Machine Learning

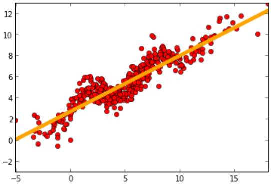
– winter term 2016/17 –

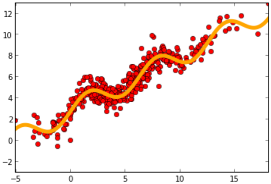
Chapter 04: Logistic Regression

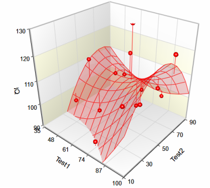
Prof. Adrian Ulges
Masters "Computer Science"
DCSM Department
University of Applied Sciences RheinMain

Classification vs. Regression

- Given: training samples $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathcal{X}$ with labels $y_1, \ldots, y_n$
- **Classification**
  - Labels indicate class membership
  - Learn a **classifier** function $\mathbb{R}^d \to \{1, \ldots, C\}$, assigning samples to classes.
- **Regression**
  - Labels are **real-valued**!
  - Learn a **regression** function $f : \mathbb{R}^d \to \mathbb{R}$, assigning samples to continuous values.
- **Regression Examples** *(incl. the "classic": linear regression)*







Logistic Regression: Approach

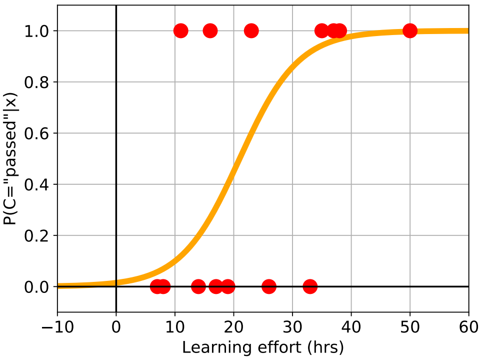
- **Logistic Regression** (aka. **Maximum Entropy**) is a common approach in statistical data analysis[1]
- Idea: Use a **regression model for classification**
  - Compute a **score** for each class using regression
  - This score should approximate the probability that the given object belongs to class c, given that its features are $\mathbf{x}$: $P(c|\mathbf{x})$
  - The classifier picks the class with **maximum score**

[1] Cox, DR (**1958**), "The regression analysis of binary sequences (with discussion)". J Roy Stat Soc B.

Logistic Regression: Approach

- Assumption: 2 classes only (0 vs. 1)
  (success/failure, well/sick, ...)
- **Given**: a test sample $\mathbf{x}$
- **Goal**: estimate $P(C = 1|\mathbf{x})$

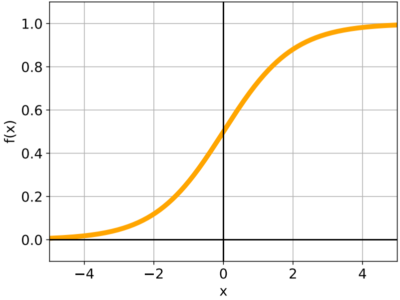
Example (math exam)



Logistic Regression: Model

- As a base model, we use the so-called **Sigmoid** function

$$P(C = 1|x) \approx f(x) = \frac{1}{1 + e^{-x}}$$



- **Property A**: $lim_{x \to -\infty} f(x) = 0$ and $lim_{x \to \infty} f(x) = 1$
- **Property B**: $P(C = 1|x = 0) = f(0) = 0.5$
  $\rightarrow$ We choose class 1 iff. $x \geq 0$.

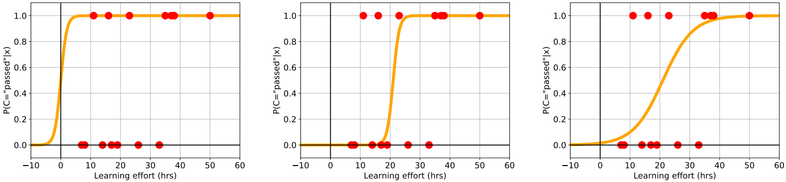
Logistic Regression: Model

Extension

- We allow a shift and scaling of the sigmoid:

$$f(x; w_0, w) = \frac{1}{1 + e^{-(w_0 + w \cdot x)}}$$

- The parameters w_0, w are estimated via learning (soon...)



