

# The perceptron algorithm

Christos Dimitrakakis

October 1, 2024

# Outline

## The Perceptron

- Introduction

- The algorithm

## Gradient methods

- Gradients for optimisation

- The perceptron as a gradient algorithm

## Lab and Assignment

# Topic

## The Perceptron

- Introduction

- The algorithm

## Gradient methods

- Gradients for optimisation

- The perceptron as a gradient algorithm

## Lab and Assignment

## Guessing gender from height

- ▶ Feature space  $\mathcal{X} \subset \mathbb{R}$ : e.g. height
- ▶ Label space  $\mathcal{Y} = \{-1, 1\}$ : e.g. gender
- ▶ Can we find some  $\beta_1 \in \mathbb{R}$  and a direction  $\beta_0 \in \{-1, +1\}$  so as to separate the genders?

## Online learning: At time $t$

- ▶ We choose a separator  $\beta_0^t, \beta_1^t$
- ▶ We observe a new datapoint  $x_t, y_t$
- ▶ We make a mistake at time  $t$  if:

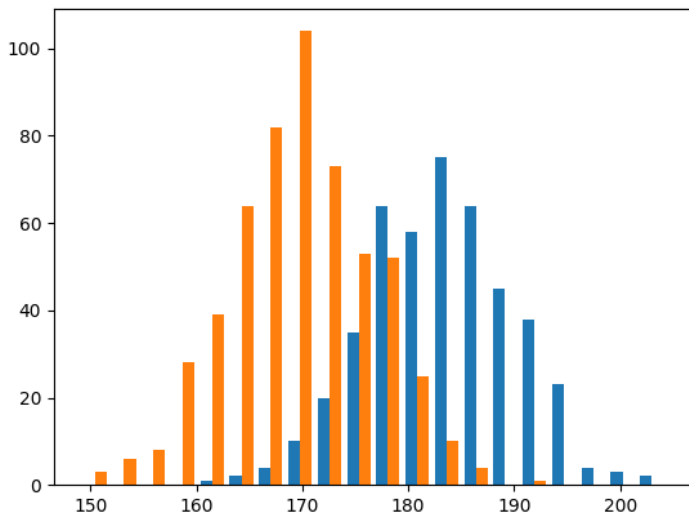
$$\beta^t x_t - \beta_0^t \leq 0.$$

- ▶ If we stop making mistakes, then we are classifying everything perfectly.

## Can you find a threshold that makes a small number of mistakes?

`./src/Perceptron/perceptron_simple.py`

# Non-separable classes



## More complex example

- ▶ Feature space  $\mathcal{X} \subset \mathbb{R}^n$ : e.g. height and weight for  $n = 2$
- ▶ Label space  $\mathcal{Y} = \{-1, 1\}$ : e.g. gender
- ▶ Can we find some line so as to separate the genders?

`./src/Perceptron/show_class_data_labels.py`

## More complex example

- ▶ Feature space  $\mathcal{X} \subset \mathbb{R}^n$ : e.g. height and weight for  $n = 2$
- ▶ Label space  $\mathcal{Y} = \{-1, 1\}$ : e.g. gender
- ▶ Can we find some line so as to separate the genders?

`./src/Perceptron/show_class_data_labels.py`

## Linear separator

$$f(x) = \beta_0 + \beta^\top x = \beta_0 + \sum_{i=1}^n \beta_i x_i.$$

## More complex example

- ▶ Feature space  $\mathcal{X} \subset \mathbb{R}^n$ : e.g. height and weight for  $n = 2$
- ▶ Label space  $\mathcal{Y} = \{-1, 1\}$ : e.g. gender
- ▶ Can we find some line so as to separate the genders?

`./src/Perceptron/show_class_data_labels.py`

## Linear separator

$$f(x) = \beta_0 + \beta^\top x = \beta_0 + \sum_{i=1}^n \beta_i x_i.$$

If we augment  $x$  an additional component  $x_0 = 1$ , we can write

$$f(x) = \beta^\top x = \sum_{i=0}^n \beta_i x_i.$$



# The perceptron algorithm

## Input

- ▶ Feature space  $X \subset \mathbb{R}^n$ .
- ▶ Label space  $Y = \{-1, 1\}$ .
- ▶ Data  $(x_t, y_t)$ ,  $t \in [T]$ , with  $x_t \in X, y_t \in Y$ .

