Logistic regression

$$\mathbf{y_{ji}} = y_j(\mathbf{x_i}) = \frac{e^{-(w_i^T T(\mathbf{x}))}}{\sum_{m=1}^{m=r} e^{-(w_k^T T(\mathbf{x}))}}$$

$$-\sum_{i=1}^{i=4} p_i log(p_i)$$

$$-\sum_{i=1}^{i=4} p_i log(q_i)$$

case 1: $k = i$

$$\mathbf{y_{ij}} = y_i(\mathbf{x_j}) = \frac{e^{-(w_i^T T(\mathbf{x_j}))}}{\sum_{m=1}^{m=r} e^{-(w_m^T T(\mathbf{x_j}))}}$$

$$-\frac{(\sum_{m=1}^{m=r} e^{-(w_m^T T(\mathbf{x_j}))}) e^{-(w_i^T T(\mathbf{x_j}))} T(\mathbf{x_j}) - e^{-(w_i^T T(\mathbf{x_j}))} e^{-(w_k^T T(\mathbf{x_j}))} T(\mathbf{x_j})}{(\sum_{m=1}^{m=r} e^{-(w_m^T T(\mathbf{x_j}))})^2}$$

$$= -y_{ij} T(\mathbf{x_j}) - y_{ij} y_{kj} T(\mathbf{x_j})$$

$$= -y_{ij} T(\mathbf{x_j})(1 - y_{kj})$$

2. case 2: $k \neq i$, we get the following.

$$-y_{ij}T(\mathbf{x_j})(0 - y_{kj})$$

$$-y_{ij}T(\mathbf{x_j})(I_{ki} - y_{kj})$$

where, I_{ki} is equal to 1 if $k = i$, 0, otherwise.

Differentiating with respect to $\mathbf{w_k}$, we get the following.

$$\sum_{j=1}^{j=100} \sum_{i=1}^{i=4} \frac{t_{ij}}{y_{ij}} y_{ij}T(\mathbf{x_j})(I_{ki} - y_{kj})$$

$$= \sum_{j=1}^{j=100} (t_{1j} T(\mathbf{x_j}) y_{kj} + t_{2j} T(\mathbf{x_j}) y_{kj} + t_{3j}$$

$$T(\mathbf{x_j}) y_{kj} + t_{4j} T(\mathbf{x_j}) y_{kj} + T(\mathbf{x_j}) y_{kj} - t_{kj} T(\mathbf{x_j}))$$

$$= \sum_{j=1}^{j=100} T(\mathbf{x_j})(y_{kj} - t_{kj})$$

$$\mathbf{w_k}(t+1) = \mathbf{w_k}(t+1) - \eta \sum_{j=1}^{j=100} T(\mathbf{x_j})(y_{kj} - t_{kj})$$

The steps involved in logistic regression (for 4 class problem) is given below.

1. Initialize the weight vectors $\mathbf{w_k}$ for $k = 1 \cdots 4$.
2. Compute y_{kj} for $k = 1 \cdots 4$ and $j = 1 \cdots 100$
3. Update the weight vectors using $\mathbf{w_k}(t+1) = \mathbf{w_k}(t+1) - \eta \sum_{j=1}^{j=100} T(\mathbf{x_j})(y_{kj} - t_{kj})$
4. Repeat (2) and (3) for finite number of iterations to obtain the optimal weight vectors $\mathbf{w_k}$ for $k = 1 \cdots 4$.

IRLS

$$g(\mathbf{x}) = f(\mathbf{x_o}) + \frac{\mathbf{x} - \mathbf{x_o^T}}{\mathbf{1}!}\nabla\mathbf{f} + \frac{(\mathbf{x} - \mathbf{x_o})^T\mathbf{H}(\mathbf{x} - \mathbf{x_o})}{2!}$$

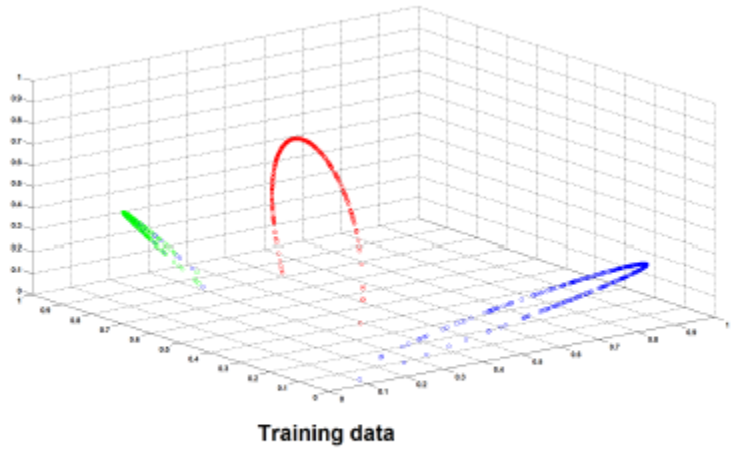
$$\nabla\mathbf{f} + (\mathbf{x} - \mathbf{x_o})^T\mathbf{H} = 0 \Rightarrow \mathbf{x} = \mathbf{x_0} - \eta\mathbf{H}^{-1}\nabla\mathbf{f}$$

Differentiating with respect to $\mathbf{w_k}$, we get the following.

