Classification

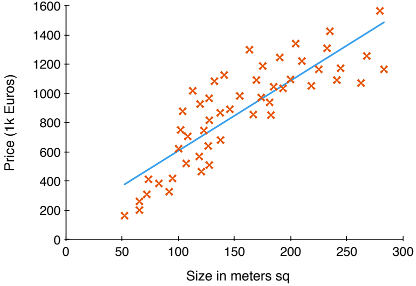
Lecture 2 - DAMLF | ML1

human+machine INTELLIGENCE

Frankfurt School

Review: Regression and Gradient Descent

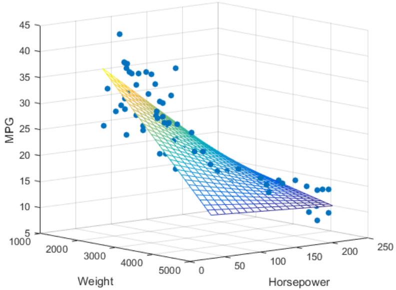
Regression Analysis



Univariate Linear Regression:

$$h(x; \boldsymbol{\theta}) = \theta_0 + \theta_1 x$$

Regression Analysis



Univariate Linear Regression:

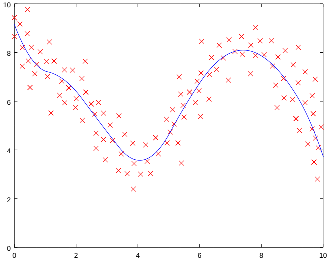
$$h(x; \boldsymbol{\theta}) = \theta_0 + \theta_1 x$$

Multiple Linear Regression:

$$h(x; \boldsymbol{\theta}) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 \cdots + \theta_n x_n$$

where $x_0 = 1$

Regression Analysis



Univariate Linear Regression:

$$h(x;\boldsymbol{\theta}) = \theta_0 + \theta_1 x$$

Multiple Linear Regression:

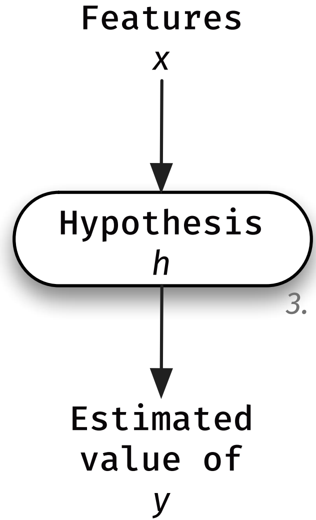
$$h(x;\boldsymbol{\theta}) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 \cdots + \theta_n x_n$$

where $x_0 = 1$

Polynomial Regression:

$$h(x;\boldsymbol{\theta}) = \theta_0 + \theta_1 x + \theta_2 x^2 \cdots + \theta_n x^n$$

Regression Analysis



Univariate Linear Regression:

$h(x; \boldsymbol{\theta}) = \theta_0 + \theta_1 x$

Multiple Linear Regression:

$h(x; \boldsymbol{\theta}) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 \cdots + \theta_n x_n$

where $x_0 = 1$

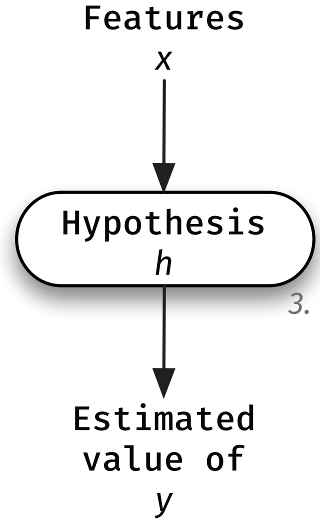
Polynomial Regression:

$h(x; \boldsymbol{\theta}) = \theta_0 + \theta_1 x + \theta_2 x^2 \cdots + \theta_n x^n$

Cost Function $J(\theta)$:

$MSE = \frac{1}{m} \sum_{i=1}^{m} (y^{(i)} - h(x^{(i)}; \boldsymbol{\theta}))^2$

Regression Analysis



Univariate Linear Regression:

$h(x;\boldsymbol{\theta}) = \theta_0 + \theta_1 x$

Normal Equation: $\hat{\boldsymbol{\theta}} = (X^T X)^{-1} X^T y$

Multiple Linear Regression:

$h(x;\boldsymbol{\theta}) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 \cdots + \theta_n x_n$

where $x_0 = 1$

Normal Equation: $\hat{\boldsymbol{\theta}} = (X^T X)^{-1} X^T y$

Polynomial Regression:

$h(x;\boldsymbol{\theta}) = \theta_0 + \theta_1 x + \theta_2 x^2 \cdots + \theta_n x^n$

Normal Equation: $\hat{\boldsymbol{\theta}} = (X^T X)^{-1} X^T y$

Cost Function $J(\theta)$**:**

$MSE = \frac{1}{m} \sum_{i=1}^{m} (y^{(i)} - h(x^{(i)};\boldsymbol{\theta}))^2$

Normal Equation & the MSE Estimator

The **Ordinary Least Squares** or **Mean Squared Error Estimator** $\hat{\boldsymbol{\theta}}$

$$\hat{\boldsymbol{\theta}} \quad := \quad \min_{\theta} J(\theta) \quad = \quad \min_{\theta_0, \theta_1, \ldots, \theta_n} J(\theta_0, \theta_1, \ldots, \theta_n)$$

where $\hat{\theta}$ is solved[1] by

$$\begin{aligned}(X\hat{\boldsymbol{\theta}} - y)^T X &= 0 \\ y^T X &= \hat{\boldsymbol{\theta}}^T (X^T X) \\ X^T y &= (X^T X)\hat{\boldsymbol{\theta}} \\ \hat{\boldsymbol{\theta}} &= (X^T X)^{-1} X^T y\end{aligned}$$

[1] In statistics, β and $\hat{\beta}$ are sometimes used instead of θ and $\hat{\theta}$, respectively.

What is a Learning Algorithm?

"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E"

–Tom Mitchell

Ingredients:

Task *to perform*

Type of **Experience**

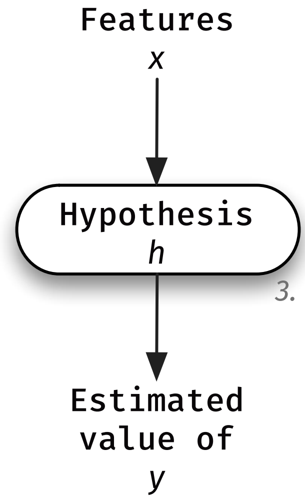
Performance *measure*

What is a Learning Algorithm?

"A computer program is said to learn from experience *E with respect to some class of tasks T and* perfor-mance *measure P, if its performance at* tasks *in T, as measured by P, improves with experience E"*

–Tom Mitchell

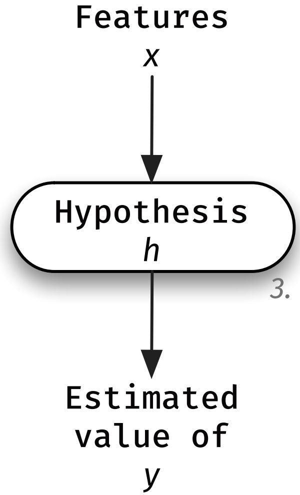
Gradient Descent for Multiple Linear Regression



Multiple Linear Regression

1. **Hypothesis**
   $h(x; \boldsymbol{\theta}) = \theta_0 x_0 + \theta_1 x_1 + \cdots + \theta_n x_n$
2. **Parameters**
   $\boldsymbol{\theta}$ ($n + 1$*-dimensional scalar-valued vector*)
3. **Cost Function**
   $J(\boldsymbol{\theta}) = \frac{1}{2m} \sum_{i=1}^{m} (h(x^{(i)}; \boldsymbol{\theta}) - y^{(i)})^2$
4. **Goal**
   $\min_{(\boldsymbol{\theta})} J(\boldsymbol{\theta})$

Gradient Descent for Multiple Linear Regression



For $n \geqslant 1$:

» Enter loop

Repeat:

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

simultaneously update θ_j for $j = 0, \ldots, n$

Stop at convergence

» Exit loop

Note: *remember that* $x_0 = 1$; *that is.,* $x_0^{(i)} = 1$ *for all* $i = 1 \ldots m$.

