ECE M146 Introduction to Machine Learning

Prof. Lara Dolecek

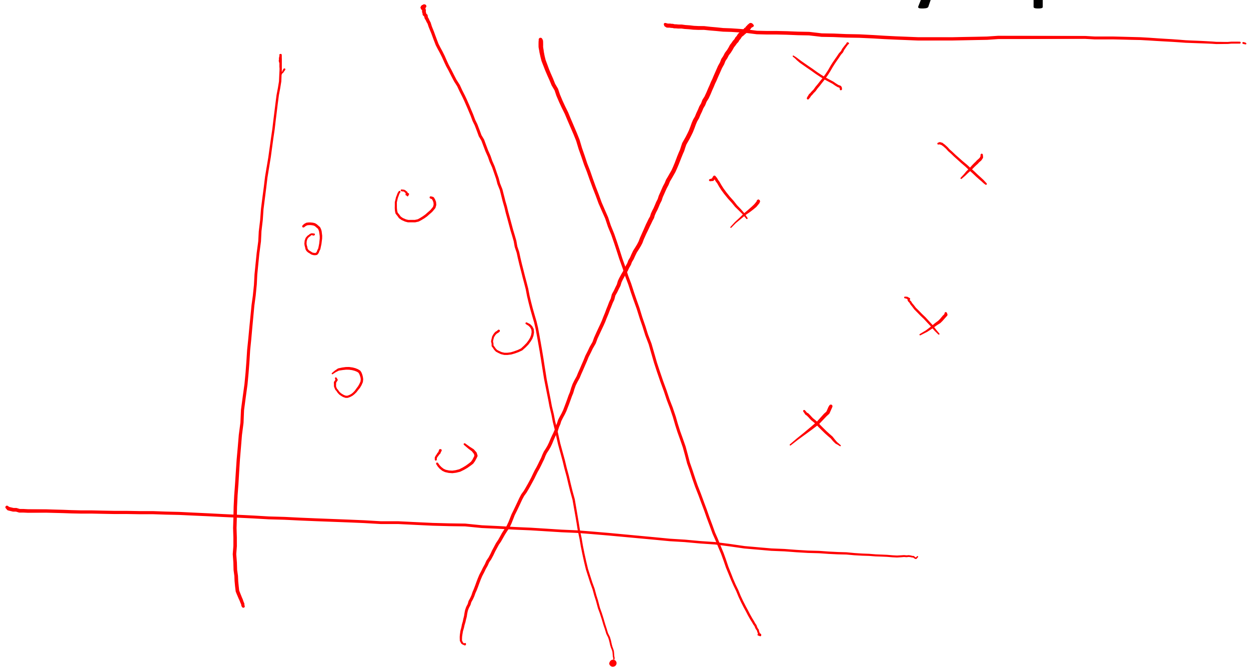
ECE Department, UCLA

Today’s Lecture

- Perceptron Algorithm

Perceptron Algorithm

- Is an algorithm used to perform **binary classification**
- It produces values **on-line**
- Makes no assumptions on the statistics of the underlying training set except that the data has to be **linearly separable**



Perceptron Algorithm

- Is an algorithm used to perform **binary classification**
- It produces values **on-line**
- Makes no assumptions on the statistics of the underlying training set except that the data has to be **linearly separable**
- Picture:

Linearly separating hyperplane

- Goal of the perceptron is to find a decision boundary as a **linearly separating hyperplane**

data in D dimensions D-1 dimensions

- This hyperplane is not necessarily the "best" one but it does separate the training data

Linearly separating hyperplane

- Goal of the perceptron is to find a decision boundary as a **linearly separating hyperplane**

- This hyperplane is not necessarily the "best" one but it does separate the training data

