Introduction to Probabilistic Machine Learning

Ralf Herbrich

Non-Bayesian Classification

Overview

1. Geometry of Linear Classifiers
2. Fisher's Linear Discriminant
3. Perceptron Learning Algorithm
4. Logistic Regression

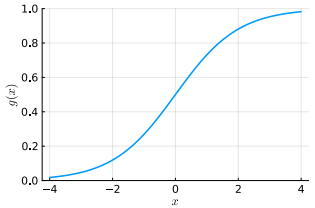
Overview

1. **Classification Learning**
2. Fisher's Linear Discriminant
3. Perceptron Learning Algorithm
4. Logistic Regression

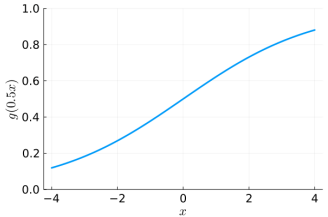
Classification Learning Setting

- **Given**:
  1. **Training Data**: $D \in (\mathcal{X} \times \{0,1\})^n$ of n labelled examples (x, y) from input space $\mathcal{X}$
  2. **Linear Basis Functions**: Basis function mapping $\boldsymbol{\phi}: \mathcal{X} \to \mathbb{R}^M$ and linear function model
  $$f(x; \boldsymbol{w}) := \boldsymbol{w}^{\mathrm{T}} \boldsymbol{\phi}(x)$$
  3. **Likelihood of functions**: Let $g: \mathbb{R} \to [0,1]$ be a fixed monotone function. Then
  $$p(D|f) = p(D|\boldsymbol{w}) = \prod_{i=1}^{n} \mathrm{Ber}\left(y_i; g\left(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{\phi}(x_i)\right)\right)$$
  4. **Prior belief over functions**:
  $$p(f) = p(\boldsymbol{w}) = \prod_{j=1}^{M} \mathcal{N}\left(w_j; 0, \tau^2\right)$$
- **Bayesian Inference**:
  - **Posterior belief over functions**:
  $$p(f|D) = p(\boldsymbol{w}|D) = \frac{\prod_{i=1}^{n} \mathrm{Ber}\left(y_i; g\left(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{\phi}(x_i)\right)\right) \cdot \prod_{j=1}^{M} \mathcal{N}\left(w_j; 0, \tau^2\right)}{\int_{\mathbb{R}^M} \prod_{i=1}^{n} \mathrm{Ber}\left(y_i; g\left(\widetilde{\boldsymbol{w}}^{\mathrm{T}} \boldsymbol{\phi}(x_i)\right)\right) \cdot \prod_{j=1}^{M} \mathcal{N}\left(\widetilde{w}_j; 0, \tau^2\right) \, d\widetilde{\boldsymbol{w}}}$$

$$g(t) = \frac{\exp(t)}{1 + \exp(t)}$$



$$\lim_{s \to \infty} g(s \cdot t) = \mathbb{I}(t > 0)$$

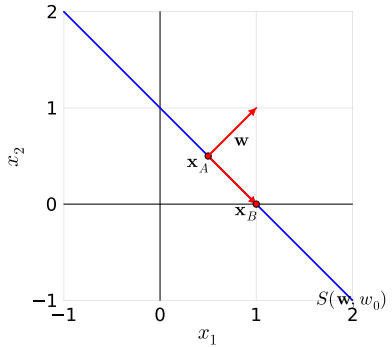


Geometry of Linear Classifiers

- **Linear Model**: Consider the case where $\phi_0 = 1, \phi_i(\boldsymbol{x}) = x_i$ and $\lim_{s\to\infty} g(s \cdot t) = \mathbb{I}(t > 0)$.

$$f(x; \boldsymbol{w}, w_0) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + w_0$$

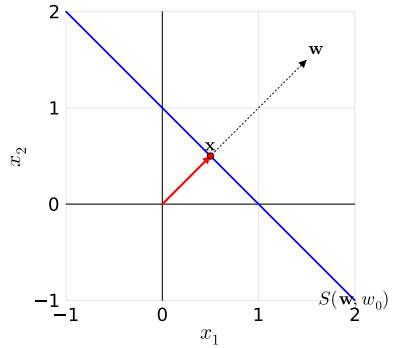
- **Decision Surface**. *The decision surface $S(\boldsymbol{w}, w_0) \subseteq \mathcal{X}$ are all points x where $f(x; \boldsymbol{w}, w_0) = 0$.*



$$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_A + w_0 = 0$$

$$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_B + w_0 = 0$$

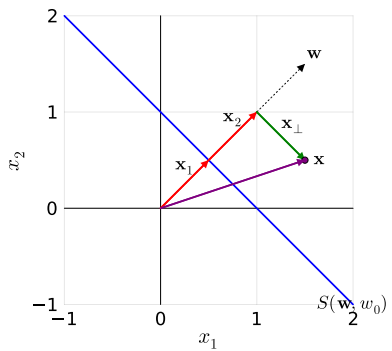
$$\boldsymbol{w}^{\mathrm{T}}(\boldsymbol{x}_B - \boldsymbol{x}_A) = 0$$



$$\boldsymbol{x} = \alpha\boldsymbol{w}$$

$$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + w_0 = \alpha\boldsymbol{w}^{\mathrm{T}}\boldsymbol{w} + w_0 = 0$$

$$\|\boldsymbol{x}\| = \alpha\|\boldsymbol{w}\| = -\frac{w_0}{\|\boldsymbol{w}\|^2} \cdot \|\boldsymbol{w}\| = -\frac{w_0}{\|\boldsymbol{w}\|}$$



