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

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

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

**Combine?**

\[ F(x_i) = \text{sign}(\hat{w}_1 f_1(x_i) + \hat{w}_2 f_2(x_i) + \hat{w}_3 f_3(x_i) + \hat{w}_4 f_4(x_i)) \]

---

**Boosting = Greedy learning ensembles from data**

1. **Training data**
2. **Learn classifier**
   \[ f_1(x) \]
3. **Predict** \( \hat{y} = \text{sign}(f_1(x)) \)
4. **Weighted data**
5. **Learn classifier & coefficient**
   \[ \hat{w}_1 f_2(x) \]
6. **Predict** \( \hat{y} = \text{sign}(\hat{w}_1 f_1(x) + \hat{w}_2 f_2(x)) \)

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

Learning from weighted data in general

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**

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

\[ \text{while } \left\| \nabla \ell(\boldsymbol{w}^{(t)}) \right\| > \epsilon \]

\[ \text{for } j = 0, \ldots, D \]

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

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

\[ t \leftarrow t + 1 \]

\[ \text{Difference between truth and prediction} \]

---

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

\[ \text{Sum over data points} \]

\[ \text{Weigh each point by } \alpha_i \]

\[ \text{Resulting } \boldsymbol{w}^* \text{ is "best" logistic classifier (with given features) on data with weights } \alpha_i \]
How to learn the “best” decision stump?

<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>

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:

- \( \text{Income} \)?
  - > $100K
    - 2
    - 1.2
    - \( \hat{y} = \text{Safe} \)
  - ≤ $100K
    - 3
    - 6.5
    - \( \hat{y} = \text{Risky} \)
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
Risky
No

Weighted error = \[
\frac{\text{total weight of mistakes}}{\text{total weight}}
\]

= \frac{1.2 + 0.6 + 0.8 + 0.7 + 0.9}{12.7}

= \frac{4.2}{12.7} = 0.33
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 >= $t^*$

Safe

Risky

Income

$10K

$120K

weights on pts
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

$\Rightarrow$ Same alg. as before, but each considered threshold has a different weighted error.
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

No

Safe

Risky

weighted_error = 0.33

Credit history?

Bad

Good

Risky

Safe

weighted_error = 0.35

Savings>$100K?

Yes

No

Safe

Risky

weighted_error = 0.4

Market conditions?

Bad

Good

Risky

Safe

weighted_error = 0.45

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

Boosted decision stumps

- Start same weight for all points: $\alpha_i = 1/N$
- For $t = 1, \ldots, T$
  - 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

Determine weight (trust) of classifier

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

\[ = \frac{1}{2} \ln \left( \frac{1 - \frac{7}{31}}{\frac{7}{31}} \right) \]

\[ = 0.62 \]
Updating weights $\alpha_i$

- Increase weight $\alpha_i$ of misclassified points

Learned decision stump $f_1(x)$

New data weights $\alpha_i$

$\alpha_i \left\{ \begin{array}{ll}
\alpha_i e^{-\tilde{w}_t} & \text{if } f_t(x_i) = y_i \\
\alpha_i e^{1.5 \tilde{w}_t} & \text{if } f_t(x_i) \neq y_i
\end{array} \right.$

Normalize weights...

$\alpha_i \leftarrow \frac{\alpha_i}{\sum_{j=1}^{N} \alpha_j}$

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

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

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

Definite $\hat{y}=1$

Uncertain $\hat{y}=\pm 1$
Decision boundary of ensemble classifier after 30 iterations

$\text{training\_error} = 0$

AdaBoost:
Revisiting convergence & overfitting
AdaBoost Theorem

Under some technical conditions...

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

May oscillate a bit

But will generally decrease, & eventually become 0!

Decision trees on loan data

39% test error

8% training error

Boosted decision stumps on loan data

32% test error

28.5% training error

Better fit & lower test error than best decision tree

Overfitting

already boosted decision stumps have better test error than best decision tree
Boosting tends to be robust to overfitting

![Graph showing performance of training and test error over iterations](image)

- Test set performance about constant for many iterations
- Less sensitive to choice of $T$

But boosting will eventually overfit, so must choose max number of components $T$

![Graph showing performance with increasing number of trees](image)

- Best test error around 31%
- Test error eventually increases to 33% (overfits)
How do we decide when to stop boosting?

Choosing $T$?

