ASSIGNMENT 4: LOGISTIC REGRESSION

Johannes Kofler,
Markus Holzleitner
Institute for Machine Learning

JOHANNES KEPLER
UNIVERSITY LINZ

Institute for
Machine Learning

Copyright statement:

Agenda:

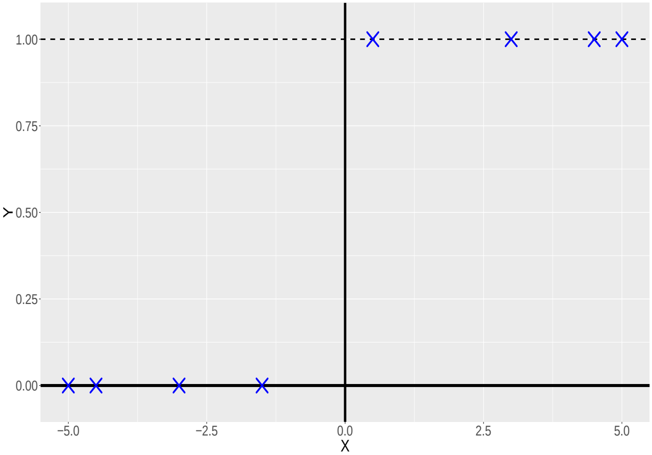
- Introduction
- Cross Entropy
- Softmax
- Gradient Checking

Logistic Regression

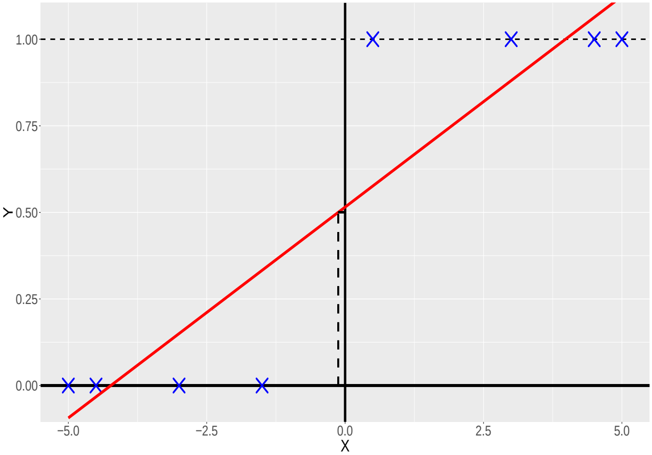
- **Given:** n datapoints $\mathbf{x}_i$ with labels $y_i \in \{0, 1\}$
- **Task:** find $g(\mathbf{x})$ such that $g(\mathbf{x}_i) = y_i$
- $\Rightarrow$ **Classification Task**
- **First (bad) idea:** fit a linear regression line $g_{\mathrm{LR}}(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$
- **Then:**

$$y_i = \begin{cases} 0 & g_{\mathrm{LR}}(x_i) < 0.5 \\ 1 & g_{\mathrm{LR}}(x) \geq 0.5 \end{cases}$$

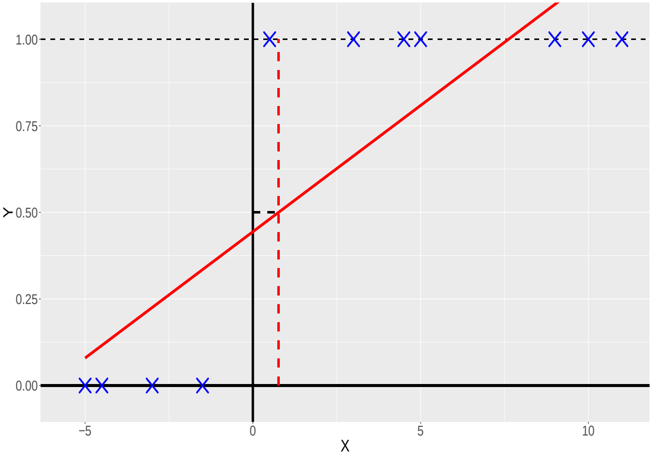
Problem with Linear Regression



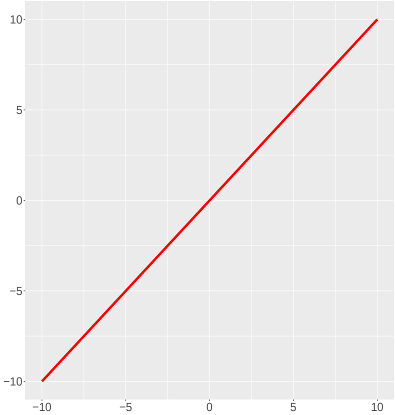
Problem with Linear Regression



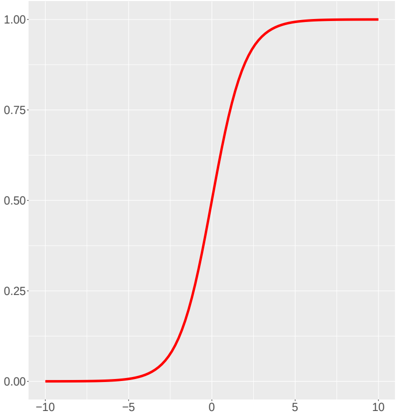
Problem with Linear Regression



Logistic Function



⇒



Also known as **sigmoid function**

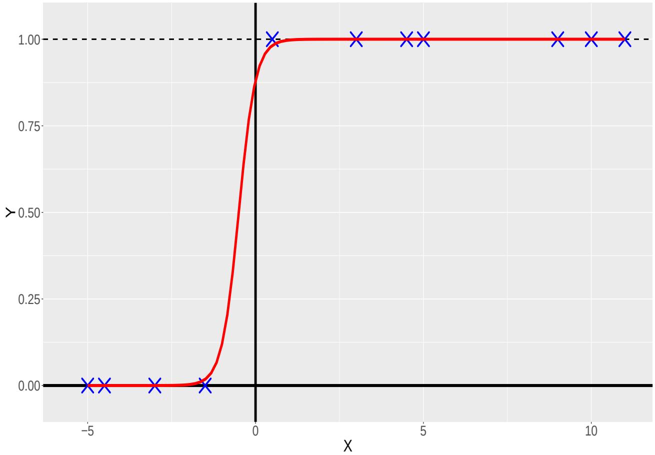
Logistic Regression

- Models relationship between a **categorical** label and some features $\mathbf{x}$
- The relationship is not linear, instead we apply the logistic function

