Deep Network Development

Lecture #3

Viktor Varga
Department of Artificial Intelligence, ELTE IK

Last week - Supervised learning

Given: The training sample, a set of (input, label) pairs

$$\{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\}$$

$$x \in X \subset \mathbb{R}^n,\ y \in Y \subset \mathbb{R}^k$$

Task: The estimation of the label (the expected output) from the input

I.e., we search for a (hypothesis-)function h_θ, for which:

$$h_\theta(x) = \hat{y} \approx y$$

Last week - Two main tasks in supervised learning

Regression: Continuous labels (The label set is infinite)

$$|Y| = \infty$$

Example: Number of cars or the age of a person

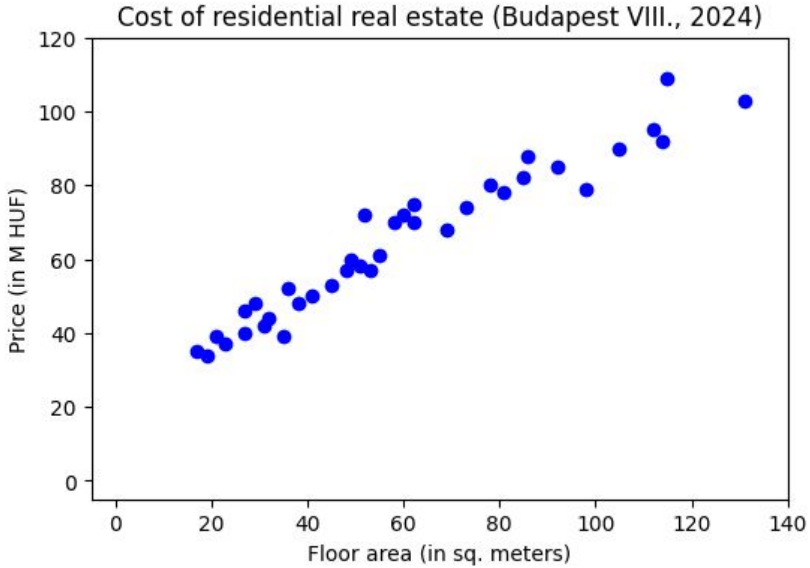
A simple method: Linear regression

Last week - Regression

Example: Let's estimate the price of apartments based on their floor area!

x: The floor area of a given apartment

y: The price of the given apartment



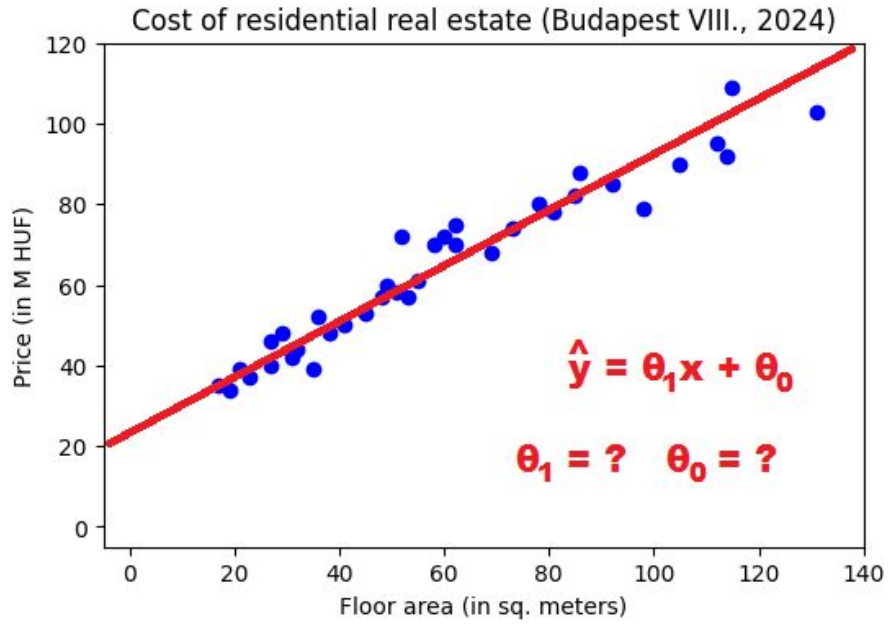
$$x^{(j)}, y^{(j)} \in \mathbb{R}$$

Last week - Linear regression

Its hypothesis function (h):

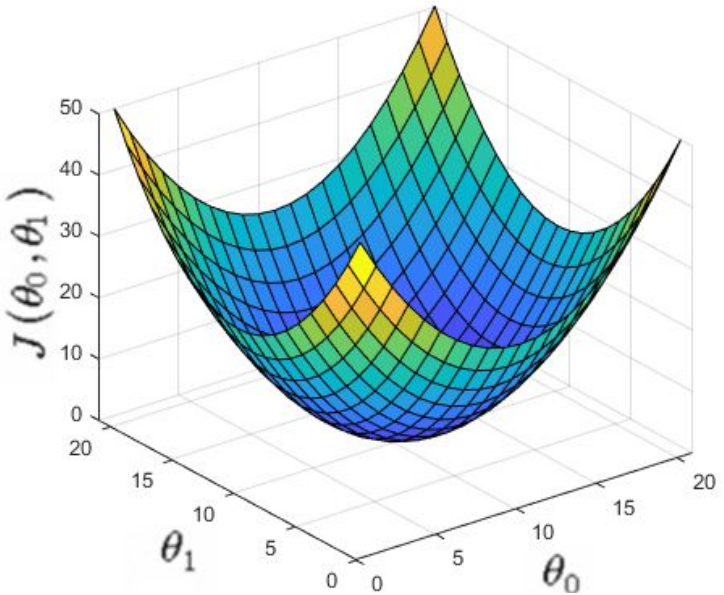
$$y \approx \hat{y} = h(x) = \theta_1 x + \theta_0$$

$$\theta_1, \theta_0 \in \mathbb{R}$$



Last week - Lin. reg., the least squares method

Its loss function (J)**:** quadratic, bivariate (θ_0, θ_1): an elliptic paraboloid

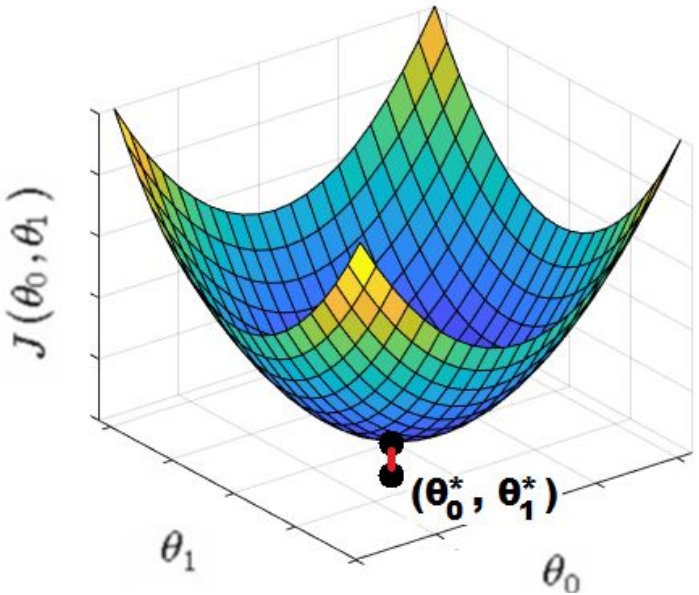


$$h(x^{(j)}) = \hat{y}^{(j)}$$

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{j=1}^{m} (\theta_1 x^{(j)} + \theta_0 - y^{(j)})^2$$

Last week - Lin. reg., the least squares method

Goal: Find those parameters for the hypothesis function which minimize the loss.



$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{j=1}^{m} (\theta_1 x^{(j)} + \theta_0 - y^{(j)})^2$$

$$\theta_0^*, \theta_1^* = \underset{\theta_0, \theta_1}{\operatorname{argmin}} J(\theta_0, \theta_1)$$

Last week - Lin. reg., the least squares method

We search for the optimal parameters using the **gradient method**.

Gradient method: Iteratively modify the parameters in the direction where the loss function has the steepest slope downwards.

The direction of greatest steepness (upwards) is given by the **gradient vector**.

$$\nabla J(\theta'_0, \theta'_1) = \begin{bmatrix} \frac{\partial}{\partial \theta_0} J(\theta'_0, \theta'_1) \\ \frac{\partial}{\partial \theta_1} J(\theta'_0, \theta'_1) \end{bmatrix}$$

Last week - Lin. reg., the least squares method

We search for the optimal parameters using the **gradient method**.

Gradient method: Iteratively modify the parameters in the direction where the loss function has the steepest slope downwards.

The direction of greatest steepness (upwards) is given by the **gradient vector**.

$$\nabla J(\theta'_0, \theta'_1) = \begin{bmatrix} \frac{\partial}{\partial \theta_0} J(\theta'_0, \theta'_1) \\ \frac{\partial}{\partial \theta_1} J(\theta'_0, \theta'_1) \end{bmatrix}$$

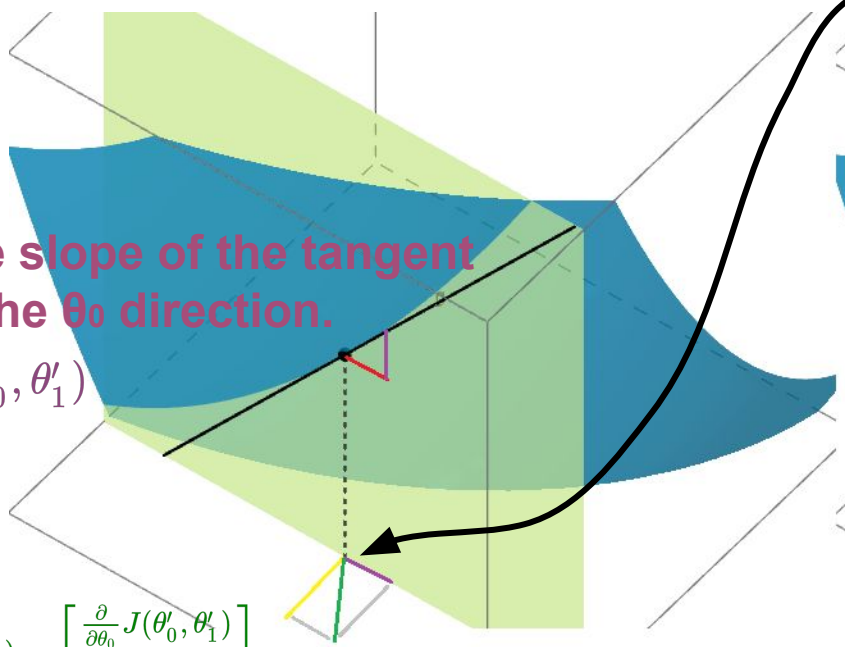
The partial derivative of the **J** loss function with respect to θ_0 in the point of the current parameter values (θ'_0, θ'_1).

The partial derivative of the **J** loss function with respect to θ_1 in the point of the current parameter values (θ'_0, θ'_1).

Last week - Lin. reg., the least squares method

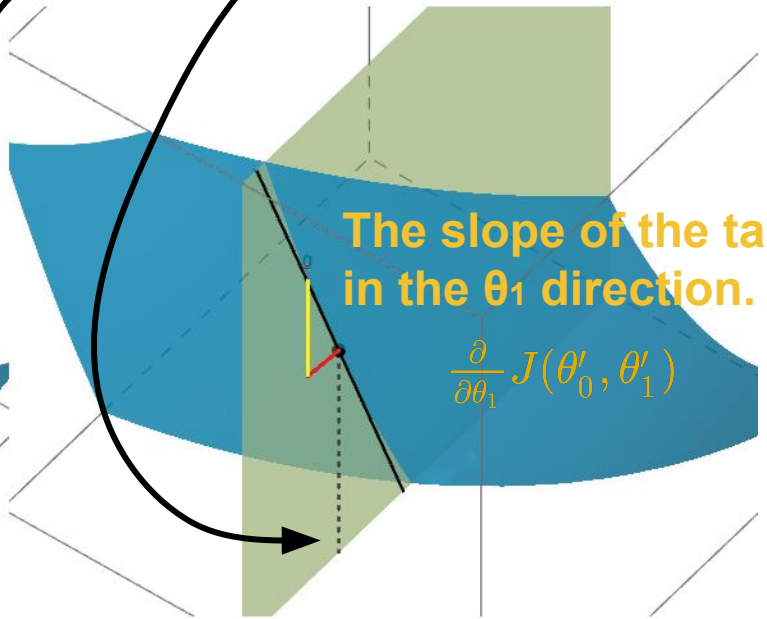
The partial derivatives of the loss function

The point of the actual parameters (θ'_0, θ'_1)



The slope of the tangent in the θ₀ direction.

$$\frac{\partial}{\partial \theta_0} J(\theta'_0, \theta'_1)$$



The slope of the tangent in the θ₁ direction.

$$\frac{\partial}{\partial \theta_1} J(\theta'_0, \theta'_1)$$

$$\nabla J(\theta'_0, \theta'_1) = \begin{bmatrix} \frac{\partial}{\partial \theta_0} J(\theta'_0, \theta'_1) \\ \frac{\partial}{\partial \theta_1} J(\theta'_0, \theta'_1) \end{bmatrix}$$

The gradient vector: direction of the steepest increase

Last week - Lin. reg., the least squares method

The partial derivatives of the loss function

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{j=1}^{m} (\theta_1 x^{(j)} + \theta_0 - y^{(j)})^2$$

In **partial differentiation**, we differentiate the function with respect to **one variable**. In this case, **we treat the other variables as constants**.

$$\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{j=1}^{m} (\theta_1 x^{(j)} + \theta_0 - y^{(j)})$$

The slope of the tangent in the **θ₀** direction.

$$\frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{j=1}^{m} (\theta_1 x^{(j)} + \theta_0 - y^{(j)}) \cdot x^{(j)}$$

The slope of the tangent in the **θ₁** direction.

