

Logistic Regression and Neural Networks

Michał Stypułkowski

Wrocław, October 2019

Problem statement

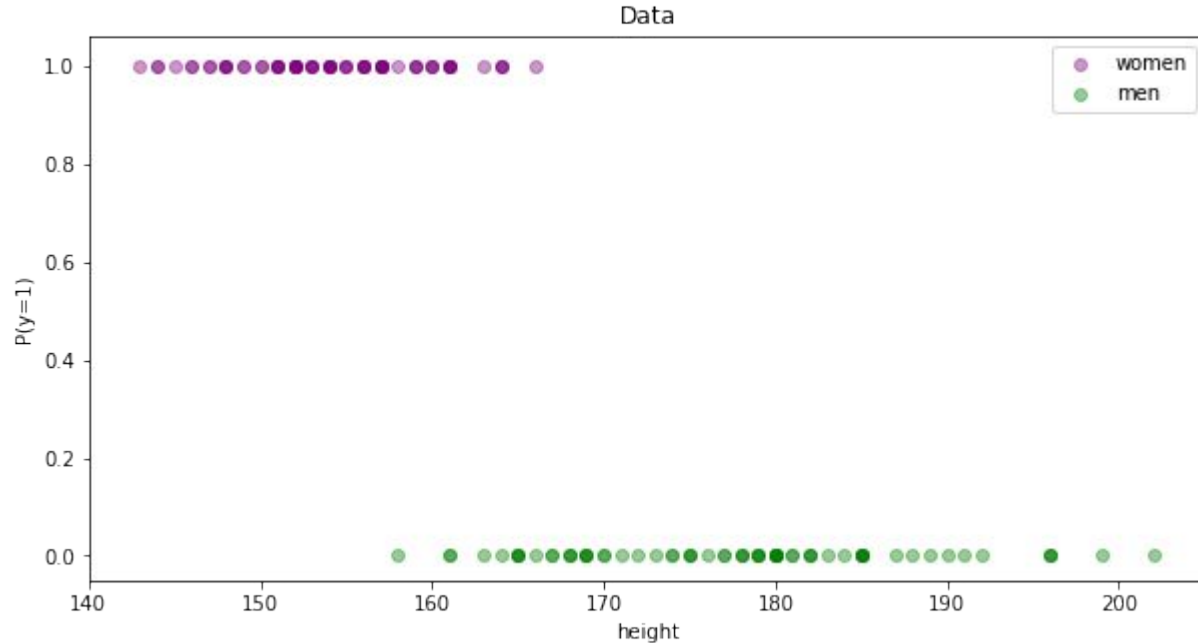
We want to create binary classification model.

- email - spam or ham
- image - cat or not cat
- transaction - fraudulent or not

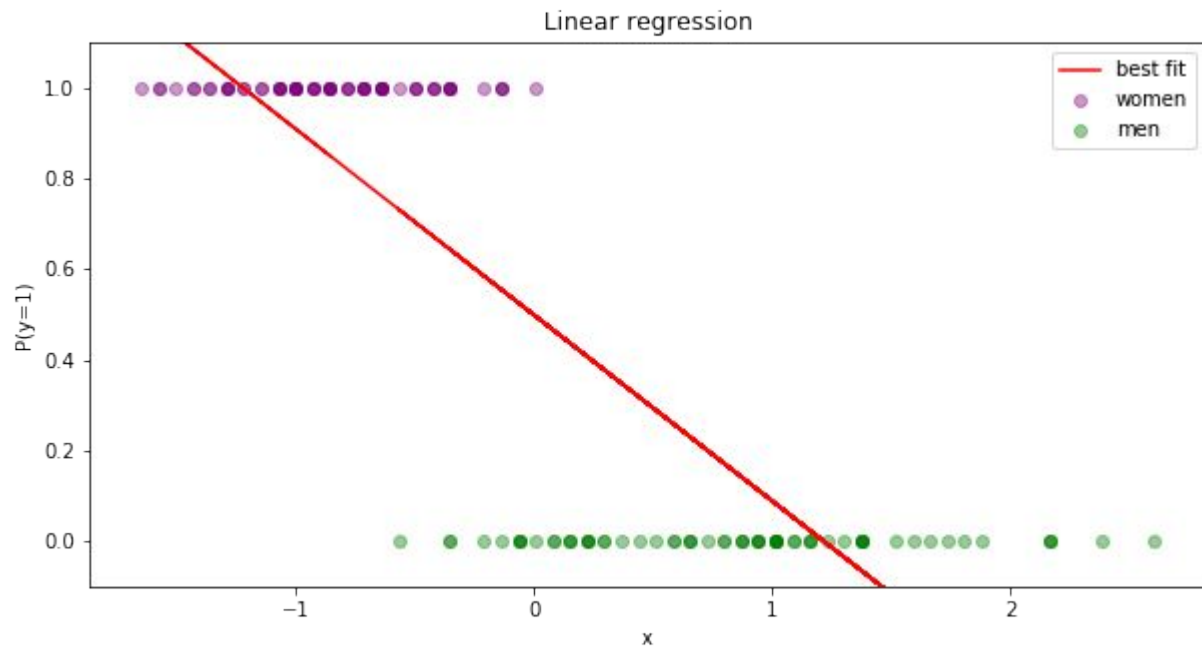
Labels $y \in \{0, 1\}$, where 0 can be interpreted as negative class and 1 as positive class.



