Introduction to Machine Learning
NPFL 054

http://ufal.mff.cuni.cz/course/npfl054

Barbora Hladká
hladka@ufal.mff.cuni.cz

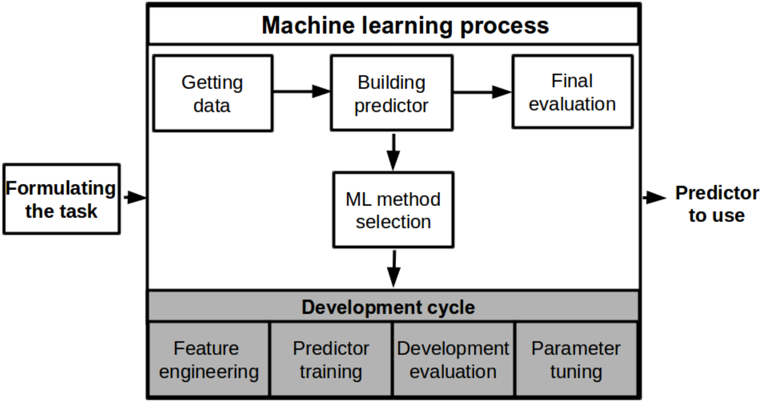
Martin Holub
holub@ufal.mff.cuni.cz

Charles University,
Faculty of Mathematics and Physics,
Institute of Formal and Applied Linguistics

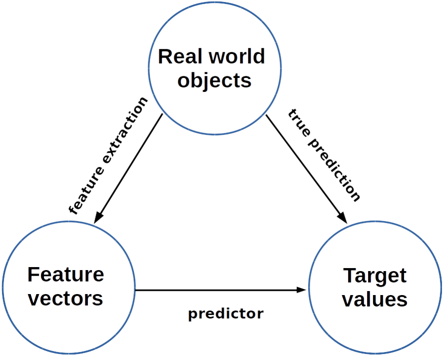
Outline

- **Linear regression**
  - `Auto` data set
- **Logistic regression**
  - `Auto` data set

Machine learning process and development cycle



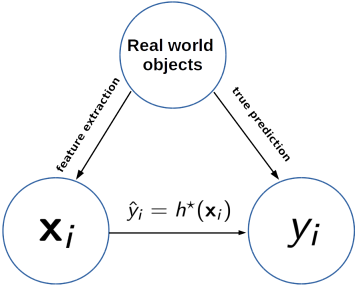
Machine learning as building a prediction function



- if target values are *continuous* numbers, we speak about **regression**
  = estimating or predicting a continuous response
- if target values are *discrete/categorical*, we speak about **classification**
  = identifying group membership

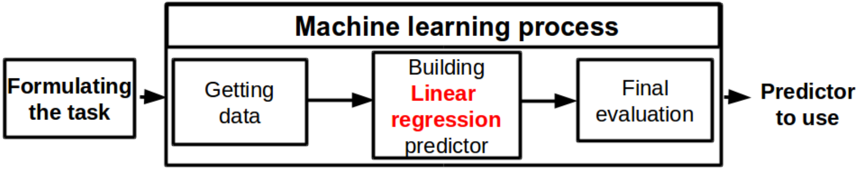
Prediction function and its relation to the data

Idealized model of supervised learning



- $\mathbf{x}_i$ are **feature vectors**, y_i are true **predictions**
- **prediction function** $h^\star$ is the "best" of all possible hypotheses h
- **learning process** is searching for $h^\star$, which means to search the **hypothesis space** and minimize a predefined **loss function**
- ideally, the learning process results in $h^\star$ so that predicted $\hat{y}_i = h^\star(\mathbf{x}_i)$ is equal to the true target values y_i

Linear regression

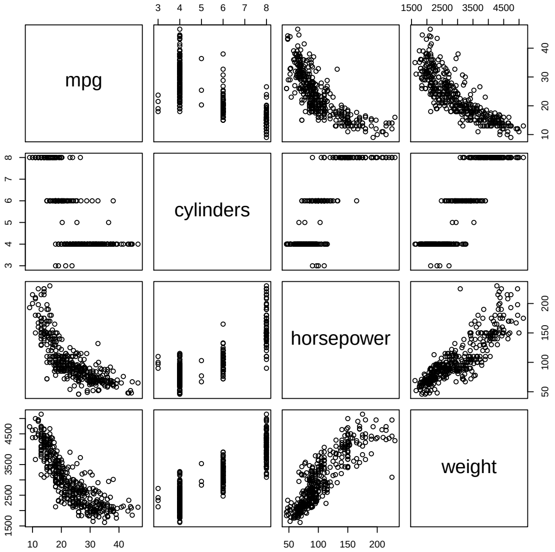


Dataset Auto from the ISLR package

392 instances on the following 9 features

mpg	Miles per gallon
cylinders	Number of cylinders between 4 and 8
displacement	Engine displacement (cu. inches)
horsepower	Engine horsepower
weight	Vehicle weight (lbs.)
acceleration	Time to accelerate from 0 to 60 mph (sec.)
year	Model year (modulo 100)
origin	Origin of car (1. American, 2. European, 3. Japanese)
name	Vehicle name

Dataset Auto from the ISLR package



Linear regression

h has a form of **linear function**

$$h(\mathbf{x}) = \Theta_0 + \Theta_1 x_1 + \dots \Theta_m x_m = \Theta_0 + \langle \Theta_1, \dots, \Theta_m \rangle^T \mathbf{x} \tag{1}$$

Linear regression is a **parametric method**.

We estimate $m + 1$ parameters ($\mathbf{\Theta}$) instead of fitting data with an entirely arbitrary function h.

Linear regression

Notation

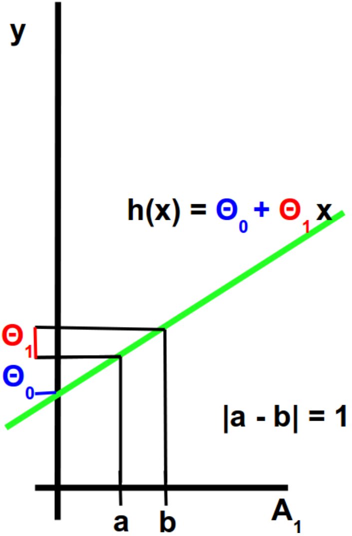
$$\mathbf{y} = \begin{pmatrix} y_1 \\ \dots \\ y_n \end{pmatrix}, \ \mathbf{\Theta} = \begin{pmatrix} \Theta_0 \\ \dots \\ \Theta_m \end{pmatrix}, \ \mathbf{X} = \begin{pmatrix} 1 & x_{11} & \dots & x_{1m} \\ 1 & x_{21} & \dots & x_{2m} \\ \dots & \dots & \dots & \dots \\ 1 & x_{n1} & \dots & x_{nm} \end{pmatrix}$$

$$\mathbf{y} = \mathbf{X}\mathbf{\Theta}$$

Simple linear regression

Simple regression is a linear regression with a single feature.

