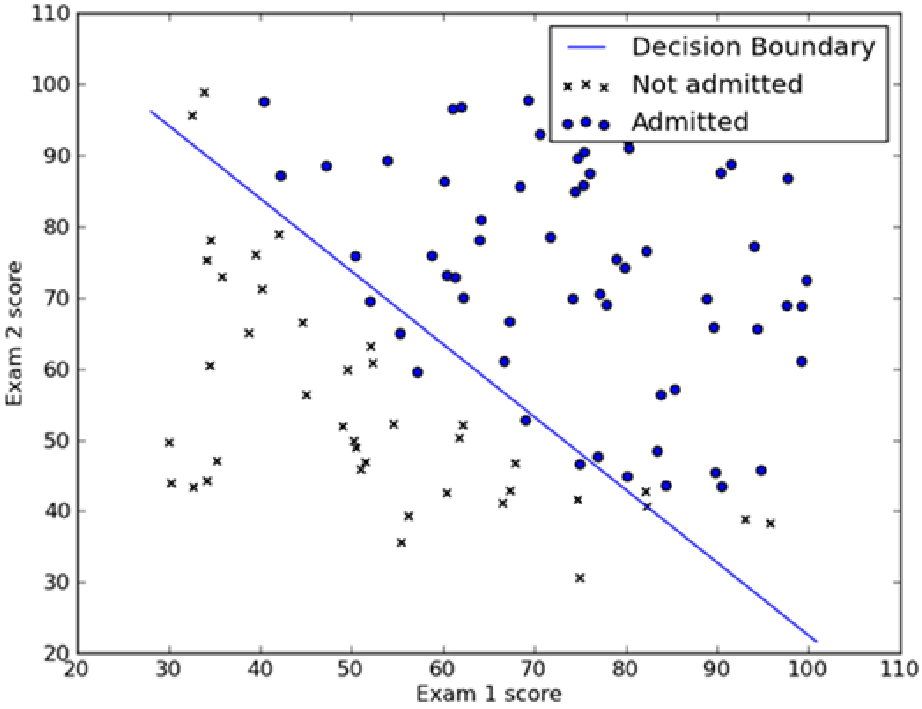
1 Logistic regression (for classification)

Categorize observations based on quantitative features. Predict target class or probabilities of target classes.



Example of a binary classification using a logictic regression model

1.1 Binary classification

- Description

  Logistic regression is a statistical method that studies the relationship between multiple variables:
    - n variables $x = \{x_1, x_2, ..., x_n\}$: the predictor, explanatory, independent variables,
    - one y variable: the response, outcome, dependent variable.

  Logistic regression expands the linear regression model with a *logistic function* to make it suitable for classification.
  Its dependent variable is therefore categorical instead of numerical.
  In case of a binary classification task, its dependent variable takes on one out of two possible values
    - $y \in \{0, 1\}$
    - 0 indicates the *negative* class
    - 1 indicates the *positive* class

- Model's hypothesis: output the estimated probability that $y = 1$ on input x

    - $z = \theta_0 + \theta_1 * x_1 + \theta_2 * x_2 + ... + \theta_n * x_n$

    - $h(x) = P(y = 1|x, \theta) = \sigma(z) = \frac{1}{1+e^{-z}} = \hat{y} \quad (0 <= h(x) <= 1)$

- Model's parameters (# $n + 1$)

  $\theta = \{\theta_0, \theta_1, ..., \theta_n\}$

- Decision boundary

    - linear

    - non-linear when adding extra higher-order polynomial terms to the features

- Cost function

$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \left(-y^{(i)} \; log \; \hat{y}^{(i)} - (1 - y^{(i)}) \; log \; (1 - \hat{y}^{(i)})\right)$

in other words
- $J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \left(- \; log \; \hat{y}^{(i)}\right)$ when $y^{(i)} = 1$
- $J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \left(- \; log \; (1 - \hat{y}^{(i)})\right)$ when $y^{(i)} = 0$

- Goal

  $minimize_{\theta} J(\theta)$

- Algorithm
  - gradient descent
    - start with some initial values for $\theta_0, \theta_1, ..., \theta_n$ (usually normal random values)
    - keep changing θs to reduce $J(\theta)$
      - $\theta_0 = \theta_0 - \alpha \frac{\partial J}{\partial \theta_0} = \theta_0 - \alpha \left(\frac{1}{m} \sum_{i=1}^{m} \left(\hat{y}^i - y^{(i)}\right)\right)$
      - $\theta_1 = \theta_1 - \alpha \frac{\partial J}{\partial \theta_1} = \theta_1 - \alpha \left(\frac{1}{m} \sum_{i=1}^{m} \left(\hat{y}^i - y^{(i)}\right) \cdot x_1^{(i)}\right)$
      - ...
      - $\theta_n = \theta_n - \alpha \frac{\partial J}{\partial \theta_n} = \theta_n - \alpha \left(\frac{1}{m} \sum_{i=1}^{m} \left(\hat{y}^i - y^{(i)}\right) \cdot x_n^{(i)}\right)$
- Hyperparameters:
  - α
- Problems:
  - the idea of feature scaling also aplied for logistic regression
  - make sure the gradient descent is working correctly
  - try adding new features if the model doesn't perform well
  - also use regularization in case of overfitting
    - cost function with the regularization term

      $J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \left(-y^{(i)} \; log \; \hat{y}^{(i)} - (1 - y^{(i)}) \; log \; (1 - \hat{y}^{(i)})\right) + \frac{\lambda}{2m} \sum_{j=1}^{n} \theta_j^2$
    - parameter update in gradient descent (for $j \in \{1, 2, \ldots, n\}$)

      $\theta_j = \theta_j - \alpha \frac{\partial J}{\partial \theta_j} = \theta_j - \alpha \left(\frac{1}{m} \sum_{i=1}^{m} \left(\hat{y}^i - y^{(i)}\right) \cdot x_j^{(i)} + \frac{\lambda}{m} \theta_j\right) = \theta_j \left(1 - \alpha \frac{\lambda}{m}\right) - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(\hat{y}^i - y^{(i)}\right) \cdot x_j^{(i)}$
    - intuition: *shrink* model parameters in order to *smooth out* the decision boundary
    - λ is the regularization parameter and needs to be tuned

1.2 Multiclass classification

- Description
  - use the one-vs-all (one-vs-rest) approach
  - turn the problem into C binary classification problems (generate C decision boundaries)
  - formally: train a logistic regression classifier $h^{(i)}(x)$ for each class i to predict the probability that $y = i$
  - on a new input x, in order to make a prediction pick the class i that maximizes $max_i h^{(i)}(x)$