Boosting question

“Can a set of weak learners be combined to create a stronger learner?” Kearns and Valiant (1988)

Yes! Schapire (1990)

Boosting

Amazing impact: • simple approach • widely used in industry • wins most Kaggle competitions
**Ensemble methods:** Each classifier “votes” on prediction

\[ x_i = (\text{Income}=$120K, \text{Credit}=\text{Bad}, \text{Savings}=$50K, \text{Market}=\text{Good}) \]

\[ f_1(x_i) = 1 \quad f_2(x_i) = -1 \quad f_3(x_i) = -1 \quad f_4(x_i) = 1 \]

Combine?

\[ F(x_i) = \text{sign}(w_1 f_1(x_i) + w_2 f_2(x_i) + w_3 f_3(x_i) + w_4 f_4(x_i)) \]

**Boosting = Greedy learning ensembles from data**

Higher weight for points where \( f_1(x) \) is wrong
**AdaBoost:** learning ensemble

[Freund & Schapire 1999]

- **Start with same weight for all points:** $\alpha_i = 1/N$

- **For $t = 1, \ldots, T$**
  - Learn $f_t(x)$ with data weights $\alpha_i$
  - Compute coefficient $\hat{w}_t$
  - Recompute weights $\alpha_i$

- **Final model predicts by:**

$$\hat{y} = sign \left( \sum_{t=1}^{T} \hat{w}_t f_t(x) \right)$$

---

**Problem 1:** How much do I trust $f_t$?

**Problem 2:** Weigh mistakes more

---

**AdaBoost:** learning ensemble

- **Start with same weight for all points:** $\alpha_i = 1/N$

- **For $t = 1, \ldots, T$**
  - Learn $f_t(x)$ with data weights $\alpha_i$
  - Compute coefficient $\hat{w}_t$
  - Recompute weights $\alpha_i$
  - Normalize weights $\alpha_i$

- **Final model predicts by:**

$$\hat{y} = sign \left( \sum_{t=1}^{T} \hat{w}_t f_t(x) \right)$$
Learning from weighted data

Often, learning from weighted data treats data point \( i \) as \( \alpha_i \) replicates of that data point.

<table>
<thead>
<tr>
<th>Credit</th>
<th>Income</th>
<th>( y )</th>
<th>Weight ( \alpha )</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>$130K</td>
<td>Safe</td>
<td>0.5</td>
</tr>
<tr>
<td>B</td>
<td>$80K</td>
<td>Risky</td>
<td>1.5</td>
</tr>
<tr>
<td>C</td>
<td>$110K</td>
<td>Risky</td>
<td>1.2</td>
</tr>
<tr>
<td>A</td>
<td>$110K</td>
<td>Safe</td>
<td>0.8</td>
</tr>
<tr>
<td>A</td>
<td>$90K</td>
<td>Safe</td>
<td>0.6</td>
</tr>
<tr>
<td>B</td>
<td>$120K</td>
<td>Safe</td>
<td>0.7</td>
</tr>
<tr>
<td>C</td>
<td>$30K</td>
<td>Risky</td>
<td>3</td>
</tr>
<tr>
<td>C</td>
<td>$60K</td>
<td>Risky</td>
<td>2</td>
</tr>
<tr>
<td>B</td>
<td>$95K</td>
<td>Safe</td>
<td>0.8</td>
</tr>
<tr>
<td>A</td>
<td>$60K</td>
<td>Safe</td>
<td>0.7</td>
</tr>
<tr>
<td>A</td>
<td>$98K</td>
<td>Safe</td>
<td>0.9</td>
</tr>
</tbody>
</table>
Gradient ascent for logistic regression on **unweighted data**

\[
\text{init } w^{(1)} = 0 \text{ (or randomly, or smartly), } t = 1
\]

while \( \| \nabla \ell(w^{(t)}) \| > \varepsilon \)

for \( j = 0, \ldots, D \)

\[
\text{partial}[j] = \sum_{i=1}^{N} h_j(x_i) \left( I[y_i = +1] - P(y = +1 | x_i, w^{(t)}) \right)
\]

\[
w_j^{(t+1)} \leftarrow w_j^{(t)} + \eta \text{ partial}[j]
\]

\[ t \leftarrow t + 1 \]

---

Modify the logistic regression gradient update for **weighted data**

\[
w_j^{(t+1)} \leftarrow w_j^{(t)} + \eta \sum_{i=1}^{N} \alpha_i h_j(x_i) \left( I[y_i = +1] - P(y = +1 | x_i, w^{(t)}) \right)
\]

Sum over data points

Weigh each point by \( \alpha_i \)
How to learn the “best” decision stump?

