I – Linear models on feature vectors

Contents:

- *Linear regression model*
- *L2-loss for regression and normal equations*
- *Linear classification model*
- *Softmax function, cross-entropy loss for classification*

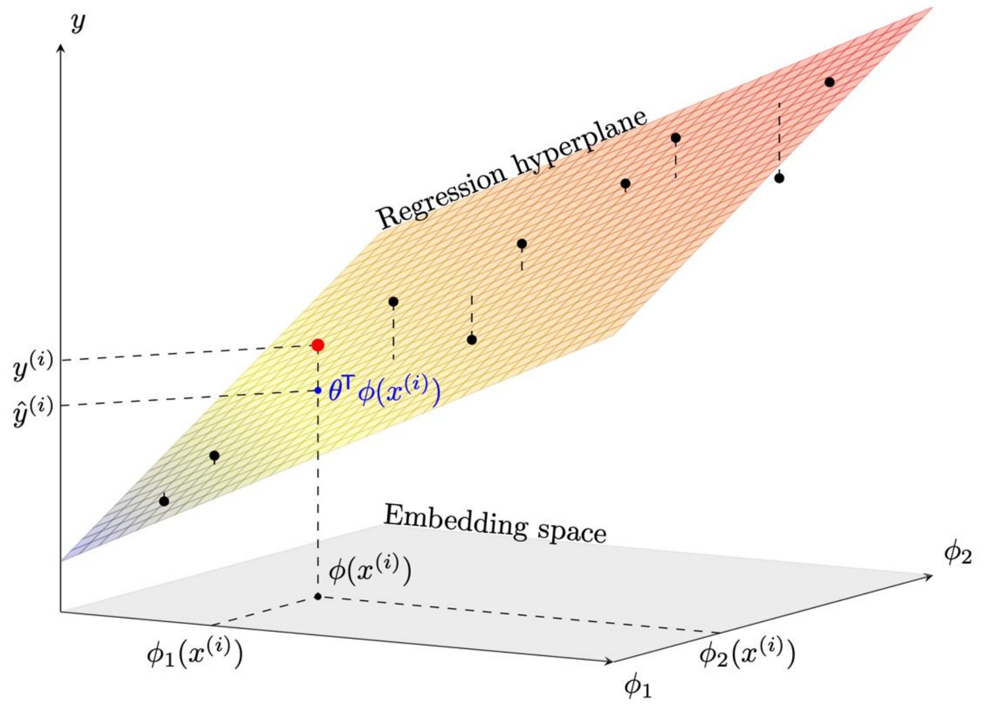
Linear regression: the model

Linear regression

- What kind of problems one can solve efficiently, even in large dimensions? → **Linear systems!**
- Let us talk first about regression: the answer is modelled as

$$f_{\boldsymbol{\theta}}\left(\phi_1^{(i)}, \phi_2^{(i)}, \cdots\right) = \boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{\phi}^{(i)} = \hat{y}^{(i)}$$

- It is sometimes convenient to add an affine term (also called **bias** in the neural network literature), which can be absorbed in the feature vector making it of dimension $d' + 1$ where $\boldsymbol{\theta} = [\theta_0, \theta_1, \theta_2, \cdots]^{\mathrm{T}}, \boldsymbol{\phi}^{(i)} = \left[1, \phi_1^{(i)}, \phi_2^{(i)}, \cdots\right]^{\mathrm{T}}$.
- **Geometric interpretation**: projection of an embedding vector onto a **hyperplane** parameterised by $\boldsymbol{\theta}$.