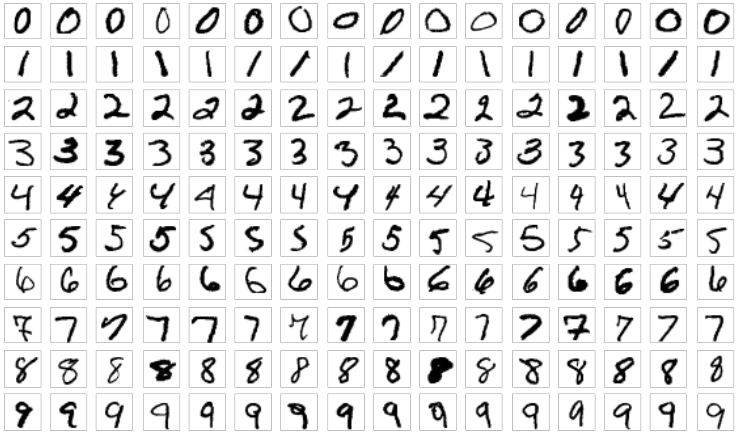
$$\boldsymbol{x} = \boldsymbol{x}_1 + \boldsymbol{x}_2 + \boldsymbol{x}_\perp = (\alpha + r)\boldsymbol{w} + \boldsymbol{x}_\perp$$

$$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} = (\alpha + r)\|\boldsymbol{w}\|^2 \Leftrightarrow r = \frac{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + w_0}{\|\boldsymbol{w}\|^2}$$

$$\|\boldsymbol{x}_2\| = r\|\boldsymbol{w}\| = \frac{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + w_0}{\|\boldsymbol{w}\|}$$

Classification Learning Example: MNIST

- **MNIST** (Modified National Institute of Standards and Technology database) is a database of handwritten digits
  - Created in 1998 from two datasets of handwritten digits of high-school students and American Census Bureau employees
  - Created by Yann LeCun, Corinna Cortes, and Christ Burges
  - 60,000 training images/10,000 test images
  - Original black-and-white images were scaled to 28x28 pixels and anti-aliased to give grey scales
- Standard dataset to compare classification learning algorithms from 1998 – 2010 (before ImageNet)
- **Smallest prediction error**: 0.18% (2018)



Yann LeCun
(1960 –)

Corinna Cortes
(1961 –)

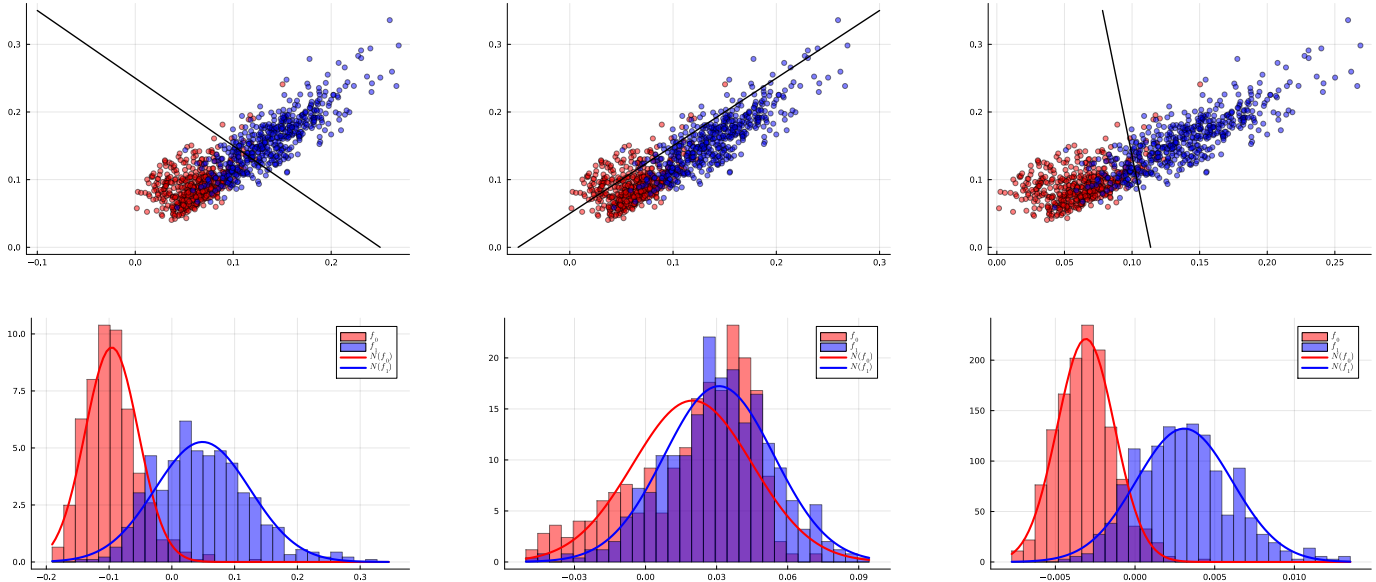
Chris Burges
(1960 –)

Overview

1. Classification Learning
2. **Fisher's Linear Discriminant**
3. Perceptron Learning Algorithm
4. Logistic Regression

Marginal Distribution after Linear Projection

- **Idea**: Given a linear basis function model $f(x; \boldsymbol{w}, w_0) := \boldsymbol{w}^{\mathrm{T}} \boldsymbol{\phi}(x) + w_0$, estimate the marginal distribution of the $f(x_i)$ with a Normal distribution separately for the class $y_i = 0$ and $y_i = 1$



Fisher's Criterion

- **Distribution of projection** per class k: $p(f(x; \boldsymbol{w}, w_0)|\boldsymbol{w}, w_0, y = k) \approx \mathcal{N}\left(f; \mu_k, \sigma_k^2\right)$

$$\mu_{\mathrm{ML},k}(\boldsymbol{w}) = \frac{1}{n_k} \sum_{y_i=k} \boldsymbol{w}^{\mathrm{T}} \boldsymbol{\phi}(x_i) + w_0$$

$$\sigma^2_{\mathrm{ML},k}(\boldsymbol{w}) = \frac{1}{n_k} \sum_{y_i=k} \left(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{\phi}(x_i) - \mu_{\mathrm{ML},k}(\boldsymbol{w})\right)^2$$

