Data-Intensive Distributed Computing
CS 431/631 451/651 (Winter 2021)

Part 7: Data Mining (2/4)

Ali Abedi

These slides are available at https://www.student.cs.uwaterloo.ca/~cs451

This work is licensed under a Creative Commons Attribution-Noncommercial-Share Alike 3.0 United States
See http://creativecommons.org/licenses/by-nc-sa/3.0/us/ for details
Stochastic Gradient Descent
**Stochastic Gradient Descent**

**Gradient Descent**

\[
\theta^{(t+1)} \leftarrow \theta^{(t)} - \gamma^{(t)} \frac{1}{n} \sum_{i=0}^{n} \nabla \ell(f(x_i; \theta^{(t)}), y_i)
\]

**Stochastic Gradient Descent (SGD)**

\[
\theta^{(t+1)} \leftarrow \theta^{(t)} - \gamma^{(t)} \nabla \ell(f(x; \theta^{(t)}), y)
\]
**Stochastic Gradient Descent**

**Gradient Descent**

\[ \theta^{(t+1)} \leftarrow \theta^{(t)} - \gamma^{(t)} \frac{1}{n} \sum_{i=0}^{n} \nabla \ell(f(x_i; \theta^{(t)}), y_i) \]

Considers all training instances in every iteration

**Stochastic Gradient Descent (SGD)**

\[ \theta^{(t+1)} \leftarrow \theta^{(t)} - \gamma^{(t)} \nabla \ell(f(x; \theta^{(t)}), y) \]

Considers a random instance in every iteration
**Stochastic Gradient Descent**

Gradient Descent

\[ \theta^{(t+1)} \leftarrow \theta^{(t)} - \gamma^{(t)} \frac{1}{n} \sum_{i=0}^{n} \nabla \ell(f(x_i; \theta^{(t)}), y_i) \]

Considers all training instances in every iteration

Stochastic Gradient Descent (SGD)

\[ \theta^{(t+1)} \leftarrow \theta^{(t)} - \gamma^{(t)} \nabla \ell(f(x; \theta^{(t)}), y) \]

Considers a random instance in every iteration
Batch Gradient Descent → Stochastic Gradient Descent

Mini-batching

Considers a random subset of instances in every iteration
Ensembles

Source: Wikipedia (Orchestra)
Ensemble Learning

Learn multiple models, combine results from different models to make prediction

- Independent

Common implementation:
Train classifiers on different input partitions of the data
Embarrassingly parallel!

Combining predictions:
- Majority voting
- Model averaging
Ensemble Learning

*independent*

Learn multiple models, combine results from different models to make prediction

Why does it work?

If errors uncorrelated, multiple classifiers being wrong is less likely
Reduces the variance component of error
MapReduce Implementation
Gradient Descent

\[ \theta^{(t+1)} \leftarrow \theta^{(t)} - \gamma^{(t)} \frac{1}{n} \sum_{i=0}^{n} \nabla \ell(f(x_i; \theta^{(t)}), y_i) \]

iterate until convergence

update model
**Stochastic Gradient Descent**

\[
\theta^{(t+1)} \leftarrow \theta^{(t)} - \gamma^{(t)} \nabla \ell(f(x; \theta^{(t)}), y)
\]

This is great because we no longer need iterations! Mappers go through the record and apply the stochastic gradient descend rule on that record and update the model. This process continues for all records.
**Stochastic Gradient Descent**

\[
\theta^{(t+1)} \leftarrow \theta^{(t)} - \gamma^{(t)} \nabla \ell(f(x; \theta^{(t)}), y)
\]

No iteration!
MapReduce Implementation

\[ \theta^{(t+1)} \leftarrow \theta^{(t)} - \gamma^{(t)} \nabla \ell(f(x; \theta^{(t)}), y) \]

How do we output the model?

Option 1: write model out as "side data"
Option 2: emit model as intermediate output
What about Spark?

\[
\theta^{(t+1)} \leftarrow \theta^{(t)} - \gamma^{(t)} \nabla \ell(f(x; \theta^{(t)}), y)
\]
Data scientists usually use provided transformations in Spark ML

val model = LinearRegressionWithSGD.train(parsedData, numIterations, stepSize)
val prediction = model.predict(point.features)
Sentiment Analysis Case Study

Binary polarity classification: \{positive, negative\} sentiment

Use the “emoticon trick” to gather data 😞 😞 😊 😊

Data

Test: 500k positive/500k negative tweets from 9/1/2011
Training: \{1m, 10m, 100m\} instances from before (50/50 split)

Features:

Sliding window byte-4grams

Models + Optimization:

Logistic regression with SGD (L2 regularization)
Ensembles of various sizes (simple weighted voting)
Ensembles with 10m examples better than 100m single classifier!

Diminishing returns...
Supervised Machine Learning

Machine Learning Algorithm

training data

training

testing/deployment

Model

?
Evaluation
How do we know how well we’re doing?

