ORIE 4741: Learning with Big Messy Data

The Perceptron

Professor Udell
Operations Research and Information Engineering
Cornell

September 12, 2019
Announcements

- sections next week (Monday and Wednesday): Julia tutorial
- resources from section will be posted on website and at https://github.com/ORIE4741/section
- homework 1 has been released, due in two weeks
- start finding project teams...
classification problem: e.g., credit card approval

- $\mathcal{X} = \mathbb{R}^d$, $\mathcal{Y} = \{-1, +1\}$
- 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: $\mathcal{X} = \mathbb{R}^2$, $\mathcal{Y} = \{\text{red, blue}\}$
Linear classification

\[\mathcal{X} = \mathbb{R}^d, \mathcal{Y} = \{-1, +1\}\]

\[\text{data } \mathcal{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}, \quad x_i \in \mathcal{X}, \ y_i \in \mathcal{Y} \text{ for each } i = 1, \ldots, n\]

make decision using a linear function

\[\text{approve credit if } \sum_{j=1}^{d} w_j x_j = w^\top x \geq b;\]

deny otherwise.

\[\text{parametrized by weights } w \in \mathbb{R}^d\]

\[\text{decision boundary is the hyperplane } \{x : w^\top x = b\}\]
Feature transformation

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

example: approve credit if $w^\top x \geq b$

e.g., $X = \mathbb{R}$, $w = 1$, $b = 2$ (picture)

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

A: Projective transformation (picture)

$\begin{align*}
\text{let } \tilde{x} &= (1, x) \\
\text{and } \tilde{w} &= (-b, w)
\end{align*}$

$\begin{align*}
\tilde{w}^\top \tilde{x} &= w^\top x - b
\end{align*}$

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

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

example: approve credit if

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

ey, $\mathcal{X} = \mathbb{R}$, $w = 1$, $b = 2$ (picture)
Feature transformation

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

example: approve credit if

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

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

Q: Can we represent this decision rule by another with no offset?
simplify notation: remove the offset $b$ using a feature transformation

eexample: approve credit if

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

eg, $\mathcal{X} = \mathbb{R}$, $w = 1$, $b = 2$ (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, $\mathcal{X} = \mathbb{R}$, $w = 1$, $b = 2$ (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

dexample: approve credit if

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

eg, $\mathcal{X} = \mathbb{R}$, $w = 1$, $b = 2$ (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 $w$
How good is your classifier?

- \( \mathcal{X} = \mathbb{R}^d, \mathcal{Y} = \{-1, +1\} \)
- 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.
How good is your classifier?

- $\mathcal{X} = \mathbb{R}^d$, $\mathcal{Y} = \{-1, +1\}$
- 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 $\theta$ 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

- $\mathcal{X} = \mathbb{R}^d$, $\mathcal{Y} = \{-1, +1\}$

- 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 $h : \mathcal{X} \to \mathcal{Y}$

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

**Definition**

The sign function is defined as

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

\[ \mathcal{X} = \mathbb{R}^d, \quad \mathcal{Y} = \{-1, +1\} \]

\[ \text{data } \mathcal{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\} \]

make decision using a linear function \( h : \mathcal{X} \rightarrow \mathcal{Y} \)

\[ h(x) = \text{sign}(w^\top x) \]

**Definition**

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

Here, \( \mathcal{H} = \{ h : \mathcal{X} \rightarrow \mathcal{Y} \mid h(x) = \text{sign}(w^\top x) \} \)
The perceptron learning rule

how to learn \( h(x) = \text{sign}(w^\top x) \) 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 \( 20 \times 20 \) 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 $h(x) = \text{sign}(w^\top x - b)$ so that $h(x_i) \approx y_i$?

perceptron algorithm [Rosenblatt, 1962]:

- initialize $w = 0$
- while there is a misclassified example $(x, y)$
  - $w \leftarrow w + yx$
Perceptron: iteration 1
Perceptron: iteration 3
Perceptron: iteration 5
Perceptron: iteration 7
Margin of classifier

correct classification means

\[
\begin{align*}
&\begin{cases}
  w^T x > 0, & y = 1 \\
  w^T x < 0, & y = -1
\end{cases}
\end{align*}
\]
Margin of classifier

correct classification means

\[
\begin{align*}
    &w^T x > 0, & y = 1 \\
    &w^T x < 0, & y = -1
\end{align*}
\implies yw^T x > 0
\]
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

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

Definition

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

\[yw^T x\]

- 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 $w^{(0)} = 0$
- 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

\[
yw^\top x
\]

