Departamento de Eletrónica, Telecomunicações e Informática

Classification - Logistic regression

Author: Petia Georgieva
Edited by: Susana Brás (Susana.bras@ua.pt)

universidade de aveiro

LOGISTIC REGRESSION - outline

1. Sigmoid function model
2. Logistic regression cost (loss) function
3. Decision Boundary
4. Nonlinearly separable data
5. Regularized logistic regression
6. Multiclass

CLASSIFICATION

Email: Spam /Not Spam ?
Tumour: Malignant /Benign ?
Online Transactions: Fraudulent (Yes /No) ?

Binary classification:
y = 1: "positive class" (e.g. malignant tumour)
y = 0: "negative class" (e.g. benign tumour)

Threshold classifier output (probabilistic interpretation):

if h(x) >=0.5, predict "y=1"
if h(x) <0.5, predict "y=0"
0<=h(x)<=1

Multiclass classification (*m* classes) => y= {0, 1, 2,..}
Build *m* binary classifiers, for each classifier one of the classes has label 1 all other classes take label 0 .

Linear vs Logistic Regression

	Linear	Logistic
Nature of the Dependent Variable	The dependent variable is continuous and can take any real value.	The dependent variable is binary or categorical, representing **two** classes.
Equation Formulation	The linear equation predicts the value of the dependent variable directly.	The logistic function transforms the linear combination into a probability, and a threshold is applied for classification.
Output Interpretation	The output is interpreted as the expected value of the dependent variable.	The output is interpreted as the probability of the event occurring (class 1).
Applications	Used for predicting continuous outcomes, such as sales, temperature, or stock prices.	Applied in binary classification problems, like spam detection, medical diagnosis, or customer churn prediction.

Sigmoid function model

Logistic Regression

Given labelled data of m examples and n features
Labels {0,1} => binary classification
x –vector of features; θ – vector of model parameters;
h(x) – logistic (sigmoid) function model (hypothesis)

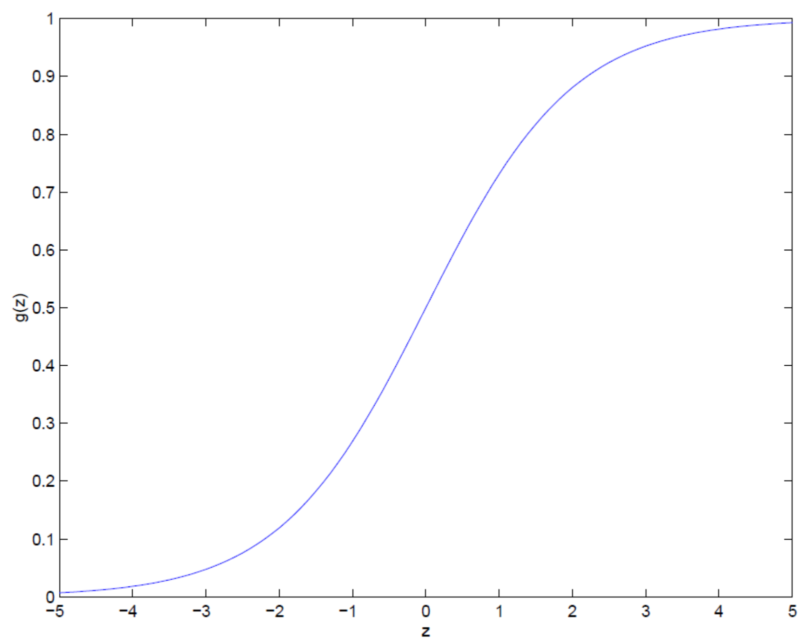
$$h_\theta(x) = \frac{1}{1+e^{-\theta^T x}} = \frac{1}{1+e^{-z}} = g(\theta^T x) = g(z)$$

$$z = \theta^T x = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots \theta_n x_n$$

Logistic regression works by applying the logistic function to a linear combination of the input features.

1. Calculate a weighted sum of the input features (similar to linear regression).
2. Apply the logistic function (also called sigmoid function) to this sum, which maps any real number to a value between 0 and 1.
3. Interpret this value as the probability of belonging to the positive class.
4. Use a threshold (typically 0.5) to make the final classification decision.

Logistic (sigmoid) function



universidade de aveiro

Logistic regression cost (loss) function

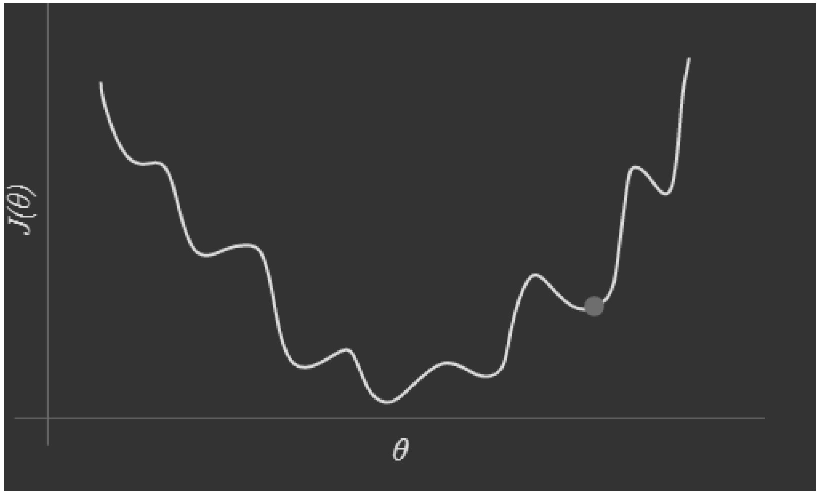
Logistic Regression Cost Function

Linear regression model => $$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots.\theta_n x_n = \vec{\theta}^T \vec{x}$$

Linear Regression cost function => $$J = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$$

Nonlinear logistic (sigmoid) model => $$h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$$

If we use the same cost function as with linear regression, but now the hypothesis is a nonlinear function, *J(θ)* will be a non-convex function (has many local minima)=> **not efficient for optimization !**



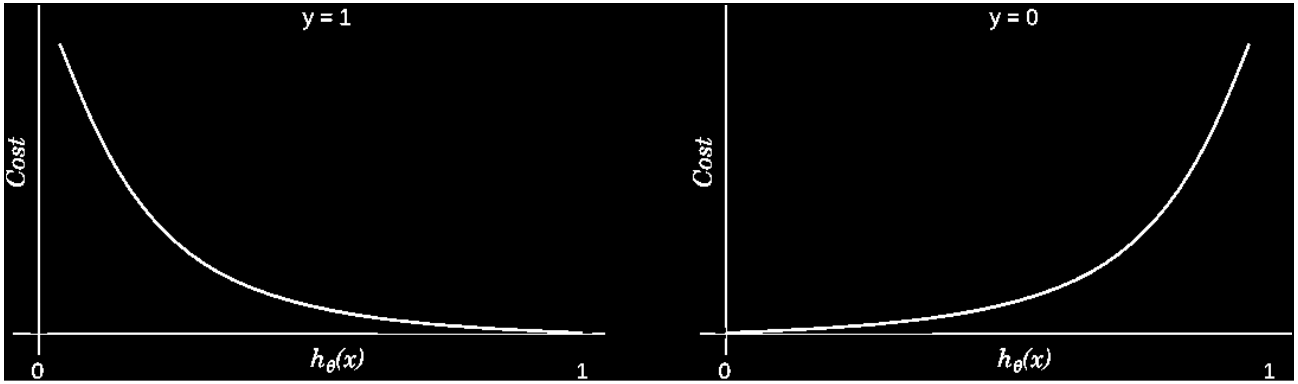
Logistic Regression Cost Function

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \mathrm{Cost}(h_\theta(x^{(i)}), y^{(i)})$$

$$\mathrm{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

Note: $y = 0$ or 1 always

$-\log(h_\theta(x))$ $-\log(1-h_\theta(x))$



LogReg cost function combined into one expression :
(also known as binary Cross-Entropy or Log Loss function)

