A Simpler Explanation of Differential Privacy and Its Applications to Machine Learning

Nina Zumel
Win-Vector, LLC
December 2, 2015
Differential Privacy

• Secure Analysis over Sensitive Data
  • 2006: AOL Search Data “Anonymized” Release
  • Netflix Data

• Can we analyze data without leaking information?

Thelma Arnold, User #4417749
Useful Aggregations

- Mixed Success in Analysis
- Recent Results for Machine Learning
- Differential Privacy to Reuse Test Data
- Reduce Upward Bias in Model Evaluation
Outline

• Define Differential Privacy

• Give an example of Recent Results
  • Reusable Hold-out
  • Nested Models
The Differential Privacy Game

Assume \( A() \) returns a value in \([0,1]\)

Assume \( Q \) is the interval \((T, 1]\) (so adversary picks \(T\))
The Differential Privacy Game

S and S’ differ by only one row

Q: “Is A(s) > T?”

Adversary: Picks S, S’ and Q (or T)

Based on answer, Adversary guesses if Learner is working on S or S’

A(s) or A(s) > T

Learner: Implements A(s)

Over many rounds of the game (with the same S, S’):

A(S) > T with probability p
A(S’) > T with probability p’

If p >> p’ (or vice versa), adversary usually wins.

If p/p’ ~1, adversary can’t do better than random guesses.
**ε-Differential Privacy**

A() is ε-differentially Private if

\[
\left| \log \left( \frac{\text{Prob}[A(S) \in Q]}{\text{Prob}[A(S') \in Q]} \right) \right| \leq \epsilon
\]

for all choices of S, S’, Q

In English: A(S) looks a lot like A(S’)

---

**Diagram Explanation:**

- **S and S’ differ by only one row**
- **S** and **S’**
- **Q:** “Is A(s) > T?”
- **Adversary:** Picks S, S’ and Q (or T)
- **Learner:** Implements A(s)
- Based on answer, Adversary guesses if Learner is working on S or S’
Example

• $A(s)$: returns the approximate mean value of $s$

• $S$: \{0,0,\ldots,0\} (100 zeros)

• $S'$: \{1,0,\ldots,0\} (1 one, 99 zeros)

• Adversary picks $T$ so that if $A(s) > T$, $s$ is $S'$ (with high probability)
Deterministic Case: $A(s) = E(s)$

- $A(S) = 0$, $A(S') = 0.01$
- Adversary picks $T=0.005$
- Not differentially private for any $\varepsilon$. 
Add Noise

- Laplacian Noise: $L(0, \sigma)$
  - $\sigma = 1/3n$
- Now sometimes $A(S) > T$
- Need more noise
Add More Noise

• Need $\sigma > 1/n$

• $\sigma = 3/n = 0.03$

• Now often $A(S) > T$

• If $R =$ ratio of green:orange

\[
\log(\text{abs}(R)) = \epsilon
\]
Stricter $\varepsilon : A(S) \rightarrow A(S')$

- We can simulate the game I described
- 1000 rounds
- $A(S)$ and $A(S')$ get closer (in % difference)
Stricter $\epsilon$: Estimates Poorer

- $E(S) = 0; E(S') = 0.01$
- Hard to balance privacy and good analysis!
Differential Privacy Applied to Reusable Holdout Data

• Standard ML Practice: Training/Test split
  • or Training/Calibration/Test

• Ideally: Look at Test only once

• In practice: Look at Test, tweak model, look at Test…

• Upward-biased performance estimates on Training
  — and Test
How Many Times Can You Use The Test Set?

- In Theory: $\exp(N)$ times, where $N$ is size of Test
- In Practice: $N \times N$ times —non-adaptively
  - not true if you tune model after a query
- New results: $N \times N$ times **adaptively**
  - Dwork, Feldman, Hardt, Pitassi, Reingold, Roth, 2015
The Idea

• Use differential privacy to evaluate candidate models on holdout sets “without looking at data.”