Goal:
Choose best feature (categorical input) or feature/threshold pair (real-valued input)

Questions:
1. For given feature or feature/threshold, how to determine classifier output using weighted data?
2. For that classifier, how to compute its accuracy on the training data?
3. For real-valued features, how to select the best threshold on weighted data?
4. Based on the above, select the best decision stump

Learning a decision stump on weighted data

For a given potential split, learn classifier:
How to learn the “best” decision stump?

**Goal:**
Choose best feature (categorical input) or feature/threshold pair (real-valued input)

**Questions:**
1. For given feature or feature/threshold, how to determine classifier output using weighted data?
2. For that classifier, how to compute its accuracy on the training data?
3. For real-valued features, how to select the best threshold on weighted data?
4. Based on the above, select the best decision stump

<table>
<thead>
<tr>
<th>Credit</th>
<th>Income</th>
<th>y</th>
<th>Weight α</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>$130K</td>
<td>Safe</td>
<td>0.5</td>
</tr>
<tr>
<td>B</td>
<td>$80K</td>
<td>Risky</td>
<td>1.5</td>
</tr>
<tr>
<td>C</td>
<td>$110K</td>
<td>Risky</td>
<td>1.2</td>
</tr>
<tr>
<td>A</td>
<td>$110K</td>
<td>Safe</td>
<td>0.8</td>
</tr>
<tr>
<td>A</td>
<td>$90K</td>
<td>Safe</td>
<td>0.6</td>
</tr>
<tr>
<td>B</td>
<td>$120K</td>
<td>Safe</td>
<td>0.7</td>
</tr>
<tr>
<td>C</td>
<td>$30K</td>
<td>Risky</td>
<td>3</td>
</tr>
<tr>
<td>C</td>
<td>$60K</td>
<td>Risky</td>
<td>2</td>
</tr>
<tr>
<td>B</td>
<td>$95K</td>
<td>Safe</td>
<td>0.8</td>
</tr>
<tr>
<td>A</td>
<td>$60K</td>
<td>Safe</td>
<td>0.7</td>
</tr>
<tr>
<td>A</td>
<td>$98K</td>
<td>Safe</td>
<td>0.9</td>
</tr>
</tbody>
</table>

Learning a decision stump on weighted data

**Compute weighted error:**

Income > $100K?  
Yes | Safe  
No | Risky

Weighted error = \[rac{\text{total weight of mistakes}}{\text{total weight}}\]
How to learn the “best” decision stump?

Goal:
Choose best feature (categorical input) or feature/threshold pair (real-valued input)

Questions:
1. For given feature or feature/threshold, how to determine classifier output using weighted data?
2. For that classifier, how to compute its accuracy on the training data?
3. For real-valued features, how to select the best threshold on weighted data?
4. Based on the above, select the best decision stump

Finding the best threshold split

Infinite possible values of t

Income = t*

Income < t*  Income >= t*

Income

$10K

Safe

Risky

$120K
Consider a threshold between points

Same weighted error for any threshold split between $v_A$ and $v_B$

Only need to consider mid-points

Finite number of splits to consider
How to learn the “best” decision stump?

Goal:
Choose best feature (categorical input) or feature/threshold pair (real-valued input)

Questions:
1. For given feature or feature/threshold, how to determine classifier output using weighted data?
2. For that classifier, how to compute its accuracy on the training data?
3. For real-valued features, how to select the best threshold on weighted data?
4. Based on the above, select the best decision stump

Finding best next decision stump $f_t(x)$

Which weak learner to choose?

