Review: (1) Define machine learning.
(2) a) what are features?
b) what is a feature-vector?
(3) what is a classifier?
(4) what sort of tradeoffs do we have to make when choosing a classifier?

Lecture 2 – Linear Classifiers and the Perceptron Algorithm

Today:
- Classification problem setup
- Linear classifiers
  - Definition
  - Examples
- Perceptron algorithm
  - Algorithm statement
  - Examples
  - Perceptron will/will not converge
  - Convergence on realizable data
  - Average perceptron

Classification problem setup

Training set: $S_n = \{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(n)}, y^{(n)})\}$

$x^{(i)}$ = feature vector for i^{th} data point

$y^{(i)}$ = label for i^{th} datapoint

$$x^{(i)} = \begin{bmatrix} x_1^{(i)} \\ x_2^{(i)} \\ \vdots \\ x_d^{(i)} \end{bmatrix} \in \mathbb{R}^d \qquad y^{(i)} \in \{-1, +1\}$$

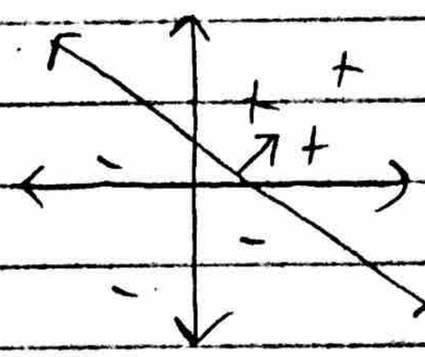
Classifier: $h : \mathbb{R}^d \to \{-1, +1\}$

$h(x^{(i)}) = \pm 1$

$h(x^{(i)}) = y^{(i)} \rightarrow$ correct prediction

$h(x^{(i)}) \neq y^{(i)} \rightarrow$ incorrect prediction

Linear classifier (separator)



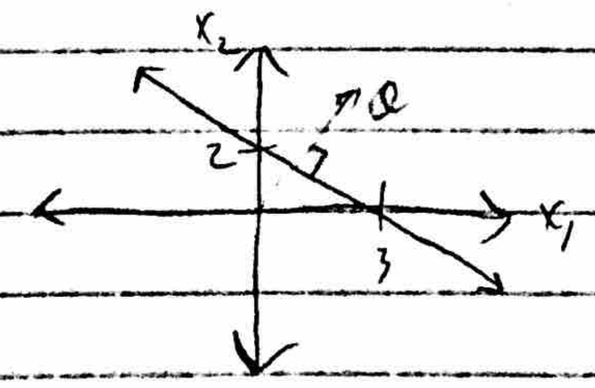
(Arrow points in direction of +)

Equation of a line

(?) What is the equation of the line I drew?

(?) What is $\theta \cdot$ product?

(?) Draw θ.



$2x_1 + 3x_2 - 6 = 0 \qquad (x_2 = -\frac{2}{3}x_1 + 2)$

$$\begin{pmatrix} 2 \\ 3 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + (-6) = 0$$

$\theta \cdot x + \theta_0 = 0$

Note: θ is perpendicular to (normal to) the line.

Example:

$-3x_1 + 2x_2 + 2 = 0$

$x_2 = 3x_1 - 2$

(?) Compute $\text{sign}(\theta \cdot x + \theta_0)$ and classify. Draw the point and confirm that it matches your expectations

a) $x = \begin{pmatrix} 2 \\ 2 \end{pmatrix} \rightarrow +$ ( ) $x = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$

b) $x = \begin{pmatrix} -3 \\ -3 \end{pmatrix} \rightarrow -$ ← otherwise by def.