- $Attr = \{A_1\}$
- $\mathbf{x} = \langle x_1 \rangle$
- $h(\mathbf{x}) = \Theta_0 + \Theta_1 x_1$
- Θ_1 is the average change in y for a unit change in A_1, if A_1 is a continuous feature



Simple linear regression

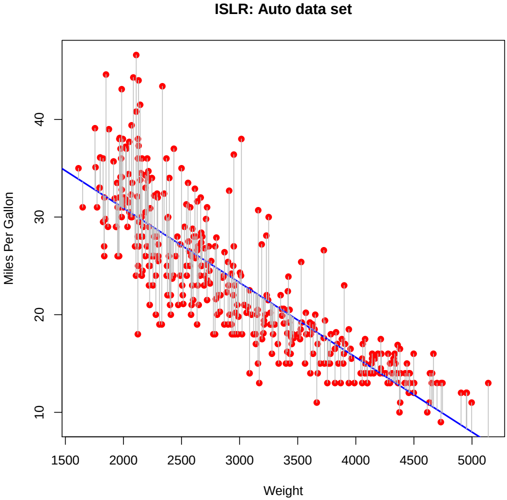
How to choose parameters Θ_0 and Θ_1?

Idea: Choose them so that $h(\mathbf{x})$ is close to y for training examples $\langle \mathbf{x}, y \rangle$

How to measure closeness? Using e.g., the **least squares criterion**

Least squares criterion

- Residual $e_i = y_i - h(\mathbf{x}_i)$
- Residual sum of squares $RSS(h) = \sum_{i=1}^{n} e_i^2$



Simple linear regression

Hypothesis $h(\mathbf{x}) = \Theta_0 + \Theta_1 x_1$
Hypothesis parameters $\mathbf{\Theta} = \langle \Theta_0, \Theta_1 \rangle$

- **Loss function**

$$\mathrm{L}(\mathbf{\Theta}) = RSS = (\mathbf{X\Theta} - \mathbf{y})^2 \tag{2}$$

- **Optimization task**

$$\mathbf{\Theta}^\star = \mathrm{argmin}_{\mathbf{\Theta}} \mathrm{L}(\mathbf{\Theta}) \tag{3}$$

The `argmin` operator will give $\mathbf{\Theta}$ for which $\mathrm{L}(\mathbf{\Theta})$ is minimal.

Simple linear regression
Solving the loss function analytically

- Find the pair (Θ_0, Θ_1) that minimizes $\mathrm{L}(\mathbf{\Theta}) = \sum_{i=1}^{n}(y_i - \Theta_0 - \Theta_1 \mathbf{x}_i)^2$

  - $\frac{\partial \mathrm{L}(\mathbf{\Theta})}{\partial \Theta_0} = -\sum_{i=1}^{n} 2(y_i - \Theta_0 - \Theta_1 \mathbf{x}_i)$
    $-\sum_{i=1}^{n} 2(y_i - \Theta_0 - \Theta_1 \mathbf{x}_i) = 0 \implies \underline{\sum_{i=1}^{n}(y_i - \Theta_0 - \Theta_1 \mathbf{x}_i) = 0}$

  - $\frac{\partial \mathrm{L}(\mathbf{\Theta})}{\partial \Theta_1} = -\sum_{i=1}^{n} 2\mathbf{x}_i(y_i - \Theta_0 - \Theta_1 \mathbf{x}_i)$
    $-\sum_{i=1}^{n} 2\mathbf{x}_i(y_i - \Theta_0 - \Theta_1 \mathbf{x}_i) = 0 \implies \underline{\sum_{i=1}^{n} \mathbf{x}_i(y_i - \Theta_0 - \Theta_1 \mathbf{x}_i) = 0}$

  - Using the Normal equations calculus (see below), the minimizers are
    - $\Theta_1 = \frac{\sum_{i=1}^{n}(\mathbf{x}_i - \overline{x})(y_i - \overline{y})}{\sum_{i=1}^{n}(\mathbf{x}_i - \overline{x})^2}$,
    - $\Theta_0 = \overline{y} - \Theta_1 \overline{x}$

    where $\overline{y} = \frac{1}{n}\sum_{i=1}^{n} y_i, \overline{x} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x_i}$

Simple linear regression
Solving the loss function analytically

Normal Equations Calculus

Find $\mathbf{\Theta}$ that minimizes $\mathbf{e} = \mathbf{y} - \mathbf{X}\mathbf{\Theta}$

Theorem

$\mathbf{\Theta}^\star$ *is a least squares solution to* $\mathbf{y} = \mathbf{X}\mathbf{\Theta} \Leftrightarrow \mathbf{\Theta}^\star$ *is a solution to the Normal equation* $\mathbf{X}^T\mathbf{X}\mathbf{\Theta} = \mathbf{X}^T\mathbf{y}$.

Simple linear regression
Solving the loss function analytically

Normal Equations Calculus

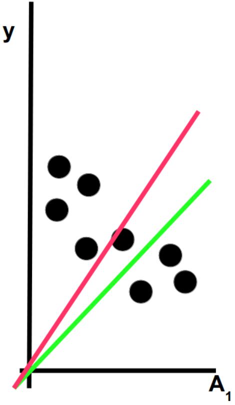
- $\mathbf{\Theta}^\star = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$
- Thus we work with a system of $(m+1)$ equations in $(m+1)$ unknowns.
- The term "normal equations" derives from the fact that the solution $\mathbf{\Theta}$ satisfies at $\mathbf{X}^T(\mathbf{y} - \mathbf{X\Theta}) = 0$ where the residual vector $\mathbf{y} - \mathbf{X\Theta}$ is a normal to the columns of $\mathbf{X}$.
- (Two non-zero vectors $\mathbf{a}$ and $\mathbf{b}$ are orthogonal if and only if $\mathbf{ab} = 0$.)

Simple linear regression
Solving the loss function analytically

- When using the Normal equations for solving the cost function analytically one has to compute $(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$
- But it is computationally expensive:-(calculating the inverse of a $(m+1) \times (m+1)$ matrix is $O(m+1)^3$ and as m increases it can take a very long time to finish.
- When m is low one can think of Normal equations as the better option for calculation $\mathbf{\Theta}$, however for greater values the **Gradient Descent Algorithm** is much more faster.

