Machine Learning Systems Design

Modeling Pipeline

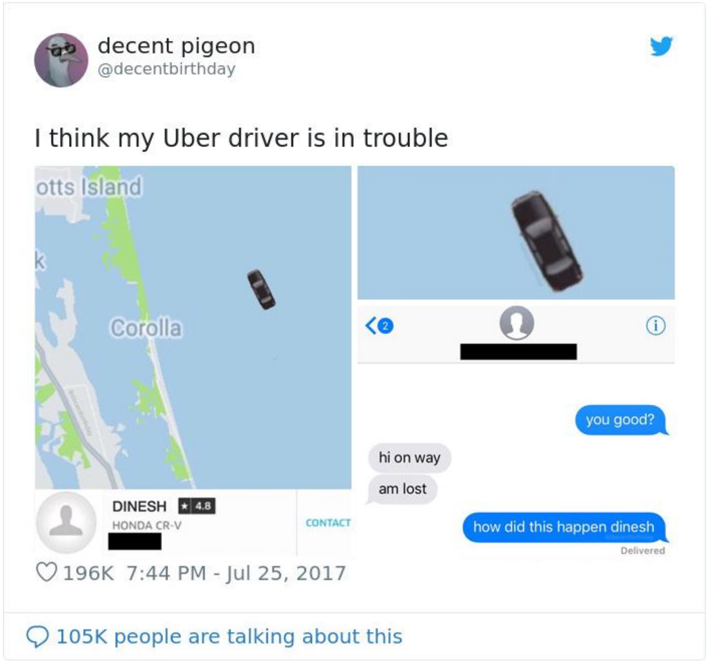
Lecture 12: Model Performance Analysis

CE 40959 Spring 2023
Ali Zarezade
SharifMLSD.github.io

Agenda

1. Model Evaluation
2. Offline Model Evaluation
3. Test Set Adequacy
4. Statistical Bounds

1. Model Evaluation



decent pigeon
@decentbirthday

I think my Uber driver is in trouble

[redacted]

[redacted]

you good?

hi on way

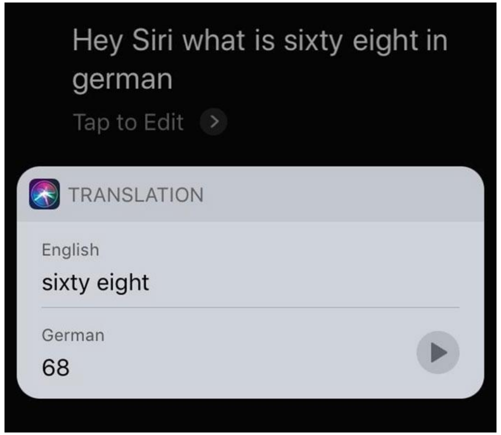
am lost

how did this happen dinesh

Delivered

196K 7:44 PM - Jul 25, 2017

105K people are talking about this



Hey Siri what is sixty eight in german

Tap to Edit

TRANSLATION

English	sixty eight
German	68

Facebook translates 'good morning' into 'attack them', leading to arrest

Palestinian man questioned by Israeli police after embarrassing mistranslation of caption under photo of him leaning against bulldozer

0 People Like You

Model evaluation

- Offline evaluation: before deployed
- Online evaluation: after deployed

Test in production. Will cover this later!

Model offline evaluation

- Baselines
- Evaluation methods

Baselines

- Numbers by themselves mean little
- Task: binary classification, 90% POSITIVE, 10% NEGATIVE
- F1 score: 0.90

Is it model good or bad?

Model selection: baselines

- **Random baseline**
  - Predict at random:
    - uniform
    - following label distribution

Model selection: baselines

- **Random baseline**
  - Predict at random:
    - uniform
    - following label distribution

- **Example**: misinformation classification
  - n = 1,000,000
  - 99% negative (label = 0)
  - 1% positive (label = 1)

	Accuracy	**F1**
Random [uniform]	0.5	?
Random [label distribution]	0.98	?