## Algorithm

- ▶  $w_1 = w_0$ .
- ▶ For  $t = 1, \dots, T$ .
  - ▶  $a_t = \text{sgn}(w_t^\top x_t)$ .
  - ▶ If  $a_t \neq y_t$ 
    - ▶  $w_{t+1} = w_t + y_t x_t$
  - ▶ Else
    - ▶  $w_{t+1} = w_t$
  - ▶ EndIf
- ▶ Return  $w_{T+1}$

# Perceptron examples

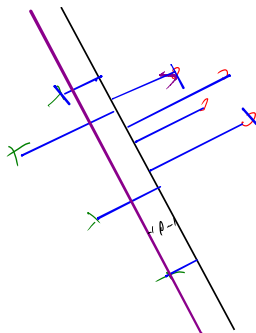
## Example 1: One-dimensional data

- ▶ Done on the board
- ▶ Shows how the algorithm works.
- ▶ Demonstrates the idea of a margin

## Example 2: Two-dimensional data

- ▶ See in-class programming exercise

# Margins and the perceptron theorem



- ▶ The **hyperplane**  $\beta^*$  separates the examples
- ▶ The **margin**  $\rho$  is the minimum distance  $\rho$  between  $\beta^*$  and any point.

## Theorem (Perceptron theorem)

The number of mistakes is bounded by  $\rho^{-2}$ , where  $\|x_t\| \leq 1$ ,  $\rho \leq y_t(x_t^\top \beta^*)$  for some **margin**  $\rho$  and **hyperplane**  $\beta^*$  with  $\|\beta^*\| = 1$ .

# Topic

## The Perceptron

Introduction

The algorithm

## Gradient methods

Gradients for optimisation

The perceptron as a gradient algorithm

## Lab and Assignment

# The gradient descent method: one dimension

- ▶ Function to minimise  $f : \mathbb{R} \rightarrow \mathbb{R}$ .
- ▶ Derivative  $\frac{d}{d\beta} f(\beta)$

## Gradient descent algorithm

- ▶ Input: initial value  $\beta^0$ , **learning rate** schedule  $\alpha_t$
- ▶ For  $t = 1, \dots, T$ 
  - ▶  $\beta^{t+1} = \beta^t - \alpha_t \frac{d}{d\beta} f(\beta^t)$
- ▶ Return  $\beta^T$

## Properties

- ▶ If  $\sum_t \alpha_t = \infty$  and  $\sum_t \alpha_t^2 < \infty$ , it finds a local minimum  $\beta^T$ , i.e. there is  $\epsilon > 0$  so that

$$f(\beta^T) < f(\beta), \forall \beta : \|\beta^T - \beta\| < \epsilon.$$

# Gradient methods for expected value

## Estimate the expected value

$x_t \sim P$  with  $\mathbb{E}_P[x_t] = \mu$ .

# Gradient methods for expected value

Estimate the expected value

$x_t \sim P$  with  $\mathbb{E}_P[x_t] = \mu$ .

Objective: mean squared error

Here  $\ell(x, \beta) = (x - \beta)^2$ .

$$\min_{\beta} \mathbb{E}_P[(x_t - \beta)^2].$$

# Gradient methods for expected value

## Estimate the expected value

$x_t \sim P$  with  $\mathbb{E}_P[x_t] = \mu$ .

## Objective: mean squared error

Here  $\ell(x, \beta) = (x - \beta)^2$ .

$$\min_{\beta} \mathbb{E}_P[(x_t - \beta)^2].$$

## Derivative

Idea: at the minimum the derivative should be zero.

$$d/d\beta \mathbb{E}_P[(x_t - \beta)^2] = \mathbb{E}_P[d/d\beta (x_t - \beta)^2] = \mathbb{E}_P[-(x_t - \beta)] = \mathbb{E}_P[x_t] - \beta.$$

Setting the derivative to 0, we have  $\beta = \mathbb{E}_P[x_t]$ . This is a simple solution.

## Real-world setting

- ▶ The objective function does not result in a simple solution
- ▶ The distribution  $P$  is not known.
- ▶ We can sample  $x \sim P$ .



# The gradient method

- ▶ Function to minimise  $f : \mathbb{R}^n \rightarrow \mathbb{R}$ .
- ▶ Derivative  $\nabla_{\beta} f(\beta) = \left( \frac{\partial f(\beta)}{\partial \beta_1}, \dots, \frac{\partial f(\beta)}{\partial \beta_n} \right)$ , where  $\frac{\partial f}{\partial \beta_n}$  denotes the **partial** derivative, i.e. varying one argument and keeping the others fixed.

