BLG 454E Learning From Data
Homework 2

Ismail Bilgen (ibilgen@itu.edu.tr)

25/04/2019

1 Code up gradient descent for logistic regression (Exercise 3.11 in [1]) [40%]

In this exercise you will reproduce the gradient descent paths shown in the Figure below:

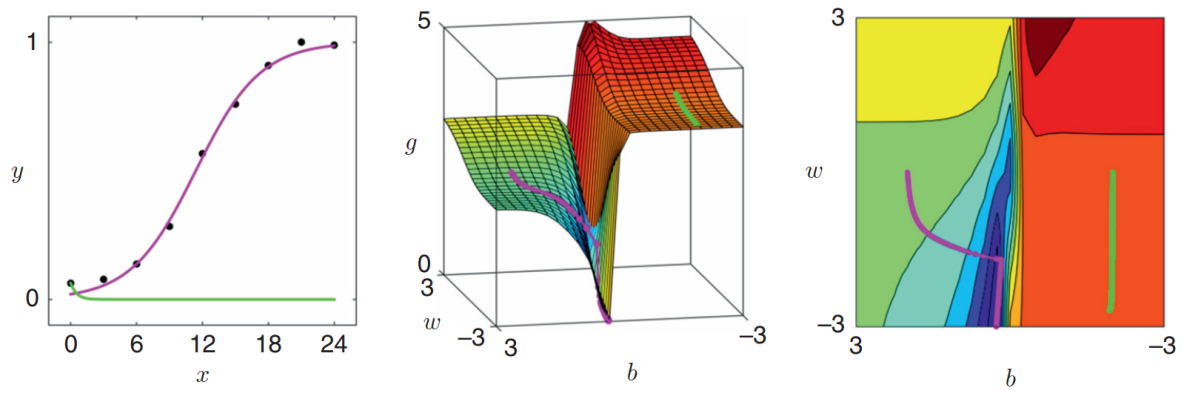

Figure 3.11: (left panel) A dataset along with two sigmoidal ts (shown in magenta and green), each found via minimizing the Least Squares cost in (3.26) using gradient descent with a different initialization. A surface (middle) and contour (right) plot of this cost function, along with the paths taken by the two runs of gradient descent. Each path has been colored to match the resulting sigmoidal t produced in the left panel (see text for further details). Data in this figure is taken from [2].

The gradient descent step shown in Equation 3.27 can be written more compactly by denoting

$$\sigma_p^{k-1} = \sigma\left(\tilde{\mathbf{x}}_p^T \tilde{\mathbf{w}}^{k-1}\right)$$
$$r_p^{k-1} = 2\left(\sigma_p^{k-1} - y_p\right)\sigma_p^{k-1}\left(1 - \sigma_p^{k-1}\right)$$

for all $p = 1, ..., P$, and $\mathbf{r}^{k-1} = [r_1^{k-1}\ r_2^{k-1}\ ...\ r_P^{k-1}]^T$, and stacking the $\tilde{\mathbf{x}}_p$ column-wise into the matrix $\tilde{\mathbf{X}}$. Then the gradient can be written as

$$\Delta g(\tilde{\mathbf{w}}^{k-1}) = \tilde{\mathbf{X}}\mathbf{r}^{k-1}$$

For programming languages like Python and MATLAB/OCTAVE that have especially efficient implementations of matrix/vector operations this can be much more efficient than explicitly summing over the P points as in Equation 3.27. In other words, these steps are shown for programming simplicity on

Python/Matlab like languages.

$$\Delta g(\tilde{w}) = 2\sum_{p=1}^{P}\left(\sigma\left(\tilde{x}_p^T\tilde{w}\right) - y_p\right)\sigma\left(\tilde{x}_p^T\tilde{w}\right)\left(1-\sigma\left(\tilde{x}_p^T\tilde{w}\right)\right)\tilde{x}_p \tag{3.27}$$

Note: See the Table 2 in Supplementary Information for size of all variables and their meaning. Also chapters 3 and 4 from Machine Learning Refined are included in the homework folder.

