https://piazza.com/tum.de/fall2018/in2357

From Regression to Classification

Categories of Learning (Rep.)

Learning

Unsupervised Learning

clustering, density estimation

Supervised Learning

learning from a training data set, **inference** on the test data

Reinforcement Learning

no supervision, but a **reward function**

Regression

target set is **continuous**, e.g.

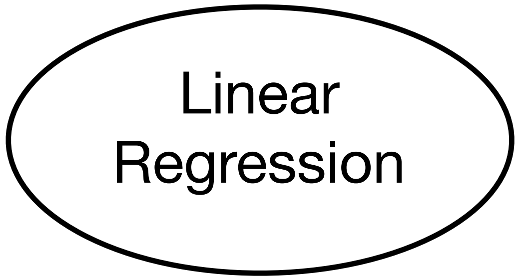
$\mathcal{Y} = \mathbb{R}$

Classification

target set is **discrete**, e.g.

$\mathcal{Y} = [1, \ldots, C]$

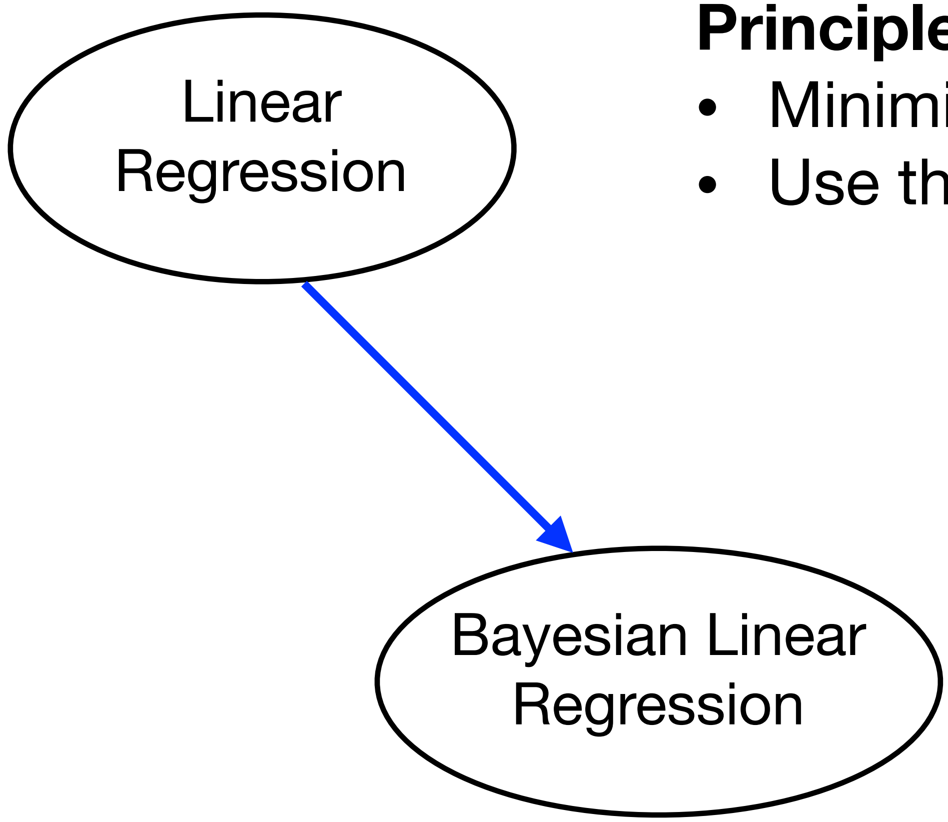
Visualisation



Principle:

- Minimise loss function during training
- Use the found parameters for prediction

Visualisation



Principle:

- Minimise loss function during training
- Use the found parameters for prediction

