ORIE 4741: Learning with Big Messy Data

The Perceptron

Professor Udell

Operations Research and Information Engineering
Cornell

October 16, 2021

Announcements

- If you're taking lecture async: remember to submit participation post after each class. (Note: answer polling questions on the async form; no need to pay for iClicker if you'll always be async.)
- Sections start next Tuesday. They are optional, attend any one you prefer. Section next week is a Python + Jupyter refresher `https://github.com/ORIE4741/section`
- Office hours: links or locations and times are posted on course website.
- hw1 will be posted this afternoon, due in two weeks at 9:10am.
- First quiz this week! It should occupy about 20 minutes; you'll have up to half an hour to complete it. Start it anytime between 10am Friday and noon Saturday.
- Start finding project teams. . .

Collaboration policy

homework: yes, you may work with other students!

- Give credit to the people who have helped you: write on your homework the names of the people you worked with.
- Give credit to the other resources that have helped you: please write on your homework the textbooks, notes, or web pages you found useful.
- write up your homework by yourself. That is, all of the text that you submit should be typed or hand-written by you.

quizzes: no, you may not work with other students!

- you may consult your notes, lecture slides, and anything on the internet
- do not talk to other students about the quiz (until after 1pm Saturday)

IP policy

coursehero or other course note websites:

- do not post any course materials there. this makes the next rendition of the course worse for everyone.
- please report to me any course materials you find online (not on our websites).

Poll

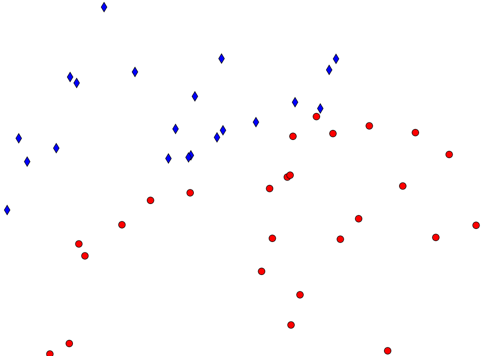
HW0 took me

A. <1 hr
B. 1–5 hrs
C. 5–10 hrs
D. more

A simple classifier: the perceptron

classification problem: *e.g.*, credit card approval

- $\mathcal{X} = \mathbf{R}^d$, $\mathcal{Y} = \{-1, +1\}$
- data $\mathcal{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, $x_i \in \mathcal{X}$, $y_i \in \mathcal{Y}$ for each $i = 1, \ldots, n$
- for picture: $\mathcal{X} = \mathbf{R}^2$, $\mathcal{Y} = \{\mathrm{red}, \mathrm{blue}\}$



Linear classification

- $\mathcal{X} = \mathbf{R}^d$, $\mathcal{Y} = \{-1, +1\}$
- data $\mathcal{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, $x_i \in \mathcal{X}$, $y_i \in \mathcal{Y}$ for each $i = 1, \ldots, n$

make decision using a linear function

- approve credit if

  $$\sum_{j=1}^{d} w_j x_j = w^\top x \geq b;$$

  deny otherwise.
- parametrized by weights $w \in \mathbf{R}^d$
- decision boundary is the hyperplane $\{x : w^\top x = b\}$

Feature transformation

simplify notation: remove the offset b using a **feature transformation**

Feature transformation

simplify notation: remove the offset b using a **feature transformation**

example: approve credit if

$$w^\top x \geq b$$

eg, $\mathcal{X} = \mathbf{R}$, $w = 1$, $b = 2$ (picture)

Feature transformation

simplify notation: remove the offset b using a **feature transformation**

example: approve credit if

$$w^\top x \geq b$$

eg, $\mathcal{X} = \mathbf{R}$, $w = 1$, $b = 2$ (picture)

Q: Can we represent this decision rule by another with no offset?

Feature transformation

simplify notation: remove the offset b using a **feature transformation**

example: approve credit if

$$w^\top x \geq b$$

eg, $\mathcal{X} = \mathbf{R}$, $w = 1$, $b = 2$ (picture)

Q: Can we represent this decision rule by another with no offset?
A: Projective transformation (picture)

Feature transformation

simplify notation: remove the offset b using a **feature transformation**

example: approve credit if

$$w^\top x \geq b$$

eg, $\mathcal{X} = \mathbf{R}$, $w = 1$, $b = 2$ (picture)

Q: Can we represent this decision rule by another with no offset?
A: Projective transformation (picture)

- let $\tilde{x} = (1, x)$, $\tilde{w} = (-b, w)$

Feature transformation

simplify notation: remove the offset b using a **feature transformation**

example: approve credit if

$$w^\top x \geq b$$

eg, $\mathcal{X} = \mathbf{R}$, $w = 1$, $b = 2$ (picture)