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left[-y^{(i)}\log(h_\theta(x^{(i)})) - (1-y^{(i)})\log(1-h_\theta(x^{(i)}))\right]$$

Log Reg with gradient descent learning

Inicialize model parameters *(e.g. θ =0)*
Repeat until J converge {

Compute LogReg Model prediction =>
(different from linear regression model)

$$h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$$

Compute LogReg cost function =>
(different from linear regression cost function)

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left[-y^{(i)}\log(h_\theta(x^{(i)})) - (1-y^{(i)})\log(1-h_\theta(x^{(i)}))\right]$$

Goal =>

$$\min_\theta J(\theta)$$

Compute cost function gradients =>
(same as linear regression gradients)

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$$

Update parameters =>
(same as linear regression parameter update)

$$\theta_j := \theta_j - \alpha\frac{\partial}{\partial \theta_j}J(\theta)$$

}

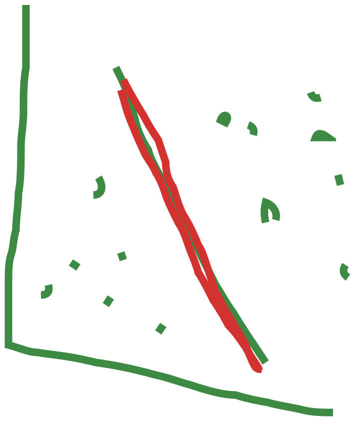
Optimization algorithms

Gradient descent (learned in class) -
updates the parameters in direction in which the gradient decreases most rapidly.

2. Other optimization algorithms
- Conjugate gradient
- AdaGrad, RMSProp
- Stochastic gradient descent with momentum
- ADAM (combination of RMSProp and stochastic optimization)
- BFGS (Broyden–Fletcher–Goldfarb–Shanno)
- Quasi-Newton methods (approximate the second derivative)

Characteristics

- Adaptive learning rate (alfa);
- Often faster than gradient descent; better convergence;
- Approximate (estimate) the true gradient over a mini-batch and not over the whole data;
- More complex algorithms



Decision Boundary

Logistic regression - trainning

The training process for logistic regression involves finding the best weights for the input features.

1. Initialize the weights (often to small random values).

2. For each training example:
    1. Calculate the predicted probability using the current weights.
    2. Compare this probability to the actual class label by calculating its loss.

3. Update the weights to minimize the loss (usually using some optimization algorithm, like gradient descent.)

4. Repeat Step 2 until log loss cannot get smaller (or other stop criteria was met).

Logistic regression - example

Training data: applicant's scores on two exams and the admission decision (historical data). Build a logistic regression model to predict whether a student gets admitted into a university.