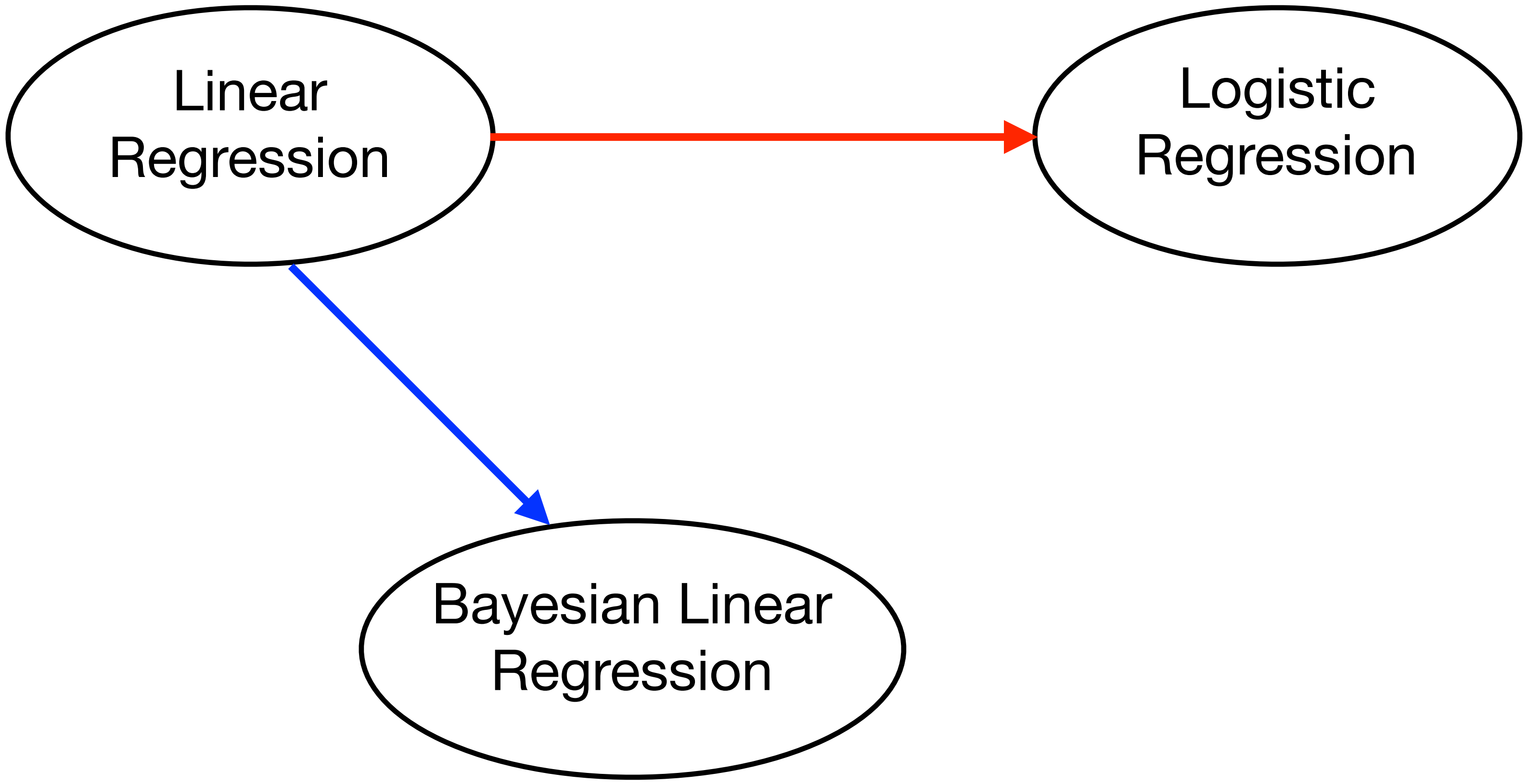
Principle:

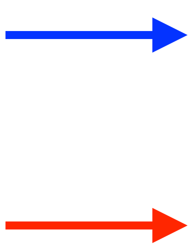
- Compute parameter posterior $p(\mathbf{w} \mid \mathbf{x}, \mathbf{t})$ from training data
- During inference, compute the predictive distribution $p(t^* \mid x^*, \mathbf{x}, \mathbf{t})$