- **Fisher Criterion**: Maximize over $\boldsymbol{w}$ *between-class* distance and minimize *within-class* distance

$$J(\boldsymbol{w}) = \frac{\left[\mu_{\mathrm{ML},0}(\boldsymbol{w}) - \mu_{\mathrm{ML},1}(\boldsymbol{w})\right]^2}{\sigma^2_{\mathrm{ML},0}(\boldsymbol{w}) + \sigma^2_{\mathrm{ML},1}(\boldsymbol{w})}$$

$$\boldsymbol{m}_k := \frac{1}{n_k} \sum_{y_i=k} \boldsymbol{\phi}(x_i)$$

$$J(\boldsymbol{w}) = \frac{[\boldsymbol{w}^{\mathrm{T}}(\boldsymbol{m}_0 - \boldsymbol{m}_1)]^2}{\frac{1}{n_0}\sum_{y_i=0}[\boldsymbol{w}^{\mathrm{T}}(\boldsymbol{\phi}(x_i) - \boldsymbol{m}_0)]^2 + \frac{1}{n_1}\sum_{y_i=1}[\boldsymbol{w}^{\mathrm{T}}(\boldsymbol{\phi}(x_i) - \boldsymbol{m}_1)]^2}$$

Sir Ronald Fisher
(1890 – 1962)

Fisher's Linear Discriminant Analysis (LDA)

- **Expanding the Fisher Criterion** by introducing matrices $\boldsymbol{S}_B$ and $\boldsymbol{S}_W$

$$J(\boldsymbol{w}) = \frac{\boldsymbol{w}^{\mathrm{T}} \overbrace{(\boldsymbol{m}_0 - \boldsymbol{m}_1)(\boldsymbol{m}_0 - \boldsymbol{m}_1)^{\mathrm{T}}}^{\boldsymbol{S}_B} \boldsymbol{w}}{\underbrace{\boldsymbol{w}^{\mathrm{T}} \left(\frac{1}{n_0} \sum_{y_i=0} (\boldsymbol{\phi}(x_i) - \boldsymbol{m}_0)(\boldsymbol{\phi}(x_i) - \boldsymbol{m}_0)^{\mathrm{T}} + \frac{1}{n_1} \sum_{y_i=1} (\boldsymbol{\phi}(x_i) - \boldsymbol{m}_1)(\boldsymbol{\phi}(x_i) - \boldsymbol{m}_1)^{\mathrm{T}} \right) \boldsymbol{w}}_{\boldsymbol{S}_W}}$$

- **Solution**: Maximize $J(\boldsymbol{w})$ over the choice of $\boldsymbol{w}$.

$$\frac{\partial J(\boldsymbol{w})}{\partial \boldsymbol{w}} = \frac{\partial}{\partial \boldsymbol{w}} \frac{\boldsymbol{w}^{\mathrm{T}} \boldsymbol{S}_B \boldsymbol{w}}{\boldsymbol{w}^{\mathrm{T}} \boldsymbol{S}_W \boldsymbol{w}}$$

$$\left. \frac{\partial J(\boldsymbol{w})}{\partial \boldsymbol{w}} \right|_{\boldsymbol{w}_{\mathrm{LDA}}} = 2\boldsymbol{S}_B \boldsymbol{w}_{\mathrm{LDA}} \cdot \left(\boldsymbol{w}_{\mathrm{LDA}}^{\mathrm{T}} \boldsymbol{S}_W \boldsymbol{w}_{\mathrm{LDA}} \right)^{-1} - 2\left(\boldsymbol{w}_{\mathrm{LDA}}^{\mathrm{T}} \boldsymbol{S}_B \boldsymbol{w}_{\mathrm{LDA}} \right) \left(\boldsymbol{w}_{\mathrm{LDA}}^{\mathrm{T}} \boldsymbol{S}_W \boldsymbol{w}_{\mathrm{LDA}} \right)^{-2} \boldsymbol{S}_W \boldsymbol{w}_{\mathrm{LDA}} = \boldsymbol{0}$$

$$\boldsymbol{S}_W \boldsymbol{w}_{\mathrm{LDA}} = \frac{\boldsymbol{w}_{\mathrm{LDA}}^{\mathrm{T}} \boldsymbol{S}_W \boldsymbol{w}_{\mathrm{LDA}}}{\boldsymbol{w}_{\mathrm{LDA}}^{\mathrm{T}} \boldsymbol{S}_B \boldsymbol{w}_{\mathrm{LDA}}} \boldsymbol{S}_B \boldsymbol{w}_{\mathrm{LDA}}$$

$$\boldsymbol{S}_W \boldsymbol{w}_{\mathrm{LDA}} \propto (\boldsymbol{m}_0 - \boldsymbol{m}_1)[(\boldsymbol{m}_0 - \boldsymbol{m}_1)^{\mathrm{T}} \boldsymbol{w}_{\mathrm{LDA}}] \propto (\boldsymbol{m}_0 - \boldsymbol{m}_1)$$

$$\boldsymbol{w}_{\mathrm{LDA}} \propto \boldsymbol{S}_W^{-1} (\boldsymbol{m}_0 - \boldsymbol{m}_1)$$

Fisher's Linear Discriminant Analysis (ctd.)

- **Question**: How to determine the **optimal threshold** w_0?
- **Idea**: Choose it such that the weighted mid-point between the two means $\mu_{\mathrm{ML},0}(\boldsymbol{w}_{\mathrm{LDA}})$ and $\mu_{\mathrm{ML},1}(\boldsymbol{w}_{\mathrm{LDA}})$ is exactly at zero (remember $g(x) = \mathbb{I}(x > 0)$)!

$$\frac{n_0}{n_0 + n_1} \cdot \mu_{\mathrm{ML},0}(\boldsymbol{w}_{\mathrm{LDA}}) + \frac{n_1}{n_0 + n_1} \cdot \mu_{\mathrm{ML},1}(\boldsymbol{w}_{\mathrm{LDA}}) = 0$$

