PRS L10

Linear Classifiers and the Perceptron Algorithm

Linear Classifiers

- the goal of classification is to classify items that have similar features values into a single class
- linear classifier achieves this goal via a discriminant function that is the linear combination of the features

- **Binary classification**
- Image -> Compute feature vector x of size n -> Compute g(x) -> Threshold function -> {c1, c2}

$$g(x) = wx^T + w_0 = \sum_{i=1}^{n} w_i x_i + w_0$$

Linear Classifiers

- Image -> Compute feature vector **x** of size n -> **Compute g(x)** -> Threshold function -> Class {c1, c2}

$$g(x) = wx^T + w_0 = \sum_{i=1}^{n} w_i x_i + w_0$$

- w is a learnable parameter named **weight**
- wo is a learnable parameter named **bias**

Linear Classifiers

- Image -> Compute feature vector **x** of size n -> Compute **g(x)** -> Threshold function -> Class {c1, c2}

$$g(x) = wx^T + w_0 = \sum_{i=1}^{n} w_i x_i + w_0$$

- **Threshold function**
- if g(x)>0 decide that the image belongs to class +1
- if g(x) < 0 decide that the image belongs to class -1
- If g(x)=0 the image can belong to either class

Linear Classifiers

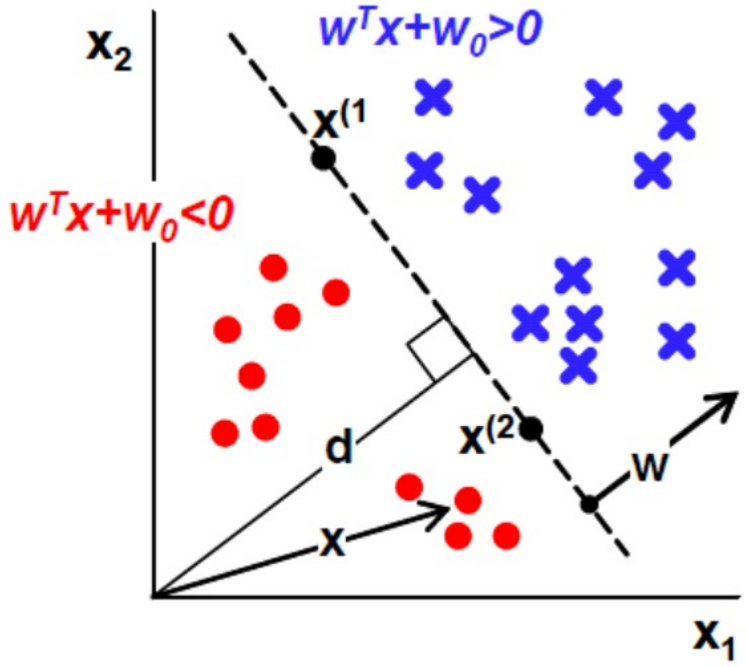


Figure 2. Image for 2D case depicting: linear decision regions (red and blue), decision boundary (dashed line), weight vector (w) and bias (w_0=d).

Learning w and wo

- How to find w and wo?
  - Optimization problem
  - Minimize the loss function with gradient descent
- Let's denote the vector [w0 w] with $\overline{w}$
- Let's denote $\begin{bmatrix} 1 \\ x^T \end{bmatrix}$ as $\overline{x}$
- There are n images in the training set
- Compute the loss function L ; y is the ground truth

$$L(\overline{\boldsymbol{w}}) = \frac{1}{n}\sum_{i=1}^{n} max(0, -y_i \overline{\boldsymbol{w}} \cdot \overline{\boldsymbol{x}}_i^T) = \frac{1}{n}\sum_{i=1}^{n} L_i\,(\overline{\boldsymbol{w}})$$

- Find the minimum of the loss function L using gradient descent

Learning w and wo

- Compute the loss function L, y is the ground truth

$$L(\bar{\boldsymbol{w}}) = \frac{1}{n}\sum_{i=1}^{n} max(0, -y_i \bar{\boldsymbol{w}} \cdot \bar{\boldsymbol{x}}_i^T) = \frac{1}{n}\sum_{i=1}^{n} L_i\,(\bar{\boldsymbol{w}})$$