• Reduce the bias from test set performance estimates: test set estimates should approximate true out-of-sample performance.
Example: Stepwise Regression

• Use the training set to train a model with k parameters, and the test set to evaluate its accuracy, and pick the best (most improved) k-parameter model.

• Greedy: kth-step uses previous best k-1 parameters

• Run until k=50
Experiment

• Simulated data

• Binary classification (50% positive class)

• 110 candidate variables
  • 10 with signal, 100 with pure noise

• 1000 rows training, 1000 rows test

• Estimate true out-of-sample performance with “fresh” set of 10,000 rows
Naive Method

• Test set more up-biased than training!

• Algorithm only picked 1 signal variable (the first)

• Neither test nor training sets estimate true model performance
Thresholdout

• Goal — Use Test to both:
  • Evaluate models
  • Estimate out-of-sample model performance

• Improvement:
  Accuracy(k) - Accuracy(k-1)

• Tolerance:
  $\sigma/2 + L(0, \sigma/2)$

• Never directly inspect Test, so leak information slower

Dwork, Feldman, Hardt, Pitassi, Reingold, Roth, 2015
Result

• Test performance tracks Fresh performance

• Found all 10 signal variables
  • But started picking noise early
  • Last signal variable: #36

• Peak accuracy ~61%
For Comparison: LARGE Test Set

• N=10,000, no DP

• Found 9 signal variables immediately

• Accuracy ~62.5% (9 vars)

• Test set only slightly upwardly biased
  • So large, we don’t contaminate it much
Takeaways

• Can think of Thresholdout as simulating a larger test set.

• DP designed to minimize excess generalization error — not find best possible model
  • The two are related, of course

• Stepwise Regression is dangerous
  • LOTS of queries
Differential Privacy applied to Nested Models

Submodel

Build Submodel

Build Full Model

Model
Example: Effects Coding

- For categorical variables with many levels.
  - K levels = K-1 indicator vars
- Re-encode the categorical variable as a few numerical variables.

<table>
<thead>
<tr>
<th>Make_Model</th>
<th>Price</th>
<th>SoldInWeek</th>
</tr>
</thead>
<tbody>
<tr>
<td>VW_Golf</td>
<td>$26,000</td>
<td>Yes</td>
</tr>
<tr>
<td>Mazda_Miata</td>
<td>$24,000</td>
<td>No</td>
</tr>
<tr>
<td>VW_Golf</td>
<td>$32,000</td>
<td>Yes</td>
</tr>
<tr>
<td>Toyota_Prius</td>
<td>$21,500</td>
<td>No</td>
</tr>
</tbody>
</table>
Bayesian Model or Model by Counts

<table>
<thead>
<tr>
<th>Make_Model</th>
<th>P(SoldIn Week)</th>
<th>Impact</th>
</tr>
</thead>
<tbody>
<tr>
<td>VW_Golf</td>
<td>0.6</td>
<td>0.2</td>
</tr>
<tr>
<td>Mazda_Miata</td>
<td>0.34</td>
<td>-0.06</td>
</tr>
<tr>
<td>Chevy_Camaro</td>
<td>0.16</td>
<td>-0.24</td>
</tr>
<tr>
<td>Toyota_Prius</td>
<td>0.72</td>
<td>0.32</td>
</tr>
<tr>
<td>Lotus_Elise</td>
<td>1.0</td>
<td>0.6</td>
</tr>
<tr>
<td>Overall</td>
<td>0.4</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Make_Model</th>
<th>N_SoldIn Week</th>
<th>N_NotSoldInWeek</th>
<th>LogDiff</th>
<th>IsRare</th>
</tr>
</thead>
<tbody>
<tr>
<td>VW_Golf</td>
<td>60</td>
<td>40</td>
<td>0.41</td>
<td>No</td>
</tr>
<tr>
<td>Mazda_Miata</td>
<td>68</td>
<td>132</td>
<td>-0.66</td>
<td>No</td>
</tr>
<tr>
<td>Chevy_Camaro</td>
<td>8</td>
<td>42</td>
<td>-1.6</td>
<td>No</td>
</tr>
<tr>
<td>Toyota_Prius</td>
<td>108</td>
<td>42</td>
<td>0.94</td>
<td>No</td>
</tr>
<tr>
<td>Lotus_Elise</td>
<td>1</td>
<td>0</td>
<td>1E+06</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Bayesian Model by Counts
Can’t use Training Data to Effects Code!

