Outline

- **Linear regression**
  - Auto data set

- **Logistic regression**
  - Auto data set
Machine learning process and development cycle

Machine learning process

Formulating the task

Getting data → Building predictor → Final evaluation → ML method selection

Development cycle

Feature engineering → Predictor training → Development evaluation → Parameter tuning

Predictor to use
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
Idealized model of supervised learning

- $x_i$ are feature vectors, $y_i$ are true predictions
- prediction function $h^*$ is the “best” of all possible hypotheses $h$
- learning process is searching for $h^*$, which means to search the hypothesis space and minimize a predefined loss function
- ideally, the learning process results in $h^*$ so that predicted $\hat{y}_i = h^*(x_i)$ is equal to the true target values $y_i$
Linear regression
### Dataset Auto from the ISLR package

392 instances on the following 9 features

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpg</td>
<td>Miles per gallon</td>
</tr>
<tr>
<td>cylinders</td>
<td>Number of cylinders between 4 and 8</td>
</tr>
<tr>
<td>displacement</td>
<td>Engine displacement (cu. inches)</td>
</tr>
<tr>
<td>horsepower</td>
<td>Engine horsepower</td>
</tr>
<tr>
<td>weight</td>
<td>Vehicle weight (lbs.)</td>
</tr>
<tr>
<td>acceleration</td>
<td>Time to accelerate from 0 to 60 mph (sec.)</td>
</tr>
<tr>
<td>year</td>
<td>Model year (modulo 100)</td>
</tr>
<tr>
<td>name</td>
<td>Vehicle name</td>
</tr>
</tbody>
</table>
Dataset Auto from the ISLR package
Linear regression

$h$ has a form of **linear function**

\[ h(x) = \Theta_0 + \Theta_1 x_1 + \ldots \Theta_m x_m = \Theta_0 + \langle \Theta_1, \ldots, \Theta_m \rangle^T x \quad (1) \]

**Linear regression** is a **parametric method**.

We estimate \( m + 1 \) parameters (\( \Theta \)) instead of fitting data with an entirely arbitrary function \( h \).
Linear regression

Notation

\[ y = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}, \quad \Theta = \begin{pmatrix} \Theta_0 \\ \vdots \\ \Theta_m \end{pmatrix}, \quad X = \begin{pmatrix} 1 & x_{11} & \cdots & x_{1m} \\ 1 & x_{21} & \cdots & x_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \cdots & x_{nm} \end{pmatrix} \]

\[ y = X\Theta \]
Simple regression is a linear regression with a single feature.

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

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

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

- Residual $e_i = y_i - h(x_i)$
- Residual sum of squares $RSS(h) = \sum_{i=1}^{n} e_i^2$
Simple linear regression

Hypothesis

\[ h(x) = \Theta_0 + \Theta_1 x_1 \]

Hypothesis parameters

\[ \Theta = \langle \Theta_0, \Theta_1 \rangle \]

- Loss function

\[ L(\Theta) = RSS = (X\Theta - y)^2 \]  \hspace{1cm} (2)

- Optimization task

\[ \Theta^* = \arg\min_{\Theta} L(\Theta) \]  \hspace{1cm} (3)

The \arg\min operator will give \( \Theta \) for which \( L(\Theta) \) is minimal.
Simple linear regression
Solving the loss function analytically

• Find the pair \((\Theta_0, \Theta_1)\) that minimizes \(L(\Theta) = \sum_{i=1}^{n} (y_i - \Theta_0 - \Theta_1 x_i)^2\)

\[ \frac{\partial L(\Theta)}{\partial \Theta_0} = - \sum_{i=1}^{n} 2(y_i - \Theta_0 - \Theta_1 x_i) \]
\[ - \sum_{i=1}^{n} 2(y_i - \Theta_0 - \Theta_1 x_i) = 0 \implies \sum_{i=1}^{n} (y_i - \Theta_0 - \Theta_1 x_i) = 0 \]

\[ \frac{\partial L(\Theta)}{\partial \Theta_1} = - \sum_{i=1}^{n} 2x_i(y_i - \Theta_0 - \Theta_1 x_i) \]
\[ - \sum_{i=1}^{n} 2x_i(y_i - \Theta_0 - \Theta_1 x_i) = 0 \implies \sum_{i=1}^{n} x_i(y_i - \Theta_0 - \Theta_1 x_i) = 0 \]

• Using the Normal equations calculus (see below), the minimizers are

\[ \Theta_1 = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n} (x_i - \bar{x})^2} \]

\[ \Theta_0 = \bar{y} - \Theta_1 \bar{x} \]

where \(\bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i, \bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i\)
Simple linear regression
Solving the loss function analytically

