Re, y_i \in \Re, \quad i = 1 \ldots 20$. Your task is to estimate a function $f : \Re \to \Re$ that can approximate the underlying input-output mapping well.

1. (0.4) Find the optimal parameters $\mathbf{w}_0, \mathbf{w}_1, \ldots, \mathbf{w}_{10}$ if $f_K(x)$ is a polynomial of degree 0 up to 10, respectively.
2. (0.1) Plot the estimated functions within the interval $[-1, 1]$.
3. (0.4) Plot the value of the loss as a function of the polynomial's order

(a) on the training set and the (b) test set. Plot the two functions together. What do you observe? Do you have a possible interpretation of this result?

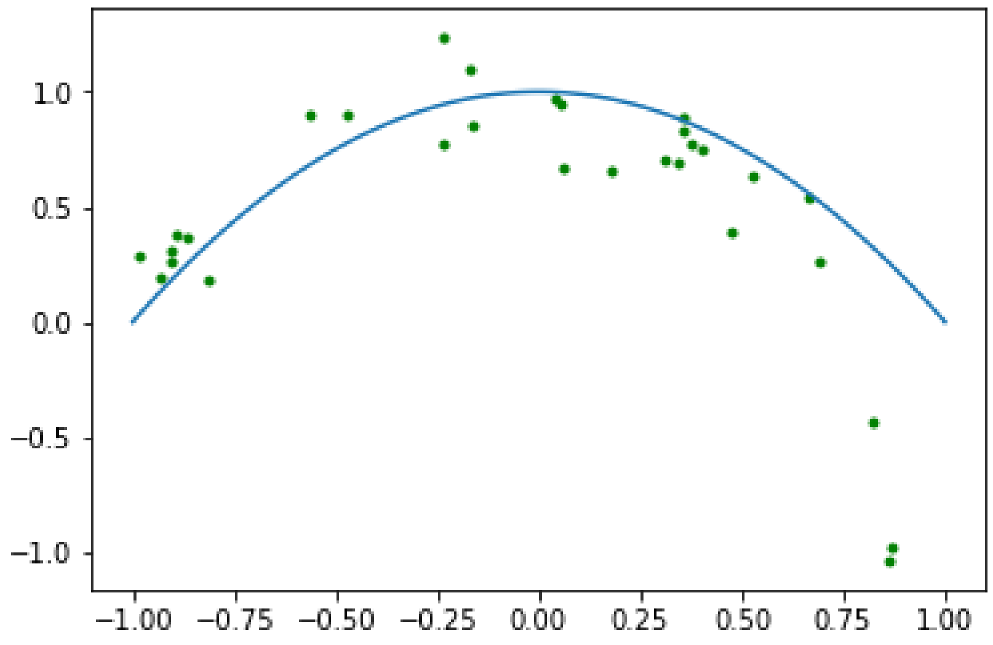
Import test and training data from 1Ddata.mat



Define function poly

Poly should take a vector value x and a K+1 dimensional vector w as arguments and return the value of the K-th order polynomial:

$$f_K(x) = \langle \mathbf{w}_K^T \phi_K(x) \rangle = w_0 + w_1 x + w_2 x^2 + \ldots + x_K x^K$$



Define embedding function

Now we need to define an embedding function that takes a point (or a vector of points) $x \in R$ to a K-dimensional space using the following expression:

$$\phi_K(x) = \begin{bmatrix} 1 \\ x \\ x^2 \\ \dots \\ x^K \end{bmatrix}$$

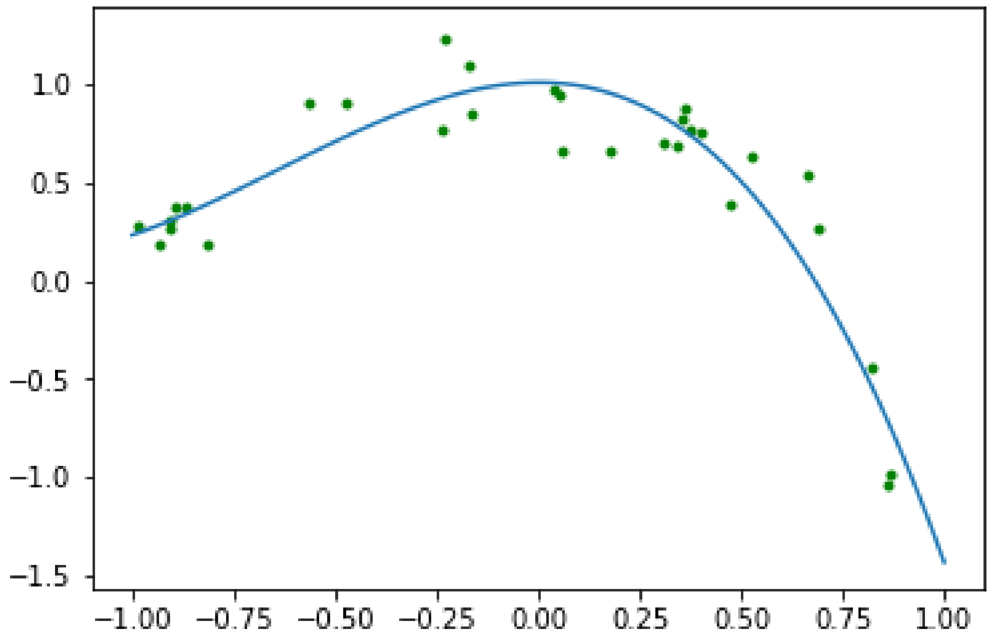
K should be an argument to the function. Keep in mind that we should also be able to apply the function to vectors. That means, if we have N data samples $x_1, x_2, \dots, x_N$ and feed vector $\vec{x} = [x_1, x_2, \dots, x_N]^T$ to the function then the output should be the following array:

$$\phi_K(\vec{x}) = \begin{bmatrix} 1 & x_1 & x_1^2 & \dots & x_1^K \\ 1 & x_2 & x_2^2 & \dots & x_2^K \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_N & x_N^2 & \dots & x_N^K \end{bmatrix}$$

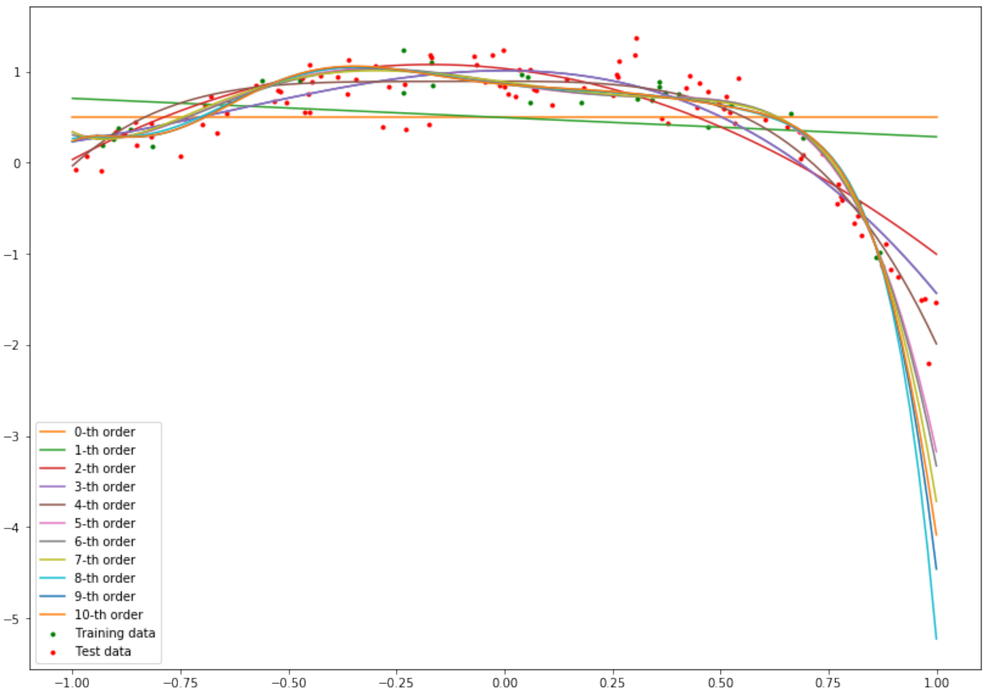
```
[[  1.   2.   4.   8.  16.]
 [  1.   3.   9.  27.  81.]]
```

Define function that fits K-th order polynomial and returns parameter w

Input arguments for the function are [Nx1] vector y and [Nx1] vector x. Hint: In python 3.6 you can write A@B to do matrix multiplication instead of np.matmul(A,B).



