TP 1 : Estimators – Stochastic gradient descent

Exercise 1 : Comparing different estimators for the uniform model (10pt)

We consider the parametric family of uniform laws $\{\mathcal{U}(\theta), \theta \in \mathbb{R}\}$ on $\mathbb{R}$ where $\mathcal{U}(\theta)$ admits the following density with regard to the Lebesgue's measure :

$$p_\theta(x) = \frac{1}{\theta}\mathbb{1}_{[0,\theta]} \,.$$

We consider independent random samples $X_1, X_2, \dots X_n$ of density p_θ. The goal of this exercise is to compare different estimators for the parameter θ.

1. Calculate $\mathbb{E}_\theta(X_1)$ and deduce an estimator $\hat{\theta}_1$ of θ using the method of moments. (1pt)
2. Calculate the quadratic risk of $\hat{\theta}_1$. (3pt)
3. Calculate the maximum likelihood estimator $\hat{\theta}_2$. (2pt)
4. Calculate the quadratic risk of $\hat{\theta}_2$. (3pt)
5. Which estimator is preferable ? (1 pt)

Exercise 2 : Box-Muller and Marsaglia-Bray algorithm (Bonus +3pt)

Let R a random variable with Rayleigh distribution with parameter 1 and Θ with uniform distribution on $[0, 2\pi]$. We also assume that R and Θ are independent. The density of R with regard to the Lebesgue's measure writes :

$$\forall r \in \mathbb{R}, \qquad f_R(r) = r\exp\left(-\frac{r^2}{2}\right)\mathbb{1}_{\mathbb{R}^+}(r)$$

1. Let X and Y such that

$$X = R\cos(\Theta) \qquad \text{and} \qquad Y = R\sin(\Theta)\,.$$

Prove that both X and Y have $\mathcal{N}(0, 1)$ distribution and are independent.

2. Write an algorithm for sampling independent Gaussian distribution $\mathcal{N}(0, 1)$.

3. Consider the Marsaglia-Bray algorithm given below.

a) What is the distribution of (V_1, V_2) at the end of the "while" loop ?
b) What is the expected number of steps in the "while" loop ?

Algorithm 1: Marsaglia-Bray algorithm

```
1 $V_1 = 1, V_2 = 1$ while $V_1^2 + V_2^2 > 1$ do
2     Sample $U_1, U_2$ independant r.v. with distribution $\mathcal{U}([0,1])$ ;
3     Set $V_1 = 2U_1 - 1$ and $V_2 = 2U_2 - 1$.
4 end
5 Set $S = \sqrt{-2\log(V_1^2 + V_2^2)}$ ;
6 Set $X = S\frac{V_1}{\sqrt{V_1^2+V_2^2}}$ and $Y = S\frac{V_2}{\sqrt{V_1^2+V_2^2}}$ ;
7 return $(X, Y)$.
```

c) Set

$$T_1 = \frac{V_1}{\sqrt{V_1^2 + V_2^2}}, \qquad \text{and} \qquad V = V_1^2 + V_2^2\,.$$

Show that T_1 and V are independent, $V \sim \mathcal{U}([0,1])$ and T_1 has the same distribution as $\cos(\Theta)$ with $\Theta \sim \mathcal{U}([0, 2\pi])$.

d) What is the distribution of the output (X, Y) ?

Exercise 3 : Learning a linear classifier, [Bot91, BCN16] (10 pt)

We consider the supervised learning problem of assigning inputs $x \in \mathbb{R}^d$ to one of two categories labeled by $y \in \{-1, +1\}$. The statistical relationship between inputs and labels is governed by an unknown joint distribution $\mathbb{P}(x, y)$. If this distribution were known, classification could be optimally performed by computing the conditional probability $\mathbb{P}(y \mid x)$ and applying the Bayes decision rule.