Advantages:

- Less tendency of overfitting
- Probabilistic interpretation, uncertainty estimation

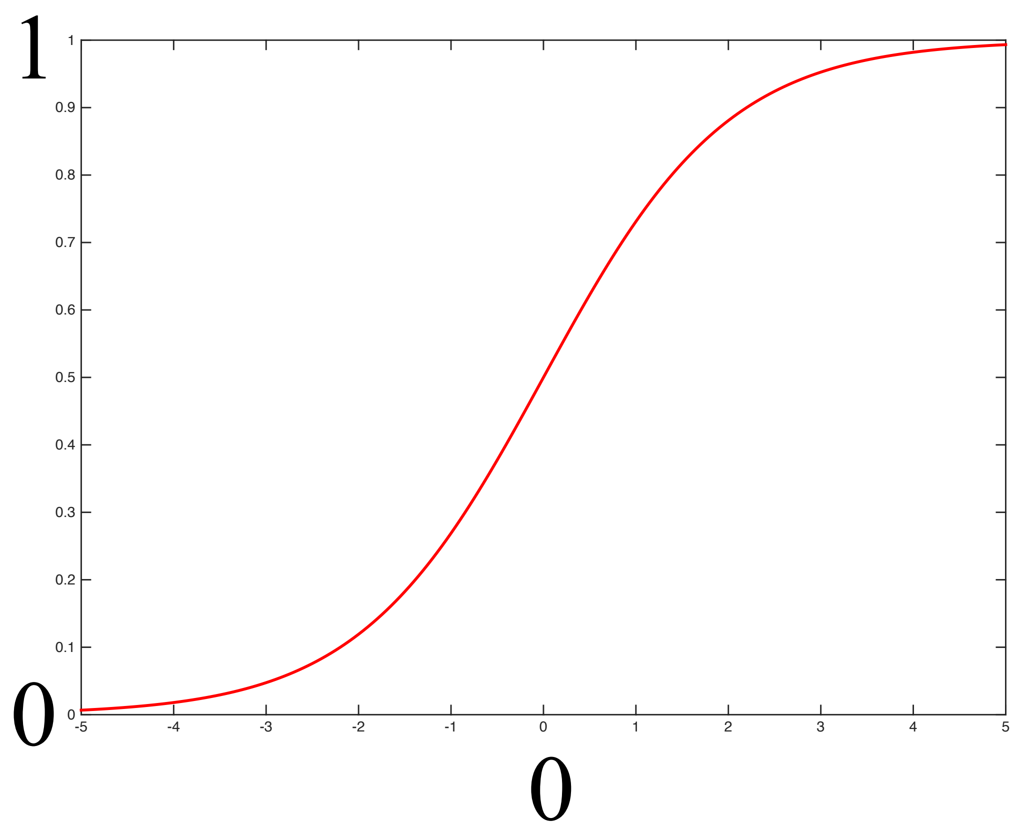
Visualisation





Logistic Regression

To convert the regression problem into a classification problem, we use a **sigmoid** function $\sigma()$, e.g.:



$$\sigma(a) = \frac{1}{1 + \exp(-a)}$$

This can be interpreted as a **classification probability**

Logistic Regression

To convert the regression problem into a classification problem, we use a **sigmoid** function $\sigma()$

We still use our linear prediction model

$$f(x, \mathbf{w}) = \mathbf{w}^T \phi(x)$$

but now we use the sigmoid function to model a **foreground class** probability

$$y_i = \sigma(\mathbf{w}^T \phi(\mathbf{x}_i)) \qquad i = 1, \ldots, N$$

Thus, we consider a **two-class** problem (binary classification).