Q: Can we represent this decision rule by another with no offset?
A: Projective transformation (picture)

- let $\tilde{x} = (1, x)$, $\tilde{w} = (-b, w)$
- then $\tilde{w}^\top \tilde{x} = w^\top x - b$

now rename $\tilde{x}$ and $\tilde{w}$ as x and w

Geometry of classification

- $\mathcal{X} = \mathbf{R}^d$, $\mathcal{Y} = \{-1, +1\}$
- data $\mathcal{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, $x_i \in \mathcal{X}$, $y_i \in \mathcal{Y}$ for each $i = 1, \ldots, n$
- approve credit if $w^\top x \geq 0$; deny otherwise.

Geometry of classification

- $\mathcal{X} = \mathbf{R}^d$, $\mathcal{Y} = \{-1, +1\}$
- data $\mathcal{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, $x_i \in \mathcal{X}$, $y_i \in \mathcal{Y}$ for each $i = 1, \ldots, n$
- approve credit if $w^\top x \geq 0$; deny otherwise.

if $\|w\| = 1$, inner product $w^\top x$ measures distance of x to classification boundary

- define θ to be angle between x and w
- geometry: distance from x to boundary is $\|x\| \cos(\theta)$
- definition of inner product:

$$w^\top x = \|w\| \|x\| \cos(\theta) = \|x\| \cos(\theta)$$

since $\|w\| = 1$

Linear classification

- $\mathcal{X} = \mathbf{R}^d$, $\mathcal{Y} = \{-1, +1\}$
- data $\mathcal{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, $x_i \in \mathcal{X}$, $y_i \in \mathcal{Y}$ for each $i = 1, \ldots, n$

make decision using a linear function $h : \mathcal{X} \to \mathcal{Y}$

$$h(x) = \mathbf{sign}(w^\top x)$$

Definition

The sign function is defined as

$$\mathbf{sign}(z) = \begin{cases} 1 & z > 0 \\ 0 & z = 0 \\ -1 & z < 0 \end{cases}$$

Linear classification

- $\mathcal{X} = \mathbf{R}^d$, $\mathcal{Y} = \{-1, +1\}$
- data $\mathcal{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$

make decision using a linear function $h : \mathcal{X} \to \mathcal{Y}$

$$h(x) = \mathbf{sign}(w^\top x)$$

Definition

The **hypothesis set** $\mathcal{H}$ is the set of candidate functions might we choose to map $\mathcal{X}$ to $\mathcal{Y}$.

Here, $\mathcal{H} = \{h : \mathcal{X} \to \mathcal{Y} \mid h(x) = \mathbf{sign}(w^\top x)\}$

Poll

Is this function $h : \mathbf{R}^2 \to \mathbf{R}$ a linear classifier?

$$h(x) = \textbf{sign}(x_1 - 5x_2 - 17) = \begin{cases} 1 & x_1 - 5x_2 > 17 \\ 0 & x_1 - 5x_2 = 17 \\ -1 & x_1 - 5x_2 < 17 \end{cases}$$

A. Yes

B. No

Poll

Is this function $h : \mathbf{R}^2 \to \mathbf{R}$ a linear classifier?

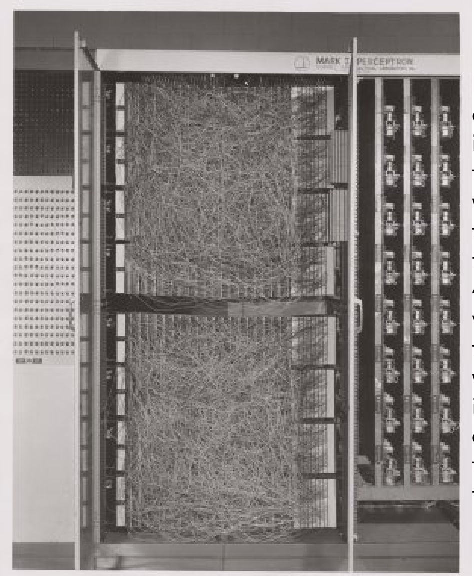
$$h(x) = \mathbf{sign}(x_1^2 - 2x_2 + 27)$$

A. Yes

B. No

The perceptron learning rule

how to learn $h(x) = \mathbf{sign}(w^\top x)$ so that $h(x_i) \approx y_i$?



Frank Rosenblatt's Mark I Perceptron machine was the first implementation of the perceptron algorithm. The machine was connected to a camera that used 20×20 cadmium sulfide photocells to produce a 400-pixel image. The main visible feature is a patchboard that allowed experimentation with different combinations of input features. To the right of that are arrays of potentiometers that implemented the adaptive weights.