In general, a hyperplane in d-dimensions can be written as

$\theta \cdot x + \theta_0 = 0$

$\theta = \begin{pmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_d \end{pmatrix} \in \mathbb{R}^d \qquad x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{pmatrix} \in \mathbb{R}^d \qquad \theta_0 \in \mathbb{R}$

In previous example, we had $\theta = \begin{pmatrix} \theta_1 \\ \theta_2 \end{pmatrix} \in \mathbb{R}^2$ and $x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \in \mathbb{R}^2$ and $\theta_0 \in \mathbb{R}$ $= -6$

(?) If x is on the line (hyperplane): $\theta \cdot x + \theta_0 = 0$

(!) If x is in the direction of θ: $\theta \cdot x + \theta_0 > 0$

If x is in the opposite direction of θ: $\theta \cdot x + \theta_0 < 0$

Terminology: Def: $\text{sign}(\theta \cdot x + \theta_0) = \begin{cases} +1, & \theta \cdot x + \theta_0 > 0 \\ -1, & \theta \cdot x + \theta_0 \leq 0 \end{cases}$

The sign function causes the line (hyperplane) to act as a decision boundary.

A linear classifier classifies a data point based on which side of the decision boundary it lies on. (Remember: 0 points towards positive)

For a decision boundary parameterized by θ and θ_0, a linear classifier separates the function

$h(x; \theta, \theta_0) = \text{sign}(\theta \cdot x + \theta_0) = \begin{cases} +1, & \theta \cdot x + \theta_0 > 0 \\ -1, & \theta \cdot x + \theta_0 \leq 0 \end{cases}$

h is a function of x parameterized by θ and θ_0

Goal: find the decision boundary (i.e. θ and θ_0) which best separate (classify) the data

Example

$\theta = \begin{pmatrix} -3 \\ 1 \end{pmatrix}$

$\theta_0 = 2$

Good decision boundary

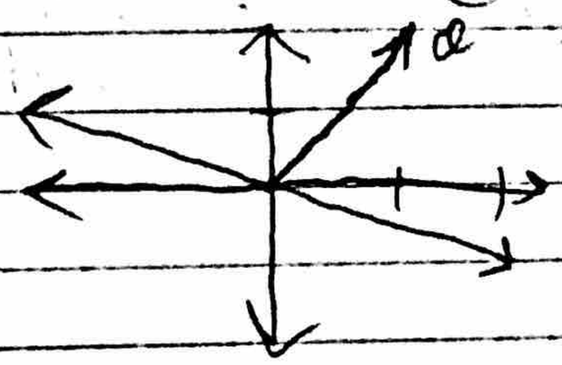
Bad decision boundary

Simplification: Linear separator through the origin
(Offsets allow for more powerful classifiers, but linear classifiers without offsets are easier to reason about so we will often leave out the offset for simplicity)

Linear classifier without offset: $h(x;\theta) = \text{sign}(\theta \cdot x) = \begin{cases} +1, & \theta \cdot x > 0 \\ -1, & \theta \cdot x \leq 1 \end{cases}$

Note: Same as $\theta_0 = 0$.

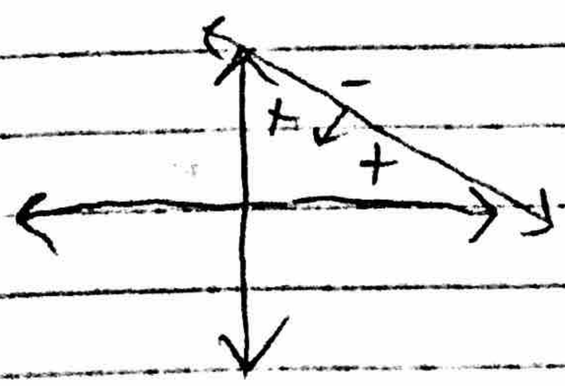
Example: $\theta = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$ $\quad x_1 + 2x_2 = 0$
$x_2 = -\frac{1}{2}x_1$



(Note: Since $\theta_0 = 0$, $x = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ will always satisfy $\theta \cdot x = 0$.
Conclusion: Linear classifiers without offset always go through the origin.)