Complete gradient descent function

The surface in this figure was generated via the wrapper *nonconvex_logistic_growth* with the dataset *bacteria_data.csv*, and inside the wrapper you must complete a short gradient descent function to produce the descent paths called

$$[in, out] = grad\_descent(\tilde{\mathbf{X}}, \mathbf{y}, \tilde{\mathbf{w}}^0)$$

where "in" and "out" contain the gradient steps $\tilde{\mathbf{w}}^k = \tilde{\mathbf{w}}^{k-1} - \alpha_k \nabla g(\tilde{\mathbf{w}}^{k-1})$ taken corresponding objective value $g(\tilde{\mathbf{w}}^k)$ respectively, $\tilde{\mathbf{X}}$ is the input data matrix, $\mathbf{y}$ the output values, and $\tilde{\mathbf{w}}^0$ the initial point.

Almost all of this function has already been constructed for you. For example, the step length is fixed at $\alpha_k = 10^{-2}$ for all iterations, etc., and you must only enter the gradient of the associated cost function. Pressing "run" in the editor will run gradient descent and will reproduce **Fig. 3.11**.

2 Code up gradient descent for ℓ_2 regularized logistic regression (Exercise 3.13 in [1]) [40%]

In this exercise you will reproduce Fig. 3.13 by coding up gradient descent to minimize the regularized logistic regression Least Squares cost function shown in Equation 3.29.

$$g(b, w) = \sum_{p=1}^{P}\left(\sigma\left(b + x_p^T\, w\right) - y_p\right)^2 + \lambda\|w\|_2^2 \tag{3.29}$$

The gradient of the cost function can be written as Eq. 3.40

$$\nabla g(\tilde{\mathbf{w}}) = 2\sum_{p=1}^{P}\left(\sigma\left(\tilde{\mathbf{x}}_p^T\,\tilde{\mathbf{w}}\right) - y_p\right)\sigma\left(\tilde{\mathbf{x}}_p^T\,\tilde{\mathbf{w}}\right)\left(1-\sigma\left(\tilde{\mathbf{x}}_p^T\,\tilde{\mathbf{w}}\right)\right)\tilde{\mathbf{x}}_\mathbf{p} + 2\lambda\begin{bmatrix}0\\ \mathbf{w}\end{bmatrix} \tag{3.40}$$

Complete gradient descent function

The surface in this figure was generated via the wrapper *l2reg_nonconvex_logistic_growth* with the dataset *bacteria_data.csv*, and inside the wrapper you must complete a short gradient descent function to produce the descent paths called as 1

$$[in, out] = grad\_descent(\tilde{\mathbf{X}}, \mathbf{y}, \tilde{\mathbf{w}}^0) \tag{1}$$

where "in" and "out" contain the gradient steps $\tilde{\mathbf{w}}^k = \tilde{\mathbf{w}}^{k-1} - \alpha_k \nabla g(\tilde{\mathbf{w}}^{k-1})$ taken and corresponding objective value $g(\tilde{\mathbf{w}}^k)$ respectively, $\tilde{\mathbf{X}}$ is the input data matrix whose pth column is the input data $\tilde{\mathbf{x}}_p$, $\mathbf{y}$ the output values stacked into a column vector, and $\tilde{\mathbf{w}}^0$ the initial point.

Almost all of this function has already been constructed for you. For example, the step length is fixed at $\alpha_k = 10^{-2}$ for all iterations, etc., and you must only enter the gradient of the associated cost function. Pressing "run" in the editor will run gradient descent and will reproduce **Fig. 3.13**.