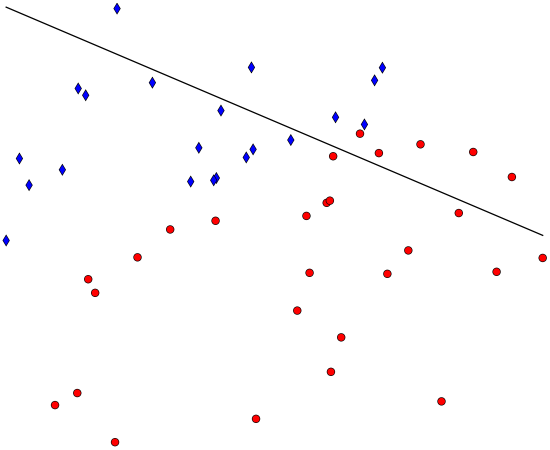
The perceptron learning rule

how to learn $h(x) = \textbf{sign}(w^\top x - b)$ so that $h(x_i) \approx y_i$?

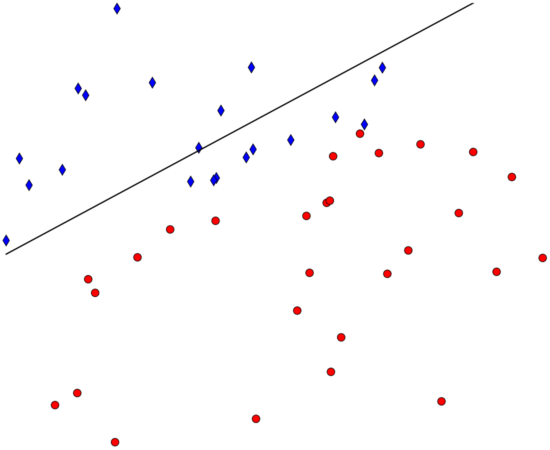
perceptron algorithm [Rosenblatt, 1962]:

- **initialize** $w = \mathbf{0}$
- **while** there is a misclassified example (x, y)
  - $w \leftarrow w + yx$

