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

All reviews for restaurant

Novel intelligent restaurant review app

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.

Intelligent restaurant review system

All reviews for restaurant: The seaweed salad was just OK, vegetable salad was just ordinary. I like the interior decoration and blackboard menu on the wall. My wife tried their ramen and it was pretty forgettable. The service is somewhat hectic. The sushi was amazing, and the rice is just outstanding.

Break all reviews into sentence pieces.

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

Output: y

Predicted class

+ 

-
Spam filtering

Input: $x$
Output: $y$

Text of email, sender, IP,...

Not spam
Spam

Multiclass classifier

Output $y$ has more than 2 categories

Input: $x$
Output: $y$

Webpage

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  
...
Reading your mind

Output $y$

“Hammer”

“House”

Inputs $x$ are brain region intensities

Linear classifiers
Representing classifiers

How does it work???

Sentence from review
Input: $x$

Classifier

MODEL

Output: $y$
Predicted class

+
Count positive & negative words in sentence

If number of positive words > number of negative words:
\[
\hat{y} = \text{positive}\n\]
Else:
\[
\hat{y} = \text{negative}\n\]
Problems with threshold classifier

- How do we get list of positive/negative words?
  - Addressed by learning a classifier

- 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 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, 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.

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

\( \hat{y} = +1 \) positive review

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

---

Simple linear classifier

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

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

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

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

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

- 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, \( \text{tf-idf(“awful”)} \)

  ... 

- Feature \( D+1 = h_D(x) \) ... some other function of \( x[1], \ldots, x[d] \)

Training Data \( \x \) \( \rightarrow \) Feature extraction \( \rightarrow \) \( \mathbf{h}(\mathbf{x}) \) \( \rightarrow \) ML model

\( \hat{y} = \text{sign}(\hat{\mathbf{w}}^T \mathbf{h}(\mathbf{x})) \) (either -1 or +1)

\( \hat{\mathbf{w}} \)

ML algorithm

Quality metric

ML algorithm

Training Data \( \x \) \( \rightarrow \) Feature extraction \( \rightarrow \) \( \mathbf{h}(\mathbf{x}) \) \( \rightarrow \) ML model

\( \hat{y} = \text{sign}(\hat{\mathbf{w}}^T \mathbf{h}(\mathbf{x})) \) (either -1 or +1)

\( \hat{\mathbf{w}} \)

ML algorithm

Quality metric
Decision boundaries

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

<table>
<thead>
<tr>
<th>Input</th>
<th>Coefficient</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>#awesome</td>
<td>$w_0$</td>
<td>0.0</td>
</tr>
<tr>
<td>#awful</td>
<td>$w_1$</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>$w_2$</td>
<td>-1.5</td>
</tr>
</tbody>
</table>

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

Decision boundary separates + and predictions

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></td>
<td>$w_2$</td>
<td>-1.5</td>
</tr>
</tbody>
</table>

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

Decision boundary separates + and predictions
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_1</td>
<td>1.0</td>
</tr>
<tr>
<td>#awful</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)...
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>
</tbody>
</table>

Data set: 
- (x,y) 
  - (Sentence1, -) 
  - (Sentence2, +) 
  - ...

Training set 
Learn classifier 
Evaluate?
Classification error

Learned classifier

\[ \hat{y} = \] Correct

Test example

(Sushi was great,)

Mistakes

Correct

Hide label

Classification error & accuracy

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

- Often, measure accuracy
  - Fraction of correct predictions
  \[ \text{accuracy} = \frac{\# \text{ of correct}}{\text{Total \# of sentences}} \]
  - 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)
    \[ \text{accuracy} = 0.5 \]

• For k classes, accuracy = 1/k
  – \( \frac{1}{3} \) for 3 classes, \( \frac{1}{4} \) 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>+ True Positive</td>
</tr>
<tr>
<td>-</td>
<td>- False Positive</td>
</tr>
<tr>
<td>+</td>
<td>- False Negative (FN)</td>
</tr>
<tr>
<td>-</td>
<td>+ True Negative</td>
</tr>
</tbody>
</table>
Cost of different types of mistakes can be different (& high) in some applications

- **Spam filtering**:
  - False negative: Annoying
  - False positive: Email lost

- **Medical diagnosis**:
  - False negative: Disease not treated depends on context
  - False positive: Wasteful treatment

**Confusion matrix – binary classification**

<table>
<thead>
<tr>
<th>True label</th>
<th>Predicted label</th>
</tr>
</thead>
<tbody>
<tr>
<td>True Negative</td>
<td>50</td>
</tr>
<tr>
<td>False Negative</td>
<td>5</td>
</tr>
</tbody>
</table>

100 test examples

Accuracy = \( \frac{50 + 35}{100} = 0.85 \)
Confusion matrix – multiclass classification

<table>
<thead>
<tr>
<th>True label</th>
<th>Healthy</th>
<th>Cold</th>
<th>Flu</th>
</tr>
</thead>
<tbody>
<tr>
<td>Healthy</td>
<td>60</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>Cold</td>
<td>4</td>
<td>12</td>
<td>4</td>
</tr>
<tr>
<td>Flu</td>
<td>0</td>
<td>2</td>
<td>8</td>
</tr>
</tbody>
</table>

Accuracy = \frac{60}{100} = 0.8

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

Even with infinite data, test error will not go to 0
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

Linear classifiers:

Logistic regression
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?

<table>
<thead>
<tr>
<th>&quot;The sushi &amp; everything else were awesome!&quot;</th>
<th>&quot;The sushi was good, the service was OK&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Definite +1</strong></td>
<td><strong>Not sure</strong></td>
</tr>
<tr>
<td>**P(y=+1</td>
<td>x= &quot;The sushi &amp; everything else were awesome!&quot;)** = 0.99</td>
</tr>
</tbody>
</table>

Many classifiers provide a degree of certainty:

\[ P(y|x) \]

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

Predict most likely class

\( \hat{P}(y|x) \) = 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