CSE/STAT 416

Classification

Tanmay Shah
University of Washington
April 10, 2024

❓ Questions? Raise hand or sli.do #cs416
⏰ Before Class: Does a straw have two holes or one?
🎵 Listening to: nothing, enjoy the calm
We have now finished the “Regression” component of the course!
Next two weeks (4 lectures): Classification
HW1 due tomorrow 11:59PM
  - Up to Sat 4/13 11:59PM if you use late days
HW2 released Fri
1. Housing Prices - Regression
   - Regression Model
   - Assessing Performance
   - Ridge Regression
   - LASSO

2. Sentiment Analysis – Classification
   - Classification Overview
   - Logistic Regression
Regression vs. Classification

Regression problems involve predicting **continuous values**.
- E.g., house price, student grade, population growth, etc.

Classification problems involve predicting **discrete labels**
- e.g., spam detection, object detection, loan approval, etc.
Spam Filtering

Input: $x$

Text of email
Sender
Subject

Output: $y$

Spam

Not Spam (ham)
Object Detection

Input: $x$

Pixels

Output: $y$

Class
(+ Probability)

Top Predictions
- Labrador retriever
- golden retriever
- redbone
- bloodhound
- Rhodesian ridgeback
ML Pipeline

- Training Data
- Pre-Processing
- ML model
- Optimization algorithm
- Quality metric
In our example, we want to classify a restaurant review as positive or negative.
Converting Text to Numbers (Vectorizing):

Bag of Words

Idea: One feature per word!

Example: "Sushi was great, the food was awesome, but the service was terrible"

<table>
<thead>
<tr>
<th>sushi</th>
<th>was</th>
<th>great</th>
<th>the</th>
<th>food</th>
<th>awesome</th>
<th>but</th>
<th>service</th>
<th>terrible</th>
</tr>
</thead>
</table>

This has to be too simple, right?

Stay tuned (today and Wed) for issues that arise and how to address them 😊
**Pre-Processing: Sample Dataset**

<table>
<thead>
<tr>
<th>Review</th>
<th>Sentiment</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Sushi was great, the food was awesome, but the service was terrible”</td>
<td>+1</td>
</tr>
<tr>
<td>“Terrible food; the sushi was rancid.”</td>
<td>-1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sushi</th>
<th>was</th>
<th>great</th>
<th>the</th>
<th>food</th>
<th>awesome</th>
<th>but</th>
<th>service</th>
<th>terrible</th>
<th>rancid</th>
<th>Sentiment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>+1</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>-1</td>
</tr>
</tbody>
</table>
How to Implement Sentiment Analysis?

**Attempt 1:** Simple Threshold Analysis

**Attempt 2:** Linear Classifier

**Attempt 3 (Wed):** Logistic Regression
**Attempt 1:**

**Simple Threshold Classifier**

**Idea:** Use a list of good words and bad words, classify review by the most frequent type of word

<table>
<thead>
<tr>
<th>Word</th>
<th>Good?</th>
</tr>
</thead>
<tbody>
<tr>
<td>sushi</td>
<td>None</td>
</tr>
<tr>
<td>was</td>
<td>None</td>
</tr>
<tr>
<td>great</td>
<td>Good</td>
</tr>
<tr>
<td>the</td>
<td>None</td>
</tr>
<tr>
<td>food</td>
<td>None</td>
</tr>
<tr>
<td>but</td>
<td>None</td>
</tr>
<tr>
<td>awesome</td>
<td>Good</td>
</tr>
<tr>
<td>service</td>
<td>None</td>
</tr>
<tr>
<td>terrible</td>
<td>Bad</td>
</tr>
<tr>
<td>rancid</td>
<td>Bad</td>
</tr>
</tbody>
</table>

**Simple Threshold Classifier**

Input $x$: Sentence from review

- Count the number of positive and negative words, in $x$

  - If $\text{num}_{\text{positive}} > \text{num}_{\text{negative}}$:
    - $\hat{y} = +1$
  
  - Else:
    - $\hat{y} = -1$

**Example:** "Sushi was great, the food was awesome, but the service was terrible"
Limitations of Attempt 1 (Simple Threshold Classifier)

Words have different degrees of sentiment.
- Awesome > Great
- How can we weigh them differently?