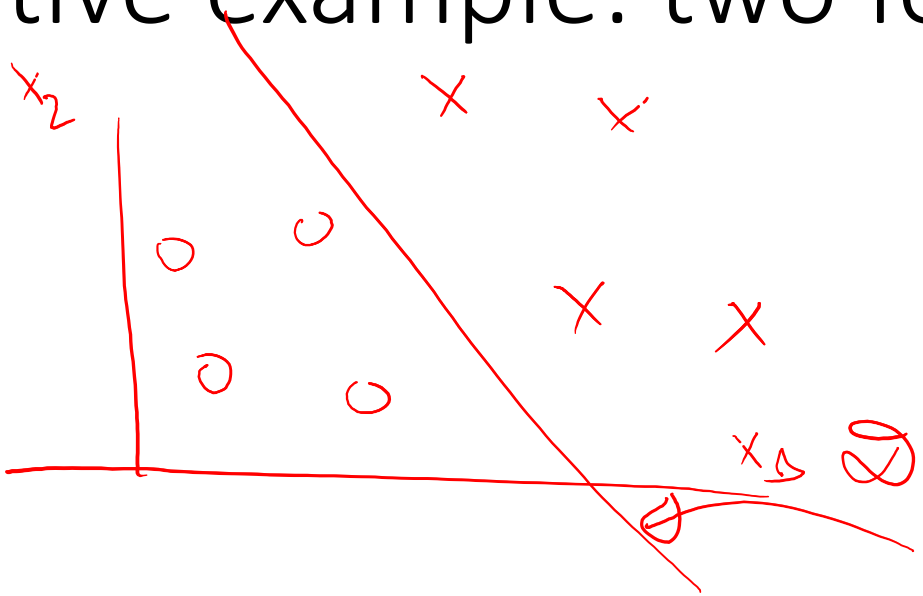
- This decision boundary produced by perceptron algorithm is found in a finite number of steps
  - We will see a proof for this as well.

Key idea

- Perceptron processes labeled training data one at the time, until there are no more misclassified points left with respect to the current classification rule.

Illustrative example: two features

- Picture:



separating hyperplane

x_1 and x_2 are features

$D=2$

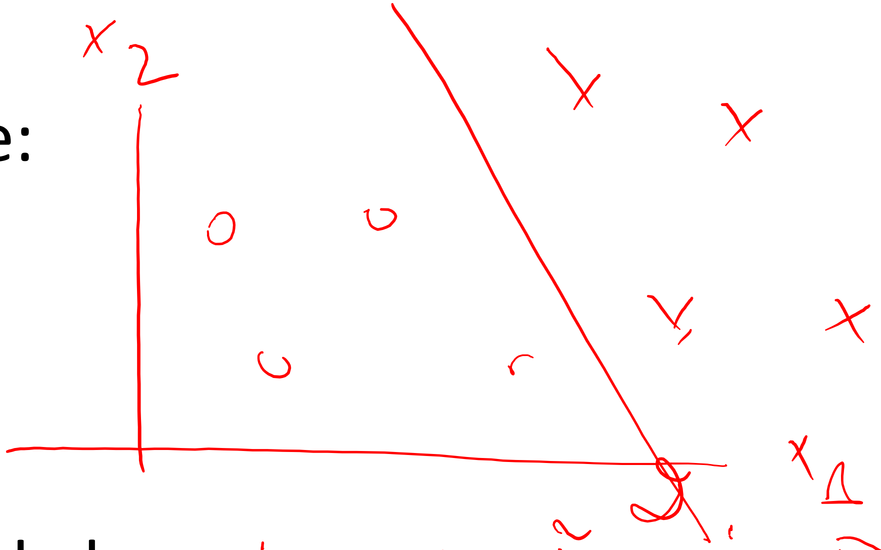
- Line: $w_0 + w_1 \cdot x_1 + w_2 \cdot x_2$
- Input data and their labels:

$\tilde{x}_i = \begin{bmatrix} \tilde{x}_{i1} \\ \tilde{x}_{i2} \end{bmatrix}$ $y_i \in \{+1, -1\}$

$1 < i \le N$

Illustrative example: two features

- Picture:



$w_0 + w_1 \cdot x_1 + w_2 \cdot x_2$

$w \perp \partial$

$\tilde{x} = \begin{bmatrix} \tilde{x}_1 \\ \tilde{x}_2 \end{bmatrix}$

- One side has: $w_0 + w_1 \tilde{x}_1 + w_2 \cdot \tilde{x}_2 > 0$
- The other side has: $w_0 + w_1 \cdot \tilde{x}_1 + w_2 \cdot \tilde{x}_2 < 0$
- Points on the decision boundary satisfy:

$w_0 + w_1 x_1 + w_2 \cdot x_2 = 0$

Convenient transformation

- Instead of 2D, view the data points in 3D.
- Mapping: $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \to \tilde{x} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix}$

$w_0 \cdot 1 + w_1 \cdot x_1 + w_2 \cdot x_2$

- Decision rule now becomes:

$$\boxed{w^T \cdot x \underset{\text{class 2}}{\overset{\text{class 1}}{\gtrless}} 0}$$

$w^T x = 0 \quad x \in \mathcal{D} \quad w = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix}$

Convenient transformation

- Instead of 2D, view the data points in 3D.
- Mapping:

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \rightarrow \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix}$$

- Decision rule now becomes:

$$w^T \cdot x \overset{\text{class 1}}{\underset{\text{class 2}}{\gtrless}} 0$$

- Convenience is that we no longer deal with the intercept term separately.
- Note that now in 3D, the decision boundary passes through the origin.

Perceptron Algorithm

<u>Initialize</u>

- Set k=1
- Set vector $w^{(k)}$ to be the all zero vector.

Perceptron Algorithm

<u>Initialize</u>

- Set k=1
- Set vector $w^{(k)}$ to be the all zero vector.

$1 \le j \le N$

- <u>While</u> there exists a misclassified point, i.e., there exists index j s.t.

$$y_j \, (w^{(k)T} * x_j) < 0$$

Update:

- $w^{(k+1)} = w^{(k)} + y_j x_j$ and increment iteration k by one.

Perceptron Algorithm

Initialize

- Set k=1
- Set vector $w^{(k)}$ to be the all zero vector.

- While there exists a misclassified point, i.e., there exists index j s.t.

$$y_j \left(w^{(k)T} * x_j\right) < 0$$

Update:

- $w^{(k+1)T} = w^{(k)T} + y_j x_j$ and increment iteration k by one.

Return $w^{(k)}$.

Perceptron Algorithm

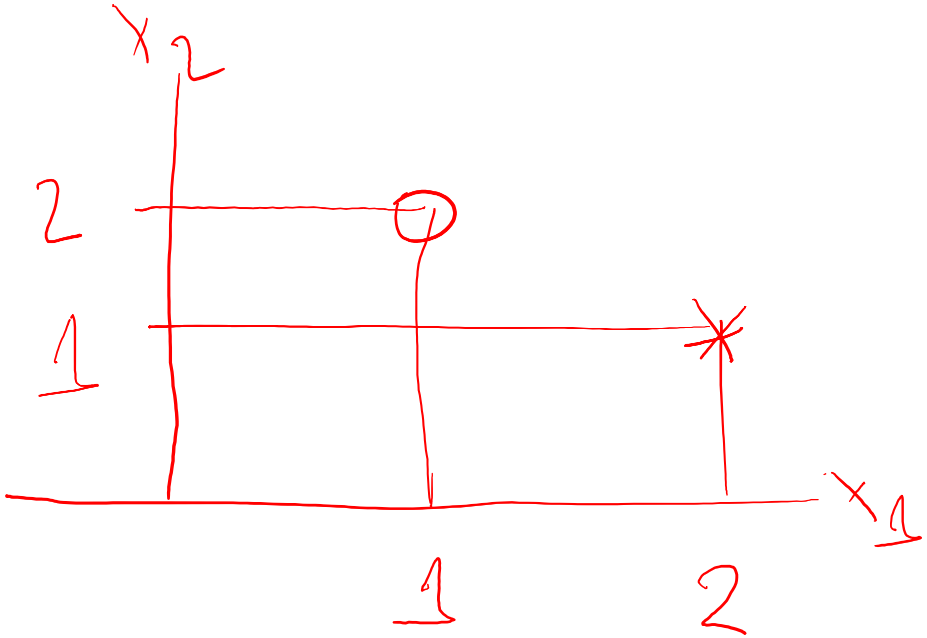
Interpretation:

- While there are misclassified data points, tilt $w^{(k)}$ towards the right classification rule.

$$w^{(k+1)} = w^{(k)} + y_j \cdot x_j$$

Illustrative example: two data points

- Picture:

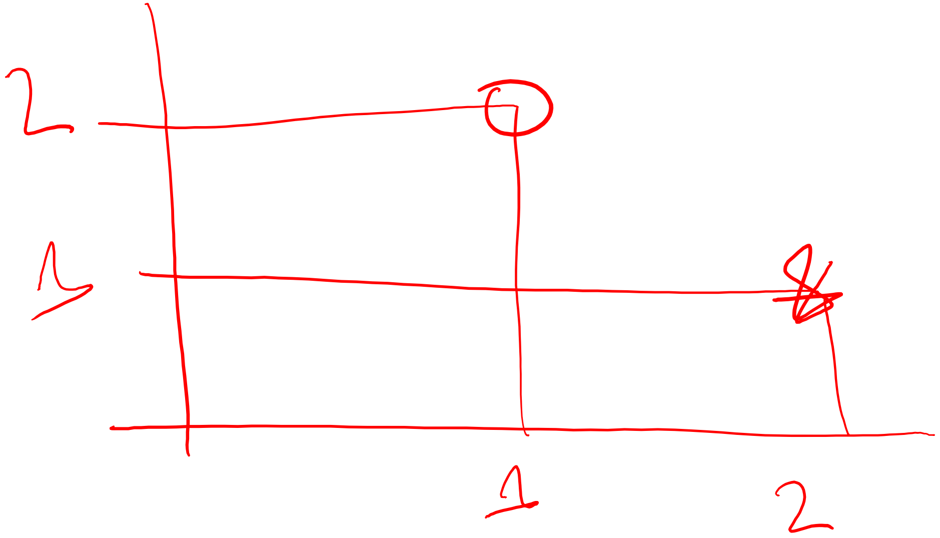


Point $\tilde{x}_x$ has label $\tilde{y}_x = -1$

Point $\tilde{x}_o$ has label $\tilde{y}_o = +1$

Illustrative example: two data points

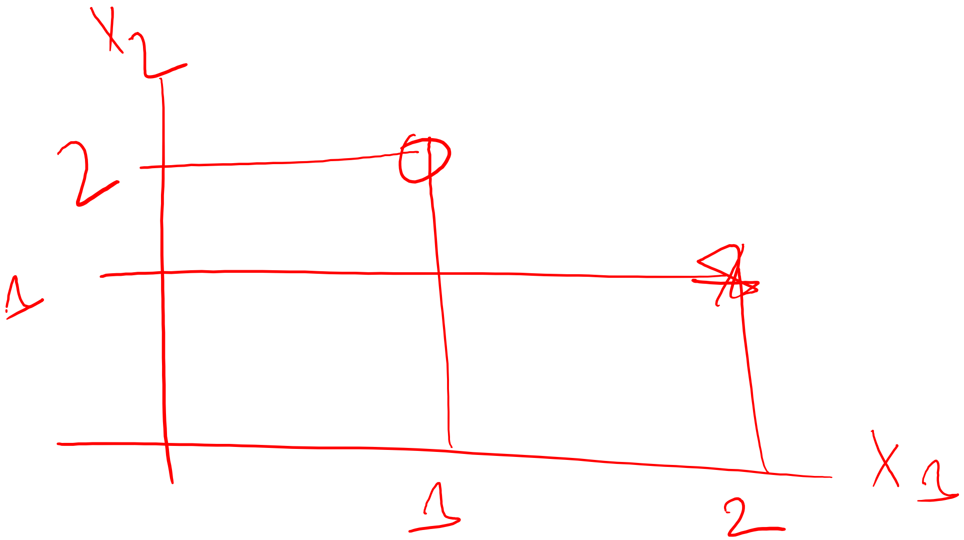
- Picture:



- Goal is to find vector w such that w produces correct labels for the training data, and as such can be used to predicts labels on the testing data.

Illustrative example: two data points

- Picture:



- First, perform transformation from 2D to 3D so that the intercept term is absorbed.

$$X_x = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \qquad X_o = \begin{bmatrix} 1 \\ 1 \\ 2 \end{bmatrix}$$

$\mathrm{sign}(w^T x)$ is the correct label!

Illustrative example: two data points

- Initialize: $w = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$ $k = 1$
- Evaluate at x_o

  sign $\text{sign}(w^T x_o) = \text{sign}(0) = +1$

  Recall $y_o = +1$
- Evaluate at x_*

  sign $\text{sign}(w^T \cdot x_*) = \text{sign}(0) = +1$

  Recall $y_* = -1$ not the same!
- Point x_* is misclassified. Pick this point for the update.