Simple linear regression
Gradient Descent Algorithm

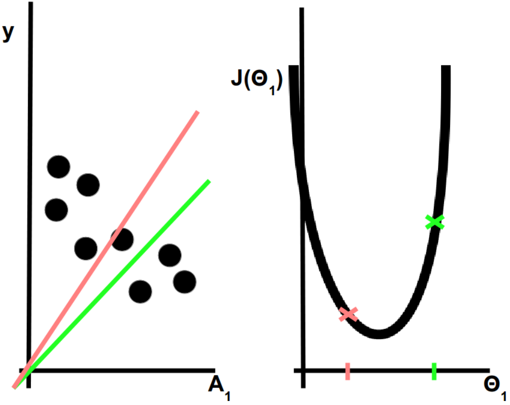
Simplification: $\Theta_0 = 0$, $\Theta_1 \neq 0$



Hypothesis	$h(\mathbf{x}) = \Theta_1 x_1$
Hypothesis parameters	Θ_1 $(\boldsymbol{\Theta} = \langle 0, \Theta_1 \rangle)$
Loss function	$\mathrm{L}(\boldsymbol{\Theta}) = RSS$
Optimization task	$\Theta_1^\star = \mathrm{argmin}_{\Theta_1} \mathrm{L}(\boldsymbol{\Theta})$

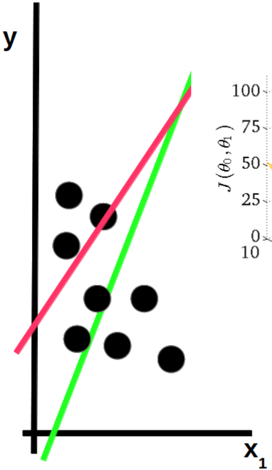
Simple linear regression
Gradient Descent Algorithm

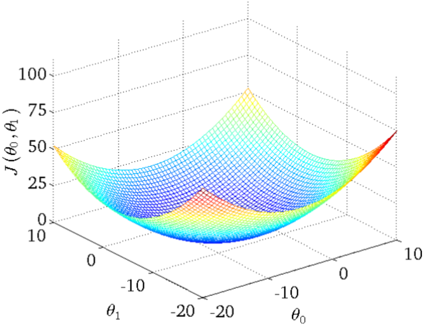
$\Theta_0 = 0$, $\Theta_1 \neq 0$



Simple linear regression
Gradient Descent Algorithm

$\Theta_0 \neq 0, \Theta_1 \neq 0$

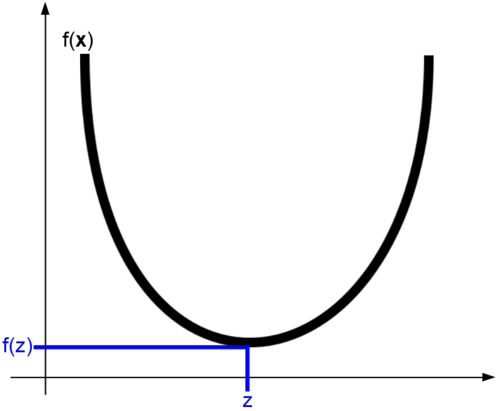




Credits: Andrew Ng

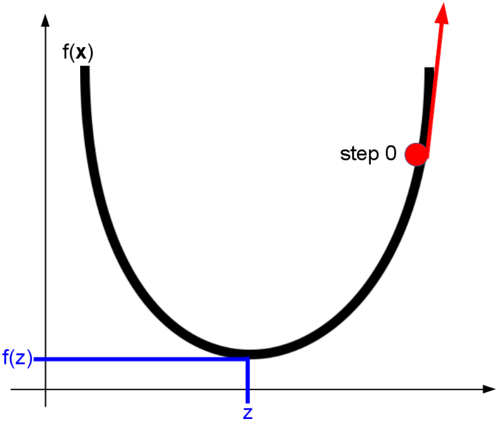
Gradient Descent Algorithm

Gradient descent algorithm is an optimization algorithm to find a local minimum of a function f.



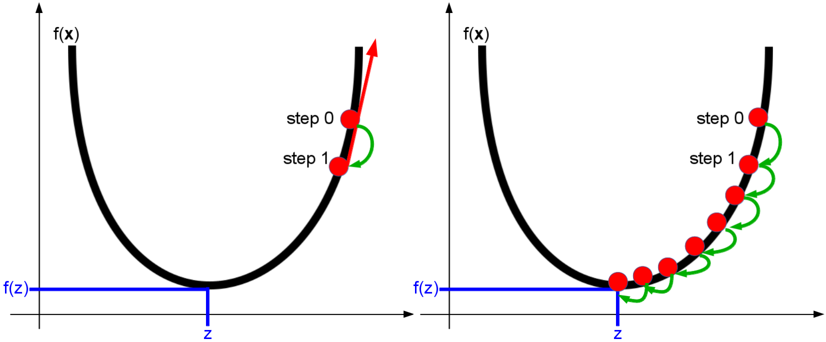
Gradient Descent Algorithm

1. Start with some $\mathbf{x}_0$.

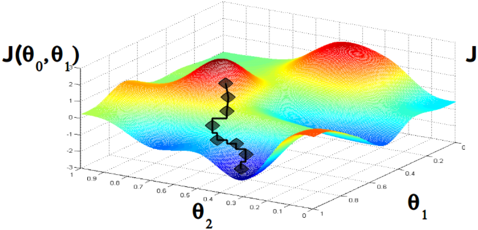


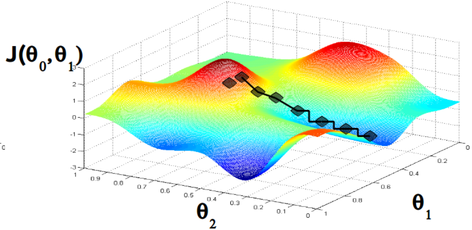
Gradient Descent Algorithm

2. Keep changing $\mathbf{x}_i$ to reduce $f(\mathbf{x}_i)$ until you end up at a minimum.



Gradient Descent Algorithm





Credits: Andrew Ng

Gradient Descent Algorithm

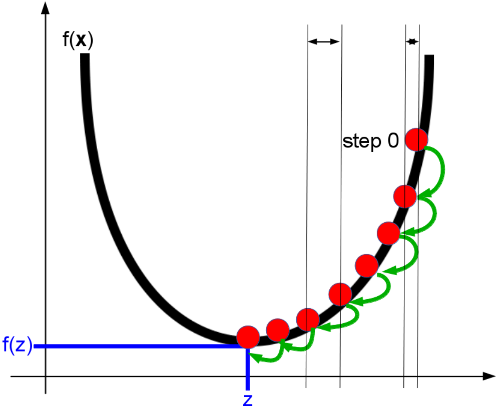
- We are seeking the solution to the minimum of a function $f(\mathbf{x})$. Given some initial value $\mathbf{x}_0$, we can change its value in many directions.
- What is the best direction to minimize f? We take the **gradient** ∇f of f

$$\nabla f(x_1, x_2, \ldots, x_m) = \langle \frac{\partial f(x_1, x_2, \ldots, x_m)}{\partial x_1}, \ldots, \frac{\partial f(x_1, x_2, \ldots, x_m)}{\partial x_m} \rangle \tag{4}$$

