Classification: Analyzing Sentiment

STAT/CSE 416: Machine Learning
Emily Fox
University of Washington
April 17, 2018

Predicting sentiment by topic: *An intelligent restaurant review system*
It’s a big day & I want to book a table at a nice Japanese restaurant

Seattle has many ★★★★★ sushi restaurants

What are people saying about the food? the ambiance?...

Positive reviews not positive about everything

Sample review:

Watching the chefs create incredible edible art made the experience very unique.

My wife tried their ramen and it was pretty forgettable.

All the sushi was delicious! Easily best sushi in Seattle.
From reviews to topic sentiments

Novel intelligent restaurant review app

Experience ★★★★★
Ramen ★★★
Sushi ★★★★★

Easily best sushi in Seattle.

Intelligent restaurant review system

Break all reviews into sentences

The seaweed salad was just OK, vegetable salad was just ordinary.
I like the interior decoration and the blackboard menu on the wall.
All the sushi was delicious.
My wife tried their ramen and it was pretty forgettable.
The sushi was amazing, and the rice is just outstanding.
The service is somewhat hectic. Easily best sushi in Seattle.
Core building block

Easily best sushi in Seattle.

Sentence Sentiment Classifier

Intelligent restaurant review system

All reviews for restaurant

Break all reviews into sentences

Sentence Sentiment Classifier

Average predictions

Sushi ★★★★★

Most &

Easily best sushi in Seattle.
Classifier applications

Classifier

Sentence from review

Input: $x$

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

Classifier MODEL

Predicted class

Output: $y$

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

©2018 Emily Fox
**Spam filtering**

**Input:** $x$

**Output:** $y$

- **Not spam**
- **Spam**

**Text of email, sender, IP,...**

---

**Multiclass classifier**

*Output $y$ has more than 2 categories*

**Input:** $x$

- **Webpage**

**Output:** $y$

- **Education**
- **Finance**
- **Technology**
Image classification

Input: $x$
Image pixels

Output: $y$
Predicted object

Personalized medical diagnosis

Input: $x$

Output: $y$
Disease Classifier
MODEL

Healthy
Cold
Flu
Pneumonia
...

©2018 Emily Fox
STAT/CSE 416: Intro to Machine Learning
Reading your mind

Output $y$

“Hammer”

“House”

Inputs $x$ are brain region intensities

Linear classifiers
Representing classifiers

How does it work???

Sentence from review

Classifier MODEL

Input: \( x \)

Output: \( y \)

Predicted class
List of positive words

great, awesome, good, amazing,...

List of negative words

bad, terrible, disgusting, sucks,...

**Simple threshold classifier**

Count positive & negative words in sentence

If \( \text{number of positive words} > \text{number of negative words} \):

\[
\hat{y} = +
\]

Else:

\[
\hat{y} = -
\]

---

Sentence from review

Input: \( x \)

---

Sushi was great, the food was awesome, but the service was terrible.

---

List of positive words

great, awesome, good, amazing,...

List of negative words

bad, terrible, disgusting, sucks,...

**Simple threshold classifier**

Count positive & negative words in sentence

If \( \text{number of positive words} > \text{number of negative words} \):

\[
\hat{y} = +
\]

Else:

\[
\hat{y} = -
\]
Problems with threshold classifier

• How do we get list of positive/negative words?

• Words have different degrees of sentiment:
  – Great > good
  – How do we weigh different words?

• Single words are not enough:
  – Good ➔ Positive
  – Not good ➔ Negative

Addressed by learning a classifier

Addressed by more elaborate features

A (linear) classifier

Will use training data to learn a weight for each word

<table>
<thead>
<tr>
<th>Word</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>good</td>
<td>1.0</td>
</tr>
<tr>
<td>great</td>
<td>1.5</td>
</tr>
<tr>
<td>awesome</td>
<td>2.7</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, we, where, ...</td>
<td>0.0</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Scoring a sentence

<table>
<thead>
<tr>
<th>Word</th>
<th>Weight</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>0.0</td>
</tr>
<tr>
<td>we, where, ...</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

Input x:
Sushi was great, the food was awesome, but the service was terrible.

Called a linear classifier, because output is weighted sum of input.

Simple linear classifier

\[
Score(x) = \text{weighted count of words in sentence}
\]

If \(Score(x) > 0\):
\[
\hat{y} = +
\]
Else:
\[
\hat{y} = -
\]
More generically...

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

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

$= \sum_{j=0}^{D} w_j h_j(x_i) = \mathbf{w}^\top \mathbf{h}(x_i)$

- 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 $x[1], \ldots, x[d]$

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

(either -1 or +1)
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>

\[
\text{Score}(x) = 1.0 \ #\text{awesome} - 1.5 \ #\text{awful}
\]

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

**Input** | **Coefficient** | **Value**
---|---|---
#awesome | $w_0$ | 0.0
#awful | $w_2$ | -1.5

$\text{Score}(x) = 1.0 \text{ #awesome} - 1.5 \text{ #awful}$

Decision boundary separates + and − predictions

### Decision boundary: effect of changing coefficients

**Input** | **Coefficient** | **Value**
---|---|---
#awesome | $w_0$ | 1.0
#awful | $w_2$ | -1.5

$\text{Score}(x) = 1.0 \text{ #awesome} - 1.5 \text{ #awful}$
Decision boundary: effect of changing coefficients

<table>
<thead>
<tr>
<th>Input</th>
<th>Coefficient</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>#awesome</td>
<td>$w_0$</td>
<td>1.0</td>
</tr>
<tr>
<td>#awful</td>
<td>$w_1$</td>
<td>1.0</td>
</tr>
<tr>
<td>#awesome</td>
<td>$w_2$</td>
<td>-3.0</td>
</tr>
</tbody>
</table>

Score($x$) = **1.0** + 1.0 #awesome − **3.0** #awful

For more inputs (linear features)...

