Machine Learning HW 2

Victor Manuel Garca Rosales

22/08/2017

1. Gradient Problems

Note: For the following problems use the developed data sets.

Problem 1 Implement the Gradient Solution of the Linear Regression under the following constraints:

(a) Implement step size using the Golden Method.

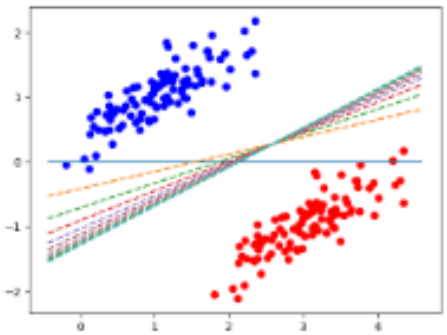

Figura 1: Gradient Descent regularization with different iteration values, learning rate fixed

```
0---After 1000 iterations b = 0, m = 0, error = 1.1974851109
1000---After 1000 iterations b = 0.413988786646, m = -0.266991366136, error = 0.89811612
2000---After 1000 iterations b = 0.710282951706, m = -0.381247876068, error = 0.79582897
3000---After 1000 iterations b = 0.90536119521, m = -0.456473443366, error = 0.751489327
4000---After 1000 iterations b = 1.03379945603, m = -0.506001473037, error = 0.732268884
5000---After 1000 iterations b = 1.11836238085, m = -0.538610409217, error = 0.723937166
6000---After 1000 iterations b = 1.17403806667, m = -0.560079923064, error = 0.720325516
7000---After 1000 iterations b = 1.21069458011, m = -0.574215312892, error = 0.718759930
8000---After 1000 iterations b = 1.23482899279, m = -0.583521962638, error = 0.718081277
9000---After 1000 iterations b = 1.25071893488, m = -0.589649400973, error = 0.717787093
10000---After 1000 iterations b = 1.26118077054, m = -0.593683666962, error = 0.71765956
11000---After 1000 iterations b = 1.26806877586, m = -0.596339801741, error = 0.71760429
12000---After 1000 iterations b = 1.27260379425, m = -0.598088583796, error = 0.71758032
13000---After 1000 iterations b = 1.2755896212, m = -0.599239970653, error = 0.71756994
14000---After 1000 iterations b = 1.27755547045, m = -0.599998036354, error = 0.71756543
15000---After 1000 iterations b = 1.27884977296, m = -0.600497141936, error = 0.71756348
16000---After 1000 iterations b = 1.2797019334, m = -0.600825749841, error = 0.71756264
17000---After 1000 iterations b = 1.28026299036, m = -0.601042103172, error = 0.71756227
18000---After 1000 iterations b = 1.28063238665, m = -0.601184548814, error = 0.71756211
```

19000---After 1000 iterations b = 1.28087559479, m = -0.601278334108, error = 0.71756204

As seen in this log file (b) Use Ridge Regularization.

(c) Wrap the method in a way that you can take λ values in an interval (0, 100) in steps of one unit.

(d) Explain what happens

First is need to be done the Gradient Descent, getting the matrix and the labels that we pass through the function. The Gradient is updated through every step done, we try to do a search for global optimization, doing the Golden Method Search, we have assumed that the function is convex. The Gradient converges to 0 in small steps since the Golden Search restrict us in that way.

Problem 2 Given the Fisher Linear Discriminant (Original Equations) please:

(a) Integrate the Ridge Regularization. $\hat{y} = w_0 + w^T X$ $w_0 = \frac{1}{n}\sum y^i - w^T x^i$

$$\int \jmath(\theta) = \int \frac{1}{N}\sum (y_i - \Theta x_i)^2 + \frac{\lambda}{2}\sum_{i=1}^{M} \Theta_i$$

(b) Wrap the method in a way that you can take λ values in an interval (0, 100) in steps of one unit.

(c) Explain what happens

2. Bayesian Classification

Problem 1 Do the following experiment:

(a) Generate samples from a the likelihood times a prior as follow:

$$\mathcal{N}(\mu, \sigma^2)\mathbf{x}U(0, 1)$$

Using the algorithm general slice sampler provided to you (Modify as necessary).

```
// SlicerSampler.py
def Set_Distribution(self, mu=0.0, sigma=1.0,a=0,b=1):
        """
    The test distributions for the slice sampler decomposition
    """

    def p1(x):
        return np.exp(-(x - mu) ** 2 / (2 * (sigma ** 2)))

    def p2(x):
        if(type(x) is np.float64 or type(x) is float):
                return np.array([(b-a)**-1 if a <= x <= b else 0])
        else:
                y = []
            for i in range(len(x)):
                y.append((b-a)**-1 if a <= x[i] <= b else 0)
            return np.array(y)

    self.set_p((p1, p2), (0, mu + sigma))
```

The Figure 1 shows the density function of a normal distribution and the Figure 2 shows the density of an uniform distribution, those will be used to compute the actual values that are shown in the Figure 3

and 4, the Figure 3 shows an overview of the actual values (smooth), and the Figure 4 shows the detailed values.

Since the uniform distribution is fixed to the range from 0 to 1, the boundaries of the result of the multiplication between the two distributions will be limited by the same range, so that's why the interval of the non-zero values. This technique can be used to constraint the results values and therefore avoid outliers values.