Logistic Regression

Again, we formulate a model of the **likelihood** of the training data:

$$p(\mathbf{t} \mid \mathbf{x}, \mathbf{w}) = \prod_{i=1}^{N} p(t_i \mid x_i, \mathbf{w}) \qquad t_i \in \{0, 1\}$$

But now, we use the **Bernoulli** distribution:

$$p(t_i \mid x_i, \mathbf{w}) = y_i^{t_i}(1 - y_i)^{(1-t_i)}$$

And again, we aim to maximise the (log)-likelihood

$$\arg\max_{\mathbf{w}} p(\mathbf{t} \mid \mathbf{x}, \mathbf{w})$$

Logistic Regression

We minimise the negative log-likelihood:

$$E(\mathbf{w}) = -\log p(t_1, \dots, t_N \mid \mathbf{x}, \mathbf{w})$$

Logistic Regression

We minimise the negative log-likelihood:

$$E(\mathbf{w}) = -\log p(t_1, \dots, t_N \mid \mathbf{x}, \mathbf{w})$$
$$= -\sum_{i=1}^{N} (t_i \log y_i + (1 - t_i) \log(1 - y_i))$$

"Cross entropy"

Logistic Regression

We minimise the negative log-likelihood:

$$
\begin{aligned}
E(\mathbf{w}) &= -\log p(t_1, \ldots, t_N \mid \mathbf{x}, \mathbf{w}) \\
&= -\sum_{i=1}^{N} (t_i \log y_i + (1 - t_i) \log(1 - y_i))
\end{aligned}
$$

"Cross entropy"

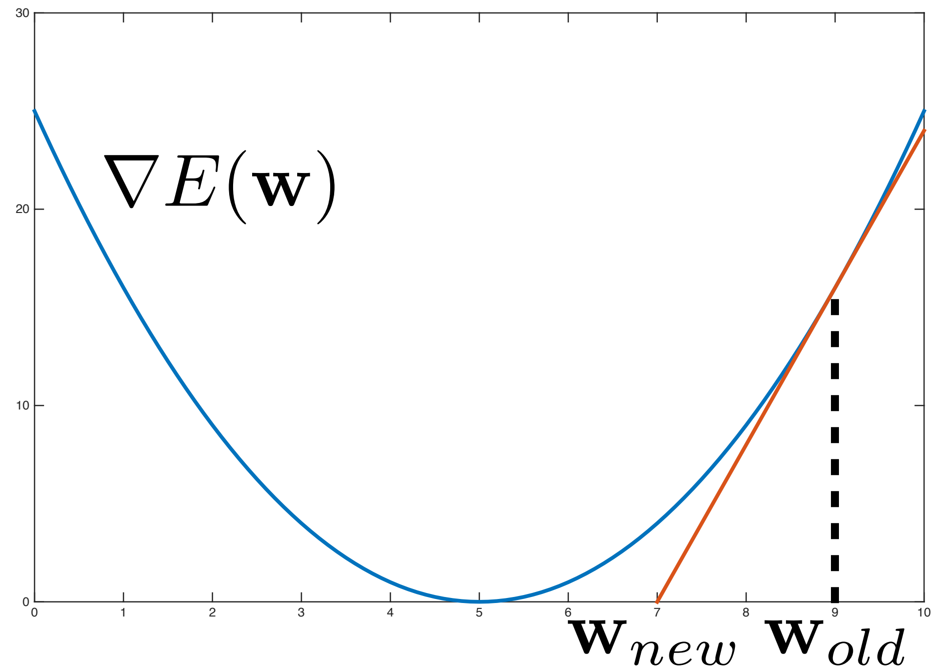
$$
\nabla E(\mathbf{w}) = \sum_{i=1}^{N} (y_i - t_i) \boldsymbol{\phi}(\mathbf{x}_i)
$$

Minimisation

Problem: The error equation can not be solved in closed form

$$\nabla E(\mathbf{w}) = \sum_{i=1}^{N} (\sigma(\mathbf{w}^T \phi(\mathbf{x}_i)) - t_i)\phi(\mathbf{x}_i)$$