Model selection: baselines

- **Random baseline**
  - Predict at random:
    - uniform
    - following label distribution

- **Example**: misinformation classification
  - n = 1,000,000
  - 99% negative (label = 0)
  - 1% positive (label = 1)

	Accuracy	**F1**
Random [uniform]	0.5	0.02
Random [label distribution]	0.98	0.01

Model selection: baselines

- **Random baseline**
  - Predict at random:
    - uniform
    - following label distribution
- **Zero rule baseline**
  - Always predict the most common class

- **Example**: misinformation classification
  - n = 1,000,000
  - 99% negative (label = 0)
  - 1% positive (label = 1)

	Accuracy	**F1**
Random [uniform]	0.5	0.02
Random [label distribution]	0.98	0.01
Most common [preds = [0] * n]	?	?

Model selection: baselines

- **Random baseline**
  - Predict at random:
    - uniform
    - following label distribution
- **Zero rule baseline**
  - Always predict the most common class
- **Simple heuristics**
  - E.g.: classify tweets based on whether they contain links to unreliable sources

- **Example**: misinformation classification
  - n = 1,000,000
  - 99% negative (label = 0)
  - 1% positive (label = 1)

	Accuracy	**F1**
Random [uniform]	0.5	0.02
Random [label distribution]	0.98	0.01
Most common [preds = [0] * n]	?	?
Simple heuristics	?	?

Model selection: baselines

- **Random baseline**
  - Predict at random:
    - uniform
    - following label distribution
- **Zero rule baseline**
  - Always predict the most common class
- **Simple heuristics**
  - E.g.: classify tweets based on whether they contain links to unreliable sources
- **Human baseline**
  - What's human-level performance?

- **Example**: misinformation classification
  - n = 1,000,000
  - 99% negative (label = 0)
  - 1% positive (label = 1)

	Accuracy	**F1**
Random [uniform]	0.5	0.02
Random [label distribution]	0.98	0.01
Most common [preds = [0] * n]	?	?
Simple heuristics	?	?
Human expert	?	?

Model selection: baselines

- **Random baseline**
  - Predict at random:
    - uniform
    - following label distribution
- **Zero rule baseline**
  - Always predict the most common class
- **Simple heuristics**
  - E.g.: classify tweets based on whether they contain links to unreliable sources
- **Human baseline**
  - What's human-level performance?
- **Existing solutions**

- **Example**: misinformation classification
  - n = 1,000,000
  - 99% negative (label = 0)
  - 1% positive (label = 1)

	Accuracy	**F1**
Random [uniform]	0.5	0.02
Random [label distribution]	0.98	0.01
Most common [preds = [0] * n]	?	?
Simple heuristics	?	?
Human expert	?	?
3rd party API	?	?

2. Offline Model Evaluation

Evaluation methods

1. Perturbation Tests
2. Invariance Tests
3. Directional Expectation Tests
4. Model Calibration
5. Confidence Measurement
6. Slice-based Evaluation

Perturbation tests

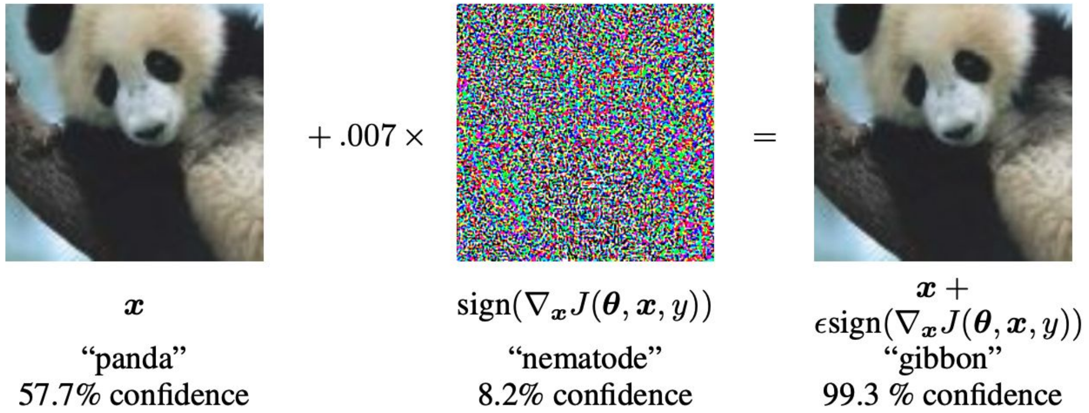
- Problem: users input might contain noise, making it different from test data
  - Examples:
    - Speech recognition: background noise
    - Object detection: different lighting
    - Text inputs: typos, intentional misspelling (e.g. looooooooong)
  - Model does well on test set, but fails in production