- 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 \(t\)
- \(\iff \text{sign}(w^{(t)^\top}x^{(t)}) \neq y^{(t)}\)
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 \(t\)
- \(\iff \text{sign}(w^{(t)\top}x^{(t)}) \neq y^{(t)}\)
- \(\iff y^{(t)}w^{(t)\top}x^{(t)} < 0.\)
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 \(t\)
- \(\iff \) \(\text{sign}(w^{(t)} \top x^{(t)}) \neq y^{(t)}\)
- \(\iff y^{(t)} w^{(t)} \top x^{(t)} < 0.\)
- compute

\[
y^{(t)} w^{(t+1)} \top x^{(t)} = y^{(t)} (w^{(t)} + y^{(t)} x^{(t)}) \top x^{(t)}
\]
\[
= y^{(t)} w^{(t)} \top x^{(t)} + (y^{(t)})^2 \|x^{(t)}\|^2
\]
\[
\geq y^{(t)} w^{(t)} \top x^{(t)}
\]
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 \(t\)
▶ \(\iff \) \(\text{sign}(w^{(t)\top}x^{(t)}) \neq y^{(t)}\)
▶ \(\iff \) \(y^{(t)}w^{(t)\top}x^{(t)} < 0.\)
▶ compute
\[
  y^{(t)}w^{(t+1)\top}x^{(t)} = y^{(t)}(w^{(t)} + y^{(t)}x^{(t)})\top x^{(t)}
  = y^{(t)}w^{(t)\top}x^{(t)} + (y^{(t)})^2\|x^{(t)}\|^2 \\
  \geq y^{(t)}w^{(t)\top}x^{(t)}
\]

▶ 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 \(t\)
- \(\iff \text{sign}(w^{(t)} \top x^{(t)}) \neq y^{(t)}\)
- \(\iff y^{(t)} w^{(t)} \top x^{(t)} < 0\).
- compute

\[
y^{(t)} w^{(t+1)} \top x^{(t)} = y^{(t)} (w^{(t)} + y^{(t)} x^{(t)}) \top x^{(t)} = y^{(t)} w^{(t)} \top x^{(t)} + (y^{(t)})^2 \|x^{(t)}\|^2 \geq y^{(t)} w^{(t)} \top x^{(t)}
\]

- 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 \( D = \{(x_1, y_1), \ldots, (x_n, y_n)\} \) is **linearly separable** if

\[
y_i = \text{sign}((w^\perp)^\top x_i) \quad i = 1, \ldots, n
\]

for some vector \( w^\perp \).

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

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

for some vector $w^\dagger$.

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

- $w^\dagger$ has positive margin $y_i w^\top x_i > 0$ for every example
- so the **minimum margin** $\rho = \min_{i=1,\ldots,n} y_i x_i^\top w^\dagger > 0$
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.
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 $w^\|$ be a vector that strictly separates the data into positive and negative examples. So the minimum margin is positive:

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

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

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

$$ (w^\|)^\top w^{(t+1)} = (w^\|)^\top (w^{(t)} + y^{(t)}x^{(t)}) $$
$$ = (w^\|)^\top w^{(t)} + y^{(t)}(w^\|)^\top x^{(t)} $$
$$ \geq (w^\|)^\top w^{(t)} + \rho.$$ 

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

$$ (w^\|)^\top w^{(t)} \geq \rho t.$$
Proof of convergence (II)

Define \( R = \max_{i=1, \ldots, n} \|x_i\| \).

Notice \( \|w(t)\| \) doesn't grow too fast:

\[
\|w(t+1)\|^2 = \|w(t) + y(t)x(t)\|^2 \\
= \|w(t)\|^2 + \|x(t)\|^2 + 2y(t)w(t)^\top x(t) \\
\leq \|w(t)\|^2 + \|x(t)\|^2 \\
\leq \|w(t)\|^2 + R^2
\]

because \((x(t), y(t))\) was misclassified by \(w(t)\).

So by induction,

\[\|w(t)\|^2 \leq tR^2.\]
So as long as there’s a misclassified example at time $t$,

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

Put it together: if there’s a misclassified example at time $t$,

$$\rho t \leq (w^\perp)^\top w(t) \leq \|w^\perp\|\|w(t)\| \leq \|w^\perp\|\sqrt{tR},$$

so

$$t \leq \left(\frac{\|w^\perp\|R}{\rho}\right)^2.$$
Proof of convergence (III)

- So as long as there’s a misclassified example at time $t$,
  \[
  (w^\|^t)^\top w(t) \geq \rho t \quad \text{and} \quad \|w(t)\|^2 \leq tR^2.
  \]

- Put it together: if there’s a misclassified example at time $t$,
  \[
  \rho t \leq (w^\|^t)^\top w(t) \leq \|w^\|^\|w(t)\| \leq \|w^\|^\sqrt{tR},
  \]
  so
  \[
  t \leq \left( \frac{\|w^\|^\|R\}}{\rho} \right)^2.
  \]
  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 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:

\[ \sum_{i=1}^{n} y_i \neq \text{sign}(w^\top x_i) \]
How to measure error?

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

- **Number of misclassifications:**
  \[
  \sum_{i=1}^{n} y_i \neq \text{sign}(w^T x_i)
  \]

- **Size of misclassifications (attempt 1):**
  \[
  \sum_{i=1}^{n} \max(-y_i w^T x_i, 0)
  \]
How to measure error?

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

- Number of misclassifications:

  \[ \sum_{i=1}^{n} y_i \neq \text{sign}(w^\top x_i) \]

- 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 \( \mathbf{w} \in \mathbb{R}^d \) defines linear decision boundary
- simplify algorithm with feature transformation
- proof of convergence: induction, Cauchy-Schwartz, linear algebra
Schema for supervised learning

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