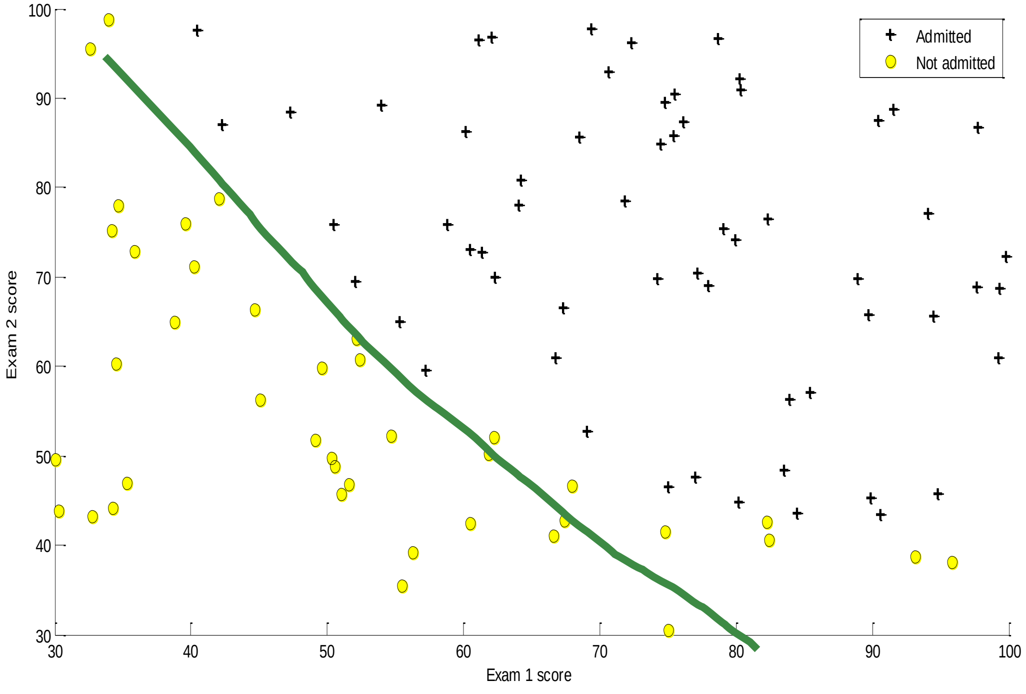


Fig. 1 Scatter plot of training data

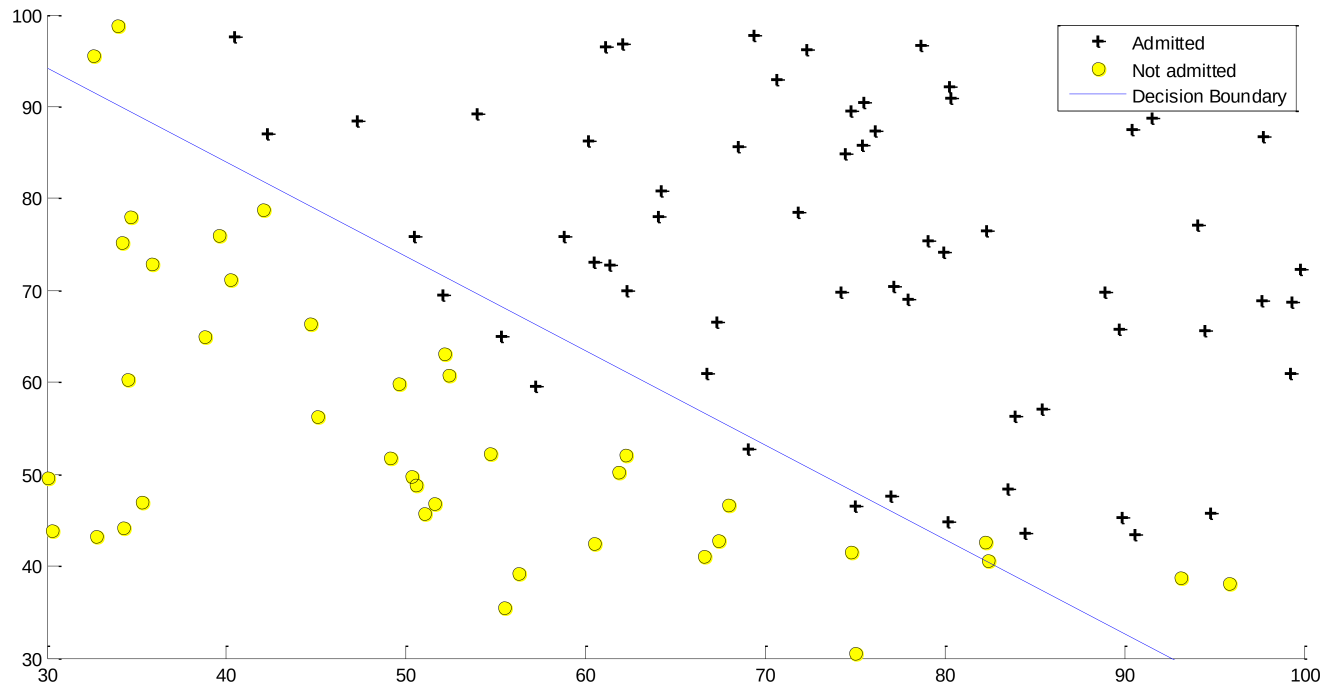
Logistic regression - example

$$z = \theta^T x = \theta_0 + \theta_1 x_1 + \theta_2 x_2 = 0 \Rightarrow \text{decision boundary}$$

$$if \quad z > 0 \Rightarrow g(z) > 0.5 \Rightarrow \text{predict class} = 1$$

$$if \quad z < 0 \Rightarrow g(z) < 0.5 \Rightarrow \text{predict class} = 0$$

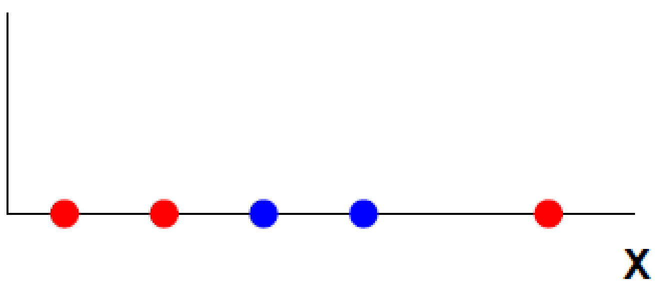
Scatter plot of training data and linear decision boundary with the optimized parameters



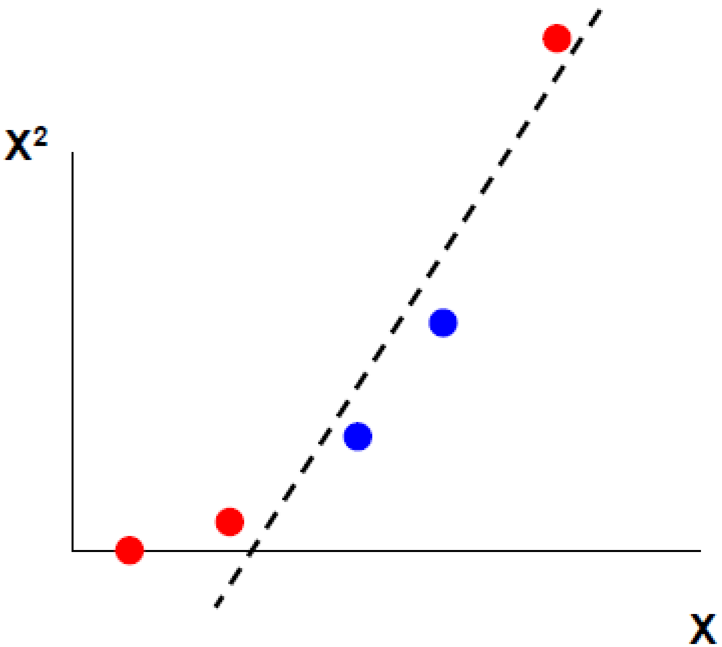
Nonlinear Separable Data

Nonlinearly Separable Data

Linear classifier cannot classify these examples.



And now ?



$$z = \theta^T x = \theta_0 + \theta_1 x + \theta_2 x^2 = 0 =>$$

Nonlinear decision boundary (in the original feature space x)

Linear decision boundary (in the extended feature space x, x^2)

$if \quad z > 0 => g(z) > 0.5 =>$ predict class $= 1$

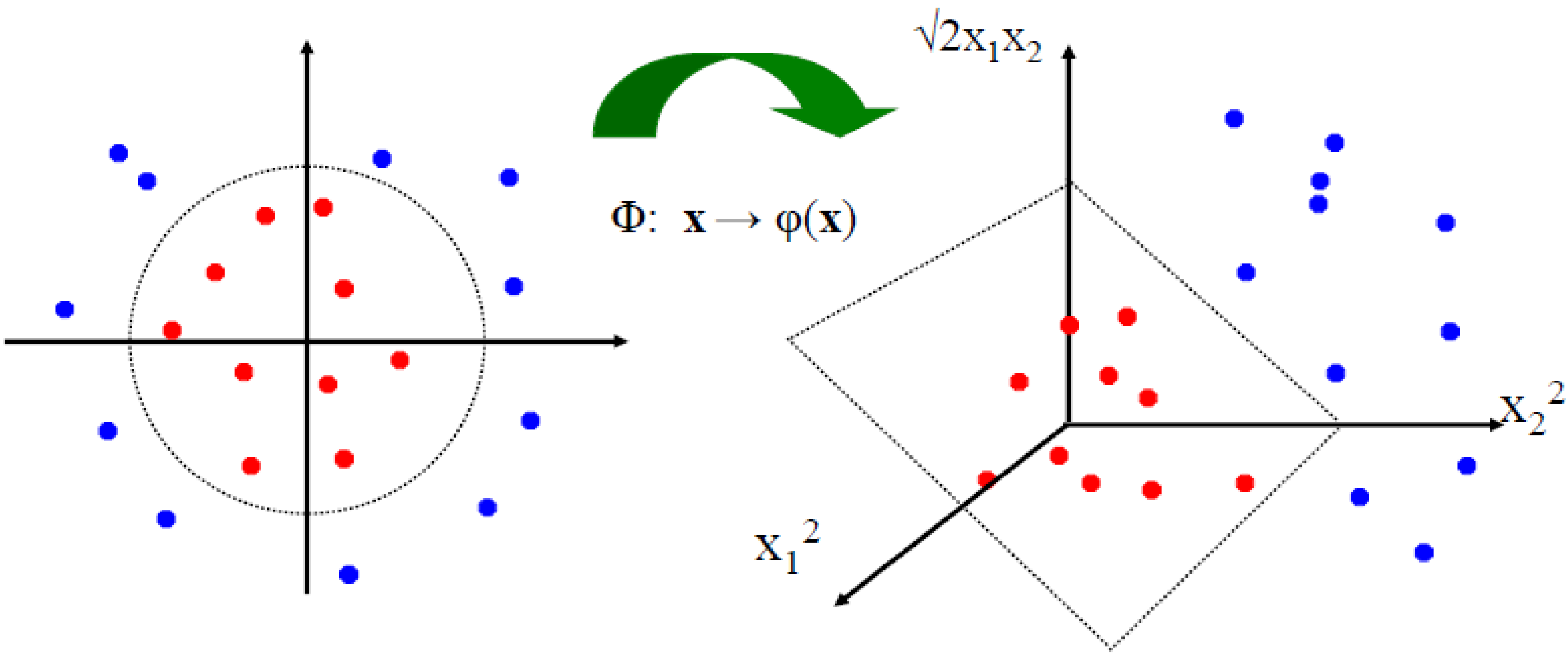
$if \quad z < 0 => g(z) < 0.5 =>$ predict class $= 0$

