Prof. Dr. Volker Roth
volker.roth@unibas.ch

Vitali Nesterov
vitali.nesterov@unibas.ch

Department of Mathematics and Computer Science
Spiegelgasse 1
4051 Basel

Exercise 3

Due date: **Wednesday, March 25th 2020**

3.1: Perceptron Convergence Theorem

Recall the Fixed-Increment Single Sample Perceptron and the Perceptron Algorithm for finding the decision boundary given labelled training points. Assume d-dimensional observations $(x_1, x_2, \ldots, x_d) \in \mathbb{R}^d$ and adopt the following notation for a single data point $\boldsymbol{x} = (1, x_1, x_2, \ldots, x_d)^\top \in \mathbb{R}^{d+1}$ and model parameters $\boldsymbol{w} = (w_0, w_1, w_2, \ldots, w_d)^\top \in \mathbb{R}^{d+1}$, where w_0 is considered as a bias. The class labels for each data point $\boldsymbol{x_i}$ are denoted with $C_i \in \{+1, -1\}$, such that a labeled dataset is given by $\mathcal{D} = ((\boldsymbol{x_1}, C_1), \ldots, (\boldsymbol{x_n}, C_n))$.

Definitions

- The **linear discriminant function** is given by

$$f(\boldsymbol{x}; \boldsymbol{w}) \triangleq \boldsymbol{w}^\top \boldsymbol{x} = w_0 + (w_1, w_2, \ldots, w_d)^\top (x_1, x_2, \ldots, x_d).$$

- The **decision boundary function** is

$$f(\boldsymbol{x}; \boldsymbol{w}) \overset{!}{=} 0.$$

- The **classification hypothesis** for a binary classification task is defined as

$$h(\boldsymbol{x}; \boldsymbol{w}) \triangleq \begin{cases} +1, & \text{if } \boldsymbol{w}^\top \boldsymbol{x} > 0 \\ -1, & \text{if } \boldsymbol{w}^\top \boldsymbol{x} < 0 \end{cases}$$

  A class is associated with a corresponding label $+1$ or -1. The classification hypothesis can be understood as a threshold function which is a signum of $f(\boldsymbol{x}; \boldsymbol{w})$. If a class is correctly predicted, then the criterion $C_i \boldsymbol{w}^\top \boldsymbol{x_i} > 0$ for a data point $\boldsymbol{x_i}$ and the corresponding label C_i is always satisfied.

- The **misclassification error** is

$$J(\boldsymbol{w}) \triangleq -C_i \boldsymbol{w}^\top \boldsymbol{x_i},$$

  which is always positive for each misclassified training point $\boldsymbol{x_i}$. Hence, the corresponding gradient of the error criterion is $\nabla_{\boldsymbol{w}} J(\boldsymbol{w}) = -C_i \boldsymbol{x_i}$.

- The **perceptron learning rule** is given by

$$w_j \leftarrow w_j + \eta C_i x_{ij},$$

where the increment follows from the gradient $-\eta \nabla_{\boldsymbol{w}} J(\boldsymbol{w}) = \eta C_i \boldsymbol{x_i}$ and $\eta \in \mathbb{R}$ is a learning rate with $0 < \eta \leq 1$.

Algorithm 1: Perceptron Algorithm

```
Input  : Labelled training points $(\boldsymbol{x_1}, C_1), \ldots, (\boldsymbol{x_n}, C_n)$ with data $\boldsymbol{x_i} \in \mathbb{R}^{d+1}$ and
         class labels $C_i \in \{+1, -1\}$, initial model parameters $\boldsymbol{w} \in \mathbb{R}^{d+1}$, and a
         learning rate $\eta \in \mathbb{R}$, such that $0 < \eta \leq 1$
Output : Model parameters $\boldsymbol{w}$
$converged \leftarrow false$;
while $converged \neq true$ do
    $converged \leftarrow true$;
    if $C_i \boldsymbol{w}^\top \boldsymbol{x_i} < 0$ then
        $converged \leftarrow false$;
        $w_j \leftarrow w_j + \eta C_i x_{ij}$;
    else
        no-op;
    end
end
```

Exercise

Prove the Perceptron Convergence Theorem for a linearly separable case. If the observations are linearly separable, i.e., there exists $\boldsymbol{w}^*$ such that $h(\boldsymbol{x_i}; \boldsymbol{w}^*) = C_i$ for all $\boldsymbol{x} \in \mathcal{D}$, then the Perceptron Convergence Theorem holds. The Algorithm 1 will terminate at a solution vector.

Assumptions

- The learning rate η is set to $\eta = 1$.
- The initial model parameters are set to $\boldsymbol{w} = \boldsymbol{0}$.
- The training data $\boldsymbol{x} \in \mathcal{D}$ and the solution vector $\boldsymbol{w}^*$ are normalized, such that

$$\max_{\boldsymbol{x} \in \mathcal{D}} ||\boldsymbol{x}|| = ||\boldsymbol{w}^*|| = 1.$$

- Define the margin γ as the smallest inner product between the solution vector $\boldsymbol{w}^*$ and some data point $\boldsymbol{x} \in \mathcal{D}$:

$$\gamma \triangleq \min_{\boldsymbol{x} \in \mathcal{D}} |(\boldsymbol{w}^*)^\top \boldsymbol{x}| \leq 1.$$

Hints

Recall the geometric interpretation of the inner product. Given $\langle \boldsymbol{u}, \boldsymbol{v} \rangle = \boldsymbol{u}^\top \boldsymbol{v}$ the inner product is a scalar, such that the length of a vector $\boldsymbol{v'}$ is multiplied by the length of a vector $\boldsymbol{u}$, where $\boldsymbol{v'}$ is a projection of $\boldsymbol{v}$ on $\boldsymbol{u}$. If $\boldsymbol{u}$ and $\boldsymbol{v}$ are orthogonal, then $\boldsymbol{u}^\top \boldsymbol{v} = 0$, otherwise the more $\boldsymbol{v}$ points in the same direction as $\boldsymbol{u}$, the larger the inner product gets.

The convergence of the Perceptron Algorithm can be shown by proving that after each update, $\boldsymbol{w}$ gets closer to the solution vector $\boldsymbol{w^*}$. Note that after each update $\boldsymbol{w}$ rotates toward $\boldsymbol{w^*}$, but also grows in length.

Use Cauchy-Schwarz inequality to show that after n updates the following expression holds:

$$n\gamma \leq |\boldsymbol{w}^\top \boldsymbol{w^*}| \leq ||\boldsymbol{w}|| \cdot ||\boldsymbol{w^*}||.$$

Show that the amount of training updates is finite and is upper bounded by

$$n \leq \frac{1}{\gamma^2}.$$

3.2: Python implementation

In the above derived results, the convergence of the Perceptron Algorithm is proved. Hence, a solution for each separable case can be found. Write a Python program which implements the perceptron algorithm (see Algorithm 1) to find a solution vector for two separable classes.

- Recall the notation of $\boldsymbol{x} \in \mathbb{R}^{d+1}$. Generate a 2-dimensional Gaussian distributed data points and add an additional offset value for the bias. Assign the corresponding class labels and ensure that the observation points are linearly separable by choosing proper distribution parameters.
- Implement the linear discriminant function in order to compute the misclassification error.
- Implement the perceptron learning rule which updates the model parameters $\boldsymbol{w}$.
- Find the corresponding polynomial for the decision boundary

$$\boldsymbol{w}^\top \boldsymbol{x} \overset{!}{=} 0.$$

- Plot the sampled data and the decision boundary.

Can you notice any problems with the solution vector $\boldsymbol{w}^*$ found by the algorithm?