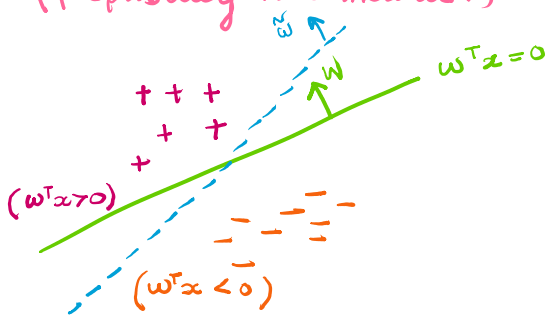
Lecture 9: Perceptron

09 February 2024 10:55

87 Perceptron is a linear, deterministic (propability not included) classification method

$$y_i = f(x_i) \in \{+1, -1\}$$



Goal: Learn a weight vector 'w', that linearly separates the training data points.

$$w^T x_i + w_0 \geq 0 \quad \text{for } y_i = 1$$

$$w^T x_i + w_0 \leq 0 \quad \text{for } y_i = -1$$

NB: i) Such a decision boundary given by 'w' may not be unique.

ii) Instead of considering w_0 separately we can increase dimension of x_i as $\begin{bmatrix} 1 \\ x_i \end{bmatrix}$ and denote $w^T x_i + w_0$ as $w^T x_i$

How does perceptron find w?

1. Goes over the training examples (x_i, y_i) one by one
2. Check if current classifier, w_t, is correct or, if $\left(y_i == \text{sgn}(w_t^T x_i) \right)$
3. If correct, no update required.
4. If not correct, $w_{t+1} \leftarrow w_t + y_i x_i$
5. STOP if no update for a certain number of iterations.

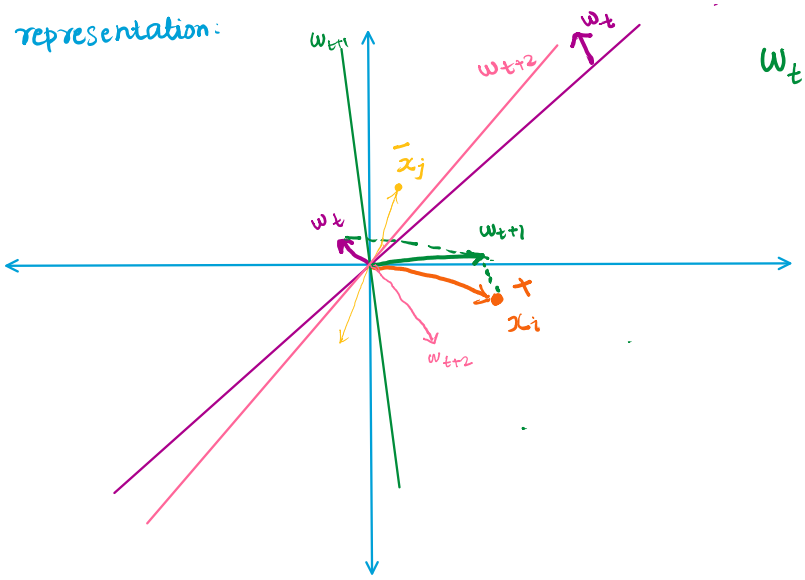
Remarks:

Perceptron is an 'online algorithm' where data points are used one by one, unlike 'batch algorithms' like LR, NB which takes all data points at once.

Algorithm: Perceptron

1. Initialize w_0 (initial w)
2. for $t = 0, 1, 2, \ldots$ maxRounds:
   - Randomly choose a training example (x_i, y_i)
   - if $y_i (w_t^T x_i) < 0$, then
     $$w_{t+1} \leftarrow w_t + y_i x_i$$
   - (equivalently, $w_{t+1} \leftarrow w_t + \frac{1}{2}(y_i - \text{sgn}(w_t^T x_i)) x_i$)
3. STOP when t = maxRounds.

Graphical representation:



$$w_{t+1} \leftarrow w_t + x_i$$

Why is perceptron doing a meaningful update?

Consider a misclassified example (x_i, y_i)

i.e., $\text{sgn}(w_{old}^T x_i) \neq y_i$

But,

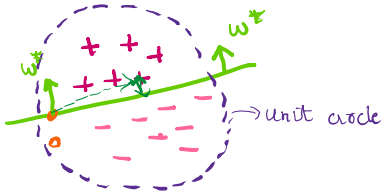
$$w_{new} = w_{old} + y_i x_i \Rightarrow y_i (w_{new}^T x) = y_i (w_{old} + y_i x_i)^T x_i$$
$$= y_i w_{old}^T x_i + \|x_i\|^2$$
$$> y_i (w_{old}^T x)$$

NOTE: We are still not guaranteed that it is correctly classified.