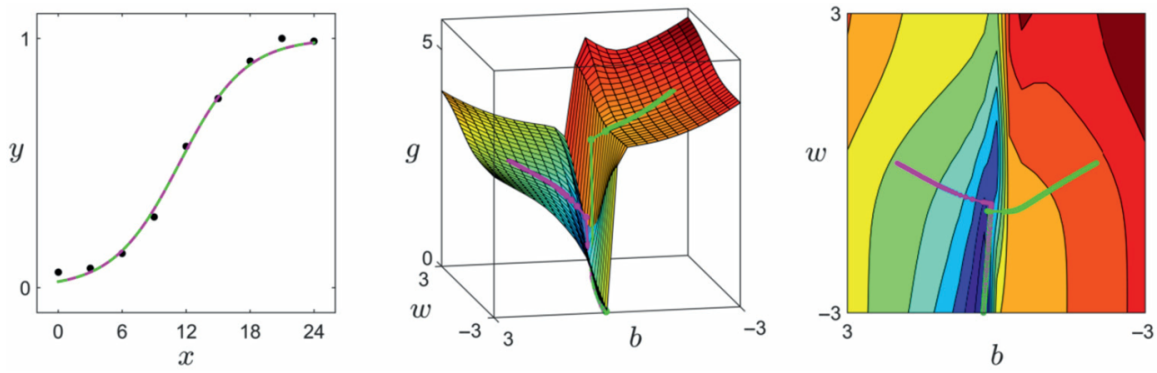

Figure 3.13: A regularized version of Fig. 4. (left panel) Plot of the bacterial growth dataset along with two overlapping sigmoidal ts (shown in magenta and green) found via minimizing the ℓ_2 regularized Least Squares cost for logistic regression in Equation 3.29 using gradient descent. (middle and right panels) The surface and contour plot of the regularized cost function along with the paths (in magenta and green) of gradient descent with same two initializations as shown in Fig. 4. While the surface is still non-convex, the large at region that originally led the initialization of the green path to a poor solution with the unregularized cost has been curved upwards by the regularizer, allowing the green run of gradient descent to reach the global minimum of the problem. Data in this figure is taken from [2].

3 Code up gradient descent for the multiclass softmax classier (Exercises 4.15 in [1]) [20%]

In this exercise you will code up gradient descent to minimize the multiclass softmax cost function on a toy dataset, reproducing the result shown in Fig. 4.20.

The gradient of the multiclass softmax perceptron is given by Equation (4.57) for each class $c = 1, ..., C$.

$$\nabla_{\tilde{\mathbf{w}}_c} g = \sum_{p=1}^{P} \left(\frac{1}{1 + \sum\limits_{\substack{j=1 \\ j \neq c}}^{C} e^{\tilde{\mathbf{x}}_p^T (\tilde{\mathbf{w}}_j - \tilde{\mathbf{w}}_c)}} - 1_{p \in \Omega_c} \right) \tilde{\mathbf{x}}_p \tag{4.57}$$

The gradient of g with respect to $\tilde{\mathbf{w}}_c$ for $c = 1, ..., C$, where $1_{p \in \Omega_c} = \begin{cases} 1 & \text{if } p \in \Omega_c \\ 0 & \text{else} \end{cases}$ is an indicator function on the set Ω_c.

Concatenating all individual classifiers' parameters into a single weight vector $\tilde{\mathbf{w}}_{all}$ as:

$$\tilde{\mathbf{w}}_{all} = \begin{bmatrix} \tilde{\mathbf{w}}_1 \\ \tilde{\mathbf{w}}_2 \\ . \\ . \\ . \\ \tilde{\mathbf{w}}_C \end{bmatrix} \tag{2}$$

the gradient of g with respect to $\tilde{\mathbf{w}}_{all}$ is formed by stacking block-wise gradients found in (4.57) into:

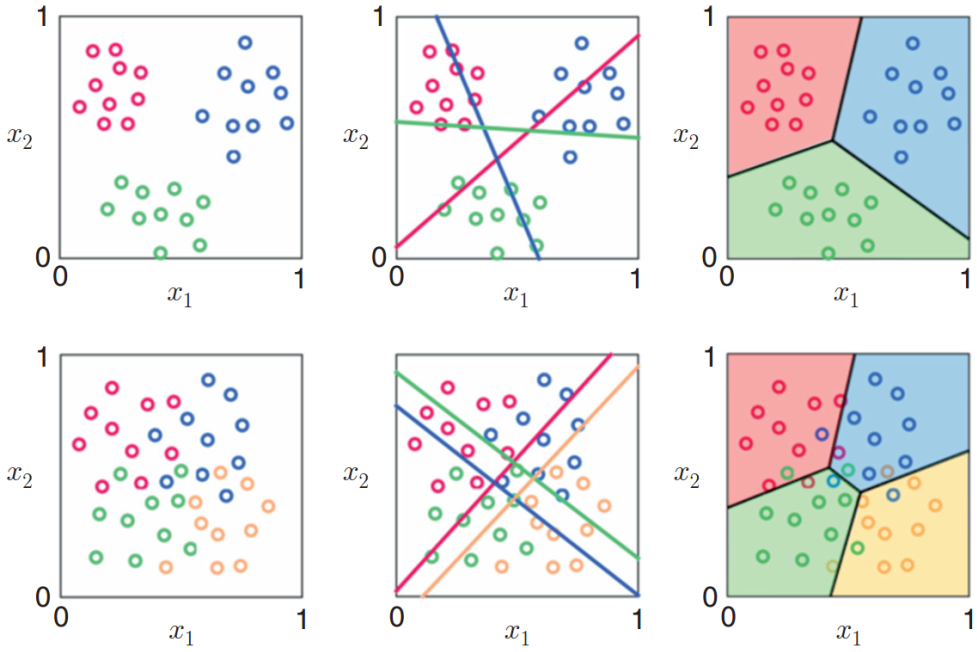

Figure 4.20: (top left panel) Toy dataset from Fig. 4.17 with $C = 3$ classes. (top middle panel) Individual linear classifiers learned by the multiclass softmax scheme. (top right panel) Final partitioning of the feature space resulting from the application of the fusion rule in 4.47. (bottom left panel) Toy dataset from Fig. 4.19 with $C = 4$ classes. (bottom middle panel) Individual linear classiers learned by the multiclass softmax scheme. (bottom right panel) Final partitioning of the feature space.

$$\nabla g = \begin{bmatrix} \nabla_{\tilde{\mathbf{w}}_1} g \\ \nabla_{\tilde{\mathbf{w}}_2} g \\ . \\ . \\ . \\ \nabla_{\tilde{\mathbf{w}}_C} g \end{bmatrix} \tag{3}$$

Complete gradient descent function

Code up gradient descent to minimize the multiclass softmax perceptron, reproducing the result shown for the C =4 class dataset shown in Fig. 4.20. This figure is generated via the wrapper *softmax_multiclass_grad_hw* and you must complete a short gradient descent function located within which takes the form in (4.84)

$$\tilde{\mathbf{W}} = \text{softmax\_multiclass\_grad}(\tilde{\mathbf{X}}, \mathbf{y}, \tilde{\mathbf{W}}^0, \text{alpha}) \tag{4.84}$$

Here $\tilde{\mathbf{W}} = [\tilde{\mathbf{w}}_1 \ \tilde{\mathbf{w}}_2 \ ... \ \tilde{\mathbf{w}}_C]$ is an $(N + 1) \times C$ matrix of weights, where $\tilde{\mathbf{w}}_c$ is the compact bias/weight vector associated with the cth classifier, $\tilde{\mathbf{X}}$ is the input data matrix, $\mathbf{y}$ the associated labels, and $\tilde{\mathbf{W}}^0$ the initialization of the weights. Almost all of this function has already been constructed for you. For example, the step length is fixed for all iterations, etc., and you must only enter the gradient of the associated cost function. All of the additional code necessary to generate the associated plot is already provided in the wrapper.

Solution Template

For each question create a table as below. In each row, show your all addition/change in the code and explain why you did this.

Table 1: Code.

The added/changed code statement	Explanation

For each question also put the screen-shot of the solution. Capture whole screen including the Matlab/Python program and the plots.

