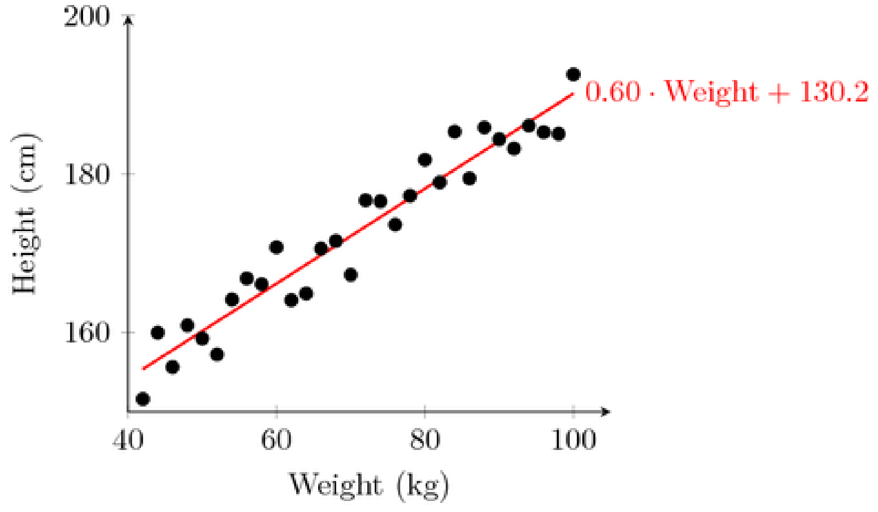
1 Linear regression (for regression)

Model a scalar target with one or more quantitative features.
Although regression computes a linear combination, features can be transformed by nonlinear functions if relationships are known or can be guessed.



Example of prediction using a linear regression model

1.1 Univariate linear regression

- Description

  Simple linear regression is a statistical method that studies the relationship between two variables:
    - x: the predictor, explanatory, independent variable,
    - y: the response, outcome, dependent variable.

- Model's hypothesis

  $h(x) = \theta_0 + \theta_1 * x = \hat{y}$

- Model's parameters (# 2)
    - θ_0
    - θ_1

- Cost function (in this case, the squared error function)

  $J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(\hat{y}^i - y^{(i)}\right)^2$

- Goal

  $minimize_{\theta_0, \theta_1} J(\theta_0, \theta_1)$

- Algorithm

    - gradient descent

        - start with some initial values for θ_0 and θ_1 (usually zero)

        - keep changing θ_0 and θ_1 to reduce $J(\theta_0, \theta_1)$

        - $\theta_0 = \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \theta_0 - \alpha \left(\frac{1}{m} \sum_{i=1}^{m} \left(\hat{y}^i - y^{(i)} \right) \right)$
        - $\theta_1 = \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) = \theta_1 - \alpha \left(\frac{1}{m} \sum_{i=1}^{m} \left(\hat{y}^i - y^{(i)} \right) \cdot x^{(i)} \right)$
    - directly use the following **formulas**
        - $\theta_1 = \frac{\sum_{i=1}^{m} (x^{(i)} - \bar{x})(y^{(i)} - \bar{y})}{\sum_{i=1}^{m} (x^{(i)} - \bar{x})^2}$
        - $\theta_0 = \bar{y} - \theta_1 \bar{x}$
- Notations
    - x: the independent variable
    - y: the dependent variable
    - m: the number of training examples
    - $x^{(i)}$: the input value of the i^{th} training example
    - $y^{(i)}$: the target value of the i^{th} training example
    - $\hat{y}^{(i)}$: the prediction made on the i^{th} training example by the current hypothesis function
    - α: the learning rate; determines how big steps we take when updating the θ parameters
- Hyperparameters:
    - α: if too small, slow gradient descent; if too large, gradient descent may fail to converge
- Problems:
    - gradient descent may converge to a local minimum

1.2 Multivariate linear regression

- Description

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

  $h(x) = \theta_0 + \theta_1 * x_1 + \theta_2 * x_2 + \ldots + \theta_n * x_n = \hat{y}$
- Model's parameters (# $n + 1$)

  $\theta = \{\theta_0, \theta_1, \ldots, \theta_n\}$
- Cost function (in this case, the squared error function)

  $J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(\hat{y}^{(i)} - y^{(i)} \right)^2$
- Goal

  $minimize_\theta J(\theta)$
- Algorithm
    - gradient descent

        - start with some initial values for $\theta_0, \theta_1, ..., \theta_n$ (usually 0)
        - keep changing θs to reduce $J(\theta)$
            - $\theta_0 = \theta_0 - \alpha \frac{\partial J}{\partial \theta_0} = \theta_0 - \alpha\left(\frac{1}{m}\sum_{i=1}^{m}\left(\hat{y}^i - y^{(i)}\right)\right)$
            - $\theta_1 = \theta_1 - \alpha \frac{\partial J}{\partial \theta_1} = \theta_1 - \alpha\left(\frac{1}{m}\sum_{i=1}^{m}\left(\hat{y}^i - y^{(i)}\right) \cdot x_1^{(i)}\right)$
            - ...
            - $\theta_n = \theta_n - \alpha \frac{\partial J}{\partial \theta_n} = \theta_n - \alpha\left(\frac{1}{m}\sum_{i=1}^{m}\left(\hat{y}^i - y^{(i)}\right) \cdot x_n^{(i)}\right)$