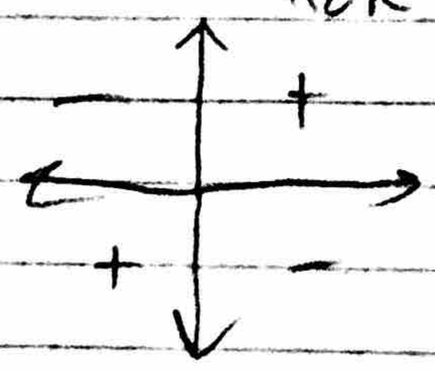
Pro: Easier to reason about.
Con: Weaker classifier

? Can you think of a set of labelled points which can only be classified with an offset?



(?) On the same topic, can you think of a set of points which even a linear classifier with offset can't classify?

XOR



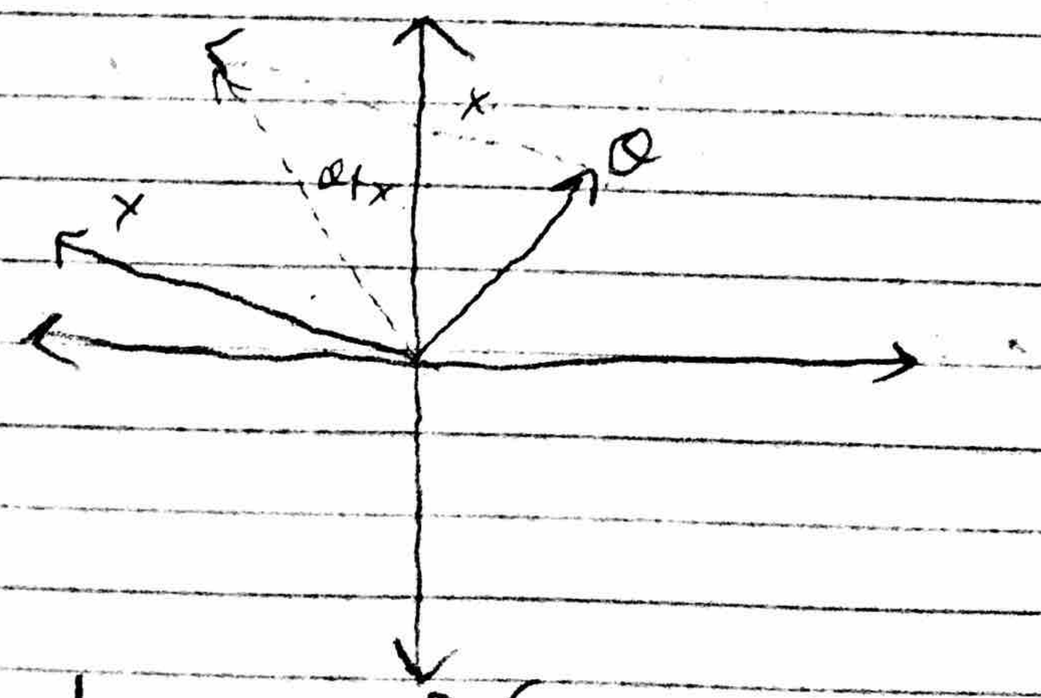
(Remember from last class? Looking ahead: The perceptron algorithm, which we're about to learn, uses linear classifiers. Therefore, perceptron is unable to classify functions such as XOR.)

(Linear classifiers, especially without offset, clearly have limitations, but they are easy to work with and to understand so still worthy learning. Furthermore, next week we will see how to make small modifications to make them much more powerful using a technique called kernels.)