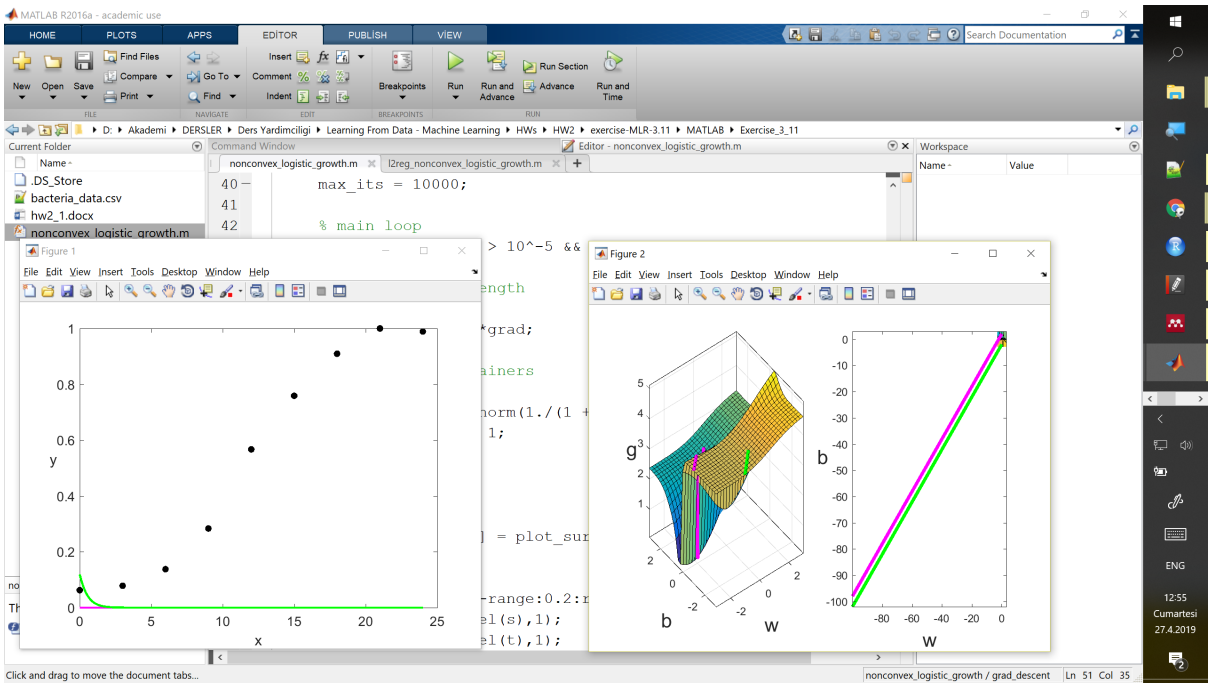

Figure 4: Screen shot of the solution. Put a screenshot of whole screen including the Matlab/Python program and the plots.

IMPORTANT: Each student should upload: (1) zipped code of all solved exercises for testing and (2) a report following this template.

Supplementary Information

The Least Squares cost function is

$$g(b,w) = \sum_{p=1}^{P} \left(\sigma\left(b + \mathbf{x}_p^T \mathbf{w}\right) - y_p\right)^2 \tag{4}$$

where $\mathbf{x}_p = [x_{1,p} \quad x_{2,p} \quad ... \quad x_{N,p}]^T$, and $\mathbf{w}_p = [w_1 \quad w_2 \quad ... \quad w_N]^T$

Using the compact notation $\tilde{\mathbf{x}}_p = \begin{bmatrix} 1 \\ \mathbf{x}_p \end{bmatrix}$ and $\tilde{\mathbf{w}} = \begin{bmatrix} b \\ \mathbf{w} \end{bmatrix}$

$$g(\tilde{w}) = \sum_{p=1}^{P} \left(\sigma\left(\tilde{\mathbf{x}}_p^T \tilde{\mathbf{w}}\right) - y_p\right)^2 \tag{5}$$

To minimize this cost we can employ gradient descent

$$\frac{d\, g(\tilde{\mathbf{w}})}{d\, \mathbf{w}} = \nabla g(\tilde{w}) = 2 \sum_{p=1}^{P} \left(\sigma\left(\tilde{x}_p^T \tilde{w}\right) - y_p\right) \sigma\left(\tilde{x}_p^T \tilde{w}\right) \left(1 - \sigma\left(\tilde{x}_p^T \tilde{w}\right)\right) \tilde{x}_p \tag{6}$$

Note that, $\sigma'(t) = \sigma(t)\left(1 - \sigma(t)\right)$ (see [3]).

Using derivative of Sigmoid function and the chain rule, one can calculate this gradient. Since $\mathbf{w}$ is a vector of length N so the derivative must be taken w.r.t. $\mathbf{w}_i$ for each $i \in [1, N]$. Note that the derivative w.r.t $\mathbf{w}_i$ gives zero for all terms except for the term that includes $\mathbf{w}_i$.