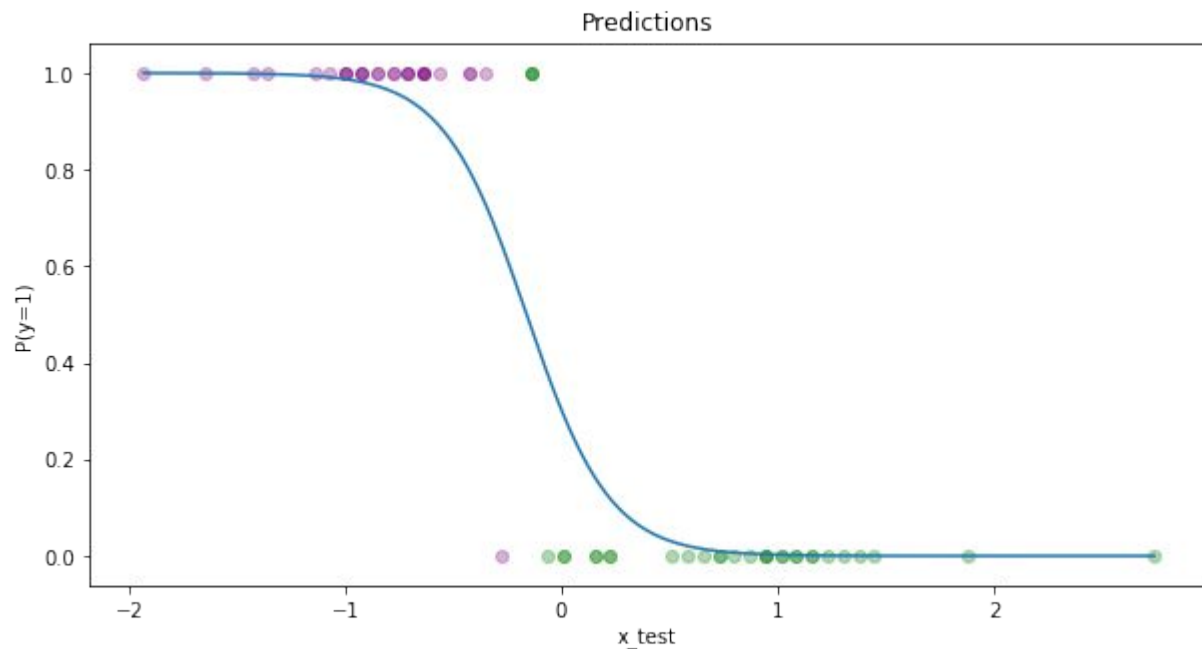
Sex classification based on height



Sex classification based on height



Sex classification based on height



Logistic regression


We want to find function $f : \mathbb{R}^d \rightarrow [0, 1]$ that makes best predictions \hat{y} based on our data points x . For training purpose we use true labels y . We choose it to be sigmoid:

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

Let's think about \hat{y} as probabilities of belonging to class 1:

$$\hat{y} = f(x; \theta) = \mathbb{P}(y = 1 | x; \theta)$$

Note that y can be interpreted as a random variable with Bernoulli distribution with parameter \hat{y} .