Perturbation tests

- Motivation: users input might contain noise, making it different from test data
- Idea: randomly add small noise to test data to see how much outputs change

Perturbation tests

- Motivation: users input might contain noise, making it different from test data
- Idea: randomly add small noise to test data to see how much outputs change
- The more sensitive the model is to noise:
  - The harder it is to maintain
  - The more vulnerable the model is to adversarial attacks


$+ .007 \times$
$=$
$\boldsymbol{x}$
"panda"
57.7% confidence
$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"nematode"
8.2% confidence
$\boldsymbol{x} + \epsilon\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"gibbon"
99.3 % confidence

Perturbation tests

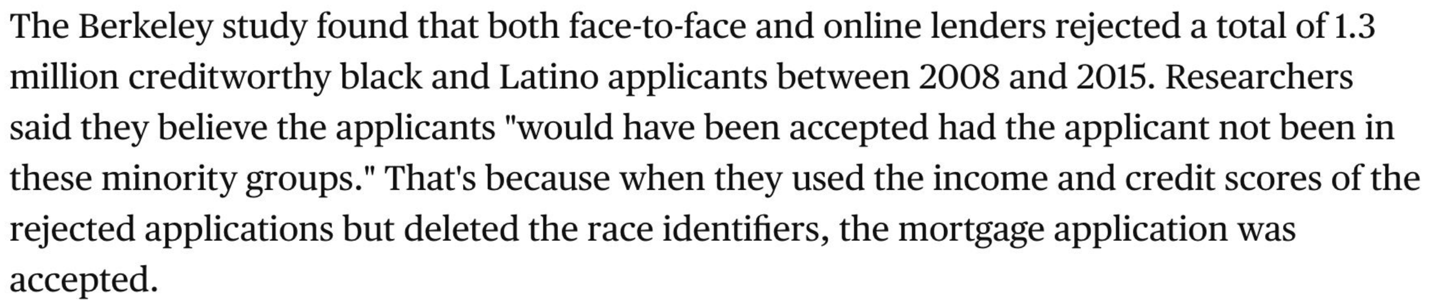
- Motivation: users input might contain noise, making it different from test data
- Idea: randomly add small noise to test data to see how much outputs change

If small changes cause model's performance to fluctuate,
you might want to make model more robust:

- Add noise to training data
- Add more training data
- Choose another model

Invariance tests

- Motivation: some input changes shouldn't lead to changes in outputs
  - Changing race/gender info shouldn't change predicted approval outcome
  - Changing name shouldn't affect resume screening results



The Berkeley study found that both face-to-face and online lenders rejected a total of 1.3 million creditworthy black and Latino applicants between 2008 and 2015. Researchers said they believe the applicants "would have been accepted had the applicant not been in these minority groups." That's because when they used the income and credit scores of the rejected applications but deleted the race identifiers, the mortgage application was accepted.

Disparity in home lending costs minorities millions, researchers find (CBS News, 2019)

Invariance tests

- Motivation: some input changes shouldn't lead to changes in outputs
- Idea: keep certain features the same, but randomly change values of sensitive features

If changing sensitive features can change model's outputs, there might be biases!