Linear regression: example and ERM

Linear regression

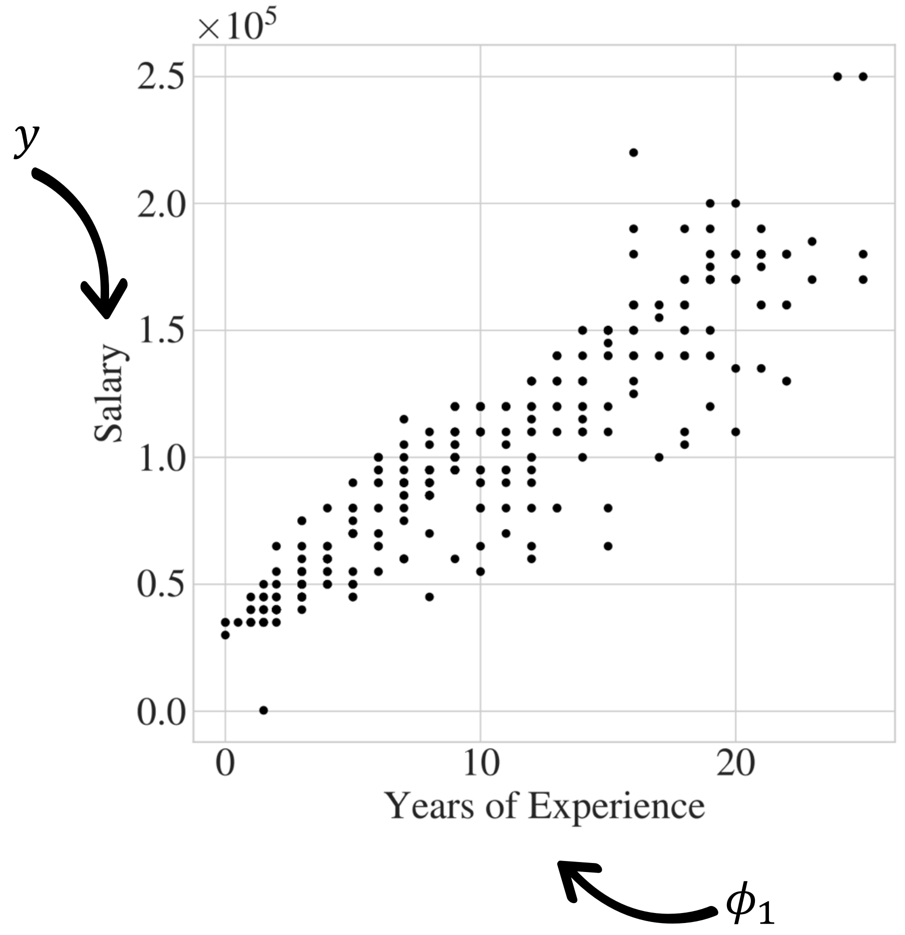
- Example: salary prediction based on the years of experience
- Data are 373 couples $\left(\phi^{(i)}, y^{(i)}\right) \Longrightarrow$ **Supervised learning**
- The target variable $y \in \mathbb{R}$ is continuous $\Longrightarrow$ **Regression**
- The linear model is

$$\hat{y}^{(i)} = \theta_0 + \theta_1 \phi_1^{(i)},$$

where $\phi_1^{(i)}$ is the nb. of years of experience of the i^{th} training example

- Now the model is fixed, how to find $\widehat{\boldsymbol{\theta}}$, the best possible parameters for our model and data?
- This is done using **empirical risk minimisation (ERM)**

$$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, R(\boldsymbol{X}, \boldsymbol{\theta}) = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, \frac{1}{n} \sum_{i=1}^{n} \ell\left(\hat{y}^{(i)}, y^{(i)}\right)$$



Linear regression: solution to ERM

Linear regression

- A common **choice** of loss for regression is a **squared loss function**

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n}\left(\hat{y}^{(i)} - y^{(i)}\right)^2$$

- ***Here***, the optimisation problem can be solved analytically in closed-form. Rewriting the risk matricially, we have

$$R(\boldsymbol{X}, \boldsymbol{\theta}) = \frac{1}{n}\|\boldsymbol{\Phi}\boldsymbol{\theta} - \boldsymbol{y}\|_2^2$$

Feature matrix

$$\boldsymbol{\Phi} = \begin{pmatrix} \phi_1^{(1)} & \cdots & \phi_1^{(d\prime)} \\ \vdots & \ddots & \vdots \\ \phi_n^{(1)} & \cdots & \phi_n^{(d\prime)} \end{pmatrix} \in \mathbb{R}^{n\times d'}$$

Target vector

$$\boldsymbol{y} = [y_1, \dots, y_n]^{\mathrm{T}} \in \mathbb{R}^n$$

The analytical minimisation of the squared loss in linear regression gives the unique solution (when $d' < n$) known as ***normal equations***