- Notations
    - x: the independent variables
    - y: the dependent variable
    - m: the number of training examples
    - n: the number of features representing each training example
    - $x^{(i)}$: the input values of the i^{th} training example
    - $x_j^{(i)}$: the value of the j^{th} feature of the i^{th} training example
    - $y^{(i)}$: the target value of the i^{th} training example
    - $\hat{y}^{(i)}$: the prediction made on the i^{th} training example by the current hypothesis function
    - α: the learning rate; determines how big steps we take when updating the θ parameters
- Hyperparameters:
    - α
- Problems:
    - make sure features are on similar scales; gradient descent may be slow otherwise
        - rescaling: $x' = \frac{x - min(x)}{max(x) - min(x)}$
        - mean normalization: $x' = \frac{x - mean(x)}{max(x) - min(x)}$
        - standardization: $x' = \frac{x - mean(x)}{std(x)}$
    - make sure the gradient descent is working correctly
        - plot the value of the cost function J over the number of iterations (# *epochs*)
        - for a sufficiently small α, $J(\theta)$ should decrease on every iteration
        - if α is too small, the gradient descent can be slow to converge
        - if α is too large, $J(\theta)$ may not decrease on every iteration; may not converge
        - try values $\alpha \in \{. . . , 0.001, 0.01, 0.1, 1, . . . \}$
    - try adding new features if the model doesn't perform well
        - e.g, use a polynomial regression

          $$\theta_0 + \theta_1 * x \quad \Rightarrow \quad \theta_0 + \theta_1 * x + \theta_2 * x^2 + \theta_3 * x^3$$
    - use regularization in case of overfitting

- cost function with the regularization term

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

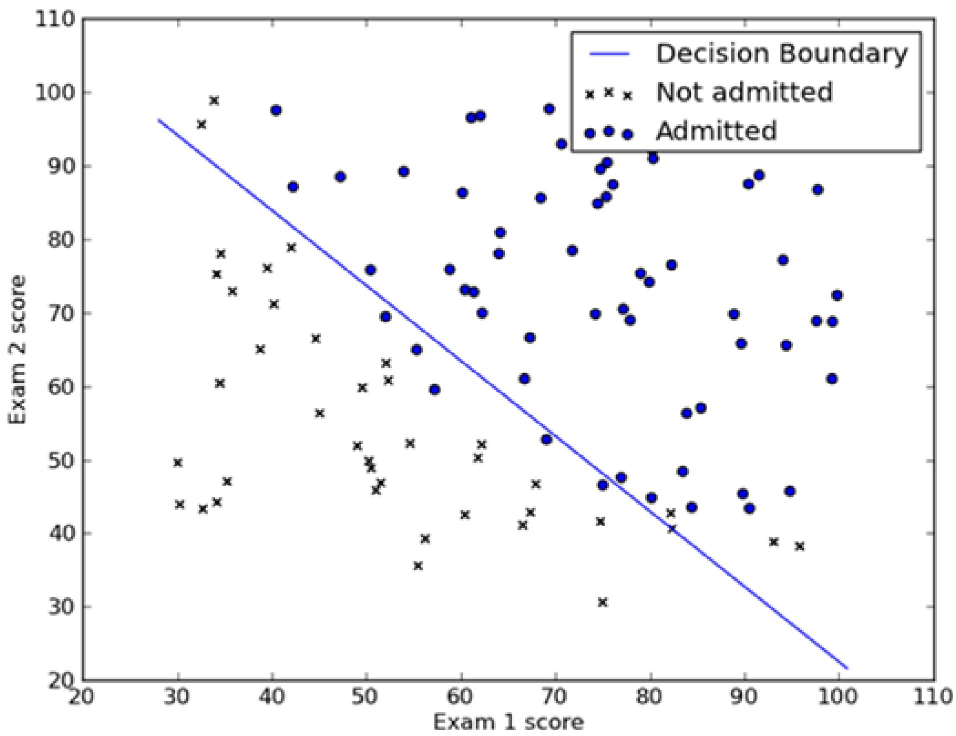
- parameter update in gradient descent (for $j \in \{1, 2, \ldots, n\}$)

  $$\theta_j = \theta_j - \alpha \frac{\partial J}{\partial \theta_j} = \theta_j - \alpha \left(\frac{1}{m} \sum_{i=1}^{m} \left(\hat{y}^i - y^{(i)}\right) \cdot x_j^{(i)} + \frac{\lambda}{m} \theta_j \right) = \theta_j \left(1 - \alpha \frac{\lambda}{m}\right) - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(\hat{y}^i - y^{(i)}\right) \cdot x_j^{(i)}$$

- intuition: *shrink* model parameters in order to *smooth out* the decision boundary
- λ is the regularization parameter and needs to be tuned

2 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

2.1 Binary classification

- Description

  Logistic regression is a statistical method that studies the relationship between multiple variables:
  - n variables $x = \{x_1, x_2, \ldots, x_n\}$: 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 $j \in \{1, 2, ..., n\}$)

$$\theta_j = \theta_j - \alpha\frac{\partial J}{\partial \theta_j} = \theta_j - \alpha\left(\frac{1}{m}\sum_{i=1}^{m}\left(\hat{y}^i - y^{(i)}\right)\cdot x_j^{(i)} + \frac{\lambda}{m}\theta_j\right) = \theta_j\left(1-\alpha\frac{\lambda}{m}\right) - \alpha\frac{1}{m}\sum_{i=1}^{m}\left(\hat{y}^i - y^{(i)}\right)\cdot x_j^{(i)}$$

  - intuition: *shrink* model parameters in order to *smooth out* the decision boundary
  - λ is the regularization parameter and needs to be tuned

2.2 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 $max_i h^{(i)}(x)$