Linear classifiers: Logistic regression

Linear classifier: Intuition
Classifier

**Input:** $x$

Sentence from review

**Classifier MODEL**

**Output:** $y$

Predicted class

$\hat{y} = +1$

$\hat{y} = -1$

Sentence from review

Input: x

Sushi was awesome, the food was awesome, but the service was awful.

Score($x$) = weighted sum of features of sentence

If Score ($x$) > 0:

$\hat{y} = +1$

Else:

$\hat{y} = -1$

Feature Coefficient

… ...

Simple linear classifier
A simple example: Word counts

<table>
<thead>
<tr>
<th>Feature</th>
<th>Coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>good</td>
<td>1.0</td>
</tr>
<tr>
<td>great</td>
<td>1.2</td>
</tr>
<tr>
<td>awesome</td>
<td>1.7</td>
</tr>
<tr>
<td>bad</td>
<td>-1.0</td>
</tr>
<tr>
<td>terrible</td>
<td>-2.1</td>
</tr>
<tr>
<td>awful</td>
<td>-3.3</td>
</tr>
<tr>
<td>restaurant, the,</td>
<td></td>
</tr>
<tr>
<td>we, where, ...</td>
<td>0.0</td>
</tr>
</tbody>
</table>

Input \( x_i \):
Sushi was great, the food was awesome, but the service was terrible.

\[
\text{Score}(x_i) = 1.2(1) + 1.7(1) - 2.1(1) = 0.8 > 0
\]

\( \hat{y}_i = 1 \)

positive review

Called a linear classifier, because score is weighted sum of features.

More generically...

Model: \( \hat{y}_i = \text{sign}(\text{Score}(x_i)) \)

\[
\text{Score}(x_i) = w_0 h_0(x_i) + w_1 h_1(x_i) + \ldots + w_D h_D(x_i)
\]

\[
= \sum_{j=0}^{D} w_j h_j(x_i) = W^T h(x_i)
\]

feature 1 = \( h_0(x) \) ... e.g., 1
feature 2 = \( h_1(x) \) ... e.g., \( x[1] = \#\text{awesome} \)
feature 3 = \( h_2(x) \) ... e.g., \( x[2] = \#\text{awful} \)
or, \( \log(x[7]) \times x[2] = \log(\#\text{bad}) \times \#\text{awful} \)
or, tf-idf("awful")

... feature D+1 = \( h_D(x) \) ... some other function of \( x[1], \ldots, x[d] \)
Suppose only two words had non-zero coefficient

<table>
<thead>
<tr>
<th>Input</th>
<th>Coefficient</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>#awesome</td>
<td>$w_1$</td>
<td>1.0</td>
</tr>
<tr>
<td>#awful</td>
<td>$w_2$</td>
<td>-1.5</td>
</tr>
</tbody>
</table>

Score($x$) = 1.0 #awesome - 1.5 #awful

Sushi was **awesome**, the food was **awesome**, but the service was **awful**.
### Decision boundary example

<table>
<thead>
<tr>
<th>Input</th>
<th>Coefficient</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w_0$</td>
<td></td>
<td>0.0</td>
</tr>
<tr>
<td>#awesome</td>
<td></td>
<td>1.0</td>
</tr>
<tr>
<td>#awful</td>
<td>$w_2$</td>
<td>-1.5</td>
</tr>
</tbody>
</table>

Score\(x\) = 10 \#awesome - 15 \#awful

Decision boundary separates + and - predictions

- Score\(x\) < 0
- Score\(x\) > 0

For more inputs (linear features)...

Score\(x\) = $w_0 + w_1 \#\text{awesome} + w_2 \#\text{awful} + w_3 \#\text{great}$

... #awesome

©2017 Emily Fox

CSE 446: Machine Learning
For general features…

For more general classifiers (not just linear features) ➞ more complicated shapes

Are you sure about the prediction?  
Class probability
How confident is your prediction?

- Thus far, we’ve outputted a prediction $+1$ or $-1$
- But, how sure are you about the prediction?

“The sushi & everything else were awesome!”

Definite $+1$

“The sushi was good, the service was OK”

Not sure

Using probabilities in classification
How confident is your prediction?

"The sushi & everything else were awesome!"

Definite +1

\[ P(y=+1|x=\text{The sushi & everything else were awesome!}) = 0.99 \]

"The sushi was good, the service was OK"

Not sure

\[ P(y=+1|x=\text{The sushi was good, the service was OK}) = 0.55 \]

Many classifiers provide a degree of certainty:

- Output label
- Input sentence

\[ P(y|x) \]

Extremely useful in practice

Goal: Learn conditional probabilities from data

Training data: \( N \) observations \((x_i, y_i)\)

<table>
<thead>
<tr>
<th>( x_1 ) = #awesome</th>
<th>( x_2 ) = #awful</th>
<th>( y ) = sentiment</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>+1</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>-1</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>-1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>+1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Optimize quality metric on training data

Find best model \( P \) by finding best \( \hat{w} \)

Useful for predicting \( \hat{y} \)
Predict most likely class
\[ \hat{P}(y|x) = \text{estimate of class probabilities} \]

If \( \hat{P}(y=+1|x) > 0.5 \):
\[ \hat{y} = +1 \]
Else:
\[ \hat{y} = -1 \]

Estimating \( \hat{P}(y|x) \) improves interpretability:
- Predict \( \hat{y} = +1 \) and tell me how sure you are

Predicting class probabilities with logistic regression
Thus far, we focused on decision boundaries

\[
\text{Score}(x_i) = w_0 h_0(x_i) + w_1 h_1(x_i) + \ldots + w_D h_D(x_i) = w^T h(x_i)
\]

Relate \( \text{Score}(x_i) \) to \( \hat{P}(y=+1|x, \hat{w}) \)?