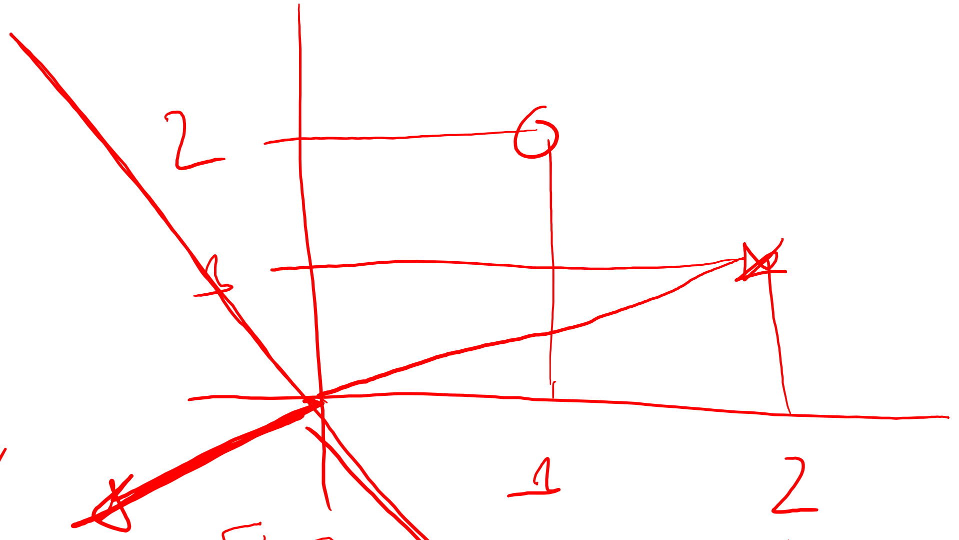
Illustrative example: two data points

- Perform the update:

$$w^{(2)} = w^{(1)} + y_* \cdot x_*$$

$$= \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + (-1) \cdot \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ -2 \\ -1 \end{bmatrix}$$

✓



- Point x_* is now correctly classified.

$$\text{sign}(w^{(1)T} \cdot x_*) = \text{sign}\left([-1\ -2\ -1]\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}\right) = \text{sign}(-6) = -1$$ ✓

Recall $y_* = -1$ ✓

$$w^{(2)} = -x_*$$

- What about point x_o ?

$$\text{sign}(w^{(2)T} \cdot x_o) = \text{sign}\left([-1\ -2\ -1]\begin{bmatrix} 1 \\ 1 \\ 2 \end{bmatrix}\right) = \text{sign}(-5) = -1$$

Illustrative example: two data points

- Check point x_o :

Misclassified

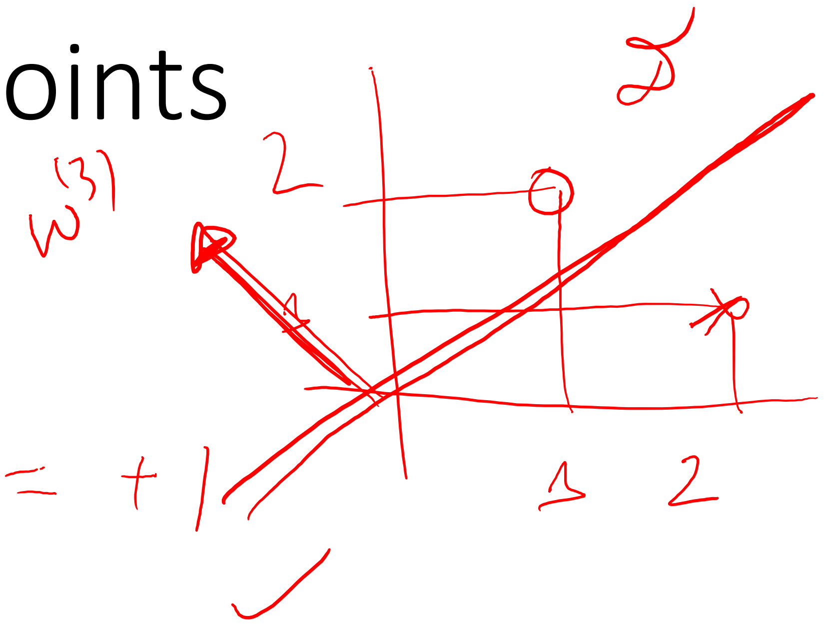
- pictorially

- math

- But this point is now misclassified! So we need to do another update:

$$w^{(3)} = \begin{bmatrix} -1 \\ -2 \\ -1 \end{bmatrix} + (+1) \begin{bmatrix} 1 \\ 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \\ +1 \end{bmatrix}$$