Single words are not enough sometimes...
- “Good” → Positive
- “Not Good” → Negative

How do we get list of positive/negative words?
Words Have Different Degrees of Sentiments

What if we generalize good/bad to a numeric weighting per word?

<table>
<thead>
<tr>
<th>Word</th>
<th>Good?</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>sushi</td>
<td>None</td>
<td>0</td>
</tr>
<tr>
<td>was</td>
<td>None</td>
<td>0</td>
</tr>
<tr>
<td>great</td>
<td>Good</td>
<td>1</td>
</tr>
<tr>
<td>the</td>
<td>None</td>
<td>0</td>
</tr>
<tr>
<td>food</td>
<td>None</td>
<td>0</td>
</tr>
<tr>
<td>but</td>
<td>None</td>
<td>0</td>
</tr>
<tr>
<td>awesome</td>
<td>Good</td>
<td>2</td>
</tr>
<tr>
<td>service</td>
<td>None</td>
<td>0</td>
</tr>
<tr>
<td>terrible</td>
<td>Bad</td>
<td>-1</td>
</tr>
<tr>
<td>rancid</td>
<td>Bad</td>
<td>-2</td>
</tr>
</tbody>
</table>
How do we get the word weights?

What if we learn them from the data?

In linear regression we learnt the weights for each feature. Can we do something similar here?

<table>
<thead>
<tr>
<th>Word</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>sushi</td>
<td>$w_1$</td>
</tr>
<tr>
<td>was</td>
<td>$w_2$</td>
</tr>
<tr>
<td>great</td>
<td>$w_3$</td>
</tr>
<tr>
<td>the</td>
<td>$w_4$</td>
</tr>
<tr>
<td>food</td>
<td>$w_5$</td>
</tr>
<tr>
<td>awesome</td>
<td>$w_6$</td>
</tr>
<tr>
<td>but</td>
<td>$w_7$</td>
</tr>
<tr>
<td>service</td>
<td>$w_8$</td>
</tr>
<tr>
<td>terrible</td>
<td>$w_9$</td>
</tr>
</tbody>
</table>
**Attempt 2: Linear Classifier**

Idea: Use labelled training data to learn a weight for each word. Use weights to score a sentence.

Model:

\[ \hat{y}_i = \text{sign}(\text{Score}(x_i)) = \text{sign}(s_i) \]

\[ = \text{sign} \left( \sum_{j=0}^{D} w_j h_j(x_i) \right) = \text{sign}(w^T h(x_i)) \]

<table>
<thead>
<tr>
<th>Word</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>sushi</td>
<td>0</td>
</tr>
<tr>
<td>was</td>
<td>0</td>
</tr>
<tr>
<td>great</td>
<td>1</td>
</tr>
<tr>
<td>the</td>
<td>0</td>
</tr>
<tr>
<td>food</td>
<td>0</td>
</tr>
<tr>
<td>awesome</td>
<td>2</td>
</tr>
<tr>
<td>but</td>
<td>0</td>
</tr>
<tr>
<td>service</td>
<td>0</td>
</tr>
<tr>
<td>terrible</td>
<td>-1</td>
</tr>
</tbody>
</table>

"Sushi was great, the food was awesome, but the service was terrible"
Consider if only two words had non-zero coefficients

\[ \hat{s} = 1 \cdot \#\text{awesome} - 1.5 \cdot \#\text{awful} \]

<table>
<thead>
<tr>
<th>Word</th>
<th>Coefficient</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>( w_0 )</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<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>

![Diagram showing decision boundary](image-url)
$Score(x) = 1 \cdot \#\text{awesome} - 1.5 \cdot \#\text{awful}$

Generally, with classification we don’t us a plot like the 3d view since it’s hard to visualize, instead use 2d plot with decision boundary.
Decision Boundary with Score