Last week - Lin. reg., the least squares method

The gradient descent algorithm with two parameters:

repeat until convergence {

$$grad_0 := \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$

$$grad_1 := \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$

$$\theta_0 := \theta_0 - \alpha \cdot grad_0$$

$$\theta_1 := \theta_1 - \alpha \cdot grad_1$$

}

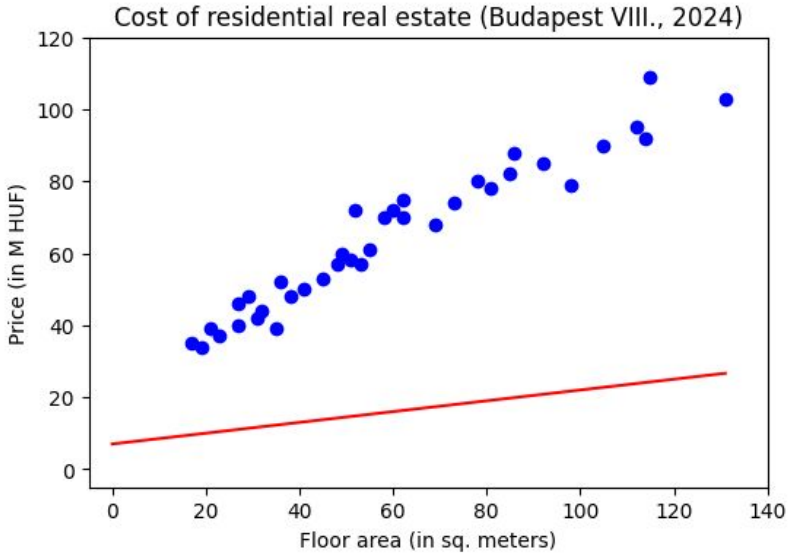
$$\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{j=1}^{m} (\theta_1 x^{(j)} + \theta_0 - y^{(j)})$$

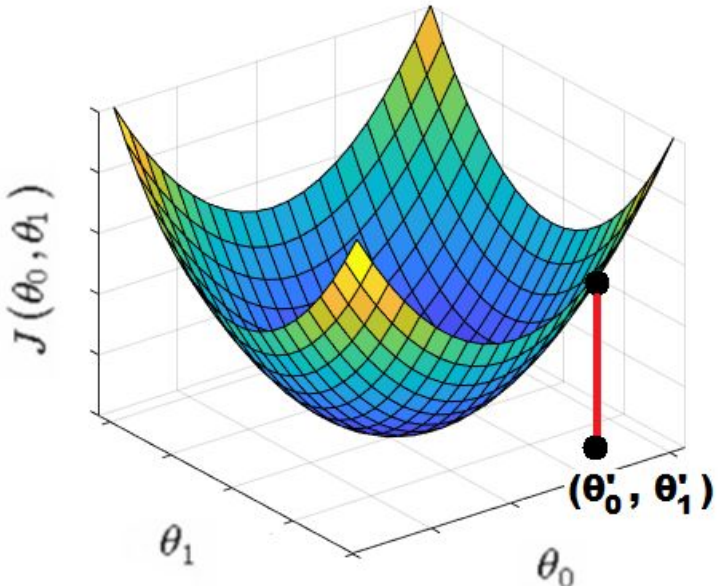
$$\frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{j=1}^{m} (\theta_1 x^{(j)} + \theta_0 - y^{(j)}) \cdot x^{(j)}$$

alpha: the learning rate;
the size of the steps can be scaled with it

Last week - Lin. reg., the least squares method

Applying gradient descent, T = 0 (before taking the first step)

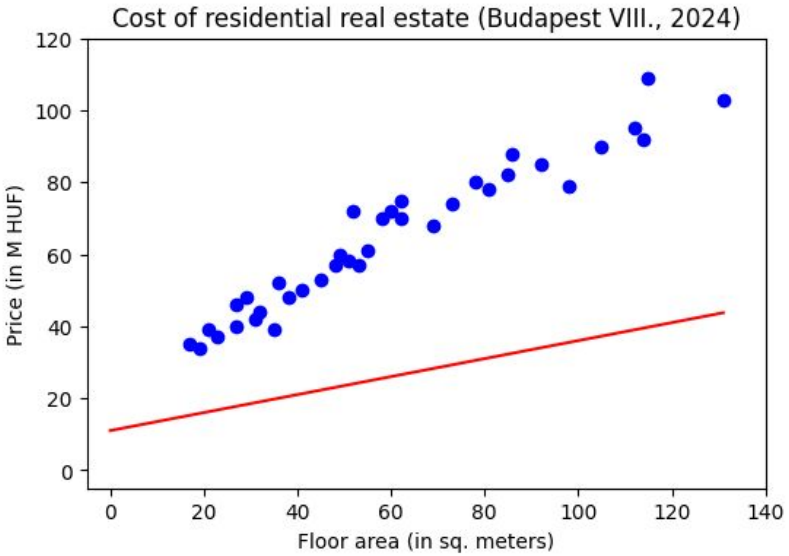


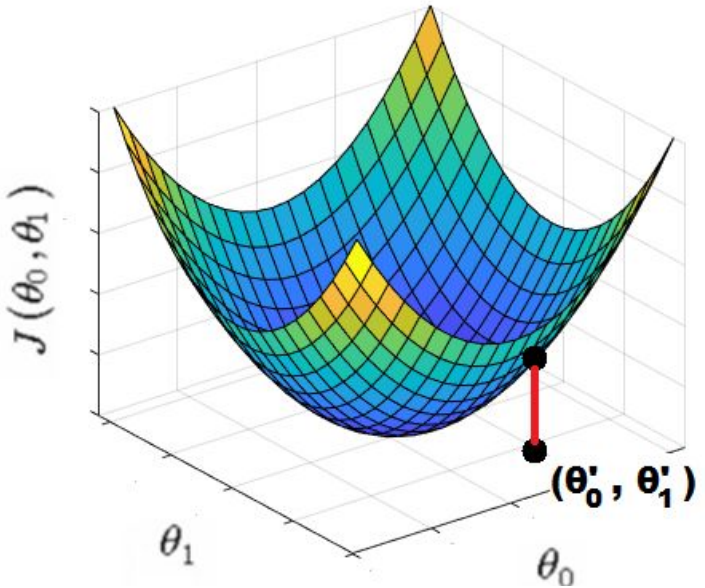


We can choose the initial parameters (θ_0, θ_1) randomly.

Last week - Lin. reg., the least squares method

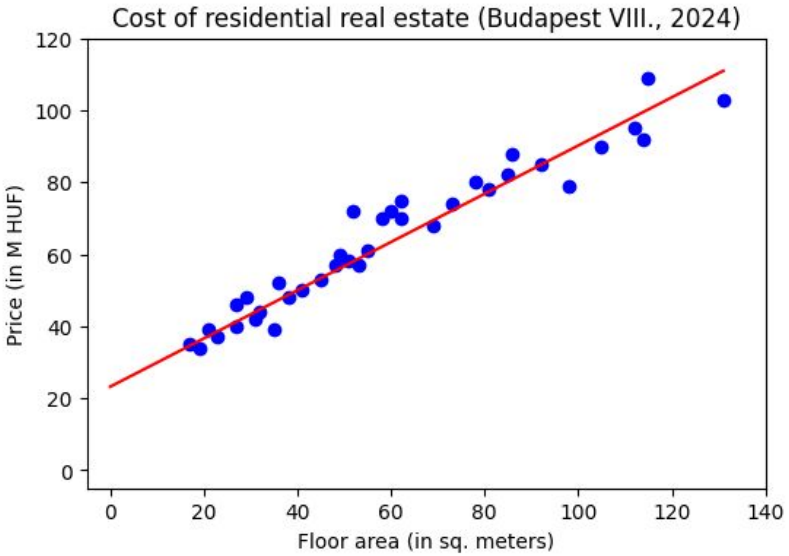
Applying gradient descent, T = 1

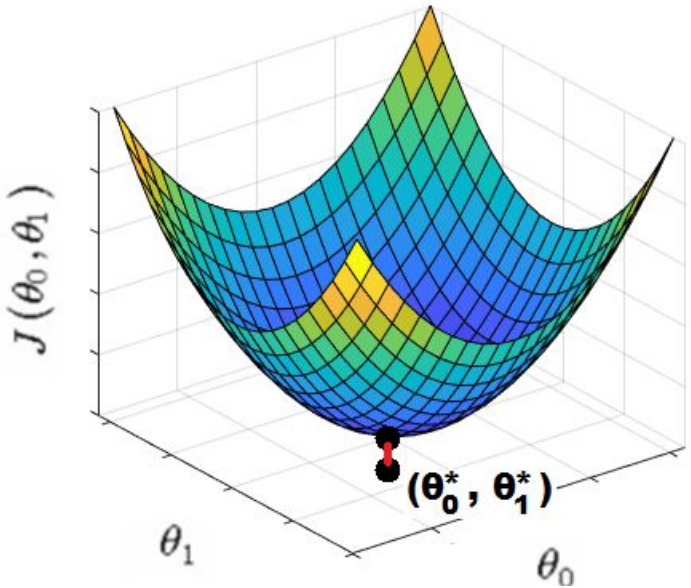




Last week - Lin. reg., the least squares method

Applying gradient descent, T = < many >



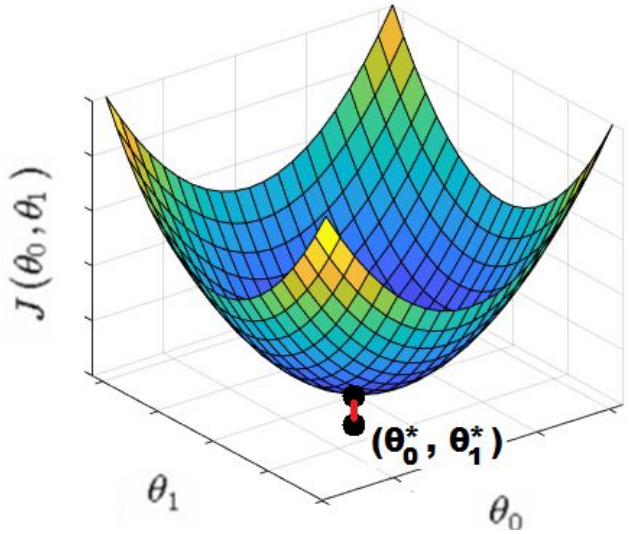


Last week - Lin. reg., the least squares method

With an appropriate (alpha) step size, we are **guaranteed to find the global minimum of the linear regression** (MSE) **loss function** (i.e., the optimal parameters) using the gradient method.

Linear regression is special:
the loss function is **convex**.

If this was not the case, finding only one of the **local minima** is guaranteed.

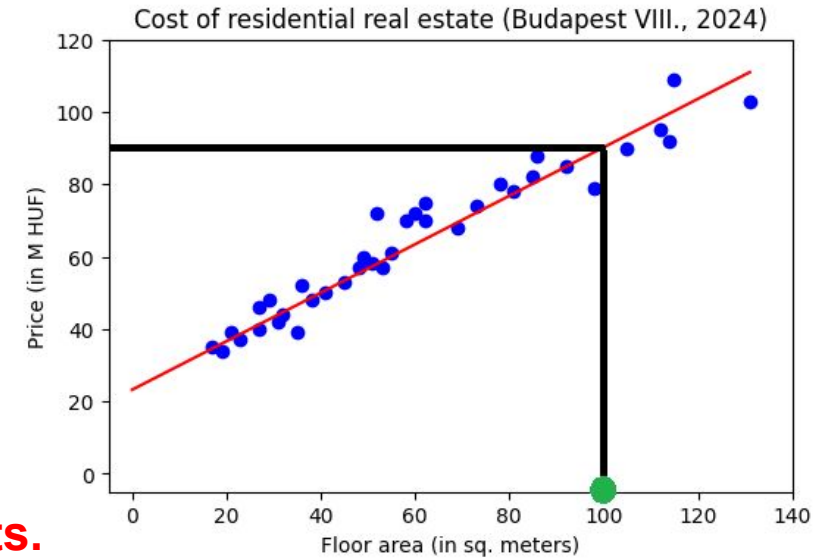


Last week - Lin. reg., the least squares method

What have we achieved?

We trained a simple (linear) regression model.

Using it, we will be able to estimate labels for new, unlabelled data.

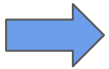


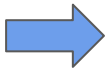
The parameters of the trained model:
$\theta_0 = 23.159$, $\theta_1 = 0.6681$

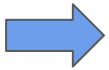
Therefore, for a 100 m^2 apartment, we estimate a price of:
23.159 + 100 * 0.6681 = **89.97 (million) forints.**

Univariate linear regression

Example tasks so far:

x: Population of a city  **y:** Number of cars in the city

x: Floor area of a flat  **y:** Price of the flat

x: Weight of a patient  **y:** Cholesterol levels of the patient

Univariate linear regression

Example tasks so far:

x: Population of a city → **y:** Number of cars in the city

x: Floor area of a flat → **y:** Price of the flat

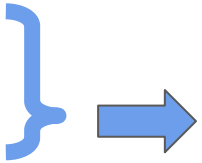
x: Weight of a patient → **y:** Cholesterol levels of the patient

We can only learn overly simplified models this way...
Many other factors can influence the value of the label.

Multivariate linear regression

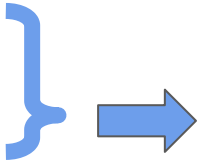
New example tasks with multiple input variables (features)**:**

x_1: Population of a city
x_2: GDP per capita in the city



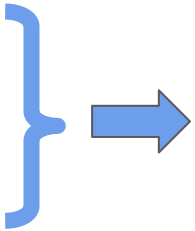
y: Number of cars in the city

x_1: Floor area of a flat
x_2: Distance from city center



y: Price of the flat

x_1: Weight of a patient
x_2: Age of a patient
x_3: Sex of a patient



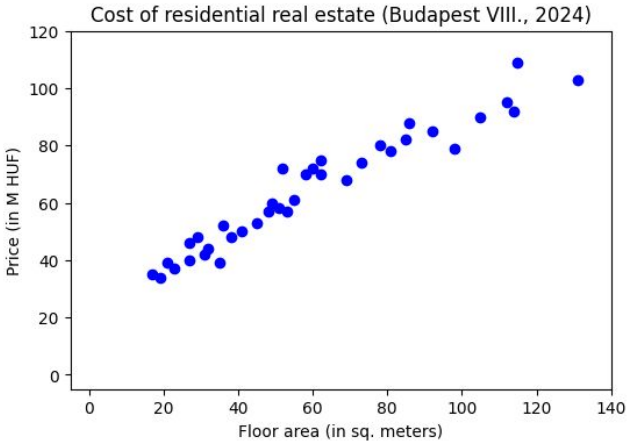
y: Cholesterol levels of the patient

Multivariate linear regression

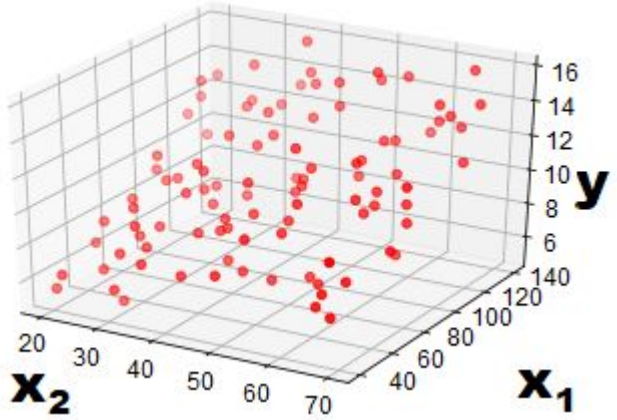
two input variables

Until now: Univariate sample

From now on: Bivariate sample



$$x^{(j)}, y^{(j)} \in \mathbb{R}$$



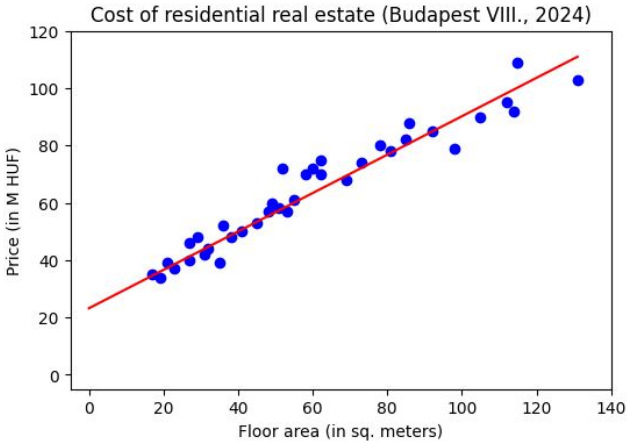
$$x^{(j)} \in \mathbb{R}^2, y^{(j)} \in \mathbb{R}$$