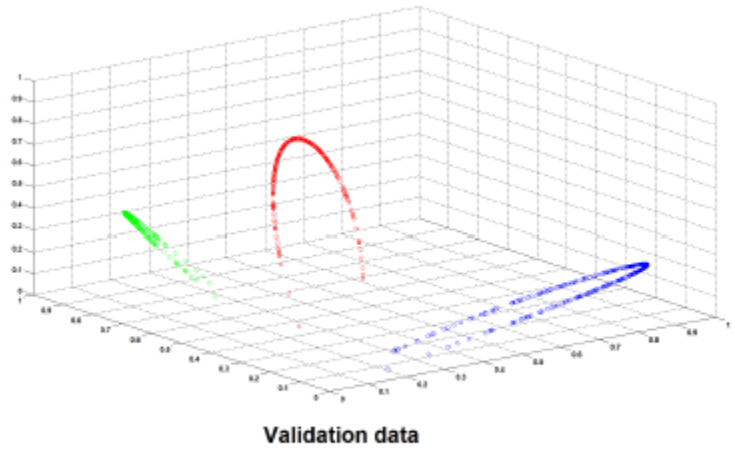
$$= \sum_{j=1}^{j=100} y_{kj} T(\mathbf{x_j}) T(\mathbf{x_j})^T (1 - y_{kj})$$

Thus to update $\mathbf{w_k}$, the following equation is used.

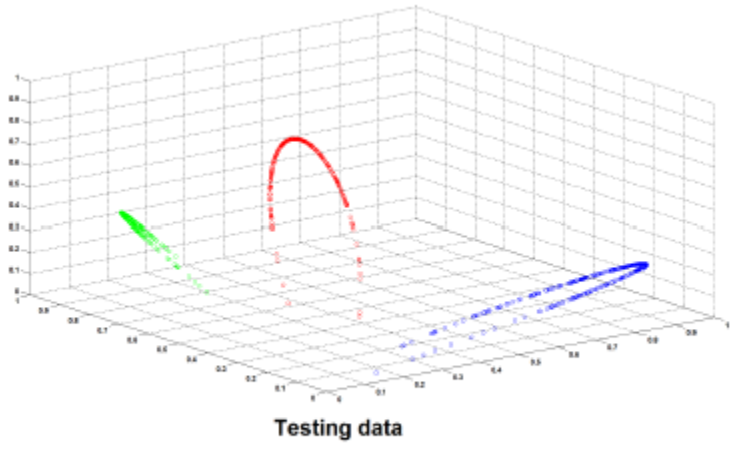
$$\mathbf{w_k}(t+1) = \mathbf{w_k}(t+1) - \eta \left(\sum_{j=1}^{j=100} y_{kj} T(\mathbf{x_j}) T(\mathbf{x_j})^T (1 - y_{kj}) \right)^{-1} \sum_{j=1}^{j=100} T(\mathbf{x_j})(y_{kj} - t_{kj})$$