Like selecting parameters in other ML approaches (e.g., $\lambda$ in regularization)

- Not on training data
  - Not useful: training error improves as $T$ increases
- Never ever ever on test data
- Validation set
  - If dataset is large
- Cross-validation
  - For smaller data

Summary of boosting
Variants ofboosting 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

Evaluating classifiers:

Precision & Recall
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

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.

Sentiment classifier

Input $x_i$: Easily best sushi in Seattle.

Output: $\hat{y}_i$
Predicted sentiment

Easy 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, 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

- **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 $y_i$
- Predicted label $\hat{y}_i$

- **True Positive**: $y_i = +1$ and $\hat{y}_i = +1$
- **False Positive** (FP): $y_i = +1$ and $\hat{y}_i = -1$
- **False Negative** (FN): $y_i = -1$ and $\hat{y}_i = +1$
- **True Negative** (TN): $y_i = -1$ and $\hat{y}_i = -1$
Confusion matrix for sentiment analysis

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: ✔

\[
\text{precision} = \frac{\hat{y}}{c} = \frac{2}{3}
\]

4 correct
2 mistakes

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: ✔

2 negative sentences shown to potential customers...

High precision means positive predictions actually likely to be positive!
Recall:
Fraction of positive data predicted to be positive

Did I find all the positive sentences?

True positive sentences: $y_i = +1$

Sentences from all reviews for my restaurant

Classifier MODEL

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

Recall: Fraction of positive data predicted to be positive

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

Subset of positive data points correctly identified

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

All positive data points \( y = +1 \)
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!

<table>
<thead>
<tr>
<th>Predicted positive ( \hat{y}_i = +1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Easily best sushi in Seattle</td>
</tr>
<tr>
<td>The seaweed salad was just OK, vegetable salad was just ordinary.</td>
</tr>
<tr>
<td>I like the interior decoration and the blackboard menu on the wall.</td>
</tr>
<tr>
<td>The service is somewhat hectic.</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>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Predicted negative ( \hat{y}_i = -1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>The seaweed salad was just OK, vegetable salad was just ordinary.</td>
</tr>
<tr>
<td>My wife tried their ramen and it was delicious.</td>
</tr>
<tr>
<td>The service is somewhat hectic.</td>
</tr>
<tr>
<td>My wife tried their ramen and it was pretty forgettable.</td>
</tr>
<tr>
<td>The service was perfect.</td>
</tr>
</tbody>
</table>

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

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

The service is somewhat hectic.
**Pessimistic model:**

High precision, low recall

- **Predicted positive** $\hat{y}_i = +1$
  - Easily best sushi in Seattle.
  - The sushi was amazing, and the rice is just outstanding.

- **Predicted negative** $\hat{y}_i = -1$
  - I like the interior decoration and the blackboard menu on the wall.
  - The service is somewhat hectic.
  - The seaweed salad was just OK, vegetable salad was just ordinary.
  - All the sushi was delicious.
  - The service was perfect.
  - My wife tried their ramen and it was pretty forgettable.
  - The service is somewhat hectic.

**True positive sentences:** $y_i = +1$

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

---

**Balancing precision & recall**

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

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

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

- **Definite +1**: P(y=+1|x= “The sushi & everything else were awesome!”) = 0.99

- **Not sure**: P(y=+1|x= “The sushi was good, the service was OK”) = 0.55

Can be used to trade off precision and recall

---

Basic classifier

**Predict most likely class**

If \( \hat{P}(y=+1|x_i) > 0.5 \):

\[ \hat{y}_i = +1 \]

Else:

\[ \hat{y}_i = -1 \]
Pessimistic: High precision, low recall

**Pessimistic classifier**

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

Sentence from review
Input: \( x_i \)

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: \( x_i \)
Prediction probability threshold

Probability $t$ above which model predicts true

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

Example threshold values

$t = 0.99$ (pessimistic)

$t = 0.01$ (optimistic)
Tradeoff precision & recall with threshold

\[ t = 0 \quad \text{Low precision, high recall} \quad \text{Pessimistic Model} \quad \text{Predict positive only when very sure} \]
\[ t = 1 \quad \text{High precision, low recall} \quad \text{Optimistic Model} \quad \text{Predict almost everything as positive} \]

Precision-recall curve
The precision-recall curve

What does the perfect algorithm look like?
Which classifier is better? A or B?

Classifier A

Classifier B

Precision  Recall

Which classifier is better? A or C?

Classifier A

Classifier C

Precision  Recall

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>

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 $k$