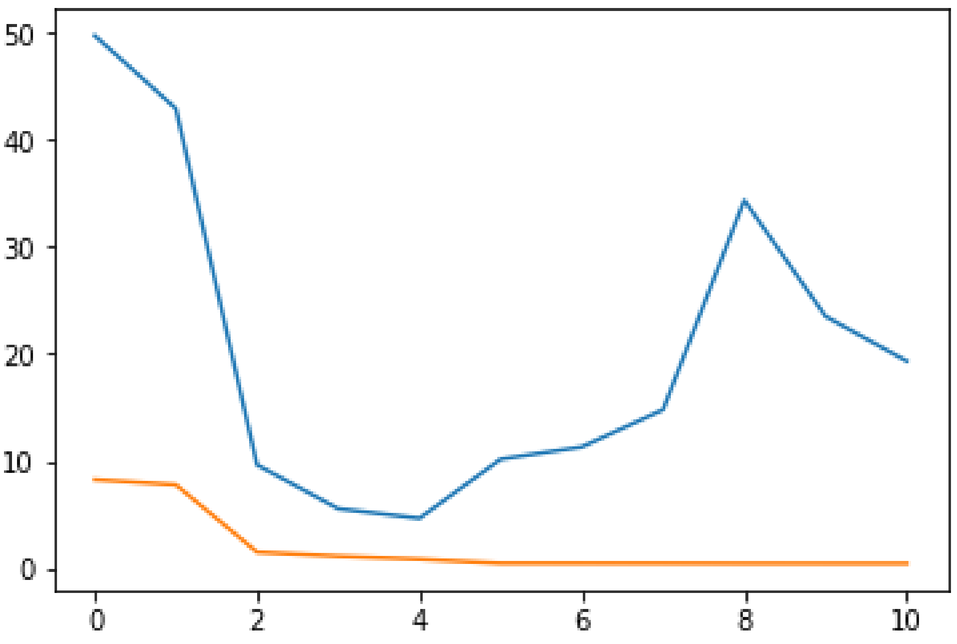
Fit 0-th to 10-th order polynomials to training data and plot resulting functions



Training and Testing Error

Calculate the training and the testing error given:

$$E(f, \mathcal{S}) = \sum_i (y_i - f_K(x_i))^2 \tag{2}$$



2 Linear versus logistic regression (0.7 points)}

The file {week_1} contains code to compare the robustness of linear and logistic regression. Linear regression has already been implemented for you. Please go through the relevant part of the code, and the relevant comments, before advancing to the remainder.

As discussed in class, the loss function (or, criterion, L) driving logistic regression equals:

$$L(\mathbf{w}) = -\sum_{i=1}^{N} \left[y^i \log \left(g \left(\left\langle \mathbf{x}^i, \mathbf{w} \right\rangle \right) \right) + (1 - y^i) \log \left(1 - g \left(\left\langle \mathbf{x}^i, \mathbf{w} \right\rangle \right) \right) \right]. \tag{3}$$

Maximize this criterion for the provided dataset. Making reference to the slides of Week 3, this requires computing first and second order differentials, corresponding to $\nabla L(\mathbf{w})$ and $\nabla^2 L(\mathbf{w})$, respectively.

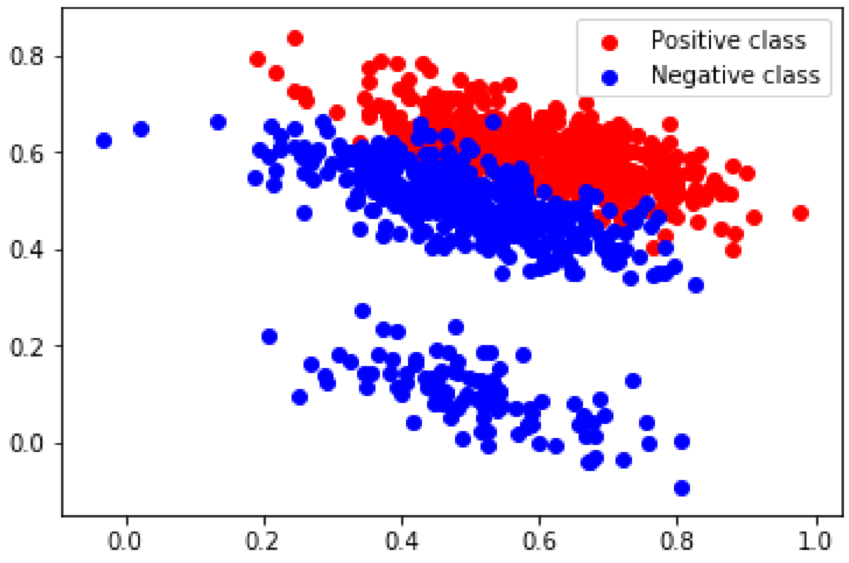
1. (.2/.7) Write the code to compute $\nabla L(\mathbf{w})$ and $\nabla^2 L(\mathbf{w})$.