Training data

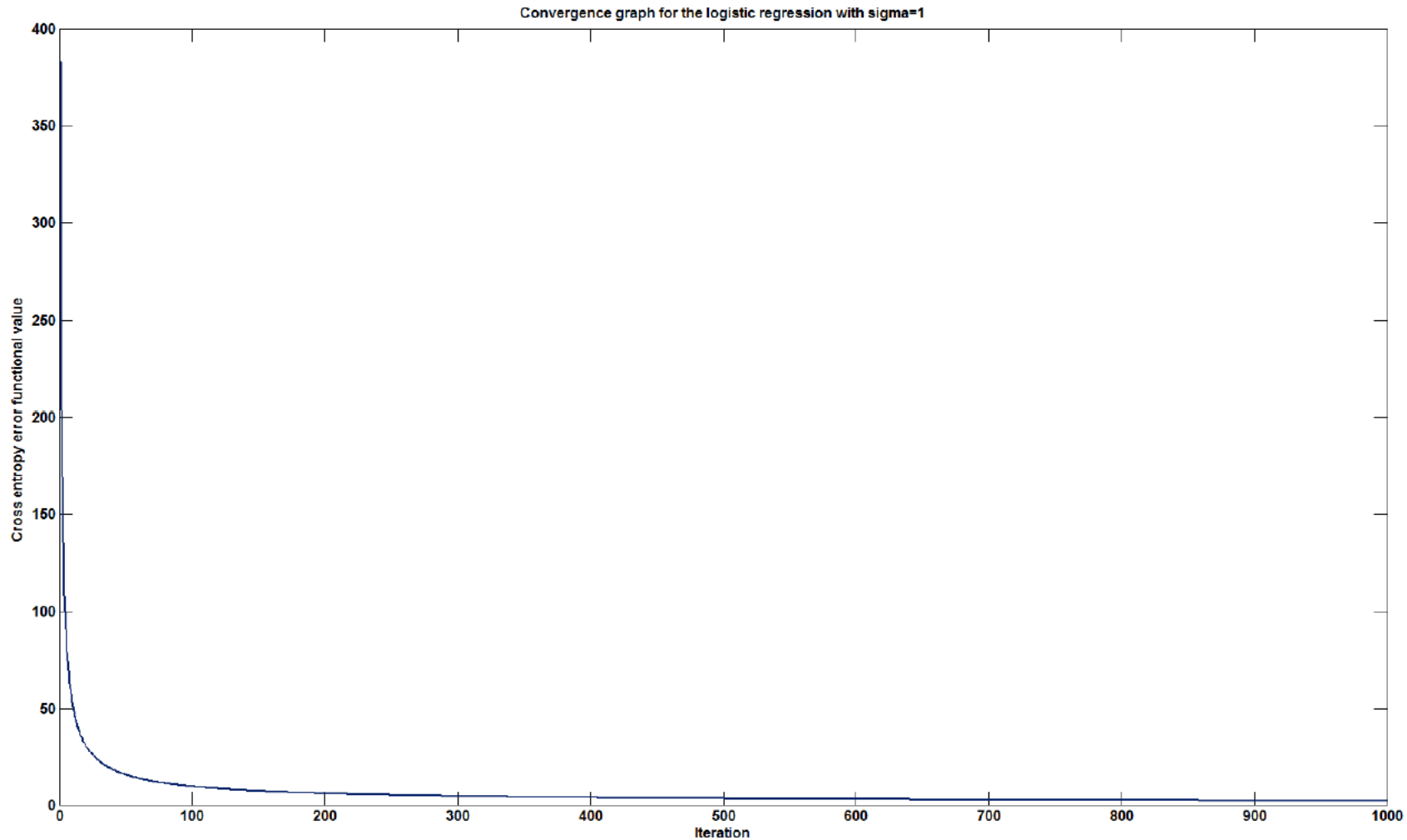


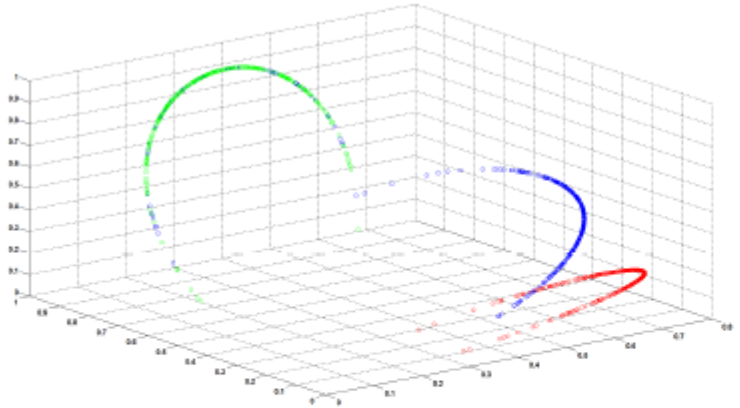
Validation data



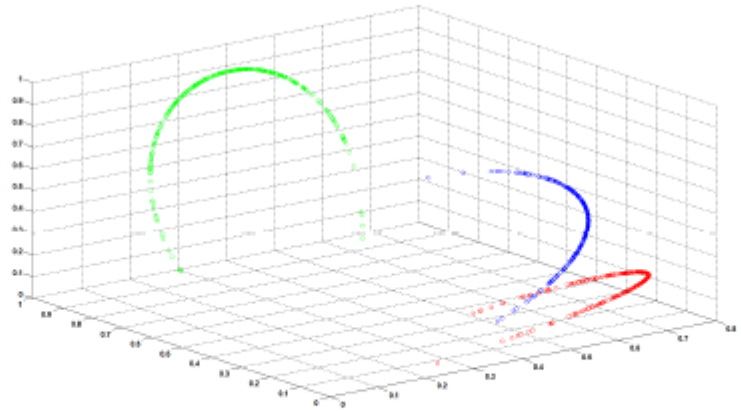
Testing data

Data used to demonstrate multi-class logistic regression.

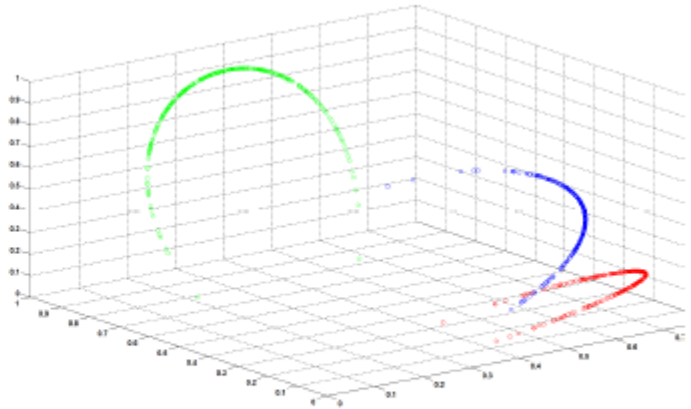

Convergence graph for the logistic regression with sigma=1
Cross entropy error functional value
Iteration
0
50
100
150
200
250
300
350
400
0
100
200
300
400
500
600
700
800
900
1000