Why isn’t this enough?
Induce: \( f : X \to Y \)
Such that loss is minimized
\[
\arg\min_{\theta} \frac{1}{n} \sum_{i=0}^{n} \ell(f(x_i; \theta), y_i)
\]

We need end-to-end metrics!
Obvious metric: accuracy
Why isn’t this enough?
Metrics

<table>
<thead>
<tr>
<th>Actual</th>
<th>Predicted</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive</td>
<td>Positive</td>
</tr>
<tr>
<td>True Positive (TP)</td>
<td>False Positive (FP)</td>
</tr>
<tr>
<td>= Type I Error</td>
<td>= Type II Error</td>
</tr>
<tr>
<td>Negative</td>
<td>Negative</td>
</tr>
<tr>
<td>False Negative (FN)</td>
<td>True Negative (TN)</td>
</tr>
<tr>
<td>= Type II Error</td>
<td></td>
</tr>
</tbody>
</table>

Precision = TP/(TP + FP)

Miss rate = FN/(FN + TN)

Recall or TPR = TP/(TP + FN)

Fall-Out or FPR = FP/(FP + TN)
Type I error  
(false positive)  
You're pregnant

Type II error  
(false negative)  
You're not pregnant
A receiver operating characteristic curve, or ROC curve, is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied.
Training/Testing Splits

\[
\arg\min_{\theta} \frac{1}{n} \sum_{i=0}^{n} \ell(f(x_i; \theta), y_i)
\]

What happens if you need more? Cross-Validation

Precision, Recall, etc.
Training/Testing Splits
Training/Testing Splits

Cross-Validation
Cross-Validation
Training/Testing Splits

Cross-Validation
Training/Testing Splits

Cross-Validation
Typical Industry Setup

A/B test

Why not cross-validation?
A/B Testing

Gather metrics, compare alternatives
A/B Testing: Complexities

Properly bucketing users
   Novelty
   Learning effects
Long vs. short term effects
Multiple, interacting tests
   Nosy tech journalists
   ...

33
Supervised Machine Learning

Machine Learning Algorithm

training data

training

Model

testing/deployment

?
Applied ML in Academia

Download interesting dataset (comes with the problem)

Run baseline model
  Train/Test

Build better model
  Train/Test

Does new model beat baseline?
  Yes: publish a paper!
  No: try again!
THE SCIENTIFIC METHOD

Observe natural phenomena → Formulate Hypothesis → Test hypothesis via rigorous Experiment → Establish Theory based on repeated validation of results

THE ACTUAL METHOD

Make up Theory based on what Funding Agency Manager wants to be true → Design minimum experiments that will prove achieve hypothesis Theory is true

Modify Hypothesis

Modify Theory to fit data

Publish Paper: rename Theory a "Hypothesis" and pretend you used the Scientific Method

Defend Theory despite all evidence to the contrary
“You can’t keep adjusting the data to prove that you would be the best Valentine’s date for Scarlett Johansson.”
Data Scientist: The Sexiest Job of the 21st Century

by Thomas H. Davenport and D.J. Patil

FROM THE OCTOBER 2012 ISSUE
<table>
<thead>
<tr>
<th>Fantasy</th>
<th>Reality</th>
</tr>
</thead>
<tbody>
<tr>
<td>🎯 Extract features</td>
<td>😞 What’s the task?</td>
</tr>
<tr>
<td>🎯 Develop cool ML</td>
<td>😞 Where’s the data?</td>
</tr>
<tr>
<td>technique</td>
<td>😞 What’s in this dataset?</td>
</tr>
<tr>
<td>🎯 #Profit</td>
<td>😞 What’s all the f#$!* crap?</td>
</tr>
<tr>
<td></td>
<td>😞 Clean the data</td>
</tr>
<tr>
<td></td>
<td>😞 Extract features</td>
</tr>
<tr>
<td></td>
<td>😞 “Do” machine learning</td>
</tr>
<tr>
<td></td>
<td>😞 Fail, iterate...</td>
</tr>
</tbody>
</table>

Dirty secret: very little of data science is about machine learning per se!
It’s impossible to overstress this: 80% of the work in any data project is in cleaning the data. – DJ Patil “Data Jujitsu”
For 'Big Data' Scientists, Hurdle to Insights Is ‘Janitor Work’
On finding things...

P. Oscar Boykin
@oscar

OH: "... so to recap, tweets are statuses, favorites are favourings, retweets are shares."

Reply Retweet Favorite More
On naming things...

CamelCase

smallCamelCase

snake_case

camel_Snake

dunder__snake

Yesterday I had a run in with the camel_Snake in our code. Today, I came across the feared dunder___snake. Yow! /via @THISWILLWORK
On feature extraction...

An actual Java regular expression used to parse log message at Twitter circa 2010

Friction is cumulative!
Data Plumbing... Gone Wrong!

[scene: consumer internet company in the Bay Area...]

Frontend Engineer
- Develops new feature, adds logging code to capture clicks

Data Scientist
- Analyze user behavior, extract insights to improve feature

Okay, let's get going... where's the click data?

Well, that's kinda non-intuitive, but okay...

Oh, BTW, where's the timestamp of the click?

[grumble, grumble, grumble]

It's over here...

Well, it wouldn't fit, so we had to shoehorn...

Hang on, I don't remember...

Uh, bad news. Looks like we forgot to log it...
Fantasy

- Extract features
- Develop cool ML technique
- #Profit

Reality

- What's the task?
- Where's the data?
- What's in this dataset?
- What's all the f#$!* crap?
- Clean the data
- Extract features
- "Do" machine learning
- Fail, iterate...

Finally works!
Congratulations, you’re halfway there...
Congratulations, you’re halfway there...

Does it actually work?
A/B testing

Is it fast enough?

Good, you’re two thirds there...
Productionize

What are your jobs’ dependencies?
How/when are your jobs scheduled?
   Are there enough resources?
How do you know if it’s working?
Who do you call if it stops working?

Infrastructure is critical here!
(plumbing)
Takeaway lesson:
Most of data science isn’t glamorous!

Source: Wikipedia (Plumbing)