Illustrative example: two data points



- Evaluate at data point x_0:

$$\text{sign}(w^{(3)T} \cdot x_0) =$$
$$\text{sign}\left([0 \;\; -1 \;\; +1] \begin{bmatrix} 1 \\ 1 \\ 2 \end{bmatrix} \right) = \text{sign}(1) = +1$$
$$y_0 = +1$$

- Evaluate at data point x_*:

$$\text{sign}(w^{(3)T} \cdot x_*) =$$
$$\text{sign}\left([0 \;\; -1 \;\; +1] \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \right) = \text{sign}(-1) = -1$$

Recap

- In words: What the update does is that it improves on a misclassified point, and may even make it correctly classified in one step – as in our example.
- In math: Misclassified point (x_j, y_j) $\quad y_j \in \{\pm 1\}$

$$y_j \left(w^{(k)T} \cdot x_j \right) < 0$$

suppose $y_j = +1$

$w^{(k)T} \cdot x_j < 0$

$$w^{(k+1)} = w^{(k)} + y_j \cdot x_j$$

$$\underbrace{w^{(k+1)T} \cdot x_j}_{\text{less negative}} = \left[w^{(k)} + y_j x_j \right]^T \cdot x_j$$

$$= \underbrace{w^{(k)T} \cdot x_j}_{\text{negative}} + \underbrace{y_j \cdot \|x_j\|^2}_{\text{positive}}$$

What about previously correctly classified points ?

- It is indeed possible that previously correctly classified points become misclassified with an update in vector w.

What about previously correctly classified points ?

- It is indeed possible that previously correctly classified points become misclassified with an update in vector w.

- If so, can this process continue on forever without ever reaching the state of having all points correctly classified ?

What about previously correctly classified points ?

- It is indeed possible that previously correctly classified points become misclassified with an update in vector w.
- If so, can this process continue on forever without ever reaching the state of having all points correctly classified ?
- Fortunately, no.

Convergence proof of the perceptron algorithm

- Assumption 1: Suppose there exists some w_{opt} that separates the two classes. (It must exist by the linear separability condition).
- Formally: $\exists w_{opt}$ $\|w_{opt}\| = 1$

$y_i \cdot (w_{opt}^T \cdot x_i) > u \quad u > 0 \quad \forall i \; 1 \le i \le N$

Convergence proof of the perceptron algorithm

- Assumption 1: Suppose there exists some w_{opt} that separates the two classes. (It must exist by the linear separability condition).
- Formally:
- Assumption 2: Bounded coordinates.

$\exists R$ s.t. $\|x_i\| < R \quad \forall i \quad 1 \le i \le N$

Convergence proof of the perceptron algorithm

- Assumption 1: Suppose there exists some w_{opt} that separates the two classes. (It must exist by the linear separability condition).
- Formally:
- Assumption 2: Bounded coordinates.
- Recall that the algorithm does not know w_{opt}, u, or R!

Convergence proof of the perceptron algorithm

- Theorem: Perceptron makes at most R^2/u^2 updates until convergence i.e., there are no misclassified points.

Convergence proof of the perceptron algorithm

- Theorem: Perceptron makes at most R^2/u^2 updates until convergence i.e., there are no misclassified points.