Training data

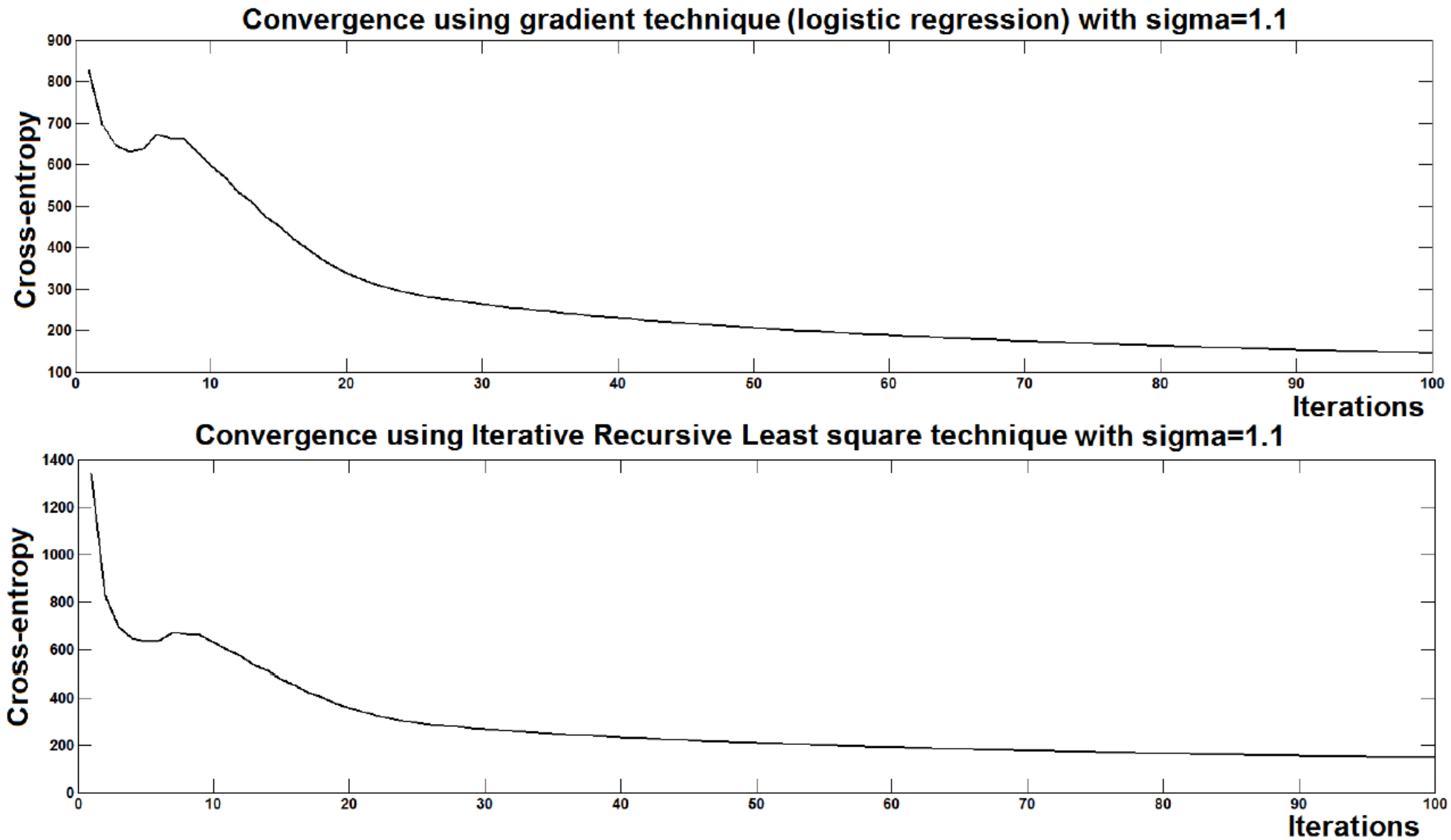


Validation data

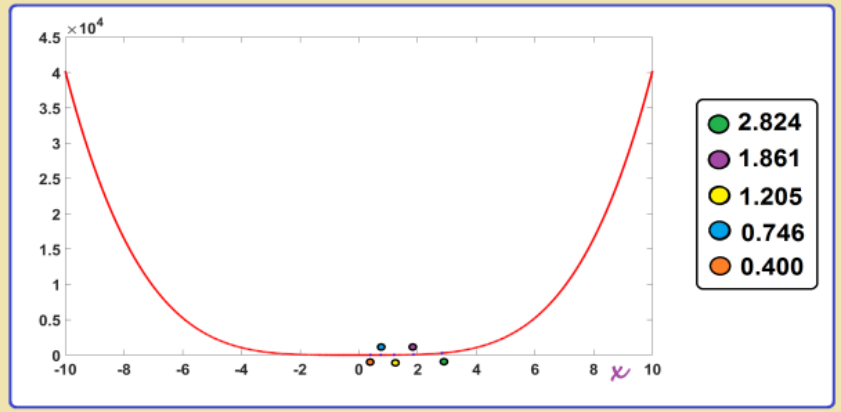


Testing data

Data used for demonstrating Iterative Recursive Least Square (IRLS) technique

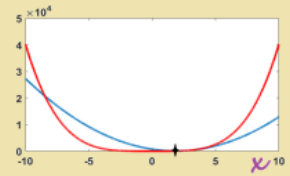


Convergence of logistic regression-steepest descent algorithm versus IRLS technique



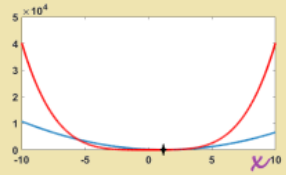
Intialize the value for x as

[1] Approximate the function using the first three terms of the Taylor series expansion constructed at the point as



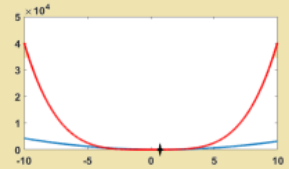
[2] Compute X corresponding to the minimum of the blue colored curve to obtain