Nonlinearly Separable Data

- The original input space (x) can be mapped to some higher-dimensional feature space ($\varphi(\mathbf{x})$)where the training set is separable:

$\mathrm{x}=(\mathrm{x}_1,\mathrm{x}_2)$

$\varphi(\mathbf{x}) = (\mathrm{x}_1^{\,2}, \mathrm{x}_2^{\,2}, \sqrt{2}\mathrm{x}_1\mathrm{x}_2)$

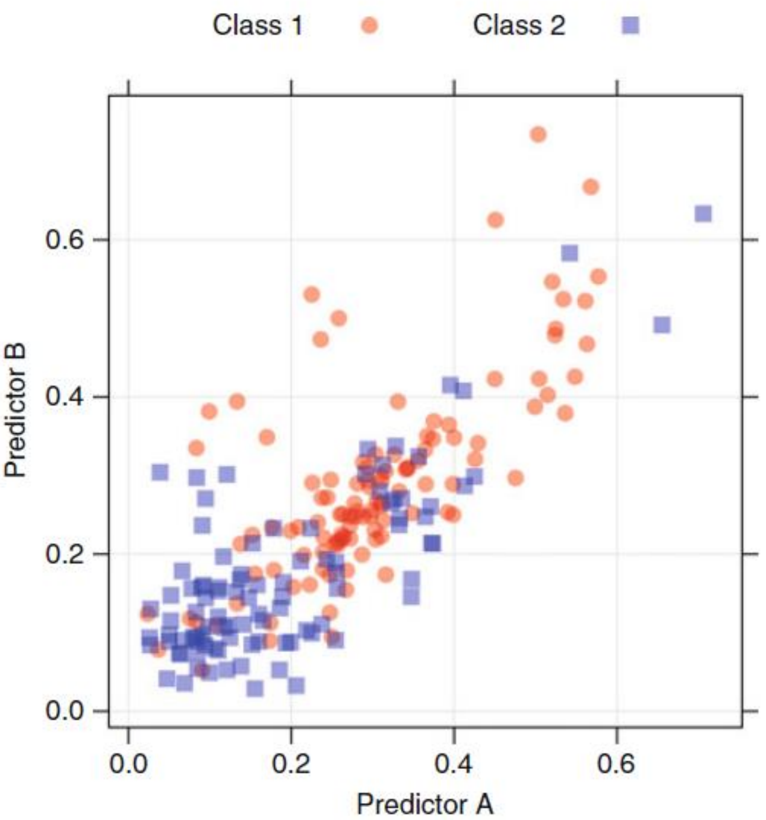


This slide is courtesy of *www.iro.umontreal.ca/~pift6080/documents/papers/****svm****_tutorial.****ppt***

Overfitting Problem

Overfitting Problem

- There now exist many techniques that can **learn the structure of a set of data so well** that when the **model is applied to the data on which the model was built**, it **correctly predicts every sample**.
- In addition to learning the general patterns in the data, the model **has also learned the characteristics of each sample's unique noise**.
- This type of model is said to be **over-fit** and **will usually have poor accuracy when predicting a new sample**.