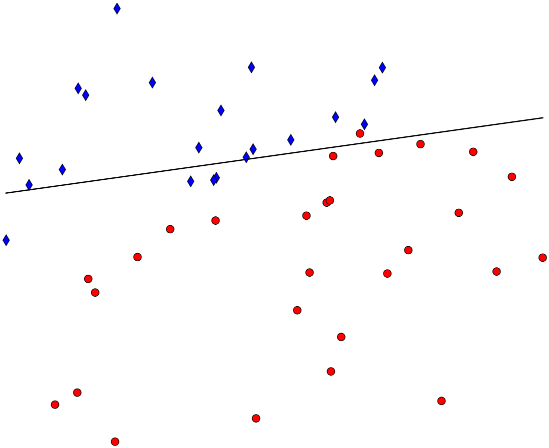
Perceptron: iteration 1



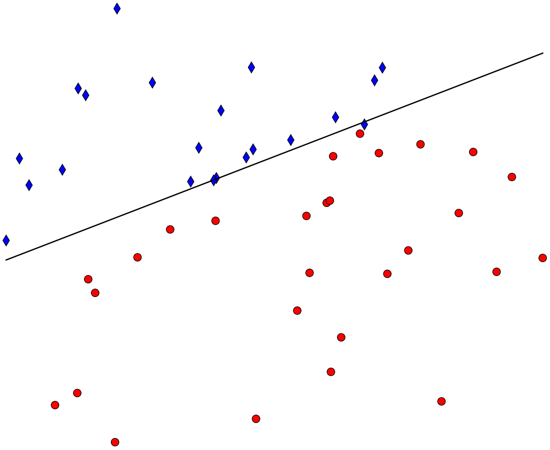
Perceptron: iteration 3



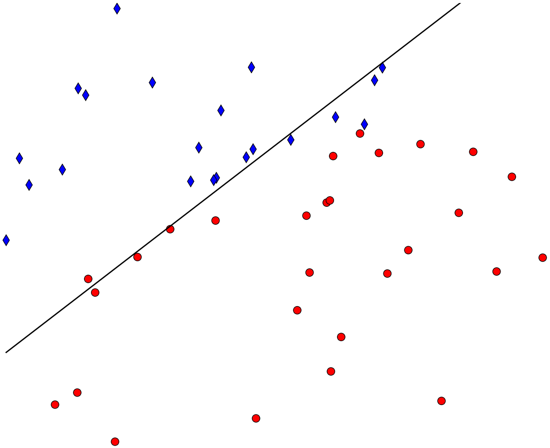
Perceptron: iteration 5



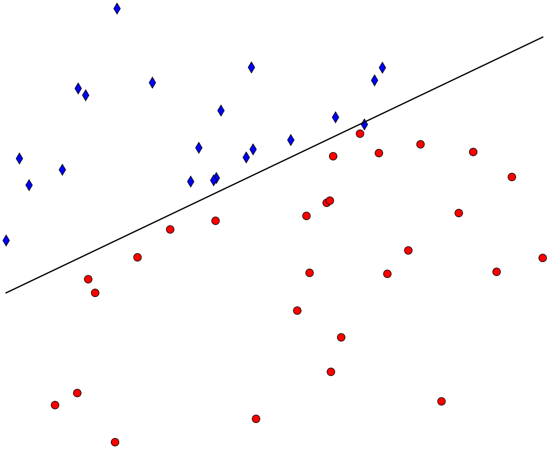
Perceptron: iteration 7



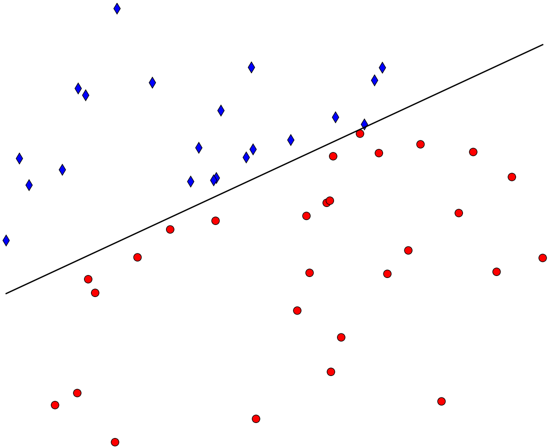
Perceptron: iteration 9



Perceptron: iteration 11



Perceptron: iteration 13



Margin of classifier

correct classification means

$$\begin{cases} w^\top x > 0, & y = 1 \\ w^\top x < 0, & y = -1 \end{cases}$$

Margin of classifier

correct classification means

$$\begin{cases} w^\top x > 0, & y = 1 \\ w^\top x < 0, & y = -1 \end{cases} \implies y w^\top x > 0$$

Margin of classifier

correct classification means

$$\begin{cases} w^\top x > 0, & y = 1 \\ w^\top x < 0, & y = -1 \end{cases} \implies yw^\top x > 0$$

Definition

The **margin** of classifier w on example (x, y) is

$$yw^\top x$$

- positive margin means (x, y) is correctly classified by w
- negative margin means (x, y) is not correctly classified by w

Margin of classifier

correct classification means

$$\begin{cases} w^\top x > 0, & y = 1 \\ w^\top x < 0, & y = -1 \end{cases} \implies yw^\top x > 0$$

Definition

The **margin** of classifier w on example (x, y) is

$$yw^\top x$$

- positive margin means (x, y) is correctly classified by w
- negative margin means (x, y) is not correctly classified by w
- **bigger** margin means (x, y) is **more** correctly classified

The perceptron learning rule

notation: use superscripts $w^{(t)}$ for iterates

perceptron algorithm [Rosenblatt, 1962]:

- **initialize** $w^{(0)} = \mathbf{0}$
- **for** $t = 1, \ldots$
  - **if** there is a misclassified example $(x^{(t)}, y^{(t)})$
    - $w^{(t+1)} = w^{(t)} + y^{(t)} x^{(t)}$
  - **else** quit

The perceptron learning rule

perceptron algorithm: for misclassified $(x^{(t)}, y^{(t)})$,

$$w^{(t+1)} = w^{(t)} + y^{(t)}x^{(t)}$$

The perceptron learning rule

perceptron algorithm: for misclassified $(x^{(t)}, y^{(t)})$,

$$w^{(t+1)} = w^{(t)} + y^{(t)} x^{(t)}$$

Q: why is this a good idea?

The perceptron learning rule

perceptron algorithm: for misclassified $(x^{(t)}, y^{(t)})$,

$$w^{(t+1)} = w^{(t)} + y^{(t)} x^{(t)}$$

Q: why is this a good idea?
A: classification is "better" for $w^{(t+1)}$ than for $w^{(t)}$:
we will show: margin on $(x^{(t)}, y^{(t)})$ is bigger for $w^{(t+1)}$. recall

Definition

The **margin** of classifier w on example (x, y) is

$$y w^\top x$$

- positive margin means (x, y) is correctly classified by w
- negative margin means (x, y) is not correctly classified by w

The perceptron learning rule

perceptron algorithm: for misclassified $(x^{(t)}, y^{(t)})$,

$$w^{(t+1)} = w^{(t)} + y^{(t)} x^{(t)}$$

why is this a good idea?

