Master in Artificial Intelligence

Introduction to Machine Learning

A basic supervised problem

Authors:
Daniel Gibert
Ferran Pares

Supervisor:
Oriol Pujol

November 30, 2014

CONTENTS

LIST OF FIGURES

1

DATA SET ANALYSIS

The following figure displays the training samples and their corresponding labels:

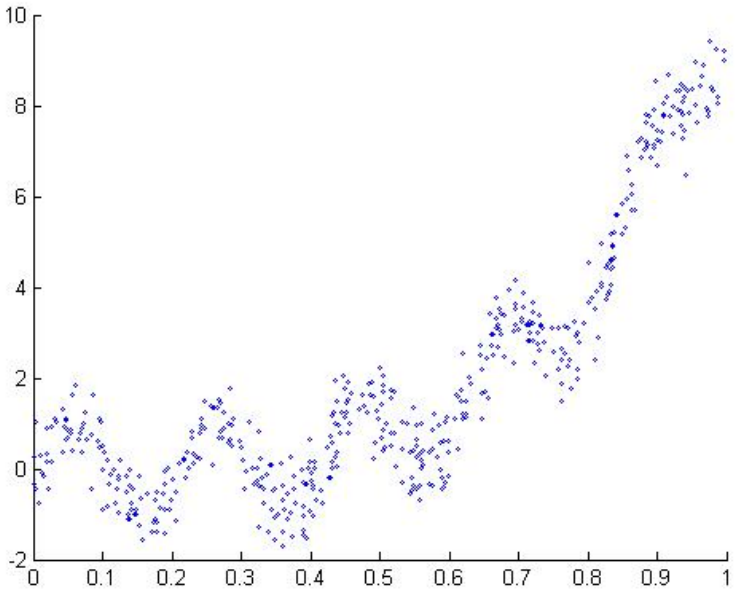

Figure 1: Training samples

The data given in *reg_data_set*1.*mat* is composed by an input value(x) as sample and an output value(y) as label:

$$x = [0, 1] \subseteq \Re$$

$$y = [-2, 10] \subseteq \Re$$

2

ANALYTICAL SOLUTION FOR THE LINEAR REGRESSION METHOD

To find the best values for $[w_0; w_1]$ we can measure the fit of the model by measuring the squared loss function over all training examples:

$$Loss(h_w) = \sum_{j=1}^{N} (y_i - (w_0 + w_1 * x_j))^2$$

$$Loss(h_w) = \sum_{j=1}^{N} (y_i - (w_0 + w_1 * x_j))^2$$

The Loss depending on $[w_0; w_1]$ can be viewed as a surface, that is convex and has a global minimum. In fact, we need to find the lowest point on this surface and use it as our weight values:

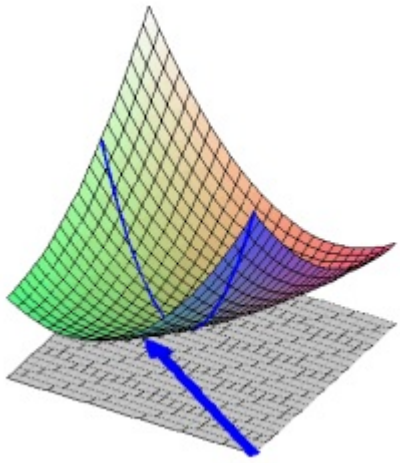

Figure 2: Training set - Convex surface

So we want to find $w^* = argmin_w Loss(h_w)$.
The sum $Loss(h_w)$ is minimized when its partial derivatives with respect to w_0 and w_1 are equal to 0:

$$\frac{\partial}{\partial w_0} \sum_{j=1}^{N} (y_j - (w_0 + w_1 * x_j))^2 = 0$$

$$\frac{\partial}{\partial w_1}\sum_{j=1}^{N}(y_j - (w_0 + w_1 * x_j))^2 = 0$$

Resulting in:

$$w_1 = (N * (\sum x_j * y_j) - (\sum x_j)(\sum y_j))/(N * (\sum x_j^2) - (\sum x_j)^2) = 7.9948$$
$$w_0 = (\sum y_j - w_1 * (\sum x_j))/N = -2.0474$$

The file named compute_weights_analitically.m contains the code to compute w_0, w_1.
However, this analytical calculus is computed with matrices using the following formula:

$$\begin{bmatrix} w_0 \\ w_1 \end{bmatrix} = \left(\begin{bmatrix} 1 & - & 1 \\ x_1 & - & x_n \end{bmatrix} * \begin{bmatrix} 1 & x_1 \\ | & | \\ 1 & x_n \end{bmatrix} \right)^{-1} * \begin{bmatrix} 1 & - & 1 \\ x_1 & - & x_n \end{bmatrix} * \begin{bmatrix} y_1 \\ | \\ y_n \end{bmatrix}$$

$$W = (\tilde{X}\tilde{X}^T)^{-1} * \tilde{X} * Y = \begin{bmatrix} -2.0474 \\ 7.9948 \end{bmatrix}$$

The figure below shows the line learned by this model.

