Linear Binary Classification

Thursday, January 16, 2020 12:38

Binary Linear Classification
(perceptron)

- Draw a linear geometry (line / plane...) between 2 categories

$\mathcal{H}$ is a set of lines/hyperplanes.

Model Setup

$$\mathcal{D} = \{(\underline{x}_1, y_1), (\underline{x}_2, y_2) \dots (\underline{x}_N, y_N)\}$$

in out

$$\underline{x}_i = [\underbrace{x_{i0} = 1}_{\text{bias}}, \underbrace{x_{i1}, x_{i2} \dots x_{id}}_{d \text{ values.}}]^T \in \mathbb{R}^{d+1}$$

$$y_i = -1 \text{ or } +1$$

$$y_i \in \{-1, +1\}$$

parameters/weights $W = [w_0, w_1 \dots w_d] \in \mathbb{R}^{d+1}$

CLASSIFICATION RULE:

function $h(\underline{x}_n) = \text{sign}(W^T \cdot \underline{x}_n) = \hat{y}_n$ (estimate)

(?) how do we find values for W

we need a way to see how good/bad our weights are.

⇒ Loss Function:

- one type is Mean Square Error
  but in this classification case, we can just simply count the number of errors.
- indicator function:

$f(\text{event}) = 1$ if $\{\hat{y}_n \neq y_n\}$ ← event is true

$$E_{in}(\underline{w}) = \frac{1}{N}\sum_{n=1}^{N} 1\{\hat{y}_n \neq y_n\}$$

E_{in}: in-sample error (for training data)

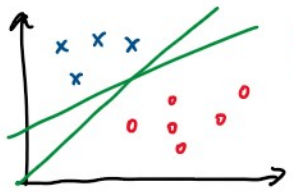
Training involves minimizing $E(\underline{w})$ for the available training data ($\mathcal{D}$)

Perceptron Learning Algorithm (PLA)

INPUT: training set $\mathcal{D}$ is linearly separable

we can put a line to separate data

ex. this is linearly separable



It can be linearly separated using non-unique solutions

OUTPUT: PLA finds $\underline{w} \in \mathbb{R}^{d+1}$ such that $E_{in}(\underline{w}) = 0$

ALGORITHM

- Initialize $\underline{w}(0) = [0, 0, 0 \ldots]$ ($d+1$ dimensions)

  set $t = 0$
- while $E_{in}(\underline{w}(t)) \neq 0$ then
  - pick any misclassified $(\underline{x}_n, y_n)$
  - update $\underline{w}(t+1) = \underline{w}(t) + y_n \underline{x}_n$
  - t++

look at it in more detail

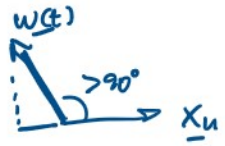
$$\underline{w}(t+1) = \underline{w}(t) + \underbrace{y_n \underline{x}_n}_{\text{misclassified}}$$

$$\underline{w}(t+1) = \underline{w}(t) + y_n \underline{x}_n$$

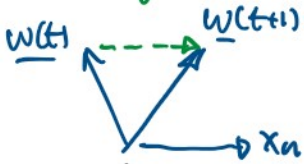
misclassified

case ① $y_n = +1$ $\hat{y}_n = -1$ $\iff$ $\underline{w}(t)^T \cdot \underline{x}_n < 0$

dot product negative



the idea is to add vector so that we get positive (correct classification)



case ② $y_n = -1$ $\hat{y}_n = +1$ $\iff$ $\underline{w}(t)^T \cdot \underline{x}_n > 0$

$y_n \cdot \underline{x}_n$ is negative since $y_n = -1$

⚠ notice in this case the "fix" doesn't fully succeed since the product is still ⊕

In summary

tell that a set is misidentified.

y_n	$\underline{w}(t)^T \underline{x}_n$	$y_n \underline{w}(t)^T \underline{x}_n$	classification
+1	>0	>0	✓
+1	<0	<0	⊗
-1	>0	<0	⊗
-1	<0	>0	✓

PLA: $y_n \underline{w}(t+1)^T \underline{x}_n > y_n \underline{w}(t)^T \underline{x}_n$

⚠ what if training data is not linearly separable?

→ the training will not converge and terminate (and run forever ∞)

→ we can modify algorithm to just minimize E_{in} (keep the best, and continue look for better solution)

→ change objective E_{in}

→ change objective E_{in}

Pocket algorithm: keep the best vector $\underline{w}$ until it finds another better set. Terminates after some iterations.