- Intuitively, the gradient of f at any point tells which direction is the steepest from that point and how steep it is. So we change $\mathbf{x}$ in the opposite direction to lower the function value.

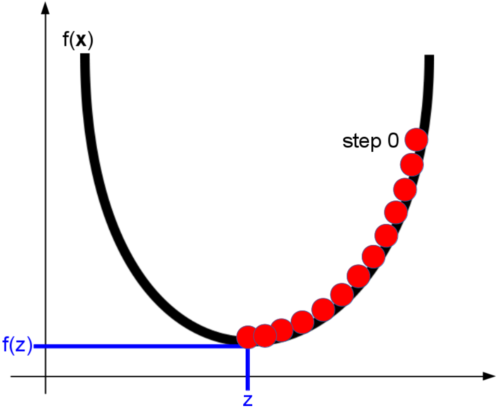
Gradient Descent Algorithm

Choice of the step



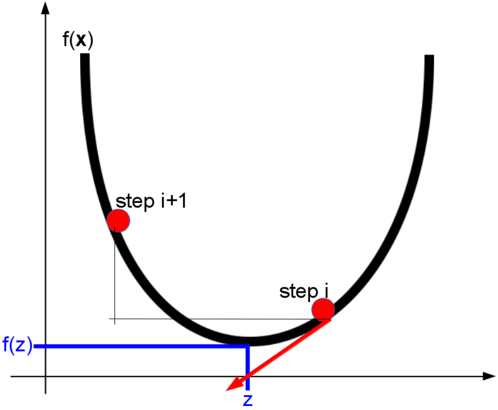
Gradient Descent Algorithm

Choice of the step



Gradient Descent Algorithm

Choice of the step



Gradient Descent Algorithm

repeat until convergence $\{$

$$\Theta_j := \Theta_j - \alpha \frac{\partial \mathrm{L}(\Theta_0, \Theta_1)}{\partial \Theta_j}, j = 0, 1 \tag{5}$$

(simultaneously update Θ_j for $j = 0$ and $j = 1$)
$\}$

– α is a positive step-size parameter that controls how big step we'll take downhill

Gradient Descent Algorithm

- If α is too large, GDA can overshoot the minimum. It may fail to converge, or even diverge.

- If α is too small, GDA can be slow.

Simple linear regression
Gradient Descent Algorithm

$$\frac{\partial \mathrm{L}(\Theta_0,\Theta_1)}{\partial \Theta_j} = \frac{\partial \sum_{i=1}^{n}(h(\mathbf{x}_i)-y_i)^2}{\partial \Theta_j} = \frac{\partial \sum_{i=1}^{n}(\Theta_0+\Theta_1 x_{i_1}-y_i)^2}{\partial \Theta_j}$$

- $j = 0 : \frac{\partial \mathrm{L}(\Theta_0,\Theta_1)}{\partial \Theta_0} = \sum_{i=1}^{n}(h(\mathbf{x}_i) - y_i)$
- $j = 1 : \frac{\partial \mathrm{L}(\Theta_0,\Theta_1)}{\partial \Theta_1} = \sum_{i=1}^{n}(h(\mathbf{x}_i) - y_i)x_{i_1}$

- $\Theta_0 := \Theta_0 - \alpha \sum_{i=1}^{n}(h(\mathbf{x}_i) - y_i)$
- $\Theta_1 := \Theta_1 - \alpha \sum_{i=1}^{n}(h(\mathbf{x}_i) - y_i)x_{i_1}$

Batch gradient descent uses all the training examples at each step.

Simple linear regression
Gradient Descent Algorithm

Squared error function $\mathrm{L}(\boldsymbol{\Theta})$ is a convex function, so there is no local optimum, just global minimum.

Assessing the accuracy of the model
Coefficient of determination R^2

R^2 measures the proportion of variance in a target value that is reduced by taking into account **x**

- $TSS = \sum_{i=1}^{n}(y_i - \overline{y})^2$ # total variance in Y
- $RSS = \sum_{i=1}^{n}(y_i - \hat{y})^2$
- $R^2 = \frac{TSS - RSS}{TSS} = 1 - \frac{RSS}{TSS}$ (%)

Multivariate linear regression

Multivariate regression is a linear regression with multiple features.

- $\mathbf{x} = \langle x_1, x_2, ..., x_m \rangle$

$$h(\mathbf{x}) = \Theta_0 + \Theta_1 x_1 + ... + \Theta_m x_m \tag{6}$$

- $\langle \Theta_0, \Theta_1, ..., \Theta_m \rangle \in \mathcal{R}^{m+1}$
- Define $x_0 = 1$, so $\mathbf{x} = \langle x_0, x_1, x_2, ..., x_m \rangle$
- Θ_i is the average change in y for a unit change in A_1 holding all other features fixed, if A_1 is a continuous feature

Multivariate linear regression

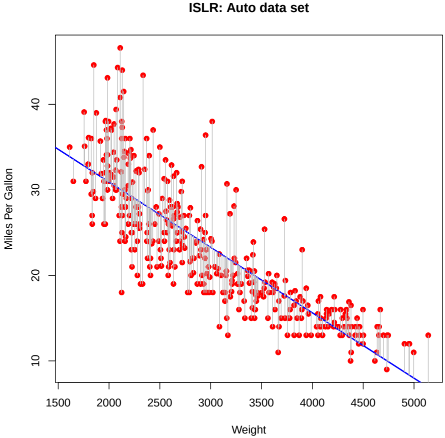
Hypothesis $h(\mathbf{x}) = \mathbf{\Theta}^T\mathbf{x}$