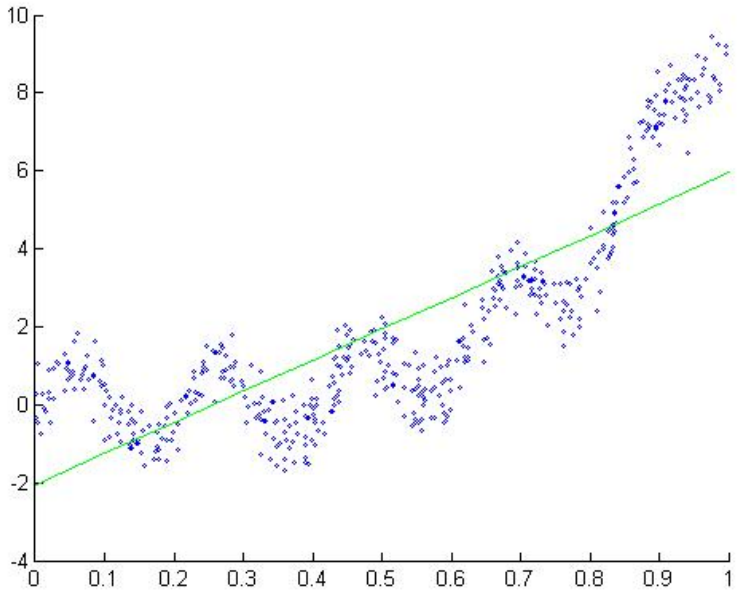

Figure 3: Line learned using optimal w_0, w_1

This is the best linear approximation to the given data. However, as it can be seen, the linear approximation model is not good enough because doesn't fit closely the training set.

3

LINEAR REGRESSION AND THE DESCENT METHOD

The file containing the gradient descent method for linear regression is called descentMethod.m. It needs as a parameters the learning rate, the number of iterations, the input values and the output values. It returns the optimal weight values for the model $[w_0; w_1]$.
The optimal values for the linear regression weights obtained by using this method are the same as the values obtained analytically, $[w_0 = -2.0474, w_1 = 7.9948]$.

This optimal values can be obtained by using different combinations of the learning rate and the number of iterations.
By theory, the learning rate(λ) determines how fast or slow it will converge to the optimal weights. If the λ is very large we will skip the optimal solution. If it is too small we will need too many iterations to converge to the best values and it will be a good idea to choose a higher number of iterations. So using a good λ is crucial.
We have executed the gradient descent method with $\lambda = 0.8$ and $num\_iter = 500$. The convergence curve obtained with the previous parameters is displayed below:

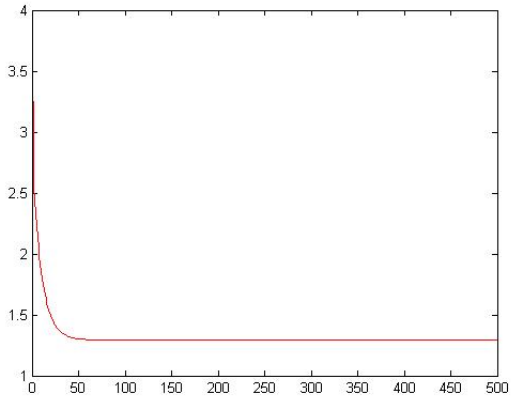

Figure 4: Convergence curve obtained with $\lambda = 0.8$

As it can be seen, the flat convergence part doesn't oscillate because when the obtained weight values are close to optimal ones, the gradient is shorter and the step tends to 0 (because the step size is multiplied by th gradient).