Normal Equations Calculus

Find $\Theta$ that minimizes $e = y - X\Theta$

Theorem

$\Theta^* \text{ is a least squares solution to } y = X\Theta \iff \Theta^* \text{ is a solution to the Normal equation } X^TX\Theta = X^Ty.$
Simple linear regression
Solving the loss function analytically

Normal Equations Calculus

- $\Theta^* = (X^T X)^{-1} X^T y$
- 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 $\Theta$ satisfies at $X^T(y - X\Theta) = 0$ where the residual vector $y - X\Theta$ is a normal to the columns of $X$.
- (Two non-zero vectors $a$ and $b$ are orthogonal if and only if $ab = 0$.)
Simple linear regression
Solving the loss function analytically

- When using the Normal equations for solving the cost function analytically one has to compute $(X^T X)^{-1} X^T y$
- 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 $\Theta$, 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: $h(x) = \Theta_1 x_1$

Hypothesis parameters: $\Theta_1$ ($\Theta = \langle 0, \Theta_1 \rangle$)

Loss function: $L(\Theta) = RSS$

Optimization task: $\Theta_1^* = \arg\min_{\Theta_1} L(\Theta)$
$\theta_0 = 0, \theta_1 \neq 0$
\[ \Theta_0 \neq 0, \Theta_1 \neq 0 \]
Gradient descent algorithm is an optimization algorithm to find a local minimum of a function \( f \).
1. Start with some \( x_0 \).
2. Keep changing $x_i$ to reduce $f(x_i)$ 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 $f(x)$. Given some initial value $x_0$, we can change its value in many directions.

• What is the best direction to minimize $f$? We take the gradient $\nabla f$ of $f$

$$\nabla f(x_1, x_2, \ldots, x_m) = \langle \frac{\partial f(x_1, x_2, \ldots, x_m)}{\partial x_1}, \ldots, \frac{\partial f(x_1, x_2, \ldots, x_m)}{\partial x_m} \rangle$$ (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 $x$ in the opposite direction to lower the function value.
Gradient Descent Algorithm

Choice of the step
Choice of the step

\[ f(x) \]

\[ f(z) \]

Step 0

\[ z \]
Gradient Descent Algorithm

Choice of the step

![Graph showing the choice of the step in gradient descent algorithm. The function f(x) is plotted, with points labeled as step i and step i+1. The point z is also marked, showing the direction of the step.](image)
repeat until convergence {

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

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

} 

$\alpha$ is a positive step-size parameter that controls how big step we’ll take downhill.
• If $\alpha$ is too large, GDA can overshoot the minimum. It may fail to converge, or even diverge.

• If $\alpha$ is too small, GDA can be slow.
Simple linear regression

Gradient Descent Algorithm

\[
\frac{\partial L(\Theta_0, \Theta_1)}{\partial \Theta_j} = \frac{\partial}{\partial \Theta_j} \sum_{i=1}^{n} (h(x_i) - y_i)^2 = \frac{\partial}{\partial \Theta_j} \sum_{i=1}^{n} (\Theta_0 + \Theta_1 x_i - y_i)^2
\]

- \( j = 0 \): \( \frac{\partial L(\Theta_0, \Theta_1)}{\partial \Theta_0} = \sum_{i=1}^{n} (h(x_i) - y_i) \)
- \( j = 1 \): \( \frac{\partial L(\Theta_0, \Theta_1)}{\partial \Theta_1} = \sum_{i=1}^{n} (h(x_i) - y_i) x_i \)

\[
\Theta_0 := \Theta_0 - \alpha \sum_{i=1}^{n} (h(x_i) - y_i) \\
\Theta_1 := \Theta_1 - \alpha \sum_{i=1}^{n} (h(x_i) - y_i) x_i
\]

**Batch** gradient descent uses all the training examples at each step.
Squared error function $L(\Theta)$ 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 $x$

- $TSS = \sum_{i=1}^{n} (y_i - \bar{y})^2$ # 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 is a linear regression with multiple features.

- \( x = \langle x_1, x_2, \ldots, x_m \rangle \)

\[
h(x) = \Theta_0 + \Theta_1 x_1 + \ldots + \Theta_m x_m
\]  

- \( \langle \Theta_0, \Theta_1, \ldots, \Theta_m \rangle \in \mathbb{R}^{m+1} \)
- Define \( x_0 = 1 \), so \( x = \langle x_0, x_1, x_2, \ldots, x_m \rangle \)
- \( \Theta_i \) 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: \( h(x) = \Theta^T x \)

Hypothesis parameters: \( \Theta = \langle \Theta_0, \Theta_1, \ldots, \Theta_m \rangle \)

Loss function: \( L(\Theta) = RSS = \sum_{i=1}^{n} (h(x_i) - y_i)^2 \)