Multi-variate Logistic Regression

- Goal: apply logistic regression in case of multiple features $\mathbf{x} \in \mathbb{R}^d$?
- We extend the sigmoid function:

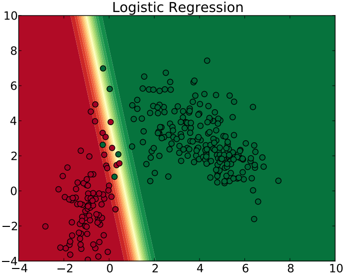
$$f(\mathbf{x}; w_0, w_1, w_2 \ldots, w_d) = \frac{1}{1 + e^{-(w_0 + w_1 \cdot x_1 + w_2 \cdot x_2 + \ldots + w_d \cdot x_d)}}$$

  or short (with vector $\boldsymbol{w} := (w_1, \ldots, w_d)$):

$$f(\mathbf{x}; w_0, \boldsymbol{w}) = \frac{1}{1 + e^{-(w_0 + \mathbf{x} \cdot \boldsymbol{w})}}$$

- The boundary between the two classes (or **decision boundary**) of this model is at $\boldsymbol{x} \cdot \boldsymbol{w} + w_0 = 0$.
  This is a **hyperplane** (in normal form)!

Logistic Regression: Illustration



- Because the decision boundary is **linear**, we call logistic regression a **linear classifier** *(there are a few more → later!)*.

Parameters

- $\boldsymbol{w}$ determines the **orientation** of the decision boundary.
- $\boldsymbol{w}$ also determines the **slope** of the decision function f.
- w_0 determines the **shift** of the boundary.

Logistic Regression: Training

Key Question: Training

- **Given**: a set of training samples $\mathbf{x}_1, ..., \mathbf{x}_n \in \mathbb{R}^d$ with Labels $y_1, ..., y_n \in \{0, 1\}$
- **Goal**: Determine w_0 and $\mathbf{w}$ ($=$ the position and slope of the decision function)

Approach: Maximum-likelihood Estimation

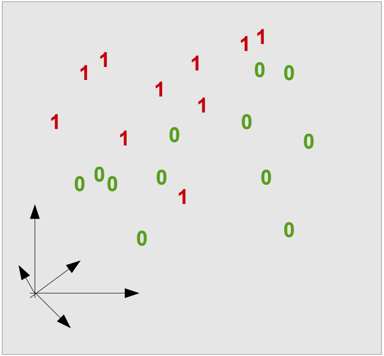
- **Idea**: Choose the parameters such that the observed samples becomes "most likely".
- For positive samples ($y_i = 1$), $f(\mathbf{x}_i)$ should be high:

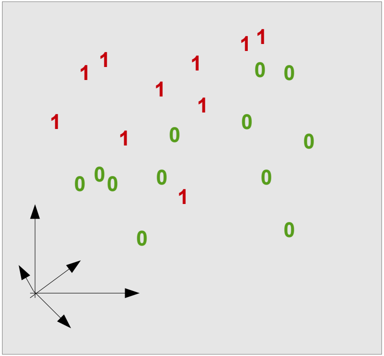
$$\Big(P(C = 1|\mathbf{x}_i)\Big) \approx f(\mathbf{x}_i) \approx 1$$

- For negative samples ($y_i = 1$), $f(\mathbf{x}_i)$ should be low:

$$\Big(P(C = 1|\mathbf{x}_i)\Big) \approx f(\mathbf{x}_i) \approx 0$$

Logistic Regression: Example





Logistic Regression: Formalization

Maximum-Likelihood Estimation

We define a likelihood function and formulate an optimization problem:

$$w_0^*, \mathbf{w}^* = \arg\max_{w_0,\mathbf{w}} \underbrace{\prod_{i:y_1=1} f(\mathbf{x}_i) \cdot \prod_{i:y_i=0} (1 - f(\mathbf{x}_i))}_{\text{"Likelihood function"}\, L(w_0,\mathbf{w})}$$