Directional expectation tests

- Motivation: some changes to inputs should cause predictable changes in outputs
  - E.g. when predicting housing prices:
    - Increasing bedroom size shouldn’t decrease the predicted price
    - Decreasing square footage shouldn’t increase the predicted price

Directional expectation tests

- Motivation: some changes to inputs should cause predictable changes in outputs
- Idea: keep most features the same, but change certain features to see if outputs change predictably

If increasing lot size consistently reduces the predicted price, you might want to investigate why!

Model calibration

“One of the most important tests of a forecast — I would argue that it is the single most important one — is called calibration.”

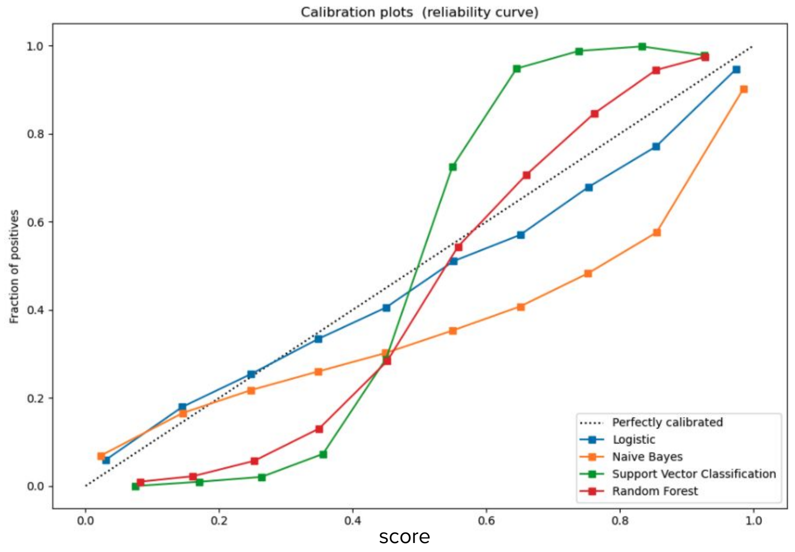
Nate Silver, **The Signal and the Noise**

Model calibration

- If you predict team A wins in A vs. B match with 60% probability:
  - In 100 A vs. B match, A should win 60% of the time!

Model calibration: binary case

For instance, a well calibrated (binary) classifier should classify the samples such that among the samples to which it gave a predict_proba value close to 0.8, approximately 80% actually belong to the positive class



Need to ensure the **top class is correct on average**

Image from Probability calibration (sklearn)

Model calibration: recsys

- Recommend movies to a user who watches 70% comedy, 30% action
- What happens if you recommend most likely watched movies?

Movie title	**Watch probability**
Comedy 1	0.8
Comedy 2	0.73
Comedy 3	0.68
Comedy 4	0.67
Action 1	0.29
Action 2	0.2
Science fiction	0.04

Model calibration: recsys

- Recommend movies to a user who watches 70% comedy, 30% action
- What happens if you recommend most likely watched movies?

Need to calibrate recommendations to include 70% comedy, 30% action

Movie title	**Watch probability**
Comedy 1	0.8
Comedy 2	0.73
Comedy 3	0.68
Comedy 4	0.67
Action 1	0.29
Action 2	0.2
Science fiction	0.04

Model calibration: CTR

- 2 ads: A & B
- Model predicts click probability: A (10%), B (8%)
- How to estimate number of clicks you’ll actually get if model isn’t calibrated?

Confidence measurement

- Usefulness threshold for each individual prediction
- Uncertain predictions can cause annoyance & catastrophic consequences

Confidence measurement

- How to measure the confidence level of each prediction?
- What to do with predictions below the confidence threshold?
    - Skip
    - Ask for more information
    - Loop in humans

Slice-based evaluation

Different performance on different slices

- Classes
  - Might perform worse on minority classes
- Subgroups
  - Gender
  - Location
  - Time of using the app
  - etc.