Gradient Descent for Multiple Linear Regression

Tips:

How to choose a good learning rate, α

Importance of feature scaling

Iterations to convergence can vary enormously from problem-to-problem

For $n \geqslant 1$:

» Enter loop

Repeat:

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

simultaneously update θ_j for $j = 0, \ldots, n$

Stop at convergence

» Exit loop

Note: *remember that $x_0 = 1$;*

that is., $x_0^{(i)} = 1$ for all $i = 1 \ldots m$.

Today: Classification Problems

Regression vs Classification

Regression Problems
Input: Features *x* are continuous valued *data*
Output: *Predicts* a continuous value *y*.

Classification Problems
Input: Features *x* are continuous valued *data*
Output: *Predicts* a **discrete** value *y*.

Binary classification problem

Predicts output variables y in $\{0, 1\}$, where

- 0 denotes the **negative** class
- 1 denotes the **positive** class

Binary classification problem

Predicts output variables y in $\{0, 1\}$, where

0 denotes the **negative** class

1 denotes the **positive** class

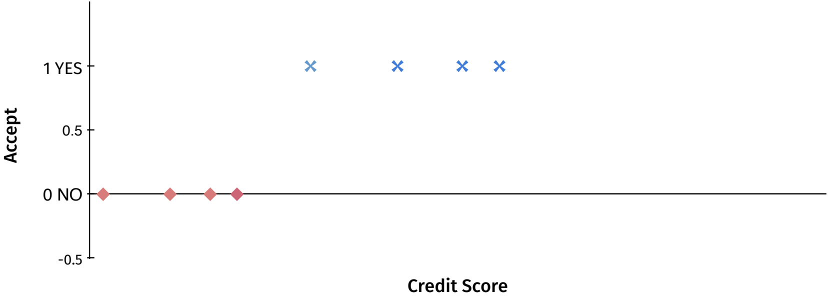
Examples:

- Email spam filter: 1 = spam; 0 = not spam
- Tumor classification: 1 = malignant; 0 = benign
- University Admissions: 1 = accept; 0 = reject
- Credit score: 1 = credit-worthy; 0 = not credit-worthy

$\vdots$

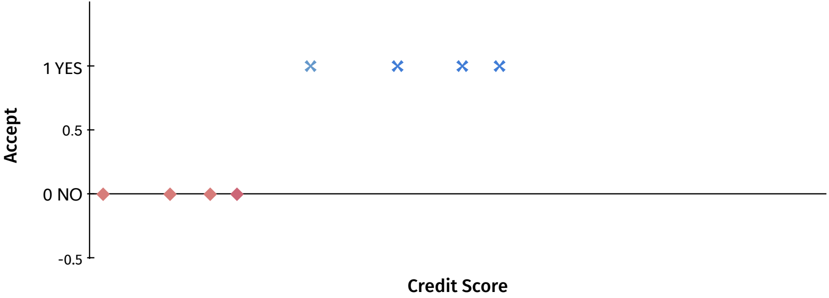
Classification is Different from Regression

Example: Credit Scoring



Classification is Different from Regression

Example: Credit Scoring



Suppose we applied **linear regression** to this example:

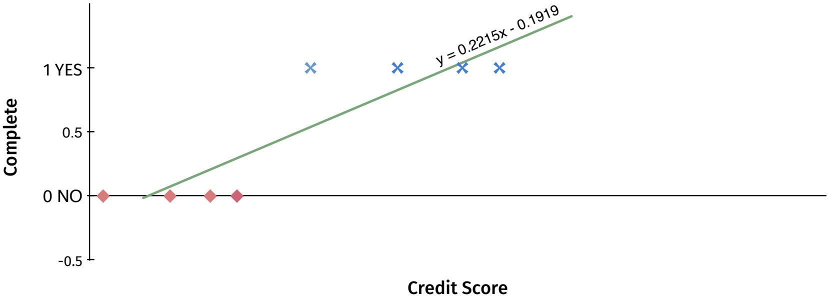
$$h(x^{(i)}; \boldsymbol{\theta}) = \boldsymbol{\theta}^T x^{(i)}$$

where

$$\boldsymbol{\theta} = (X^T X)^{-1} X^T y$$

Classification is Different from Regression

Example: $h(x; \boldsymbol{\theta}) = \boldsymbol{\theta}^T x$



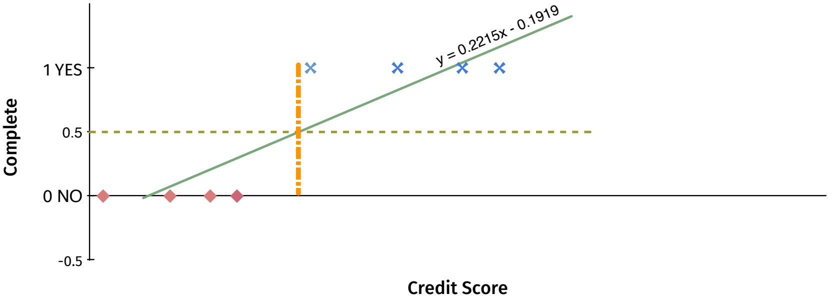
Threshold Classification

If $h(x; \boldsymbol{\theta}) \geqslant \frac{1}{2}$, predict that $y = 1$

If $h(x; \boldsymbol{\theta}) < \frac{1}{2}$, predict that $y = 0$

Classification is Different from Regression

Example: $h(x; \boldsymbol{\theta}) = \boldsymbol{\theta}^T x$



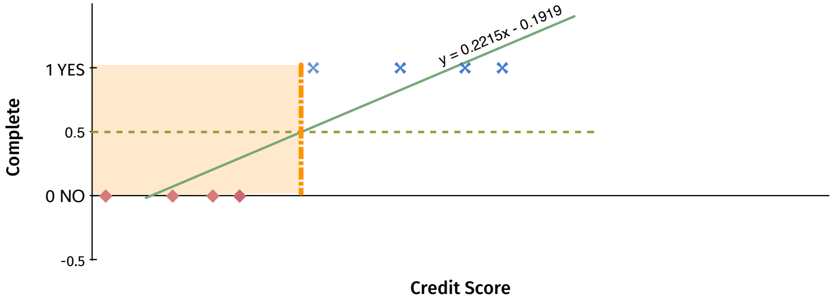
Threshold Classification

If $h(x; \boldsymbol{\theta}) \geqslant \frac{1}{2}$, predict that $y = 1$

If $h(x; \boldsymbol{\theta}) < \frac{1}{2}$, predict that $y = 0$

Classification is Different from Regression

Example: $h(x; \boldsymbol{\theta}) = \boldsymbol{\theta}^T x$



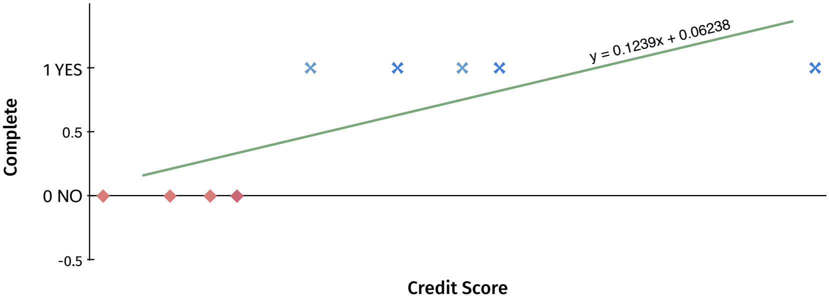
Threshold Classification

If $h(x; \boldsymbol{\theta}) \geqslant \frac{1}{2}$, predict that $y = 1$

If $h(x; \boldsymbol{\theta}) < \frac{1}{2}$, predict that $y = 0$

Classification is Different from Regression

Example: $h(x; \boldsymbol{\theta}) = \boldsymbol{\theta}^T x$



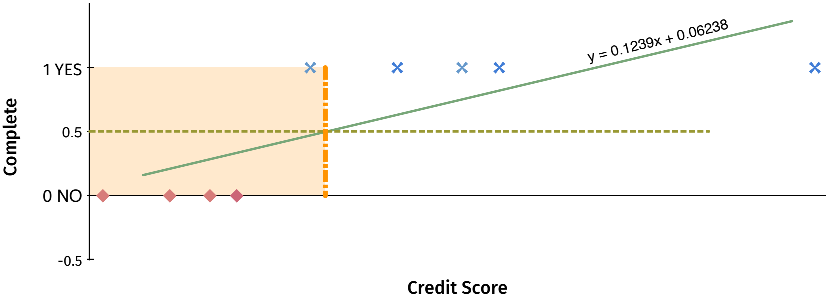
Threshold Classification

If $h(x; \boldsymbol{\theta}) \geqslant \frac{1}{2}$, predict that $y = 1$

If $h(x; \boldsymbol{\theta}) < \frac{1}{2}$, predict that $y = 0$

Classification is Different from Regression

Example: $h(x;\boldsymbol{\theta}) = \boldsymbol{\theta}^T x$



Threshold Classification

If $h(x;\boldsymbol{\theta}) \geqslant \frac{1}{2}$, predict that $y = 1$

If $h(x;\boldsymbol{\theta}) < \frac{1}{2}$, predict that $y = 0$

Classification is Different from Regression

Binary Classification Output:

$$y = 0 \qquad \text{or} \qquad y = 1$$

Classification is Different from Regression

Binary Classification Output:

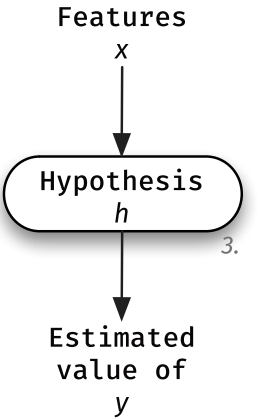
$$y = 0 \qquad \text{or} \qquad y = 1$$

However,

$$h(x; \boldsymbol{\theta}) = \theta^T x < 0 \qquad \text{and} \qquad h(x; \boldsymbol{\theta}) = \boldsymbol{\theta}^T x > 1,$$

is possible.

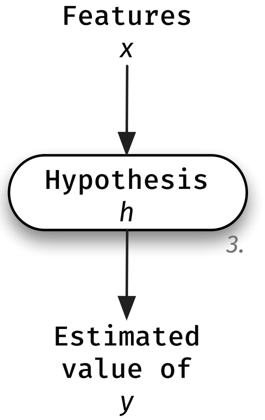
Hypothesis Representation



Recall:

When selecting a learning algorithm, you must decide how to **represent** *h*.

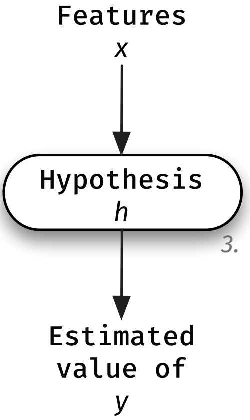
Hypothesis Representation



For binary classification, we want $h(x; \boldsymbol{\theta})$ to take values between 0 and 1:

$$0 \leqslant h(x; \boldsymbol{\theta}) \leqslant 1$$

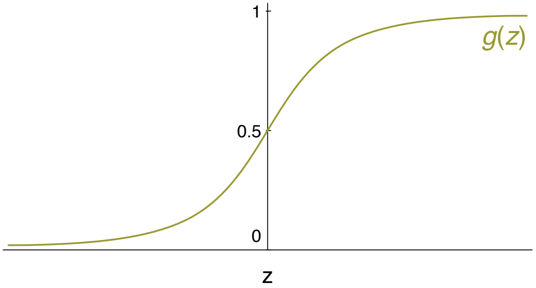
Hypothesis Representation



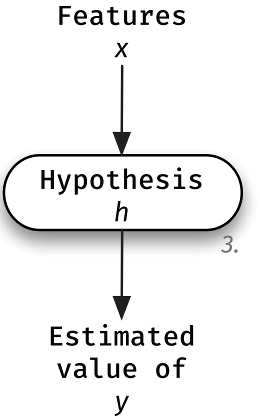
For binary classification, we want $h(x; \boldsymbol{\theta})$ to take values between 0 and 1:

$$0 \leqslant h(x; \boldsymbol{\theta}) \leqslant 1$$

Sigmoid Function $g(\cdot)$



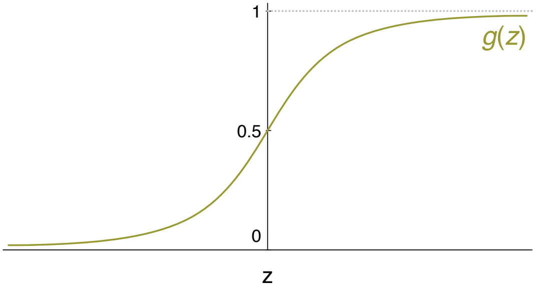
Hypothesis Representation



For binary classification, we want $h(x; \boldsymbol{\theta})$ to take values between 0 and 1:

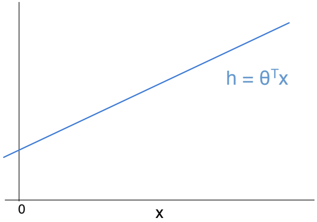
$$0 \leqslant h(x; \boldsymbol{\theta}) \leqslant 1$$

Sigmoid Function $g(\cdot)$



Hypothesis Representation

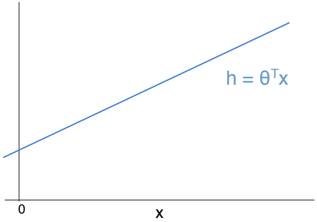
Linear Regression



$$h(x; \boldsymbol{\theta}) = \theta_0 + \theta_1 x$$
$$= \theta^T x$$

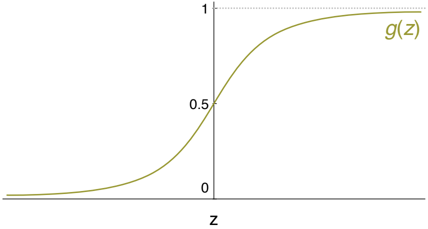
Hypothesis Representation

Linear Regression



$$h(x;\boldsymbol{\theta}) = \theta_0 + \theta_1 x$$
$$= \theta^T x$$

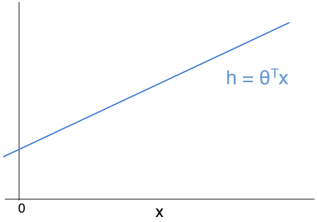
Logistic Regression



$$h(x;\boldsymbol{\theta}) = g(\theta^T x)$$

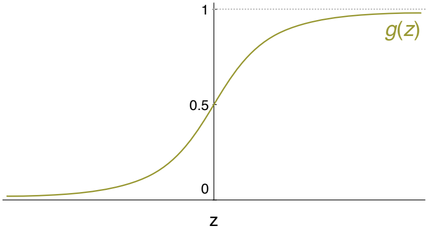
Hypothesis Representation

Linear Regression



$$h(x;\boldsymbol{\theta}) = \theta_0 + \theta_1 x$$
$$= \theta^T x$$

Logistic Regression



$$h(x;\boldsymbol{\theta}) = g(\theta^T x)$$

where

$$g(z) = \frac{1}{1 + \exp^{-1z}}$$

So:

$$h(x;\boldsymbol{\theta}) = \frac{1}{1 + \exp^{-\theta^T x}}$$

Hypothesis Interpretation

Logistic Regression

$$\begin{aligned} h(x; \boldsymbol{\theta}) &= \frac{1}{1 + e^{-\theta^T x}} \\ &= \text{estimated probability that } y = 1 \text{ given data } x \text{ parameterized by } \theta \\ &= P(y = 1 \mid x; \theta) \end{aligned}$$

Hypothesis Interpretation

Logistic Regression

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

$$= \text{estimated probability that } y = 1 \text{ given data } x \text{ parameterized by } \theta$$

$$= P(y = 1 \mid x; \theta)$$

Example:

$h(x;\boldsymbol{\theta}) = .37$ means: 37% chance that applicant will be accepted.