Score(x) = 1 \cdot \text{#awesome} - 1.5 \cdot \text{#awful}
ML Pipeline

- Train/val/test-split
- Scaling / Normalization
- Feature Selection
  - All Subsets
  - Greedy
  - LASSO
- Linear Regression
  - Polynomial Regression
  - Ridge Regression
  - LASSO Regression

Training Data

Pre-Processing

x

ML model

ŷ

Optimization algorithm

y

- Gradient Descent
- MSE / RMSE
- L1 Regularization (LASSO)
- L2 Regularization (Ridge)

Overarching Concepts:
- Overfitting
- Bias / Variance
- Hyperparameter Tuning:
  - Validation Set
  - Cross-validation
In our example, we want to classify a restaurant review as positive or negative.

**Diagram:**
- **Input:** Sentence from review
- **Classifier Model**
- **Output:** Predicted class

**Equation:**
\[
\text{Input: } x \quad \rightarrow \quad \text{Classifier Model} \quad \rightarrow \quad \text{Output: } y
\]
**Attempt 1:**
**Simple Threshold Classifier**

**Idea:** Use a list of good words and bad words, classify review by the most frequent type of word.

<table>
<thead>
<tr>
<th>Word</th>
<th>Good?</th>
</tr>
</thead>
<tbody>
<tr>
<td>sushi</td>
<td>None</td>
</tr>
<tr>
<td>was</td>
<td>None</td>
</tr>
<tr>
<td>great</td>
<td>Good</td>
</tr>
<tr>
<td>the</td>
<td>None</td>
</tr>
<tr>
<td>food</td>
<td>None</td>
</tr>
<tr>
<td>but</td>
<td>None</td>
</tr>
<tr>
<td>awesome</td>
<td>Good</td>
</tr>
<tr>
<td>service</td>
<td>None</td>
</tr>
<tr>
<td>terrible</td>
<td>Bad</td>
</tr>
<tr>
<td>rancid</td>
<td>Bad</td>
</tr>
</tbody>
</table>

**Simple Threshold Classifier**

Input $x$: Sentence from review

Count the number of positive and negative words, in $x$

If $\text{num\_positive} > \text{num\_negative}$:

- $\hat{y} = +1$

Else:

- $\hat{y} = -1$

**Example:** "Sushi was great, the food was awesome, but the service was terrible"
Idea: Use labelled training data to learn a weight for each word. Use weights to score a sentence.

Model:
\[ \hat{y}_i = \text{sign}(\text{Score}(x_i)) = \text{sign}(s_i) \]
\[ = \text{sign}\left(\sum_{j=0}^{D} w_j h_j(x_i)\right) = \text{sign}(w^T h(x_i)) \]

<table>
<thead>
<tr>
<th>Word</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>sushi</td>
<td>0</td>
</tr>
<tr>
<td>was</td>
<td>0</td>
</tr>
<tr>
<td>great</td>
<td>1</td>
</tr>
<tr>
<td>the</td>
<td>0</td>
</tr>
<tr>
<td>food</td>
<td>0</td>
</tr>
<tr>
<td>awesome</td>
<td>2</td>
</tr>
<tr>
<td>but</td>
<td>0</td>
</tr>
<tr>
<td>service</td>
<td>0</td>
</tr>
<tr>
<td>terrible</td>
<td>-1</td>
</tr>
</tbody>
</table>

"Sushi was great, the food was awesome, but the service was terrible"
Consider if only two words had non-zero coefficients

\[
\hat{s} = 1 \cdot \text{#awesome} - 1.5 \cdot \text{#awful}
\]

<table>
<thead>
<tr>
<th>Word</th>
<th>Coefficient</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>(w_0)</td>
<td>0.0</td>
<td></td>
</tr>
<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>

Decision Boundary
What happens to the decision boundary if we add an intercept?

\[ Score(x) = 1.0 + 1 \cdot \#awesome - 1.5 \cdot \#awful \]

Which graph shows the new decision boundary (black)?