Consider splitting on each feature:

- **Income>$100K?**
  - Yes
    - Safe
    - Risky
  - No
    - Safe
    - Risky
  - Weighted error = 0.33

- **Credit history?**
  - Bad
    - Risky
    - Safe
  - Good
    - Risky
    - Safe
  - Weighted error = 0.35

- **Savings>$100K?**
  - Yes
    - Safe
    - Risky
  - No
    - Risky
    - Safe
  - Weighted error = 0.4

- **Market conditions?**
  - Bad
    - Risky
    - Safe
  - Good
    - Risky
    - Safe
  - Weighted error = 0.45

$$f_t = \begin{cases} 
\text{Yes} & \text{Income>$100K?} \\
\text{No} & \text{Income>$100K?} 
\end{cases}$$
Revising AdaBoost example:
A visualization

Boosted decision stumps

- Start same weight for all points: $\alpha_i = 1/N$
- For $t = 1, ..., T$
  - Learn $f_t(x)$: pick decision stump with lowest weighted training error according to $\alpha_i$
    - Compute coefficient $\hat{w}_t$
    - Recompute weights $\alpha_i$
    - Normalize weights $\alpha_i$
- Final model predicts by:
  \[
  \hat{y} = \text{sign} \left( \sum_{t=1}^{T} \hat{w}_t f_t(x) \right)
  \]
t=1: Just learn a classifier on original data

Original data

Learned decision stump $f_1(x)$

Determine weight (trust) of classifier

$$\hat{w}_t = \frac{1}{2} \ln \left( \frac{1 - \text{weighted_error}(f_t)}{\text{weighted_error}(f_t)} \right)$$

= 0.62
Updating weights $\alpha_i$

Increase weight $\alpha_i$ of misclassified points

Learned decision stump $f_1(x)$

New data weights $\alpha_i$

\[ \alpha_i \left\{ \begin{array}{ll}
\alpha_i e^{-\hat{w}_t} & \text{if } f_t(x_i) = y_i \\
\alpha_i e^{\hat{w}_t} & \text{if } f_t(x_i) \neq y_i
\end{array} \right. \]

Normalize weights...

\[ \alpha_i' \left\{ \alpha_i \cdot \frac{1}{\sum_{j=1}^{N} \alpha_j} \right. \]

New data weights $\alpha_i'$


**t=2: Learn classifier on weighted data**

Weighted data: using $\alpha_i$ chosen in previous iteration

Learned decision stump $f_2(x)$ on weighted data

---

**Ensemble becomes weighted sum of learned classifiers**

$$\text{Score}(x) = \hat{w}_1 f_1(x) + \hat{w}_2 f_2(x)$$

$$\hat{y} = \text{sign}(\text{score}(x))$$

Define $\hat{y} = -1$

definite $\hat{y} = 1$

uncertain $\hat{y} = \pm 1$
Decision boundary of ensemble classifier after 30 iterations (30 classifiers, T=30)

training_error = 0

decision boundary is crazy!

probably overfitting

AdaBoost:
Revisiting convergence & overfitting
AdaBoost Theorem

Under some technical conditions...

Training error of boosted classifier $\to 0$ as $T \to \infty$

May oscillate a bit

But will generally decrease, & eventually become 0!

Training error

Iterations of boosting

---

Decision trees on loan data

39% test error

8% training error

Overfitting

Boosted decision stumps on loan data

32% test error

28.5% training error

Better fit & lower test error
Boosting tends to be robust to overfitting

Test set performance about constant for many iterations \( \Rightarrow \) Less sensitive to choice of \( T \)

But boosting will eventually overfit, so must choose max number of components \( T \)

Best test error around 31%

Test error eventually increases to 33% (overfits)
How do we decide when to stop boosting?

Choosing T?