Hypothesis Interpretation

Logistic Regression

$$
\begin{aligned}
h(x; \boldsymbol{\theta}) &= \frac{1}{1 + e^{-\theta^T x}} \\
&= \text{estimated probability that } y = 1 \text{ given data } x \text{ parameterized by } \theta \\
&= P(y = 1 \mid x; \theta) \\
&= 0.37
\end{aligned}
$$

Question: *What is the probability that $y = 0$?*

Hypothesis Interpretation

Logistic Regression

$$
\begin{aligned}
h(x; \boldsymbol{\theta}) &= \frac{1}{1 + e^{-\theta^T x}} \\
&= \text{estimated probability that } y = 1 \text{ given data } x \text{ parameterized by } \theta \\
&= P(y = 1 \mid x; \theta) \\
&= 0.37
\end{aligned}
$$

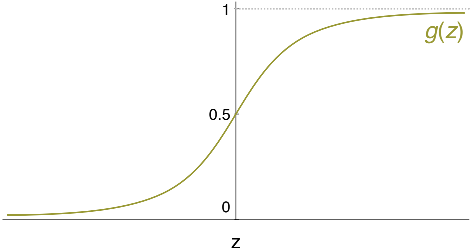
Question: *What is the probability that $y = 0$?*
Since ($y = 0$ xor $y = 1$), apply the **Law of Total Probability**:

$$
\begin{aligned}
P(y = 0 \mid x; \theta) + P(y = 1 \mid x; \theta) &= 1 \qquad \textit{(Law of Total Probability)} \\
P(y = 0 \mid x; \theta) &= 1 - P(y = 1 \mid x; \theta) \\
P(y = 0 \mid x; \theta) &= 1 - 0.37 \\
P(y = 0 \mid x; \theta) &= 0.63
\end{aligned}
$$

Using $g(\cdot)$ for prediction

$$h(x;\boldsymbol{\theta}) = g(\theta^T x)$$
$$= P(y = 1 \mid x; \theta)$$
$$g(z) = \frac{1}{1+e^{-z}}$$

Sigmoid Function $g(z)$



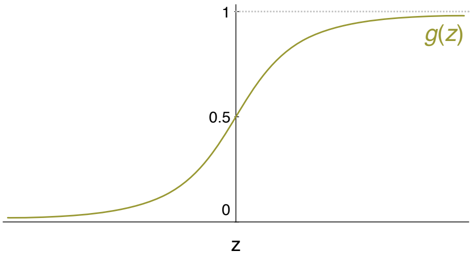
Using $g(\cdot)$ for prediction

$$h(x;\boldsymbol{\theta}) = g(\theta^T x)$$
$$= P(y=1 \mid x;\theta)$$
$$g(z) = \frac{1}{1+e^{-z}}$$

Predict that '$y = 1$' if $h(x;\boldsymbol{\theta}) \geqslant 0.5$

Predict that '$y = 0$' if $h(x;\boldsymbol{\theta}) < 0.5$

Sigmoid Function $g(z)$



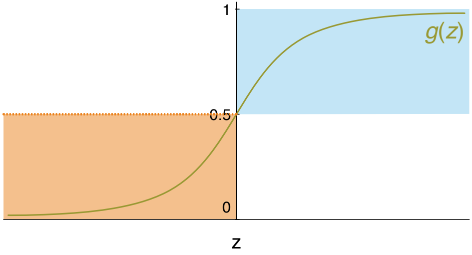
Using $g(\cdot)$ for prediction

$$h(x; \boldsymbol{\theta}) = g(\theta^T x)$$
$$= P(y = 1 \mid x; \theta)$$
$$g(z) = \frac{1}{1 + e^{-z}}$$

Predict that '$y = 1$' if $h(x; \boldsymbol{\theta}) \geqslant 0.5$

Predict that '$y = 0$' if $h(x; \boldsymbol{\theta}) < 0.5$

Sigmoid Function $g(z)$



Using $g(\cdot)$ for prediction

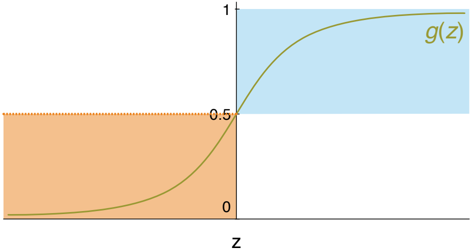
$$h(x; \boldsymbol{\theta}) = g(\theta^T x)$$
$$= P(y = 1 \mid x; \theta)$$

$$g(z) = \frac{1}{1 + e^{-z}}$$

Predict that '$y = 1$' if $h(x; \boldsymbol{\theta}) \geqslant 0.5$
when $z \geqslant 0$

Predict that '$y = 0$' if $h(x; \boldsymbol{\theta}) < 0.5$
when $z < 0$

Sigmoid Function $g(z)$



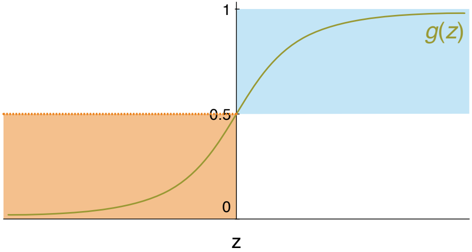
Using $g(\cdot)$ for prediction

$$h(x; \boldsymbol{\theta}) = g(\theta^T x)$$
$$= P(y = 1 \mid x; \theta)$$
$$g(z) = \frac{1}{1 + e^{-z}}$$

Predict that '$y = 1$' if $h(x; \boldsymbol{\theta}) \geqslant 0.5$
when $z \geqslant 0$
when $\theta^T x \geqslant 0$

Predict that '$y = 0$' if $h(x; \boldsymbol{\theta}) < 0.5$
when $z < 0$
when $\theta^T x < 0$

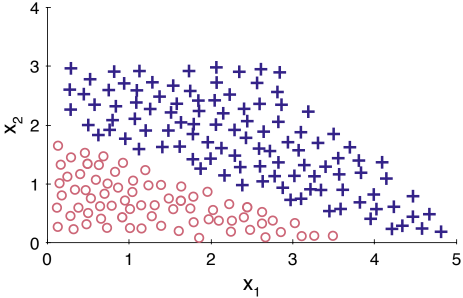
Sigmoid Function $g(z)$



Decision Boundaries

$$
\begin{aligned}
h(x;\boldsymbol{\theta}) &= g(\theta^T x) \\
&= g(\theta_0 + \theta_1 x_1 + \theta_2 x_2) \\
&= \\
&=
\end{aligned}
$$

Decision Boundary

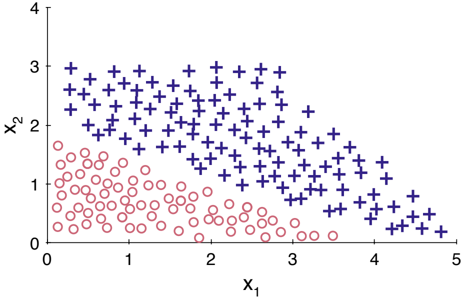


Decision Boundaries

$$
\begin{aligned}
h(x; \boldsymbol{\theta}) &= g(\theta^T x) \\
&= g(\theta_0 + \theta_1 x_1 + \theta_2 x_2) \\
&= g(-4 + x_1 + 2x_2), \quad \textit{where}
\end{aligned}
$$

$$
\theta = \begin{bmatrix} -4 \\ 1 \\ 2 \end{bmatrix}
$$

Decision Boundary



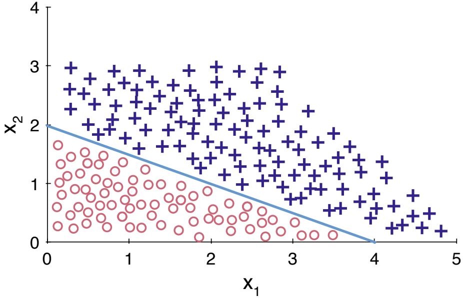
Decision Boundaries

$$
\begin{aligned}
h(x;\boldsymbol{\theta}) &= g(\theta^T x) \\
&= g(\theta_0 + \theta_1 x_1 + \theta_2 x_2) \\
&= g(-4 + x_1 + 2x_2), \quad \textit{where}
\end{aligned}
$$

$$
\theta = \begin{bmatrix} -4 \\ 1 \\ 2 \end{bmatrix}
$$

Predict that '$y = 1$' if $h(x;\boldsymbol{\theta}) \geqslant 0.5$

Decision Boundary

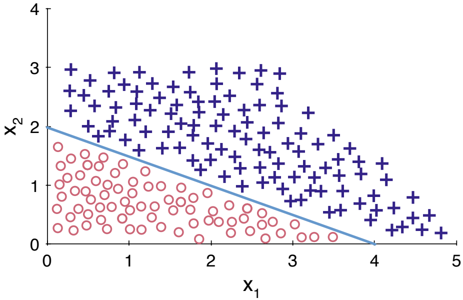


Decision Boundaries

$$h(x;\boldsymbol{\theta}) = g(\theta^T x)$$
$$= g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$
$$= g(-4 + x_1 + 2x_2), \quad \textit{where}$$