- Effects model can memorize the training data
  - “Lotus Elise always sells in a week”
- Full model may overestimate the value of effects-coded variable
  - Overfit
Training the Effects Model

Best Solution: A separate calibration set for effects model
Alternative Solution: Prune Rare Levels

Better: use significance of conditional estimate

<table>
<thead>
<tr>
<th>Make_Model</th>
<th>P(Sold/Wk)</th>
<th>Impact</th>
<th>Nobsv</th>
</tr>
</thead>
<tbody>
<tr>
<td>VW_Golf</td>
<td>0.6</td>
<td>0.2</td>
<td>100</td>
</tr>
<tr>
<td>Mazda_Miata</td>
<td>0.34</td>
<td>-0.06</td>
<td>200</td>
</tr>
<tr>
<td>Chevy_Camaro</td>
<td>0.16</td>
<td>-0.24</td>
<td>50</td>
</tr>
<tr>
<td>Toyota_Prius</td>
<td>0.72</td>
<td>0.32</td>
<td>150</td>
</tr>
<tr>
<td><strong>Lotus_Elise</strong></td>
<td><strong>1.0</strong></td>
<td><strong>0.6</strong></td>
<td><strong>1</strong></td>
</tr>
<tr>
<td><strong>Yugo_GV</strong></td>
<td>0.33</td>
<td>-0.07</td>
<td>3</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Overall</td>
<td>0.4</td>
<td>0</td>
<td>N</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Make_Model</th>
<th>Impact</th>
</tr>
</thead>
<tbody>
<tr>
<td>VW_Golf</td>
<td>0.2</td>
</tr>
<tr>
<td>Mazda_Miata</td>
<td>-0.06</td>
</tr>
<tr>
<td>Chevy_Camaro</td>
<td>-0.24</td>
</tr>
<tr>
<td>Toyota_Prius</td>
<td>0.32</td>
</tr>
<tr>
<td><strong>Lotus_Elise</strong></td>
<td><strong>0</strong></td>
</tr>
<tr>
<td><strong>Yugo_GV</strong></td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Innovative Solution: Differential Privacy

Add noise to training data before passing to effects coding
Example

- Synthetic data, 2000 rows training
- 40 categorical variables
  - 10 signal variables with 10 levels each
  - 30 noise variables with 500 levels each
- Classification: Positive class 50% prevalence
- Effects code the variables, then fit a logistic regression model
Naive Modeling

In Training: both models perfect (AUC = 1)
With Laplace Noise

In Training: $AUC = 0.95$

In Training: $AUC = 0.96$
With Calibration Set

Bayesian model:
In Training: AUC = 0.91
All training data and rare level pruning

Bayesian model:
In Training: AUC = 0.95
Takeaways

• Differential privacy alleviates the overfit from effects coding (or nested models in general) by masking rare phenomena.

• DP is a useful alternative when there’s not enough data for a calibration set.
  • Or for online situations (with learning by counts)
  • For batch, rare level pruning also works well
References

  • http://arxiv.org/abs/1411.2664

  • Abstract: https://www.sciencemag.org/content/349/6248/636.abstract


References

• Blog posts (Differential privacy mini-series):

• Our code, data and examples:
  • https://github.com/WinVector/Examples/tree/master/DiffPriv/PrivStep
  • https://github.com/WinVector/PreparingDataWorkshop/tree/master/NestedModels
Thank You