Linear Data Analysis
Supervised Learning - Perceptron Rule

Cain Susko

Queen's University
School of Computing

March 28, 2022

a Perceptron Model of Neurons

this lecture will explore the perceptron algorithm. A perceptron is a binary, artificial representation of a neuron.

given an augmented weight vector $\hat{w}$, the:

- Augmented data 'vector' is $\hat{x}_i$
- Label for $\hat{x}_i$ (y_i) is either 0 or 1
- linear weighted sum is $u_i = \hat{x}_i^\top \hat{w}$
- $\hat{x}_i$ predicted in class 1 if and only if $u_i \geq 0$
- Output is q_i which is 0 or 1 (binary function)

The objective of the Perceptron Algorithm is to find a hyperplane that puts all the data with label 1 in the positive half-space and data with label 0 in the negative half-space.

Perceptron: Basic Algorithm

The Preceptron was first created by Rosenblatt in 1959 and is the earliest known example of machine learning. Rosenblatt's key assumption was: **to iteratively update vector $\hat{w}$, use only *mis*-classified data**.

His Key result from this technique was that: **if the training 'vectors' $\hat{x}_i$ are linearly separable, then the perceptron rule will converge**.

b Deriving the Perceptron Rule

given:

Hyperplane	$\hat{w}$
Observation	$\hat{x}_i$
Label	$y_i \in \{0, 1\}$

Suppose the logic of the Perceptron Rule is as follows:

$$(y_i = 1) \wedge (q_i = 1)$$

The corresponding action is thus:

$$\hat{w} \leftarrow \hat{w}$$

There are 4 cases which can be caused:

Logic	Action	Case
$(y_i = 1) \wedge (q_i = 1)$	$\hat{w} \leftarrow \hat{w}$	TP
$(y_i = 0) \wedge (q_i = 0)$	$\hat{w} \leftarrow \hat{w}$	TN
$(y_i = 1) \wedge (q_i = 0)$	$\hat{w} \leftarrow \hat{w} + \hat{x}_i$	TP
$(y_i = 0) \wedge (q_i = 1)$	$\hat{w} \leftarrow \hat{w} - \hat{x}_i$	FP

Note: the error for data-point i is equal to:

$$e_i =^{def} y_i - q_i$$

such that the respective errors for the above table are:

Error
$e_i = 0$
$e_i = 0$
$e_i = 1$
$e_i = -1$

Thus, the Perceptron Rule for iterating through all data-points i is:

$$\hat{w}_k \leftarrow \hat{w}_{k-1} + e_i \hat{x}_i$$

c Pseudocode For The Perceptron Rule

This section will explore the Perceptron Rule in pseudocode.

```
missed <- true; // makes sure we enter while loop
while (missed):
        missed <- false;
        for each training sample i:
                // quantization
                q_i <- heaviside(x'_i * w);
                e_i <- (y_i - q_i);
                // correction by residial (error)
                w <- w + e_i * x_i;
                // checks if repeat is needed
                if (e_i != 0):
                        missed <- true;
                fi
        rof
elihw
```

For Linearly separable data, the Perceptron Rule:

- is Guaranteed to converge
- can be initialized with a random weight vector
- has the complexity of $O(m)$ for m data-vectors

For general data the Perceptron Rule:

- can find a local optimum
- may not converge but can oscillate locally
- has no requirement for ‘good’ separation.

Note: this process finds *a* hyperplane, rather than the *best* $\mathbb{H}$

Batch Learning

For m observations, the augmented design matrix is: $\hat{X} \in \mathbb{R}^{m\times(n+1)}$ where the calculation per data vector is:

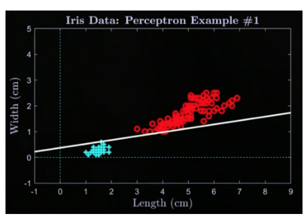
$$\hat{w}_k = \hat{w}_{k-1} + \hat{x}_i(y_i - q_i)$$

But if one were to gather the labels into $\vec{y}$ and the output into $\vec{q}$, the formula for batch iteration of the Perceptron Rule is:

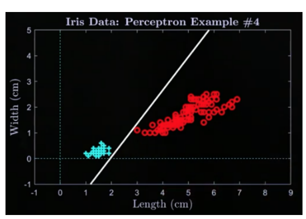
$$\hat{w}_k = \hat{w}_{k-1} + \hat{X}^\top(\vec{y} - \vec{q})$$

d Perceptron Rule for Iris Data

Given the Iris data for the length data and width of 2 classes of plants and a random start vector $\hat{w}$, the Perceptron Rule creates the following hyperplane:



From differet starting vectors, we get different hyperplanes as the Perceptron Rule only finds a **local** optimum.



Because of this lack of certainty for the fit of these hyperplanes, we will have to introduce a factor of optimization to the Rule.

Perceptron Summary

- we use the augmented weight vector $\hat{w}$
- it is simple, fast computation
- linear algebra is fundamental to this rule
- it also extends to multi-class problems

There are some limitations to the Perceptron Rule, however:

- peculiar convergence to hyperplane
- data must be linearly separable

Learning Outcomes

Students should now be able to:

- transform data to augmented form
- implement basic perceptron algorithm
- test algorithm on simple data