Same performance on different slices with different cost

- User churn prediction
  - Paying users are more critical
- Predicting adverse drug reactions
  - Patients with underlying conditions are more critical

⚠️ Focusing on improving only overall metrics might hurt performance on subgroups ⚠️

Slice-based evaluation: example

- Majority group: 90%
- Minority group: 10%

Zoom poll: Which model would you go with?

	Majority accuracy	**Minority accuracy**
Model A	98%	80%
Model B	95%	95%

Slice-based evaluation: example

- Majority group: 90%
- Minority group: 10%

Coarse-grained evaluation can hide:
- model biases
- potential for improvement

	Majority accuracy	**Minority accuracy**	**Overall accuracy**
Model A	98%	80%	96.2%
Model B	95%	95%	95%

Simpson’s paradox

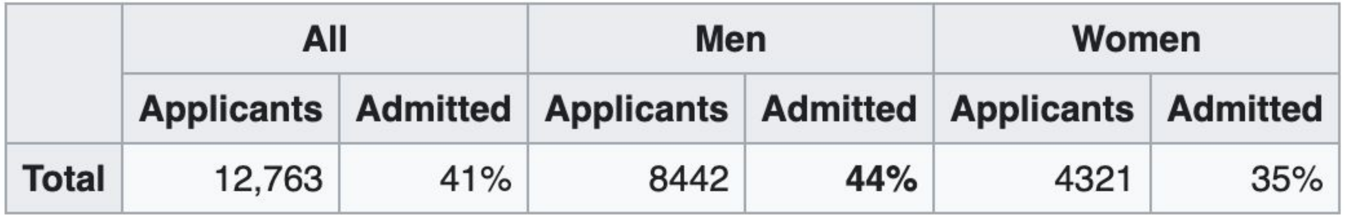
- Models A and B to predict whether a customer will buy your product
- A performs better than B overall
- B performs better than A on both female & male customers

Simpson's paradox

	Treatment 1	Treatment 2
Group A	**93% (81/87)**	87% (234/270)
Group B	**73% (192/263)**	69% (55/80)
Overall	78% (273/350)	**83% (289/350)**

Numbers from a kidney stone treatment study. (Charig et al., 1986)

