Classification: Analyzing Sentiment

Emily Fox & Carlos Guestrin
Machine Learning Specialization
University of Washington
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 restaurant

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

All reviews for restaurant

Experience
★★★★★

Ramen
★★★

Sushi
★★★★★

Easily best sushi in Seattle.
Intelligent restaurant review system

All reviews for restaurant

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

Easily best sushi in Seattle.

©2015 Emily Fox & Carlos Guestrin
Intelligent restaurant review system

All reviews for restaurant

Select sentences into subsets "sushi"

The seaweed salad was just OK, vegetable salad was just ordinary.

I like the interior decoration and the blackboard menu on the wall.

My wife tried their ramen and it was pretty forgettable.

The service is somewhat hectic.

Easily best sushi in Seattle.

Easily best sushi in Seattle.

The sushi was amazing, and the rice is just outstanding.

The sushi was delicious.

Sentence Sentiment Classifier

Average predictions

Easily best sushi in Seattle.

Most &

Sushi

★★★★★

Average predictions

Easily best sushi in Seattle.
Classifier applications
Classifier

Sentence from review → Classifier MODEL → Output: y

Input: $x$

Predicted class
Example multiclass classifier

Output $y$ has more than 2 categories

Input: $x$
Webpage

Output: $y$

Education
Finance
Technology
Spam filtering

Input: $x$

Output: $y$

Text of email, sender, IP, …

©2015 Emily Fox & Carlos Guestrin
Image classification

Input: $x$
Image pixels

Output: $y$
Predicted object

Top Predictions
- Labrador retriever
- golden retriever
- redbone
- bloodhound
- Rhodesian ridgeback
Personalized medical diagnosis

Input: $x$

Output: $y$

Disease Classifier MODEL

Healthy
Cold
Flu
Pneumonia
...

©2015 Emily Fox & Carlos Guestrin
Reading your mind

“Hammer”

“House”
Linear classifiers
Representing classifiers

How does it work???

Input: $x$

Output: $y$
Predicted class

Sentence from review

Classifier MODEL

©2015 Emily Fox & Carlos Guestrin
List of positive words | List of negative words
---|---
great, awesome, good, amazing,… | bad, terrible, disgusting, sucks,…

Simple threshold classifier

Count positive & negative words in sentence

If *number of positive words* > *number of negative words*:
\[
\hat{y} = +
\]
Else:
\[
\hat{y} = -
\]

Sentence from review

Input: \(x\)

©2015 Emily Fox & Carlos Guestrin
Sushi was great, the food was awesome, but the service was terrible.

<table>
<thead>
<tr>
<th>List of positive words</th>
<th>List of negative words</th>
</tr>
</thead>
<tbody>
<tr>
<td>great, awesome, good, amazing,…</td>
<td>bad, terrible, disgusting, sucks,…</td>
</tr>
</tbody>
</table>

**Simple threshold classifier**

Count positive & negative words in sentence

If *number of positive words* > *number of negative words*:

\[ \hat{y} = 1 \]

Else:

\[ \hat{y} = 2 \]
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>aweful</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, we, where, ...</td>
<td>0.0</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

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

\[
\text{Score}(x) = 1.2 + 1.7 - 2.1 = 0.8
\]

\[
\text{Score}(x) > 0 \implies + \\
\text{if} \\
\text{Score}(x) < 0 \implies -
\]

Called a linear classifier, because output is weighted sum of input.
Score(x) = weighted count of words in sentence

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

Input: x

Simple linear classifier

Sentence from review

©2015 Emily Fox & Carlos Guestrin
Decision boundaries
Suppose only two words had non-zero weight

<table>
<thead>
<tr>
<th>Word</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>awesome</td>
<td>1.0</td>
</tr>
<tr>
<td>awful</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>Word</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>awesome</td>
<td>1.0</td>
</tr>
<tr>
<td>awful</td>
<td>-1.5</td>
</tr>
</tbody>
</table>

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

\[ \text{Score}(x) > 0 \]

\[ \text{Score}(x) < 0 \]
Decision boundary separates positive & negative predictions

• For linear classifiers:
  - When 2 weights are non-zero
    ➔ line
  - When 3 weights are non-zero
    ➔ plane
  - When many weights are non-zero
    ➔ hyperplane

• For more general classifiers
  ➔ more complicated shapes
Training and evaluating a classifier
Training a classifier = Learning the weights

Data

(x,y)
(Sentence1, - )
(Sentence2, + )
...