- Not on training data
  - Not useful: training error improves as T increases

- Never ever ever on test data
  - If dataset is large

- Validation set
  - For smaller data

Like selecting parameters in other ML approaches (e.g., λ in regularization)

Summary of boosting
Variants of boosting and related algorithms

There are hundreds of variants of boosting, most important:

**Gradient boosting**
- Like AdaBoost, but useful beyond basic classification

Many other approaches to learn ensembles, most important:

**Random forests**
- **Bagging**: Pick random subsets of the data
  - Learn a tree in each subset
  - Average predictions
- Simpler than boosting & easier to parallelize
- Typically higher error than boosting for same # of trees (# iterations T)

Impact of boosting (*spoiler alert... HUGE IMPACT*)

**Amongst most useful ML methods ever created**

**Extremely useful in computer vision**
- Standard approach for face detection, for example (though deep learning is making serious inroads)

**Used by most winners of ML competitions (Kaggle, KDD Cup,...)**
- Malware classification, credit fraud detection, ads click through rate estimation, sales forecasting, ranking webpages for search, Higgs boson detection,...

**Most deployed ML systems use model ensembles**
- Coefficients chosen manually, with boosting, with bagging, or others
What you can do now...

- Identify notion ensemble classifiers
- Formalize ensembles as weighted combination of simpler classifiers
- Outline the boosting framework – sequentially learn classifiers on weighted data
- Describe the AdaBoost algorithm
  - Learn each classifier on weighted data
  - Compute coefficient of classifier
  - Recompute data weights
  - Normalize weights
- Implement AdaBoost to create an ensemble of decision stumps

Midterm survey feedback
Who this course was intended for

This year, we **strictly** enforced prereqs

Not the plan in future years

Trying to develop a **general audience** ML course
The gap we’re filling

- CSE 446
  - CSE majors
  - Technical course
- STAT 435
  - STAT majors
  - Technical course
- CSE 416
  - A course for everyone else...

The problem:
- "Everyone else" is really broad
- Some have math background, some have programming background, some have both, some have neither

Our “solution”

- Most people learning ML are doing so to be practitioners, not researchers
- Focus on important ideas, without getting bogged down in the math
  - Sometimes present theoretical results (e.g., adaboost theorem), but not proofs
  - Explain mathematical concepts (e.g., bias-variance tradeoff), but at an intuitive level of what these things really mean (e.g., rather than spending a lecture or two manipulating expectations)
- This choice provides an ability to cover more topics
  - Will cover recommender systems and spend more time on deep learning than in CSE 446 or STAT 435 (amongst many other things, e.g., precision-recall, multiclass classification, one-hot encoding, etc.)
- Utilize pre-implemented tools (e.g., TuriCreate)
  - Helps students with weaker programming backgrounds focus on concepts (analyzing model choices, outputs, etc.) rather than coding details

Note: Even ML researchers these days use pre-implemented libraries. Very few are coding up algorithms from scratch.
Will learn about the ML pipeline...

Level of the course

**Motto:**
*tough concepts made intuitive and applicable*

- minimize prereq knowledge
- maximize ability to develop and deploy
- learn concepts through case studies
Who you are (sample size 31 out of 107)

- Math: 35.5%
- ACMS: 22.8%
- Info: 19.4%
- HCDE: 12.9%
- Pre-CSE: 9.7%
- Pre-other: 6.5%
- Linguistics: 6.5%
- Econ: 6.5%
- ChemE: 3.2%
- Pre-Info: 3.2%
- Geography: 3.2%
- Pre-Business: 3.2%
- Philosophy: 3.2%

Does not reflect our long-term target audience distribution

Also, be aware of selection bias on opinions on course

Some things you like

- Common responses:
  - Lectures (slides+annotations, explanations, good depth+pace)
  - Quiz sections (good for reviewing and clarifying)
  - Focus on applications and higher-level understanding
  - Responsive/approachable staff
  - Jupyter
  - Good pace, depth, and content
  - Programming assignments (comprehensive, helpful, not too focused on coding details, exploring cool models on real data)
  - Concept quizzes are useful
Some things you wish were different

• Common response:
  – *Lectures too slow*
  – Slow down a bit
  – *Want more math!!!*
  – *Assignments too easy* (this was listed as a “pro” by someone 😊)
    • Wish we didn’t use Jupyter
    • Want more coding and instruction on implementing from scratch
    • Want to do more raw data processing and learn more about this in section
    • Want clearer sense of accomplishment at the end
  – Want more clarity and help on assignments
    • Wording can be vague
  – Stronger microphone