Vector Addition

(Last bit of math necessary to understand perceptron)

Graphically



(?) Can someone draw θ+x? (as a vector from the origin)

Arithmetically

$\theta = \begin{pmatrix} \theta_1 \\ \theta_2 \end{pmatrix} \quad x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$

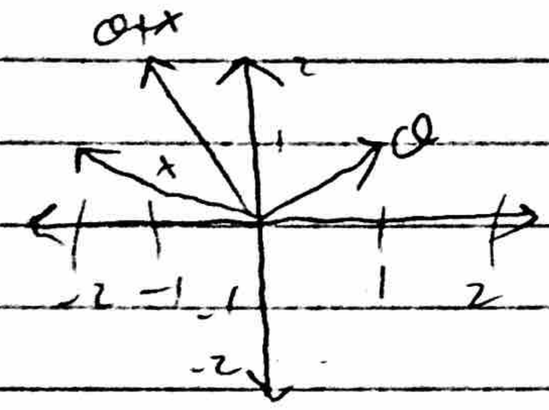
(?) Add element-wise! $\theta + x = \begin{pmatrix} \theta_1 \\ \theta_2 \end{pmatrix} + \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} \theta_1 + x_1 \\ \theta_2 + x_2 \end{pmatrix}$

Example

$\theta = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ $x = \begin{pmatrix} -2 \\ 1 \end{pmatrix}$

(?) Compute $\theta + x = \begin{pmatrix} 1 \\ 1 \end{pmatrix} + \begin{pmatrix} -2 \\ 1 \end{pmatrix} = \begin{pmatrix} -1 \\ 2 \end{pmatrix}$

(?) Draw



Perceptron algorithm (without offset)

Idea: Mistake-driven algorithm using a linear classifier.

1) Guess a linear classifier θ.
2) For each training example x:
   - Check if the linear classifier classifies x correctly.
   - If correct, continue.
   - If mistake, update θ based on x.
     - If x has label $+1$, point θ towards x.
     - If x has label -1, point θ away from x.
3) Repeat step 2 T times.

(What should T be? You'll explore this more in lab today.)

Algorithm:

$\theta = \vec{0}$

repeat T times:

  for $i = 1, 2, \ldots, n$: (n = # training examples)

    if $y^{(i)} \neq h(x^{(i)}, \theta)$: ($h(x^{(i)}, \theta) = \text{sign}(x^{(i)} \cdot \theta)$)

      $\theta = \theta + y^{(i)} x^{(i)}$

return θ

(?) Why adding $y^{(i)} x^{(i)}$?

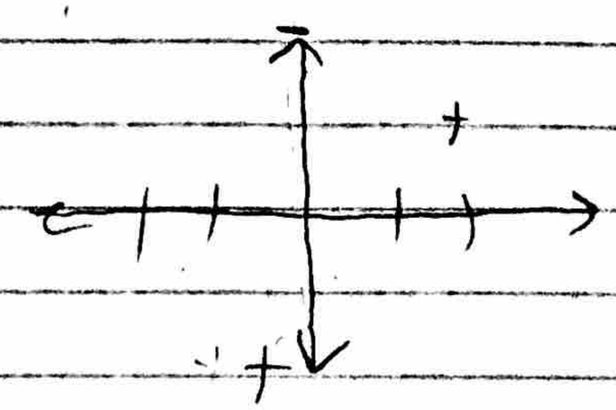
($y^{(i)}$ says whether to move toward or away from x. x tells direction.)

Example

$x^{(1)} = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$ $y^{(1)} = +1$

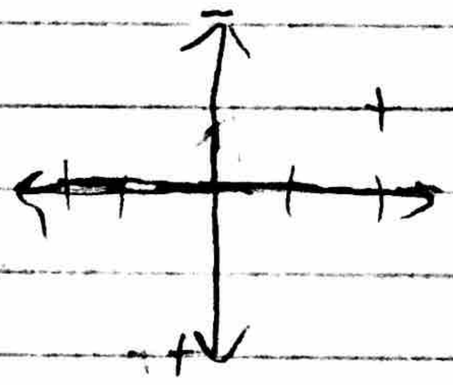
$x^{(2)} = \begin{pmatrix} 0 \\ 2 \end{pmatrix}$ $y^{(2)} = -1$

$x^{(3)} = \begin{pmatrix} -0.5 \\ -2 \end{pmatrix}$ $y^{(3)} = +1$



(?) Guess where decision boundary will be. Get help to do Perceptron

(1) $\theta = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$



(?)