The perceptron learning rule

perceptron algorithm: for misclassified $(x^{(t)}, y^{(t)})$,

$$w^{(t+1)} = w^{(t)} + y^{(t)} x^{(t)}$$

why is this a good idea?

- example $(x^{(t)}, y^{(t)})$ is misclassified at time t

The perceptron learning rule

perceptron algorithm: for misclassified $(x^{(t)}, y^{(t)})$,

$$w^{(t+1)} = w^{(t)} + y^{(t)} x^{(t)}$$

why is this a good idea?

- example $(x^{(t)}, y^{(t)})$ is misclassified at time t
- $\iff$ $\textbf{sign}(w^{(t)\top} x^{(t)}) \neq y^{(t)}$

The perceptron learning rule

perceptron algorithm: for misclassified $(x^{(t)}, y^{(t)})$,

$$w^{(t+1)} = w^{(t)} + y^{(t)} x^{(t)}$$

why is this a good idea?

- example $(x^{(t)}, y^{(t)})$ is misclassified at time t
- $\iff$ $\textbf{sign}(w^{(t)\top} x^{(t)}) \neq y^{(t)}$
- $\iff$ $y^{(t)} w^{(t)\top} x^{(t)} < 0$.

The perceptron learning rule

perceptron algorithm: for misclassified $(x^{(t)}, y^{(t)})$,

$$w^{(t+1)} = w^{(t)} + y^{(t)} x^{(t)}$$

why is this a good idea?

- example $(x^{(t)}, y^{(t)})$ is misclassified at time t
- $\iff \textbf{sign}(w^{(t)\top} x^{(t)}) \neq y^{(t)}$
- $\iff y^{(t)} w^{(t)\top} x^{(t)} < 0.$
- compute

$$\begin{aligned} y^{(t)} w^{(t+1)\top} x^{(t)} &= y^{(t)} (w^{(t)} + y^{(t)} x^{(t)})^\top x^{(t)} \\ &= y^{(t)} w^{(t)\top} x^{(t)} + (y^{(t)})^2 \|x^{(t)}\|^2 \\ &\geq y^{(t)} w^{(t)\top} x^{(t)} \end{aligned}$$

The perceptron learning rule

perceptron algorithm: for misclassified $(x^{(t)}, y^{(t)})$,

$$w^{(t+1)} = w^{(t)} + y^{(t)} x^{(t)}$$

why is this a good idea?

- example $(x^{(t)}, y^{(t)})$ is misclassified at time t
- $\iff$ $\textbf{sign}(w^{(t)\top} x^{(t)}) \neq y^{(t)}$
- $\iff$ $y^{(t)} w^{(t)\top} x^{(t)} < 0.$
- compute

$$\begin{aligned} y^{(t)} w^{(t+1)\top} x^{(t)} &= y^{(t)} (w^{(t)} + y^{(t)} x^{(t)})^\top x^{(t)} \\ &= y^{(t)} w^{(t)\top} x^{(t)} + (y^{(t)})^2 \|x^{(t)}\|^2 \\ &\geq y^{(t)} w^{(t)\top} x^{(t)} \end{aligned}$$

- so $w^{(t+1)}$ classifies $(x^{(t)}, y^{(t)})$ **better** than $w^{(t)}$ did

The perceptron learning rule

perceptron algorithm: for misclassified $(x^{(t)}, y^{(t)})$,

$$w^{(t+1)} = w^{(t)} + y^{(t)} x^{(t)}$$

why is this a good idea?

- example $(x^{(t)}, y^{(t)})$ is misclassified at time t
- $\iff$ $\mathbf{sign}(w^{(t)\top} x^{(t)}) \neq y^{(t)}$
- $\iff$ $y^{(t)} w^{(t)\top} x^{(t)} < 0.$
- compute

$$\begin{aligned} y^{(t)} w^{(t+1)\top} x^{(t)} &= y^{(t)} (w^{(t)} + y^{(t)} x^{(t)})^\top x^{(t)} \\ &= y^{(t)} w^{(t)\top} x^{(t)} + (y^{(t)})^2 \|x^{(t)}\|^2 \\ &\geq y^{(t)} w^{(t)\top} x^{(t)} \end{aligned}$$

- so $w^{(t+1)}$ classifies $(x^{(t)}, y^{(t)})$ **better** than $w^{(t)}$ did (but possibly still not correctly)

Linearly separable data

Definition

the data $\mathcal{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ is **linearly separable** if

$$y_i = \mathbf{sign}((w^\natural)^\top x_i) \quad i = 1, \ldots, n$$

for some vector $w^\natural$.

that is, there is some hyperplane that (strictly) separates the data into positive and negative examples

Linearly separable data

Definition

the data $\mathcal{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ is **linearly separable** if

$$y_i = \mathbf{sign}((w^\natural)^\top x_i) \quad i = 1, \ldots, n$$

for some vector $w^\natural$.

that is, there is some hyperplane that (strictly) separates the data into positive and negative examples

- $w^\natural$ has positive margin $y_i w^\top x_i > 0$ for every example
- so the **minimum margin** $\rho = \min_{i=1,\ldots,n} y_i x_i^\top w^\natural > 0$

The perceptron learning rule works

how do we know that the perceptron algorithm will work?