$$\theta = \begin{bmatrix} -4 \\ 1 \\ 2 \end{bmatrix}$$

Decision Boundary



Predict that '$y = 1$' if $h(x;\boldsymbol{\theta}) \geqslant 0.5$

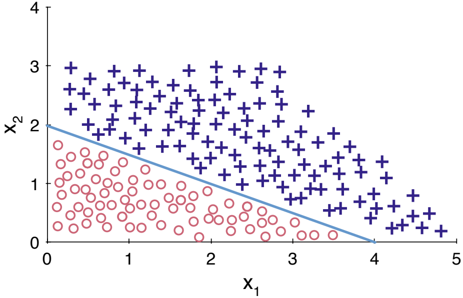
when $-4 + x_1 + 2x_2 \geqslant 0$

Decision Boundaries

$$
\begin{aligned}
h(x;\boldsymbol{\theta}) &= g(\theta^T x) \\
&= g(\theta_0 + \theta_1 x_1 + \theta_2 x_2) \\
&= g(-4 + x_1 + 2x_2), \quad \textit{where}
\end{aligned}
$$

$$
\theta = \begin{bmatrix} -4 \\ 1 \\ 2 \end{bmatrix}
$$

Decision Boundary



Predict that '$y = 1$' if $h(x;\boldsymbol{\theta}) \geqslant 0.5$

when $-4 + x_1 + 2x_2 \geqslant 0$

Predict that '$y = 0$' if $h(x;\boldsymbol{\theta}) < 0.5$

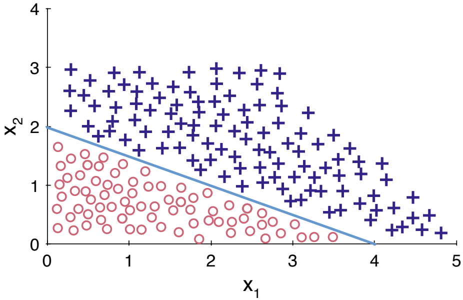
when $-4 + x_1 + 2x_2 < 0$

Decision Boundaries

$$
\begin{aligned}
h(x;\boldsymbol{\theta}) &= g(\theta^T x) \\
&= g(\theta_0 + \theta_1 x_1 + \theta_2 x_2) \\
&= g(-4 + x_1 + 2x_2), \quad \textit{where}
\end{aligned}
$$

$$
\theta = \begin{bmatrix} -4 \\ 1 \\ 2 \end{bmatrix}
$$

Decision Boundary



Predict that '$y = 1$' if $h(x;\boldsymbol{\theta}) \geqslant 0.5$

when $-4 + x_1 + 2x_2 \geqslant 0$

— The Decision Boundary — $x_1 + 2x_2 = 4$

Predict that '$y = 0$' if $h(x;\boldsymbol{\theta}) < 0.5$

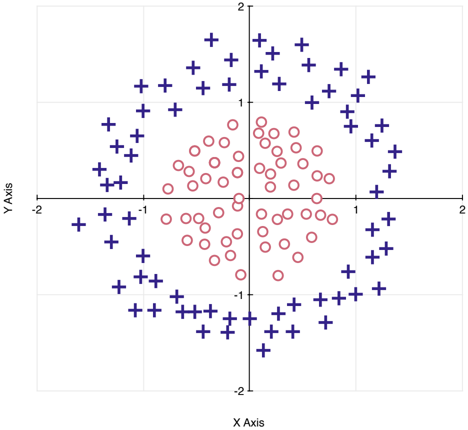
when $-4 + x_1 + 2x_2 < 0$

Non-linear Decision Boundaries

$$\begin{aligned} h(x;\boldsymbol{\theta}) &= g(\theta^T x) \\ &= g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2) \\ &= g(-1 + x_1^2 + x_2^2), \quad \textit{where} \end{aligned}$$

$$\theta = \begin{bmatrix} -1 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

Decision Boundary

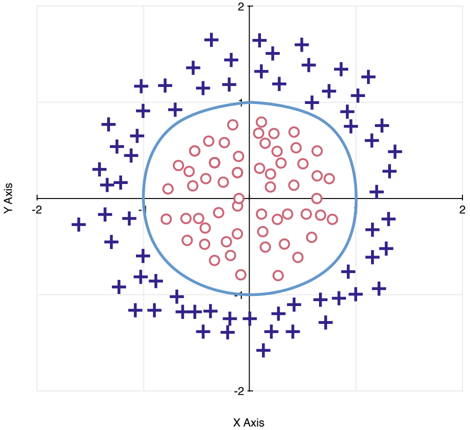


Non-linear Decision Boundaries

$$
\begin{aligned}
h(x; \boldsymbol{\theta}) &= g(\theta^T x) \\
&= g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2) \\
&= g(-1 + x_1^2 + x_2^2), \quad \textit{so:}
\end{aligned}
$$

Decision Boundary: $x_1^2 + 2x_2^2 = 1$

Decision Boundary

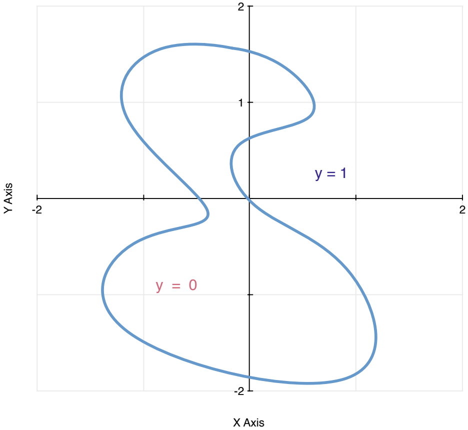


Non-linear Decision Boundaries

Note:

Decision Boundaries are functions of **parameters** rather than functions of data + parameters.

Decision Boundary



How to choose θ?

How to choose θ?

Training set: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})\}$ with m **examples**

How to choose θ?

Training set: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})\}$ with m **examples**

$n + 1$ **Features:** $x \in \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}$, where $x_0 = 1$.

How to choose θ?

Training set: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})\}$ with m **examples**

$n+1$ **Features:** $x \in \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}$, where $x_0 = 1$.

Target variable: $y \in \{0, 1\}$.

How to choose θ?

Training set: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})\}$ with m **examples**

$n+1$ **Features:** $x \in \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}$, where $x_0 = 1$.

Target variable: $y \in \{0, 1\}$.

Form of the Hypothesis: $h(x; \boldsymbol{\theta}) = \dfrac{1}{1 + e^{-\theta^T x}}$

How to choose θ?

Training set: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})\}$ with m **examples**

$n + 1$ **Features:** $x \in \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}$, where $x_0 = 1$.

Target variable: $y \in \{0, 1\}$.

Form of the Hypothesis: $h(x; \boldsymbol{\theta}) = \dfrac{1}{1 + e^{-\theta^T x}}$

How are the values of the vector θ chosen?

How to choose θ

Recall how θ are chosen for Linear Regression:

Linear Regression **Cost Function**:

$$
\begin{aligned}
J(\theta) &= \frac{1}{m}\sum_{i=1}^{m}\frac{1}{2}\left(h(x^{(i)};\boldsymbol{\theta}) - y^{(i)}\right)^2 \\
&= \frac{1}{m}\sum_{i=1}^{m}\text{Cost}\left(h(x^{(i)};\boldsymbol{\theta}), y^{(i)}\right)
\end{aligned}
$$

Simplifying, the cost function for Linear Regression:

$$
\text{Cost}\left(h(x;\boldsymbol{\theta}), y\right) = \frac{1}{2}\left(h(x;\boldsymbol{\theta}) - y\right)^2
$$

Logistic Regression Cost Function

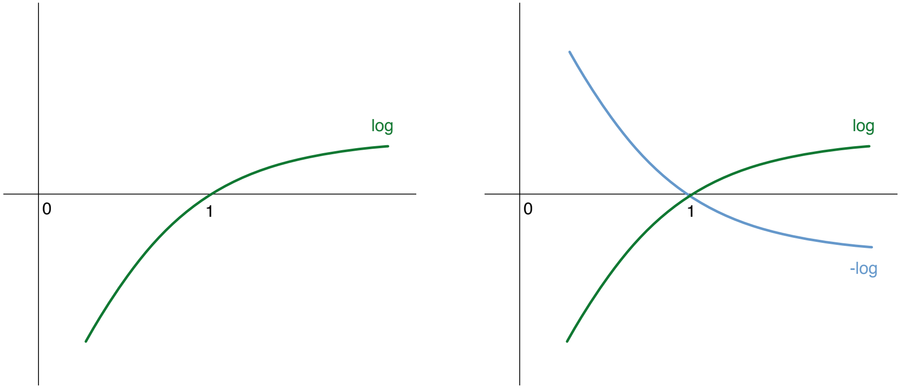
$$\text{Cost}\big(h(x;\boldsymbol{\theta}),y\big) = \begin{cases} -log(h(x;\boldsymbol{\theta})) & \text{if } y = 1 \\ -log(1 - h(x;\boldsymbol{\theta})) & \text{if } y = 0 \end{cases}$$