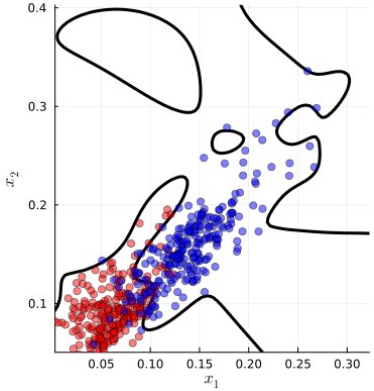
$$\frac{n_0}{n_0 + n_1}\left(\frac{1}{n_0}\sum_{y_i=0} \boldsymbol{w}_{\mathrm{LDA}}^{\mathrm{T}}\boldsymbol{\phi}(x_i) + w_{\mathrm{LDA},0}\right) + \frac{n_1}{n_0 + n_1}\left(\frac{1}{n_1}\sum_{y_i=1} \boldsymbol{w}_{\mathrm{LDA}}^{\mathrm{T}}\boldsymbol{\phi}(x_i) + w_{\mathrm{LDA},0}\right) = 0$$

$$\boldsymbol{w}^{\mathrm{T}}\underbrace{\left(\frac{1}{n_0 + n_1}\sum_{i=1}^{n_0+n_1} \boldsymbol{\phi}(x_i)\right)}_{\boldsymbol{m}} + w_{\mathrm{LDA},0} = 0$$

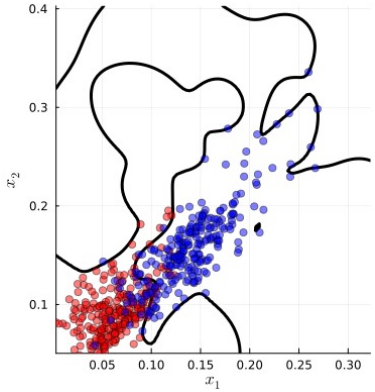
$$\boxed{w_{\mathrm{LDA},0} = -\boldsymbol{w}_{\mathrm{LDA}}^{\mathrm{T}}\boldsymbol{m}}$$

Fisher's Linear Discriminant Analysis in Pictures

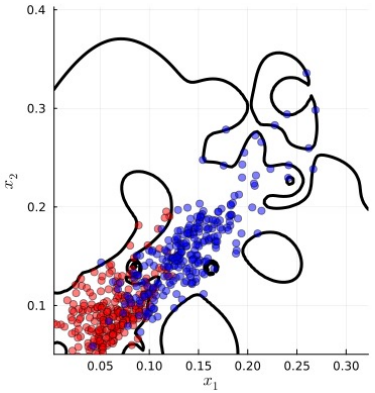
3 variances



5 variances



9 variances



Feature values of Gaussian density in 2D on a 5×5 grid at n decaying variances

Overview

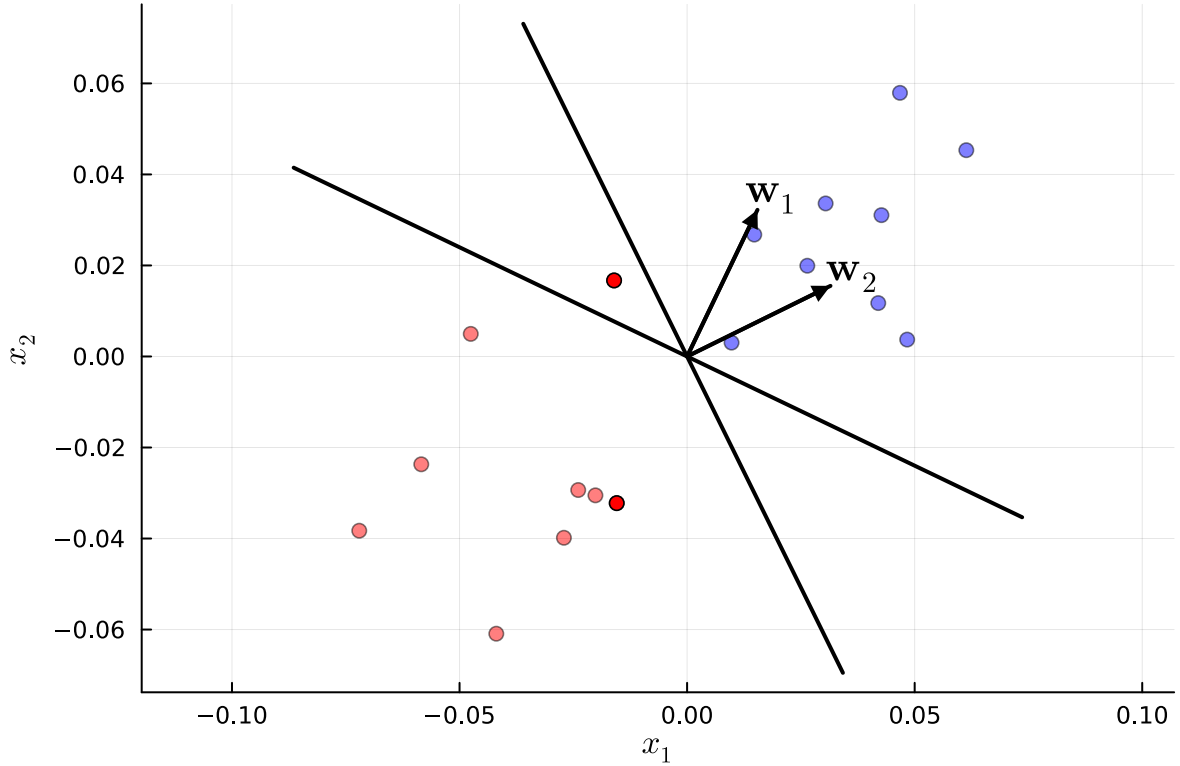
1. Classification Learning
2. Fisher's Linear Discriminant
3. **Perceptron Learning Algorithm**
4. Logistic Regression

Perceptron Learning

- **Perceptron**: First model of a classification system based on neural activity
  - Invented in 1943 by McCulloch and Pitts.
  - Implemented as a machine by Frank Rosenblatt at Cornell in 1958
  - Was intended to be a “machine”, not an algorithm (!), built on the IBM Mark 1
  - It had an array of 400 photocells
  - Weights were implemented by potentiometers; updates by electric motors
  - Only capable of learning linearly separable datasets
- **Perceptron Algorithm**: One of the simplest classification learning algorithms
  - **Given**: *A data set* $\{(x_1, y_1), \dots, (x_n, y_n)\} \subseteq (\mathcal{X} \times \{-1, 1\})$