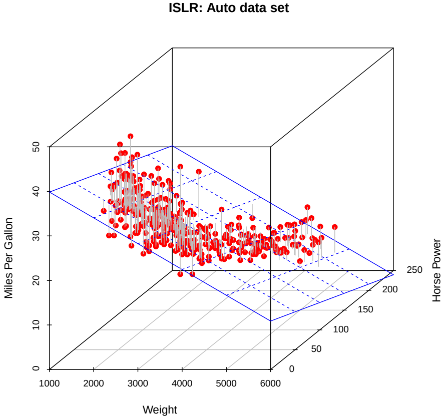
Hypothesis parameters $\mathbf{\Theta} = \langle \Theta_0, \Theta_1, \ldots, \Theta_m \rangle$

Loss function $\mathrm{L}(\mathbf{\Theta}) = RSS = \sum_{i=1}^{n}(h(\mathbf{x}_i) - y_i)^2$

Optimization task $\mathbf{\Theta}^\star = \mathrm{argmin}_{\mathbf{\Theta}}\mathrm{L}(\mathbf{\Theta})$

Multivariate linear regression
Auto data set





Multivariate linear regression
Gradient Descent Algorithm

repeat until convergence $\{$

$$\mathbf{\Theta}^{K+1} := \mathbf{\Theta}^{K} - \alpha \nabla \mathrm{L}(\mathbf{\Theta}^{K}), \tag{7}$$

where

$$\nabla \mathrm{L}(\mathbf{\Theta}^{K}) = \mathbf{X}^{T}(\mathbf{X}\mathbf{\Theta}^{K} - \mathbf{y}) \tag{8}$$

$\}$

Polynomial regression

Polynomial regression is an extension of linear regression where the relationship between features and target value is modelled as a *d*-th order polynomial.

Simple regression
$y = \Theta_0 + \Theta_1 x_1$

Polynomial regression
$y = \Theta_0 + \Theta_1 x_1 + \Theta_2 x_1^2 + \ldots \Theta_d x_1^d$

It is still a linear model with features $A_1, A_1^2, \ldots, A_1^d$.

The *linear* in linear model refers to the hypothesis parameters, not to the features. Thus, the parameters $\Theta_0, \Theta_1, \ldots, \Theta_d$ can be easily estimated using least squares linear regression.

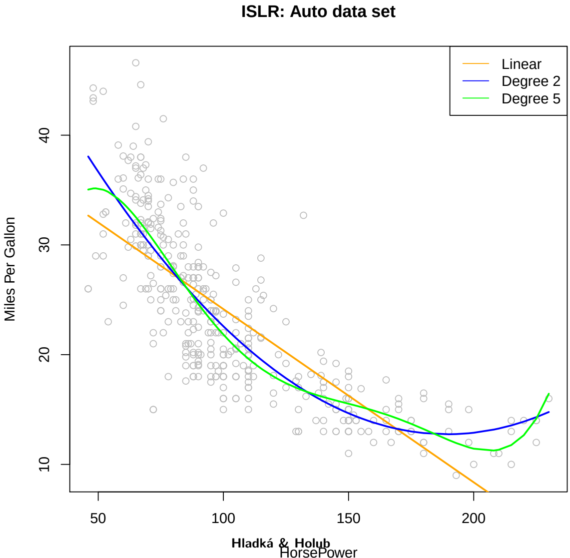
Polynomial regression

Notation

$$\mathbf{y} = \begin{pmatrix} y_1 \\ \dots \\ y_n \end{pmatrix}, \mathbf{\Theta} = \begin{pmatrix} \Theta_0 \\ \dots \\ \Theta_d \end{pmatrix}, \mathbf{X} = \begin{pmatrix} 1 & x_{11} & \dots & x_{11}^d \\ 1 & x_{21} & \dots & x_{21}^d \\ \dots & \dots & \dots & \dots \\ 1 & x_{n1} & \dots & x_{n1}^d \end{pmatrix}$$

$$\mathbf{y} = \mathbf{X\Theta}$$

Polynomial regression
Auto data set



Simple regression with a categorical feature

- assume a categorical feature with k values
- create $k-1$ dummy variables (DA^1, DA^2, $\ldots \mathrm{DA}^{k-1}$)
- then $y_i = \Theta_0 + \Theta_1 \mathrm{DA}_i^1 + \cdots + \Theta_{k-1} \mathrm{DA}_i^{k-1}$
- **Example:**
  - `mpg` $\sim$ `origin`

	DA_1^1	DA_2^1
American	0	0
European	1	0
Japanase	0	1

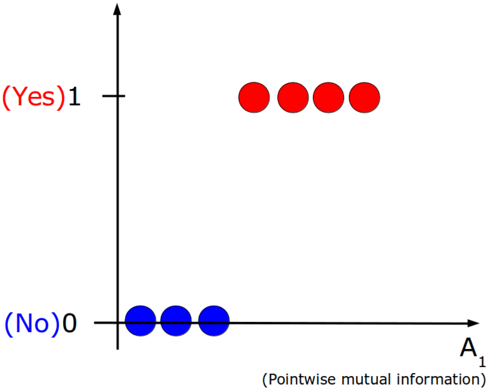
  - $y_i = \Theta_0 + \Theta_1 \mathrm{DA}_1^1 + \Theta_2 \mathrm{DA}_2^1$
  - $y_i = \Theta_0 + \Theta_1$ if the i-th car is European
  - $y_i = \Theta_0 + \Theta_2$ if the i-th car is Japanese
  - $y_i = \Theta_0$ if the i-th car is American

Interpretation

- Θ_0 as the average mpg for American cars
- Θ_1 as the average difference in mpg between European and American cars
- Θ_2 as the average difference in mpg between Japanese and American cars

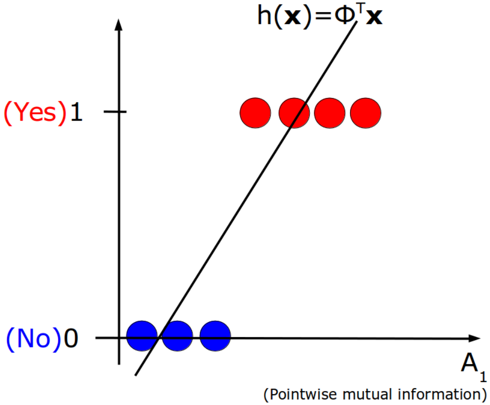
Linear regression on binary classification