- Find the minimum of the loss function L using gradient descent
- **Gradient descent idea**: a differentiable function decreases fastest in the opposite direction of the gradient

$$\bar{\boldsymbol{w}}_{k+1} \leftarrow \bar{\boldsymbol{w}}_k - \eta \boldsymbol{\nabla L}(\bar{\boldsymbol{w}}_k)$$

- The gradient of the loss function L:

$$\boldsymbol{\nabla L}(\bar{\boldsymbol{w}}) = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{\nabla L_i}(\bar{\boldsymbol{w}})$$

$$\boldsymbol{\nabla L_i}(\bar{\boldsymbol{w}}) = \begin{cases} 0, & if\ y_i \bar{\boldsymbol{w}} \cdot \bar{\boldsymbol{x}}_i^T > 0 \\ -y_i \bar{\boldsymbol{x}}_i, & otherwise \end{cases}$$

Learning w and wo with gradient descent

Batch perceptron:

- update the weights after visiting all the training examples

Online perceptron:

- update the weights after visiting each training example

Algorithm:
Batch Perceptron

```
init w, η, E_limit, max_iter
for iter=1:max_iter
  E = 0, L = 0
  ∇L = [0,0,0]
  for i=1:n
    z_i = Σ_{j=0}^{d} w_j X_ij
    if z_i · y_i ≤ 0
      ∇L ← ∇L − y_i X_i
      E ← E + 1
      L ← L − y_i z_i
    endif
  endfor
  E ← E/n
  L ← L/n
  ∇L ← ∇L /n
  if E < E_limit
    break
  w ← w − η∇L
endfor
```

Algorithm:
Online Perceptron

```
init w, η, E_limit, max_iter
for iter=1:max_iter
  E = 0
  for i=1:n
    z_i = Σ_{j=0}^{d} w_j X_ij
    if z_i · y_i ≤ 0
      w ← w + η X_i y_i
      E ← E + 1
    endif
  endfor
  E ← E/n
  if E < E_limit
    break
endfor
```

Practical Work

In this laboratory session we will find a linear classifier that discriminates between two sets of points. The points in class 1 are colored in red and the points in class 2 are colored in blue.

Each point is described by the color (that denotes the class label) and the two coordinates, x_1 and x_2.
The augmented weight vector will have the form $\bar{w} = [w_0\ w_1\ w_2]$.
The augmented feature vector will be $\bar{x} = [1\ \ x_1\ \ x_2\ ]$.

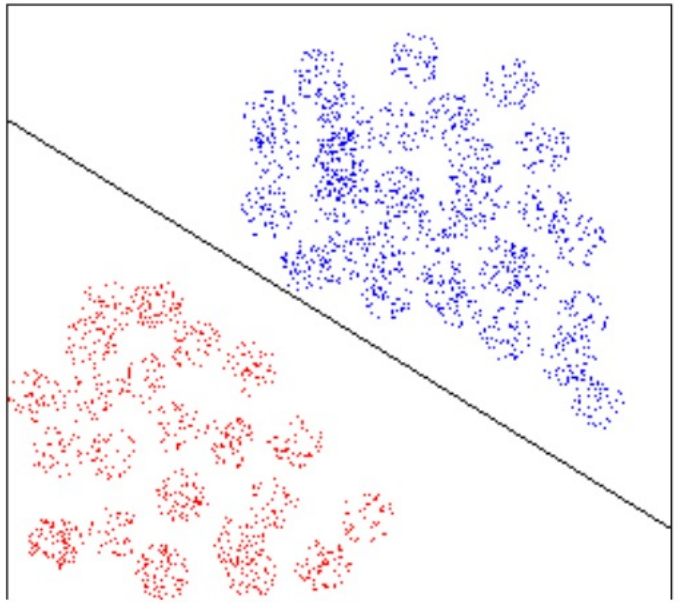
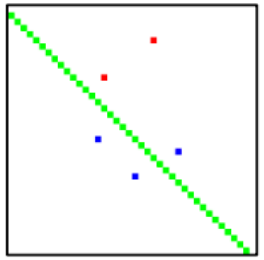


