# ASSIGNMENT 6: LOGISTIC REGRESSION

**Markus Holzleitner,**
**Johannes Kofler,**
**Angela Bitto-Nemling**
Institute for Machine Learning

JOHANNES KEPLER UNIVERSITY LINZ

Institute for Machine Learning

# Copyright statement:

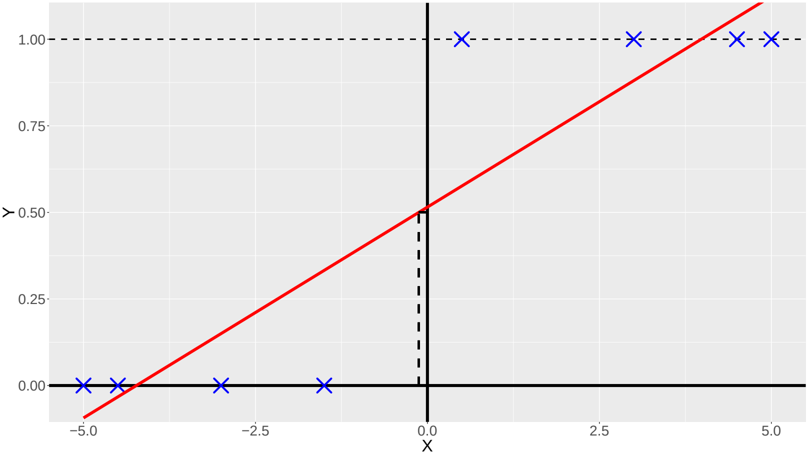
# Logistic Regression

- Given a bunch of data $\mathbf{Z} = \{(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), \ldots, (\mathbf{x}_l, \mathbf{y}_l)\}$
  - $\mathbf{x}_i$ ... vectors of features
  - $\mathbf{y}_i$ ... labels
- **Classification**: $y_i \in \{1, \ldots, M\}$
- **First (bad) idea:** fit a linear regression line $g_{LR}(\mathbf{x}_i) = \mathbf{W}^T \mathbf{x}_i$
- **Goal:**

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

- **Problem:** the resulting $y_i(\mathbf{x}_i)$ would be a step function, i.e. not continuous

# Problem with Linear Regression



# Logistic Regression

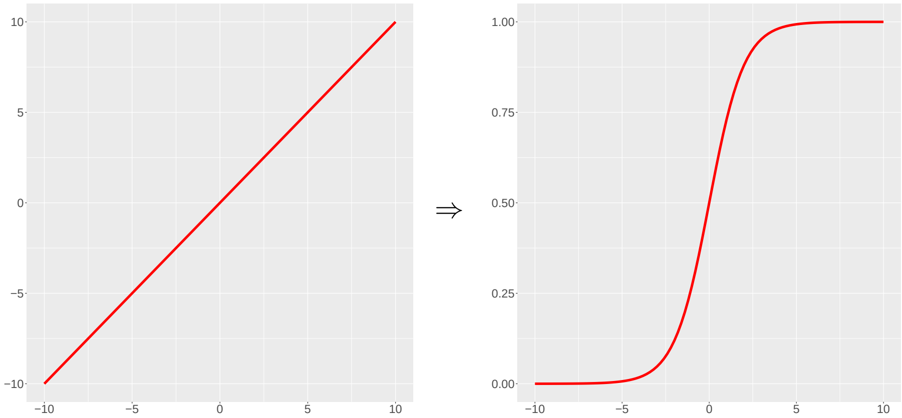
- Instead of predicting directly the class we would like to predict the class probability, which is a regression problem
- Therefore we apply the logistic function:

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

- $z$ is a linear function of features: $z = \mathbf{W}^T \mathbf{x}$
- Model:

$$g(\mathbf{x}) = \sigma(\mathbf{W}^T \mathbf{x})$$

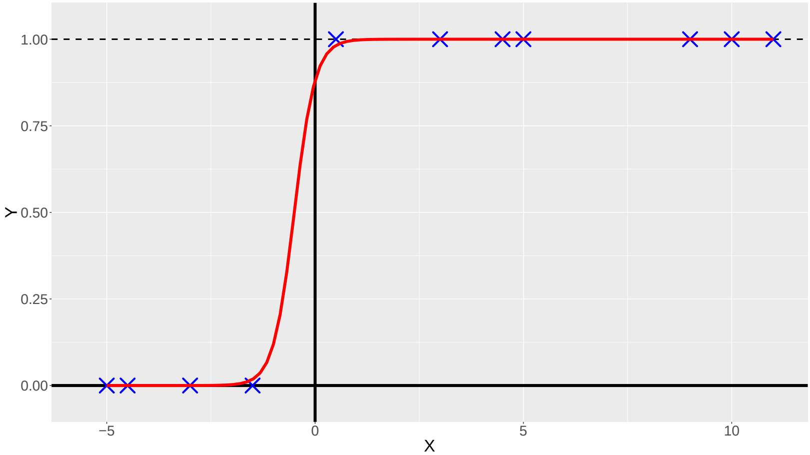
# Logistic Function




Also known as **sigmoid function**.