- Proof:
- At k=1
- For any $k \geq 1$, suppose x_j is the misclassified point at that step.
- We build the vector w.

$$w^{(k+1)} = w^{(k)} + y_j \cdot x_j$$

Convergence proof of the perceptron algorithm

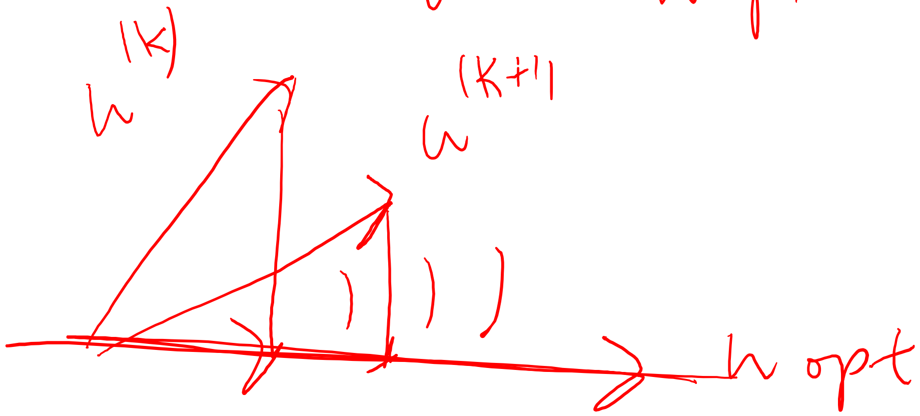
- Proof, continued.

$$w^{(k+1)T} \cdot w_{opt} = (w^{(k)} + y_j \cdot x_j)^T \cdot w_{opt}$$

$$= w^{(k)T} \cdot w_{opt} + y_j x_j^T \cdot w_{opt}$$

$$> w^{(k)T} \cdot w_{opt} + \underset{>0}{\underline{u}}$$

by Assumption 3.



Convergence proof of the perceptron algorithm

- Proof, continued.

$$w^{(k)T} \cdot w_{opt} > w^{(k-1)T} \cdot w_{opt} + \gamma$$

$$w^{(k-1)T} \cdot w_{opt} > w^{(k-2)T} \cdot w_{opt} + \gamma$$

$$w^{(k-2)T} \cdot w_{opt} > w^{(k-3)T} \cdot w_{opt} + \gamma$$

$$\cdots$$

$$w^{(3)T} \cdot w_{opt} > w^{(2)T} \cdot w_{opt} + \gamma \quad 2\gamma$$

$$w^{(2)T} \cdot w_{opt} > w^{(1)T} \cdot w_{opt} + \gamma$$

$$= \gamma$$

- Result 1:

$$\boxed{w^{(k+1)T} \cdot w_{opt} > k \cdot \gamma}$$

Convergence proof of the perceptron algorithm

- Proof, continued.

✓

- Result 1: $\underline{w^{(k+1)T}} \cdot w_{opt} > k \cdot \gamma$
- But, the projection could be getting bigger because $w^{(k+1)}$ is getting bigger, not closer.

Convergence proof of the perceptron algorithm

- Next, consider the inner product.

$$w^{(k+1)T} \cdot w_{opt} \le \|w^{(k+1)}\| \cdot \|w_{opt}\|$$

Recall

$$x^T \cdot y = \|x\| \cdot \|y\| \cos\theta$$

Convergence proof of the perceptron algorithm

- Next, consider the inner product.

$$w^{(k+1)T} \cdot w_{opt} \le \underline{\|w^{(k+1)}\|}$$

$$\|w_{opt}\| = 1$$

- This gives the lower bound on the norm of the vector $w^{(k+1)}$

$$\underline{\|w^{(k+1)}\| > k \cdot \gamma}$$

Convergence proof of the convergence algorithm

- Now, expand the quadratic norm.

$$
\begin{aligned}
\|w^{(k+1)}\|^2 &= \|w^{(k)} + y_j \cdot x_j\|^2 \\
&= \|w^{(k)}\|^2 + \|y_j x_j\|^2 + 2\left(w^{(k)T} \cdot x_j\right) \cdot y_j \\
&= \|w^{(k)}\|^2 + \|x_j\|^2 + 2\underbrace{\left(w^{(k)T} \cdot x_j\right) \cdot y_j}_{<0} \\
&< \|w^{(k)}\|^2 + \|x_j\|^2 \\
&\le \|w^{(k)}\|^2 + R^2 \qquad \text{by Assumption 2}
\end{aligned}
$$

Convergence proof of the convergence algorithm

- Now, expand the quadratic norm.