Overfitting Problem

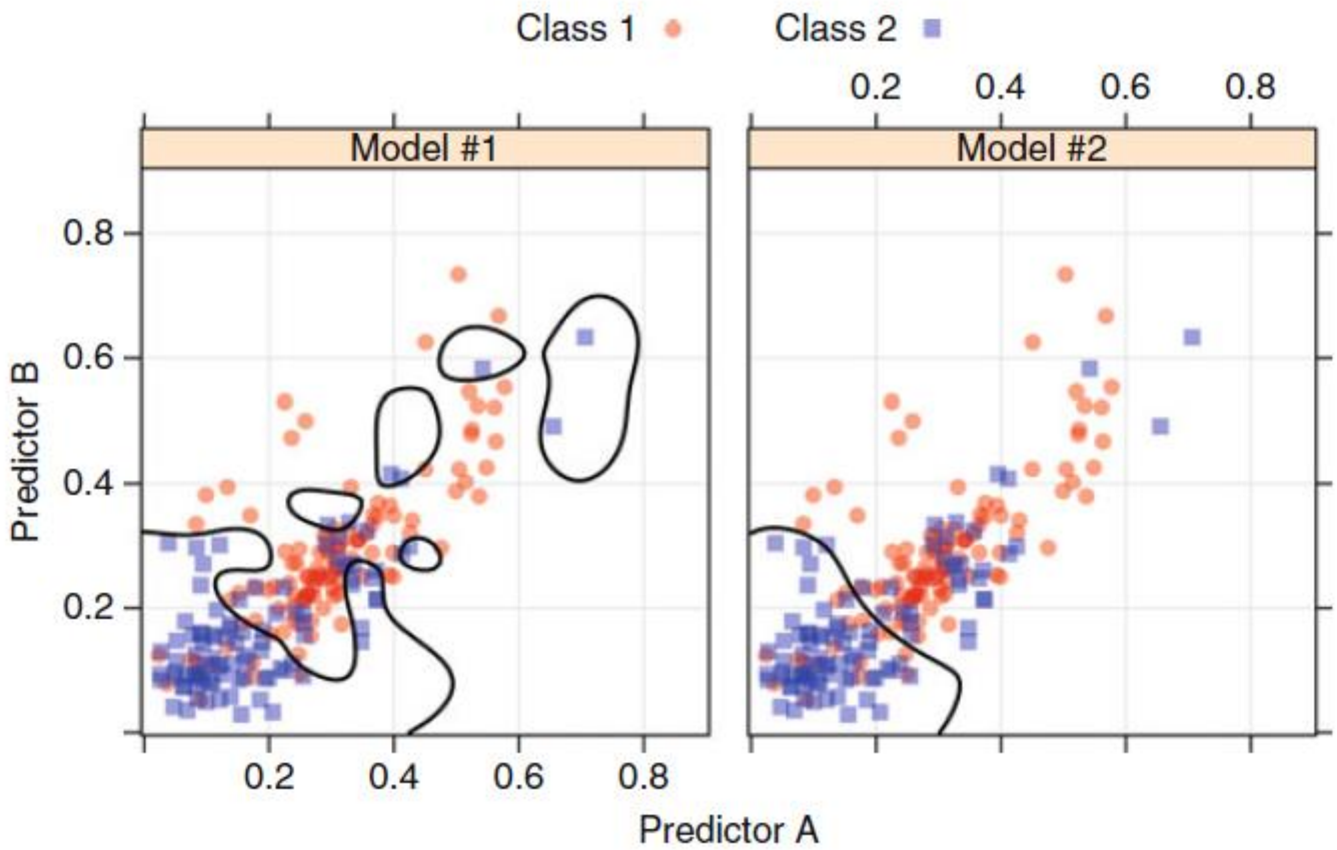


Fig. 4.2: An example of a training set with two classes and two predictors. The panels show two different classification models and their associated class boundaries

Overfitting Problem

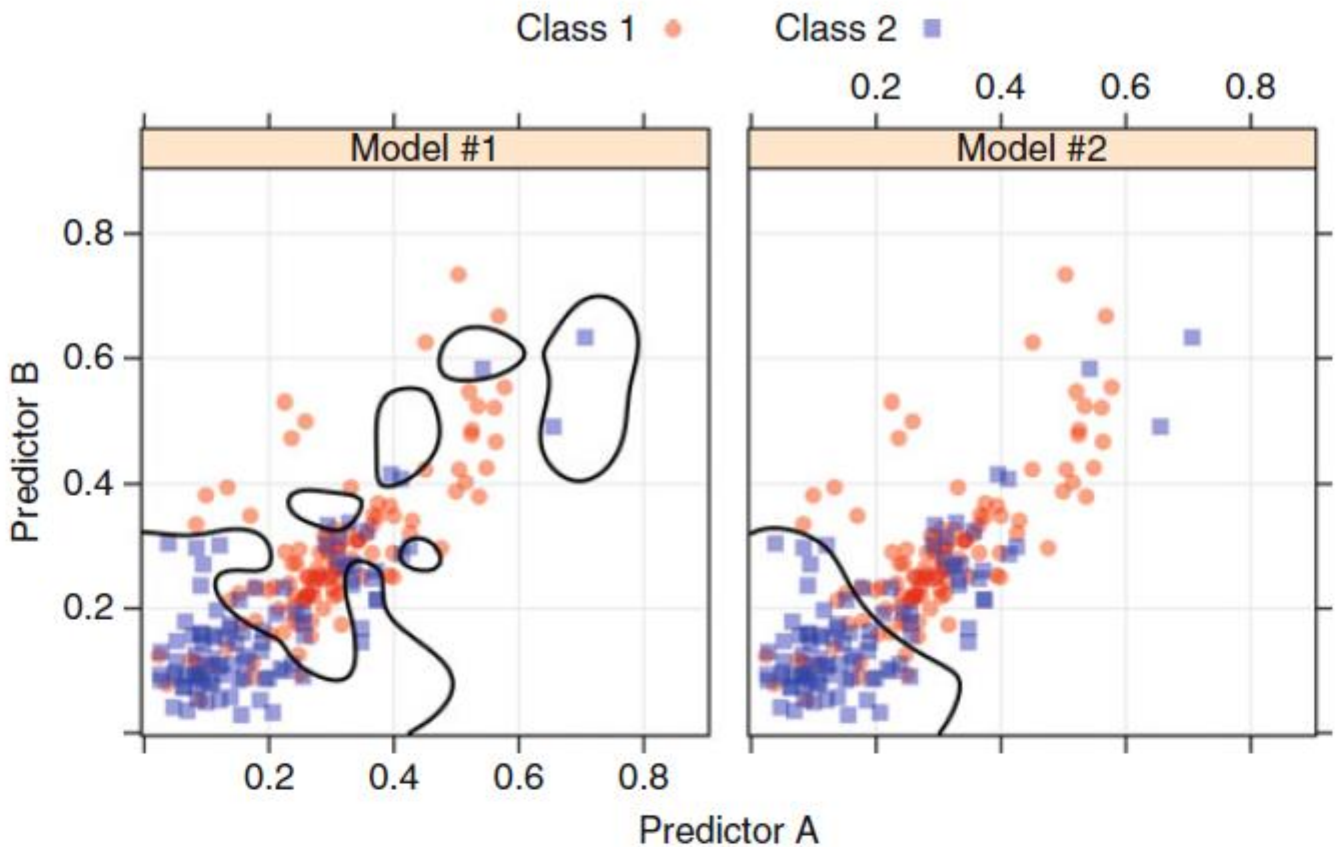
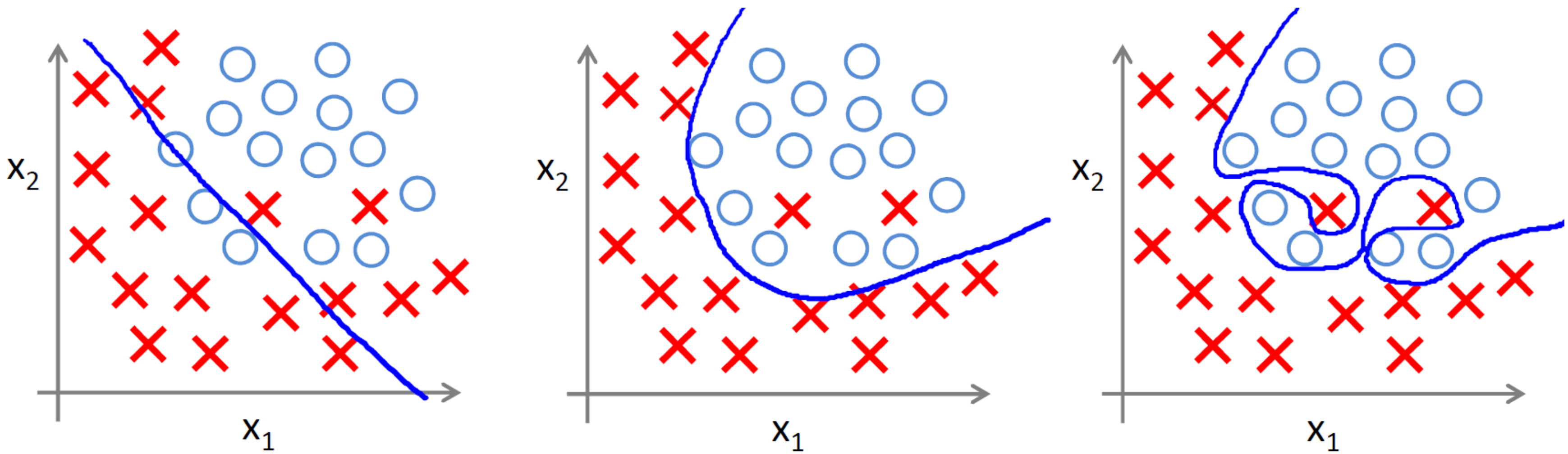


Fig. 4.2: An example of a training set with two classes and two predictors. The panels show two different classification models and their associated class boundaries

Overfitting problem

<u>Overfitting:</u> If we have too many features, the learned model may fit the training data very well but fail to generalize to new examples.



underfit- high bias model ok model overfit – high variance

Regularized logistic regression

Regularization

Regularization is a popular method in ML to prevent overfitting by reducing the model parameters θ towards zero

1 Ridge Regression (L2 norm)

— Keep all the features, but reduces the magnitude of θ.