Also known as **Fermi function** in physics.

# Logistic Regression



# Objective

- Likelihood function for a Bernoulli distribution:

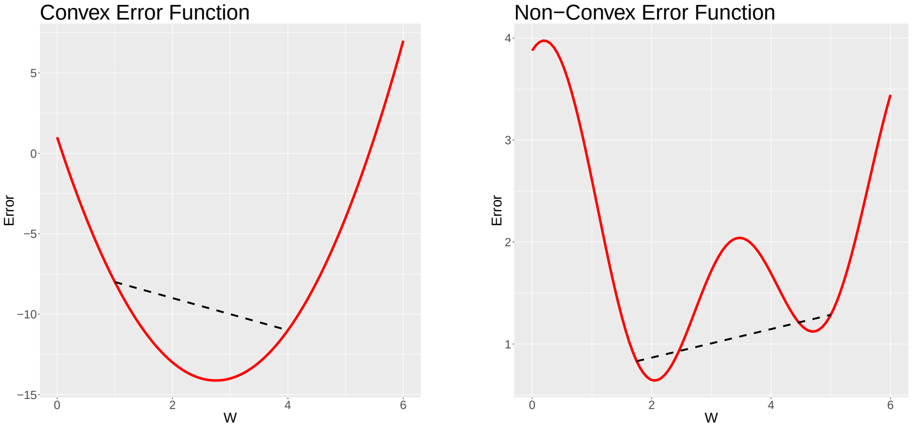
$$\mathcal{L}(\{\mathbf{x}, y\}; \mathbf{W}) = \prod_{i=1}^{n} g(\mathbf{x}_i; \mathbf{W})^{y_i} \cdot (1 - g(\mathbf{x}_i; \mathbf{W}))^{1-y_i}$$

- Taking the negative logarithm, we obtain:

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

- $L$ is known as the **Cross Entropy Loss**
- The task $\min_{\mathbf{W}} L$ does in general not have a closed-form solution
- Fortunately $L$ is a **convex function** for Logistic Regression, therefore, every local minimum is a global minimum.

# Convex vs. non-convex



# Gradients for Logistic Regression

- **Using:** $\frac{\partial \sigma(z)}{\partial z} = \sigma(z) \cdot (1 - \sigma(z))$
- **Using** $\sigma_i$ instead of $\sigma(\mathbf{W}^T \mathbf{x}_i)$

$$L = -\sum_i \Big( y_i \log \sigma(\mathbf{W}^T \mathbf{x}_i) + (1 - y_i) \log(1 - \sigma(\mathbf{W}^T \mathbf{x}_i)) \Big)$$

$$\begin{aligned}
\frac{\partial L}{\partial \mathbf{W}} &= -\sum_i \left( y_i \frac{1}{\sigma_i} \cdot \sigma_i \cdot (1 - \sigma_i) \cdot \mathbf{x}_i - \frac{1 - y_i}{1 - \sigma_i} \cdot \sigma_i \cdot (1 - \sigma_i) \cdot \mathbf{x}_i \right) \\
&= -\sum_i \Big( y_i (1 - \sigma_i) \cdot \mathbf{x}_i - (1 - y_i) \cdot \sigma_i \cdot \mathbf{x}_i \Big) \\
&= -\sum_i (y_i - y_i \sigma_i - \sigma_i + \sigma_i y_i) \mathbf{x}_i \\
&= \sum_i (\sigma_i - y_i) \mathbf{x}_i
\end{aligned}$$

# Logistic Regression Problem

- **Task:**

$$\min_{\mathbf{W}} L = \min_{\mathbf{W}} \left[ -\sum_i \big( y_i \log g(\mathbf{x}^i; \mathbf{W}) + (1 - y_i) \log(1 - g(\mathbf{x}^i; \mathbf{W})) \big) \right]$$