Multivariate linear regression

two input variables

Until now: Univariate sample

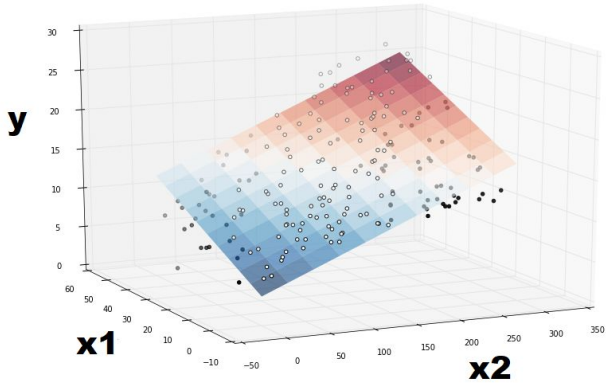
Hypothesis: Line on a plane



$$x^{(j)}, y^{(j)} \in \mathbb{R}$$

From now on: Bivariate sample

Hypothesis: Plane in 3D space



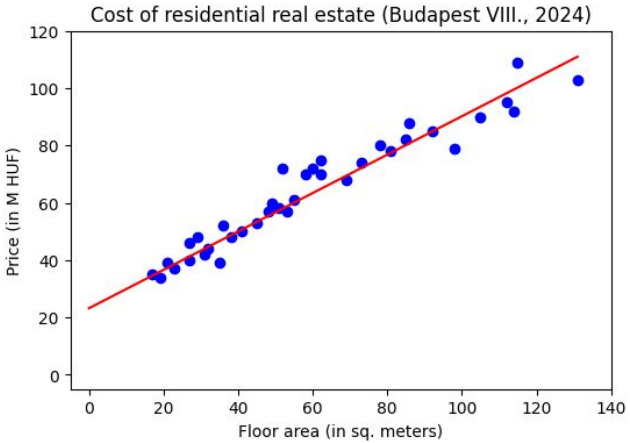
$$x^{(j)} \in \mathbb{R}^2, y^{(j)} \in \mathbb{R}$$

Multivariate linear regression

n input variables

Until now: Univariate sample

Hypothesis: Line on a plane



$$x^{(j)}, y^{(j)} \in \mathbb{R}$$

From now on: Multivariate sample

Hypothesis:

n dimensional hyperplane in an **n+1 dimensional space**

$$x^{(j)} \in \mathbb{R}^n, y^{(j)} \in \mathbb{R}$$

Multivariate linear regression

Hypothesis function - **one input variable:**

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

Hypothesis function - **two input variables:**

$$y \approx \hat{y} = h(x) = \theta_2 x_2 + \theta_1 x_1 + \theta_0$$

Hypothesis function - **n input variables:**

$$y \approx \hat{y} = h(x) = \theta_n x_n + \theta_{n-1} x_{n-1} + \cdots + \theta_1 x_1 + \theta_0$$

Multivariate linear regression

Hypothesis function - **n input variables:**

$$y \approx \hat{y} = h(x) = \theta_n x_n + \theta_{n-1} x_{n-1} + \cdots + \theta_1 x_1 + \theta_0$$

→ **n** input variables, **n+1** parameters

I.e., the **loss function J:**

$$J : \mathbb{R}^{n+1} \rightarrow \mathbb{R}$$

$$h(x^{(j)}) = \hat{y}^{(j)}$$

$$J(\theta_0, \theta_1, \ldots, \theta_n) = \frac{1}{2m} \sum_{j=1}^{m} (\theta_n x_n^{(j)} + \theta_{n-1} x_{n-1}^{(j)} + \cdots + \theta_1 x_1^{(j)} + \theta_0 - y^{(j)})^2$$

Multivariate linear regression

Hypothesis function - **n input variables:**

$$y \approx \hat{y} = h(x) = \theta_n x_n + \theta_{n-1} x_{n-1} + \cdots + \theta_1 x_1 + \theta_0$$

Is there a simpler way to formulate it?

Multivariate linear regression

Hypothesis function - **n input variables:**

$$y \approx \hat{y} = h(x) = \theta_n x_n + \theta_{n-1} x_{n-1} + \cdots + \theta_1 x_1 + \theta_0$$

Is there a simpler way to formulate it?

In vector form: The estimated label is
the dot product of the input variables and parameters!

$$y \approx \hat{y} = h(x^{(j)}) = \langle \theta, x^{(j)} \rangle = \sum_{i=0}^{n} \theta_i x_i$$

Multivariate linear regression

Hypothesis function - **n input variables:**

$$x_0 := 1$$

$$y \approx \hat{y} = h(x) = \theta_n x_n + \theta_{n-1} x_{n-1} + \cdots + \theta_1 x_1 + \theta_0 x_0$$

In vector form: The estimated label is the dot product of the input variables and parameters!

$$y \approx \hat{y} = h(x^{(j)}) = \langle \theta, x^{(j)} \rangle = \sum_{i=0}^{n} \theta_i x_i$$

An extra “dummy variable,” **x_0**, must be introduced to obtain the dot product. **The value of x_0 is always 1.**

Multivariate linear regression

Hypothesis function - **n input variables:**

$$x_0 := 1$$

$$y \approx \hat{y} = h(x) = \theta_n x_n + \theta_{n-1} x_{n-1} + \cdots + \theta_1 x_1 + \theta_0 x_0$$

$$y \approx \hat{y} = h(x^{(j)}) = \langle \theta, x^{(j)} \rangle = \sum_{i=0}^{n} \theta_i x_i$$

The hypothesis function can be written for all data points at once:

$$y \approx \hat{y} = h(x) = X\theta$$

$$X \in \mathbb{R}^{m \times n}$$

Multivariate linear regression

$$\langle x^{(j)}, \theta \rangle = \sum_{i=0}^{n} x_i^{(j)} \theta_i$$

dot product

$$\theta = \begin{bmatrix} \theta_0 \\ \dots \\ \theta_n \end{bmatrix} \in \mathbb{R}^n$$

$$x = \begin{bmatrix} 1 & x_1^{(1)} & \dots & x_n^{(1)} \\ \dots & \dots & \dots & \dots \\ 1 & x_1^{(m)} & \dots & x_n^{(m)} \end{bmatrix} \quad \langle x^{(1)}, \theta \rangle = \quad \dots \quad \hat{y}^{(1)} \approx \begin{bmatrix} y^{(1)} \\ \dots \\ y^{(m)} \end{bmatrix} \in \mathbb{R}^m$$

$$h(x) = X\theta = \hat{y} \approx y$$

Multivariate linear regression

$$\langle x^{(j)}, \theta \rangle = \sum_{i=0}^{n} x_i^{(j)} \theta_i$$

dot product

$$\theta = \begin{bmatrix} \theta_0 \\ \dots \\ \theta_n \end{bmatrix} \in \mathbb{R}^n$$

$$x = \begin{bmatrix} 1 & x_1^{(1)} & \dots & x_n^{(1)} \\ \dots & \dots & \dots & \dots \\ 1 & x_1^{(m)} & \dots & x_n^{(m)} \end{bmatrix} \quad \langle x^{(1)}, \theta \rangle = \quad \dots \quad \hat{y}^{(1)} \approx \begin{bmatrix} y^{(1)} \\ \dots \\ y^{(m)} \end{bmatrix} \in \mathbb{R}^m$$

$$h(x) = X\theta = \hat{y} \approx y$$

```
y_pred = np.dot(X, theta)
```

Multivariate linear regression

Gradient descent - multivariate case

repeat until convergence $\{$

  for i $\leftarrow 1 \ldots n$ $\{$

    $grad_i = \frac{\partial}{\partial \theta_i} J(\theta)$

  $\}$

  for i $\leftarrow 1 \ldots n$ $\{$

    $\theta_i = \theta_i - \alpha \, grad_i$

  $\}$

$\}$

$$\frac{\partial}{\partial \theta_i} J(\theta) = \frac{1}{m} \sum_{j=1}^{m} (\theta_n x_n^{(j)} + \cdots + \theta_0 x_0^{(j)} - y^{(j)}) x_i^{(j)}$$

The slope of the tangent in the θ_i direction

Multivariate linear regression

Gradient descent - multivariate case

repeat until convergence {

for i $\leftarrow 1 \ldots n$ {

$grad_i = \frac{\partial}{\partial \theta_i} J(\theta)$

}

for i $\leftarrow 1 \ldots n$ {

$\theta_i = \theta_i - \alpha \, grad_i$

}

}

$$\frac{\partial}{\partial \theta_i} J(\theta) = \frac{1}{m} \sum_{j=1}^{m} (\theta_n x_n^{(j)} + \cdots + \theta_0 x_0^{(j)} - y^{(j)}) x_i^{(j)}$$

1) Calculate the gradient vector:

Its elements are the partial derivatives of the loss function with respect to each parameter θ_i.

The gradient vector indicates the direction in the current parameters in which the loss function grows the most.

2) Update the parameters:

We subtract the gradient vector multiplied by the learning rate from the old parameters.

(We take one step in the direction of the steepest decrease of the loss function...)

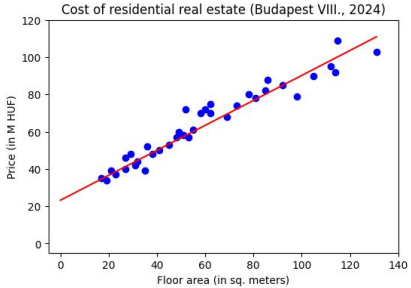
Multivariate linear regression

Gradient descent - multivariate case

The loss function remains quadratic, only the number of variables has increased.

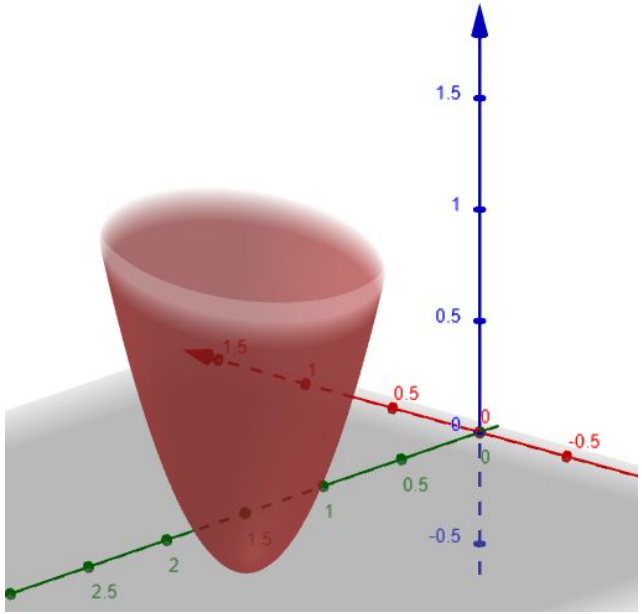
→ It is still convex, so we are **guaranteed to find the optimal solution**.

Feature scaling



Problem:

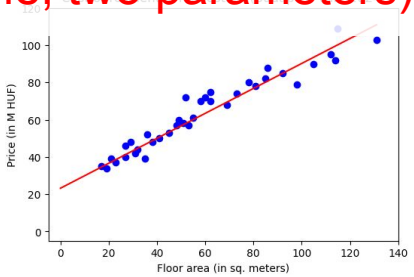
$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{j=1}^{m} (\theta_1 x^{(j)} + \theta_0 - y^{(j)})^2$$



The loss function, as we wish to see…

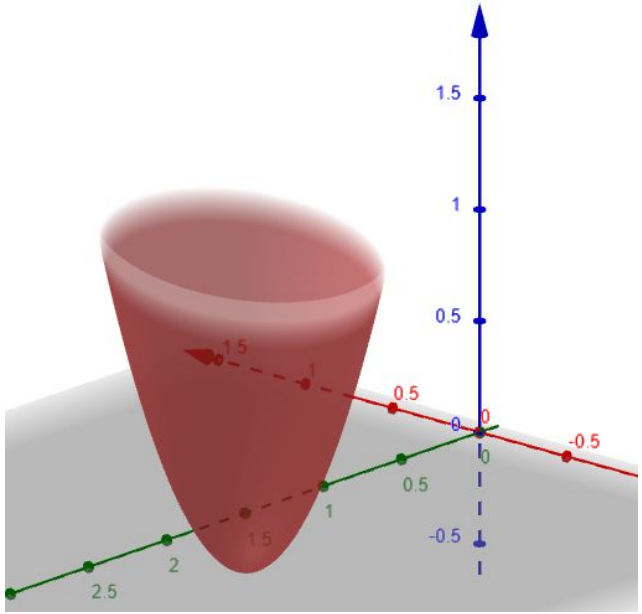
(Visualization: a single input variable, two parameters)

Feature scaling

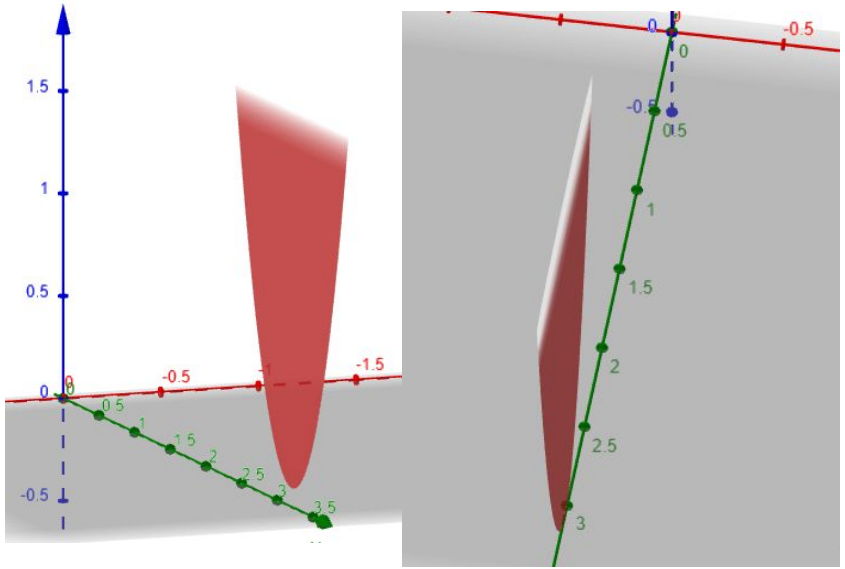


Problem:

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{j=1}^{m} (\theta_1 x^{(j)} + \theta_0 - y^{(j)})^2$$



The loss function, as we wish to see…



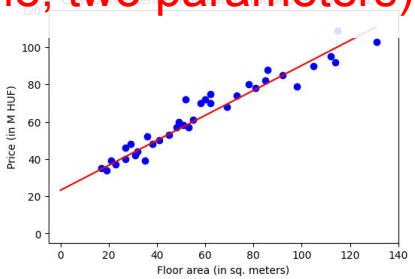
The loss function in reality

(Visualization: a single input variable, two parameters)