## Gradient descent algorithm

- ▶ Input: initial value  $\beta^0$ , learning rate schedule  $\alpha_t$
- ▶ For  $t = 1, \dots, T$ 
  - ▶  $\beta^{t+1} = \beta^t - \alpha_t \nabla_{\beta} f(\beta^t)$
- ▶ Return  $\beta^T$

## Properties

- ▶ If  $\sum_t \alpha_t = \infty$  and  $\sum_t \alpha_t^2 < \infty$ , it finds a local minimum  $\beta^T$ , i.e. there is  $\epsilon > 0$  so that

$$f(\beta^T) < f(\beta), \forall \beta : \|\beta^T - \beta\| < \epsilon.$$

## Stochastic gradient method

This is the same as the gradient method, but with added noise:

- ▶  $\beta^{t+1} = \beta^t - \alpha_t [\nabla_{\beta} f(\beta^t) + \omega_t]$
- ▶  $\mathbb{E}[\omega_t] = 0$  is sufficient for convergence.

### Example: When the cost is an expectation

In machine learning, the cost is frequently an expectation of some function  $\ell$ ,

$$f(\beta) = \int_{\mathcal{X}} dP(x) \ell(x, \beta)$$

This can be approximated with a sample

$$f(\beta) \approx \frac{1}{T} \sum_t \ell(x_t, \beta)$$

The same holds for the gradient:

$$\nabla_{\beta} f(\beta) = \int_{\mathcal{X}} dP(x) \nabla_{\beta} \ell(x, \beta) \approx \frac{1}{T} \sum_t \nabla_{\beta} \ell(x_t, \beta)$$

# Stochastic gradient for mean estimation

- If we sample  $x$  we approximate the gradient:

$$\frac{d}{d\beta} \mathbb{E}_P[(x - \beta)^2] \approx \frac{1}{T} \sum_{t=1}^T \frac{d}{d\beta} (x_t - \beta)^2 = \frac{1}{T} \sum_{t=1}^T 2(x_t - \beta)$$

# Stochastic gradient for mean estimation

- If we sample  $x$  we approximate the gradient:

$$\frac{d}{d\beta} \mathbb{E}_P[(x - \beta)^2] \approx \frac{1}{T} \sum_{t=1}^T \frac{d}{d\beta} (x_t - \beta)^2 = \frac{1}{T} \sum_{t=1}^T 2(x_t - \beta)$$

- If we update  $\beta$  after each new sample  $x_t$ , we obtain:

$$\beta^{t+1} = \beta^t + 2\alpha_t(x_t - \beta^t)$$

# Perceptron algorithm as gradient descent

## Target error function

$$\mathbb{E}_{\mathbf{P}}^{\beta}[\ell] = \int_{\mathcal{X}} d\mathbf{P}(x) \sum_y \mathbf{P}(y|x) \ell(x, y, \beta)$$

Minimises the error on the true distribution.

# Perceptron algorithm as gradient descent

## Target error function

$$\mathbb{E}_{\mathbf{P}}^{\beta}[\ell] = \int_{\mathcal{X}} d\mathbf{P}(x) \sum_y \mathbf{P}(y|x) \ell(x, y, \beta)$$

Minimises the error on the true distribution.

## Empirical error function

$$\mathbb{E}_{\mathbf{D}}^{\beta}[\ell] = \frac{1}{T} \sum_{t=1}^T \ell(x_t, y_t, \beta), \quad \mathbf{D} = (x_t, y_t)_{t=1}^T, \quad x_t, y_t \sim P.$$

Minimises the error on the empirical distribution.

# Cost functions and the chain rule

## Perceptron cost function

The cost of each example

$$\ell(x, y, \beta) = \overbrace{\mathbb{I}\{y(x^\top \beta) < 0\}}^{\text{misclassified?}} \overbrace{[-y(x^\top \beta)]}^{\text{margin of error}} \quad (1)$$

where the **indicator function**  $\mathbb{I}\{A\}$  is 1 when  $A$  is true and 0 otherwise.

# Cost functions and the chain rule

## Perceptron cost function

The cost of each example

$$\ell(x, y, \beta) = \overbrace{\mathbb{I}\{y(x^\top \beta) < 0\}}^{\text{misclassified?}} \overbrace{[-y(x^\top \beta)]}^{\text{margin of error}} \quad (1)$$

where the **indicator function**  $\mathbb{I}\{A\}$  is 1 when  $A$  is true and 0 otherwise.