Interpreting \( \text{Score}(x_i) \)

\[
\text{Score}(x_i) = w^T h(x_i)
\]

- \( \hat{y}_i = -1 \)  
  - Very sure  
  - \( \hat{P}(y=+1|x_i) = 0 \)

- \( \hat{y}_i = +1 \)  
  - Very sure  
  - \( \hat{P}(y=+1|x_i) = 1 \)

- Not sure if  
  - \( \hat{y}_i = -1 \text{ or } +1 \)  
  - \( \hat{P}(y=+1|x_i) = 0.5 \)
Why not just use regression to build classifier?

Why not just use regression to build classifier?

-∞ < Score(x_i) < +∞

Score(x_i) = w_0 h_0(x_i) + w_1 h_1(x_i) + ... + w_D h_D(x_i)

How do we link -∞, +∞ to 0,1???

But probabilities between 0 and 1

Logistic function (sigmoid, logit)

sigmoid(Score) = \frac{1}{1 + e^{-Score}}

Score | -∞ | -2 | 0.0 | +2 | +∞
-----|-----|----|-----|----|-----
sigmoid(Score) | 0  | 0.12 | 0.5 | 0.88 | 1  

Logistic function used in logistic regression

"link function"
Understanding the logistic regression model

\[ P(y=+1|x, w) = \text{sigmoid}(\text{Score}(x_i)) = \frac{1}{1 + e^{-w^T h(x)}} \]

| Score(x_i) | P(y=+1|x, w) |
|------------|--------------|
| 0          | 0.5          |
| -2         | 0.12 < 0.5   | \( \Rightarrow \hat{y} = -1 \) |
| 2          | 0.88 > 0.5   | \( \Rightarrow \hat{y} = +1 \) |
| 4          | 0.98 > 0.5   | \( \Rightarrow \hat{y} = +1 \) |

Logistic regression \( \rightarrow \) Linear decision boundary
Effect of coefficients on logistic regression model

<table>
<thead>
<tr>
<th>$w_0$</th>
<th>-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w_{#\text{awesome}}$</td>
<td>+1</td>
</tr>
<tr>
<td>$w_{#\text{awful}}$</td>
<td>-1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$w_0$</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w_{#\text{awesome}}$</td>
<td>+1</td>
</tr>
<tr>
<td>$w_{#\text{awful}}$</td>
<td>-1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$w_0$</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w_{#\text{awesome}}$</td>
<td>+3</td>
</tr>
<tr>
<td>$w_{#\text{awful}}$</td>
<td>-3</td>
</tr>
</tbody>
</table>

---

**Linear classifiers:**

![Positive example](image1)

![Negative example](image2)

Parameter learning

---

CSE 446: Machine Learning
Emily Fox
University of Washington
January 23, 2017
Quality metric for logistic regression: Maximum likelihood estimation

Finding best coefficients

<table>
<thead>
<tr>
<th>x[1] = #awesome</th>
<th>x[2] = #awful</th>
<th>y = sentiment</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>+1</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>-1</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>-1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>+1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>+1</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>-1</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>-1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>+1</td>
</tr>
</tbody>
</table>
### Finding best coefficients

<table>
<thead>
<tr>
<th>x[1] = #awesome</th>
<th>x[2] = #awful</th>
<th>y = sentiment</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>-1</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>-1</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>-1</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>-1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>-1</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>-1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>-1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>x[1] = #awesome</th>
<th>x[2] = #awful</th>
<th>y = sentiment</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>+1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>+1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>+1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>+1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>+1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>+1</td>
</tr>
</tbody>
</table>

\[
P(y=+1|x_i, w) = 0.0 \quad \text{Want } \hat{w} \text{ that makes } \quad P(y=+1|x_i, w) = 1.0
\]
Learn logistic regression model with maximum likelihood estimation (MLE)

<table>
<thead>
<tr>
<th>Data point</th>
<th>x[1]</th>
<th>x[2]</th>
<th>y</th>
<th>Choose ( w ) to maximize</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_1,y_1 )</td>
<td>2</td>
<td>1</td>
<td>+1</td>
<td>( P(y=+1</td>
</tr>
<tr>
<td>( x_2,y_2 )</td>
<td>0</td>
<td>2</td>
<td>-1</td>
<td>( P(y=-1</td>
</tr>
<tr>
<td>( x_3,y_3 )</td>
<td>3</td>
<td>3</td>
<td>-1</td>
<td>( P(y=-1</td>
</tr>
<tr>
<td>( x_4,y_4 )</td>
<td>4</td>
<td>1</td>
<td>+1</td>
<td>( P(y=+1</td>
</tr>
</tbody>
</table>

\[
\ell(w) = \prod_{i=1}^{N} P(y_i | x_i, w)
\]