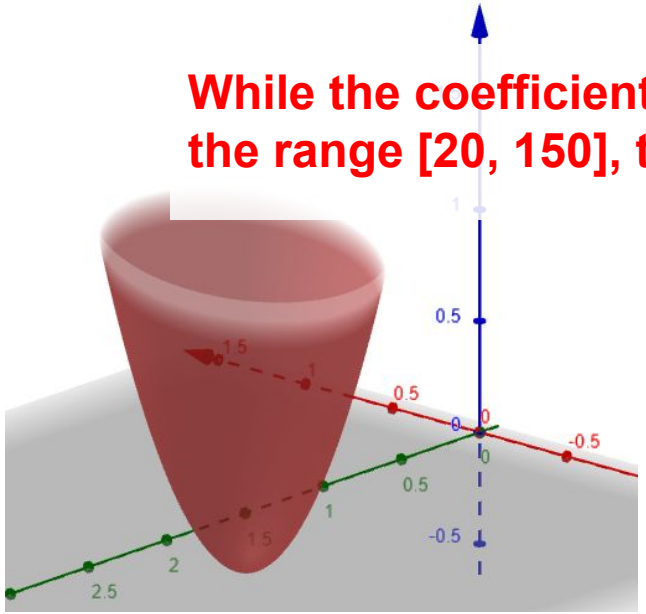
Feature scaling

Problem:

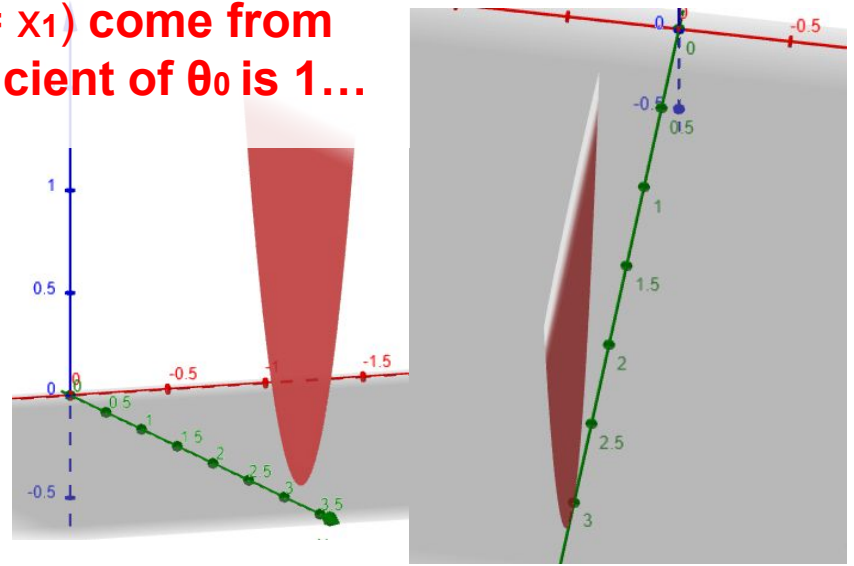
$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{j=1}^{m} (\theta_1 x^{(j)} + \theta_0 - y^{(j)})^2$$



While the coefficients of θ_1 (= x_1) come from the range [20, 150], the coefficient of θ_0 is 1…



The loss function, as we wish to see…



The loss function in reality

Image sources: Jannes Klaas, kaggle.com

Feature scaling

Problem: Gradient descent does not work well on such surfaces...

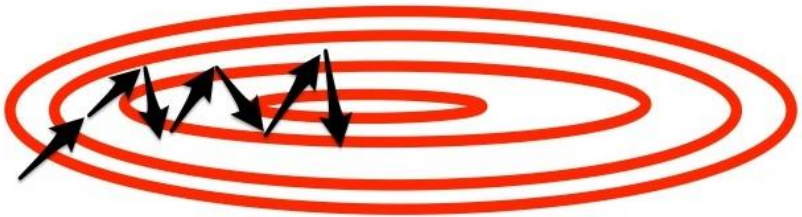


Image sources: Jannes Klaas, kaggle.com

Feature scaling

Feature scaling: Scaling input variables to the same order of magnitude

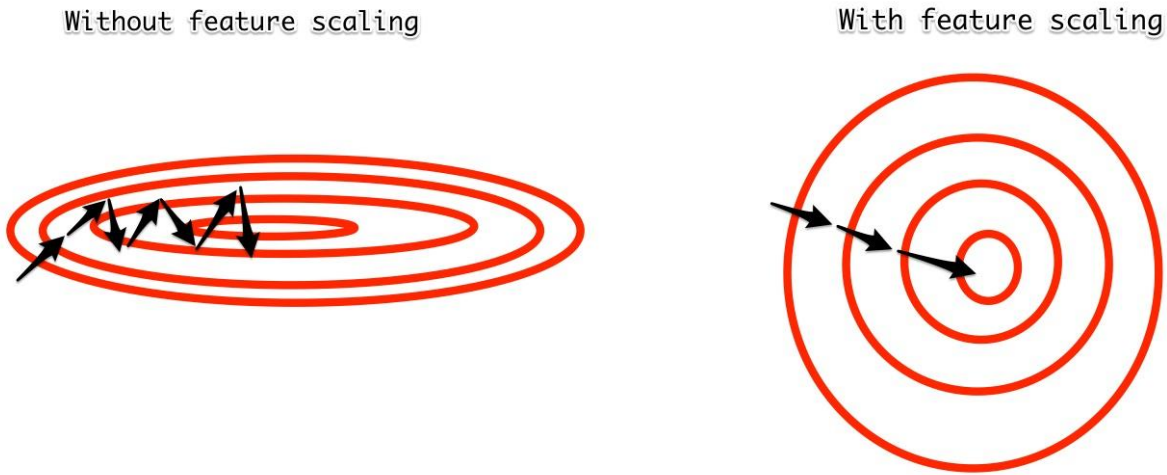


Image sources: Jannes Klaas, kaggle.com

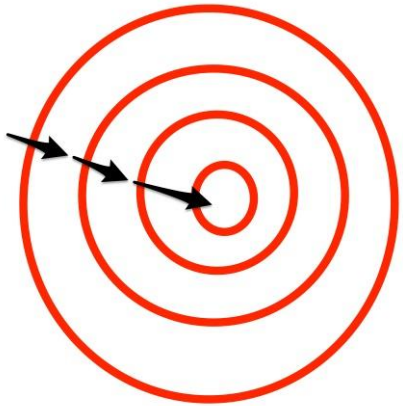
Feature scaling

Loss function for 2 parameters - top view (contour lines)

Feature scaling: Scaling input variables to the same order of magnitude

Without feature scaling

With feature scaling



If the input variables vary greatly in magnitude, the loss function will be extremely elongated. In this case, **convergence will be very slow.**

For this reason, it is advisable to **scale the input variables** to the same order of magnitude.

Feature scaling

Independently for each variable!

Feature scaling: Scaling input variables to the same order of magnitude

- **Min-max scaling** to the [0, 1] interval:

$$\forall i, j : x_i^{(j)} = \frac{x_i^{(j)} - min_j(\{x_i^{(j)}\})}{max_j(\{x_i^{(j)}\}) - min_j(\{x_i^{(j)}\})}$$

- **Standardization** (mean = 0, standard deviation = 1):

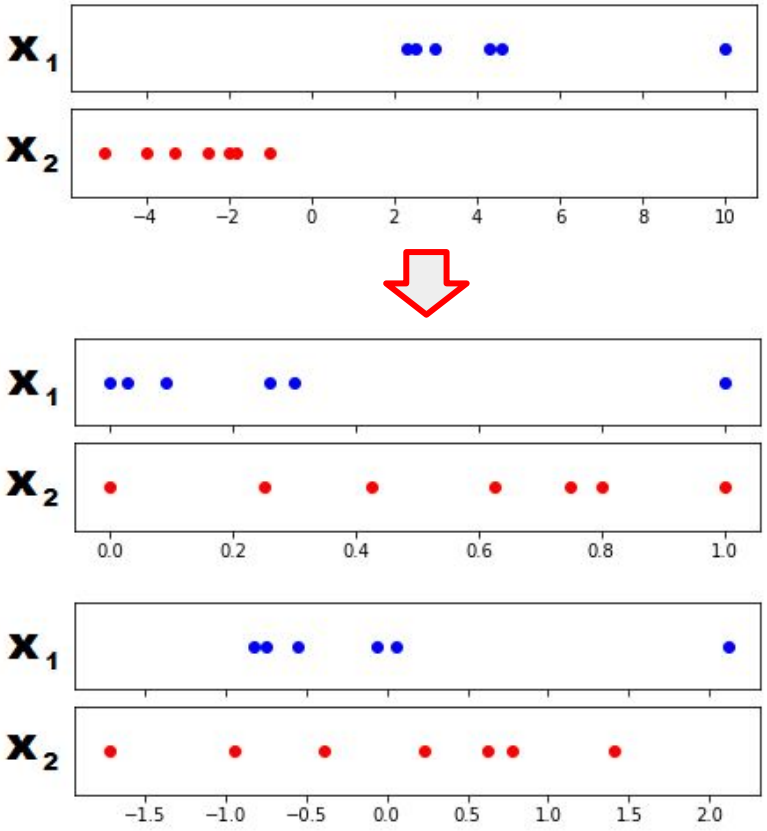
$$\forall i, j : x_i^{(j)} = \frac{x_i^{(j)} - \mu_j(\{x_i^{(j)}\})}{\sigma_j(\{x_i^{(j)}\})}$$

μ: mean
σ: standard deviation

Feature scaling

Independently for each variable!

Feature scaling: Scaling input variables to the same order of magnitude



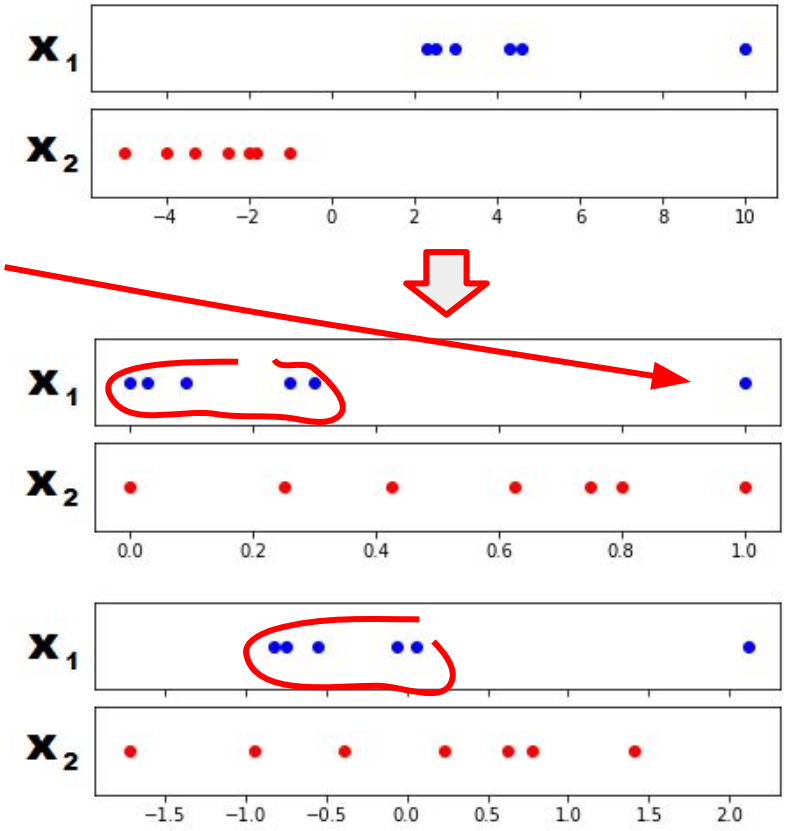
- **Min-max scaling** to the [0, 1] interval:

- **Standardization** (mean = 0, standard dev. = 1):

Feature scaling

Independently for each variable!

Feature scaling: Scaling input variables to the same order of magnitude



Min-max scaling: More sensitive to outliers

- **Min-max scaling** to the [0, 1] interval:

- **Standardization** (mean = 0, standard dev. = 1):

The two main types of tasks in supervised learning

Regression: Continuous labels (The label set is infinite)

$$|Y| = \infty$$ **Example:** Number of cars or the age of a person

Classification: Discrete labels (The label set is finite)

$$|Y| < \infty$$ **Example:** Categorization of examples

- What is the profession of the person in the image?

Classification

Example: Categorize the photos by the animals that appear in them!

What labels should we learn? One possible approach:

For example, let “dog” = 1, “cat” = 2.

Classification

Example: Categorize the photos by the animals that appear in them!

What labels should we learn? One possible approach:

For example, let “dog” = 1, “cat” = 2.

We can use regression. The category label closest to the estimate is the one we estimate.

Classification

Example: Categorize the photos by the animals that appear in them!

What if we have more than two categories?

For example, let “dog” = 1, “cat” = 2, “parrot” = 3.

What’s the problem with that?

Classification

Example: Categorize the photos by the animals that appear in them!

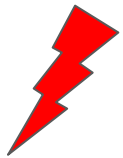
What if we have more than two categories?

For example, let “dog” = 1, “cat” = 2, “parrot” = 3.

What’s the problem with that?

In case of regression, we assume:

½ dog + ½ parrot = 1 cat



We need a different approach…

Classification

Example: Categorize the photos by the animals that appear in them!

Let’s estimate probabilities!

Label: What is the probability that a data point belongs to a given category?

Binary classification

The simplest case: Binary classification (two categories)

$$\hat{y} = h(x) = P(x \text{ is a cat})$$
$$\hat{y} = h(x) = 1 - P(x \text{ is a dog})$$

The estimate is a continuous value. Let **P = 0.5** be the threshold.

Binary classification

The simplest case: Binary classification (two categories)

$$\hat{y} = h(x) = P(x \text{ is a cat})$$
$$\hat{y} = h(x) = 1 - P(x \text{ is a dog})$$

The estimate is a continuous value. Let **P = 0.5** be the threshold.

If our estimate (y hat) is greater than 0.5, we say *"It's a cat."*
If it is less than 0.5, we say *"It's a dog."*

Binary classification

The simplest case: Binary classification (two categories)

$$\hat{y} = h(x) = P(x \text{ is a cat})$$
$$\hat{y} = h(x) = 1 - P(x \text{ is a dog})$$

The estimate is a continuous value. Let **P = 0.5** be the threshold.

What can we do if our hypothesis function estimates a value outside the interval [0,1]?

Binary classification

What can we do if our hypothesis function estimates a value outside the interval [0,1]?

Binary classification

What can we do if our hypothesis function estimates a value outside the interval [0,1]?

Estimates less than 0 can be considered as 0, and estimates greater than 1 can be considered as 1.

Binary classification

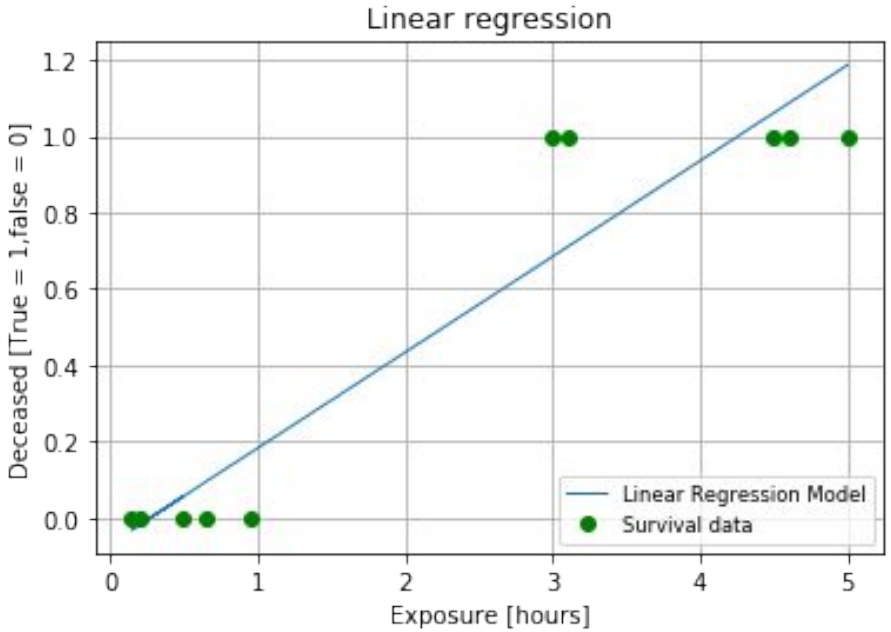
What can we do if our hypothesis function estimates a value outside the interval [0,1]?