What we plan to address

• Assignments:
  – Will give less starter code for tasks done in past assignments (already had this in the works, but will adjust even more than initially planned)
  – Will try to include parts in the assignments that allow for more open-ended exploration
  – Will make sure take-home message of assignments is clear

• Lectures:
  – Not planning to make the in-class mathematical level harder
  – Will stop brushing off math in language used
  – Will provide [links to readings](#) with more mathematical depth
  – Will add [poll everywhere](#) to help spice things up
Evaluating classifiers: Precision & Recall

STAT/CSE 416: Machine Learning
Emily Fox
University of Washington
May 1, 2018

Using reviews to promote my restaurant

Goal: increase # guests by 30%

Need an automated, “authentic” marketing campaign

Reviews

Great quotes
“Easily best sushi in Seattle.”

Great spokespeople
How do I find sentences with positive sentiment?

All reviews for my restaurant

What are the positive things being said about my restaurant?

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.
**Sentiment classifier**

Input $x_i$: Easily best sushi in Seattle.

Output: $\hat{y}_i$

Easily best sushi in Seattle.

**Use the sentiment classifier model!**

Sentences from all reviews for my restaurant

- 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.

Sentences predicted to be positive

$\hat{y} = +1$

- Easily best sushi in Seattle.
- I like the interior decoration and the blackboard menu on the wall.
- All the sushi was delicious.
- The sushi was amazing, and the rice is just outstanding.

Sentences predicted to be negative

$\hat{y} = -1$

- The seaweed salad was just OK, vegetable salad was just ordinary.
- My wife tried their ramen and it was pretty forgettable.
- The service is somewhat hectic.

Show sentences with +1 prediction on website.
What does it mean for a classifier to be good?

Previously, we asked the question: “What is good accuracy?”
We explored accuracy of random classifier as baseline

- For binary classification:
  - Half the time, you’ll get it right! (on average)
    \[ \text{classification error} = 0.5 \]

- For k classes, \[ \text{error} = 1 - \frac{1}{k} \]
  - error = 0.666 for 3 classes, 0.75 for 4 classes,…

At the very, very, very least, you should healthily beat random…
Otherwise, it’s (usually) pointless…

We explored the pitfalls of imbalanced problems:
*Is 90% accuracy good? Depends …*

90% of sentences are negative!

90% accuracy by predicting every sentence is negative!!!

Amazing “performance” but not useful for me right now!
Automated marketing campaign cares about something else...

Website shows 10 sentences from recent reviews

- **PRECISION**: Did I (mistakenly) show a negative sentence???
- **RECALL**: Did I not show a (great) positive sentence???

Accuracy doesn’t capture these issues well...

---

**Precision**:
Fraction of positive predictions that are actually positive
What fraction of the positive predictions are correct?

Sentences predicted to be positive: $\hat{y}_i = +1$

- Easily best sushi in Seattle: ✓
- The seaweed salad was just OK, vegetable salad was just ordinary: ✗
- I like the interior decoration and the blackboard menu on the wall: ✓
- The service is somewhat hectic: ✗
- The sushi was amazing, and the rice is just outstanding: ✓
- All the sushi was delicious: ✓

Only 4 out of 6 sentences predicted to be positive are actually positive

Precision: Fraction of positive predictions that are actually positive

Subset of positive predictions that are actually positive

Positive sentences (correct predictions) $y_i = +1$

Negative sentences (incorrect predictions) $y_i = -1$

All sentences predicted to be positive $\hat{y}_i = +1$
Types of error: Review

True label

Predicted label

<table>
<thead>
<tr>
<th>True label</th>
<th>Predicted label</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y_i=+1$</td>
<td>$\hat{y}_i=+1$</td>
</tr>
<tr>
<td>$y_i=-1$</td>
<td>$\hat{y}_i=-1$</td>
</tr>
</tbody>
</table>

Confusion matrix for sentiment analysis

Predicted sentiment

True sentiment

<table>
<thead>
<tr>
<th>True sentiment</th>
<th>Predicted sentiment</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y_i=+1$</td>
<td>$\hat{y}_i=+1$</td>
</tr>
<tr>
<td>$y_i=-1$</td>
<td>$\hat{y}_i=-1$</td>
</tr>
</tbody>
</table>
**Precision - Formula**