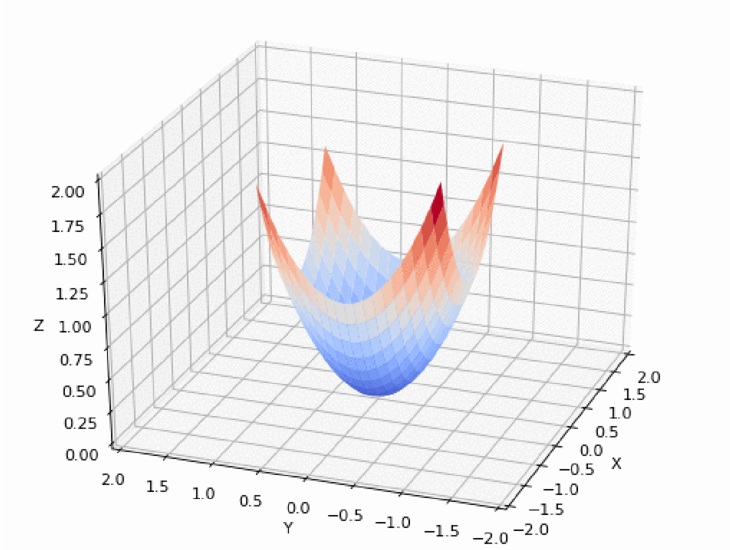
  - $g(\mathbf{x}; \mathbf{W}) = \sigma(\mathbf{W}^T \mathbf{x}) = \frac{1}{1 + \mathrm{e}^{-\mathbf{W}^T \mathbf{x}}}$

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

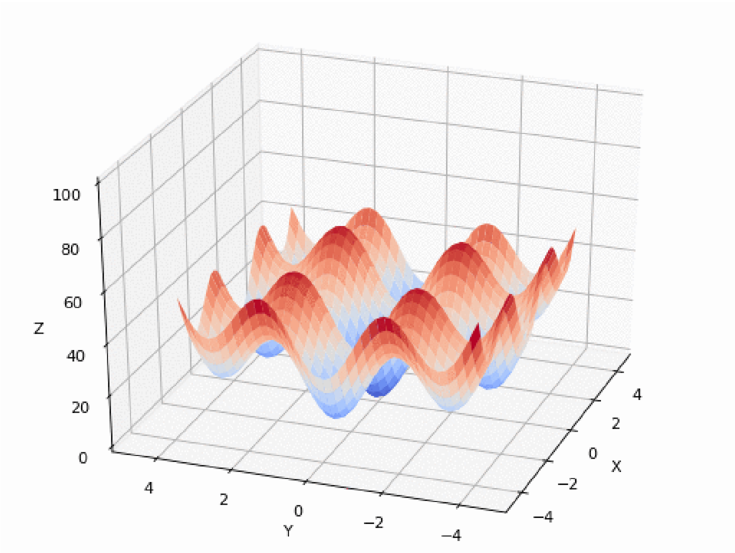
# Gradient Descent

- **Given:** a function $f(x)$
- **Task:** find $x$ that maximizes (or minimizes) $f(x)$
- **Idea:** start at some value $x_0$, and take a small step $\eta$ in the direction in which the function decreases strongest.
  Find derivative $f' = \frac{\partial f}{\partial x}$.
  - $-f'(x_0)$ is the direction of steepest descent at $x_0$.
- **Solution:**
  - Iteratively calculate $x_{i+1} = x_i - \eta \cdot f'(x_i)$.
  - Each $x_{i+1}$ should be a better solution than $x_i$.
  - Eventually you'll reach a (**local**) minimum.

# Gradient descent



# Gradient descent for non-convex problems



# Gradient Descent in Logistic Regression

The minimization of the loss 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 $\eta$ is the learning rate,

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

# 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

- Central difference quotient:

$$\frac{\partial L}{\partial W_{ij}} \approx \frac{L(.;\mathbf{W} + \epsilon\,\mathbf{e}_{ij}) - L(.;\mathbf{W} - \epsilon\,\mathbf{e}_{ij})}{2\,\epsilon}$$

- Central difference quotient for logistic regression ($\mathbf{W}$ is a vector):

$$\frac{\partial L}{\partial W_{i}} \approx \frac{L(.;\mathbf{W} + \epsilon\,\mathbf{e}_{i}) - L(.;\mathbf{W} - \epsilon\,\mathbf{e}_{i})}{2\,\epsilon}$$

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

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