Estimates less than 0 can be considered as 0, and estimates greater than 1 can be considered as 1.

Since we are estimating probabilities, **the labels are continuous values** in practice, so we can try **linear regression!**

Classification solved with linear regression - a counterexample

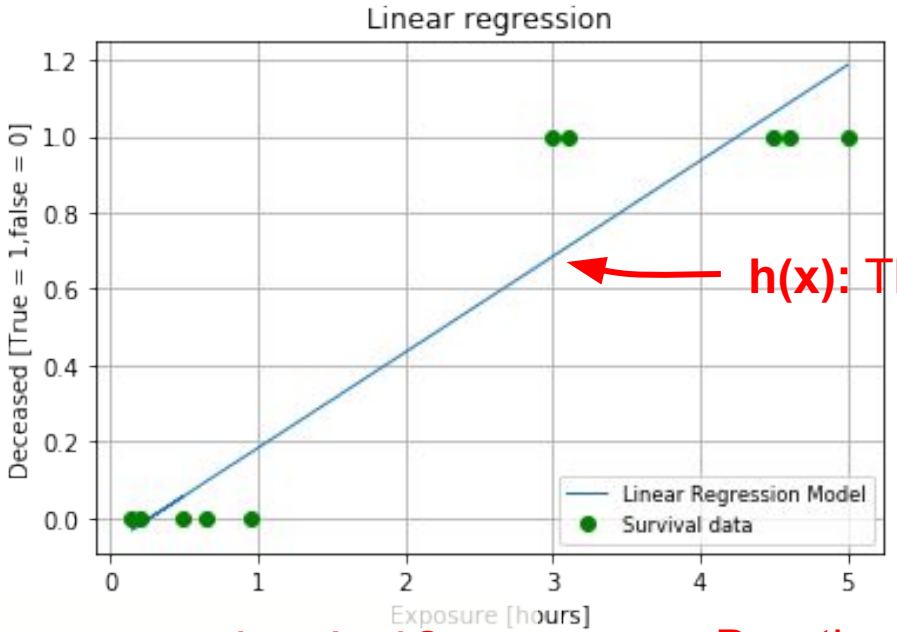
Example: We exposed people to radioactivity for a certain period of time. Will they survive?



Classification solved with linear regression - a counterexample

Example: We exposed people to radioactivity for a certain period of time. Will they survive?

y: Did they die?
(1: yes, 0: no)



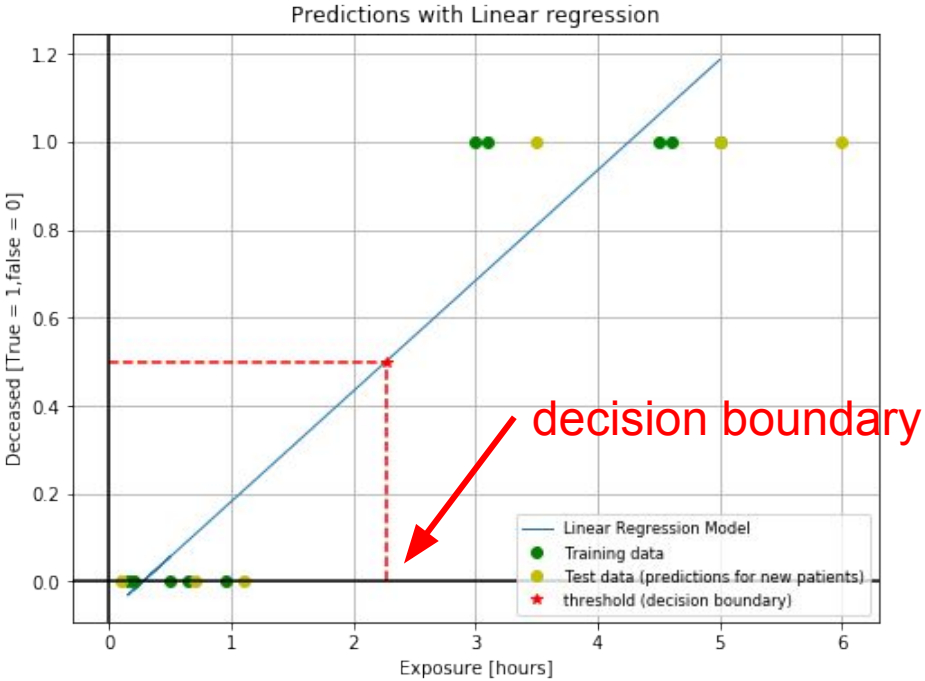
h(x): The fitted hypothesis

How do we estimate a category (yes/no)?

x: Duration of exposure (hours)

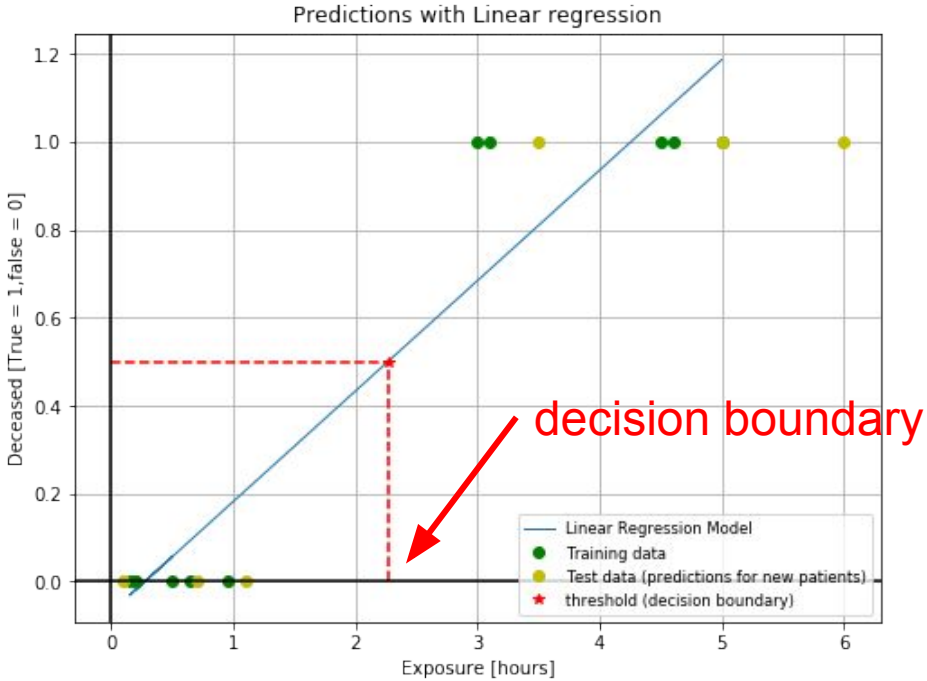
Classification solved with linear regression - a counterexample

Decision boundary: $\{x \mid \hat{y} = h(x) = 0.5\}$



Classification solved with linear regression - a counterexample

Decision boundary: $\{x \mid \hat{y} = h(x) = 0.5\}$



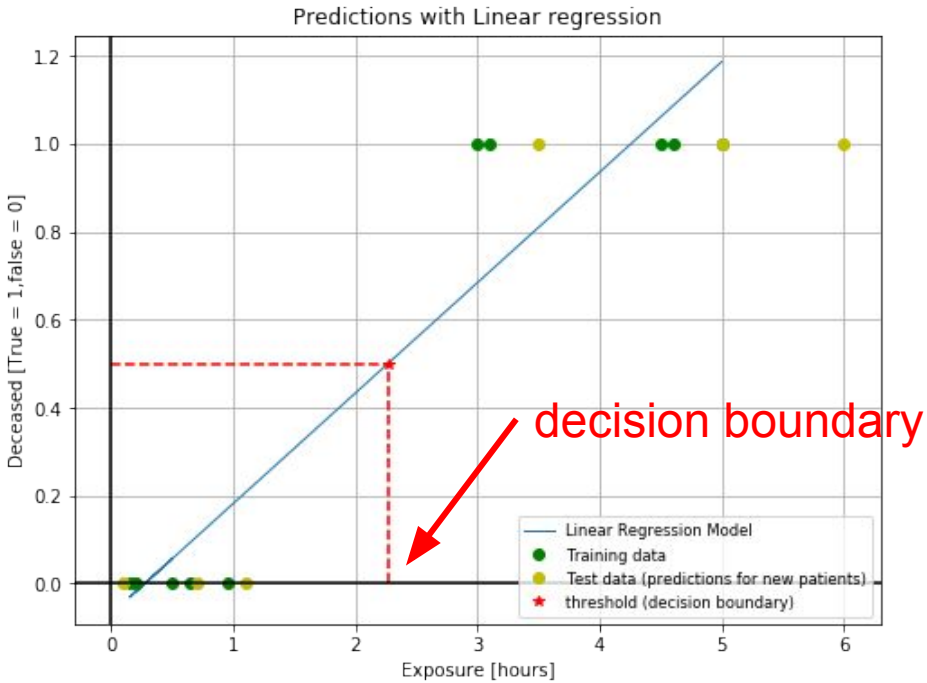
... the set of x inputs where the estimate of the hypothesis function is exactly 0.5
(assuming that h is continuous)
→ typically a (hyper)surface

- **one variable:** a point
- **two variables:** a line or curve in the plane

...

Classification solved with linear regression - a counterexample

Decision boundary: $\{x \mid \hat{y} = h(x) = 0.5\}$

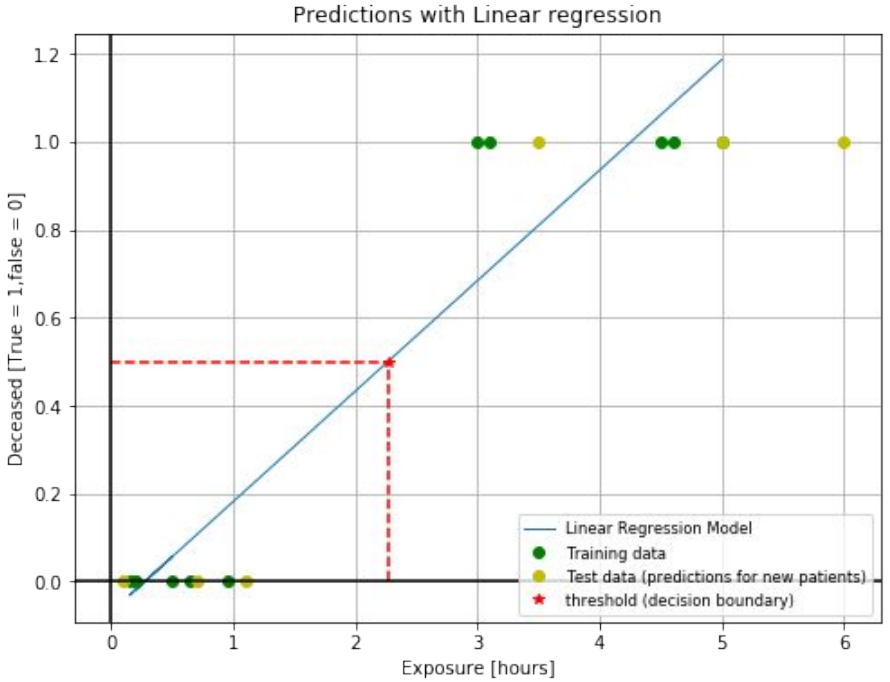


The **decision boundary** learned by linear regression **in this specific example: x ≈ 2.25**

That is, our estimate will be that **the person died if** they were exposed to radiation for **more than 2 and a quarter hours**.

Classification solved with linear regression - a counterexample

Could there be a problem with this approach?

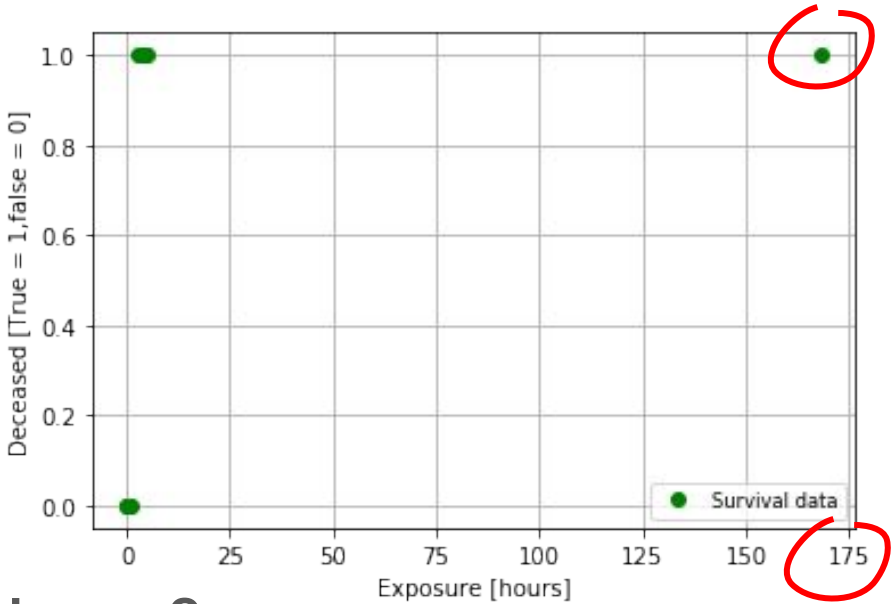


Classification solved with linear regression - a counterexample

Could there be a problem with this approach?

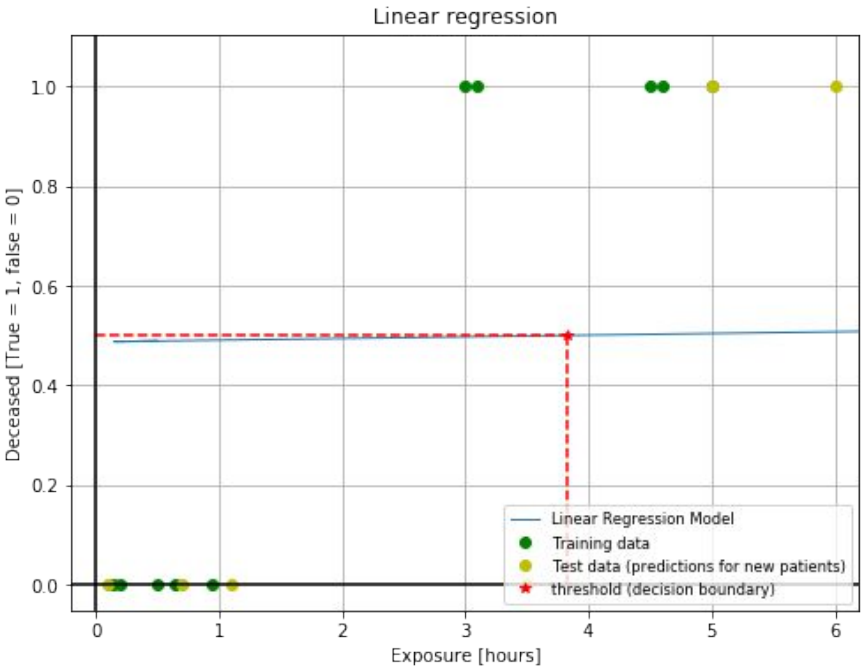
Let's add a single new data point!