$h(x^{(1)}; \theta) = \text{sign}(\theta \cdot x^{(1)})$

$= \text{sign}\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix} \cdot \begin{pmatrix} 2 \\ 1 \end{pmatrix}\right) = \text{sign}(0) = -1$

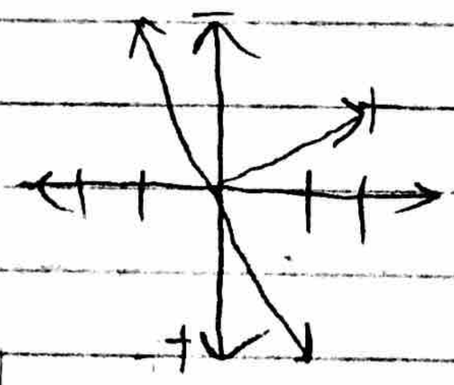
$h(x^{(1)}; \theta) = -1$

$y^{(1)} = +1$

$y^{(1)} \neq h(x^{(1)}; \theta) \rightarrow$ mistake

update: $\theta = \theta + y^{(1)} x^{(1)} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} + 1 \cdot \begin{pmatrix} 2 \\ 1 \end{pmatrix} = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$

(2) $\theta = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$



$h(x^{(2)}; \theta) = \text{sign}(\theta \cdot x^{(2)})$

$= \text{sign}\left(\begin{pmatrix} 2 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 2 \end{pmatrix}\right) = \text{sign}(2) = +1$

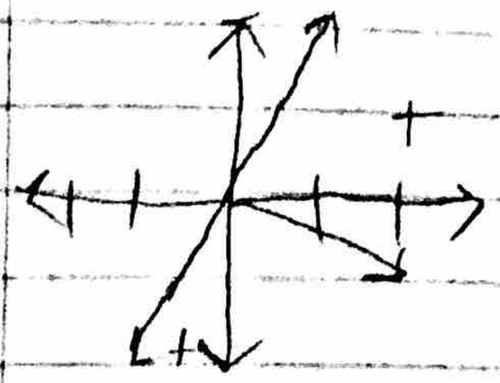
$h(x^{(2)}; \theta) = +1$

$y^{(2)} = -1$

$y^{(2)} \neq h(x^{(2)}; \theta) \rightarrow$ mistake

update: $\theta = \theta + y^{(2)} x^{(2)} = \begin{pmatrix} 2 \\ 1 \end{pmatrix} - 1 \cdot \begin{pmatrix} 0 \\ 2 \end{pmatrix} = \begin{pmatrix} 2 \\ -1 \end{pmatrix}$

(3) $\theta = \begin{pmatrix} 2 \\ -1 \end{pmatrix}$



$h(x^{(3)}, \theta) = \text{sign}(\theta \cdot x^{(3)}) = \text{sign}\left(\begin{pmatrix} 2 \\ -1 \end{pmatrix} \cdot \begin{pmatrix} -0.5 \\ -2 \end{pmatrix}\right)$

$= \text{sign}(-1 + 2) = \text{sign}(1) = +1$

$y^{(3)} = +1$

$y^{(3)} = h(x^{(3)}; \theta) \rightarrow$ correct

no update

(If we check $x^{(1)}$ and $x^{(2)}$, we will see that they are also correctly classified.)

(Since there are no more mistakes, θ will no longer change and we can say that the perceptron converged.)

Perceptron algorithm (with offset)

$\theta = \vec{0}$, $\theta_0 = 0$

repeat T times:

  for $i = 1, 2, \ldots, n$:

    if $y^{(i)} \neq h(x^{(i)}; \theta, \theta_0)$: (note θ_0)

      $\theta = \theta + y^{(i)} x^{(i)}$

      $\theta_0 = \theta_0 + y^{(i)}$

return θ, θ_0

(*) Explain that we can think of θ_0 just as an extra dimension of x which is always 1. So we can actually think of a classifier with offset as a classifier with no offset in $d+1$ dimensions with $\theta = \begin{pmatrix} \theta_1 \\ \vdots \\ \theta_d \\ \theta_0 \end{pmatrix}$ and $x = \begin{pmatrix} x_1 \\ \vdots \\ x_d \\ 1 \end{pmatrix}$

b/c $\theta \cdot x = \theta_1 x_1 + \ldots + \theta_d x_d + \theta_0 \cdot 1$.