— Works well when each of the features contributes a bit to predict y.

2 Lasso Regression (L1 norm)

- May shrink some coefficients of θ to exactly zero.
- Serve as a feature selection tools (reduces the number of features).

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left[-y^{(i)}\log\left(h_\theta\left(x^{(i)}\right)\right) - \left(1-y^{(i)}\right)\log\left(1-h_\theta\left(x^{(i)}\right)\right)\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}\left|\theta_j\right|$$

Regularized Logistic Regression

Unregularized Log Reg cost function:

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left[-y^{(i)}\log(h_\theta(x^{(i)})) - (1-y^{(i)})\log(1-h_\theta(x^{(i)}))\right]$$

Regularized Log Reg cost function (ridge regression)

λ is the regularization parameter (hyper-parameter) that needs to be selected

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left[-y^{(i)}\log(h_\theta(x^{(i)})) - (1-y^{(i)})\log(1-h_\theta(x^{(i)}))\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

Small λ => lower bias, higher variance

High λ => higher bias, lower variance

Regularized Logistic Regression

Unregularized cost function gradients (for all parameters j=0,1, ..n)

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

Regularized cost function gradient for j=0 (no change)

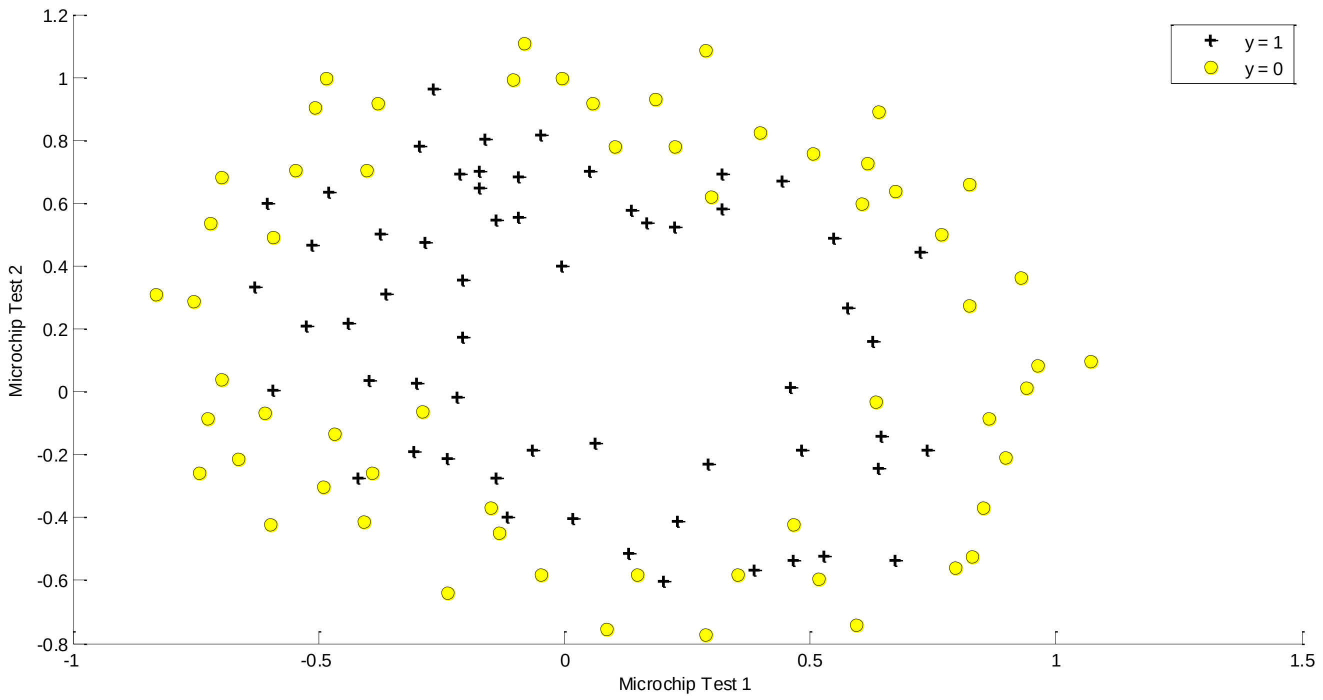
$$\frac{\partial J(\theta)}{\partial \theta_0} = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

Regularized cost function gradients for j=1,2,...n

$$\frac{\partial J(\theta)}{\partial \theta_j} = \left(\frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} \right) + \frac{\lambda}{m} \theta_j$$

Regularized Log Reg -example

Predict whether microchips from a fabrication plant passes quality assurance (QA). During QA, each microchip goes through various tests to ensure it is functioning correctly. Suppose we have the test results for some microchips on two different tests. From these two tests, we would like to determine whether the microchips should be accepted (y=1) or rejected (y=0).



universidade de aveiro

Regularized Log Reg -example

Dataset is not linearly separable => logistic regression will only be able to find a linear decision boundary. One way to fit the data better is to create more features. For example add polynomial terms of x1 and x2 .

$$\mathrm{mapFeature}(x) = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_1^2 \\ x_1 x_2 \\ x_2^2 \\ x_1^3 \\ \vdots \\ x_1 x_2^5 \\ x_2^6 \end{bmatrix}$$

NONLINEAR decision boundary $\Rightarrow$

$$z = \theta^T x = \theta_0 + \theta_1 x_1 + \theta_1 x_2 + \theta_3 x_1^2 + \theta_4 x_1 x_2 + \ldots\ldots\theta_{28} x_2^6 = 0$$

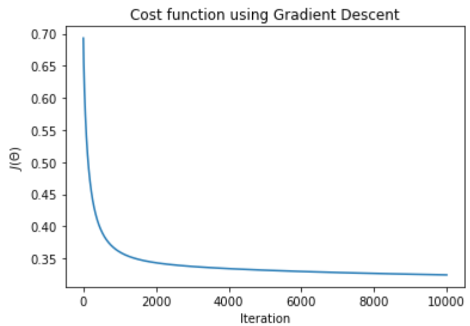
$$if \quad z > 0 => g(z) > 0.5 => \text{predict class} = 1$$