1. **Initialize**: A weight vector $\boldsymbol{w} = \boldsymbol{0}$
2. **Iterate** through all examples and **if** $y_i \boldsymbol{w}^{\mathrm{T}} \boldsymbol{\phi}(x_i) \leq 0$ **then** $\boldsymbol{w} \leftarrow \boldsymbol{w} + y_i \boldsymbol{\phi}(x_i)$
3. **Stop** when no mistakes were made on all data

Perceptron Algorithm in Action



Novikoff's Perceptron Convergence Theorem

- **Cauchy (1821)**. *For any two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ of an inner product space we have*
$$\langle \boldsymbol{x}, \boldsymbol{y} \rangle^2 \leq \|\boldsymbol{x}\|^2 \cdot \|\boldsymbol{y}\|^2$$
- **Proof**. If $\boldsymbol{y} = \boldsymbol{0}$ then it's trivially true. Thus, choose $\boldsymbol{y} \neq \boldsymbol{0}$ and $c = -\langle \boldsymbol{x}, \boldsymbol{y} \rangle / \|\boldsymbol{y}\|^2$
$$0 \leq \|\boldsymbol{x} + c\boldsymbol{y}\|^2 = \|\boldsymbol{x}\|^2 + 2c\langle \boldsymbol{x}, \boldsymbol{y} \rangle + c^2\|\boldsymbol{y}\|^2 = \|\boldsymbol{x}\|^2 - \langle \boldsymbol{x}, \boldsymbol{y} \rangle^2 / \|\boldsymbol{y}\|^2$$
- **Novikoff (1961)**. *For any data set $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\} \subseteq (\mathbb{R}^m \times \{-1, 1\})$ with margin $\gamma = \max_{\boldsymbol{w}} \min_{i \in \{1,\ldots,n\}} \frac{y_i \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i}{\|\boldsymbol{w}\|}$ the perceptron learning algorithm converges after at most $(R/\gamma)^2$ mistake updates where $R = \max_{i \in \{1,\ldots,n\}} \|\boldsymbol{x}_i\|$.*
- **Proof**. Suppose that $\boldsymbol{w}_t$ is the solution after t mistake updates and $\boldsymbol{w}^*$ is the solution with margin γ. Then
$$\boldsymbol{w}_t^{\mathrm{T}} \boldsymbol{w}^* = (\boldsymbol{w}_{t-1} + y_i \boldsymbol{x}_i)^{\mathrm{T}} \boldsymbol{w}^* = \boldsymbol{w}_{t-1}^{\mathrm{T}} \boldsymbol{w}^* + y_i \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{w}^* \geq \boldsymbol{w}_{t-1}^{\mathrm{T}} \boldsymbol{w}^* + \gamma \cdot \|\boldsymbol{w}^*\| \geq \cdots \geq t \cdot \gamma \cdot \|\boldsymbol{w}^*\|$$
For the squared norm of the $\boldsymbol{w}_t$ we also know
$$\|\boldsymbol{w}_t\|^2 = \|\boldsymbol{w}_{t-1}\|^2 + 2y_i \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{w}_{t-1} + \|\boldsymbol{x}_i\|^2 \leq \|\boldsymbol{w}_{t-1}\|^2 + R^2 \leq \cdots \leq t \cdot R^2$$
Because of the Cauchy-Schwarz inequality we have
$$t^2 \cdot \gamma^2 \cdot \|\boldsymbol{w}^*\|^2 \leq \left(\boldsymbol{w}_t^{\mathrm{T}} \boldsymbol{w}^*\right)^2 \leq \|\boldsymbol{w}_t\|^2 \cdot \|\boldsymbol{w}^*\|^2 \leq t \cdot R^2 \cdot \|\boldsymbol{w}^*\|^2$$

Augustin-Louis Cauchy (1789 – 1857)

Overview

1. Classification Learning
2. Fisher's Linear Discriminant
3. Perceptron Learning Algorithm
4. **Logistic Regression**

Logistic Likelihood

- **Logistic Likelihood**: *Given a dataset* $D \in (\mathcal{X} \times \{0,1\})^n$ *of* n *labelled examples from input space* $\mathcal{X}$*, the basis function mapping* $\boldsymbol{\phi}: \mathcal{X} \to \mathbb{R}^M$ *and the* logistic sigmoid function $g(t) \coloneqq \frac{\exp(t)}{1+\exp(t)}$*, the* logistic likelihood *is given by*

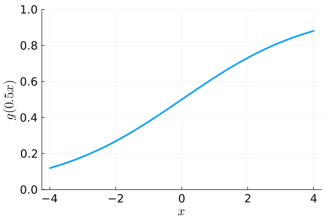
$$P(\boldsymbol{y}|X, \boldsymbol{w}) = \prod_{i=1}^{n} g\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(x_i)\right)^{y_i} \cdot \left[1 - g\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(x_i)\right)\right]^{1-y_i}$$

- Taking the **negative logarithm** results in a *cross-entropy* between the empirical class distribution and the predicted class distribution

$$-\log_2 P(\boldsymbol{y}|X, \boldsymbol{w}) = -\sum_{i=1}^{n} \left[y_i \log_2\left(g\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(x_i)\right)\right) + (1-y_i)\log_2\left(1 - g\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(x_i)\right)\right)\right]$$

- **Maximum Likelihood Approach**: Find $\boldsymbol{w}$ which minimizes $-\log_2\left(P(\boldsymbol{y}|X, \boldsymbol{w})\right)$
  - No unique solution for linearly separable data without scale constraint on $\boldsymbol{w}$ because
    - There exists many $\widetilde{\boldsymbol{w}}$ with $y_i g\left(\widetilde{\boldsymbol{w}}^{\mathrm{T}}\boldsymbol{\phi}(x_i)\right) + (1-y_i)\log_2\left(1 - g\left(\widetilde{\boldsymbol{w}}^{\mathrm{T}}\boldsymbol{\phi}(x_i)\right)\right) > 0.5$
    - Scaling $\widetilde{\boldsymbol{w}}$ makes $-\log_2\left(P(\boldsymbol{y}|X, \widetilde{\boldsymbol{w}})\right)$ strictly smaller because $\lim_{s\to\infty} g(s \cdot t) = \mathbb{I}(t > 0)$