$$\sigma(t) = \frac{1}{1 + e^{-t}}$$

- t is a linear function of features: $t = \mathbf{w}^T\mathbf{x}$
- Model: $g(\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x})$

Logistic Regression



Objective

- Likelihood function for a Bernoulli distribution:

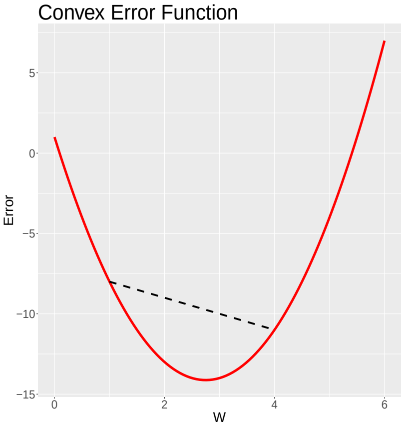
$$\mathcal{L}(\{\mathbf{x}, y\}; \mathbf{w}) = \prod_{i=1}^{n} g(\mathbf{x}_i; \mathbf{w})^{y_i} \cdot (1 - g(\mathbf{x}_i; \mathbf{w}))^{1-y_i}$$

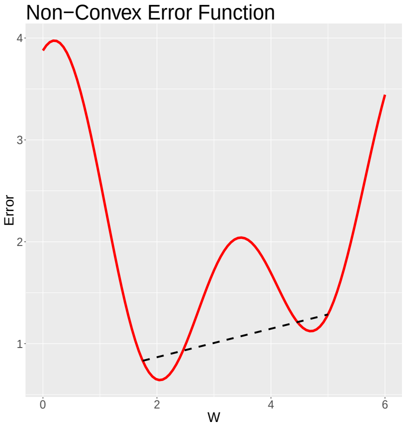
- Taking the negative logarithm, we obtain:

$$\begin{aligned} L &= -\log \mathcal{L}(\{\mathbf{x}, y\}; \mathbf{w}) = \\ &= -\sum_{i} [y_i \log g(\mathbf{x}_i; \mathbf{w}) + (1 - y_i) \log(1 - g(\mathbf{x}_i; \mathbf{w}))] \end{aligned}$$

Also known as the **Cross Entropy Error**, makes Logistic Regression a **convex problem.**

Convex vs. Non-Convex





Logistic Regression Problem

- **Task:**

$$\min_{\mathbf{w}} L \;=\; \min_{\mathbf{w}}\Big(-\sum_i \left[y_i \log g(\mathbf{x}_i;\mathbf{w}) + (1-y_i)\log(1-g(\mathbf{x}_i;\mathbf{w}))\right]\Big)$$

  Where $g(\mathbf{x};\mathbf{w}) = \sigma(\mathbf{w}^T\mathbf{x}) = \frac{1}{1+\mathrm{e}^{-\mathbf{w}^T\mathbf{x}}}$

- **Note:** no closed-form solution!
  You have to use methods like Gradient Descent, Newton, BFGS, Conjugate Gradient, ...

- Derivative of sigmoid function:

$$\frac{\partial\sigma(x)}{\partial x} = \sigma(x)\cdot(1-\sigma(x))$$

Gradient Descent

The minimization of a function $L(.;\mathbf{w})$ can be done by Gradient Descent

$$\mathbf{w}_{n+1} = \mathbf{w}_n - \eta \frac{\partial L}{\partial \mathbf{w}}$$

where η is the learning rate

and $\mathbf{w}_0$ is some initial guess for $\mathbf{w}$

Softmax

- Generalization of the sigmoid function
- Suitable for **multi-class** classification
- For K classes with $y \in \{1, \ldots, K\}$ the probability of $\mathbf{x}$ belonging to class k is

$$p(y = k|\mathbf{x}) = \frac{e^{\mathbf{w}_k^T \mathbf{x}}}{\sum_{j=1}^{K} e^{\mathbf{w}_j^T \mathbf{x}}}$$

- With the objective:

$$\min_{\mathbf{w}} L \;=\; \min_{\mathbf{w}} \Big(- \sum_k \sum_i [y_i]_k \log p(y_i = k|\mathbf{x}; \mathbf{w}) \Big)$$

where $[y_i]_k$ is the k-th entry of the *one-hot* vector $[y_i]$

Gradient Checking

- Method for checking if the symbolic computation/implementation of the gradient was correct
- Logistic Regression gradient is easy, but once we get to Neural Networks, you'll be glad to know this trick
- **Idea:** compare your gradient with a numerical approximation of the gradient

Gradient Checking (2)

- Central difference quotient:

$$\frac{\partial L}{\partial w_i} \approx \frac{L(.;\mathbf{w} + \epsilon\ \mathbf{e}_i) - L(.;\mathbf{w} - \epsilon\ \mathbf{e}_i)}{2\ \epsilon}$$

  with $\mathbf{e}_i = (0\,0\ \ldots\ 1\ \ldots\ 0)^T$

- Good choice: $\epsilon = 10^{-4}$