[3] Repeat [1] at



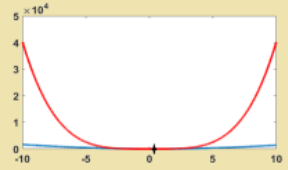
[4]Repeat [2] to obtain

[5]Repeat [1] at



[6] Repeat [2] to obtain

[7]Repeat [1] at



[8] Repeat [2] to obtain

This illustrates the usage of Taylor series based function approximation to minimize the function in Newton's iterative method

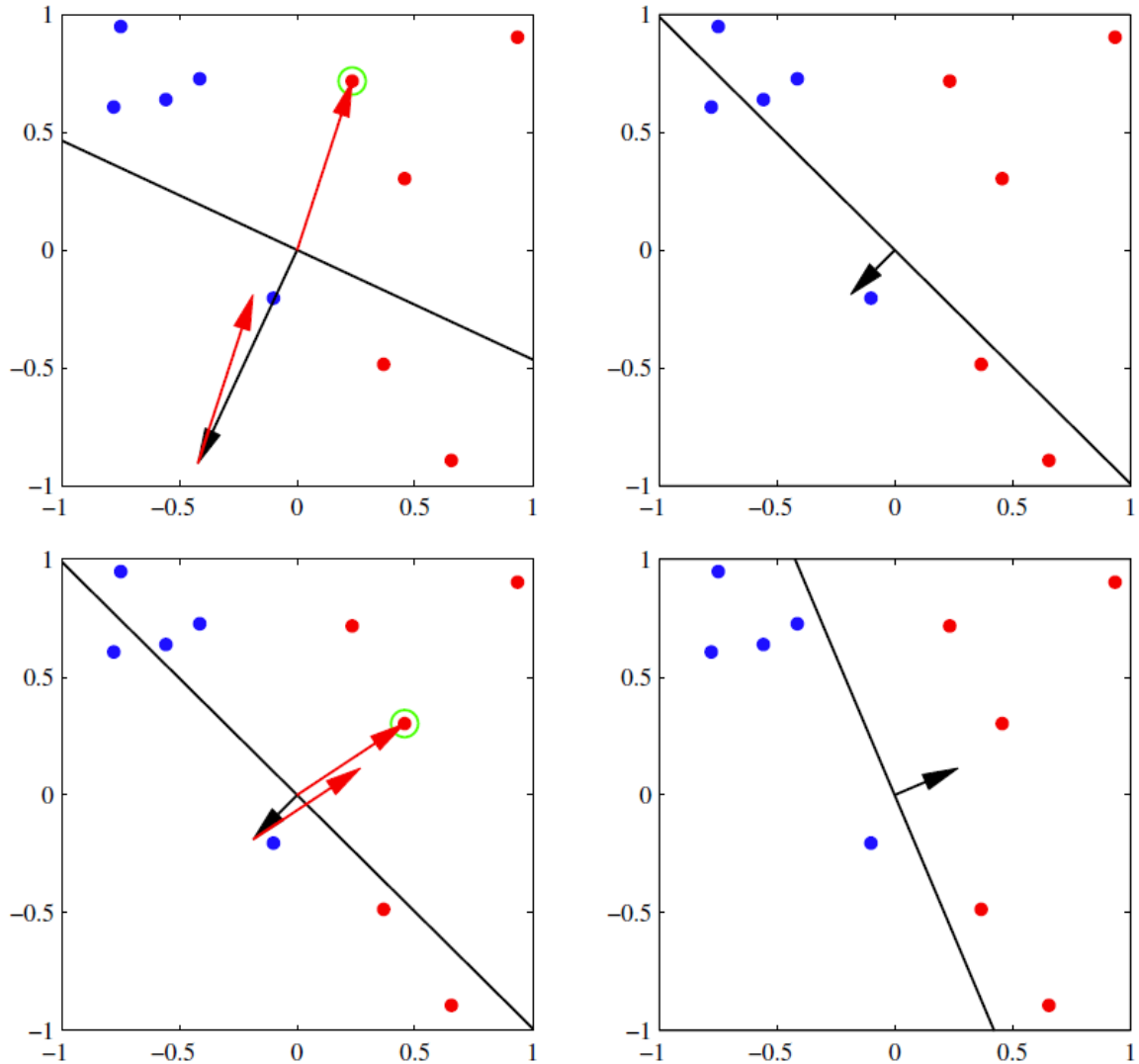
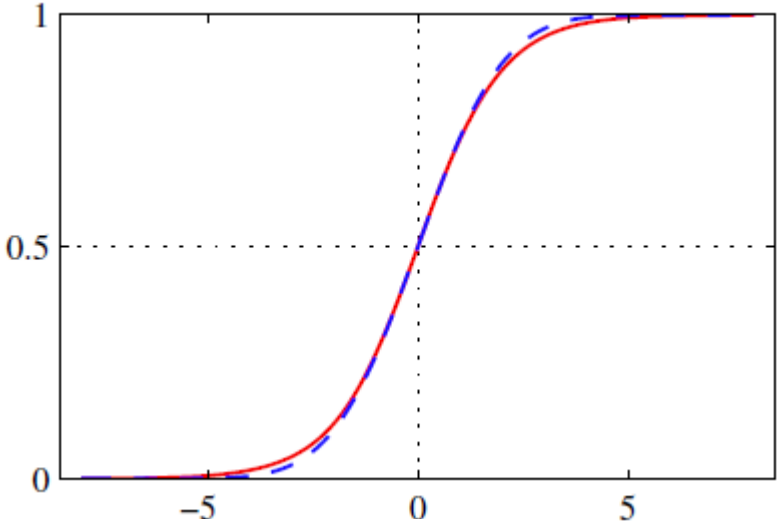
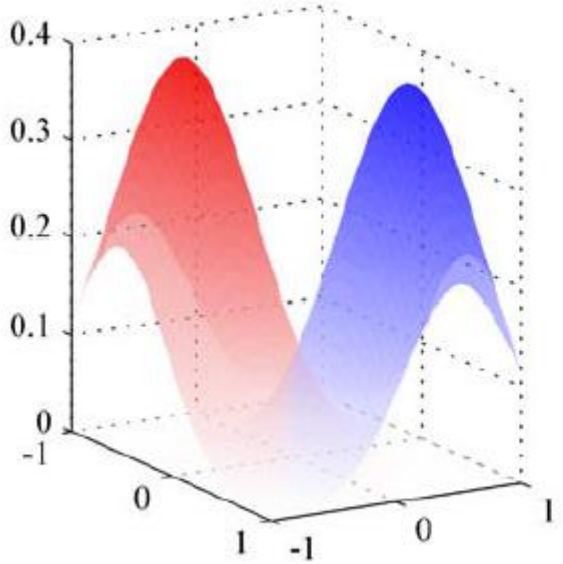
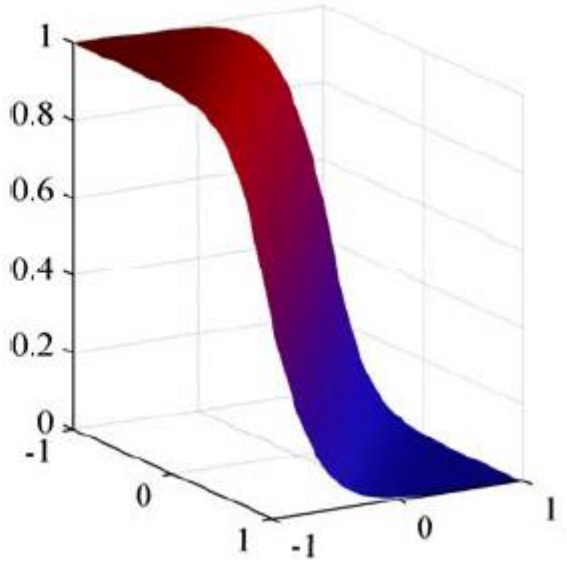