Simpson’s paradox: Berkeley graduate admission ‘73

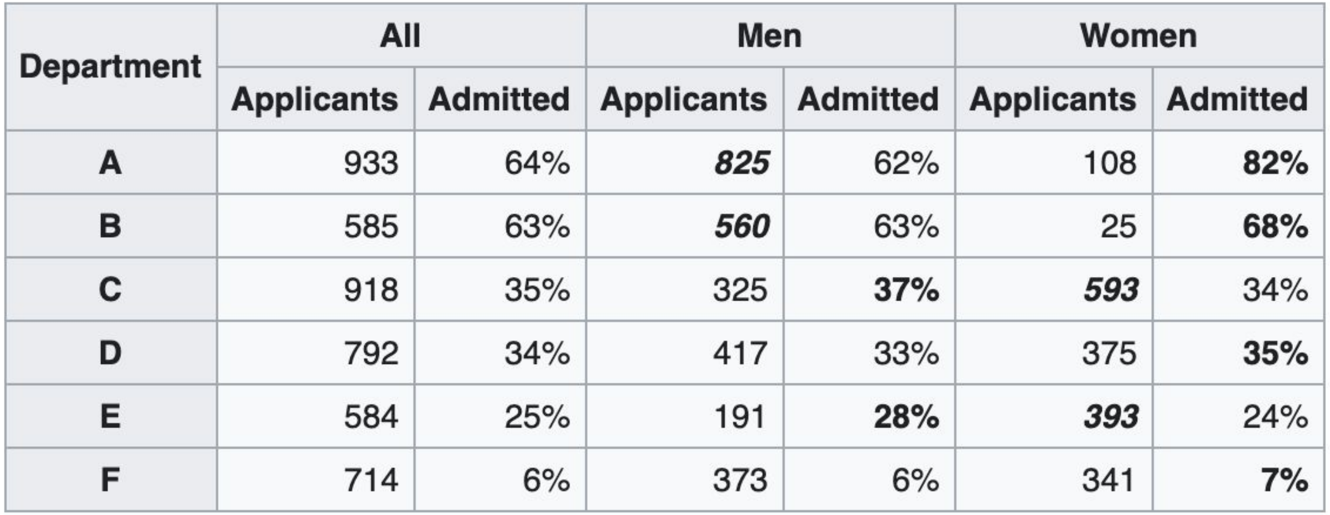


	All		Men		Women	
	Applicants	Admitted	Applicants	Admitted	Applicants	Admitted
Total	12,763	41%	8442	**44%**	4321	35%

Bias against women in the process, or is there…?

[Sex Bias in Graduate Admissions: Data from Berkeley]() (Bickel et al., 1975)

Simpson’s paradox: Berkeley graduate admission ‘73



Department	All		Men		Women	
	Applicants	Admitted	Applicants	Admitted	Applicants	Admitted
A	933	64%	***825***	62%	108	**82%**
B	585	63%	***560***	63%	25	**68%**
C	918	35%	325	**37%**	***593***	34%
D	792	34%	417	33%	375	**35%**
E	584	25%	191	**28%**	***393***	24%
F	714	6%	373	6%	341	**7%**

⚠️ Aggregation can conceal and contradict actual situation ⚠️

Sex Bias in Graduate Admissions: Data from Berkeley (Bickel et al., 1975)

Slice-based evaluation

- Evaluate your model on different slices
  - E.g. when working with website traffic data, slice data among:
    - gender
    - mobile vs. desktop
    - browser
    - location
- Check for consistency over time
  - E.g. evaluate your model on data slices from each day

Slice-based evaluation

- Improve model’s performance both overall and on critical data
- Help avoid biases
- Even when you don’t think slices matter, slicing can:
    - give you confidence on your model (to convince your boss)
    - might reveal non-ML problems

How to identify slices?

- Heuristics
  - Might require subject matter expertise
- Error analysis
  - Patterns among misclassified samples
- Slice finder
  - Exhaustive/beam search
  - Clustering
  - Decision tree

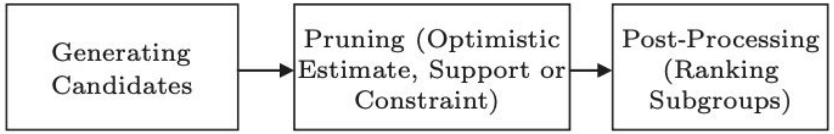

Generating Candidates
Pruning (Optimistic Estimate, Support or Constraint)
Post-Processing (Ranking Subgroups)

Fig.1. Methodology for subgroup discovery.

How to identify slices?

Will go into details next TA session!

- Heuristics
  - Might require subject matter expertise
- Error analysis
  - Patterns among misclassified samples
- Slice finder
  - Exhaustive/beam search
  - Clustering
  - Decision tree

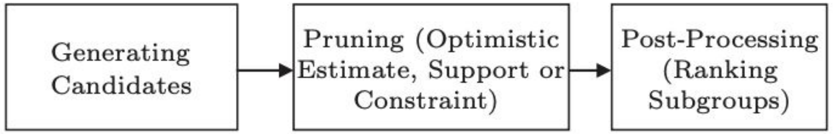

Generating Candidates
Pruning (Optimistic Estimate, Support or Constraint)
Post-Processing (Ranking Subgroups)

Fig.1. Methodology for subgroup discovery.