Optimization task: \( \Theta^* = \text{argmin}_\Theta L(\Theta) \)
Multivariate linear regression

Auto data set

ISLR: Auto data set

Weight

Miles Per Gallon

ISLR: Auto data set

Horse Power

ISLR: Auto data set

Weight

Horse Power

Miles Per Gallon

Weight
repeat until convergence 

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

where

$$\nabla L(\Theta^K) = X^T (X\Theta^K - y) \quad (8)$$

}
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

\[ y = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}, \quad \Theta = \begin{pmatrix} \Theta_0 \\ \vdots \\ \Theta_d \end{pmatrix}, \quad X = \begin{pmatrix} 1 & x_{11} & \cdots & x_{1d} \\ 1 & x_{21} & \cdots & x_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \cdots & x_{nd} \end{pmatrix} \]

\[ y = X\Theta \]
Polynomial regression
Auto data set

ISLR: Auto data set

- Orange line: Linear
- Blue line: Degree 2
- Green line: Degree 5
Simple regression with a categorical feature

• assume a categorical feature with \( k \) values
• create \( k - 1 \) dummy variables ( \( DA^1, DA^2, \ldots DA^{k-1} \))
• then \( y_i = \Theta_0 + \Theta_1 DA^1_i + \cdots + \Theta_{k-1} DA^{k-1}_i \)

**Example:**

- \( mpg \sim \text{origin} \)

<table>
<thead>
<tr>
<th></th>
<th>( DA^1_i )</th>
<th>( DA^2_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>American</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>European</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Japanese</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

- \( y_i = \Theta_0 + \Theta_1 DA^1_i + \Theta_2 DA^2_i \)
- \( y_i = \Theta_0 + \Theta_1 \) 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**

• \( \Theta_0 \) as the average mpg for American cars
• \( \Theta_1 \) as the average difference in mpg between European and American cars
• \( \Theta_2 \) as the average difference in mpg between Japanese and American cars
- $Attr = \{ A_1 \}$ (e.g., Pointwise mutual information)
- $Y = \{ 0, 1 \}$
Linear regression on binary classification

- Fit the data with a linear function $h$

$$h(x) = \Phi^T x$$

(Pointwise mutual information)

(Yes) 1

(No) 0

$A_1$
Linear regression on binary classification

- prediction function of $x$
  - if $h(x) \geq 0.5$, predict 1
  - if $h(x) < 0.5$, predict 0
Linear regression on binary classification

- Add one more training instance
It can happen that $h(x) > 1$ or $h(x) < 0$ but we predict 0 and 1.
Logistic regression

Machine learning process

Formulating the task

Getting data

Building Logistic regression predictor

Final evaluation

Predictor to use
Logistic regression

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

$$h(x) = g(\Theta^T x) = \frac{1}{1 + e^{-\Theta^T x}} = \frac{e^{\Theta^T x}}{1 + e^{\Theta^T x}}$$ (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 $h(x)$ as estimated probability of $y = 1$ given $x$ parameterized by $\Theta$, i.e. $h(x) = \Pr(y = 1|\mathbf{x}; \Theta)$

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

- log-odds (logit) is linear in $\mathbf{x}$
  \[
  \log \frac{h(x)}{1 - h(x)} = \Theta^T \mathbf{x} \in (-\infty, +\infty)
  \]

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

Interpretation of $\Theta$ for continuous features

Suppose $\Theta = \langle \Theta_0, \Theta_1 \rangle$

- linear regression $h(x) = \Theta_0 + \Theta_1 x_1$: $\Theta_1$ gives an average change in a target value with one-unit change in $A_1$

- logistic regression $\log \frac{h(x)}{1-h(x)} = \Theta_0 + \Theta_1 x_1$: $\Theta_1$ gives an average change in logit $h(x)$ with one-unit change in $A_1$
Logistic regression
Interpretation of Θ for binary features

Example:

<table>
<thead>
<tr>
<th>Disease</th>
<th>Female</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0 (Male)</td>
<td>1 (Female)</td>
</tr>
<tr>
<td>No</td>
<td>74</td>
<td>77</td>
</tr>
<tr>
<td>Yes</td>
<td>17</td>
<td>32</td>
</tr>
<tr>
<td>Total</td>
<td>91</td>
<td>109</td>
</tr>
</tbody>
</table>

- the odds of having the disease for male:
  \[ \frac{Pr(\text{disease} = \text{yes} | \text{female} = 0)}{Pr(\text{disease} = \text{no} | \text{female} = 0)} = \frac{17/91}{74/91} = 0.23 \]
- the odds of having the disease for female:
  \[ \frac{Pr(\text{disease} = \text{yes} | \text{female} = 1)}{Pr(\text{disease} = \text{no} | \text{female} = 1)} = \frac{32/109}{77/109} = 0.42 \]
- 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 $\Theta$ for binary features

