ML Theory TP1 Perceptron :

Moussa JAMOR

Anas NOURI

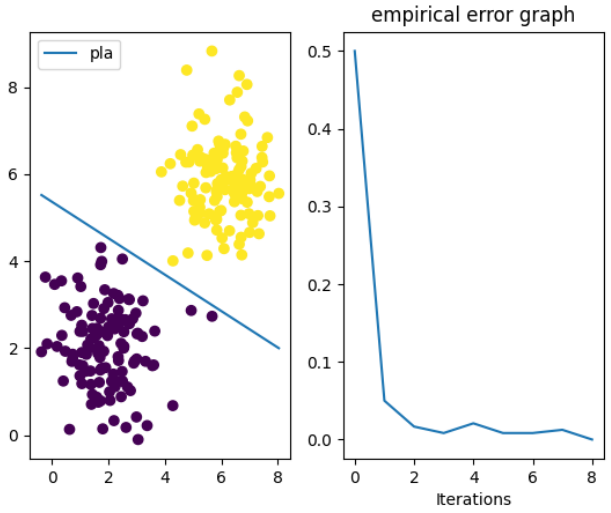
▼ Part1 :

💡 **How Run Code:**

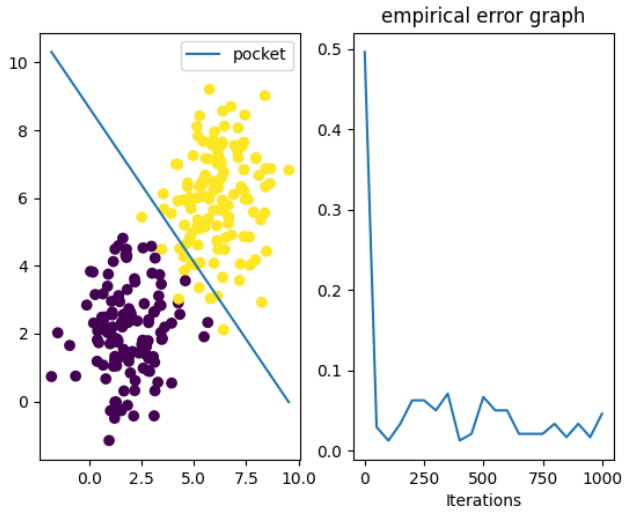
1. Each algorithm exist in separate file with his name (like “pocket.py”) you should just run the file and will display graph have the line, and also the empirical loss evolution.

2. There is two additional files “data.py” and “plot.py” this two files, we are defined some functions using only **numpy** and **matplotlib** to generate data, and plot in appropriate way.

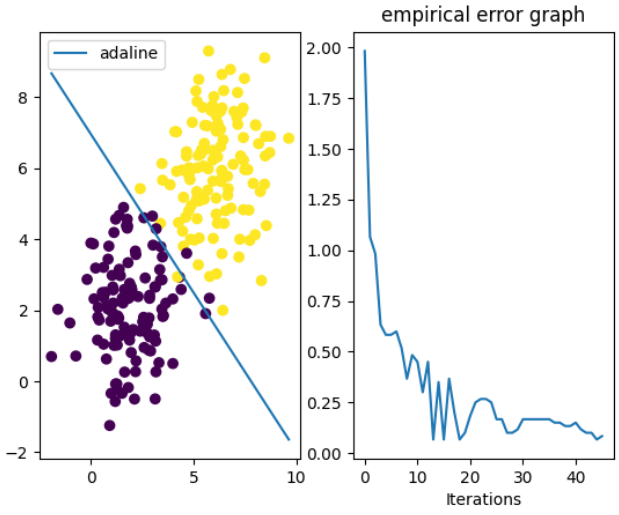
▼ PLA :



▼ **Pocket :**



▼ **Adaline :**



▼ algorithms Benchmarking :

Algorithm	Number of iterations	loss function
PLA	8	0.0
Pocket	1000	0.04583
Adaline	40	0.08333

💡 **Conclusions:**

- When data is linearly separable the best algorithms is **PLA**, it's converge very quickly. **but if the data is not linearly separable the PLA never converge**.
- By compare the Pocket method and Adaline in terms of iterations we see that Adaline make loss * 2 that Pocket method but in 40 <<< 1000. The problem in Pocket algorithm is the T_{max} parameter. The model can provide less error but if the number of iteration not yet acheived T_{max} value the algorithm keep going.