In practice, learning refers to the process of inferring this decision mechanism from a finite collection of observations. This dataset is composed of n labeled samples :

$$\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\},$$

where each $x_i \in \mathbb{R}^d$ denotes a feature vector, and $y_i \in \{-1, +1\}$ indicates its corresponding class label. The goal is to construct a classification model $h(x)$ that performs well not just on these training examples but also on unseen data — that is, it should generalize.

In this exercise, we only consider **linear classifiers**, that is functions of the form

$$h(x; w, \tau) = \langle w, x \rangle + \tau,$$

where $w \in \mathbb{R}^d$ is a weight vector and $\tau \in \mathbb{R}$ is a bias term. This model attempts to separate the two classes using a hyperplane in the feature space. The bias τ can be adjusted to favor either precision ($\mathbb{P}[y = 1 \mid h(x) = 1]$) or recall ($\mathbb{P}[h(x) = 1 \mid y = 1]$) depending on application needs.

Prediction is done via the sign of $h(x)$, with the predicted label being $\text{sign}(h(x))$. However, optimizing such a model directly using classification accuracy is computationally challenging due to the non-differentiability of the sign function. To overcome this, a **continuous loss function** is introduced to approximate the classification objective and facilitate optimization.

The Adaline Model

One early and influential model is the **Adaline** (Adaptive Linear Neuron), proposed by Widrow and Hoff in 1960. It employs a linear prediction function :

$$h(x; w, \tau) = \langle w, x \rangle + \tau,$$

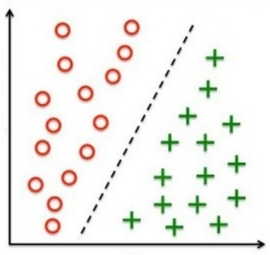
For convenience, we can consider the extended feature vector $\bar{x} = (x, 1)$ in order to regroup τ, w in one parameter $\bar{w}$. Then $h(\bar{x}, \bar{w}) = \langle \bar{w}, \bar{x} \rangle$. The Adaline model evaluates the classifier's performance using the **mean squared error (MSE)** :

$$L(w) = \mathbb{E}_{(x,y)\sim\mathbb{P}} \left[(y - h(\bar{x}, \bar{w}))^2 \right].$$

However, since $\mathbb{P}$ is unknown, we cannot compute this quantity exactly. Instead, we approximate it using the available dataset. This leads to the **empirical risk**, defined as :

$$\boxed{L_n(w) = \frac{1}{n} \sum_{i=1}^{n} (y_i - h(\bar{x}_i, \bar{w}))^2.}$$

Minimizing $L_n(w)$ provides an estimate of the true minimizer of $L(w)$, under the assumption that the training data is representative of the underlying distribution.



1. (Python) Generate a set of observations $(\{(x_i, y_i)\}_{i=1}^{N}$ by sampling random points x_i in $\mathbb{R}^2$ and choosing an arbitrary normal vector w^* to define the separating hyperplane of the two classes. The sign of $\langle w^*, x_i \rangle$ gives the label $y_i \in \{-1, +1\}$. (2pt)
2. Write an algorithm for minimizing $L_n(w)$ with Stochastic Gradient Descent.(2pt)
3. (Python) Implement and test the algorithm on the set of observations generated at the first question. What is the vector $\hat{w}$ estimated ? Is it far from w^* ? (2pt)

4. (Python) Noise your observations $\{z_i\}_{i=1}^n$ with an additive Gaussian noise and perform the optimisation again. Compare with the result of question three.(1pt)

5. (Python) Test the algorithm on the *Heart Disease (Diagnostic) Data Set* [WSM95] : `https://archive.ics.uci.edu/dataset/45/heart+disease`. (3pt)

Références

[BCN16] Léon Bottou, Frank E. Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *eprint arXiv :1606.04838*, 2016.

[Bot91] Léon Bottou. Stochastic gradient learning in neural networks. In *Neuro-Nîmes 91*, 1991.

[WSM95] William H. Wolberg, W. Nick Street, and Olvi L. Mangasarian. UCI machine learning repository, 1995.