The perceptron learning rule works

how do we know that the perceptron algorithm will work?

we'll prove that

Theorem

If the data is linearly separable, then the perceptron algorithm eventually makes no mistakes.

The perceptron learning rule works

how do we know that the perceptron algorithm will work?

we'll prove that

Theorem

If the data is linearly separable, then the perceptron algorithm eventually makes no mistakes.

downside: it could take a long time. . .

Proof of convergence (I)

Let $w^\natural$ be a vector that strictly separates the data into positive and negative examples. So the minimum margin is positive:

$$\rho = \min_{i=1,\ldots,n} y_i x_i^\top w^\natural > 0.$$

Suppose for simplicity that we start with $w^{(0)} = 0$.

- Notice $w^{(t)}$ becomes aligned with $w^\natural$:

$$\begin{aligned} (w^\natural)^\top w^{(t+1)} &= (w^\natural)^\top (w^{(t)} + y^{(t)} x^{(t)}) \\ &= (w^\natural)^\top w^{(t)} + y^{(t)} (w^\natural)^\top x^{(t)} \\ &\geq (w^\natural)^\top w^{(t)} + \rho. \end{aligned}$$

- So by induction, as long as there's a misclassified example at time t,

$$(w^\natural)^\top w^{(t)} \geq \rho t.$$

Proof of convergence (II)

- Define $R = \max_{i=1,\ldots,n} \|x_i\|$.
- Notice $\|w^{(t)}\|$ doesn't grow too fast:

$$\begin{aligned}
\|w^{(t+1)}\|^2 &= \|w^{(t)} + y^{(t)}x^{(t)}\|^2 \\
&= \|w^{(t)}\|^2 + \|x^{(t)}\|^2 + 2y^{(t)}w^{(t)\top}x^{(t)} \\
&\leq \|w^{(t)}\|^2 + \|x^{(t)}\|^2 \\
&\leq \|w^{(t)}\|^2 + R^2
\end{aligned}$$

  because $(x^{(t)}, y^{(t)})$ was misclassified by $w^{(t)}$.
- So by induction,

$$\|w^{(t)}\|^2 \leq tR^2.$$

Proof of convergence (III)

- So as long as there's a misclassified example at time t,

$$(w^\natural)^\top w^{(t)} \geq \rho t \qquad \text{and} \qquad \|w^{(t)}\|^2 \leq tR^2.$$

- Put it together: if there's a misclassified example at time t,

$$\rho t \leq (w^\natural)^\top w^{(t)} \leq \|w^\natural\|\|w^{(t)}\| \leq \|w^\natural\|\sqrt{t}R,$$

so

$$t \leq \left(\frac{\|w^\natural\|R}{\rho}\right)^2.$$

Proof of convergence (III)

- So as long as there's a misclassified example at time t,

$$(w^\natural)^\top w^{(t)} \geq \rho t \qquad \text{and} \qquad \|w^{(t)}\|^2 \leq tR^2.$$

- Put it together: if there's a misclassified example at time t,

$$\rho t \leq (w^\natural)^\top w^{(t)} \leq \|w^\natural\| \|w^{(t)}\| \leq \|w^\natural\| \sqrt{t} R,$$

so

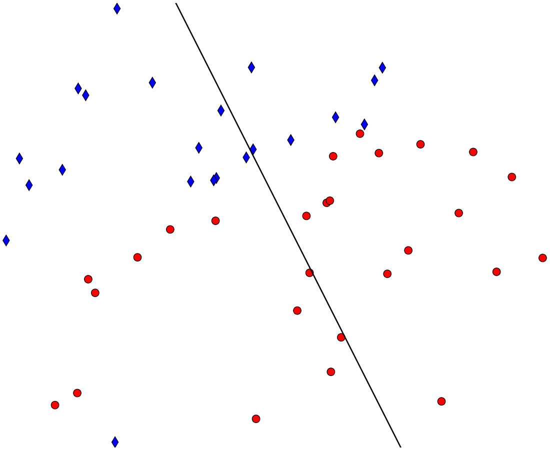
$$t \leq \left(\frac{\|w^\natural\| R}{\rho} \right)^2 .$$

This bounds the maximum running time of the algorithm!