We rewrite the optimization problem:

$$\begin{aligned} w_0^*, \mathbf{w}^* &= \arg\max_{w_0,\mathbf{w}} \prod_{i:y_1=1} f(\mathbf{x}_i) \cdot \prod_{i:y_i=0} (1 - f(\mathbf{x}_i)) \\ &= \arg\max_{w_0,\mathbf{w}} \prod_{i} f(\mathbf{x}_i)^{y_i} \cdot (1 - f(\mathbf{x}_i))^{1-y_i} && \text{// log} \\ &= \arg\max_{w_0,\mathbf{w}} \sum_{i} y_i \cdot log(f(\mathbf{x}_i)) + (1 - y_i) \cdot log(1 - f(\mathbf{x}_i)) \end{aligned}$$

Logistic Regression: Formalization

$$
\begin{aligned}
w_0^*, \mathbf{w}^* &= \underset{w_0, \mathbf{w}}{\arg\max} \underbrace{\prod_{i:y_1=1} f(\mathbf{x}_i) \cdot \prod_{i:y_i=0} (1 - f(\mathbf{x}_i))}_{\text{"Likelihood-Funktion"}\ L(w_0, \mathbf{w})} \\
&= \underset{w_0, \mathbf{w}}{\arg\max} \prod_i f(\mathbf{x}_i)^{y_i} \cdot (1 - f(\mathbf{x}_i))^{1-y_i} \qquad // \log \\
&= \underset{w_0, \mathbf{w}}{\arg\max} \sum_i y_i \cdot log(f(\mathbf{x}_i)) + (1 - y_i) \cdot log(1 - f(\mathbf{x}_i)) \\
&= \underset{w_0, \mathbf{w}}{\arg\max} \sum_i log(1 - f(\mathbf{x}_i)) + y_i \cdot log\left(\frac{f(\mathbf{x}_i)}{1 - f(\mathbf{x}_i)}\right) \\
&= \underset{w_0, \mathbf{w}}{\arg\max} \sum_i log\left(\frac{\cancel{1} + exp(-(w_0 + \mathbf{x}_i\mathbf{w})) - \cancel{1}}{1 + exp(-(w_0 + \mathbf{x}_i\mathbf{w}))}\right) + y_i \cdot log\left(\frac{\frac{1}{\cancel{(1+exp(-(w_0+\mathbf{x}_i\mathbf{w})))}}}{\frac{exp(-(w_0+\mathbf{x}_i\mathbf{w}))}{\cancel{(1+exp(-(w_0+\mathbf{x}_i\mathbf{w})))}}}\right) \\
&= \underset{w_0, \mathbf{w}}{\arg\max} \sum_i -log\left(\frac{1 + exp(-(w_0 + \mathbf{x}_i\mathbf{w}))}{exp(-(w_0 + \mathbf{x}_i\mathbf{w}))}\right) - y_i \cdot \cancel{log}\left(\cancel{exp}(-(w_0 + \mathbf{x}_i\mathbf{w}))\right) \\
&= \underset{w_0, \mathbf{w}}{\arg\max} \sum_i -log\left(e^{w_0 + \mathbf{x}_i\mathbf{w}} + 1\right) + \sum_i y_i \cdot (w_0 + \mathbf{x}_i\mathbf{w})
\end{aligned}
$$

Logistic Regression: Formalization

$$\arg\max_{w_0,\mathbf{w}} \underbrace{\sum_i -log\left(e^{w_0+\mathbf{x}_i\mathbf{w}}+1\right)+\sum_i y_i\cdot(w_0+\mathbf{x}_i\mathbf{w})}_{\text{"Log-Likelihood Function"}\, L(w_0,\mathbf{w})}$$

Remarks

- There is no analytical solution for maximizing the Log-Likelihood function L.
- We solve the problem **numerically**: For example, by finding roots of the gradient using **Newton's method**.
- The weights $\mathbf{w}$ indicate the **importance** of the single features for the classification problem.