Summary: • Perceptron is a mistake driven online learning algorithm

Summary:
- Perceptron is a mistake driven online learning algorithm
- Guaranteed to converge for linearly separable training example.

Analysing convergence of Perceptron



If the data is linearly separable,

$$\exists w^* \text{ s.t. } y_i(w^{*T}x_i) \geq 0 \quad \forall i = 1, 2, \ldots n$$

Assume $\|w^*\| = 1$, $\|x_i\| \leq 1$:

Define margin of separation, $\gamma = \min_i |w^{*T}x_i| = \min_i \|w^*\| \|x_i\| \cos\theta_i$

Theorem: If $\exists$ a unit vector w^* s.t. $y_i w^{*T} x_i \geq \gamma$, $\forall (x_i, y_i) \in D$
Then the number of weight updates by perceptron is at most $\frac{1}{\gamma^2}$

Proof: Track two quantities: i) $w_t^T w^*$ ii) $\|w_t\|_2^2$

We claim that $w_t^T w^*$ on every update increases by atleast γ.

$$w_{t+1}^T w^* = (w_t + y_i x_i)^T w^*$$
$$= w_t^T w^* + y_i \underbrace{w^{*T} x_i}_{\geq \gamma} \geq w_t^T w^* + \gamma$$

We now claim: $\|w_t\|^2$ increases by atmost one.

$$\|w_{t+1}\|^2 = (w_t + y_i x_i)^T (w_t + y_i x_i) = \|w_t\|^2 + 2y_i \underbrace{w_t^T x_i}_{<0} + \underbrace{\|x_i\|^2}_{\leq 1}$$
(-ve, as misclassified initially)
$$< \|w_t\|^2 + 1$$

Say $w_0 = 0$. After k updates, we've: $w_{k+1}^T w^* \geq k\gamma$

and, $\|w_{k+1}\|^2 < k$

$$\therefore \sqrt{k} > \|w_{k+1}\| \geq w_{k+1}^T w^* \quad \left(w_{k+1}^T w^* = \|w_{k+1}\| \underbrace{\|w^*\|}_{(=1)} \cos\theta \leq 1\right)$$

$$\Rightarrow \quad k < \frac{1}{\gamma^2}$$

Using the theorem above, we could conclude that perceptron algorithm converges, with finite number of mistakes if data is linearly separable.

Limitations of Perceptrons:

i) Though this gives the convergence, but doesn't give the rate of convergence

ii) The no. of iterations can be large if γ is small

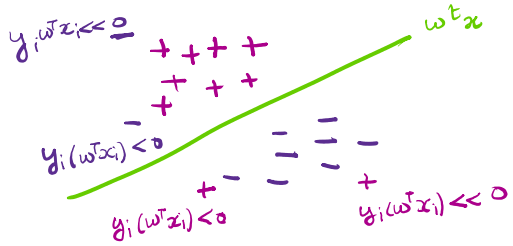
i) Though this gives the convergence, but doesn't give the rate of convergence

ii) The no. of iterations can be large if γ is small

iii) It may not converge if points are not linearly separable.

(Qn: Find one such example of non-convergence)

Loss Function view of Perceptron:



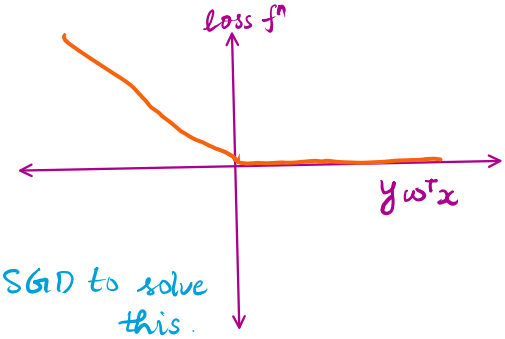
maximize $y_i(w^T x_i)$

$\Rightarrow$ minimize $\sum_{i \in \text{misclassified examples}} (-y_i(w^T x_i))$ (over w)

for any i, $y_i w^T x_i \geq 0 \Rightarrow$ loss $= 0$

$y_i w^T x_i < 0 \Rightarrow$ loss $= -y_i w^T x_i$

Loss function, $L_i(w, D) = \max\{0, -y_i w^T x_i\}$ ← Hinge loss.



$L(w, D) = \sum_i L_i(w, D)$

We adopt stochastic gradient descent, SGD to solve this.

Applying SGD:

- Randomly pick i. Compute $\nabla_w L_i(w, D)$
- Update $w_{t+1} \leftarrow w_t - \underbrace{\nabla_w L_i(w, D)}_{-y_i x_i}$ (eq. $w_t + y_i x_i$)

Thus Hinge loss with SGD is the perceptron algorithm itself.

NB: Hinge loss is non differentiable at origin. In such case choose a suitable gradient.