Illustration of the convergence of the perceptron learning algorithm, showing data points from two classes (red and blue) in a two-dimensional feature space (ϕ_1, ϕ_2). The top left plot shows the initial parameter vector $\mathbf{w}$ shown as a black arrow together with the corresponding decision boundary (black line), in which the arrow points towards the decision region which classified as belonging to the red class. The data point circled in green is misclassified and so its feature vector is added to the current weight vector, giving the new decision boundary shown in the top right plot. The bottom left plot shows the next misclassified point to be considered, indicated by the green circle, and its feature vector is again added to the weight vector giving the decision boundary shown in the bottom right plot for which all data points are correctly classified.

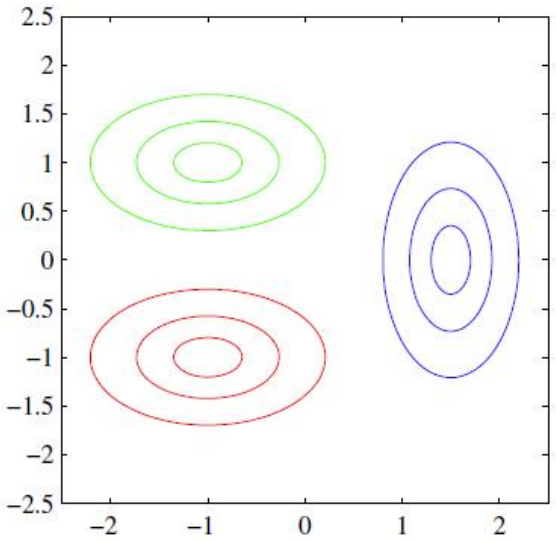


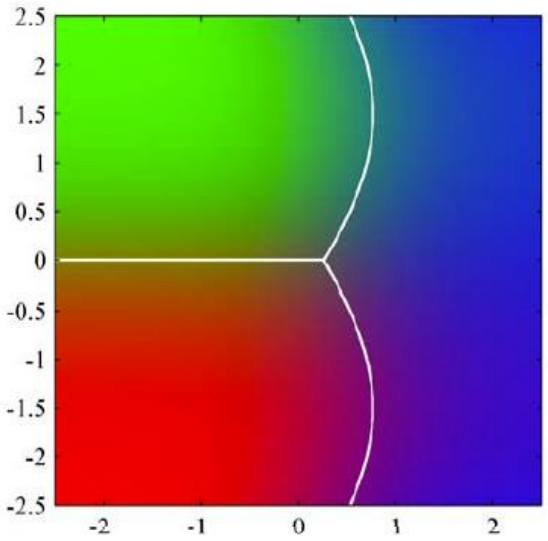
Plot of the logistic sigmoid function $\sigma(a)$ defined by (4.59), shown in red, together with the scaled probit function $\Phi(\lambda a)$, for $\lambda^2 = \pi/8$, shown in dashed blue, where $\Phi(a)$ is defined by (4.114). The scaling factor $\pi/8$ is chosen so that the derivatives of the two curves are equal for $a = 0$.