2. (.2/.7) Use these to implement the Newton-Raphson algorithm. Consider that convergence is achieved when $|w^t - w^{t-1}|_2 < .001|w^t|_2$

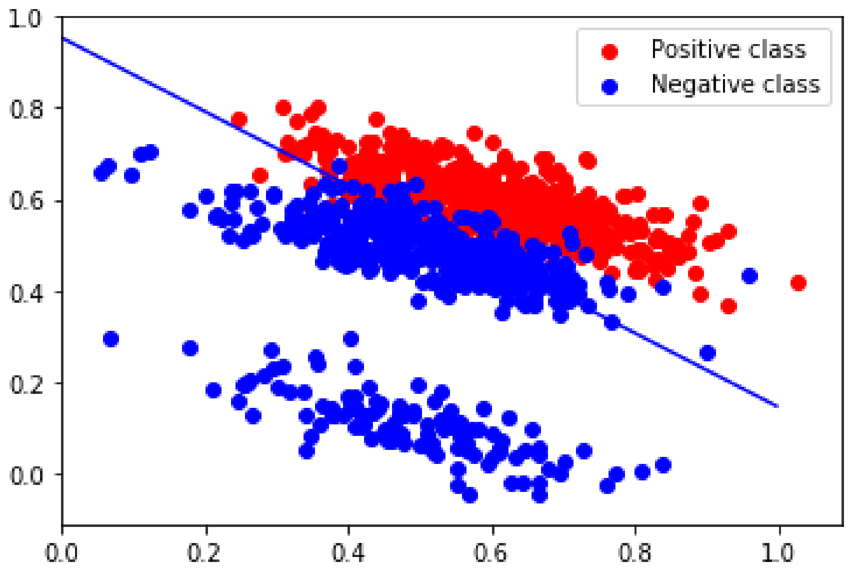
3. (.2/.7) Plot the loss function, $L(w)$, of Eq. 1 as a function of Newton-Raphson iteration.

4. (.1/.7) Compare the robustness of the fitted boundaries for the two experiments.

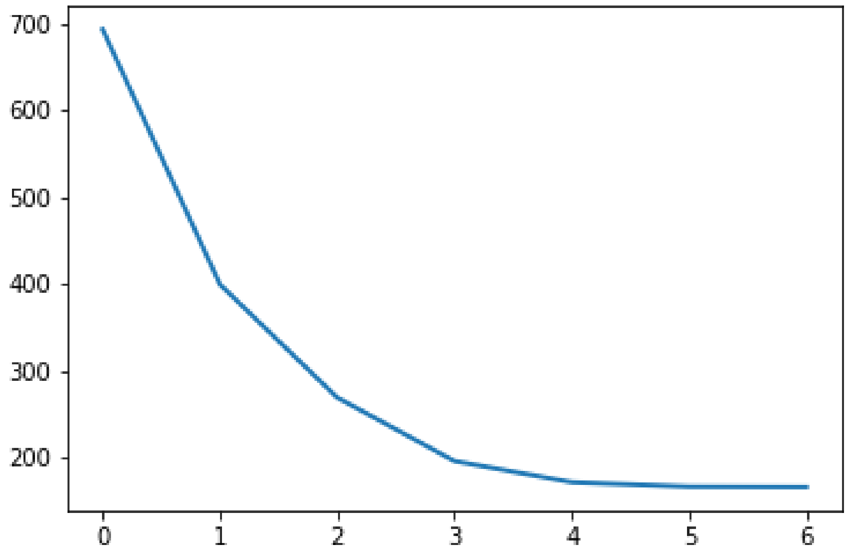
Data generation and visualization



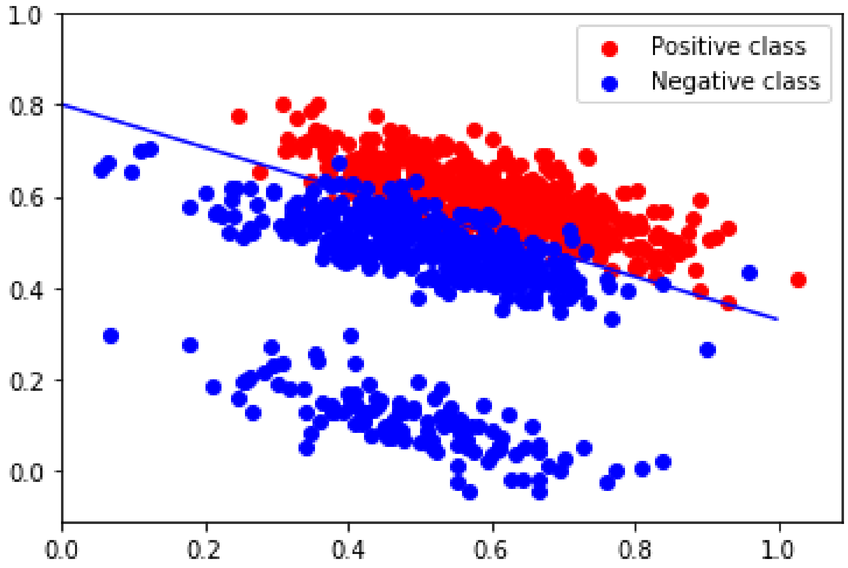
Linear Regression (Done for you)



Logistic regression



700
600
500
400
300
200
0
1
2
3
4
5
6


Positive class
Negative class