Logistic regression

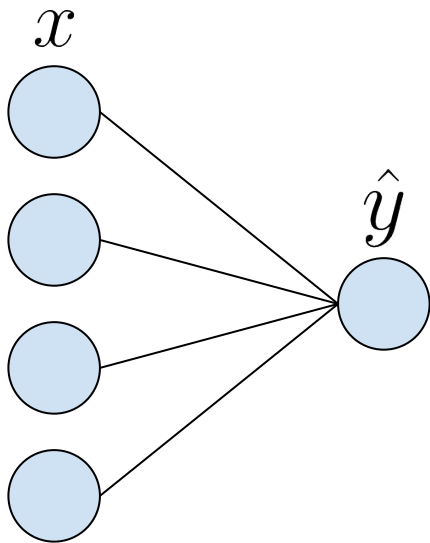
We can use MLE method for optimization!

$$L(\theta) = \prod_{i=1}^m \hat{y}_i^{y_i} (1 - \hat{y}_i)^{1-y_i}$$

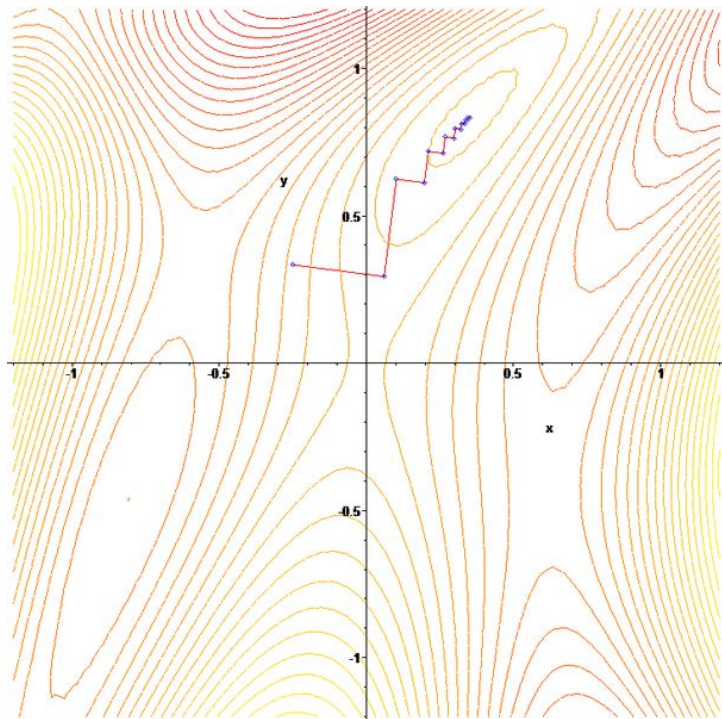
$$l(\theta) = \sum_{i=1}^m y_i \log \hat{y}_i + (1 - y_i) \log (1 - \hat{y}_i)$$

This loss function is called **binary cross entropy**.

Unfortunately there is no analytical solution maximizing it.



Gradient descent



https://en.wikipedia.org/wiki/Gradient_descent

for n_epochs:

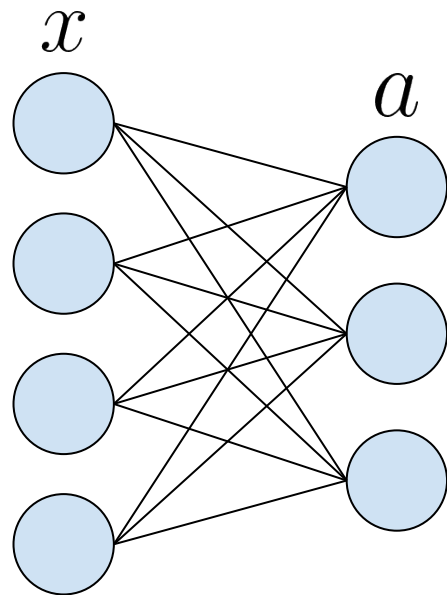
$$\theta = \theta - \alpha \frac{\partial l(\theta)}{\partial \theta}$$

Softmax regression

Now let's suppose we have $c > 2$ classes to consider.