$$\|w^{(k+1)}\|^2 < \|w^{(k)}\|^2 + R^2$$

$$\|w^{(k)}\|^2 < \|w^{(k-1)}\|^2 + R^2$$

$$\|w^{(3)}\|^2 < \|w^{(2)}\|^2 + R^2$$

$$\|w^{(2)}\|^2 < \|w^{(1)}\|^2 + R^2 = R^2$$

- Result 2:

$$\boxed{\|w^{(k+1)}\|^2 < k \cdot R^2} \quad (2)$$

Combining the two results

$$k \cdot \mu < w^{(k+1)T} \cdot w_{opt}$$

$$k^2 \cdot \mu^2 < (w^{(k+1)T} \cdot w_{opt})^2$$

$$(w^{(k+1)T} \cdot w_{opt})^2 \leq \|w^{(k+1)}\|^2$$

$$\|w^{(k+1)}\|^2 < k \cdot R^2$$

Together

$$k^{\cancel{2}} \cdot \mu^2 < \cancel{k} \cdot R^2$$

$$\boxed{k < R^2/\mu^2}$$

Combining the two results

(1)

(2)

$$\boxed{k < R^2/\gamma^2}$$

- Conclusion:

Observe that the projection is increasing faster than the magnitude of current w.

Further comments on the perceptron algorithm

- In practice, add/design features to make the data set linearly separable in a higher dimensional space.

Further comments on the perceptron algorithm

- In practice, add/design features to make the data set linearly separable in a higher dimensional space.

- The discussion and examples thus far were for the binary classification.

Further comments on the perceptron algorithm

- In practice, add/design features to make the data set linearly separable in a higher dimensional space.

- The discussion and examples thus far were for the binary classification.

- This can be extended to the multi-class classification problem as well.
- Procedure: L classes. $1 \le \ell \le L$

Train on ℓ vs. not(ℓ); get w_ℓ for each class. At test time $\arg\max_\ell (w_\ell^T \cdot x)$

Further comments on the perceptron algorithm

- Observe that perceptron is an **on-line algorithm**: it processes one data point at the time and immediately updates the current value of the vector w.

Further comments on the perceptron algorithm

- Observe that perceptron is an **on-line algorithm**: it processes one data point at the time and immediately updates the current value of the vector w.

- One consequence is that **order** in which data is processed matters! We could get different final w's depending on which point was processed first, second, etc.

Further comments on the perceptron algorithm

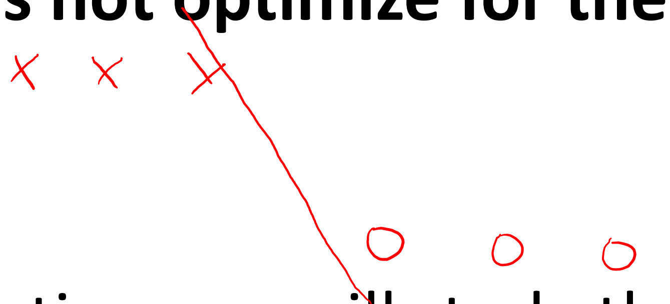
- Observe that perceptron is an **on-line algorithm**: it processes one data point at the time and immediately updates the current value of the vector w.

- One consequence is that **order** in which data is processed matters! We could get different final w's depending on which point was processed first, second, etc.

- In practice, can save older w's and weigh them by a committee vote i.e., how many steps each has survived. Better generalization.

Further comments on the perceptron algorithm

- **Perceptron does not optimize for the margin**
- Example:

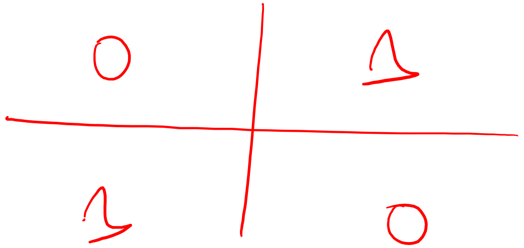


- For this optimization, we will study the support vector machine (SVM)

Further comments on the perceptron algorithm

- **Perceptron does not optimize for the margin**
- Example:

- For this optimization, we will study the support vector machine (SVM)
- **Perceptron cannot process a simple non-linear function**
- Example: XOR function



- This requires a non-linear classifier. Perceptron generalizes to neural nets.