Instead, we need to apply an iterative approach, e.g. Newton-Raphson



$$\mathbf{w}_{new} = \mathbf{w}_{old} - \underline{H^{-1}}\nabla E(\mathbf{w})$$

Hessian Matrix

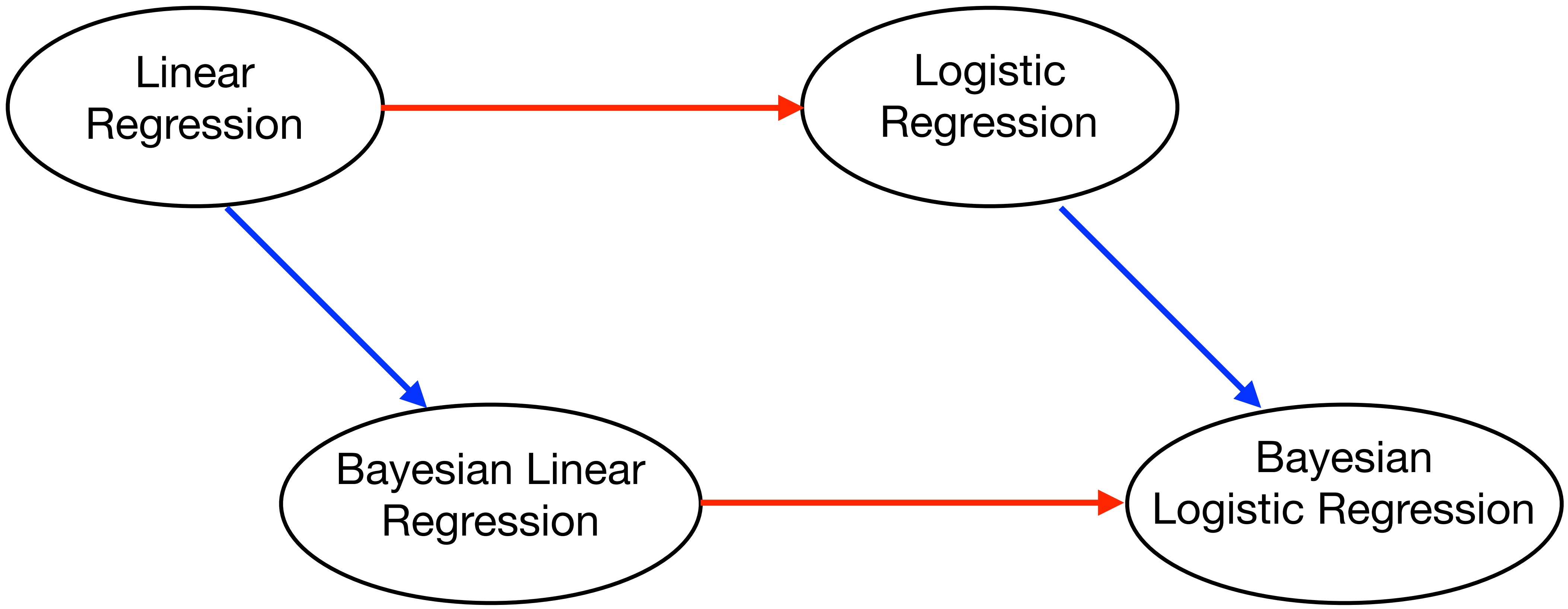
Iterative Weighted Least Squares

The update rule for the logistic regression methods is then:

$$\mathbf{w}_{new} = \mathbf{w}_{old} - (\Phi^T R \Phi)^{-1} \Phi^T (\mathbf{y} - \mathbf{t})$$

Where the weighting metric **R** depends also on the weights **w**

Visualisation



Bayesian Logistic Regression

- We can also use the Bayesian formulation to do classification
- Idea: formulate a prior distribution over $\mathbf{w}$

- **Problem:** The likelihood is not Gaussian, therefore we won't have a closed form solution for the posterior

- Therefore: We approximate the posterior using **Laplace approximation**