▼ Part 2 :

Find All types of loss function for classification

▼ Binary Cross Entropy/log loss

$$\text{Log Loss} = -\frac{1}{N}\sum_{i=1}^{N} y_i \log \hat{y}_i + (1-y_i)\log(1-\hat{y}_i)$$

- **Advantage**
  - A cost function is a differential.
- **Disadvantage**
  - Multiple local minima
  - Not intuitive

▼ **Hinge loss**

A loss function used for training classifiers. The hinge loss is used for "maximum-margin" classification, most notably for support vector machine (SVMs).

For an intended output *t* = ±1 and a classifier score *y*, the hinge loss of the prediction *y* is defined as

$$\ell(y) = \max(0, 1 - t \cdot y)$$

▼ **Square loss(MSE: Mean Square Error)**

If a vector of **n** predictions is generated from a sample of **n** data points on all variables, and **Y** is the vector of observed values of the variable being predicted, with **Ŷ**

being the predicted values (e.g. as from a least-squares fit), then the within-sample MSE of the predictor is computed as

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n} \left(Y_i - \hat{Y}_i\right)^2.$$

▼ **Focal Loss**

Focal loss focuses on the examples that the model gets wrong rather than the ones that it can confidently predict, ensuring that predictions on hard examples improve over time rather than becoming overly confident with easy ones.

How exactly is How exactly is this done?

Focal loss achieves this through something called

▼ **Down Weighting**

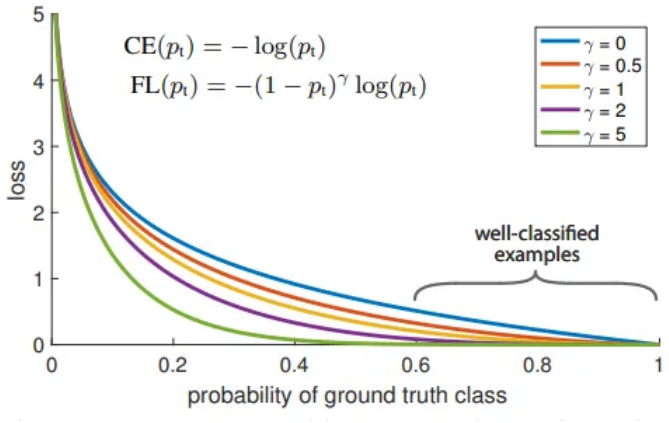
Down weighting is a technique that reduces the influence of easy examples on the loss function, resulting in more attention being paid to hard examples. This technique can be implemented by adding a modulating factor to the Cross-Entropy loss.

$$FocalLoss = -\sum_{i=1}^{i=n}(i - p_i)^{\gamma} log_b(p_i)$$

Where γ (Gamma)Where γ (Gamma) is the

focusing parameter

to be tuned using cross-validation. The image below shows how Focal Loss behaves for different values of γ.



▼ **Logistic loss**

To obtain the optimal weights that maximize the classification accuracy, an exponential loss can be used. With the given ith instance xi and its class label yi ∈ {−1, 1}, the exponential loss L(xi, yi) is defined as an exponential form of the classification margin, as shown in (9).

$$L(xi, yi) = e^{-y \cdot f(xi)}$$

With a training set of n instances, the total exponential loss is given as a summation of (9). One of the proposed weighting methods, namely exponential naïve Bayes (ENB), is defined in (10), which is the solution for minimizing the total exponential loss. Once we find the optimal weights, we can use (8) to classify new instances.