- $Attr = \{A_1\}$ (e.g., Pointwise mutual information)
- $Y = \{0, 1\}$



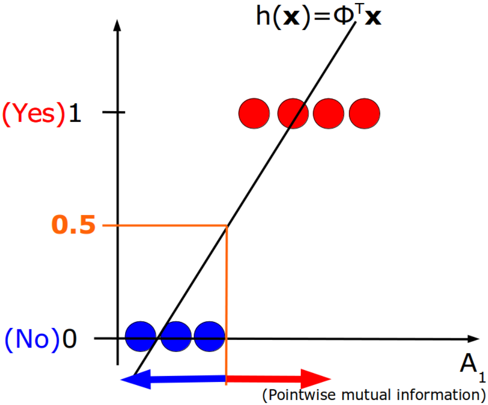
Linear regression on binary classification

- Fit the data with a linear function h



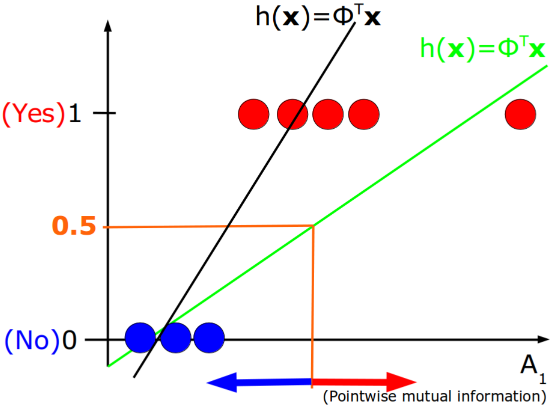
Linear regression on binary classification

- **prediction function of x**
  - if $h(\mathbf{x}) \geq 0.5$, predict 1
  - if $h(\mathbf{x}) < 0.5$, predict 0



Linear regression on binary classification

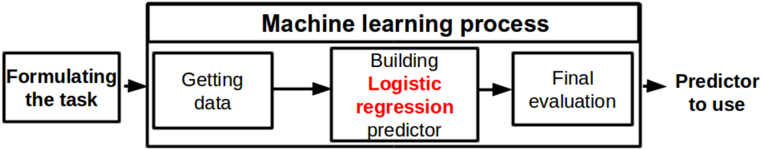
- Add one more training instance



Linear regression on binary classification

It can happen that $h(\mathbf{x}) > 1$ or $h(\mathbf{x}) < 0$ but we predict 0 and 1.

Logistic regression



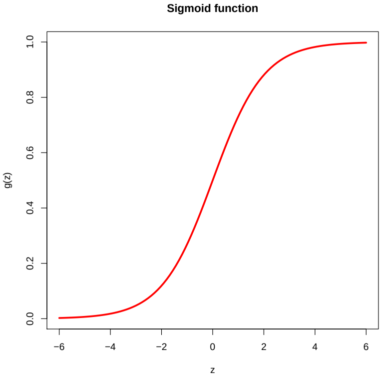
Logistic regression

h has a form of **sigmoid function** $g(z) = \frac{1}{1+e^{-z}}$

$$h(\mathbf{x}) = g(\mathbf{\Theta}^T\mathbf{x}) = \frac{1}{1 + e^{-\mathbf{\Theta}^T\mathbf{x}}} = \frac{e^{\mathbf{\Theta}^T\mathbf{x}}}{1 + e^{\mathbf{\Theta}^T\mathbf{x}}} \tag{9}$$

Sigmoid function

- $g(z) = \frac{1}{1+e^{-z}}$
- $\lim_{z \to +\infty} g(z) = 1$
- $\lim_{z \to -\infty} g(z) = 0$



Logistic regression

- We interpret the output of $h(\mathbf{x})$ as estimated probability of $y = 1$ given $\mathbf{x}$ parameterized by $\mathbf{\Theta}$, i.e. $h(\mathbf{x}) = \Pr(y = 1|\mathbf{x}; \mathbf{\Theta})$

- the ratio of the probability of success and the probability of failure
  odds $= \dfrac{h(\mathbf{x})}{1 - h(\mathbf{x})} = e^{\mathbf{\Theta}^T\mathbf{x}} \in (0, +\infty)$

- **log-odds (logit) is linear in x**

$$\log \frac{h(\mathbf{x})}{1 - h(\mathbf{x})} = \mathbf{\Theta}^T\mathbf{x} \in (-\infty, +\infty)$$

- **recall linear regression** $h(\mathbf{x}) = \mathbf{\Theta}^T\mathbf{x}$

Logistic regression
Interpretation of Θ for continuous features

Suppose $\mathbf{\Theta} =< \Theta_0, \Theta_1 >$

- linear regression $h(\mathbf{x}) = \Theta_0 + \Theta_1 x_1$: Θ_1 gives an average change in a target value with one-unit change in A_1
- logistic regression $\log \frac{h(\mathbf{x})}{1-h(\mathbf{x})} = \Theta_0 + \Theta_1 x_1$: Θ_1 gives an average change in logit $h(\mathbf{x})$ with one-unit change in A_1

Logistic regression
Interpretation of Θ for binary features

Example:

disease	female		Total
	0 (male)	1 (female)	
no	74	77	151
yes	17	32	49
Total	91	109	200

- the odds of having the disease for male:
  $\Pr(\text{disease} = \mathit{yes}|\text{female} = 0)/\Pr(\text{disease} = \mathit{no}|\text{female} = 0) = \frac{17/91}{74/91} = 0.23$
- the odds of having the disease for female:
  $\Pr(\text{disease} = \mathit{yes}|\text{female} = 1)/\Pr(\text{disease} = \mathit{no}|\text{female} = 1) = \frac{32/109}{77/109} = 0.42$
- the ratio of the odds for female to the odds for male $0.42/0.23 = 1.81$, i.e. the odds for female are about 81% higher than the odds for males

Logistic regression
Interpretation of Θ for binary features

- $\log \frac{p_1}{1-p_1} = \Theta_0 + \Theta_1 * \mathrm{female}$

  If $\mathrm{female} == 0$ then $\frac{p_1}{1-p_1} = e^{\Theta_0}$
  – the intercept Θ_0 is the log odds for men

- $\log \frac{p_2}{1-p_2} = \Theta_0 + \Theta_1 * \mathrm{female}$ If $\mathrm{female} == 1$ then $\frac{p_2}{1-p_2} = e^{\Theta_0+\Theta_1}$

- odds ratio $= \frac{p_2}{1-p_2} / \frac{p_1}{1-p_1} = e^{\Theta_1}$
  – the parameter Θ_1 is the log of odds ratio between men and women

Assume the output of logistic regression $\Theta_0 = -1.471$, $\Theta_1 = 0.593$. Then relate the odds for males and famels and the parameters:
$-1.471 = \log(0.23)$, $0.593 = \log(1.81)$