Describe in English what we are doing in terms of the model’s predictions.
What happens to the decision boundary if we add an intercept?

\[ \text{Score}(x) = 1.0 + 1 \cdot \#\text{awesome} - 1.5 \cdot \#\text{awful} \]

Which graph shows the new decision boundary (black)?

Describe in English what we are doing in terms of the model’s predictions.
Score \( x \) = 1 \cdot \#awesome - 1.5 \cdot \#awful

Generally, with classification we don't use a plot like the 3d view since it's hard to visualize, instead use 2d plot with decision boundary
What if we want to use a more complex decision boundary?
- Need more complex model/features! (Come back Wed)
Single Words Are Sometimes Not Enough!

What if instead of making each feature one word, we made it two?
- **Unigram**: a sequence of one word
- **Bigram**: a sequence of two words
- **N-gram**: a sequence of n-words

"Sushi was good, the food was good, the service was not good”

<table>
<thead>
<tr>
<th>sushi</th>
<th>was</th>
<th>good</th>
<th>the</th>
<th>food</th>
<th>service</th>
<th>not</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Longer sequences of words results in more context, more features, and a greater chance of overfitting.
Evaluating Classifiers
ML Pipeline

Training Data \(x\) \rightarrow \text{Pre-Processing} \rightarrow h(x) \rightarrow \text{ML model} \rightarrow \hat{y} \rightarrow \text{Optimization algorithm} \rightarrow \hat{w} \rightarrow \text{Quality metric} \rightarrow y
Classification Error

Ratio of examples where there was a mistaken prediction

What’s a mistake?
- If the true label was positive ($y = +1$), but we predicted negative ($\hat{y} = -1$)
- If the true label was negative ($y = -1$), but we predicted positive ($\hat{y} = +1$)

Classification Error

Classification Accuracy
What’s a good accuracy?

For binary classification:
  Should at least beat random guessing...
  Accuracy should be at least 0.5

For multi-class classification ($k$ classes):
  Should still beat random guessing
  Accuracy should be at least: $1 / k$
  - 3-class: 0.33
  - 4-class: 0.25
  - ...

Besides that, higher accuracy means better, right?
Imagine I made a “Dummy Classifier” for detecting spam
The classifier ignores the input, and always predicts spam.
This actually results in 90% accuracy! Why?
- Most emails are spam...

This is called the **majority class classifier**.

A classifier as simple as the majority class classifier can have a high accuracy if there is a **class imbalance**.

A class imbalance is when one class appears much more frequently than another in the dataset.

This might suggest that accuracy isn’t enough to tell us if a model is a good model.
Always digging in and ask critical questions of your accuracy.

Is there a **class imbalance**?

How does it compare to a baseline approach?
- Random guessing
- Majority class
- ...

Most important: **What does my application need?**
- What’s good enough for user experience?
- What is the impact of a mistake we make?
Brain Break
Confusion Matrix

For binary classification, there are only two types of mistakes:

\[
\hat{y} = +1, \ y = -1 \\
\hat{y} = -1, \ y = +1
\]

Generally we make a **confusion matrix** to understand mistakes.

![Confusion Matrix Diagram]

**Tip on remembering:** complete the sentence “My prediction was a …”
## Confusion Matrix Example

<table>
<thead>
<tr>
<th>True Label</th>
<th>Predicted Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

- **True Positive (TP)**: Correctly predicted positive cases.
- **False Negative (FN)**: Missed positive cases.
- **False Positive (FP)**: Incorrectly predicted positive cases.
- **True Negative (TN)**: Correctly predicted negative cases.
Which is Worse?

What’s worse, a false negative or a false positive? It entirely depends on your application!

Detecting Spam
- False Negative: Annoying
- False Positive: Email lost

Medical Diagnosis
- False Negative: Disease not treated
- False Positive: Wasteful treatment

In almost every case, how treat errors depends on your context.
Errors and Fairness

We mentioned on the first day how ML is being used in many contexts that impact crucial aspects of our lives.