The left-hand plot shows the class-conditional densities for two classes, denoted red and blue. On the right is the corresponding posterior probability $p(\mathcal{C}_1|\mathbf{x})$, which is given by a logistic sigmoid of a linear function of $\mathbf{x}$. The surface in the right-hand plot is coloured using a proportion of red ink given by $p(\mathcal{C}_1|\mathbf{x})$ and a proportion of blue ink given by $p(\mathcal{C}_2|\mathbf{x}) = 1 - p(\mathcal{C}_1|\mathbf{x})$.





The left-hand plot shows the class-conditional densities for three classes each having a Gaussian distribution, coloured red, green, and blue, in which the red and green classes have the same covariance matrix. The right-hand plot shows the corresponding posterior probabilities, in which the RGB colour vector represents the posterior probabilities for the respective three classes. The decision boundaries are also shown. Notice that the boundary between the red and green classes, which have the same covariance matrix, is linear, whereas those between the other pairs of classes are quadratic.

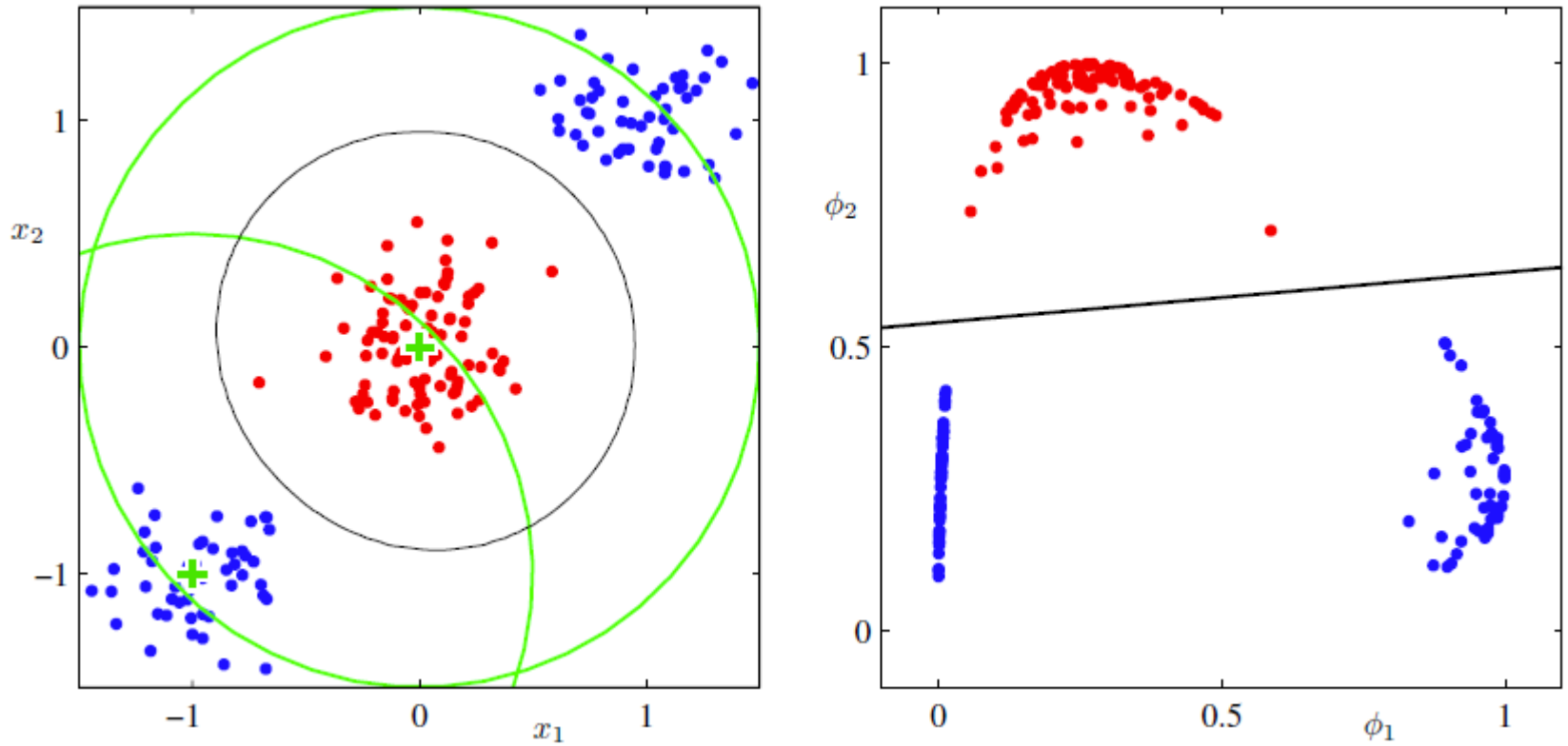


Illustration of the role of nonlinear basis functions in linear classification models. The left plot shows the original input space (x_1, x_2) together with data points from two classes labelled red and blue. Two 'Gaussian' basis functions $\phi_1(\mathbf{x})$ and $\phi_2(\mathbf{x})$ are defined in this space with centres shown by the green crosses and with contours shown by the green circles. The right-hand plot shows the corresponding feature space (ϕ_1, ϕ_2) together with the linear decision boundary obtained given by a logistic regression model of the form discussed in Section 4.3.2. This corresponds to a nonlinear decision boundary in the original input space, shown by the black curve in the left-hand plot.