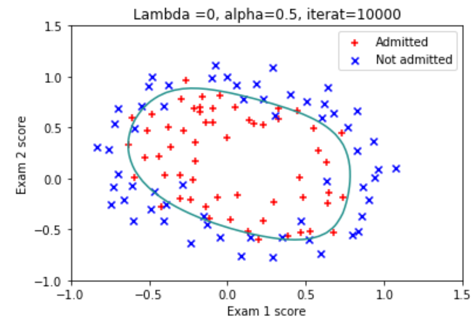
$$if \quad z < 0 => g(z) < 0.5 => \text{predict class} = 0$$

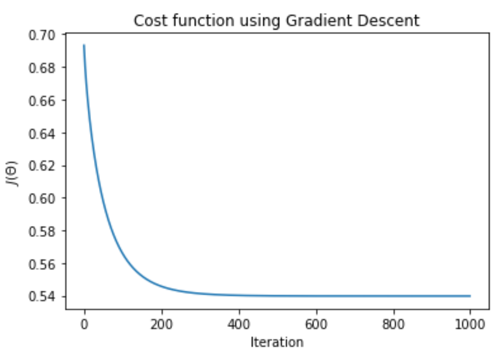
universidade de aveiro

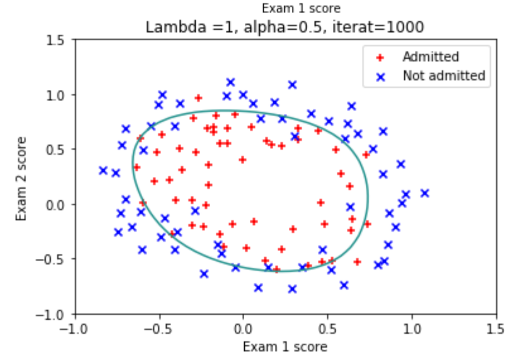
Regularized Log Reg -example

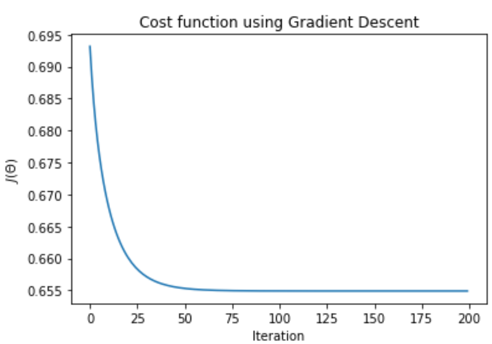
Accuracy on training data: :84.75% (λ =0) | 83.90 % (λ =1) | 71.2 % (λ =10))

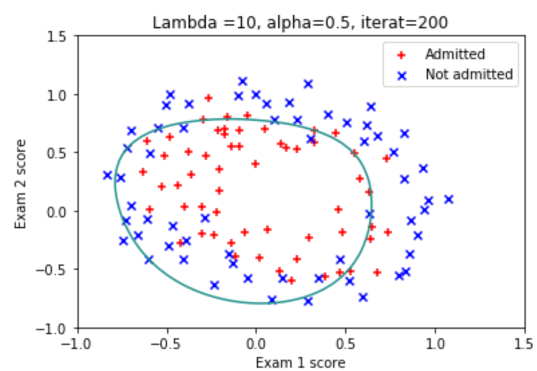










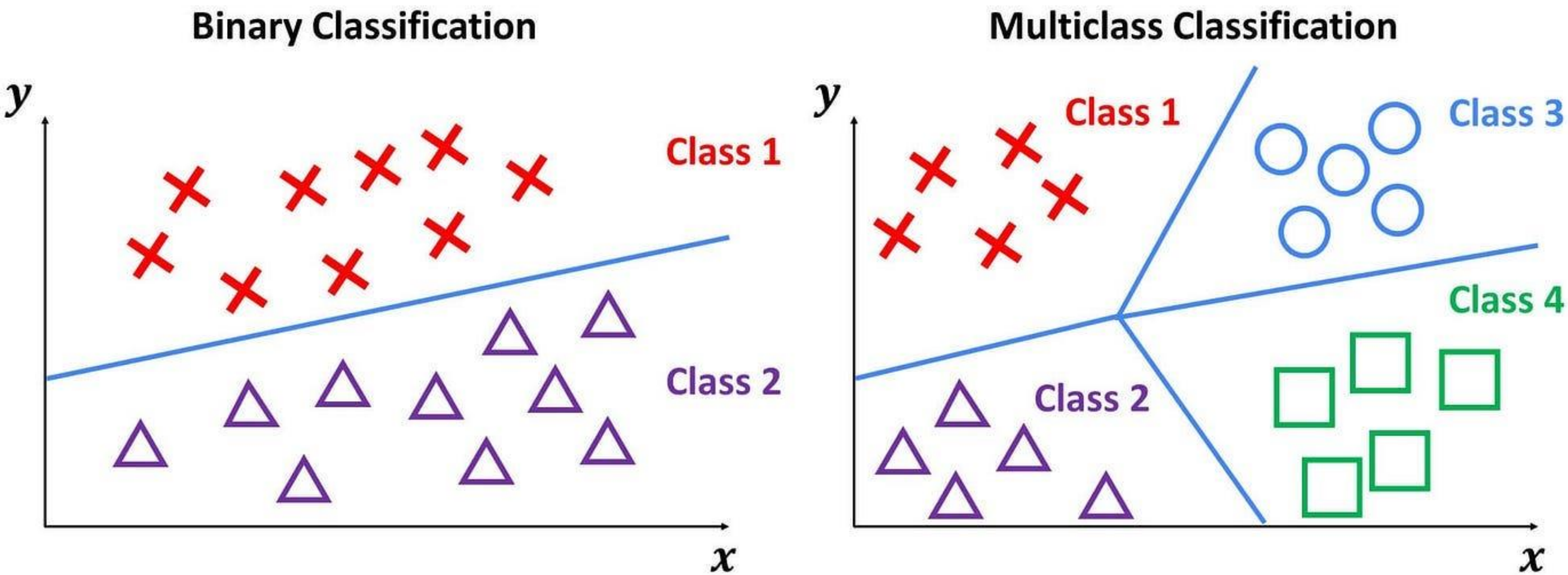


Multiclass

universidade
de aveiro

Multiclass Classification

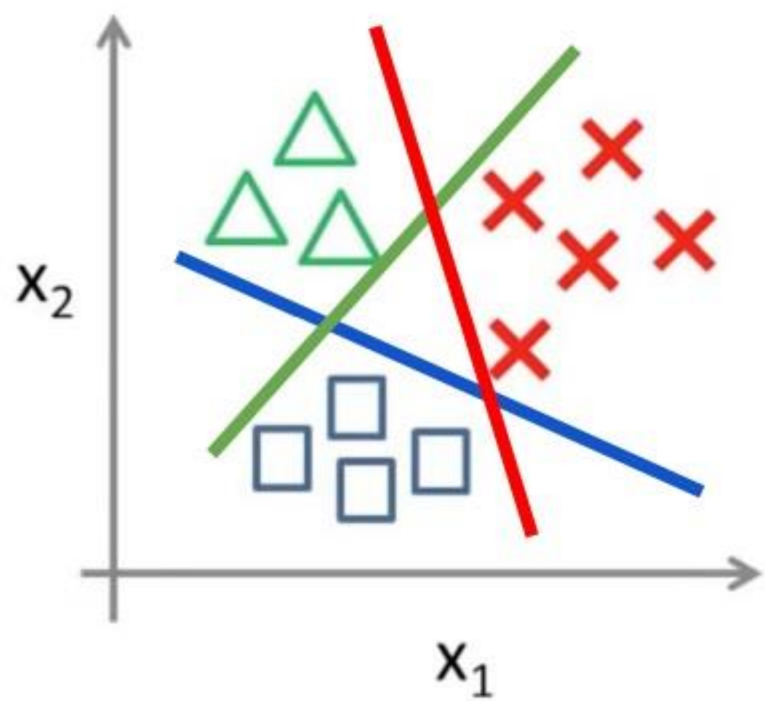
Logistic regression, by default, is limited to two-class classification problems.
How to deal with multiclass problem?