Models making errors is a given, what we do about that is a choice:

- Are the errors consequential enough that we shouldn’t use a model in the first place?
- Do different demographic groups experience errors at different rates?
  - If so, we would hopefully want to avoid that model!

Will talk more about how to define whether or not a model is fair / discriminatory next week. Will use these notions of error as a starting point!
## Binary Classification Measures

**Notation**

\[
C_{TP} = \#TP, \quad C_{FP} = \#FP, \quad C_{TN} = \#TN, \quad C_{FN} = \#FN
\]

\[
N = C_{TP} + C_{FP} + C_{TN} + C_{FN}
\]

\[
N_P = C_{TP} + C_{FN}, \quad N_N = C_{FP} + C_{TN}
\]

<table>
<thead>
<tr>
<th>Metric</th>
<th>Formula</th>
</tr>
</thead>
</table>
| **Error Rate**          | \[
\frac{C_{FP} + C_{FN}}{N}
\]                                      |
| **Accuracy Rate**       | \[
\frac{C_{TP} + C_{TN}}{N}
\]                                      |
| **False Positive rate (FPR)** | \[
\frac{C_{FP}}{N_N}
\]                                      |
| **False Negative Rate (FNR)** | \[
\frac{C_{FN}}{N_P}
\]                                      |
| **True Positive Rate or Recall** | \[
\frac{C_{TP}}{N_P}
\]                                      |
| **Precision**           | \[
\frac{C_{TP}}{C_{TP} + C_{FP}}
\]                                      |
| **F1-Score**            | \[
\frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}
\]                                      |

See more!
### Multiclass Confusion Matrix

Consider predicting *(Healthy, Cold, Flu)*

<table>
<thead>
<tr>
<th>True Label</th>
<th>Predicted Label</th>
</tr>
</thead>
<tbody>
<tr>
<td></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>
Suppose we trained a classifier and computed its confusion matrix on the training dataset. **Is there a class imbalance in the dataset and if so, which class has the highest representation?**

<table>
<thead>
<tr>
<th>True Label</th>
<th>Pupper</th>
<th>Doggo</th>
<th>Woofer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pupper</td>
<td>2</td>
<td>27</td>
<td>4</td>
</tr>
<tr>
<td>Doggo</td>
<td>4</td>
<td>25</td>
<td>4</td>
</tr>
<tr>
<td>Woofer</td>
<td>1</td>
<td>30</td>
<td>2</td>
</tr>
</tbody>
</table>
Suppose we trained a classifier and computed its confusion matrix on the training dataset. **Is there a class imbalance in the dataset and if so, which class has the highest representation?**

<table>
<thead>
<tr>
<th>True Label</th>
<th>Predicted Label</th>
<th>Pupper</th>
<th>Doggo</th>
<th>Woofer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pupper</td>
<td>2</td>
<td>27</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Doggo</td>
<td>4</td>
<td>25</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Woofer</td>
<td>1</td>
<td>30</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>
Learning Theory
How much data?

The more the merrier
  But data quality is also an extremely important factor

Theoretical techniques can bound how much data is needed
  Typically too loose for practical applications
  But does provide some theoretical guarantee

In practice
  More complex models need more data
Learning Curve

How does the true error of a model relate to the amount of training data we give it?

Hint: We’ve seen this picture before
Learning Curve

What if we use a more complex model?

True error

Amount of training data
Next Time

We will address the issues highlighted with the Linear Classifier approach from today by predicting the probability of a sentiment, rather than the sentiment itself.

\[ P(y|x) \]

Normally assume some structure on the probability (e.g., linear)

\[ P(y|x,w) \approx w^T x \]

Use machine learning algorithm to learn approximate \( \hat{w} \) such that \( \hat{P}(y|x) \) is close to \( P(y|x) \), where:

\[ \hat{P}(y|x) = P(y|x, \hat{w}) \]
Recap

**Theme:** Describe high level idea and metrics for classification

**Ideas:**

- Applications of classification
- Linear classifier
- Decision boundaries
- Classification error / Classification accuracy
- Class imbalance
- Confusion matrix
- Learning theory