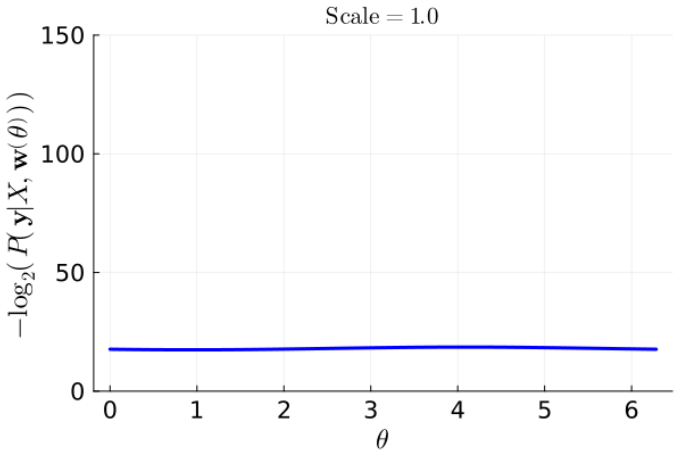
$$g(s \cdot t) = \frac{\exp(s \cdot t)}{1 + \exp(s \cdot t)}$$



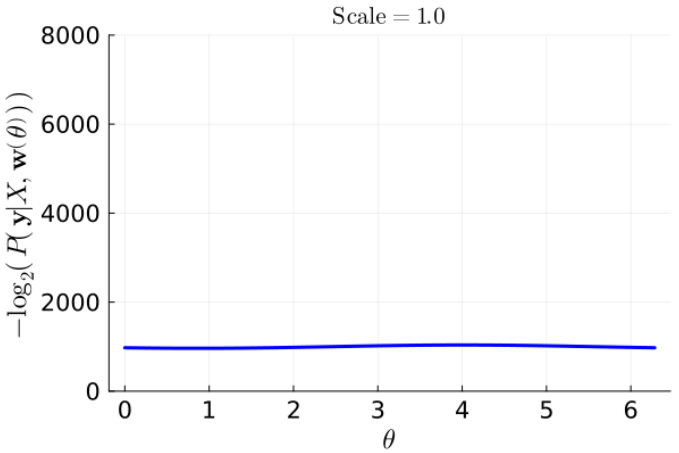
Negative Log-Logistic Likelihood in Action

$$\boldsymbol{w}(\theta) = \begin{bmatrix} s \cdot \sin(\theta) \\ s \cdot \cos(\theta) \end{bmatrix}$$

$n = 18$



$n = 1000$

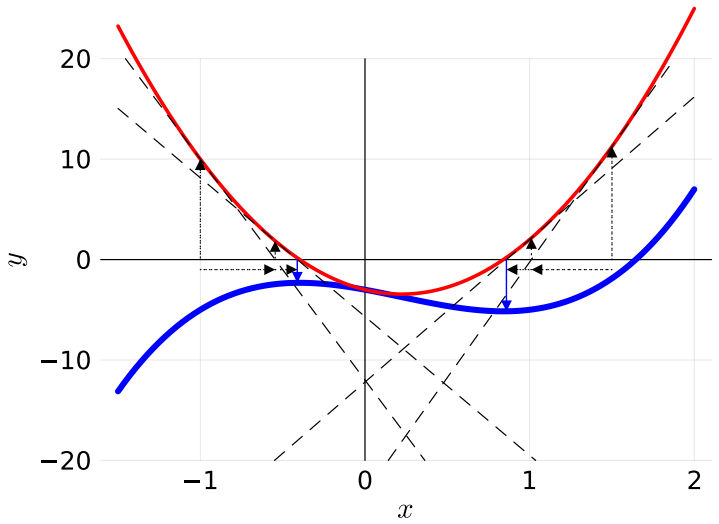


Newton-Raphson Algorithm

- **Problem**: Find the local extrema of a function $f: \mathbb{R} \to \mathbb{R}$
- **Idea**: Find the zeros of the first derivative f' of the function!
- **Newton-Raphson Algorithm**: Approximate f' at a point x_t with a linear function $g(x) = ax + b$ and find the update x_{t+1} such that $g(x_{t+1}) = 0$ via

$$a = f''(x_t)$$
$$b = f'(x_t) - f''(x_t) \cdot x_t$$

$$x_{t+1} = -\frac{b}{a} = \frac{f''(x_t) \cdot x_t - f'(x_t)}{f''(x_t)}$$



$$x_{t+1} = x_t - \frac{f'(x_t)}{f''(x_t)}$$

Sir Isaac Newton
(1643 – 1727)

Newton-Raphson in Many Dimensions

- **Problem**: Find the local extrema of a function $f: \mathbb{R}^M \to \mathbb{R}$
- **Generalized Newton-Raphson**: Start with an initial $\boldsymbol{w}_0$ and iterate

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \boldsymbol{H}^{-1}(\boldsymbol{w}_t) \cdot \frac{\partial f(\boldsymbol{w})}{\partial \boldsymbol{w}_t}, \qquad \boldsymbol{H}_{i,j}(\boldsymbol{w}) = \frac{\partial^2 f(\boldsymbol{w})}{\partial w_i \partial w_j}$$

- The matrix $\boldsymbol{H}$ is called *Hessian matrix* (first discovered by Ludwig Otto Hesse)
- **Example**: Linear regression $f(\boldsymbol{w}) = \sum_{i=1}^{n} \left(\boldsymbol{\phi}(x_i)^{\mathrm{T}} \boldsymbol{w} - y_i\right)^2$

$$\frac{\partial f(\boldsymbol{w})}{\partial \boldsymbol{w}} = 2 \cdot \sum_{i=1}^{n} \left(\boldsymbol{\phi}(x_i)^{\mathrm{T}} \boldsymbol{w} - y_i\right) \boldsymbol{\phi}(x_i)$$
$$= 2 \cdot \left(\boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{\Phi} \boldsymbol{w} - \boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{y}\right)$$
$$\boldsymbol{H}(\boldsymbol{w}) = 2 \cdot \boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{\Phi}$$