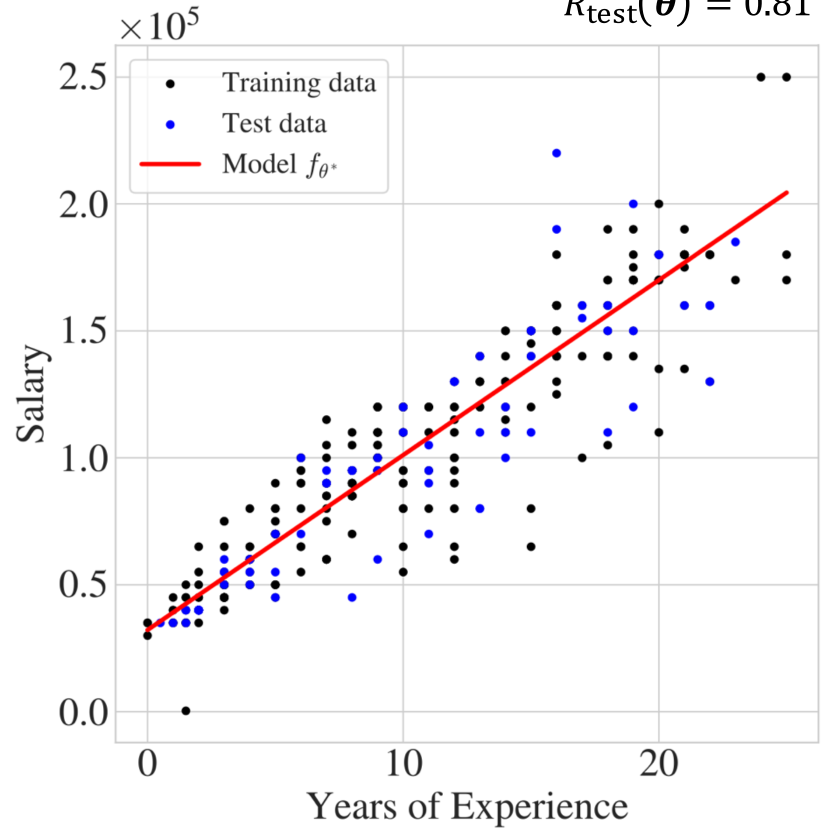
$$\hat{\boldsymbol{\theta}} = \left(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{y}$$

Linear regression: illustration in 1D

Linear regression

1. I **first** separated the dataset into **training and test sets**, $n = 298$ and $n_{\text{test}} = 75$ chosen randomly.
2. Then, I computed the optimal parameters minimising the empirical risk using the normal equations on the training features
$$\hat{\boldsymbol{\theta}} = \left(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{y}.$$
3. I computed the risk on the train and test sets and found they are close.

$$R_{\text{train}}(\hat{\boldsymbol{\theta}}) = 0.87$$
$$R_{\text{test}}(\hat{\boldsymbol{\theta}}) = 0.81$$



➕ Exactly solvable model, low variance

➖ Cannot represent local relationships, may be biased

Linear regression: ERM and MLE

Linear regression

Remark

- The choice of the squared loss can also be motivated from a probabilistic point of view
- Assuming a Gaussian distribution for the error $\epsilon^{(i)} = \hat{y}^{(i)} - y^{(i)} \sim \mathcal{N}(0, \sigma^2)$ and **independent** observations, the ***likelihood*** can be written

$$p(\boldsymbol{X}|\boldsymbol{\theta}) = \prod_i p\left(y^{(i)} \middle| \boldsymbol{x}^{(i)}, \boldsymbol{\theta}\right)$$

- Maximising the log-likelihood to obtain the parameters of the model gives

$$\max_\theta \log p(\boldsymbol{X}|\boldsymbol{\theta}) = \max_\theta -\frac{1}{2\sigma_\epsilon^2} \sum_i \left(y^{(i)} - \hat{y}^{(i)}\right)^2$$

→ **The maximum likelihood estimator (MLE) is the same as the empirical risk minimiser under a squared loss function to measure the error of the model**

