განვიხილოთ უფრო კონკრეტული ამოცანა და ამ ამოცანას ჩავუღრმავდეთ ბოლომდე. ავიღოთ ბინარული კლასიფიკაციის მაგალითი. ვთქვათ მოდელის მიზანია დაადგინოს სურათზე არის თუ არა კატა. სურათი, როგორც მოგეხსენებათ არის არასტრუქტურიზებული ინფორმაცია, რომელიც შედგება 3 შრიანი მატრიცისაგან. მაგალითისათვის თუ ავიღებთ 64*64 ფოტოს გვექნება RGB მატრიცა 12288 პარამეტრით. ე.ი გამოდის რომ შესავალი მოდელში არის 12288 პარამეტრი.
ავიღოთ რაიმე ალგორითმი/მოდელი რომელსაც უნდა ვასწავლოთ გარჩევა ჩვენ მიერ დასმული შეკითხვის, არის თუ არა კატა სურათზე. ავიღოთ ლოჯისტიკური რეგრესია. ლოჯისტიკური რეგრესია არის ალგორითმი რომელსაც გამოიყენებენ supervised დასწავლისას როდესაც გამოსავალზე შედეგი იღებს მნიშვნელობებს 0 ს ან 1 ს. მიზანი ამ ალგორითმისა არის, რომ შეამციროს ერორი მის მიერ ნაწინასწარმეტყველებ აუთფუთსა და რეალურს შორის

• The input features vector: $x \in \mathbb{R}_{nx}$

, where n_x is the number of features

• The training label: $y \in 0,1$

• The weights: $w \in \mathbb{R}_{nx}$, where n_x is the number of features

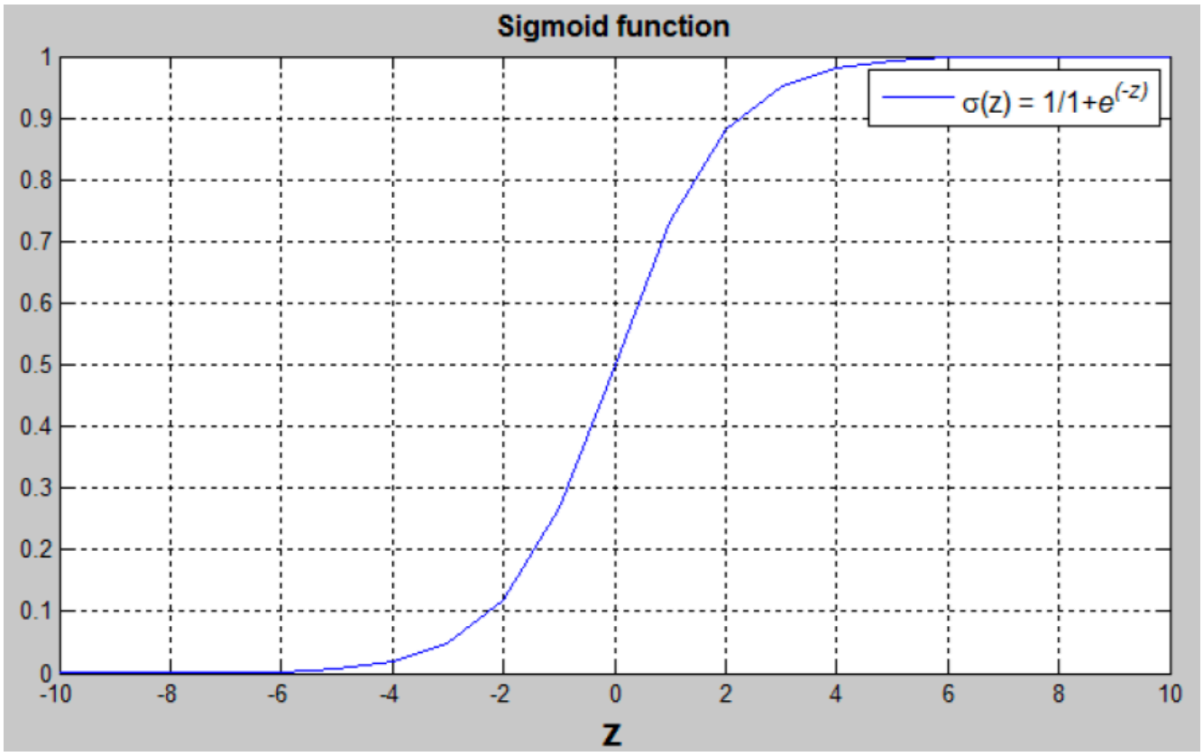
• The threshold: $b \in \mathbb{R}$

• The output: y

ˆ

$= \sigma(w_T x + b)$

• Sigmoid function: s = $\sigma(w_T x + b) = \sigma(z) = \frac{1}{1+e^{-z}}$



$(w_T x + b)$ is a linear function $(ax + b)$, but since we are looking for a probability constraint between [0,1], the sigmoid function is used. The function is bounded between [0,1] as shown in the graph above.
Some observations from the graph:

• If z is a large positive number, then $\sigma(z) = 1$
• If z is small or large negative number, then $\sigma(z) = 0$
• If $z = 0$, then $\sigma(z) = 0.5$

ახლა ვნახოთ როგორ ხდება შეცდომის დაინდენტიფიცირება:

Logistic Regression: Cost Function

To train the parameters w and b, we need to define a cost function.