Thus, if we start with $\boldsymbol{w}_0 = \boldsymbol{0}$ and note that the Hessian does not depend on $\boldsymbol{w}$

$$\boldsymbol{w}_1 = \boldsymbol{0} - \left(\boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{\Phi}\right)^{-1} \cdot \left(\boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{\Phi} \boldsymbol{0} - \boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{y}\right) = \left(\boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{\Phi}\right)^{-1} \boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{y}$$

Ludwig Otto Hesse
(1811 – 1874)

Matrix Derivatives

$$\frac{\partial \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}}{\partial \boldsymbol{x}} = \boldsymbol{A}^{\mathrm{T}}$$

Newton-Raphson for Logistic Regression: Gradient

- **Negative log-Logistic Likelihood** $E(\boldsymbol{w})$:

$$E(\boldsymbol{w}) = -\sum_{i=1}^{n} \left[y_i \log_2 g\left(\boldsymbol{\phi}(x_i)^{\mathrm{T}} \boldsymbol{w}\right) + (1 - y_i) \log_2 \left(1 - g\left(\boldsymbol{\phi}(x_i)^{\mathrm{T}} \boldsymbol{w}\right)\right) \right]$$

- **Gradient** $\partial E(\boldsymbol{w}) / \partial \boldsymbol{w}$: Split into two cases depending on y_i
  - **Case** $y_i = 1$:

$$\frac{\partial y_i \log_2 g\left(\boldsymbol{\phi}(x_i)^{\mathrm{T}} \boldsymbol{w}\right)}{\partial \boldsymbol{w}} = \frac{g\left(\boldsymbol{\phi}(x_i)^{\mathrm{T}} \boldsymbol{w}\right) \cdot \left[1 - g\left(\boldsymbol{\phi}(x_i)^{\mathrm{T}} \boldsymbol{w}\right)\right]}{g\left(\boldsymbol{\phi}(x_i)^{\mathrm{T}} \boldsymbol{w}\right)} \cdot \boldsymbol{\phi}(x_i)$$

  - **Case** $y_i = 0$:

$$\frac{\partial y_i \log_2 \left(1 - g\left(\boldsymbol{\phi}(x_i)^{\mathrm{T}} \boldsymbol{w}\right)\right)}{\partial \boldsymbol{w}} = -\frac{g\left(\boldsymbol{\phi}(x_i)^{\mathrm{T}} \boldsymbol{w}\right) \cdot \left[1 - g\left(\boldsymbol{\phi}(x_i)^{\mathrm{T}} \boldsymbol{w}\right)\right]}{1 - g\left(\boldsymbol{\phi}(x_i)^{\mathrm{T}} \boldsymbol{w}\right)} \cdot \boldsymbol{\phi}(x_i)$$

- Putting it all **together** and noticing that we can use y_i

$$\frac{\partial E(\boldsymbol{w})}{\partial \boldsymbol{w}} = \sum_{i=1}^{n} \left(g\left(\boldsymbol{\phi}(x_i)^{\mathrm{T}} \boldsymbol{w}\right) - y_i\right) \cdot \boldsymbol{\phi}(x_i) = \boldsymbol{\Phi}^{\mathrm{T}} (\boldsymbol{g}(\boldsymbol{w}) - \boldsymbol{y})$$

Sigmoid Derivative

$$\frac{dg(t)}{dt} = g(t) \cdot [1 - g(t)]$$

Matrix Derivatives

$$\frac{\partial \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}}{\partial \boldsymbol{x}} = \boldsymbol{A}^{\mathrm{T}}$$

Newton-Raphson for Logistic Regression: Hessian

- The **Hessian** is the derivative of the first derivate

$$\frac{\partial^2 E(\boldsymbol{w})}{\partial \boldsymbol{w} \partial \boldsymbol{w}} = \frac{\partial \sum_{i=1}^{n} \left(g\left(\boldsymbol{\phi}(x_i)^{\mathrm{T}} \boldsymbol{w}\right) - y_i \right) \cdot \boldsymbol{\phi}(x_i)}{\partial \boldsymbol{w}} = \frac{\partial \sum_{i=1}^{n} g\left(\boldsymbol{\phi}(x_i)^{\mathrm{T}} \boldsymbol{w}\right) \cdot \boldsymbol{\phi}(x_i)}{\partial \boldsymbol{w}}$$

- Using **chain rule of differentiation** again, we have

$$\frac{\partial^2 E(\boldsymbol{w})}{\partial \boldsymbol{w} \partial \boldsymbol{w}} = \sum_{i=1}^{n} \frac{\partial g\left(\boldsymbol{\phi}(x_i)^{\mathrm{T}} \boldsymbol{w}\right)}{\partial \boldsymbol{w}} \cdot \boldsymbol{\phi}^{\mathrm{T}}(x_i)$$

$$\frac{\partial^2 E(\boldsymbol{w})}{\partial \boldsymbol{w} \partial \boldsymbol{w}} = \sum_{i=1}^{n} g_i(\boldsymbol{w}) \cdot [1 - g_i(\boldsymbol{w})] \cdot \boldsymbol{\phi}(x_i) \boldsymbol{\phi}^{\mathrm{T}}(x_i)$$

$$\frac{\partial^2 E(\boldsymbol{w})}{\partial \boldsymbol{w} \partial \boldsymbol{w}} = \underbrace{[\boldsymbol{\phi}(x_1) \quad \cdots \quad \boldsymbol{\phi}(x_n)]}_{\boldsymbol{\Phi}^{\mathrm{T}}} \underbrace{\begin{bmatrix} g_1(\boldsymbol{w}) \cdot [1 - g_1(\boldsymbol{w})] & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & g_n(\boldsymbol{w}) \cdot [1 - g_n(\boldsymbol{w})] \end{bmatrix}}_{\boldsymbol{R}(\boldsymbol{w})} \underbrace{\begin{bmatrix} \boldsymbol{\phi}^{\mathrm{T}}(x_1) \\ \vdots \\ \boldsymbol{\phi}^{\mathrm{T}}(x_n) \end{bmatrix}}_{\boldsymbol{\Phi}}$$