## Reminder: The chain rule

Let  $z = g(y)$ ,  $y = f(x)$  so that  $z = g(f(x))$ . Then  $\frac{dz}{dx} = \frac{dz}{dy} \frac{dy}{dx}$



# Cost functions and the chain rule

## Perceptron cost function

The cost of each example

$$\ell(x, y, \beta) = \overbrace{\mathbb{I}\{y(x^\top \beta) < 0\}}^{\text{misclassified?}} \overbrace{[-y(x^\top \beta)]}^{\text{margin of error}} \quad (1)$$

where the **indicator function**  $\mathbb{I}\{A\}$  is 1 when  $A$  is true and 0 otherwise.

## Reminder: The chain rule

Let  $z = g(y)$ ,  $y = f(x)$  so that  $z = g(f(x))$ . Then  $\frac{dz}{dx} = \frac{dz}{dy} \frac{dy}{dx}$

## Derivative: Chain rule

$$\blacktriangleright \nabla_\beta \ell(x, y, \beta) = -\mathbb{I}\{y(x^\top \beta) < 0\} \nabla_\beta [y(x^\top \beta)].$$

# Cost functions and the chain rule

## Perceptron cost function

The cost of each example

$$\ell(x, y, \beta) = \overbrace{\mathbb{I}\{y(x^\top \beta) < 0\}}^{\text{misclassified?}} \overbrace{[-y(x^\top \beta)]}^{\text{margin of error}} \quad (1)$$

where the **indicator function**  $\mathbb{I}\{A\}$  is 1 when  $A$  is true and 0 otherwise.

## Reminder: The chain rule

Let  $z = g(y)$ ,  $y = f(x)$  so that  $z = g(f(x))$ . Then  $\frac{dz}{dx} = \frac{dz}{dy} \frac{dy}{dx}$

## Derivative: Chain rule

- ▶  $\nabla_\beta \ell(x, y, \beta) = -\mathbb{I}\{y(x^\top \beta) < 0\} \nabla_\beta [y(x^\top \beta)]$ .
- ▶  $\frac{\partial \beta}{\partial \beta_i} [y(x_t^\top \beta)] = y x_{t,i}$  (gradient of Perceptron's output)

# Cost functions and the chain rule

## Perceptron cost function

The cost of each example

$$\ell(x, y, \beta) = \overbrace{\mathbb{I}\{y(x^\top \beta) < 0\}}^{\text{misclassified?}} \overbrace{[-y(x^\top \beta)]}^{\text{margin of error}} \quad (1)$$

where the **indicator function**  $\mathbb{I}\{A\}$  is 1 when  $A$  is true and 0 otherwise.

## Reminder: The chain rule

Let  $z = g(y)$ ,  $y = f(x)$  so that  $z = g(f(x))$ . Then  $\frac{dz}{dx} = \frac{dz}{dy} \frac{dy}{dx}$

## Derivative: Chain rule

- ▶  $\nabla_\beta \ell(x, y, \beta) = -\mathbb{I}\{y(x^\top \beta) < 0\} \nabla_\beta [y(x^\top \beta)]$ .
- ▶  $\frac{\partial \beta}{\partial \beta_i} [y(x_t^\top \beta)] = y x_{t,i}$  (gradient of Perceptron's output)
- ▶ Gradient update:  $\beta^{t+1} = \beta^t - \nabla_\beta \ell(x, y, \beta) = \beta^t + y x_t$

# Cost functions and the chain rule

## Perceptron cost function

The cost of each example

$$\ell(x, y, \beta) = \overbrace{\mathbb{I}\{y(x^\top \beta) < 0\}}^{\text{misclassified?}} \overbrace{[-y(x^\top \beta)]}^{\text{margin of error}} \quad (1)$$

where the **indicator function**  $\mathbb{I}\{A\}$  is 1 when  $A$  is true and 0 otherwise.

## Reminder: The chain rule

Let  $z = g(y)$ ,  $y = f(x)$  so that  $z = g(f(x))$ . Then  $\frac{dz}{dx} = \frac{dz}{dy} \frac{dy}{dx}$

## Derivative: Chain rule

- ▶  $\nabla_\beta \ell(x, y, \beta) = -\mathbb{I}\{y(x^\top \beta) < 0\} \nabla_\beta [y(x^\top \beta)]$ .
- ▶  $\frac{\partial \beta}{\partial \beta_i} [y(x_t^\top \beta)] = y x_{t,i}$  (gradient of Perceptron's output)
- ▶ Gradient update:  $\beta^{t+1} = \beta^t - \nabla_\beta \ell(x, y, \beta) = \beta^t + y x_t$

# Cost functions and the chain rule

## Perceptron cost function

The cost of each example

$$\ell(x, y, \beta) = \overbrace{\mathbb{I}\{y(x^\top \beta) < 0\}}^{\text{misclassified?}} \overbrace{[-y(x^\top \beta)]}^{\text{margin of error}} \quad (1)$$

where the **indicator function**  $\mathbb{I}\{A\}$  is 1 when  $A$  is true and 0 otherwise.