Intuition:

Logistic Regression Cost Function

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

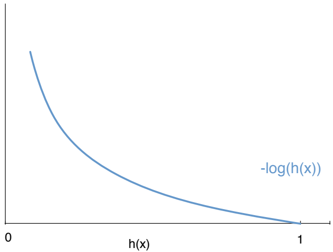
Intuition:



Logistic Regression Cost Function

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

If $y = 1$:



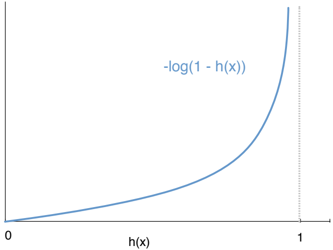
Cost = 0 if $y = 1$ and $h(x;\boldsymbol{\theta}) = 1$;

Cost $\rightarrow \infty$ as $h(x;\boldsymbol{\theta}) \rightarrow 0$

Logistic Regression Cost Function

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

If $y = 0$:



Cost = 0 if $y = 0$ and $h(x;\boldsymbol{\theta}) = 0$;

Cost $\rightarrow \infty$ as $h(x;\boldsymbol{\theta}) \rightarrow 1$

Logistic Regression Cost Function

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

Simplificaton:

$$\mathrm{Cost}(h(x;\boldsymbol{\theta}), y) = -y \log\left(h(x;\boldsymbol{\theta})\right) - (1-y)\log\left(1 - h(x;\boldsymbol{\theta})\right)$$

if $y = 1$:

$$\begin{aligned} \mathrm{Cost}(h(x;\boldsymbol{\theta}), y) &= -1 \log\left(h(x;\boldsymbol{\theta})\right) - (1-1)\log\left(1 - h(x;\boldsymbol{\theta})\right) \\ &= -\log\left(h(x;\boldsymbol{\theta})\right) - 0 \\ &= -\log\left(h(x;\boldsymbol{\theta})\right) \end{aligned}$$

if $y = 0$:

$$\begin{aligned} \mathrm{Cost}(h(x;\boldsymbol{\theta}), y) &= -0 \log(h_\theta x) - (1-0)\log(1 - h(x;\boldsymbol{\theta})) \\ &= 0 - \log(1 - h(x;\boldsymbol{\theta})) \\ &= -\log(1 - h(x;\boldsymbol{\theta})) \end{aligned}$$

Logistic Regression Cost Function

$$
\begin{aligned}
J(\theta) &= \frac{1}{m}\sum_{i=1}^{m} \mathrm{Cost}\big(h(x^{(i)};\boldsymbol{\theta}), y^{(i)}\big) \\
&= -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log\big(h(x;\boldsymbol{\theta})\big) + (1-y^{(i)}) \log\big(1-h(x;\boldsymbol{\theta})\big)\right]
\end{aligned}
$$

Logistic Regression Cost Function

$$\begin{aligned} J(\theta) &= \frac{1}{m}\sum_{i=1}^{m}\text{Cost}\big(h(x^{(i)};\boldsymbol{\theta}), y^{(i)}\big) \\ &= -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)}\log\big(h(x;\boldsymbol{\theta})\big) + (1 - y^{(i)})\log\big(1 - h(x;\boldsymbol{\theta})\big)\right] \end{aligned}$$

$J(\theta)$ is derived from the **Principle of Maximum Likelihood Estimation** (MLE).

Logistic Regression Cost Function

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\mathrm{Cost}\left(h(x^{(i)};\boldsymbol{\theta}), y^{(i)}\right)$$

$$= -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)}\log\left(h(x;\boldsymbol{\theta})\right) + (1-y^{(i)})\log\left(1-h(x;\boldsymbol{\theta})\right)\right]$$

In order to **fit parameters** θ, we first $\min_{\theta} J(\theta)$

Then, given $\min_{\theta} J(\theta)$ and a **new** x, we **make a prediction** for **unknown** y using $h(x;\boldsymbol{\theta})$ calculated by:

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

Gradient Descent and Binary Logistic Regression

Algorithm:
Gradient Descent

» Enter loop
Repeat:

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

for all θ_j simultaneously
Stop at convergence

» Exit loop

Model:
Logistic Regression

Hypothesis:

$$h(x; \boldsymbol{\theta}) = g(\theta^T x)$$

Cost Function:

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

Derivative Term:

$$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \left(h(x^{(i)}; \boldsymbol{\theta}) - y^{(i)}\right) x_j^{(i)}$$

Gradient Descent and Binary Logistic Regression

Algorithm:
Gradient Descent

» Enter loop
Repeat:

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h(x^{(i)}; \boldsymbol{\theta}) - y^{(i)} \right) x_j^{(i)}$$

for all θ_j simultaneously
Stop at convergence

» Exit loop

Remarks:

Gradient Descent and Binary Logistic Regression

Algorithm:
Gradient Descent

» Enter loop
Repeat:

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h(x^{(i)}; \boldsymbol{\theta}) - y^{(i)} \right) x_j^{(i)}$$

for all θ_j simultaneously
Stop at convergence

» Exit loop

Remarks:

Compare to Gradient Descent for Linear Regression:

Gradient Descent and Binary Logistic Regression

Algorithm:
Gradient Descent

» Enter loop
  Repeat:

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h(x^{(i)}; \boldsymbol{\theta}) - y^{(i)} \right) x_j^{(i)}$$

  for all θ_j simultaneously
  Stop at convergence

» Exit loop

Remarks:

Compare to Gradient Descent for Linear Regression:

Same algorithm?!

Gradient Descent and Binary Logistic Regression

Algorithm:
Gradient Descent

» Enter loop
Repeat:

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h(x^{(i)}; \boldsymbol{\theta}) - y^{(i)} \right) x_j^{(i)}$$

for all θ_j simultaneously
Stop at convergence

» Exit loop

Remarks:

Compare to Gradient Descent for Linear Regression:

Same algorithm?!

The Difference:

$$h(x; \boldsymbol{\theta}) = \theta^T x$$

- linear regression

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

- logistic regression

Simplifying the Cost Function

Gradient Descent & Advanced Algorithms

So far, you have learned the fundamentals of regression-based machine learning.

So far, you have learned the fundamentals of regression-based machine learning.

But, to be serious – to deal with thousands (or more) features and millions (or more) training examples – we need more sophisticated algorithms.

Optimization Task

Here's what we know how to do now.

Optimization Task

Here's what we know how to do now.

1. Start with a cost function $J(\theta)$

Optimization Task

Here's what we know how to do now.

1. Start with a cost function $J(\theta)$
2. Goal: $\min_{\theta} J(\theta)$

Optimization Task

Here's what we know how to do now.

1. Start with a cost function $J(\theta)$
2. Goal: $\min\limits_{\theta} J(\theta)$
3. Given θ, compute two things:

$$
\begin{array}{rl}
J(\theta) & \text{the cost of fitting with } \theta \\
\frac{\partial}{\partial \theta_j} J(\theta) & \text{the partial derivatives for each } \theta_j \in \theta
\end{array}
$$

Optimization Task

Here's what we know how to do now.

1. Start with a cost function $J(\theta)$
2. Goal: $\min_{\theta} J(\theta)$
3. Given θ, compute two things:

$$\begin{array}{rl} J(\theta) & \text{the cost of fitting with } \theta \\ \frac{\partial}{\partial \theta_j} J(\theta) & \text{the partial derivatives for each } \theta_j \in \theta \end{array}$$

4. Compute the partial derivative (well, this has been done for you)

Optimization Task

Here's what we know how to do now.

1. Start with a cost function $J(\theta)$
2. Goal: $\min_{\theta} J(\theta)$
3. Given θ, compute two things:

$$
\begin{aligned}
J(\theta) &\quad \text{the cost of fitting with } \theta \\
\frac{\partial}{\partial \theta_j} J(\theta) &\quad \text{the partial derivatives for each } \theta_j \in \theta
\end{aligned}
$$

4. Compute the partial derivative (well, this has been done for you)
5. Plug the derivative term into *Gradient Descent*:

» Enter loop
    Repeat:

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h(x^{(i)}; \boldsymbol{\theta}) - y^{(i)} \right) x_j^{(i)}$$

for all θ_j simultaneously
    Stop at convergence

» Exit loop

Optimization Task

Now let's look at this optimization task anew.