Logistic regression

Hypothesis $h(\mathbf{x}) = \frac{1}{1+e^{-\mathbf{\Theta}^T\mathbf{x}}}$

Hypothesis parameters $\mathbf{\Theta} = \langle \Theta_0, \dots, \Theta_m \rangle$

- **Loss function**

$$\mathrm{L}(\mathbf{\Theta}) = -\sum_{i=1}^{n} y_i \log \mathrm{P}(y_i|\mathbf{x_i};\mathbf{\Theta}) + (1-y_i)\log(1-\mathrm{P}(y_i|\mathbf{x_i};\mathbf{\Theta})) \tag{10}$$

- **Optimization task**
  $\mathbf{\Theta}^\star = \mathrm{argmin}_{\mathbf{\Theta}} \mathrm{L}(\mathbf{\Theta})$

The `argmin` operator will give $\mathbf{\Theta}$ for which $\mathrm{L}(\mathbf{\Theta})$ is minimal.

Logistic regression
Estimating Θ by maximizing the likelihood

(**Maximum likelihood principle** will be taught in details later on.)

- **likelihood of the data**

$$\mathcal{L}(y_1, \ldots, y_n; \mathbf{\Theta}, \mathbf{X}) = \prod_{i=1}^{n} \mathrm{P}(y_i | \mathbf{x_i}; \mathbf{\Theta})$$

- **log likelihood of the data**

$$\begin{aligned} \ell(y_1, \ldots, y_n; \mathbf{\Theta}, \mathbf{X}) &= \log \mathcal{L}(y_1, \ldots, y_n; \mathbf{\Theta}, \mathbf{X}) \\ &= \sum_{i=1}^{n} \log \mathrm{P}(y_i | \mathbf{x_i}; \mathbf{\Theta}) \\ &= \sum_{i=1}^{n} y_i \log \mathrm{P}(y_i | \mathbf{x_i}; \mathbf{\Theta}) + (1 - y_i) \log(1 - \mathrm{P}(y_i | \mathbf{x_i}; \mathbf{\Theta})) \end{aligned}$$

loss function $\mathrm{L}(\mathbf{\Theta}) = \ell(y_1, \ldots, y_n; \mathbf{\Theta}, \mathbf{X})$

Logistic regression

prediction function of x

- $h(\mathbf{x}) = g(\mathbf{\Theta}^T \mathbf{x})$
- $g(z) \geq 0.5$ whenever $z \geq 0$ and $g(z) < 0.5$ whenever $z < 0$
  - if $h(\mathbf{x}) \geq 0.5$, i.e. $\mathbf{\Theta}^T \mathbf{x} \geq 0$, predict 1
  - if $h(\mathbf{x}) < 0.5$, i.e. $\mathbf{\Theta}^T \mathbf{x} < 0$, predict 0

Decision boundary

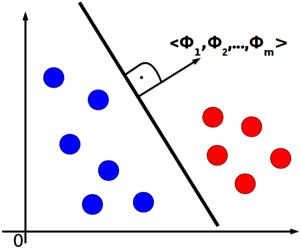
partitions a feature space into two sets, one for each class. Decision boundary takes a form of function h.

Hyperplane

Assume a **linear** decision boundary, called **hyperplane**, of the form

$$h(\mathbf{x}) = \mathbf{\Theta}^T x = \Theta_0 + \sum_{i=1}^{m} \Theta_i x_i$$

where direction of $\langle \Theta_1, \Theta_2, \ldots, \Theta_m \rangle$ is perpendicular to the hyperplane and Θ_0 determines position of the hyperplane with respect to the origin

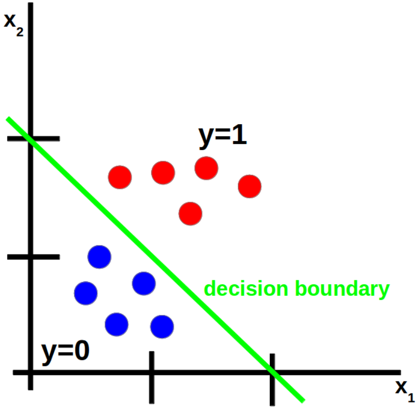


Hyperplane

- Logistic regression models imply a linear decision boundary.
- A condition for instance $\mathbf{x}$ to be on the hyperplane is $h(\mathbf{x}) = \mathbf{\Theta}^T\mathbf{x} = 0$.
- Decision boundaries are the set of points with $\log \mathrm{odds} = 0$

Logistic regression

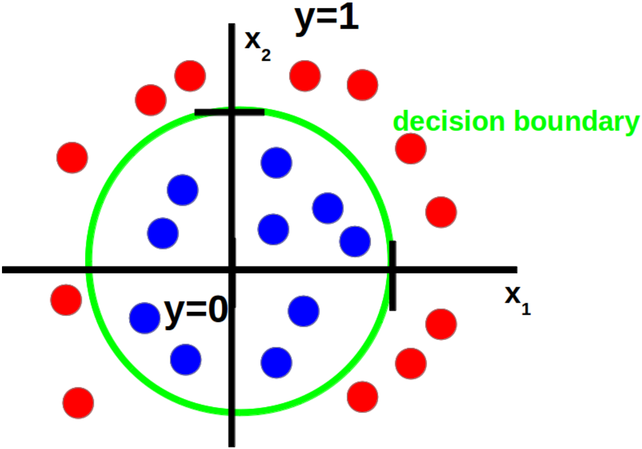
- Predict $y = 1$ if $h(\mathbf{x}) \geq 0.5$, i.e. $\mathbf{\Theta}^T\mathbf{x} \geq 0$
- Predict $y = 0$ if $h(\mathbf{x}) < 0.5$, i.e. $\mathbf{\Theta}^T\mathbf{x} < 0$



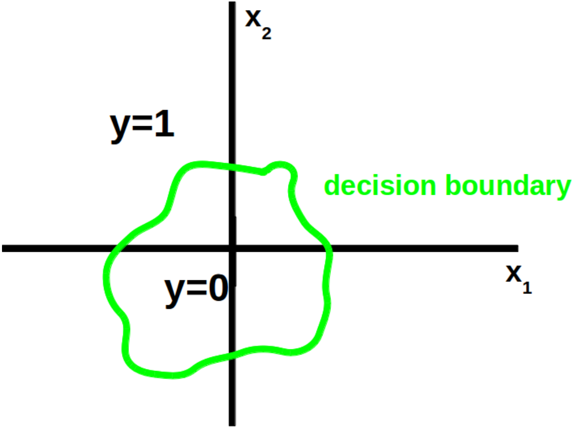
Non-linear decision boundary

- Let $h(\mathbf{x}) = g(\Theta_0 + \Theta_1 x_1 + \Theta_2 x_2 + \Theta_3 x_1^2 + \Theta_4 x_2^2)$ (a higher degree polynomial)
- Assume $\Theta_0 = -1$, $\Theta_1 = 0, \Theta_2 = 0$, $\Theta_3 = 1, \Theta_4 = 1$
- Predict $y = 1$ if $-1 + x_1^2 + x_2^2 \geq 0$, i.e. $x_1^2 + x_2^2 \geq 1$