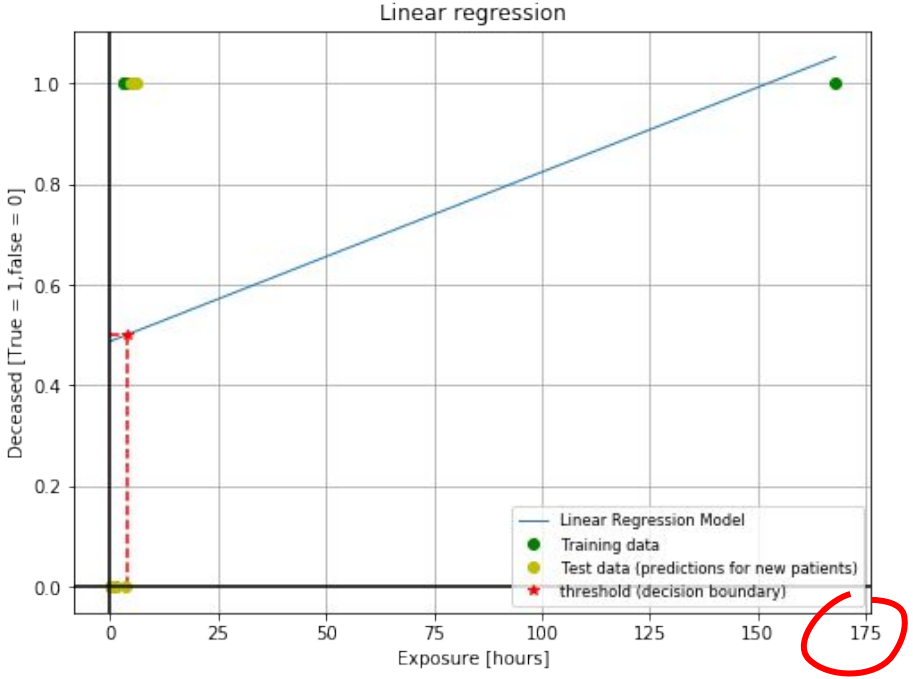
We found a dead person who had been exposed to radiation for a full week.



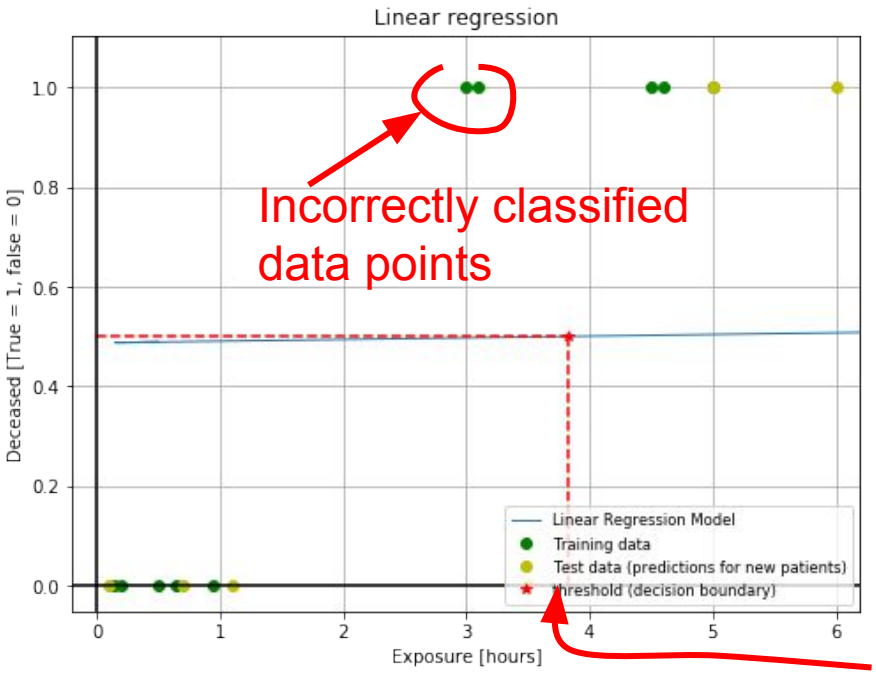
What does linear regression learn this way?

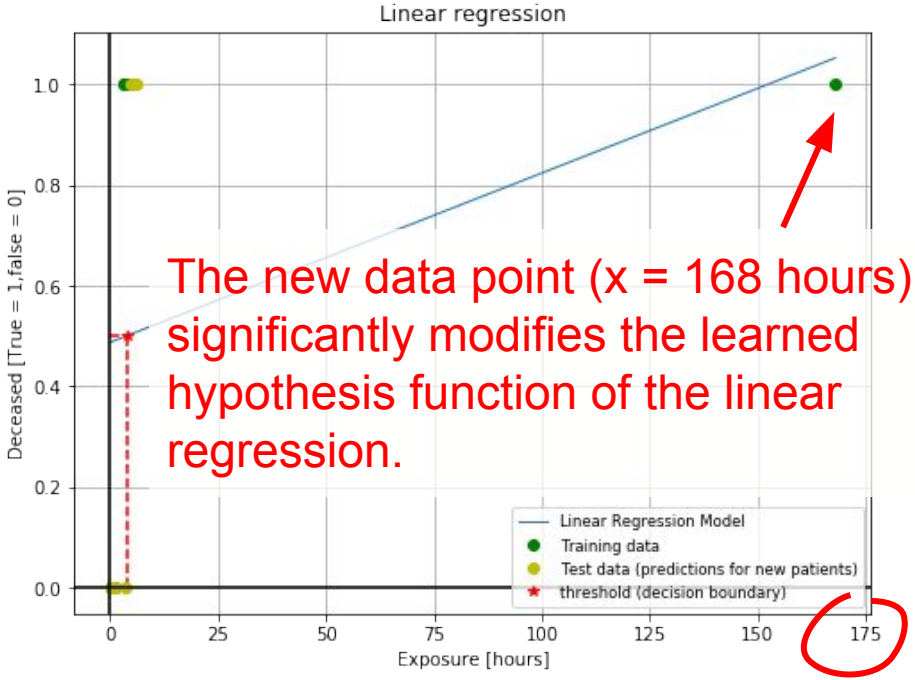
Classification solved with linear regression - a counterexample





Classification solved with linear regression - a counterexample





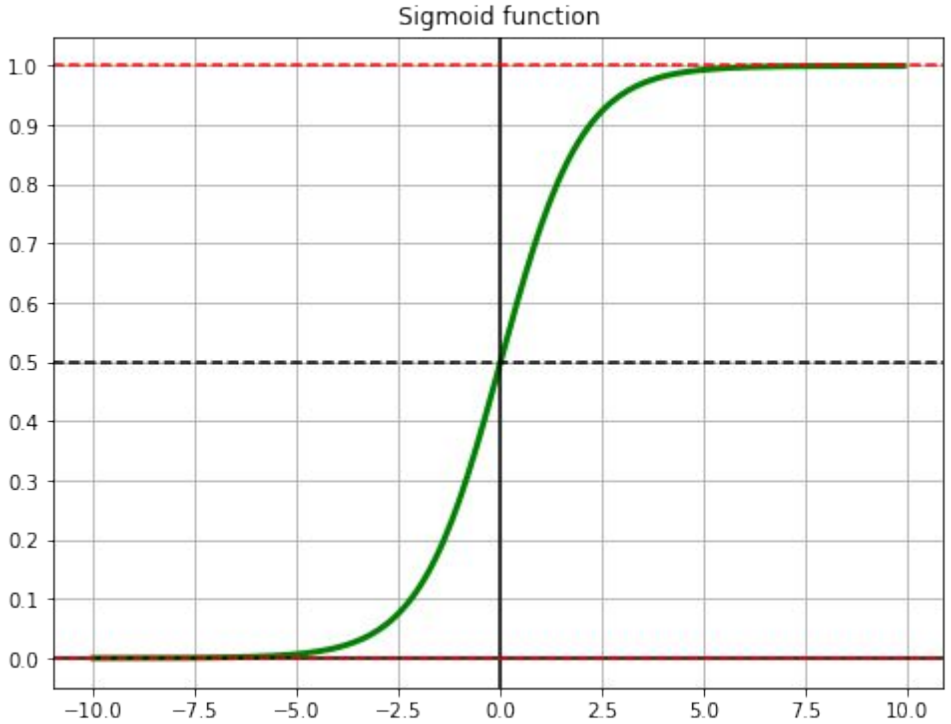
The new data point **significantly influenced the decision boundary**: it now falls at 3 hours and 48 minutes.

Classification solved with linear regression - a counterexample

Linear regression is not ideal for learning classification tasks,
as it is **very sensitive to outliers**.

Instead of a straight line, what could we fit to the data points?

Classification - The sigmoid function



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

Maps between 0 and 1:
Ideal for estimating probabilities!

Classification - hypothesis function

How does this become a hypothesis function?

- Linear regression hypothesis function: $h_\theta(x) = \theta_1 x + \theta_0 = \hat{y}$

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

Classification - hypothesis function

How does this become a hypothesis function?

- Linear regression hypothesis function: $h_\theta(x) = \theta_1 x + \theta_0 = \hat{y}$

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

Logistic regression: $h_\theta(x) = g(\theta_1 x + \theta_0) = \frac{1}{1+e^{-(\theta_1 x + \theta_0)}} = \hat{y}$

Classification - hypothesis function

Hypothesis function of (univariate) **logistic regression:**

$$h_\theta(x) = g(\theta_1 x + \theta_0) = \frac{1}{1+e^{-(\theta_1 x+\theta_0)}} = \hat{y}$$

How do the parameters affect the shape of the curve?

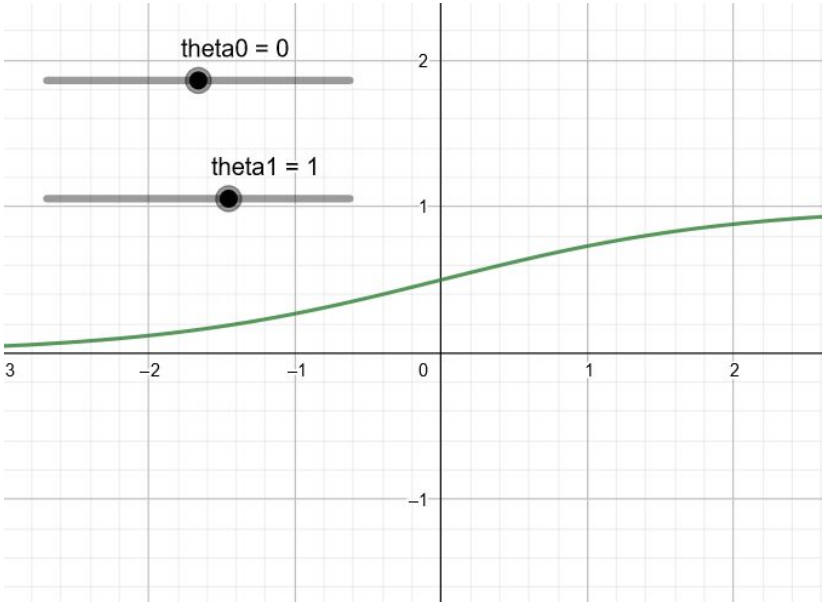
https://www.geogebra.org/graphing/f5x9h7cr

Classification - hypothesis function

$$h_\theta(x) = g(\theta_1 x + \theta_0) = \frac{1}{1+e^{-(\theta_1 x+\theta_0)}} = \hat{y}$$

$$\theta_0 = 0$$

$$\theta_1 = 1$$

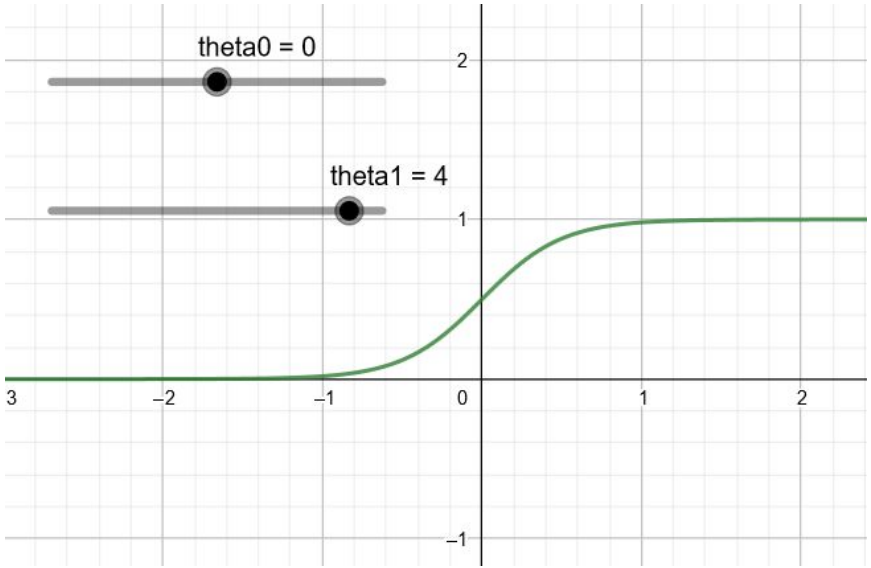


Classification - hypothesis function

$$h_\theta(x) = g(\theta_1 x + \theta_0) = \frac{1}{1+e^{-(\theta_1 x+\theta_0)}} = \hat{y}$$

$$\theta_0 = 0$$

$$\theta_1 = 4$$

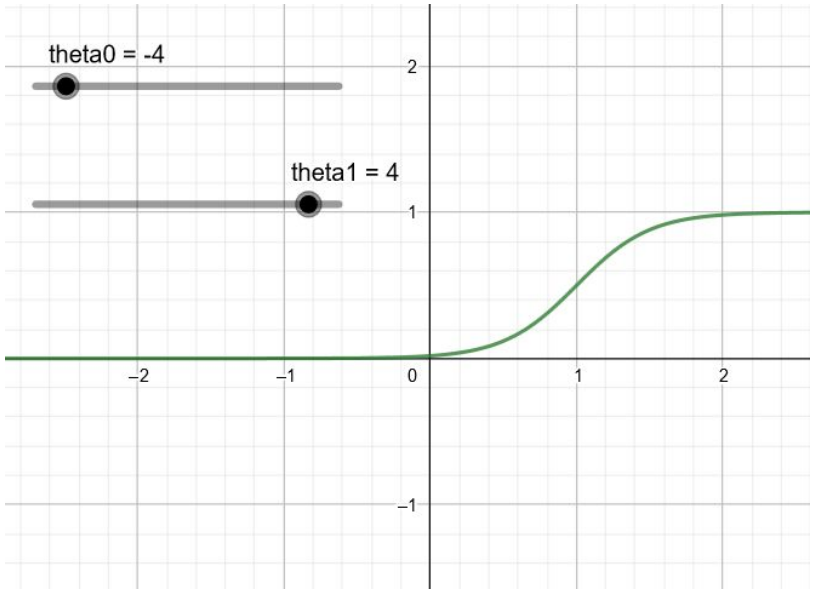


Classification - hypothesis function

$$h_\theta(x) = g(\theta_1 x + \theta_0) = \frac{1}{1+e^{-(\theta_1 x+\theta_0)}} = \hat{y}$$

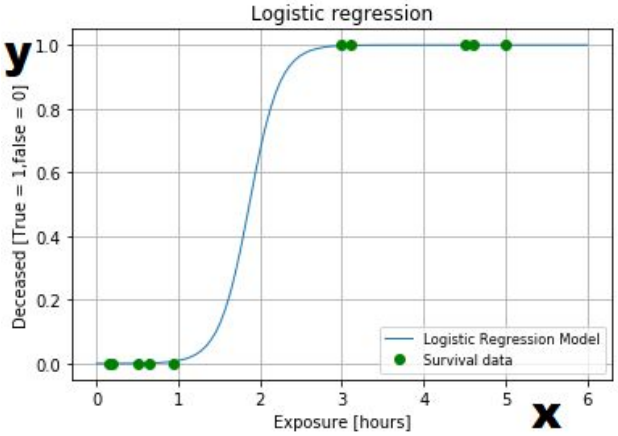
$$\theta_0 = -4$$

$$\theta_1 = 4$$



Logistic regression

1) **Training / fitting:** We fit the hypothesis function to the dataset.