Linear classification: the model

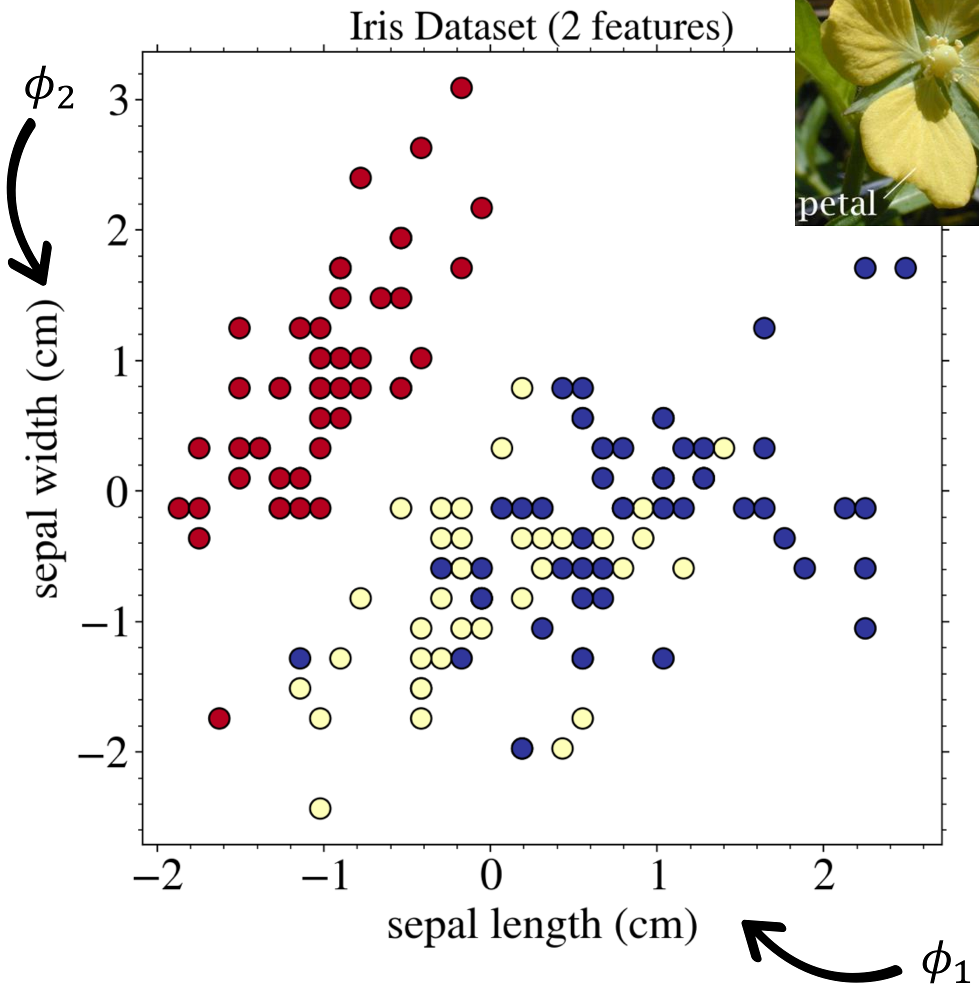
Linear classification

- Consider a K-class classification problem for which features $\boldsymbol{\phi}$ allow linear separability
- Example: Iris dataset with 150 couples $\left(\phi^{(i)}, y^{(i)}\right) \Longrightarrow$ **Supervised learning**
- The target variable $y \in \{0,1,2\} \Longrightarrow$ **Classification**
- A natural loss function for classification is the **0-1 loss**

$$\ell(y, \hat{y}) = \begin{cases} 1 & \text{if } \hat{y}^{(i)} \neq y^{(i)}, \\ 0 & \text{otherwise.} \end{cases}$$

- The optimal decision rule (in Bayes sense) is

$$\hat{y} = \operatorname{argmax}_k p(y = k|\boldsymbol{\phi}).$$



We thus need a **model** $p_{\boldsymbol{\theta}}(y = k|\boldsymbol{\phi})$ of the conditional probability distribution to perform classification!

Linear classification: the model

Linear classification

- The simplest models assume a linear log probability

$$\log p_\theta\left(y^{(i)} = k \middle| \boldsymbol{\phi}^{(i)}\right) = \boldsymbol{\theta}_k^\mathrm{T}\, \boldsymbol{\phi}^{(i)} - \log Z$$

where Z is a normalizing constant so that probabilities sum to one.