4. Test Set Adequacy

Evaluation of test set adequacy

- Software engineering uses tests to find bugs in the code before it goes to production
- The same applies to the code that interacts with the statistical model (input, feature eng., output, serving)
- But the model itself also needs to be evaluated with test examples
- The test examples should reveal the model's flaws before it goes to production

Neuron Coverage

- A measure of how well a test set activates the units (neurons) of a neural network model
- A good test set should activate all or most of the units
- A method to create such a test set is to:
    - Label random examples and feed them to the model
    - Check which units are activated by the examples (above a threshold)
    - Mark the activated units as covered if the prediction is correct
    - Repeat until all or most of the units are covered

Mutation test

- We create different versions of the model by changing the training data or the model structure
- We call these versions mutants
- We apply the test set to each mutant and see if it makes a wrong prediction or not
- We call this killing a mutant
- A good test set should kill all or most of the mutants

3. Statistical Bound

Statistical interval for classification error

- Suppose we have a classification model and we want to estimate its error ratio (err), which is the fraction of incorrect predictions on a test set of size N.
- We can use a statistical technique to obtain an interval that contains err with high probability (confidence level).
- The interval is given by $[err - \delta, err + \delta]$ where $\delta = \frac{z_N}{\sqrt{N}}\sqrt{err(1-err)}$ and z_N is a constant that depends on the confidence level.
- The table below shows some values of z_N for different confidence levels

confidence level	80%	90%	95%	98%	99%
z_N	1.28	1.64	1.96	2.33	2.58

Statistical interval for classification error

- For example, if we have a test set of size 100 and we observe an error ratio of 0.2, then with 99% confidence, the true error ratio lies in the interval $[0.2 - \frac{2.58}{10}\sqrt{0.2(1-0.2)}, 0.2 + \frac{2.58}{10}\sqrt{0.2(1-0.2)}]$ which is approximately $[0.12, 0.28]$.

Bootstrapping statistical interval

- A general technique for estimating the uncertainty of any metric for classification or regression
- Based on creating B random samples of the test set by sampling with replacement
- For each sample Sb, compute the metric mb using the model
- Sort the B values of mb in ascending order
- To obtain a c% confidence interval, find the smallest interval [a,b] that covers at least c% of the sum of all mb values

Bootstrapping statistical interval example

Suppose we have B = 10 bootstrap samples of the test set and we compute the metric mb for each sample as [9.8,7.5,7.9,10.1,9.7,8.4,7.1,9.9,7.7,8.5] and we want a confidence level of c = 80%.

- We sort them in ascending order: [7.1, 7.5, 7.7, 7.9, 8.4, 8.5, 9.7, 9.8, 9.9, 10.1]
- We find the smallest interval [a,b] that covers at least 80% of the sum of all mb values
- The sum of all mb values is S = 86.6
- The smallest interval that covers at least 80% of S is [7.46, 9.92], which covers 69.4 out of 86.6
- Therefore, our confidence interval is [7.46, 9.92]

Bootstrapping prediction interval for regression

Given a model f and an input x, find an interval [$f_{min}(x)$,$f_{max}(x)$] where f(x) lies with confidence c%:

- Choose a confidence level c (usually 95% or 99%)
- Choose a number of bootstrap samples B (usually 100)
- For each bootstrap sample, train a model and get a prediction for x
- Sort the predictions and find the smallest interval [a,b] that covers at least c% of them
- Return f(x) and the interval [a,b] as the prediction interval

Bootstrapping prediction interval for regression

- Suppose we have a model f that predicts house prices based on features x. We want to find the prediction interval for a house with x = [3 bedrooms, 2 bathrooms, 1500 sqft].
- We use bootstrapping with c = 95% and B = 100.
- We get 100 predictions for x, ranging from $200k to $300k.
- The smallest interval that covers 95% of them is [a,b] = [$210k, $290k].
- We return f(x) = $250k and the interval [$210k, $290k] as the prediction interval.

Machine Learning Systems Design

Modeling Pipeline

Next Lecture: Hyperparameter tuning and AutoML

CE 40959 Spring 2023
Ali Zarezade
SharifMLSD.github.io