## Reminder: The chain rule

Let  $z = g(y)$ ,  $y = f(x)$  so that  $z = g(f(x))$ . Then  $\frac{dz}{dx} = \frac{dz}{dy} \frac{dy}{dx}$

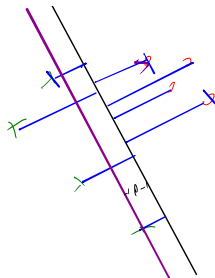
## Derivative: Chain rule

- ▶  $\nabla_\beta \ell(x, y, \beta) = -\mathbb{I}\{y(x^\top \beta) < 0\} \nabla_\beta [y(x^\top \beta)]$ .
- ▶  $\frac{\partial \beta}{\partial \beta_i} [y(x_t^\top \beta)] = y x_{t,i}$  (gradient of Perceptron's output)
- ▶ Gradient update:  $\beta^{t+1} = \beta^t - \nabla_\beta \ell(x, y, \beta) = \beta^t + y x_t$

The classification error cost function is **not** differentiable :(

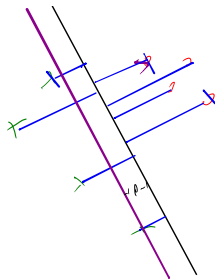
# Margins and confidences

We can think of the output of the network as a measure of confidence



# Margins and confidences

We can think of the output of the network as a measure of confidence



By applying the **logit** function, we can bound a real number  $x$  to  $[0, 1]$ :

$$f(x) = \frac{e^x}{1 + e^x} = \frac{1}{1 + e^{-x}}$$

# Logistic regression

Output as a measure of confidence, given the parameter  $\beta$

$$P_{\beta}(y = 1|x) = \frac{1}{1 + \exp(-x_t^{\top} \beta)}$$

The original output  $x_t^{\top} \beta$  is now passed through the logit function.



# Logistic regression

Output as a measure of confidence, given the parameter  $\beta$

$$P_{\beta}(y = 1|x) = \frac{1}{1 + \exp(-x_t^{\top} \beta)}$$

The original output  $x_t^{\top} \beta$  is now passed through the logit function.

## Negative Log likelihood

$$\ell(x_t, y_t, \beta) = -\ln P_{\beta}(y_t|x_t) = \ln(1 + \exp(-y_t x_t^{\top} \beta))$$

$$\begin{aligned} \nabla_{\beta} \ell(x_t, y_t, \beta) &= \frac{1}{1 + \exp(-y_t x_t^{\top} \beta)} \nabla_{\beta} [1 + \exp(-y_t x_t^{\top} \beta)] \\ &= \frac{1}{1 + \exp(-y_t x_t^{\top} \beta)} \exp(-y_t x_t^{\top} \beta) [\nabla_{\beta} (-y_t x_t^{\top} \beta)] \\ &= -\frac{1}{1 + \exp(x_t^{\top} \beta)} (x_t)_i^n e \end{aligned}$$

$$\blacktriangleright \mathbb{E}_P(\ell) = \int_X dP(x) \sum_{y \in Y} P(y|x) P_{\beta}(y_t + x_t)$$

# Topic

## The Perceptron

Introduction

The algorithm

## Gradient methods

Gradients for optimisation

The perceptron as a gradient algorithm

## Lab and Assignment

# Lab demonstration

- ▶ How to use kNN and LogisticRegression with sklearn (and perhaps statsmodels, time permitting)
- ▶ Use an example where there is no default 'class' label

# Assignment

1. In the class data, find one categorical variable of interest that we want to predict.
2. Formulate the appropriate classification problem.
3. Perform model selection through train/validate or crossvalidation to find the best model (kNN or perceptron) and hyperparameters (k for the kNN)
4. Discuss anything of interest in the data such as: feature scaling/selection, missing data, outliers.
5. We cannot independently measure the quality of the model, as we have no test set. What can we do?