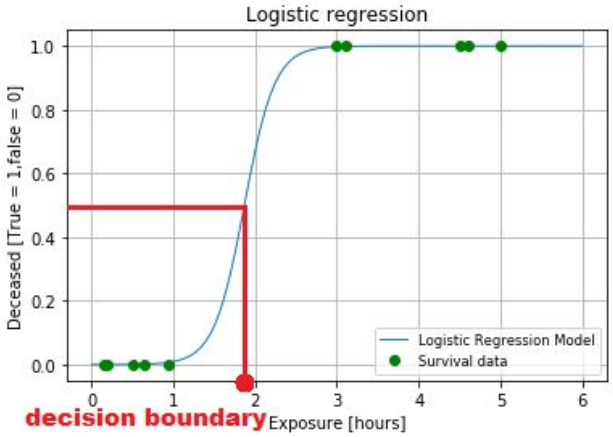
Logistic regression

1) **Training / fitting:** We fit the hypothesis function to the dataset.

Decision boundary: $\{x \mid \hat{y} = h(x) = 0.5\}$

Decision boundary:
The set of **x** input points where **h(x) = 0.5.**

→ In our case, **the decision boundary is a single value on the x axis: ≈ 1.9 hours**

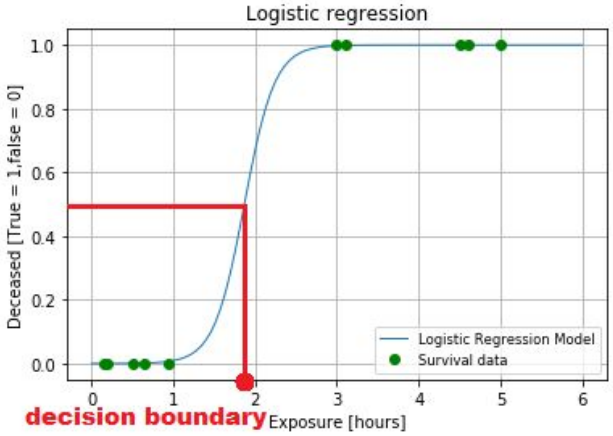


Logistic regression

1) **Training / fitting:** We fit the hypothesis function to the dataset.

   Decision boundary: $\{x \mid \hat{y} = h(x) = 0.5\}$

2) **Estimating labels to new data points:**
   - **If h(x) < 0.5** → We estimate label: “0”.
   - **If h(x) ≥ 0.5** → We estimate label: “1”.



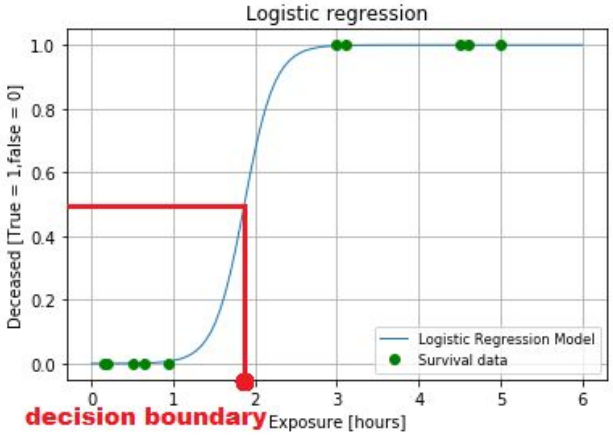
Logistic regression

1) **Training / fitting:** We fit the hypothesis function to the dataset.

Decision boundary: $\{x \mid \hat{y} = h(x) = 0.5\}$

Decision boundary: x ≈ 1.9 hours

2) **Estimating labels to new data points:**
   - **If x < 1.9 hours** → We estimate label: “0”.
   - **If x ≥ 1.9 hours** → We estimate label: “1”.



Logistic regression

“We fit the hypothesis function to the dataset...”

How? What type of loss function should we use?
Is Mean Squared Error (MSE) **loss a good choice?**

$$J(\theta) = \frac{1}{2m} \sum_{j=1}^{m} (h_\theta(x^{(j)}) - y^{(j)})^2$$

where, $h_\theta(x) = g(\theta_1 x + \theta_0) = \frac{1}{1+e^{-(\theta_1 x + \theta_0)}} = \hat{y}$

Logistic regression

$$J(\theta) = \frac{1}{2m} \sum_{j=1}^{m} (h_\theta(x^{(j)}) - y^{(j)})^2$$

MSE loss

where, $$h_\theta(x) = g(\theta_1 x + \theta_0) = \frac{1}{1+e^{-(\theta_1 x + \theta_0)}} = \hat{y}$$

Logistic regression

$$J(\theta) = \frac{1}{2m} \sum_{j=1}^{m} (h_\theta(x^{(j)}) - y^{(j)})^2$$

MSE loss

where, $$h_\theta(x) = g(\theta_1 x + \theta_0) = \frac{1}{1+e^{-(\theta_1 x + \theta_0)}} = \hat{y}$$

Problem: The sigmoid is **not convex**, nor is its square...

→ There may be **multiple local minima**
→ Gradient descent **does not necessarily find the optimal solution**

Logistic regression

Loss function for logistic regression:

$$J(\theta) = \begin{cases} -log(h_\theta(x)) & \text{if } y = 1 \\ -log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

where $$h_\theta(x) = g(\theta_1 x + \theta_0) = \frac{1}{1+e^{-(\theta_1 x+\theta_0)}} = \hat{y}$$

Logistic regression

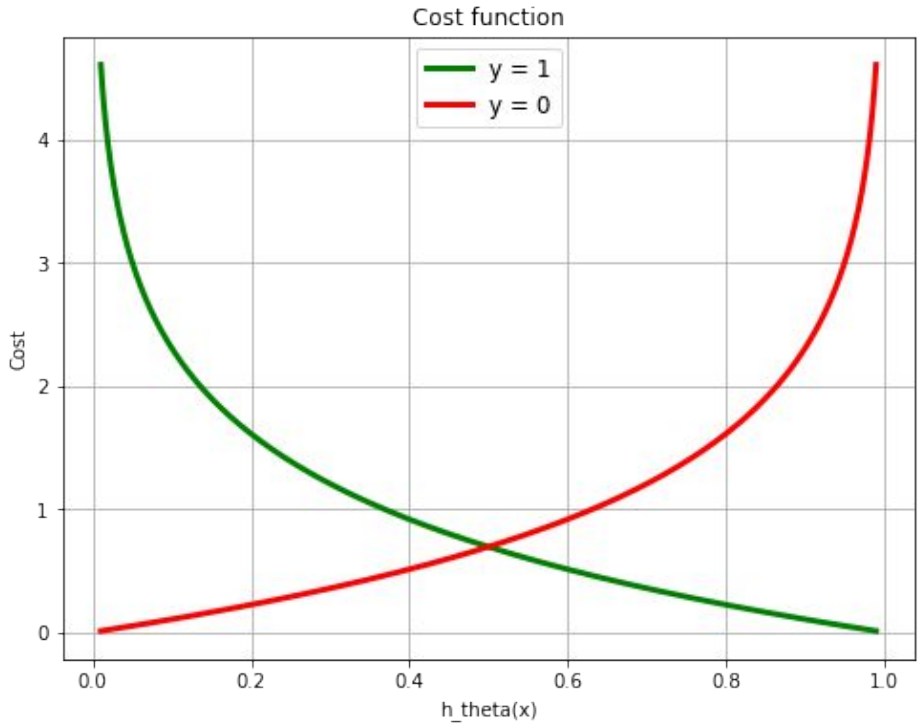
Loss function for logistic regression:

$$J(\theta) = \begin{cases} -log(h_\theta(x)) & \text{if } y = 1 \\ -log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

It can be derived that:

The loss function is convex.

(The second derivative of the loss function is always positive)

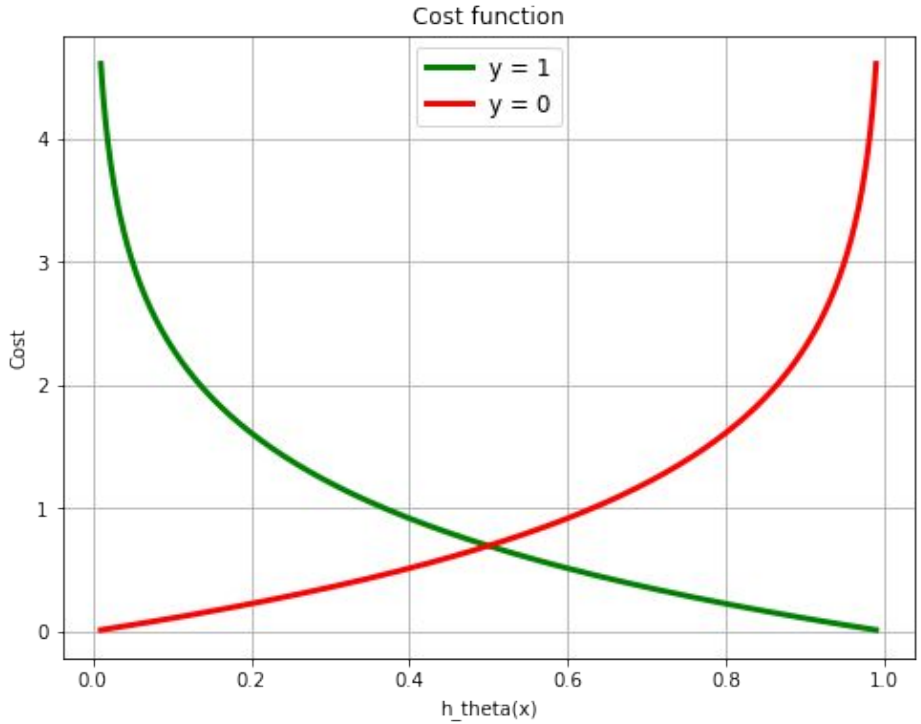


Logistic regression

Loss function for logistic regression:

$$J(\theta) = \begin{cases} -log(h_\theta(x)) & \text{if } y = 1 \\ -log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

Combining the two cases into a single equation?



Logistic regression

Loss function, combined form (logistic loss, binary crossentropy)**:**

$$J(\theta) = \begin{cases} -log(h_\theta(x)) & \text{if } y = 1 \\ -log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

$$J(\theta) = \frac{1}{m} \sum_{j=1}^{m} [-y^{(j)}\ log(h_\theta(x^{(j)})) - (1 - y^{(j)})\ log(1 - h_\theta(x^{(j)}))]$$

= 0, if the true label (y) is 0

= 0, if the true label (y) is 1

Logistic regression

We need derivatives for gradient descent…

$$J(\theta) = \frac{1}{m}\sum_{j=1}^{m}[-y^{(j)}\ log(h_\theta(x^{(j)})) - (1-y^{(j)})\ log(1-h_\theta(x^{(j)}))]$$

$$h_\theta(x) = \frac{1}{1+e^{-(\theta_1 x+\theta_0)}}$$

$$\frac{\partial}{\partial\theta_0}J(\theta) = \frac{1}{m}\sum_{j=1}^{m}(h_\theta(x^{(j)}) - y^{(j)})$$

$$\frac{\partial}{\partial\theta_1}J(\theta) = \frac{1}{m}\sum_{j=1}^{m}(h_\theta(x^{(j)}) - y^{(j)})x^{(j)}$$

Logistic regression

Gradient descent for logistic regression:

repeat until convergence {

$$grad_0 := \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$

$$grad_1 := \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$

$$\theta_0 := \theta_0 - \alpha \cdot grad_0$$

$$\theta_1 := \theta_1 - \alpha \cdot grad_1$$

}

$$\frac{\partial}{\partial \theta_0} J(\theta) = \frac{1}{m} \sum_{j=1}^{m} (h_\theta(x^{(j)}) - y^{(j)})$$

$$\frac{\partial}{\partial \theta_1} J(\theta) = \frac{1}{m} \sum_{j=1}^{m} (h_\theta(x^{(j)}) - y^{(j)}) x^{(j)}$$

$$h_\theta(x) = \frac{1}{1+e^{-(\theta_1 x + \theta_0)}}$$

Logistic regression

Gradient descent for logistic regression:

repeat until convergence {

$grad_0 := \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$

$grad_1 := \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$

$\theta_0 := \theta_0 - \alpha \cdot grad_0$

$\theta_1 := \theta_1 - \alpha \cdot grad_1$

}

$$\frac{\partial}{\partial \theta_0} J(\theta) = \frac{1}{m} \sum_{j=1}^{m} (h_\theta(x^{(j)}) - y^{(j)})$$

$$\frac{\partial}{\partial \theta_1} J(\theta) = \frac{1}{m} \sum_{j=1}^{m} (h_\theta(x^{(j)}) - y^{(j)}) x^{(j)}$$

Remember: The hypothesis function is not linear here!

$$h_\theta(x) = \frac{1}{1+e^{-(\theta_1 x + \theta_0)}}$$

Univariate binary classification

Example tasks with a single input variable:

x: Duration of exposure to radiation → **y:** Did the person die?

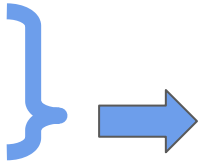
x: Population of a settlement → **y:** Is the settlement a town?

x: Weight of an animal → **y:** Is it a dog or a cat?

Multivariate binary classification

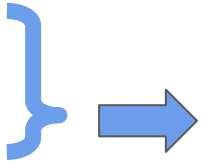
Example tasks with multiple input variables (features)**:**

x_1: Duration of exposure to radiation
x_2: Distance to radiation source



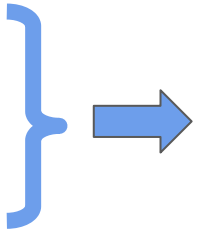
y: Did the person die?

x_1: Population of a settlement
x_2: Annual number of tourists



y: Is the settlement a town?