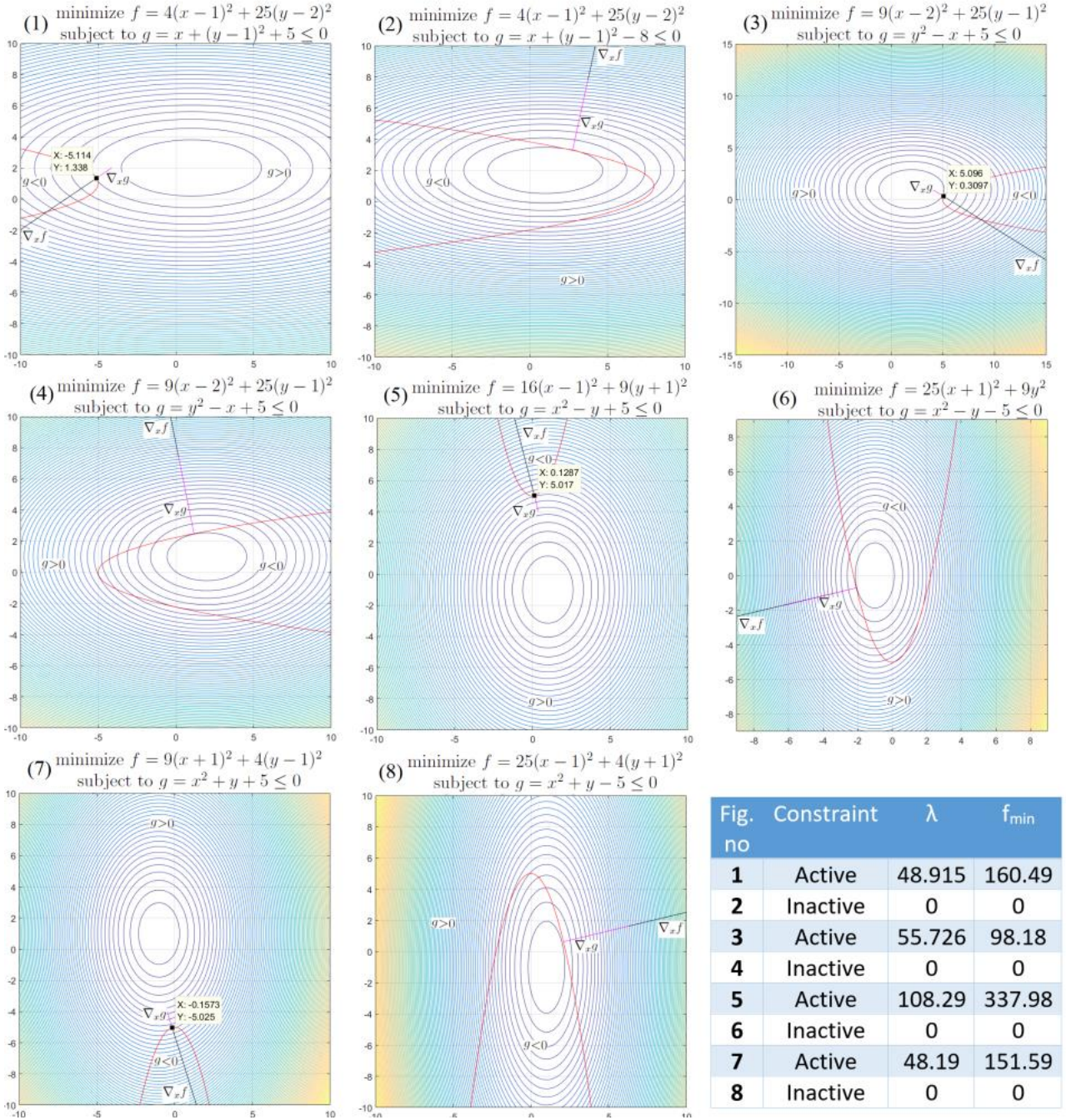


Fig. no	Constraint	λ	f_{min}
1	Active	48.915	160.49
2	Inactive	0	0
3	Active	55.726	98.18
4	Inactive	0	0
5	Active	108.29	337.98
6	Inactive	0	0
7	Active	48.19	151.59
8	Inactive	0	0

Illustration

Minimizing the function $f(x)$ with the constraint $g(x) \le 0$, satisfies $\nabla f(x) + \lambda \nabla g(x) = 0$ at the extremal points. If the optimal point satisfying the constraint happens to coincide with the one that minimizes the function $f(x)$ without the constraint, the case is identified as the inactive constraint ($\lambda = 0, g(x) < 0$). In such cases, both the gradients computed at the extremal point are in the same direction (refer 2,4,6,8). In the other cases (active constraints), it is observed from the figure (refer 1,3,5,7) that $\nabla f(x)$ and $\nabla g(x)$ are in the opposite direction computed at the extremal point and hence $\lambda > 0$, i.e. positive and the optimal point is the extremal point which lies on $g(x) = 0$. Thus the figure illustrates the usage of Karush-Kuhn-Tucker conditions ((a) $\nabla f(x) + \lambda \nabla g(x) = 0$ (b) $\lambda g(x) = 0$ and (c) $\lambda \ge 0$) for optimizing a function using inequality constraints.