Recap:

$\hat{y}^{(i)} = \sigma(w^T x^{(i)} + b)$, where $\sigma(z^{(i)}) = \frac{1}{1+e^{-z^{(i)}}}$

$x^{(i)}$ the i-th training example

$Given \{(x^{(1)}, y^{(1)}), \cdots, (x^{(m)}, y^{(m)})\},\ we\ want\ \hat{y}^{(i)} \approx y^{(i)}$

Loss (error) function:

The loss function measures the discrepancy between the prediction ($\hat{y}^{(i)}$) and the desired output ($y^{(i)}$). In other words, the loss function computes the error for a single training example.

$$L(\hat{y}^{(i)}, y^{(i)}) = \frac{1}{2}(\hat{y}^{(i)} - y^{(i)})^2$$

$$L(\hat{y}^{(i)}, y^{(i)}) = -(y^{(i)} \log(\hat{y}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})$$

- If $y^{(i)} = 1$: $L(\hat{y}^{(i)}, y^{(i)}) = -\log(\hat{y}^{(i)})$ where $\log(\hat{y}^{(i)})$ and $\hat{y}^{(i)}$ should be close to 1
- If $y^{(i)} = 0$: $L(\hat{y}^{(i)}, y^{(i)}) = -\log(1 - \hat{y}^{(i)})$ where $\log(1 - \hat{y}^{(i)})$ and $\hat{y}^{(i)}$ should be close to 0

Cost function

The cost function is the average of the loss function of the entire training set. We are going to find the parameters $w\ and\ b$ that minimize the overall cost function.

$$J(w, b) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)}) = -\frac{1}{m}\sum_{i=1}^{m}[(y^{(i)} \log(\hat{y}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})]$$

ლოს ფუნქცია აირჩევა იმის მიხედვით შემდეგ გრადიენტ-დისენთით გავდივართ მინიმუმზე თუ არა. ლოჯისტიკური რეგრესიის ამოცანის შესრულებისას ეს კარგად ჩანს….

გრადიენტ დისენთი არის ოპტიმიზაციის ალგორითმი რომელიც გამოიყენება ლოს ფუნქციის მინიმიზაციისთვის. იგი აკეთებს პარამეტრების ცვლილებას იტერაციულად ისე რომ ლოსი მცირდება. მიზანი არის ვიპოვოთ მოდელის ისეთი w,b პარამეტრები რომ ლოსი იყოს მინიმალური. ინიციალიზაცია ძირითადად ხდება რენდომად ინიცირებული პარამეტრებით. გრადიენტი შინაარსობრივად გამოხატავს ლოს ფუნქციის ცვლილებას იმ შემთხვევასი თუ პარამეტრი მცირედით შეიცვალა.

1. **Objective**: The goal of gradient descent is to find the set of model parameters (weights and biases) that minimize a given loss function. This loss function quantifies how well the model performs on the training data compared to the actual target values.
2. **Initialization**: Gradient descent starts by initializing the model parameters with random values or pre-trained weights.
3. **Gradient Calculation**: The algorithm calculates the gradient of the loss function with respect to each model parameter. The gradient represents the direction of the steepest ascent of the loss function. It indicates how much the loss would increase or decrease if the parameter values were adjusted slightly.
4. **Update Rule**: Based on the gradient information, gradient descent updates the model parameters in the opposite direction of the gradient to minimize the loss function. The update rule is typically defined as follows:

$$\theta_{t+1} = \theta_t - \alpha \nabla L(\theta_t)$$

where:

- θ_t is the current parameter vector.
- α (learning rate) is a hyperparameter that determines the step size of the parameter updates.
- $\nabla L(\theta_t)$ is the gradient of the loss function with respect to the parameters at the current iteration t.

5. **Iteration**: Steps 3 and 4 are repeated iteratively until a stopping criterion is met, such as reaching a predefined number of iterations or achieving a sufficiently low value of the loss function.
6. **Convergence**: Gradient descent converges to a local minimum of the loss function, where further parameter updates do not significantly reduce the loss. However, the algorithm may get stuck in a suboptimal solution if the loss function is non-convex or if the learning rate is too high.

ამის შემდეგ უნდა დავითვალო წარმოებულები ლოჯისტიკური რეგრესიის ლოსის შემთხვევაში ….
ასევე უნდა ვთქვა თავიდან რა არის აქტივაცია, განვიხილო ეს წარმოებულები აქტივაციით…