Questions:

(?) 1) Does the perceptron algorithm always converge?
(i.e. reach a state where all training points are classified correctly)
No - example is XOR where θ will keep spinning in circles

(?) 2) Are there any cases where the perceptron algorithm is guaranteed to converge?
Yes - whenever the points are linearly separable

Def: A training set S_n is linearly separable if there exists a $\hat{\theta}$ such that $h(x^{(i)}; \hat{\theta}) = y^{(i)}$ for all $i = 1, 2, \ldots, n$.

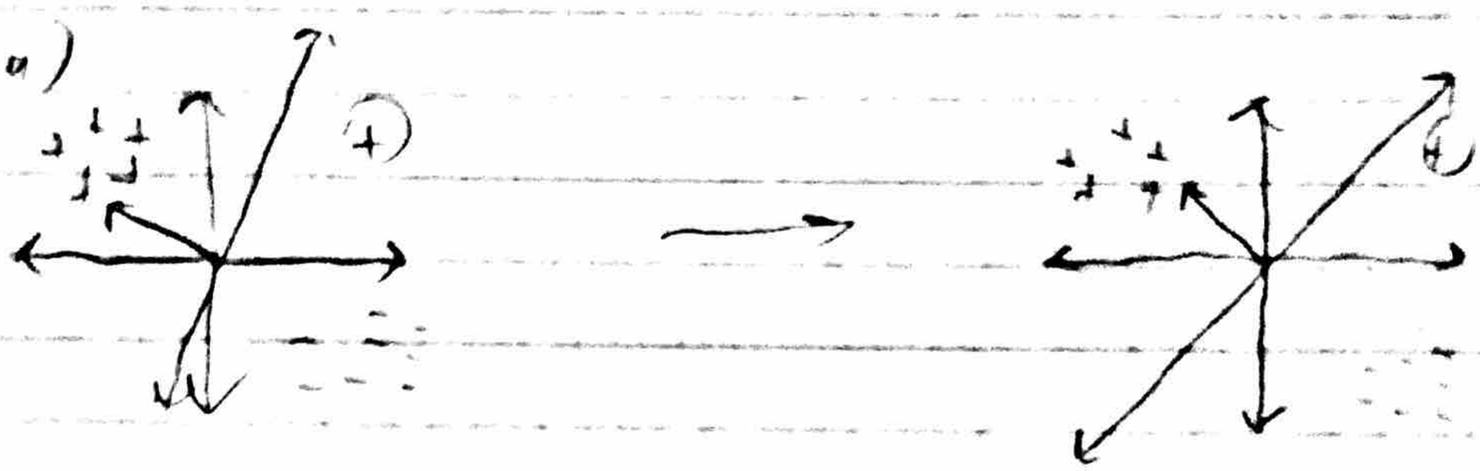
(Talk to me after class or in lab if you want to see proof of convergence)

(?) 3) When the perceptron does converge, does it always converge to the same solution?
No - depends on initialization and order in which data points are traversed