1. Start with a cost function $J(\theta)$
2. Goal: $\min_{\theta} J(\theta)$
3. Given θ, compute two things:

   $J(\theta)$ the cost of fitting with θ

   $\frac{\partial}{\partial \theta_j} J(\theta)$ the partial derivatives for each $\theta_j \in \theta$
4. Optimization algorithms:

   - Gradient Descent

Optimization Task

Now let's look at this optimization task anew.

1. Start with a cost function $J(\theta)$
2. Goal: $\min_{\theta} J(\theta)$
3. Given θ, compute two things:

$$J(\theta) \quad \text{the cost of fitting with } \theta$$

$$\frac{\partial}{\partial \theta_j} J(\theta) \quad \text{the partial derivatives for each } \theta_j \in \theta$$

4. Optimization algorithms:

- Gradient Descent

  however, asymptotic rate of convergence is inferior to more advanced algorithms

- **BFGS**
  - *Broyden-Fletcher-Goldfarb-Shanno* Algorithm

Optimization Task

Now let's look at this optimization task anew.

1. Start with a cost function $J(\theta)$
2. Goal: $\min_{\theta} J(\theta)$
3. Given θ, compute two things:

   $J(\theta)$ the cost of fitting with θ

   $\frac{\partial}{\partial \theta_j} J(\theta)$ the partial derivatives for each $\theta_j \in \theta$
4. Optimization algorithms:

   - Gradient Descent

     however, asymptotic rate of convergence is inferior to more advanced algorithms

   - **BFGS**
     - *Broyden-Fletcher-Goldfarb-Shanno* Algorithm
     - BFGS is a *quasi-Newton* approximation method (stores histories of dense matrices in memory)
     - L-BFGS refers to *limited memory* BFGS (stores vectors as estimates of full matrices)

Advanced Optimization Algorithms

Hessian based methods: use 2nd derivative information

- Broyden-Fletcher-Goldfarb-Shanno (BFGS) Algorithm
- Limited Memory BFGS (L-BFGS)
- Conjugate gradient

Advantages:

- α is picked and adjusted automatically (at every iteration)
- Often faster than gradient descent (i.e., fewer iterations)

Advanced Optimization Algorithms

Hessian based methods: use 2nd derivative information

- Broyden-Fletcher-Goldfarb-Shanno (BFGS) Algorithm
- Limited Memory BFGS (L-BFGS)
- Conjugate gradient

Advantages:

- α is picked and adjusted automatically (at every iteration)
- Often faster than gradient descent (i.e., fewer iterations)

Disadvantages:

- Each step is more computationally / memory expensive
- More complicated to implement / harder to inspect

How do these advanced algorithms work?

Hessian based methods: use 2nd derivative information

- Broyden-Fletcher-Goldfarb-Shanno (BFGS) Algorithm
- Limited Memory BFGS (L-BFGS)
- Conjugate gradient

Line Search Strategy

- Find a descent direction (pos, neg, 0)
  - by gradient descent or Newton's method, etc.
- Compute a step-size in that direction.
- Repeat.

Simulated annealing

- probabilistic technique for finding a **global minima**
  - *for more, look it up!*

How do these advanced algorithms work?

In Python, there is a built-in implementation of BFGS called `fmin_bfgs`.

Let's see an example.

BFGS Example

using **fmin_bfgs** to compute theta_new

```
» theta_new = fmin_bfgs(computeCostFunction,
:         initial_theta, fprime=computeGradient,
:          maxiter=100, args=(X,y), disp=True)
```

BFGS Example

using **fmin_bfgs** to compute theta_new

```
» theta_new = fmin_bfgs(computeCostFunction,
:           initial_theta, fprime=computeGradient,
:            maxiter=100, args=(X,y), disp=True)
```

fmin_bfgs(−1−, −2−, −3−, −4−, −5−, −6−)

BFGS Example

using **fmin_bfgs** to compute theta_new

```
» theta_new = fmin_bfgs(computeCostFunction,
:         initial_theta, fprime=computeGradient,
:          maxiter=100, args=(X,y), disp=True)
```

fmin_bfgs(–1–, –2–, –3–, –4–, –5–, –6–)

1. Optimization function to be minimized
2. Initial parameter vector
3. `fprime` = —
   # optional parameter to
   # compute gradient with —
4. `maxiter` = —
   # maximum number of iterations set to —
5. `args = ( ,  )`
   `# features`
   `# target`
6. `disp` = —
   # optional parameter to print convergence
   # message when set to `True`

BFGS Example

`initial_theta` $\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix}$

Compute Cost Function (J):

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

Gradient (grad):

$$\frac{\partial}{\partial\theta_j} J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left(h(x^{(i)};\boldsymbol{\theta}) - y^{(i)}\right) x_j^{(i)}$$

using **fmin_bfgs** to compute theta_new

```
» theta_new = fmin_bfgs(computeCostFunction,
:         initial_theta, fprime=computeGradient,
:          maxiter=100, args=(X,y), disp=True)
```

fmin_bfgs(–1–, –2–, –3–, –4–, –5–, –6–)

1. Optimization function to be minimized
2. Initial parameter vector
3. `fprime =` —
   # optional parameter to
   # compute gradient with —
4. `maxiter =` —
   # maximum number of iterations set to —
5. `args = ( ,  )`
   `# features`
   `# target`
6. `disp =` —
   # optional parameter to print convergence
   # message when set to `True`

Multi-class Classification

Multi-class Classification

y	$y = 1$	$y = 2$
answer:	yes	no

Multi-class Classification

y	$y = 1$	$y = 2$	$y = 3$
answer:	yes	no	
parents' answer:	yes	no	maybe

Multi-class Classification

y	$y = 1$	$y = 2$	$y = 3$	$y = 4$
answer:	yes	no		
parents' answer:	yes	no	maybe	
cardinal points:	North	South	East	West

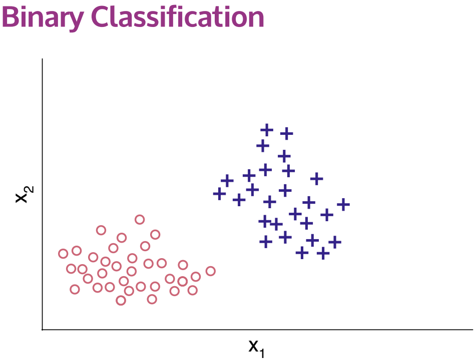
Multi-class Classification

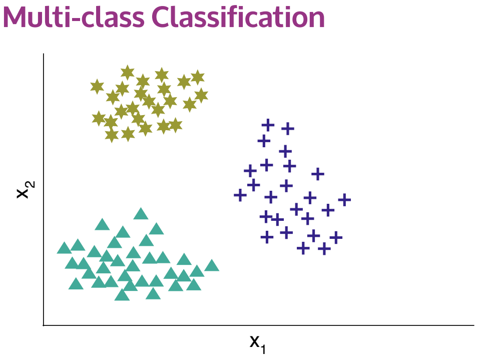
y	$y = 1$	$y = 2$	$y = 3$	$y = 4$	$y = 5$
answer:	yes	no			
parents' answer:	yes	no	maybe		
cardinal points:	North	South	East	West	
fruit:	apples	oranges	pears	bananas	mangos

Multi-class Classification

y	$y = 1$	$y = 2$	$y = 3$	$y = 4$	$y = 5$
answer:	yes	no			
parents' answer:	yes	no	maybe		
cardinal points:	North	South	East	West	
fruit:	apples	oranges	pears	bananas	mangos
$\vdots$					

