How to Train Your Perceptron

16-385 Computer Vision (Kris Kitani)
Carnegie Mellon University
Let’s start easy
world’s smallest perceptron!

\[ y = wx \]

(a.k.a. line equation, linear regression)
Learning a Perceptron

Given a set of samples and a Perceptron

\[ \{x_i, y_i\} \]

\[ y = f_{\text{PER}}(x; w) \]

Estimate the parameters of the Perceptron

\[ w \]
Given training data:

<table>
<thead>
<tr>
<th>$x$</th>
<th>$y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>10.1</td>
</tr>
<tr>
<td>2</td>
<td>1.9</td>
</tr>
<tr>
<td>3.5</td>
<td>3.4</td>
</tr>
<tr>
<td>1</td>
<td>1.1</td>
</tr>
</tbody>
</table>

What do you think the weight parameter is?

$$y = wx$$
Given training data:

<table>
<thead>
<tr>
<th>$x$</th>
<th>$y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>10.1</td>
</tr>
<tr>
<td>2</td>
<td>1.9</td>
</tr>
<tr>
<td>3.5</td>
<td>3.4</td>
</tr>
<tr>
<td>1</td>
<td>1.1</td>
</tr>
</tbody>
</table>

What do you think the weight parameter is?

$$y = wx$$

not so obvious as the network gets more complicated so we use …
An Incremental Learning Strategy
(gradient descent)

Given several examples

\[ \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\} \]

and a perceptron

\[ \hat{y} = wx \]
An Incremental Learning Strategy
(gradient descent)

Given several examples
\[
\{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}
\]

and a perceptron
\[
\hat{y} = wx
\]

Modify weight \( w \) such that \( \hat{y} \) gets ‘closer’ to \( y \)
An Incremental Learning Strategy
(gradient descent)

Given several examples
\[
\{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}
\]

and a perceptron
\[
\hat{y} = wx
\]

Modify weight \(w\) such that \(\hat{y}\) gets ‘closer’ to \(y\).
An Incremental Learning Strategy
(gradient descent)

Given several examples
\[ \{ (x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N) \} \]

and a perceptron
\[ \hat{y} = wx \]

Modify weight \( w \) such that \( \hat{y} \) gets ‘closer’ to \( y \)

- perceptron parameter
- perceptron output
- what does this mean?
- true label
Before diving into gradient descent, we need to understand …

**Loss Function**
defines what is means to be **close** to the true solution

**YOU get to chose the loss function!**
(some are better than others depending on what you want to do)
Squared Error (L2)
(a popular loss function)

\[ \ell(\hat{y}, y) = (\hat{y} - y)^2 \]
L1 Loss
\[ \ell(\hat{y}, y) = |\hat{y} - y| \]

L2 Loss
\[ \ell(\hat{y}, y) = (\hat{y} - y)^2 \]

Zero-One Loss
\[ \ell(\hat{y}, y) = 1[\hat{y} = y] \]

Hinge Loss
\[ \ell(\hat{y}, y) = \max(0, 1 - y \cdot \hat{y}) \]
back to the...

World’s Smallest Perceptron!

\[ y = wx \]

(a.k.a. line equation, linear regression)

function of \textbf{ONE} parameter!
Learning a Perceptron

Given a set of samples and a Perceptron

\[ \{x_i, y_i\} \]

\[ y = f_{\text{PER}}(x; w) \]

**What is this activation function?**

Estimate the parameter of the Perceptron

\[ w \]
Learning a Perceptron

Given a set of samples and a Perceptron

\[
\{x_i, y_i\}
\]

\[y = f_{\text{PER}}(x; w)\]

Estimate the parameter of the Perceptron

\[w\]
Learning Strategy
(gradient descent)

Given several examples

\[ \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\} \]

and a perceptron

\[ \hat{y} = wx \]

Modify weight \( w \) such that \( \hat{y} \) gets ‘closer’ to \( y \)
Let’s demystify this process first…

Code to train your perceptron:
Let’s demystify this process first…

Code to train your perceptron:

\[
\text{for } n = 1 \ldots N \\
\quad w = w + (y_n - \hat{y})x_i;
\]

just one line of code!
Let's demystify this process first…

Code to train your perceptron:

```
for n = 1 … N
    w = w + (y_n - \hat{y})x_i;
```

just one line of code!

Now where does this come from?