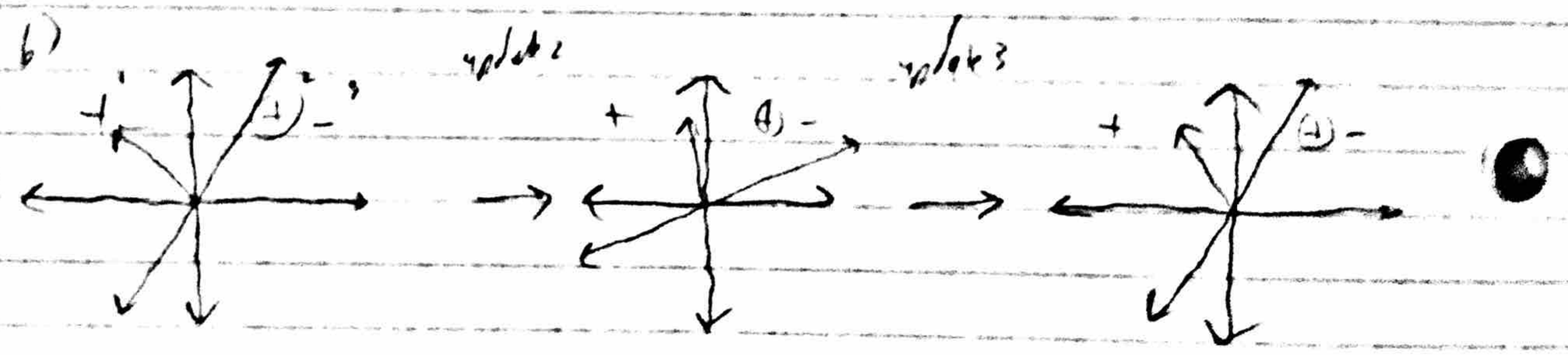
2-8

(?) 4) Once the perceptron misclassifies a point, is it guaranteed to correctly classify the point in the future?
No - even if the update is 0 it can still misclassify the point. Also updates based on other points could change how it's classified

a)



b)



5) What can we say about how the update step affects the classification?
The agreement increases

Def' Agreement $= y^{(i)}(\theta \cdot x^{(i)} + \theta_0)$ ... or just $y^{(i)}(\theta \cdot x^{(i)})$ if $\theta_0 = 0$

$y^{(i)}\,\theta \cdot x^{(i)} > 0 \iff y^{(i)} = \text{sign}(\theta \cdot x^{(i)}) = h(x^{(i)}; \theta) \iff$ correct
$y^{(i)}\,\theta \cdot x^{(i)} \leq 0 \iff y^{(i)} \neq \text{sign}(\theta \cdot x^{(i)}) = h(x^{(i)}; \theta) \iff$ misclassified

Claim: Agreement on $x^{(i)}$ increases after updating θ.

Proof

$\|x\| = \sqrt{x \cdot x} = \sqrt{x_1^2 + \dots x_d^2} \geq 0$

Agreement before update $= \rho = y^{(i)}\,\theta \cdot x^{(i)}$
update: $\theta' = \theta + y^{(i)} x^{(i)}$
Agreement after update $= \rho' = y^{(i)}\,\theta' \cdot x^{(i)} = y^{(i)}\left((\theta + y^{(i)} x^{(i)}) \cdot x^{(i)}\right)$
$= y^{(i)}\left(\theta \cdot x^{(i)} + y^{(i)} x^{(i)} \cdot x^{(i)}\right) = y^{(i)}\,\theta \cdot x^{(i)} + y^{(i)^2} x^{(i)^2} = \rho + \underbrace{1 \cdot \|x^{(i)}\|^2}_{\geq 0} \quad \therefore \rho' \geq \rho$

Improvement i Average Perceptron

Problem: θ can jump around and final theta may be biased toward last-points.

Idea: Average θ across all time steps. Gives more weight to θ's which worked for a large number of points without being updated.

Algorithm

$\theta = \vec{0}$

$\theta_{sum} = \vec{0}$

repeat T times:

for $i = 1, 2, \ldots, n$:

if $y^{(i)} \neq h(x^{(i)}, \theta)$:

$\theta = \theta + y^{(i)} x^{(i)}$

$\theta_{sum} = \theta_{sum} + \theta$ (Note: not in the if so happens on every step)

return $\frac{\theta_{sum}}{nT}$ (works for $\bar{\theta} = \frac{\theta_1 + \theta_2 + \ldots + \theta_{nT}}{nT}$)

Questions?