If we change the learning rate to 0.1, we will need too many number of iterations, but the optimal values will be obtained as the following figure shows:

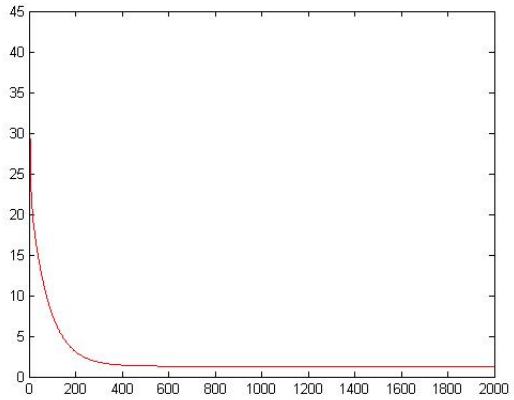

Figure 5: Convergence curve obtained with $\lambda = 0.1$

With the $\lambda = 0.1$ the algorithm needs near 600 iterations instead of the 80 needed with the $\lambda = 0.8$.

The descent algorithm is modified changing the descent direction (gradient) in the following way:

$$\triangle x = -\frac{f(x)}{\|f(x)\|_2}$$

The optimal weight values obtained computing this changed descent algorithm are $[w_0 = -2.0134, w_1 = 8.0132]$, using $\lambda = 0.5$ and 500 iterations.
The convergence curve is the following:

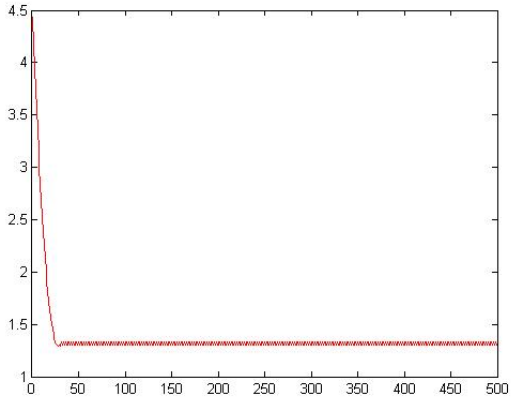

Figure 6: Convergence curve obtained with $\lambda = 0.5$

The convergence curve is oscillating at the flat part. This is because the gradient in this method is normalized, so the step size is fixed and determined by the λ value. So, using a fixed step size, this method is not able to get closer approximation to the optimal weight values.

4

A SECOND MODEL - POLYNOMIAL MODEL

Polynomial regression is a form of linear regression in which the relationship between the independent variable x and the dependent variable y is modelled as a pth degree polynomial.
The polynomial model of degree p for the univariate case, can be expressed as a linear combination of transformed data points $z = \{1, x, x^2, ..., x^p\}$,

$$f(x; w) = \sum_{i=0}^{p} w_i x^i = \sum_{i=0}^{p} w_i z_i = W^T Z$$

In particular, if $p = 3$, the linear combination of $z = \{1, x, x^2, x^3\}$, can be expressed as:

$$f(x; w) = \sum_{i=0}^{3} w_i x^i = \sum_{i=0}^{3} w_i z_i = W^T Z$$

The optimal value of the weights for this model is $[w_0 = 0.3901, w_1 = -2.1077, w_2 = -2.1699, w_3 = 13.6737]$. As it can be seen in the figure displayed below, the curve fits better than the curve obtained with p=1 linear regression. However, it is not the best possible curve that can be obtained using polynomial regression.

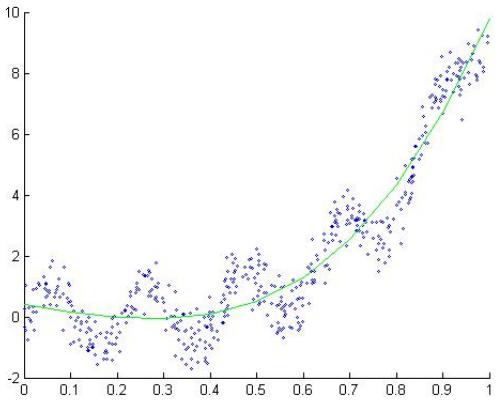

Figure 7: Polynomial regression with $p = 3$

5

EVALUATING A MODEL - POLYNOMIAL REGRESSION

For this part of the exercise, we have used another data set, called reg_data_set2.mat. The examples given by this data set had been splitted in two. The first ten examples are used for training, and the second ten examples are used for validation:

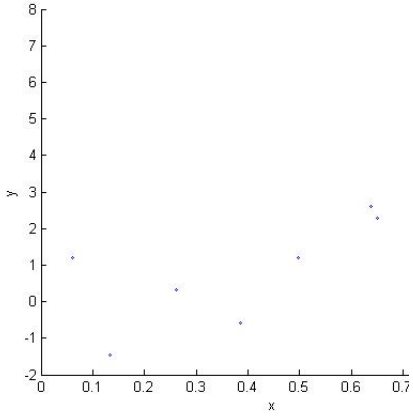

Figure 8: Training set

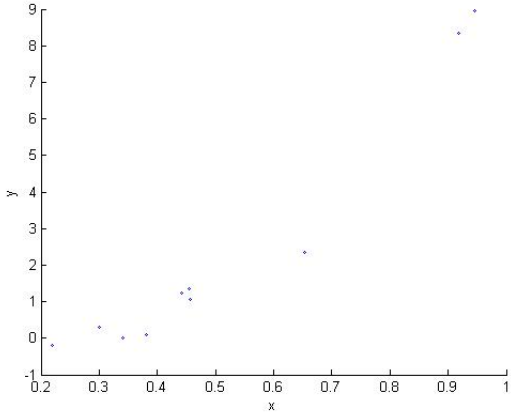

Figure 9: Validation set

The 6 model plots for the validation set are:

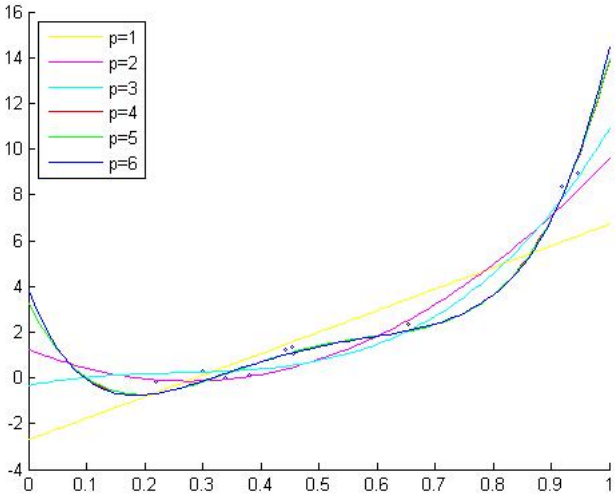

Figure 10: Validation plots

The 6 model plots for the training set are:

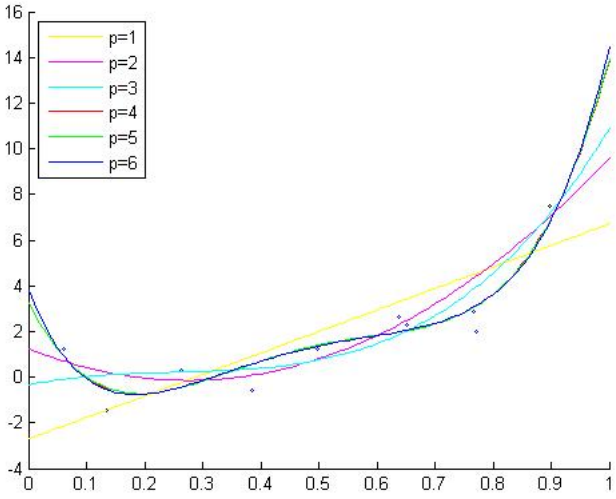

Figure 11: Training plots

In consequence, we have computed the following RMS error on the training set and on the validation set:

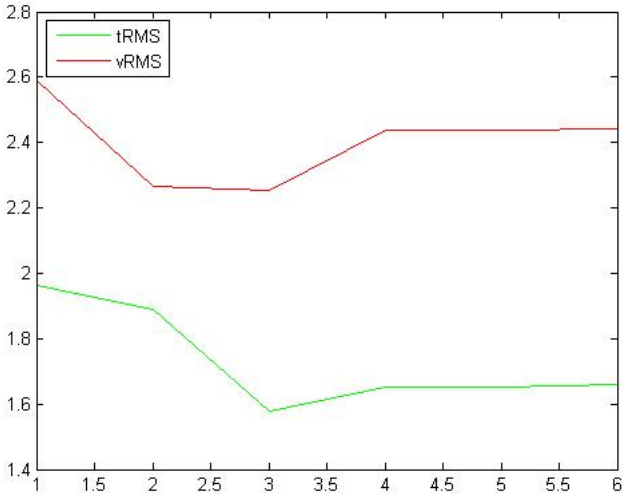

Figure 12: RMS error

The green line shows the RMS error between the model and the training set, and the red line shows the same RMS error between the model and the validation set. As expected, it has less error between the training set because is the one used to compute the model.
At this domain, both error lines agree that the optimal model is the one using p=3. It is considered the optimal because p=3 is the corresponding lowest RMS error reference point.

Keep in mind that this is the best result in the $1 \leq p \leq 6$ domain. Probably, if it is compute for further p values, another best p could be found that has less RMS error, so it fits better to the real model behind the given data.