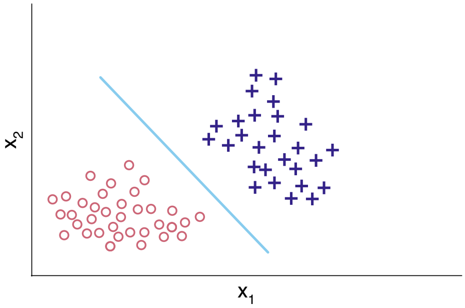
binary vs multi-class classification



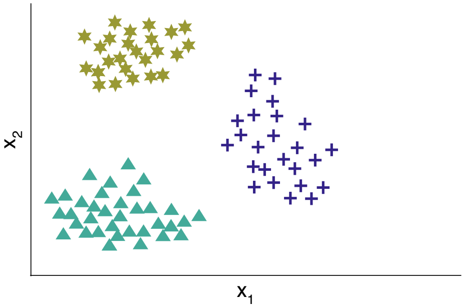


binary vs multi-class classification

Binary Classification

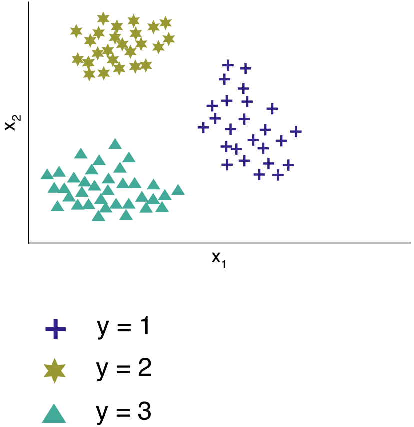


Multi-class Classification



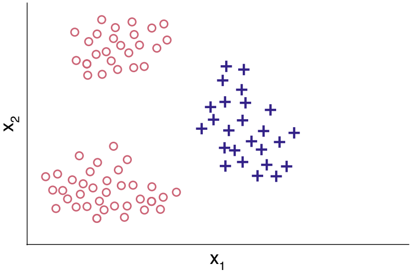
One-vs-Rest (OvR)

Multi-class Classification



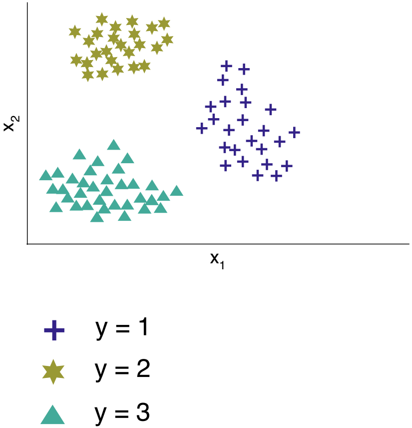
One-vs-Rest (OvR)

Hypothesis for Class 1: $h_\theta^{(1)}(x)$



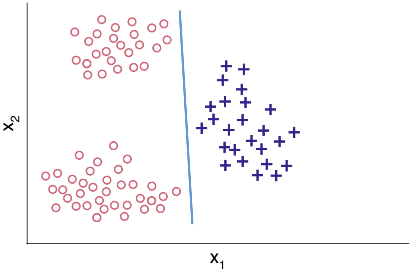
$y = 1$ vs rest (i.e., $y = 2, 3$)

Multi-class Classification



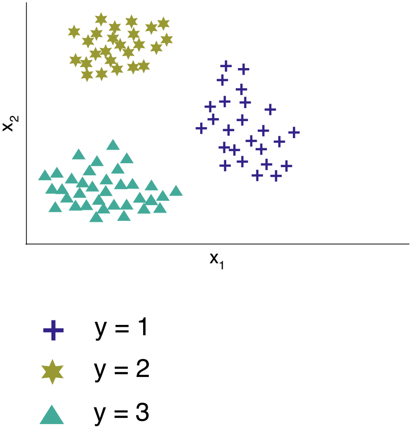
One-vs-Rest (OVR)

Hypothesis for Class 1: $h_\theta^{(1)}(x)$



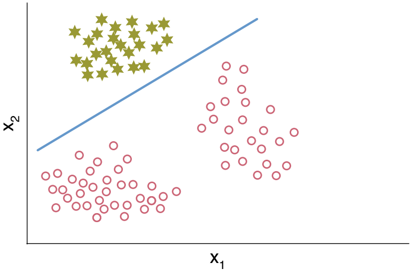
$y = 1$ vs rest (i.e., $y = 2, 3$)

Multi-class Classification



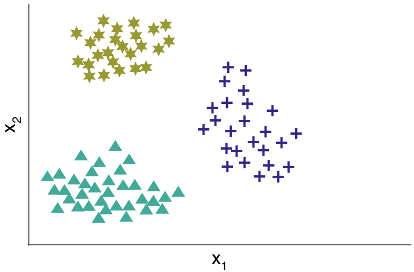
One-vs-Rest (OVR)

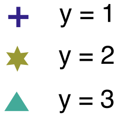
Hypothesis for Class 2: $h_\theta^{(2)}(x)$



$y = 2$ vs rest (i.e., $y = 1, 3$)

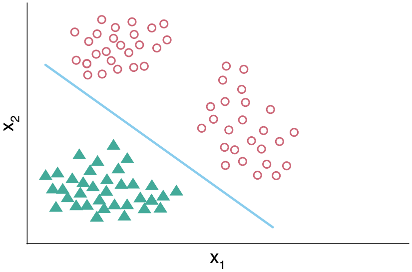
Multi-class Classification





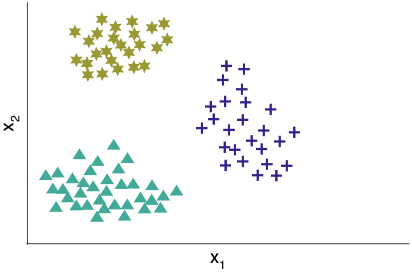
One-vs-Rest (OVR)

Hypothesis for Class 3: $h_\theta^{(3)}(x)$



$y = 3$ vs rest (i.e., $y = 1, 2$)

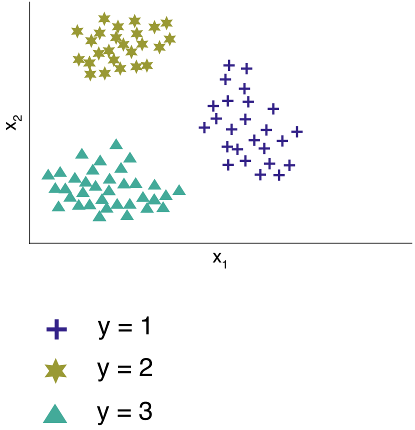
Multi-class Classification



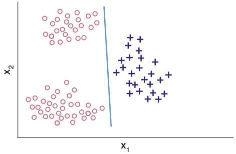
+ y = 1
★ y = 2
▲ y = 3

One-vs-Rest (OVR)

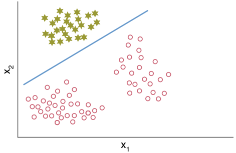
Multi-class Classification



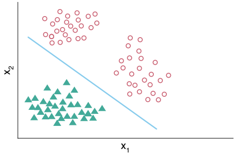
$h_\theta^{(1)}(x)$



$h_\theta^{(2)}(x)$



$h_\theta^{(3)}(x)$



$$h_\theta^{(i)}(x) = P(y = i \mid x; \theta) \quad (\text{for } i = 1, 2, 3)$$

One-vs-rest (OVR)

How to implement OvR:

1. Train a logistic regression classifier $h_\theta^{(i)}(x)$ for **each** class i to predict the probability that $y = i$.

One-vs-rest (OVR)

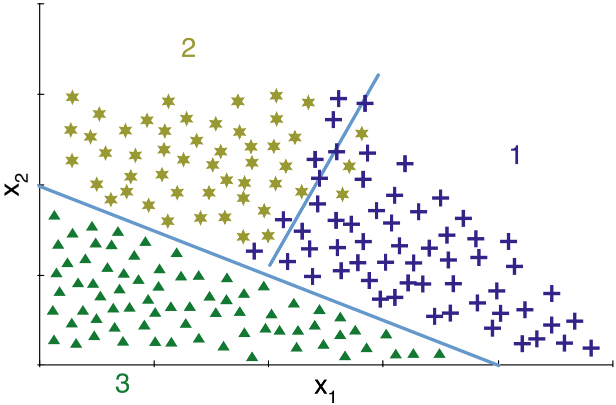
How to implement OvR:

1. Train a logistic regression classifier $h_\theta^{(i)}(x)$ for **each** class i to predict the probability that $y = i$.

2. To make a prediction on a **new** input x, pick the class i that **maximizes**:

$$\max_i h_\theta^{(i)}(x)$$

Example



$$h_\theta^{(1)}(x) = P(y = 1 \mid x; \theta)$$

$$h_\theta^{(2)}(x) = P(y = 2 \mid x; \theta)$$

$$h_\theta^{(3)}(x) = P(y = 3 \mid x; \theta)$$

References