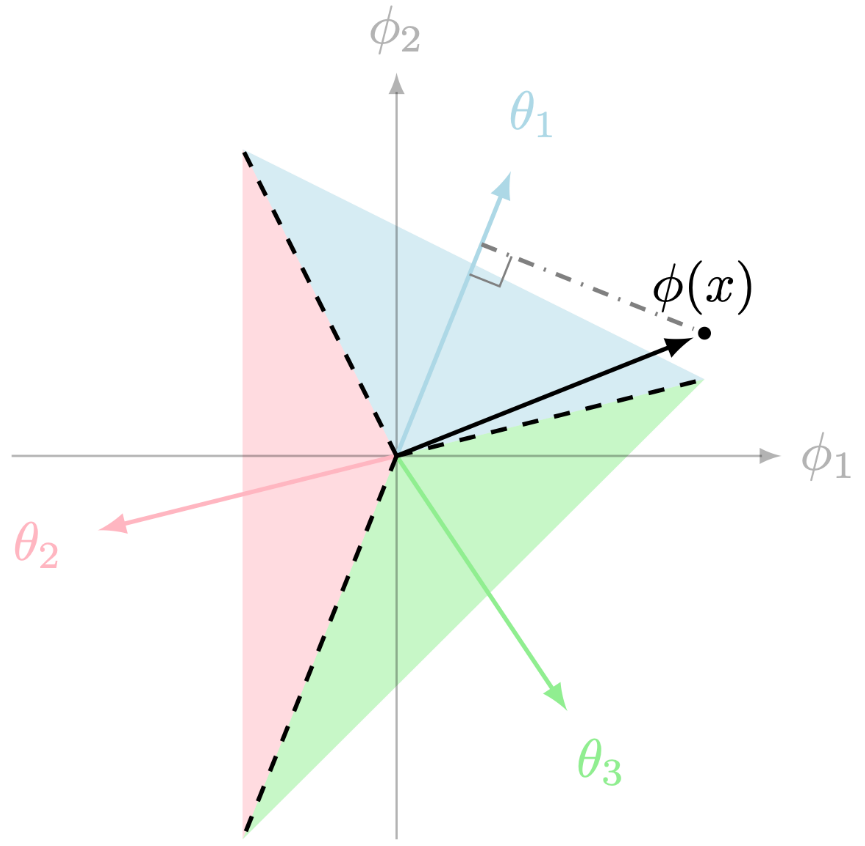
- It means that

$$p_\theta\left(y^{(i)} = k \middle| \boldsymbol{\phi}^{(i)}\right) = \frac{\exp\left(\boldsymbol{\theta}_k^\mathrm{T} \boldsymbol{\phi}^{(i)}\right)}{\sum_{j=1}^{K} \exp\left(\boldsymbol{\theta}_j^\mathrm{T} \boldsymbol{\phi}^{(i)}\right)}$$

which is called the **softmax function** allowing to turn the linear responses for each class into probabilities.

- And the classification rule is

$$\hat{y} = \mathrm{argmax}_k\, \boldsymbol{\theta}_k^\mathrm{T}\, \boldsymbol{\phi}^{(i)}$$



Geometrically, it corresponds to computing the overlap between the feature $\boldsymbol{\phi}^{(i)}$ and a vector representative for each class, $\boldsymbol{\theta}_k$, and associating **the class maximising the dot product**, leading to **linear decision** boundaries shown as hyperplanes.