- $\log \frac{p_1}{1-p_1} = \Theta_0 + \Theta_1 \times \text{female}$

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

- $\log \frac{p_2}{1-p_2} = \Theta_0 + \Theta_1 \times \text{female}$ If 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 $\Theta_1$ 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 females and the parameters:

$-1.471 = \log(0.23)$, $0.593 = \log(1.81)$
Logistic regression

Hypothesis

\[ h(x) = \frac{1}{1 + e^{-\Theta^T x}} \]

Hypothesis parameters

\[ \Theta = \langle \Theta_0, \ldots, \Theta_m \rangle \]

- Loss function

\[ L(\Theta) = -\sum_{i=1}^{n} y_i \log P(y_i|x_i; \Theta) + (1 - y_i) \log (1 - P(y_i|x_i; \Theta)) \]  \hspace{1cm} (10)

- Optimization task

\[ \Theta^* = \arg\min_{\Theta} L(\Theta) \]

The argmin operator will give \( \Theta \) for which \( L(\Theta) \) is minimal.
Logistic regression
Estimating $\Theta$ by maximizing the likelihood

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

- likelihood of the data

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

- log likelihood of the data

$$\ell(y_1, \ldots, y_n; \Theta, X) = \log L(y_1, \ldots, y_n; \Theta, X)$$

$$= \sum_{i=1}^{n} \log P(y_i|x_i; \Theta)$$

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

**loss function** $L(\Theta) = \ell(y_1, \ldots, y_n; \Theta, X)$
Logistic regression

prediction function of x

- $h(x) = g(\Theta^T x)$

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

partitions a feature space into two sets, one for each class. Decision boundary takes a form of function $h$. 
Assume a **linear** decision boundary, called **hyperplane**, of the form

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

where direction of \( \langle \Theta_1, \Theta_2, \ldots, \Theta_m \rangle \) is perpendicular to the hyperplane and \( \Theta_0 \) determines position of the hyperplane with respect to the origin.
• Logistic regression models imply a linear decision boundary.
• A condition for instance $\mathbf{x}$ to be on the hyperplane is $h(\mathbf{x}) = \Theta^T \mathbf{x} = 0$.
• Decision boundaries are the set of points with log odds $= 0$. 
Logistic regression

- Predict $y = 1$ if $h(x) \geq 0.5$, i.e. $\Theta^T x \geq 0$
- Predict $y = 0$ if $h(x) < 0.5$, i.e. $\Theta^T x < 0$
Non-linear decision boundary

- Let $h(x) = g(\Theta_0 + \Theta_1 x_1 + \Theta_2 x_2 + \Theta_3 x_1^2 + \Theta_4 x_2^2)$ (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

decision boundary

y=1

y=0

x_1

x_2
Logistic regression

Gradient Descent Algorithm

Loss function

\[ L(\Theta) = -\sum_{i=1}^{n} y_i \log(h(x_i)) + (1 - y_i) \log(1 - h(x_i)) \]

Optimization task

\[ \Theta^* = \arg\min_{\Theta} L(\Theta) \]

Use Gradient descent algorithm

Repeat until convergence

\[
\Theta_j := \Theta_j - \alpha \frac{\partial L(\theta)}{\partial \Theta_j}
\]  \hspace{1cm} (11)

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

Logistic regression
Gradient descent algorithm

Repeat until convergence

\[
\Theta_j := \Theta_j - \alpha \sum_{i=1}^{n} (h(x_i) - y_i)x_{ij}
\] (simultaneously update $\Theta_j$ for $j = 1, \ldots, m$)

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

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

1. Project \( x \) onto \( \Theta^* \) to convert it into a real number \( z \) in the range \( (-\infty, +\infty) \)
   
   \( \text{i.e. } z = (\Theta^*)^T x \)

2. Map \( z \) to the range \( [0, 1] \) 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

Diagram showing two sets of data points: one set is represented by black circles, and the other set is represented by green circles. A green line is drawn to separate the two sets of data points.
Logistic regression for multi-class classification

One-vs-all algorithm

![Diagram showing a red line separating two classes of data points. The red line is a decision boundary that discriminates between the two classes.](image)
Logistic regression for multi-class classification

One-vs-all algorithm

Diagram showing a scatter plot with two classes, one represented by grey circles and the other by blue circles, separated by a decision boundary.
Logistic regression for multi-class classification

One-vs-all algorithm

New instance $\mathbf{x}$:

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

Classify $\mathbf{x}$ into class $i \in \{\text{red}, \text{green}, \text{blue}\}$ that maximizes $h(\mathbf{x})$. 
Summary of Examination Requirements

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