Fraction of positive predictions that are correct

\[
\text{precision} = \frac{\# \text{ true positives}}{\# \text{ true positives} + \# \text{ false positives}}
\]

- Best possible value : 1.0
- Worst possible value : 0.0

**Example: Calculating precision**

**Sentences predicted to be positive:** \( \hat{y}_i = +1 \)

- "Easily best sushi in Seattle." ✓
- "The seaweed salad was just OK, vegetable salad was just ordinary." ✗
- "I like the interior decoration and the blackboard menu on the wall." ✓
- "The service is somewhat hectic." ✗
- "The sushi was amazing, and the rice is just outstanding." ✓
- "All the sushi was delicious." ✓

4 correct 2 mistakes

\[
\text{precision} = \frac{\# \text{ true positives}}{\# \text{ true positives} + \# \text{ false positives}} = \frac{4}{4 + 2} = \frac{4}{6} = 0.67
\]
Why precision is important

Shown on website

Sentences predicted to be positive: $\hat{y}_i = +1$

- Easily best sushi in Seattle. ✓
- The seaweed salad was just OK, vegetable salad was just ordinary. ✗
- I like the interior decoration and the blackboard menu on the wall. ✓
- The service is somewhat hectic. ✗
- The sushi was amazing, and the rice is just outstanding. ✓
- All the sushi was delicious. ✓

High precision means positive predictions actually likely to be positive!

2 negative sentences shown to potential customers... 😞

Recall:
Fraction of positive data predicted to be positive
Did I find all the positive sentences?

Classifier MODEL

Sentences from all reviews for my restaurant

True positive sentences: $y_i = +1$

Predicted positive $\hat{y}_i = +1$

- Easily best sushi in Seattle.
- The seaweed salad was just OK, vegetable salad was just ordinary.
- I like the interior decoration and the blackboard menu on the wall.
- The service is somewhat hectic.
- The sushi was amazing, and the rice is just outstanding.
- All the sushi was delicious.

Predicted negative $\hat{y}_i = -1$

- The seaweed salad was just OK, vegetable salad was just ordinary.
- My wife tried their ramen and it was delicious.
- The service is somewhat hectic.
- My wife tried their ramen and it was pretty forgettable.
- The service was perfect.

What fraction of positive sentences were missed out?

Found 4 positive sentences

Model could not find 2 sentences that were actually positive

Missed 2 positive sentences

Predicted positive $\hat{y}_i = +1$

- Easily best sushi in Seattle.
- The seaweed salad was just OK, vegetable salad was just ordinary.
- I like the interior decoration and the blackboard menu on the wall.
- The service is somewhat hectic.
- The sushi was amazing, and the rice is just outstanding.
- All the sushi was delicious.

Predicted negative $\hat{y}_i = -1$

- The seaweed salad was just OK, vegetable salad was just ordinary.
- My wife tried their ramen and it was delicious.
- The service is somewhat hectic.
- My wife tried their ramen and it was pretty forgettable.
- The service was perfect.
Recall: Fraction of positive data predicted to be positive

- Sentences predicted to be positive (correct predictions) \( \hat{y}_i = +1 \)
- Sentences predicted to be negative (incorrect predictions) \( \hat{y}_i = -1 \)

Recall - Formula

Fraction of positive data points correctly classified

\[
\text{Recall} = \frac{\# \text{ true positives}}{\# \text{ true positives} + \# \text{ false negatives}}
\]

- Best possible value : 1.0
- Worst possible value : 0.0
Why is recall important?

Want to show positive sentences on website

High recall means positive data points are very likely to be discovered!

2 positive sentences not shown to potential customers... 😞

Precision-recall extremes
### Optimistic model: High recall, low precision

**Sentences from all reviews for my restaurant**

**Optimistic model**

- Predict almost everything as positive

**True positive sentences: \( y_i = +1 \)**

**Predicted positive \( \hat{y}_i = +1 \)**

- Easily best sushi in Seattle.
- The service is somewhat hectic.
- The sushi was amazing, and the rice is just outstanding.
- I like the interior decoration and the blackboard menu on the wall.
- All the sushi was delicious.
- The seaweed salad was just OK, vegetable salad was just ordinary.