Non-linear decision boundary



More complicated decision boundary



Logistic regression
Gradient Descent Algorithm

Loss function $\mathrm{L}(\boldsymbol{\Theta}) = -\sum_{i=1}^{n} y_i \log(h(\mathbf{x}_i)) + (1 - y_i)\log(1 - h(\mathbf{x}_i))$

Optimization task $\boldsymbol{\Theta}^\star = \mathrm{argmin}_{\boldsymbol{\Theta}} \mathrm{L}(\boldsymbol{\Theta})$

Use Gradient descent algorithm

Repeat until convergence

$\{$

$$\Theta_j := \Theta_j - \alpha \frac{\partial \mathrm{L}(\theta)}{\partial \Theta_j} \tag{11}$$

(simultaneously update Θ_j for $j = 1, \ldots, m$)

$\}$

Logistic regression
Gradient descent algorithm

Repeat until convergence
{

$$\Theta_j := \Theta_j - \alpha \sum_{i=1}^{n} (h(\mathbf{x}_i) - y_i)\mathbf{x}_{i_j} \tag{12}$$

(simultaneously update Θ_j for $j = 1, \ldots, m$)
}

Have you already meet it? Yes, see linear regression.

- linear regression $h(\mathbf{x}) = \mathbf{\Theta}^T\mathbf{x}$
- logistic regression $h(\mathbf{x}) = \frac{1}{1+e^{-\mathbf{\Theta}^T\mathbf{x}}}$

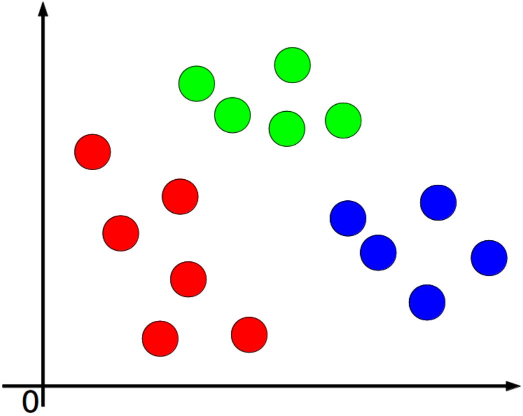
Logistic regression
Summary

Classification of $\mathbf{x}$ by $h^\star$

1. Project $\mathbf{x}$ onto $\mathbf{\Theta}^\star$ to convert it into a real number z in the range $\langle -\infty, +\infty \rangle$
   - i.e. $z = (\mathbf{\Theta}^\star)^T \mathbf{x}$
2. Map z to the range $\langle 0, 1 \rangle$ using the sigmoid function $g(z) = 1/(1 + e^{-z})$

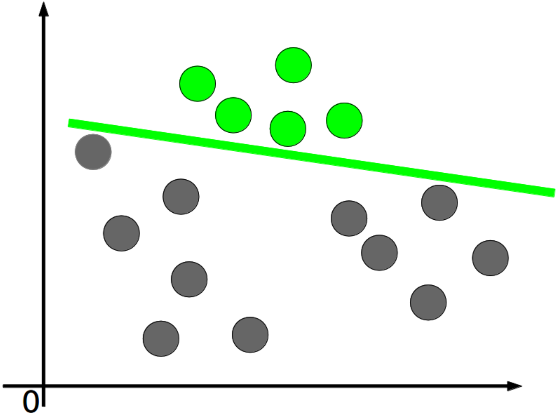
Logistic regression for multi-class classification

One-vs-all algorithm



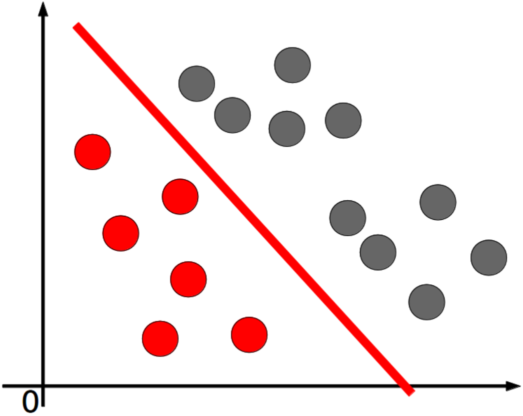
Logistic regression for multi-class classification

One-vs-all algorithm



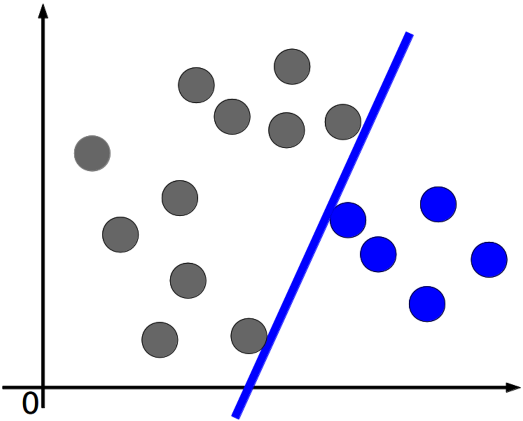
Logistic regression for multi-class classification

One-vs-all algorithm



Logistic regression for multi-class classification

One-vs-all algorithm



Logistic regression for multi-class classification

One-vs-all algorithm

New instance $\mathbf{x}$:

- $h(\mathbf{x}) = \Pr(y = \mathit{red}|\mathbf{x}; \Theta)$
- $h(\mathbf{x}) = \Pr(y = \mathit{blue}|\mathbf{x}; \Theta)$
- $h(\mathbf{x}) = \Pr(y = \mathit{green}|\mathbf{x}; \Theta)$

Classify $\mathbf{x}$ into class $i \in \{\mathit{red}, \mathit{green}, \mathit{blue}\}$ that maximizes $h(\mathbf{x})$.

Summary of Examination Requirements

- Simple linear regression
- Multivariete linear regression
- Polynomial linear regression
- Coefficient of determination
- Gradient Descent Algorithm
- Logistic regression