$$w_{ENB} = \arg\min_{w} \sum_{i=1}^{n} \exp\left(-y_i(P_0 + wP_{x_i})\right)$$

- **Find all types of activation functions**

1. **Sigmoid Activation Function:**
   - Formula:
   $$f(x) = \frac{1}{1 + e^{-x}}$$
   - Output Range: (0, 1)
   - Used in the output layer for binary classification problems.
2. **Hyperbolic Tangent (tanh) Activation Function:**
   - Formula:
   $$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$
   - Output Range: (-1, 1)
   - Similar to sigmoid but with a range from -1 to 1.
3. **Rectified Linear Unit (ReLU) Activation Function:**
   - Formula:
   $$f(x) = \max(0, x)$$
   - Output Range: [0, +∞)
   - Widely used in hidden layers due to its simplicity and effectiveness.
4. **Leaky ReLU Activation Function:**
   - Formula:
   $$f(x) = \begin{cases} x, & \text{if } x \geq 0 \\ \alpha x, & \text{otherwise} \end{cases}$$
   - Output Range: (-∞, +∞)
   - Addresses the "dying ReLU" problem by allowing small negative values.
5. **Parametric ReLU (PReLU) Activation Function:**
   - Formula:

$$f(x) = \begin{cases} x, & \text{if } x \geq 0 \\ \alpha x, & \text{otherwise} \end{cases}$$

- Output Range: (-∞, +∞)
- Similar to Leaky ReLU but with the leak factor learned during training.

6. **Exponential Linear Unit (ELU) Activation Function:**
   - Formula:

$$f(x) = \begin{cases} x, & \text{if } x \geq 0 \\ \alpha(e^x - 1), & \text{otherwise} \end{cases}$$

   - Output Range: (-α, +∞)
   - Combines benefits of ReLU and sigmoid, with smoother transitions.
7. **Swish Activation Function:**
   - Formula:

$$f(x) = x \cdot \sigma(\beta x)$$

   - Output Range: (0, +∞)
   - Introduced as an alternative to ReLU, depends on a trainable parameter \(\beta\).
8. **Scaled Exponential Linear Unit (SELU) Activation Function:**
   - Formula:

$$f(x) = \lambda \cdot \begin{cases} x, & \text{if } x > 0 \\ \alpha(e^x - 1), & \text{otherwise} \end{cases}$$

   - Output Range: (-αλ, +∞)
   - Introduced as a self-normalizing activation function.
9. **Softmax Activation Function:**
   - Formula (for multi-class classification):

$$f(x)_i = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}}$$

- Outputs a probability distribution over multiple classes.

10. **Linear Activation Function:**
    - Formula:

$$f(x) = x$$

    - Used in the output layer for regression problems.

These are some of the most commonly used activation functions in neural networks. The choice of activation function depends on the specific problem, the architecture of your neural network, and considerations like avoiding vanishing gradients or training stability.)

- **Discussion for Pocket**

The learning rate and iteration choice are pretty arbitrary. This is essentially unavoidable in the
context of training neural networks with gradient descent methods. Nowadays, there are several

tricks" that can be applied to search for parameters like the learning rate α more efficiently, but the
problem of searching for those kinds of parameters persists

- **Demonstrate that if the data is L.S then the PLA converges**

Theorem

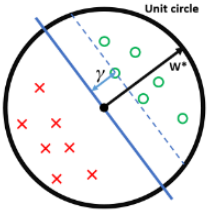
The Perceptron was arguably the first algorithm with a strong formal guarantee. If a data set is linearly separable, the Perceptron will find a separating hyperplane in a finite number of updates. (If the data is not linearly separable, it will loop forever.)

The argument goes as follows: Suppose $\exists \mathbf{w}^*$ such that $y_i(\mathbf{x}^\top \mathbf{w}^*) > 0 \; \forall (\mathbf{x}_i, y_i) \in D$.

Now, suppose that we rescale each data point and the $\mathbf{w}^*$ such that

$$||\mathbf{w}^*|| = 1 \qquad \text{and} \qquad ||\mathbf{x}_i|| \leq 1 \;\; \forall \mathbf{x}_i \in D$$

Let us define the Margin γ of the hyperplane $\mathbf{w}^*$ as $\gamma = \min_{(\mathbf{x}_i, y_i) \in D} |\mathbf{x}_i^\top \mathbf{w}^*|$.