**Predicted negative \( \hat{y}_i = -1 \)**

- The service is somewhat hectic.
- My wife tried their ramen and it was pretty forgettable.
- The service was perfect.
- My wife tried their ramen and it was delicious.

### Pessimistic model: High precision, low recall

**Sentences from all reviews for my restaurant**

**Pessimistic model**

- Predict positive only when very sure

**True positive sentences: \( y_i = +1 \)**

**Predicted positive \( \hat{y}_i = +1 \)**

- Easily best sushi in Seattle.
- The service is somewhat hectic.
- The sushi was amazing, and the rice is just outstanding.
- I like the interior decoration and the blackboard menu on the wall.
- All the sushi was delicious.
- The seaweed salad was just OK, vegetable salad was just ordinary.

**Predicted negative \( \hat{y}_i = -1 \)**

- The service is somewhat hectic.
- My wife tried their ramen and it was pretty forgettable.
- The service was perfect.
- My wife tried their ramen and it was delicious.
Balancing precision & recall

PESSIMISTIC MODEL
Finds few positive sentences, but includes no false positives

Want to find many positive sentences, but minimize risk of incorrect predictions!!

OPTIMISTIC MODEL
Finds all positive sentences, but includes many false positives

Tradeoff precision and recall
Can we tradeoff precision & recall?

Low precision, high recall

Optimistic Model
Predict almost everything as positive

High precision, low recall

Pessimistic Model
Predict positive only when very sure

How confident is your prediction?

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

Can be used to tradeoff precision and recall
Basic classifier

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

Pessimistic: High precision, low recall

If \( \hat{P}(y=+1|x_i) > 0.999 \):
\[
\hat{y}_i = +1
\]
Else:
\[
\hat{y}_i = -1
\]

Predict positive only when very sure
Optimistic: Low precision, high recall

Optimistic classifier

If \( \hat{P}(y=+1|x_i) > 0.001 \):
\[
\hat{y}_i = +1
\]
Else:
\[
\hat{y}_i = -1
\]

Sentence from review

Input: \( x_i \)

Predict positive almost always

Prediction probability threshold

Probability \( t \) above which model predicts true

Set \( \hat{y} = +1 \) if \( \hat{P}(y|x) \geq t \)
Example threshold values

\[ t = 0.99 \text{ (pessimistic)} \]

\[ t = 0.01 \text{ (optimistic)} \]

Tradeoff precision & recall with threshold

\[ t = 0 \rightarrow t = 1 \]

Low precision, high recall

High precision, low recall

Optimistic Model
Predict almost everything as positive

Pessimistic Model
Predict positive only when very sure
Precision-recall curve

The precision-recall curve

Classifier A

Pessimistic
Optimistic
What does the perfect algorithm look like?

Which classifier is better? A or B?
Which classifier is better? A or C?

How do we decide???

Compare algorithms

Often, reduce precision-recall to single number to compare algorithms
- F1 measure, area-under-the-curve (AUC), ...

Precision at k

Showing k=5 sentences on website

<table>
<thead>
<tr>
<th>Sentences model most sure are positive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Easily best sushi in Seattle.</td>
</tr>
<tr>
<td>My wife tried their ramen and it was pretty forgettable.</td>
</tr>
<tr>
<td>The sushi was amazing, and the rice is just outstanding.</td>
</tr>
<tr>
<td>All the sushi was delicious.</td>
</tr>
<tr>
<td>The service was perfect.</td>
</tr>
</tbody>
</table>

precision at k = 0.8
Summary of precision-recall

What you can do now...

- Classification accuracy/error are not always right metrics
- **Precision** captures fraction of positive predictions that are correct
- **Recall** captures fraction of positive data correctly identified by the model
- Trade-off precision & recall by setting probability thresholds
- Plot precision-recall curves.
- Compare models by computing precision at $k$