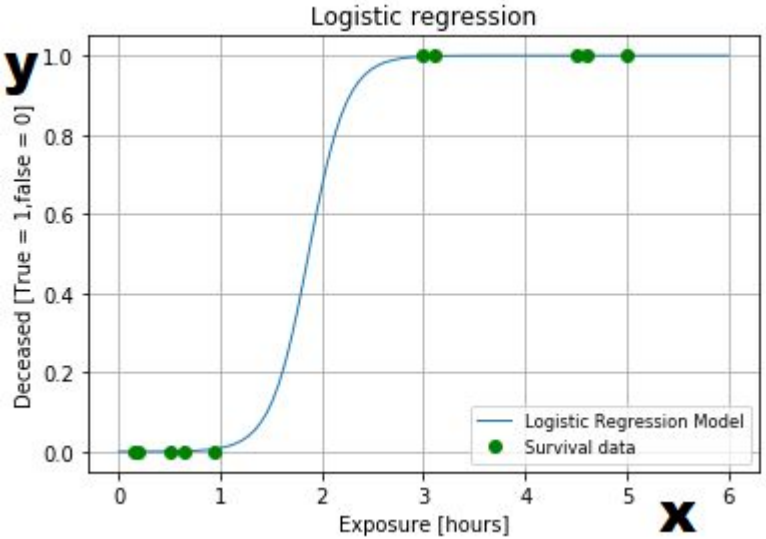
x_1: Weight of an animal
x_2: Number of hairballs coughed up
x_3: Degree of drooling



y: Is it a dog or a cat?

Multivariate logistic regression

Hypothesis function in case of two input variables:

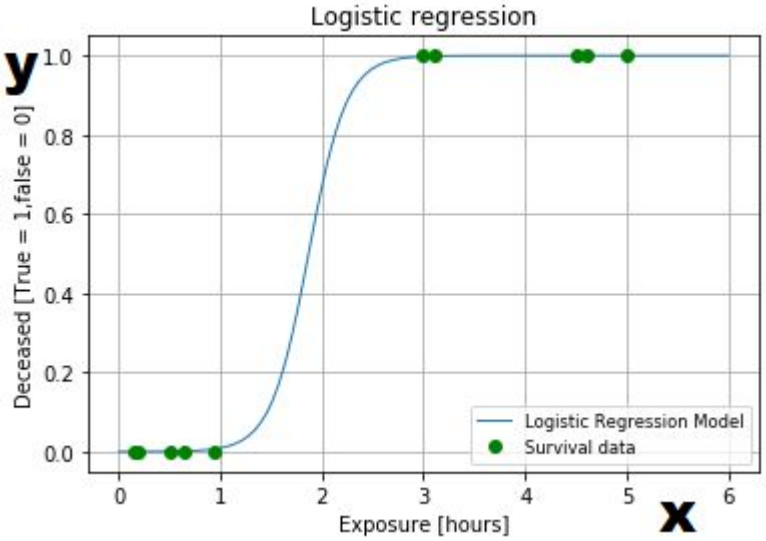


instead

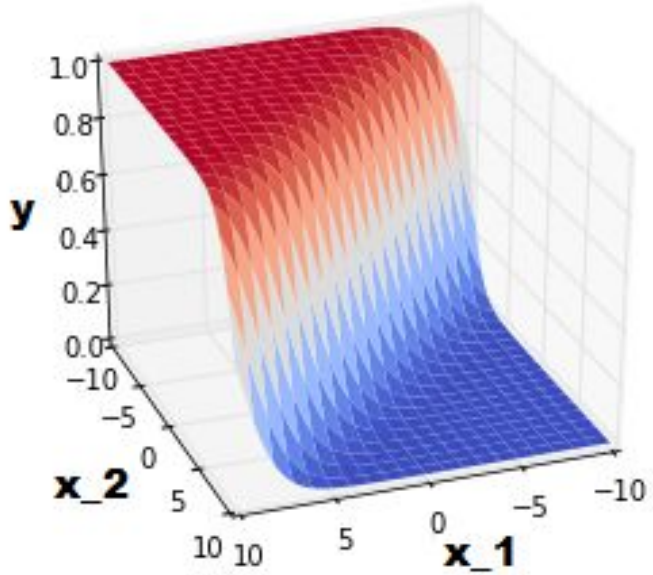
???

Multivariate logistic regression

Hypothesis function in case of two input variables:

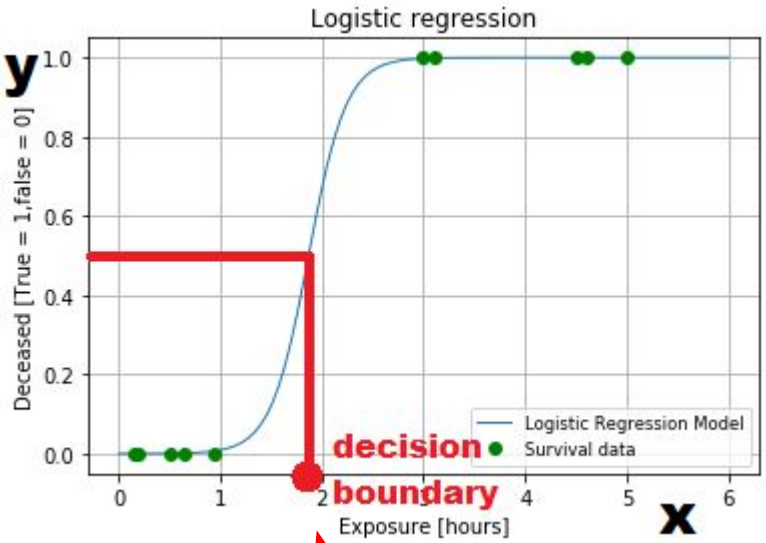


instead

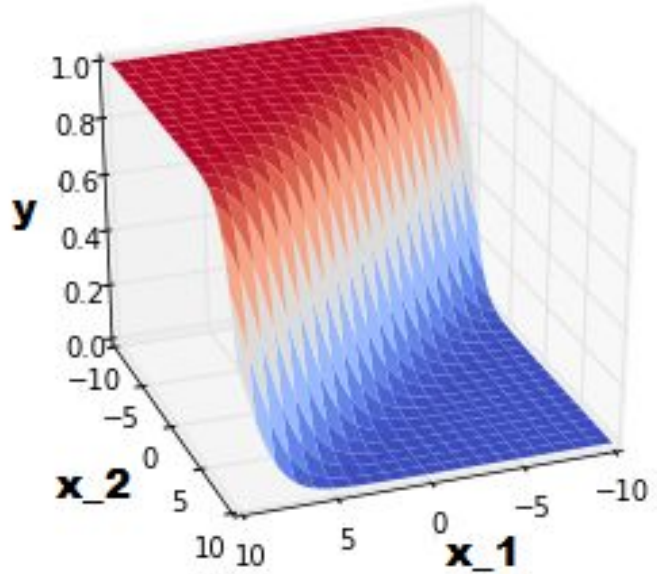


Multivariate logistic regression

Decision boundary in case of two input variables:



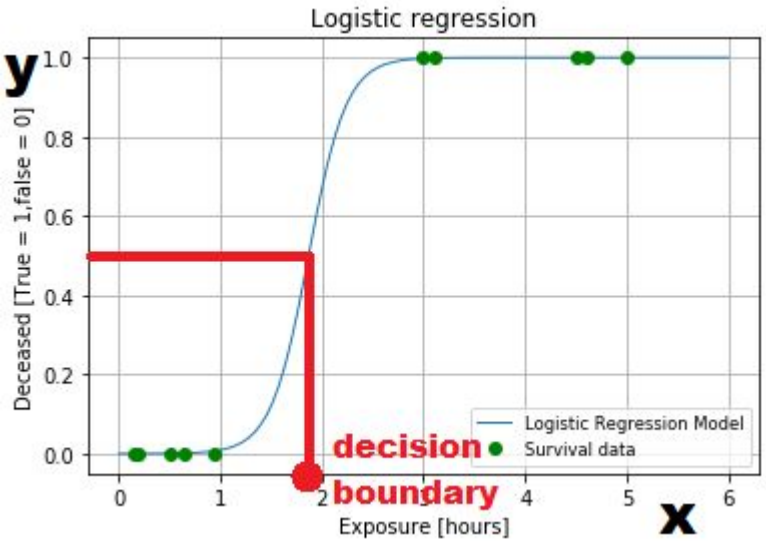
instead



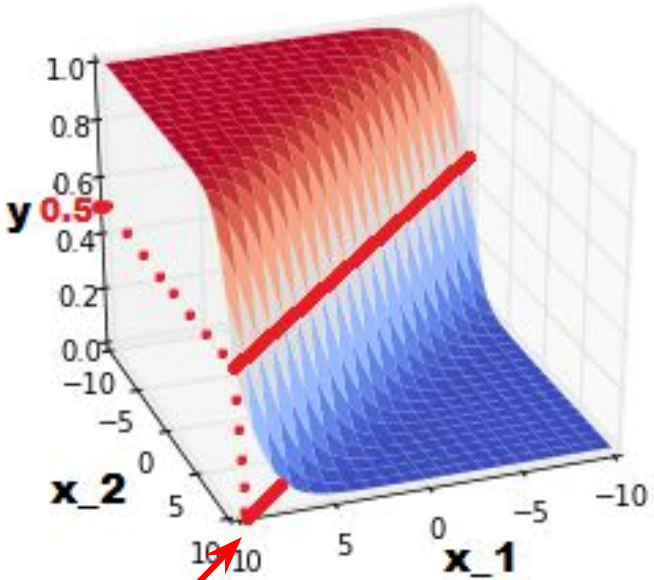
univariate decision boundary (point on x axis)

Multivariate logistic regression

Decision boundary in case of two input variables:



instead

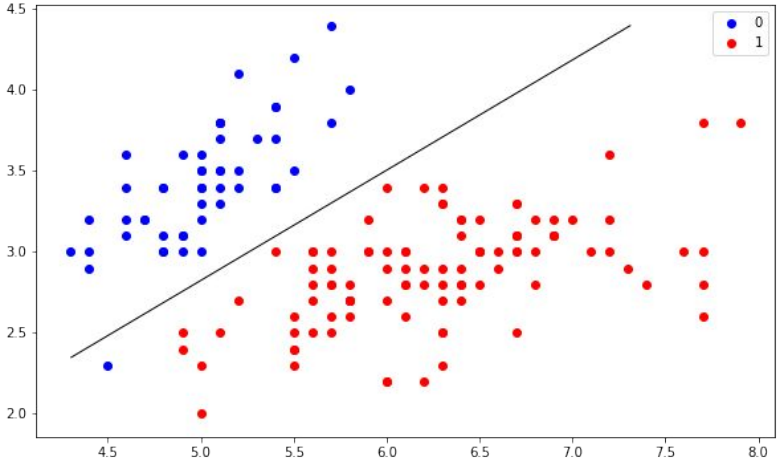


bivariate decision boundary (line in the input plane)

Multivariate logistic regression

Decision boundary in case of two input variables:

instead



The same from the top view...

Multivariate logistic regression

$$\langle x^{(j)}, \theta \rangle = \sum_{i=0}^{n} x_i^{(j)} \theta_i$$

dot product

$$\theta = \begin{bmatrix} \theta_0 \\ \dots \\ \theta_n \end{bmatrix} \in \mathbb{R}^n$$

$$x = \begin{bmatrix} 1 & x_1^{(1)} & \dots & x_n^{(1)} \\ \dots & \dots & \dots & \dots \\ 1 & x_1^{(m)} & \dots & x_n^{(m)} \end{bmatrix} \quad \begin{matrix} \langle x^{(1)}, \theta \rangle \\ \dots \end{matrix} \quad \begin{matrix} g(\langle x^{(1)}, \theta \rangle) = \hat{y}^{(1)} \approx \begin{bmatrix} y^{(1)} \\ \dots \\ y^{(m)} \end{bmatrix} \\ \dots \end{matrix}$$

$$h(x) = g(X\theta) = \hat{y} \approx y \qquad g(z) = \frac{1}{1+e^{-z}}$$

Multivariate linear / logistic regression

Gradient descent - multivariate case

repeat until convergence {

for i $\leftarrow$ 1 . . . n {

$grad_i = \frac{\partial}{\partial \theta_i} J(\theta)$

}

for i $\leftarrow$ 1 . . . n {

$\theta_i = \theta_i - \alpha \, grad_i$

}

}

Linear regression:

$$\frac{\partial}{\partial \theta_i} J(\theta) = \frac{1}{m} \sum_{j=1}^{m} (\theta_n x_n^{(j)} + \cdots + \theta_0 x_0^{(j)} - y^{(j)}) x_i^{(j)}$$

Logistic regression:

$$\frac{\partial}{\partial \theta_i} J(\theta) = \frac{1}{m} \sum_{j=1}^{m} \left[\frac{1}{1 + e^{-(\theta_n x_n^{(j)} + \cdots + \theta_0 x_0^{(j)})}} - y^{(j)} \right] x_i^{(j)}$$

The slope of the tangent in the θ_i direction

Multivariate linear / logistic regression

Gradient descent - multivariate case

repeat until convergence {

  for i $\leftarrow 1 \ldots n$ {

    $grad_i = \frac{\partial}{\partial \theta_i} J(\theta)$

  }

  for i $\leftarrow 1 \ldots n$ {

    $\theta_i = \theta_i - \alpha \, grad_i$

  }

}

1) Calculate the gradient vector:
Its elements are the partial derivatives of the loss function with respect to each parameter θ_i.
The gradient vector indicates the direction in the current parameters in which the loss function grows the most.

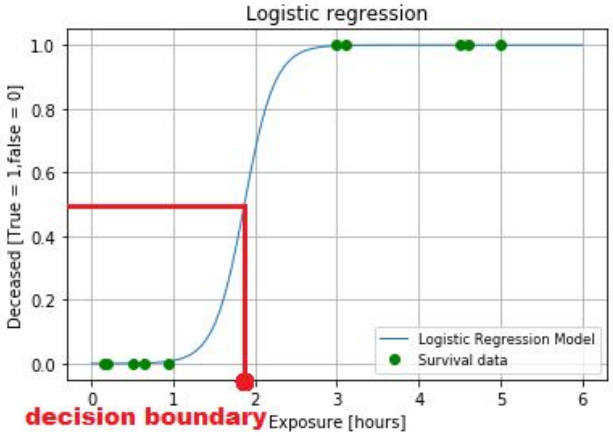
2) Update the parameters:
We subtract the gradient vector multiplied by the learning rate from the old parameters.
(We take one step in the direction of the steepest decrease of the loss function...)

Logistic regression

What have we achieved?

Logistic regression can be used to perform **binary classification**.

Gradient descent can find the **globally optimal solution** for logistic regression.



Logistic regression

What have we achieved?

Logistic regression can be used to perform **binary classification**.

Gradient descent can find the **globally optimal solution** for logistic regression.

Don't get confused: Logistic regression estimates the probability of belonging to a category, which is a continuous value.
However, the estimate is constrained between 0 and 1, so **it can't be used to solve arbitrary regression problems**. We'll use it to estimate probabilities, i.e., to solve (binary) classification problems.