(a) Explain what is going on in terms of likelihood times a prior.

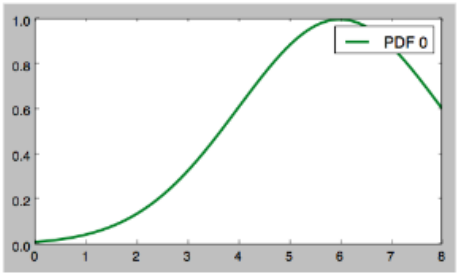

Figura 2: PDF of a normal distribution

Figura 3: PDF of an uniform distribution

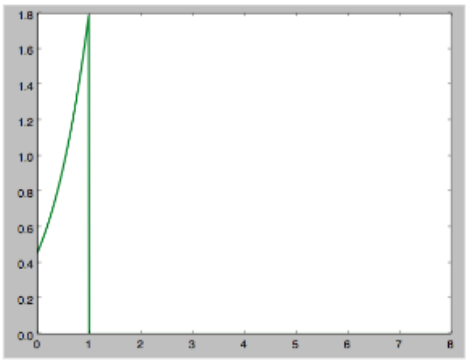

Figura 4: Smooth curve of the multiplication of PDF 0 (normal) & PDF 1 (uniform)

Problem 2 Derive the gradient of the negative log-likelihood for the multi class logistic regression case.

$$-l(\theta) = \sum_{n=1}^{N} -log\left[\frac{exp(\theta_{yn}^{T} x_n)}{\sum_{c=1}^{C} exp(\theta_c^{T} x_n)}\right]$$

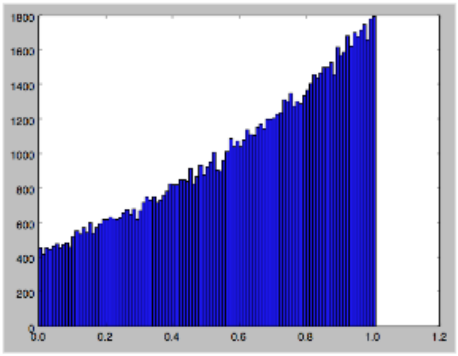

Figura 5: Detailed curve of the multiplication of PDF 0 (normal) & PDF 1 (uniform)

$$-l(\theta) = -\sum_{n=1}^{N} log\left[\frac{exp(\theta_{yn}^T x_n)}{\sum_{c=1}^{C} exp(\theta_c^T x_n)}\right]$$

$$-l(\theta) = -\sum_{n=1}^{N}\left[\theta_{y_n}^T x_n - log\sum exp(\theta_c^T x_n))\right]$$

$$\frac{\partial}{\partial\theta_{y'}^j} - l(\theta) = \frac{\partial}{\partial\theta_{y'}^j} - \sum_{n=1}^{N}\left[\theta_{y_n}^T x_n - log\sum exp(\theta_c^T x_n))\right]$$

$$\frac{\partial}{\partial\theta_{y'}^j} - l(\theta) = -\frac{\partial}{\partial\theta_{y'}^j}\sum_{n=1}^{N}\left[\theta_{y_n}^T x_n - log\sum exp(\theta_c^T x_n))\right]$$

$$-l(\theta) = -\sum_{n=1}^{N}\left[[y_n = y']x_n^{(j)} - p(y'|x_n))x_n^{(j)}\right]$$

(a) Under Ridge Regularization.

$$\frac{\partial}{\partial\theta_{y'}^j} - l(\theta) = -\sum_{n=1}^{N}\left[[y_n = y']x_n^{(j)} - p(y'|x_n))x_n^{(j)}\right] + \lambda\sum_{i=1}^{M}\epsilon_i$$

Problem 3 Do the following problems:

1. Machine Learning Theodoridis problem 7.20 at Chapter 7. Consider a two-dimensional class problem that involves two classes, w_1 and w_2 which are model by Gaussian distribution with means $\mu_1 = [0,0]^T$ and a covariance matrices $\sum_1 = \begin{bmatrix} 4 & 1{,}8 \\ 1{,}8 & 1 \end{bmatrix}$ and $\sum_2 = \begin{bmatrix} 4 & 1{,}2 \\ 1{,}2 & 1 \end{bmatrix}$, respectively.

   a) Form and plot a data set X consisting from 5000 points from w_1 and another 500 points from w_2.

   b) Assign each one of the points of w_1 or w_2, according to the Bayes decision rule, and plot the points with different colors, according to the class they are assigned to.

   c) Compute the error classification probability.

   d) Assign each one of the points of X to either w_1 or w_2, according to the Naive Bayes decision rule, and plot the points with different colors, according to the class they are assigned to

   e) Compute the error classification probability, for the naive Bayes classifier.

   f) Repeat steps (i)-(v) for the case where $\sum_1 = \sum_2 = \begin{bmatrix} 4 & 0 \\ 0 & 1 \end{bmatrix}$

2. Pattern Recognition and Machine Learning Bishop Problem 4.8 at Chapter 4. Using (4.57) and (4.58), derive the result (4.56) for the posterior class probability in the two-class generative model with Gaussian densities, and verify the results (4.66) and (4.67) for the parameters w and w_0