Find “best” classifier
Maximize likelihood over all possible \( w_0, w_1, w_2 \)

\[
\ell(w) = \prod_{i=1}^{N} P(y_i | x_i, w)
\]

\( \ell(w_0=0, w_1=1, w_2=1.5) = 10^{-6} \)

\( \ell(w_0=1, w_1=1, w_2=1.5) = 10^{-5} \)

\( \ell(w_0=1, w_1=0.5, w_2=1.5) = 10^{-4} \)

Best model:
Highest likelihood \( \ell(w) \)
\( \bar{w} = (w_0=1, w_1=0.5, w_2=1.5) \)
Gradient descent for logistic regression

Our optimization objective

- Can compute gradient, but no closed-form solution to:
  \[ \nabla \ell(w) = 0 \]

- Use gradient descent

- As with MLE for Gaussians, rewrite objective as:
  \[ \hat{w} = \arg\max_w \ell(w) = \arg\max_w \ell(w) \]
Step 1: Rewrite log-likelihood

For simpler math, we'll rewrite likelihood with indicators:

\[ \ell(w) = \sum_{i=1}^{N} \ln P(y_i \mid x_i, w) \]

\[ = \sum_{i=1}^{N} [\mathbb{I}[y_i = +1] \ln P(y = +1 \mid x_i, w) + \mathbb{I}[y_i = -1] \ln P(y = -1 \mid x_i, w)] \]

Indicator function:

\[
\mathbb{I}[y_i = +1] = \begin{cases} 
1 & \text{if } y_i = +1 \\
0 & \text{if } y_i = -1 
\end{cases}
\]

Step 2: Express probabilities in terms of \( w \) and \( h(x) \)

Probability model predicts \( y = +1 \):

\[
P(y=+1|x, w) = \frac{1}{1 + e^{-w^T h(x)}}
\]

Probability model predicts \( y = -1 \):

\[
P(y=-1|x, w) = 1 - P(y=+1|x, w) = \frac{e^{-w^T h(x)}}{1 + e^{-w^T h(x)}}
\]
Step 3: Plugging in for 1 data point

\[ P(y = +1 | x, w) = \frac{1}{1 + e^{-w^T h(x)}} \quad \text{and} \quad P(y = -1 | x, w) = \frac{e^{-w^T h(x)}}{1 + e^{-w^T h(x)}} \]

\[
\ell(w) = \mathbb{1}[y_i = +1] \ln P(y = +1 | x_i, w) + \mathbb{1}[y_i = -1] \ln P(y = -1 | x_i, w)
\]

\[
\ell(w) = -\mathbb{1}[y_i = +1] \ln (1 + e^{-w^T h(x_i)}) + (1 - \mathbb{1}[y_i = +1]) (-w^T h(x_i) - \ln (1 + e^{-w^T h(x_i)}))
\]

\[
= -(1 - \mathbb{1}[y_i = +1]) w^T h(x_i) - \ln (1 + e^{-w^T h(x_i)})
\]

Step 4: Gradient for 1 data point

\[
\ell(w) = -(1 - \mathbb{1}[y_i = +1]) w^T h(x_i) - \ln \left(1 + e^{-w^T h(x_i)}\right)
\]

\[
\frac{\partial \ell(w)}{\partial w_j} = -(1 - \mathbb{1}[y_i = +1]) h_j(x_i) \left(1 - P(y = +1 | x_i, w)\right) + h_j(x_i) \mathbb{1}[y_i = +1] P(y = +1 | x_i, w)
\]

\[
= h_j(x_i) \left[ \mathbb{1}[y_i = +1] P(y = +1 | x_i, w) \right]
\]
Step 5: Gradient over all data points

\[
\frac{\partial \ell(w)}{\partial w_j} = \sum_{i=1}^{N} h_j(x_i) \left( \mathbb{1}[y_i = +1] - P(y = +1 \mid x_i, w) \right)
\]

| \(y_i = +1\) | \(P(y=+1|x_i, w) \approx 1\) |
| \(y_i = -1\) | \(P(y=+1|x_i, w) \approx 0\) |

Summary of gradient ascent for logistic regression

init \(w^{(1)} = 0\) (or randomly, or smartly), \(t = 1\)
while \(||\nabla \ell(w^{(t)})|| > \epsilon\)
for \(j = 0, \ldots, D\)
\[
\text{partial}[j] = \sum_{i=1}^{N} h_j(x_i) \left( \mathbb{1}[y_i = +1] - P(y = +1 \mid x_i, w^{(t)}) \right)
\]
\[w^{(t+1)}_j \leftarrow w^{(t)}_j + \eta \text{ partial}[j]\]
\(t \leftarrow t + 1\)