We represent the true label y as an **one-hot vector** oh , e.g. if $c = 5$ and $y = 3$ then one-hot vector representing this label is $(0, 0, 0, 1, 0)$.

For this task, we need to define new activation and loss functions.




Softmax regression

We make prediction based on vector a , where

$$a_i = \frac{\exp(\theta_i^T x)}{\sum_{j=1}^c \exp(\theta_j^T x)}$$

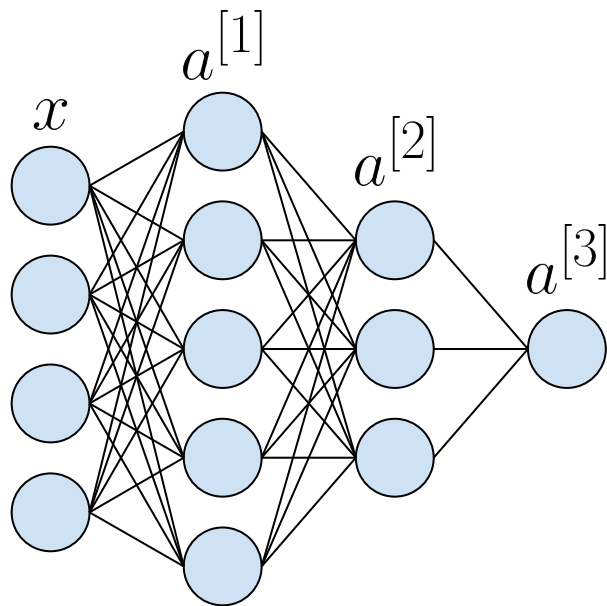
It can be interpreted as probability of belonging to i -th class. Predicted class is $\operatorname{argmax}_i a_i$.

The loss function is now $\sum_{i=1}^m \sum_{j=0}^{c-1} oh_j^{(i)} \log a_j^{(i)}$. It is called **cross entropy**.



Neural networks

Why don't we combine several layers of logistic regression - like operations?



Forward propagation

$$g^{[1]} = W^{[1]T}x + b^{[1]}$$

$$a^{[1]} = f^{[1]}(g^{[1]})$$

$$g^{[2]} = W^{[2]T}a^{[1]} + b^{[2]}$$

$$a^{[2]} = f^{[2]}(g^{[2]})$$

$$g^{[3]} = W^{[3]T}a^{[2]} + b^{[3]}$$

$$a^{[3]} = f^{[3]}(g^{[3]})$$



Backward propagation

We need to compute gradients of the loss function with respect to every parameter W and b .

For example:

$$\frac{\partial L}{\partial W^{[3]}} = \frac{\partial L}{\partial a^{[3]}} \cdot \frac{\partial a^{[3]}}{\partial g^{[3]}} \cdot \frac{\partial g^{[3]}}{\partial W^{[3]}}$$

Gradient descent algorithm is used (or its advanced variations).



Parameters vs hyperparameters

Parameters:

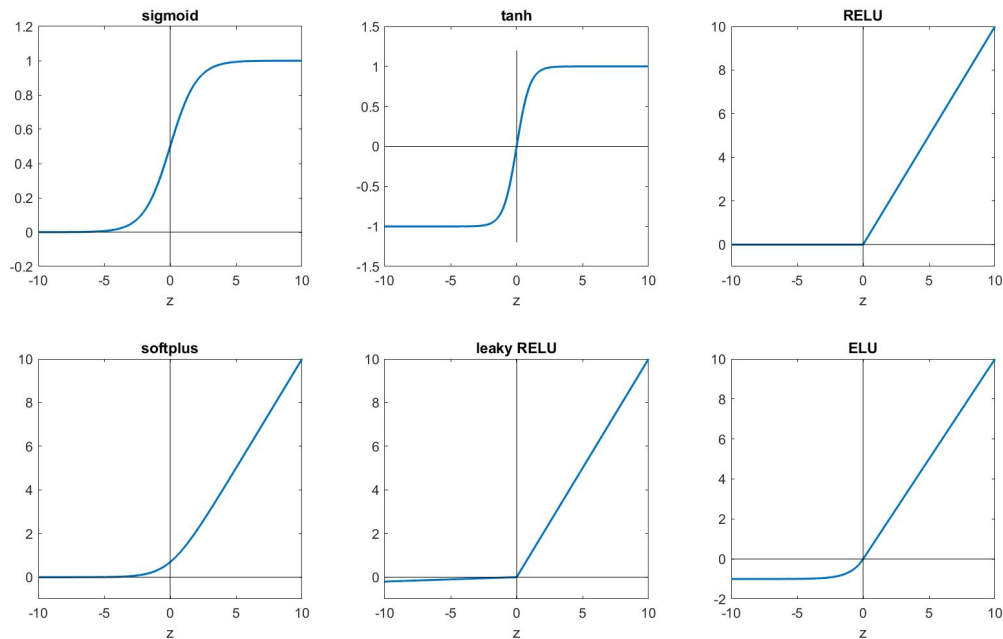
- W, b

Hyperparameters:

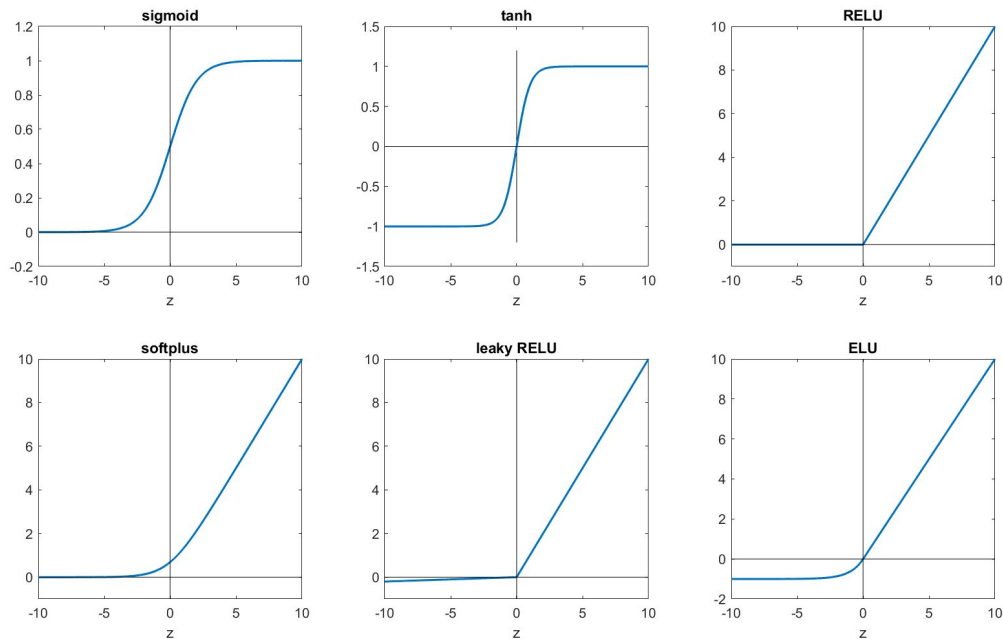
- learning rate
- number of neurons
- number of layers
- activations
- ...



Activation functions



Activation functions



Why do we even need them?

Loss functions

- Mean Square Error / L2 Loss / Reconstruction Loss
- Mean Absolute Error / L1 Loss
- Cross Entropy Loss
- Kullback–Leibler Divergence
- Likelihood



Highly recommended

[Andrew Ng's course](#) - available on Coursera and YouTube.

Amazing introduction to the Deep Learning world. Detailed yet easy to understand.