Logistic Regression: Regularization

- **Observation**: Even though logistic regression is fairly robust, the model tends to **overfit** when ...
  - ... single features get a very extreme weight
  - ... many unimportant weights get a weight $\neq 0$.
- To avoid this, we **regularize** the problem, such that the entries in $\mathbf{w}$ tend to be small (or even 0).
- We define the **norm** of the weight vector $\mathbf{w}$

$$||\mathbf{w}||_1 := |w_1| + |w_2| + ... + |w_d| \qquad \text{L1 norm}$$

$$||\mathbf{w}||_2 := \sqrt{w_1^2 + w_2^2 + ... + w_d^2} \qquad \text{L2 norm}$$

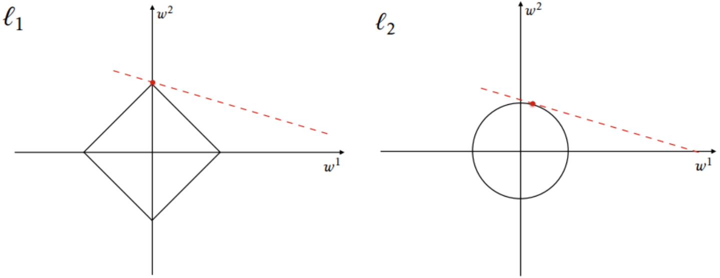
- We adapt the optimization problem such that high weights in $\mathbf{w}$ are sanctioned (with $C \in \mathbb{R}$):

$$\arg\max_{w_0, \mathbf{w}} \; L(w_0, \mathbf{w}) \; - \; C \cdot ||\mathbf{w}||_1 \qquad // \text{ L1-Regularization}$$

$$\arg\max_{w_0, \mathbf{w}} \; L(w_0, \mathbf{w}) \; - \; C \cdot ||\mathbf{w}||_2 \qquad // \text{ L2-Regularization}$$

What Difference does L1 vs. L2 make?

Example: Optimizing a Linear Function (regularized)



Left: $\mathbf{w} = (0, 1)$ (= L1 solution). **Right**: $\mathbf{w} = (0.15, 0.99)$ (=L2 solution).

- **L1-Regularization** enforces the weights of uninformative features to be 0 (the weight vector is **sparse**). Put differently: The classifier conducts a built-in **feature selection**.
- **L2-Regularization** reduces outliers (= extreme weights)

Logistic Regression with > 2 Classes

Approach

- Divide the problem into many binary problems

Approach 1: One-vs-rest

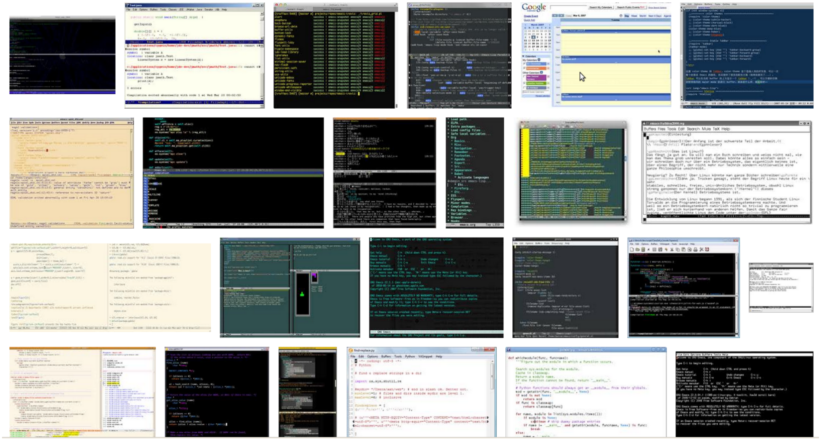
- Learn one classifier φ_c for each class c
- This classifier separates samples in c from **all other** samples (binary!)
- Given a new sample $\mathbf{x}$, compute all class scores $\varphi_1(\mathbf{x}), \varphi_2(\mathbf{x}), ..., \varphi_C(\mathbf{x})$ and choose the highest-scored class
- How many classifiers do we need? $\rightarrow C$.

Approach 2: One-vs-one

- Learn one classifier φ_{c_1,c_2} for each **pair** of classes c_1, c_2
- This classifier is trained only on Samples from these two classes (binary!)
- Given a new sample, compute all Scores $\varphi_{1,1}(\mathbf{x}), \varphi_{1,2}(\mathbf{x}), ..., \varphi_{C-1,C}(\mathbf{x})$
- Choose the class that wins most **comparisons**
- How many classifiers do we need? $\rightarrow \frac{C \cdot (C-1)}{2}$.

Logistic Regression: Discussion

Logistic Regression: Code Sample



- Bag-of-Words Features
- Logistic Regression
- Inspect term weights