ℓ_2 regularized Least Squares cost function has an extra term of $\lambda \|\mathbf{w}_2^2\|$. So,

$$g(b,w) = \sum_{p=1}^{P} \left(\sigma\left(b + x_p^T\, w\right) - y_p\right)^2 + \lambda \|w\|_2^2 \tag{7}$$

In compact form, the gradient of regularized Least Squares cost function is the gradient in (6) plus the derivative of ℓ_2 regularizer term

$$\nabla g(\tilde{\mathbf{w}}) = 2 \sum_{p=1}^{P} \left(\sigma\left(\tilde{\mathbf{x}}_p^T \tilde{\mathbf{w}}\right) - y_p\right) \sigma\left(\tilde{\mathbf{x}}_p^T \tilde{\mathbf{w}}\right) \left(1 - \sigma\left(\tilde{\mathbf{x}}_p^T \tilde{\mathbf{w}}\right)\right) \tilde{\mathbf{x}}_p + 2\lambda \begin{bmatrix} 0 \\ \mathbf{w} \end{bmatrix} \tag{8}$$

The derivative of the ℓ_2 regularizer term,

$$\frac{\partial}{\partial \tilde{\mathbf{w}}} (\lambda \|w\|_2^2) = \frac{\partial}{\partial \tilde{\mathbf{w}}} (\sum_{n=1}^{N} w_n^2) = \lambda \frac{\partial}{\partial \tilde{\mathbf{w}}} (w_1^2 + w_2^2 + ... + w_N^2)$$

Say $A = (w_1^2 + w_2^2 + ... + w_N^2)$, since $\tilde{\mathbf{w}} = [b\ w_1\ ...\ w_N]^T$

$$\frac{\partial}{\partial \tilde{\mathbf{w}}} A = \begin{bmatrix} \frac{\partial A}{\partial b} & \frac{\partial A}{\partial w_1} ... \frac{\partial A}{\partial w_N} \end{bmatrix}^T = \begin{bmatrix} 0 & 2w_1 & 2w_2 & ... & 2w_N \end{bmatrix}^T = 2\lambda \begin{bmatrix} 0 \\ \mathbf{w} \end{bmatrix}$$

For derivation of the gradient of the multiclass softmax cost function, see the Section 4.4.2 in the book [1].

Table 2: The variables

$\mathbf{x}_p \in \mathbb{R}^N$	$[x_{1,p} \dots x_{N,p}]^T$ (N denotes the number of features)	Sample $\mathbf{x}_p$ where $p = 1 \dots P$ (P denotes the number of samples)
$\mathbf{w} \in \mathbb{R}^N$	$[w_1 \dots x_N]^T$	Weight vector
$\tilde{\mathbf{x}}_p^T \in \mathbb{R}^{N+1}$	$[1\, x_{1,p} \dots x_{N,p}]$	Compact notation
$\tilde{\mathbf{w}} \in \mathbb{R}^{N+1}$	$[b\, w_1 \dots x_N]^T$	Compact notation
$\sigma_p^{k-1} \in \mathbb{R}^1$	$\sigma([1\, x_{1,p} \dots x_{N,p}] \times [b\, w_1 \dots x_N]^T)$	σ_p^{k-1} value of $(k-1)$th iteration
$r_p^{k-1} \in \mathbb{R}^1$		r_p^{k-1} value of $(k-1)$th iteration
$r^{k-1} \in \mathbb{R}^P$	$[r_1^{k-1} \dots r_P^{k-1}]^T$	Stacked r_p^{k-1} value of $(k-1)$th iteration for all $p = 1 \dots P$
$y_p \in \mathbb{R}^1$		Response value of sample x_p
$\tilde{\mathbf{X}} \in \mathbb{R}^{P \times (N+1)}$		Stacked the $\tilde{\mathbf{x}}_p$ vectors column-wise into the matrix
$\tilde{\mathbf{w}}_c \in \mathbb{R}^{N+1}$	$[b_c\, w_{1,c} \dots x_{N,c}]^T$	Compact notation of weight vector for $class\, c$

References

[1] Watt, J., Borhani, R., and Katsaggelos, A. (2016). Machine Learning Refined: Foundations, Algorithms, and Applications. Cambridge: Cambridge University Press. doi:10.1017/CBO9781316402276

[2] Jianqiang Lin, Sang-Mok Lee, Ho-Joon Lee, and Yoon-Mo Koo. Modeling of typical microbial cell growth in batch culture. Biotechnology and Bioprocess Engineering, 5(5) 382385, 2000.

[3] Derivative of Sigmoid Function, *https://towardsdatascience.com/derivative-of-the-sigmoid-function-536880cf918e*. (Accessed on 24.4.2019.)