Linear classification: fitting parameters

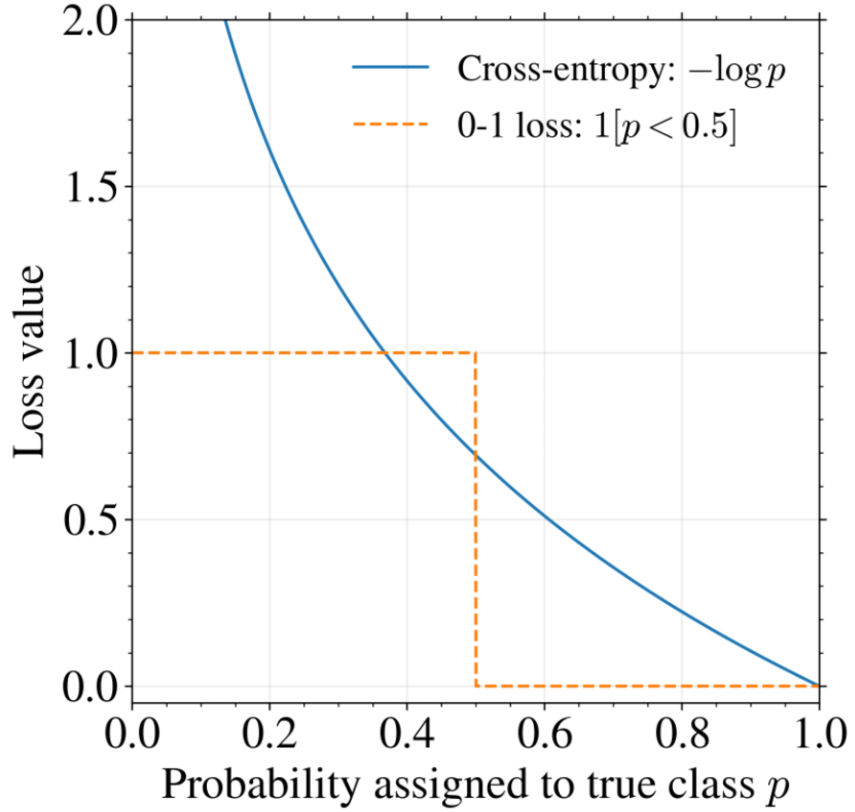
Linear classification

- Now the model is specified, we need minimise the risk to obtain the parameters $\boldsymbol{\theta}_k$ using some training data

- For optimisation, the 0-1 loss is not suitable since it is not differentiable, but we can relax it using the probabilities

$$\ell(y, \hat{y}) = -\sum_{k=1}^{K} 1_{y=k} \log p_\theta(y = k | \boldsymbol{\phi}^{(i)})$$

which is **now differentiable and convex.**



Linear classification: fitting parameters

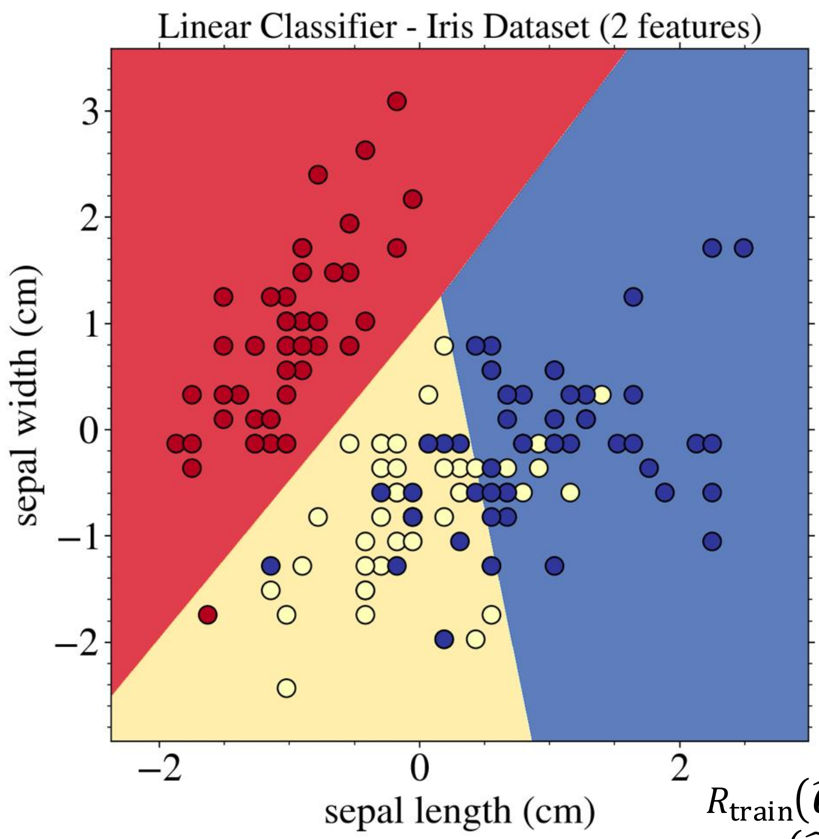
Linear classification

- Now the model is specified, we need minimise the risk to obtain the parameters $\boldsymbol{\theta}_k$ using some training data

- For optimisation, the 0-1 loss is not suitable since it is not differentiable, but we can relax it using the probabilities

$$\ell(y, \hat{y}) = -\sum_{k=1}^{K} 1_{y=k} \log p_\theta(y = k|\boldsymbol{\phi}^{(i)})$$

which is **now differentiable and convex.**



$R_{\text{train}}(\widehat{\boldsymbol{\theta}}) = 0.38$

$R_{\text{test}}(\widehat{\boldsymbol{\theta}}) = 0.48$

This risk is referred to as ***cross-entropy*** and it is the most widely used cost function for classification problems. The parameters of the model are then obtained by minimising the risk, i.e.

$$\widehat{\boldsymbol{\theta}} = \text{argmin}_\theta \, R(\boldsymbol{X}, \boldsymbol{\theta}).$$