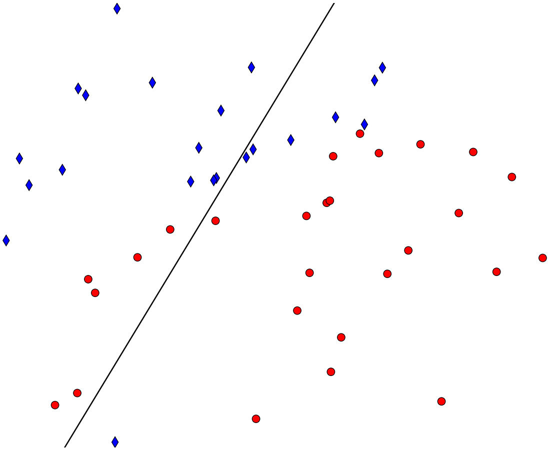
Understanding the bound

- is the bound tight? why or why not?
- what does the bound tell us about **non**-separable data?

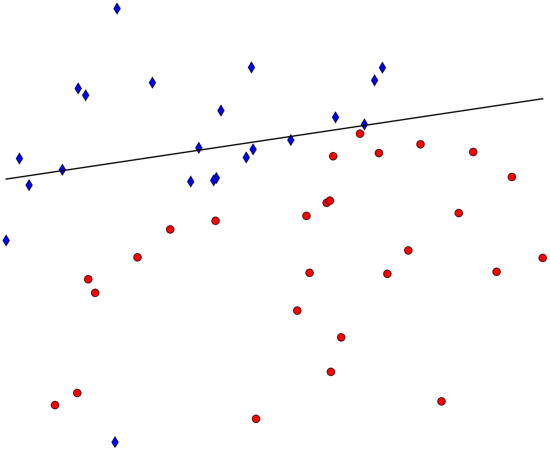
Perceptron with outlier: iteration 1



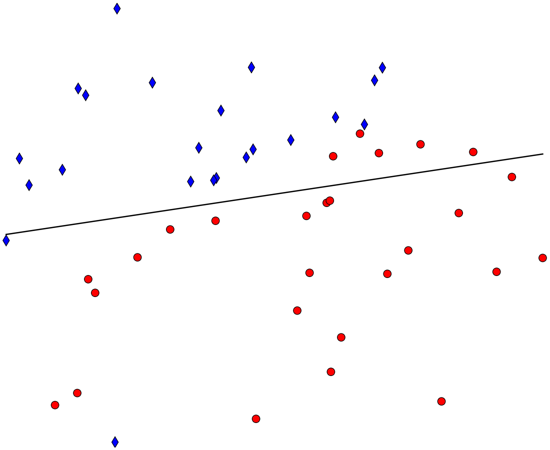
Perceptron with outlier: iteration 2



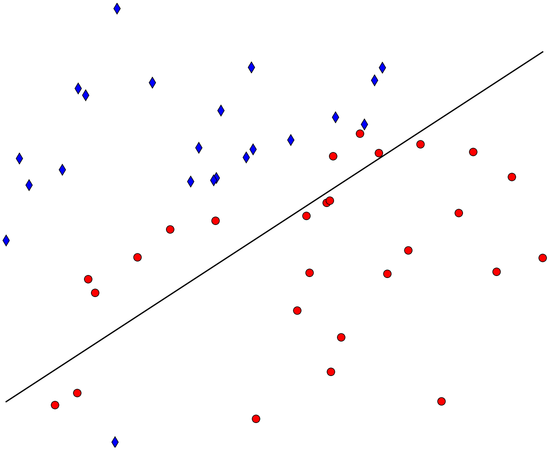
Perceptron with outlier: iteration 3



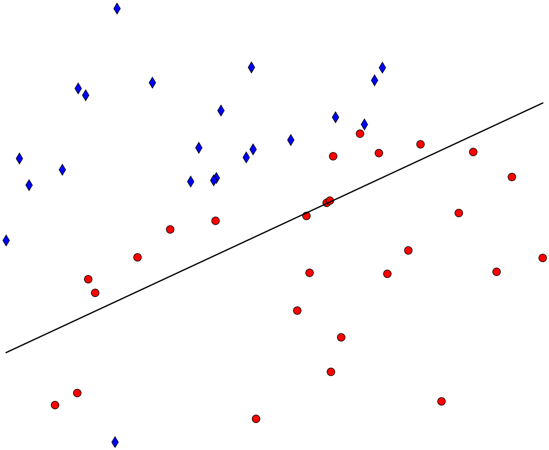
Perceptron with outlier: iteration 4



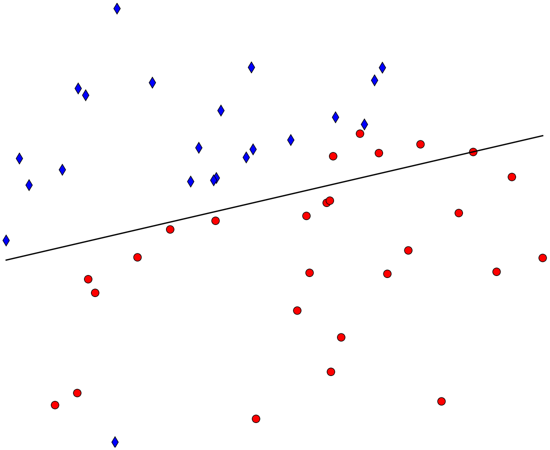
Perceptron with outlier: iteration 5



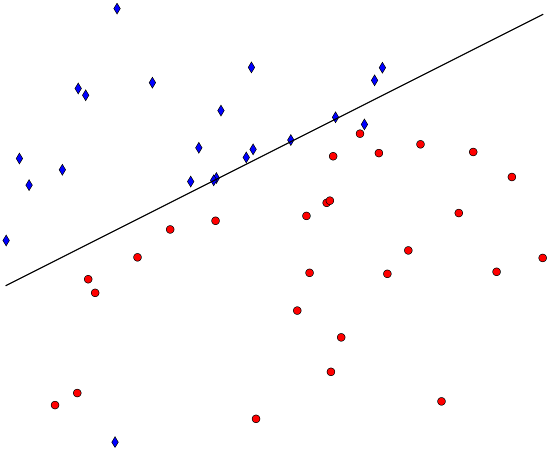
Perceptron with outlier: iteration 47



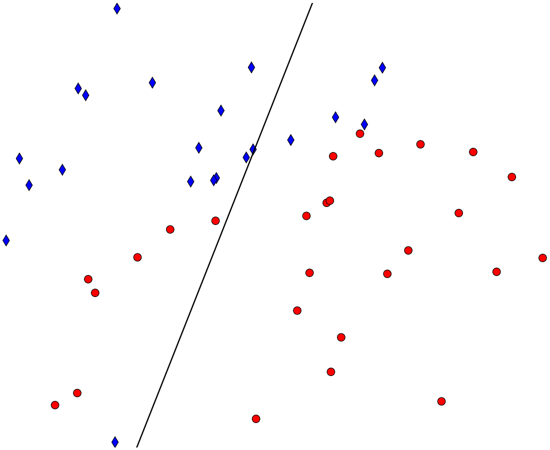
Perceptron with outlier: iteration 48



Perceptron with outlier: iteration 49



Perceptron with outlier: iteration 50



How to measure error?

Q: How to measure the quality of an (imperfect) linear classifier?

How to measure error?

Q: How to measure the quality of an (imperfect) linear classifier?

- Number of misclassifications:

$$\sum_{i=1}^{n} y_i \neq \mathbf{sign}(w^\top x_i)$$

How to measure error?

Q: How to measure the quality of an (imperfect) linear classifier?

- Number of misclassifications:

$$\sum_{i=1}^{n} y_i \neq \textbf{sign}(w^\top x_i)$$

- Size of misclassifications (attempt 1):