Training set

Learn classifier

Test set

Evaluate?

<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

Test example:

($Sushi$ was great, $\hat{y} = +$)

Correct classifier:

$\hat{y} = +$

Mistakes:

$\hat{y} = +$

Hide label

Correct
Mistakes

0
0
Classification error & accuracy

• Error measures fraction of mistakes
  \[ \text{error} = \frac{\# \text{ of mistakes}}{\text{Total } \# \text{ of Sentences}} \]
  – Best possible value is 0.0

• Often, measure accuracy
  – Fraction of correct predictions
  \[ \text{accuracy} = \frac{\# \text{ of Correct}}{\text{Total } \# \text{ of Sentences}} \]
  – 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)
    ⇒ accuracy = 0.5

• For k classes, accuracy = 1/k
  – 0.333 for 3 classes, 0.25 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

True Positive
False Positive (FP)
False Negative (FN)
True Negative
Cost of different types of mistakes can be different (& high) in some applications

<table>
<thead>
<tr>
<th>False negative</th>
<th>Spam filtering</th>
<th>Medical diagnosis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Annoying</td>
<td>Disease not treated</td>
<td></td>
</tr>
<tr>
<td>False positive</td>
<td>Email lost Higher cost</td>
<td>Wasteful treatment</td>
</tr>
</tbody>
</table>
## Confusion matrix – binary classification

<table>
<thead>
<tr>
<th>True Label</th>
<th>Predicted Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>True Positive</td>
<td>50</td>
</tr>
<tr>
<td>False Positive</td>
<td>10</td>
</tr>
<tr>
<td>False Negative</td>
<td>5</td>
</tr>
<tr>
<td>True Negative</td>
<td>35</td>
</tr>
</tbody>
</table>

100 test examples

Accuracy: \[
\frac{85}{100} = 0.85
\]
Confusion matrix – multiclass classification

<table>
<thead>
<tr>
<th>True label</th>
<th>Predicted label</th>
</tr>
</thead>
<tbody>
<tr>
<td>Healthy</td>
<td>Healthy</td>
</tr>
<tr>
<td>Healthy</td>
<td>60</td>
</tr>
<tr>
<td>Cold</td>
<td>4</td>
</tr>
<tr>
<td>Flu</td>
<td>0</td>
</tr>
</tbody>
</table>

100 test examples

Accuracy = \( \frac{80}{100} = 0.8 \)
Learning curves:

*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 sometimes can 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
Learning curves

Test error vs. Amount of training data
Is there a limit?
Yes, for most models...

Even with infinite data, test error will not go to zero.
More complex models tend to have less bias...

Sentiment classifier using single words can do OK, but...

Never classify 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 ➔ 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.

- Classifier based on single words
- Classifier based on bigrams

Test error vs. Amount of training data
Class probabilities
How confident is your prediction?

• Thus far, we’ve outputted a prediction.

• But, how sure are you about prediction?
  
  - “The sushi & everything else were awesome!”  
    \[ P(y=+|x) = 0.99 \]
  
  - “The sushi was good, the service was OK.”  
    \[ P(y=+|x) = 0.55 \]

Many classifiers provide a confidence level:

\[ P(y|x) \]

Output label
Input sentence

Extremely useful in a practice
Summary of classification
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
• Use class probability to express degree of confidence in prediction