Multiclass Classification

Possible solutions, and common approaches:

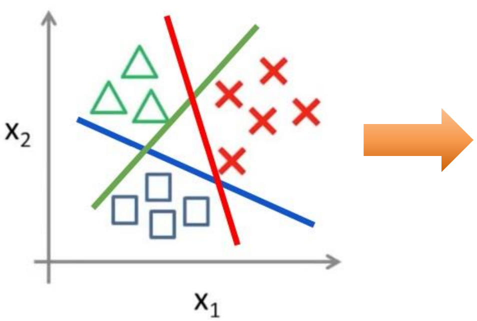
1. One vs All
2. Softmax Regression (Multimodal Logistic Regression)

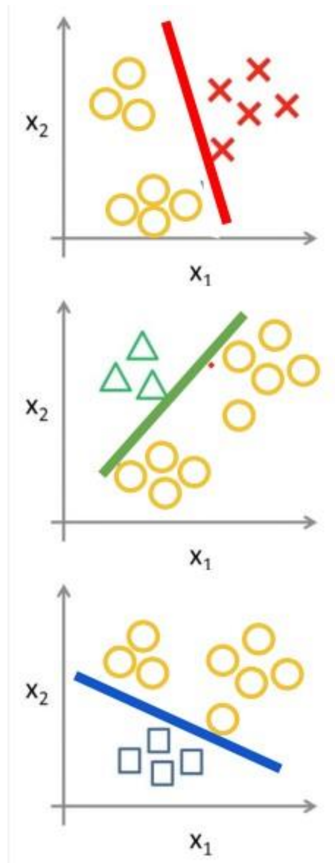


universidade de aveiro

Multiclass Classification
One vs All

1. For each class, define a logistic regression.
2. Calculate the probability of an observation to belong to that class.
3. Predict the class based on the highest probability.





P(Y=0|x)

P(Y=1|x)

P(Y=2|x)

Predict the class with the highest probability

Multiclass Classification

One vs All

For **K classes** train **K binary classifiers**:

for c=1:K

Make y_binary=1 (only for examples of class c)

y_binary=0 (for examples of all other classes)

theta=Train classifier with training data X and output y_binary.

Save the learned parameters of all classifiers in one matrix where each raw is the learned parameters of one classifier:

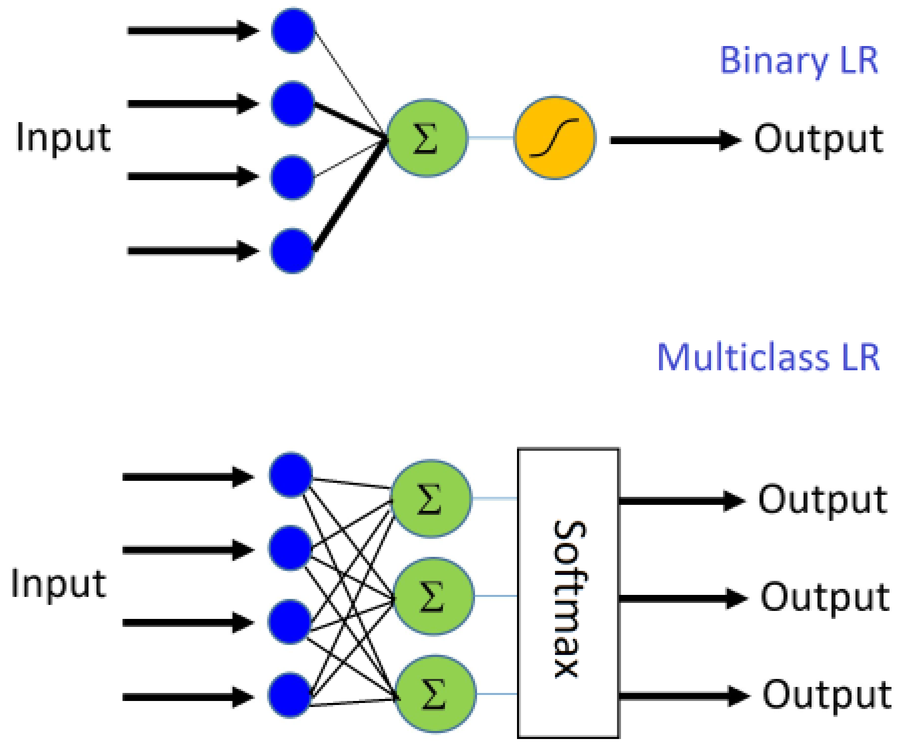
theta_all(c,:)=theta

End

New observation: winner-takes-all strategy, the binary classifier with the highest output score assigns the class.

Multiclass Classification

Softmax



Multiclass Classification

Softmax

In Softmax, the probability that an observation belongs to a class is determined by:

$$
\begin{bmatrix} P(Y=1|x;\theta) \\ P(Y=2|x;\theta) \\ \vdots \\ P(Y=K|x;\theta) \end{bmatrix} = \underbrace{\frac{1}{\sum_{j=1}^{K} \exp(\theta^{(j)\top} x)}}_{\text{Normalizes probabilities so they sum to 1.}} \begin{bmatrix} \exp(\theta^{(1)\top} x) \\ \exp(\theta^{(2)\top} x) \\ \vdots \\ \exp(\theta^{(K)\top} x) \end{bmatrix}
$$

Predict the class with the highest probability

Separate $\theta^{(j)} \in R^d$ for each class

Special case K=2, Softmax is simplified by the binary logistic regression.

Log Reg – Pros and Cons

Pros:

1. **Simplicity**: Easy to implement and understand.
2. **Interpretability**: The weights directly show the importance of each feature.
3. **Efficiency**: Doesn't require too much computational power.
4. **Probabilistic Output**: Provides probabilities rather than just classifications.

Cons:

1. **Linearity Assumption**: Assumes a linear relationship between features and log-odds of the outcome.
2. **Feature Independence**: Assumes features are not highly correlated.
3. **Limited Complexity**: May underfit in cases where the decision boundary is highly non-linear.
4. **Requires More Data**: Needs a relatively large sample size for stable results.