$$\sum_{i=1}^{n} \max(-y_i w^\top x_i, 0)$$

How to measure error?

Q: How to measure the quality of an (imperfect) linear classifier?

- Number of misclassifications:

$$\sum_{i=1}^{n} y_i \neq \textbf{sign}(w^\top x_i)$$

- Size of misclassifications (attempt 1):

$$\sum_{i=1}^{n} \max(-y_i w^\top x_i, 0)$$

- Size of misclassifications (attempt 2):

$$\sum_{i=1}^{n} \max(1 - y_i w^\top x_i, 0)$$

Recap: Perceptron

- a simple learning algorithm to learn a linear classifier
- themes we'll see again: linear functions, iterative updates, margin
- how we plotted the data: axes $= \mathcal{X}$, color $= \mathcal{Y}$
- vector $w \in \mathbf{R}^d$ defines linear decision boundary
- simplify algorithm with feature transformation
- proof of convergence: induction, Cauchy-Schwartz, linear algebra

Schema for supervised learning

- unknown target function $f : \mathcal{X} \to \mathcal{Y}$
- training examples $\mathcal{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$
- hypothesis set $\mathcal{H}$
- learning algorithm $\mathcal{A}$
- final hypothesis $g : \mathcal{X} \to \mathcal{Y}$

Generalization

how well will our classifier do on **new** data?

Generalization

how well will our classifier do on **new** data?

- if we know nothing about the new data, no guarantees
- but if the new data looks statistically like the old...