Figure 3. The decision boundary obtained from the perceptron algorithm

Practical Work

Consider the point from the file points00 as (x,y) pairs or (column, row):

- Red points: (23, 5), (15,11) – class +1
- Blue points: (14, 21), (27,23), (20, 27) – class -1

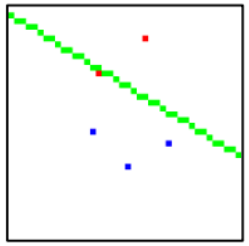


Learning rate = 0.01

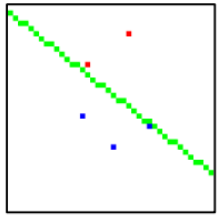
w=[1.000000 1.000000 -1.000000]
max iter = 10
$E_{limit}=10^{-5}$

Practical Work

Iteration 0
i=0: w=[1.000000 1.000000 -1.000000] xi=[1 23 5] yi = 1 zi=19.000000
i=1: w=[1.000000 1.000000 -1.000000] xi=[1 15 11] yi = 1 zi=5.000000
i=2: w=[1.000000 1.000000 -1.000000] xi=[1 14 21] yi = -1 zi=-6.000000
i=3: w=[1.000000 1.000000 -1.000000] xi=[1 27 23] yi = -1 zi=5.000000 wrong
update w0 = w0 - 0.01, w1 = w1 - 27*0.01, w2 = w2 – 23*0.01
i=4: w=[0.990000 0.730000 -1.230000] xi=[1 20 27] yi = -1 zi=-17.620000



Iteration 1
i=0: w= [0.990000 0.730000 -1.230000] xi= [1 23 5] yi = 1 zi=11.630000
i=1: w= [0.990000 0.730000 -1.230000] xi= [1 15 11] yi = 1 zi=-1.590000 wrong
update w0 = w0 + 0.01, w1 = w1 + 15*0.01, w2 = w2 + 11*0.01
i=2: w= [1.000000 0.880000 -1.120000] xi= [1 14 21] yi = -1 zi=-10.200000
i=3: w= [1.000000 0.880000 -1.120000] xi= [1 27 23] yi = -1 zi=-1.000000
i=4: w= [1.000000 0.880000 -1.120000] xi= [1 20 27] yi = -1 zi=-11.640000



Iteration 2
i=0: w= [1.000000 0.880000 -1.120000] xi= [1 23 5] yi = 1 zi=15.640000
i=1: w= [1.000000 0.880000 -1.120000] xi= [1 15 11] yi = 1 zi=1.880000
i=2: w= [1.000000 0.880000 -1.120000] xi= [1 14 21] yi = -1 zi=-10.200000
i=3: w= [1.000000 0.880000 -1.120000] xi= [1 27 23] yi = -1 zi=-1.000000
i=4: w= [1.000000 0.880000 -1.120000] xi= [1 20 27] yi = -1 zi=-11.640000
All classified correctly

Practical work

1. Read the points from a single file test0*.bmp and construct the training set (X,Y). Assign the class label +1 to red points and -1 to blue points2.
2. Implement and apply the online perceptron algorithm to find the linear classifier that divides the points into two groups. Suggestion for parameters: $\eta=10^{-4}$, w = [0.1, 0.1, 0.1], Elimit=10^{-5} , max_iter = 10^5 . Note: for a faster convergence use a larger learning rate only for $w0$
3. Draw the final decision boundary based on the weight vector w.

$$w0 + w1x + w2y = 0.$$

4. Implement the batch perceptron algorithm and find suitable parameters values. Show the loss function at each step. It must decrease slowly.

5. Visualize the decision boundary at intermediate steps, while the learning algorithm is running.