To summarize our setup:

- All inputs $\mathbf{x}_i$ live within the unit sphere
- There exists a separating hyperplane defined by $\mathbf{w}^*$, with $\|\mathbf{w}\|^* = 1$ (i.e. $\mathbf{w}^*$ lies exactly on the unit sphere).
- γ is the distance from this hyperplane (blue) to the closest data point.

Theorem: If all of the above holds, then the Perceptron algorithm makes at most $1/\gamma^2$ mistakes.

Proof:
Keeping what we defined above, consider the effect of an update ($\mathbf{w}$ becomes $\mathbf{w} + y\mathbf{x}$) on the two terms $\mathbf{w}^\top \mathbf{w}^*$ and $\mathbf{w}^\top \mathbf{w}$. We will use two facts:

- $y(\mathbf{x}^\top \mathbf{w}) \leq 0$: This holds because $\mathbf{x}$ is misclassified by $\mathbf{w}$ - otherwise we wouldn't make the update.
- $y(\mathbf{x}^\top \mathbf{w}^*) > 0$: This holds because $\mathbf{w}^*$ is a separating hyper-plane and classifies all points correctly.

1. Consider the effect of an update on $\mathbf{w}^\top \mathbf{w}^*$:

$$(\mathbf{w} + y\mathbf{x})^\top \mathbf{w}^* = \mathbf{w}^\top \mathbf{w}^* + y(\mathbf{x}^\top \mathbf{w}^*) \geq \mathbf{w}^\top \mathbf{w}^* + \gamma$$

   The inequality follows from the fact that, for $\mathbf{w}^*$, the distance from the hyperplane defined by $\mathbf{w}^*$ to $\mathbf{x}$ must be at least γ (i.e. $y(\mathbf{x}^\top \mathbf{w}^*) = |\mathbf{x}^\top \mathbf{w}^*| \geq \gamma$).

   This means that for each update, $\mathbf{w}^\top \mathbf{w}^*$ grows by **at least** γ.

2. Consider the effect of an update on $\mathbf{w}^\top \mathbf{w}$:

$$(\mathbf{w} + y\mathbf{x})^\top (\mathbf{w} + y\mathbf{x}) = \mathbf{w}^\top \mathbf{w} + \underbrace{2y(\mathbf{w}^\top \mathbf{x})}_{<0} + \underbrace{y^2(\mathbf{x}^\top \mathbf{x})}_{0 \leq \; \leq 1} \leq \mathbf{w}^\top \mathbf{w} + 1$$

   The inequality follows from the fact that
   - $2y(\mathbf{w}^\top \mathbf{x}) < 0$ as we had to make an update, meaning $\mathbf{x}$ was misclassified
   - $0 \leq y^2(\mathbf{x}^\top \mathbf{x}) \leq 1$ as $y^2 = 1$ and all $\mathbf{x}^\top \mathbf{x} \leq 1$ (because $\|\mathbf{x}\| \leq 1$).

   This means that for each update, $\mathbf{w}^\top \mathbf{w}$ grows by **at most** 1.

3. Now we know that after M updates the following two inequalities must hold:
   (1) $\mathbf{w}^\top \mathbf{w}^* \geq M\gamma$
   (2) $\mathbf{w}^\top \mathbf{w} \leq M$.
   We can then complete the proof:

$$\begin{aligned}
M\gamma &\leq \mathbf{w}^\top \mathbf{w}^* && \text{By (1)} \\
&= \|\mathbf{w}\| \cos(\theta) && \text{by definition of inner-product, where } \theta \text{ is the angle between } \mathbf{w} \text{ and } \mathbf{w}^* \\
&\leq ||\mathbf{w}|| && \text{by definition of cos, we must have } \cos(\theta) \leq 1 \\
&= \sqrt{\mathbf{w}^\top \mathbf{w}} && \text{by definition of } \|\mathbf{w}\| \\
&\leq \sqrt{M} && \text{By (2)}
\end{aligned}$$

$$\Rightarrow M\gamma \leq \sqrt{M}$$

$\Rightarrow M^2\gamma^2 \leq M$

$\Rightarrow M \leq \frac{1}{\gamma^2}$ And hence, the number of updates M is bounded from above by a constant.