Score($x$) = $w_0$ + $w_1$ #awesome + $w_2$ #awful + $w_3$ #great
For general features...

For more general classifiers (not just linear features)

⇒ more complicated shapes

Training and evaluating a classifier
Training a classifier = Learning the weights

<table>
<thead>
<tr>
<th>Word</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>good</td>
<td>1.0</td>
</tr>
<tr>
<td>awesome</td>
<td>1.7</td>
</tr>
<tr>
<td>bad</td>
<td>-1.0</td>
</tr>
<tr>
<td>awful</td>
<td>-3.3</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Classification error

Learned classifier

\[ \hat{y} = \mp \]

Test example

(Sushi was great)

\[ \hat{y} = \mp \]

Correct
Mistakes

Classification error & accuracy

• Error measures fraction of mistakes

\[ \text{error} = \text{__________} \]

– Best possible value is 0.0

• Often, measure accuracy

– Fraction of correct predictions

\[ \text{accuracy} = \text{__________} \]

– Best possible value is 1.0
What’s a good accuracy?

What if you ignore the sentence, and just guess?

- For binary classification:
  - Half the time, you’ll get it right! (on average)
  \[ \text{accuracy} = \frac{1}{2} \]

- For $k$ classes, accuracy = $\frac{1}{k}$
  - _____ for 3 classes, _____ for 4 classes,…

At the very, very, very least, you should healthily beat random… Otherwise, it’s (usually) pointless…
Is a classifier with 90% accuracy good? Depends...

2010 data shows: "90% emails sent are spam!"

Predicting every email is spam gets you 90% accuracy!!!

Majority class prediction

Amazing performance when there is class imbalance (but silly approach)
- One class is more common than others
- Beats random (if you know the majority class)

So, always be digging in and asking the hard questions about reported accuracies

- Is there class imbalance?
- How does it compare to a simple, baseline approach?
  - Random guessing
  - Majority class
  - ...
- Most importantly: What accuracy does my application need?
  - What is good enough for my user’s experience?
  - What is the impact of the mistakes we make?
False positives, false negatives, and confusion matrices

Types of mistakes

<table>
<thead>
<tr>
<th>True label</th>
<th>Predicted label</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>+</td>
</tr>
<tr>
<td>+</td>
<td>True Positive</td>
</tr>
<tr>
<td>-</td>
<td>False Negative (FN)</td>
</tr>
<tr>
<td></td>
<td>-</td>
</tr>
<tr>
<td>+</td>
<td>False Positive (FP)</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

©2018 Emily Fox

STAT/CSE 416: Intro to Machine Learning
Cost of different types of mistakes can be different (& high) in some applications

<table>
<thead>
<tr>
<th>Spam filtering</th>
<th>Medical diagnosis</th>
</tr>
</thead>
<tbody>
<tr>
<td>False negative</td>
<td>Annoying</td>
</tr>
<tr>
<td>False positive</td>
<td>Email lost</td>
</tr>
<tr>
<td></td>
<td>Disease not treated</td>
</tr>
</tbody>
</table>

Confusion matrix – binary classification

<table>
<thead>
<tr>
<th>Predicted label</th>
<th>True label</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Confusion matrix – multiclass classification

True label

Predicted label

Healthy | Cold | Flu

Healthy

Cold

Flu

Learning curves (again):

*How much data do I need?*
How much data does a model need to learn?

- The more the merrier 😊
  - But data quality is most important factor

- Theoretical techniques can sometimes bound how much data is needed
  - Typically too loose for practical application
  - But provide guidance

- In practice:
  - More complex models require more data
  - Empirical analysis can provide guidance

Test error vs. amount of training data
More complex models tend to have less bias...

- Sentiment classifier using single words can do OK, but...
- Never classifies correctly: “The sushi was not good.”
- More complex model: consider pairs of words (bigrams)

<table>
<thead>
<tr>
<th>Word</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>good</td>
<td>+1.5</td>
</tr>
<tr>
<td>not good</td>
<td>-2.1</td>
</tr>
</tbody>
</table>

Less bias $\Rightarrow$ potentially more accurate, needs more data to learn

Models with less bias tend to need more data to learn well, but do better with sufficient data
Summary of classification intro
What you can do now...

• Identify a classification problem and some common applications
• Describe decision boundaries and linear classifiers
• Train a classifier
• Measure its error
  – Some rules of thumb for good accuracy
• Interpret the types of error associated with classification
• Describe the tradeoffs between model bias and data set size