- Putting it all **together**, we have

$$\frac{\partial^2 E(\boldsymbol{w})}{\partial \boldsymbol{w} \partial \boldsymbol{w}} = \boldsymbol{\Phi}^{\mathrm{T}} \cdot \boldsymbol{R}(\boldsymbol{w}) \cdot \boldsymbol{\Phi}$$

Sigmoid Derivative

$$\frac{dg(t)}{dt} = g(t) \cdot [1 - g(t)]$$

Matrix Derivatives

$$\frac{\partial \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}}{\partial \boldsymbol{x}} = \boldsymbol{A}^{\mathrm{T}}$$

Logistic Regression: Iterative Reweighted Least Square

- **Given**: A data set $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\} \subseteq (\mathbb{R}^M \times \{0,1\})$ and a basis function mapping $\boldsymbol{\phi}: \mathcal{X} \to \mathbb{R}^M$

1. **Initialize**: A weight vector $\boldsymbol{w} = \boldsymbol{0}$ and compute $\boldsymbol{\Phi} = \left[\phi_j(x_i)\right] \in \mathbb{R}^{n \times M}$
2. **Iterate** until convergence
   - Compute $\boldsymbol{t} = \boldsymbol{\Phi w}$ and $g_i = \frac{\exp(t_i)}{1+\exp(t_i)}$ and $\boldsymbol{R} = \operatorname{diag}\left(\begin{bmatrix} g_1(1-g_1) \\ \vdots \\ g_n(1-g_n) \end{bmatrix}\right)$
   - Update $\boldsymbol{w} \leftarrow \boldsymbol{w} - \left(\boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{R} \boldsymbol{\Phi}\right)^{-1} \boldsymbol{\Phi}^{\mathrm{T}} (\boldsymbol{g} - \boldsymbol{y})$

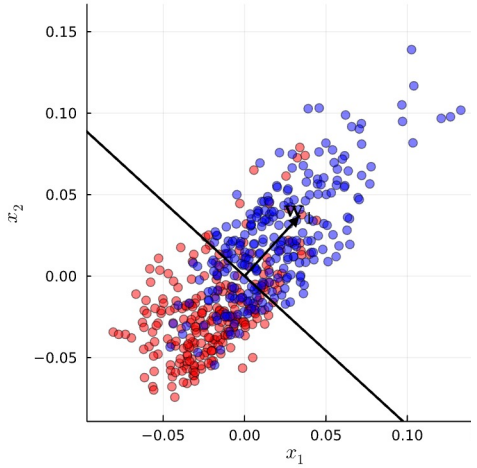
- **Rewriting the update equation** by using $\boldsymbol{z} = \boldsymbol{\Phi w} - \boldsymbol{R}^{-1}(\boldsymbol{g} - \boldsymbol{y})$

$$\left(\boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{R} \boldsymbol{\Phi}\right)^{-1} [\boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{R} \boldsymbol{\Phi w} - \boldsymbol{\Phi}^{\mathrm{T}} (\boldsymbol{g} - \boldsymbol{y})] = \left(\boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{R} \boldsymbol{\Phi}\right)^{-1} \boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{R} \boldsymbol{z}$$

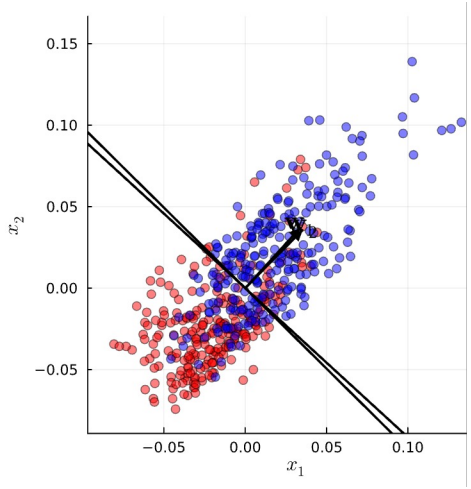
the algorithm is equivalent to least-square with a per-iteration reweighting $\boldsymbol{R}$

Logistic Regression in Action

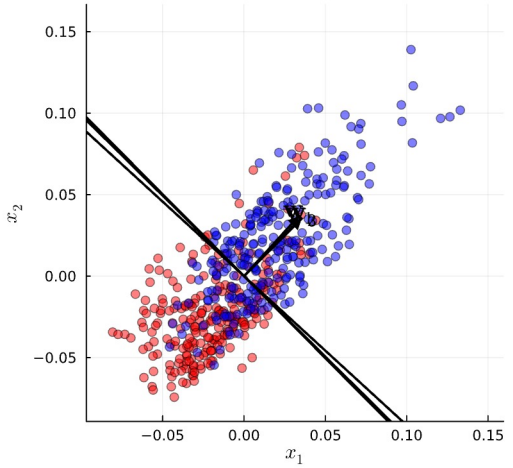
Iteration 1



Iteration 2



Iteration 3



Summary

1. **Classification**
   - The likelihood model is no longer conjugate to the prior over the weight vector
   - A linear classifier has an interpretable geometry in high-dimensional spaces
2. **Maximum Likelihood: Fisher's Linear Discriminant**
   - Rooted in the idea of finding a projection direction where inter-class distances are maximized while intra-class distances are minimized
   - Relatively simple and efficient to compute!
3. **Perceptron Learning Algorithm**
   - *The* first classification learning algorithm which is still very relevant and performant!
   - Elegant geometric convergence proof which bounds the number of update steps
4. **Logistic Regression**
   - Assumes a logistic likelihood model (rather than a probit)
   - Iterative optimization procedure using gradient descent

HPI Hasso Plattner Institut
Digital Engineering · Universität Potsdam

See you next week!