presentation starting soon… sit down

Introduction to Online Machine Learning Algorithms

Flink Forward- San Francisco

Trevor Grant
@rawkintrevo
April 11th, 2017

Table of Contents

- Who is this guy?
- Why should I care?
- What's going on here?
- This seems boring and mathy, maybe I should leave…

Branding

- Trevor Grant
- Things I do:
  - Open Source Technical Evangelist, IBM
  - PMC Apache Mahout
  - Blog: http://rawkintrevo.org
- Schooling
  - MS Applied Math, Illinois State
  - MBA, Illinois State
- How to get ahold of me:
  - @rawkintrevo
  - trevor.grant@ibm.com / rawkintrevo@apache.org
  - Mahout Dev and User Mailing Lists

Why does any of this matter?

- To disambiguate terms related to machine learning / streaming machine learning.

- Hopefully after this you
  - Won’t keep using words wrong
  - Will know when someone else is
    - be pretentious
    - or don’t

- Bonus material:
  - We build a fairly cool, yet super simple online recommender
  - Apache Flink + Apache Spark + Apache Mahout

Math. Eewww.

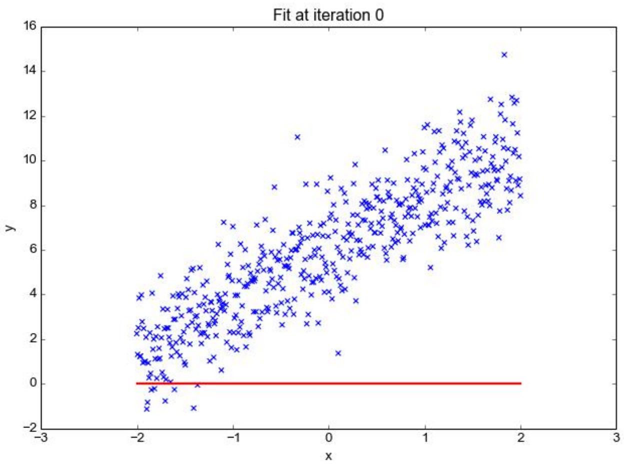
This talk invokes the following types of maths

- Weighted Averaging
- Matrix Times Vector

Also there’s pictures.

Types of Pictures

- Useful animations



http://eli.thegreenplace.net/images/2016/regressionfit.gif

- Unrelated animal pictures

Macs are good for keeping cat butts warm…
and not much else.

Table of Contents

- On the virtues of not throwing around buzzwords…
- Online vs. Offline
- Lambda vs. Kappa (w.r.t. machine learning)
- Statistical vs Adversarial
- Real-Time (one buzzword to rule them all)

Online vs. Offline

Online

- Input processed piece by piece in a serial fashion
- Each new piece of information generates an event
  - Not mini-batching
  - Possibly on a sliding window of record 1
- Not necessarily low latency

Offline

- Input processed in batches
- Not necessarily high latency

Fast offline, slow online and stack order

Slow Online

Stock broker in Des Moines Iowa writes Python program that get's EOD prices/statistics as they are published and then executes orders.

Fast Offline

HFT algorithm, executes trades based on tumbling windows of 15 milliseconds worth of activity

Online doesn't mean fast, online doesn't mean streaming, online ***only means that it processes information as soon as it is received***.

Consider an online algorithm (the slow online example), exists behind an offline EOD batch job.

- This is an extreme case, but no algorithm receives data as it is created.
- Best case- limited by speed of light (?)

Lambda vs. Kappa (Machine Learning)

Lambda

Leaning happens (i.e. models are fitted) offline

Model used by streaming engine to make decisions online

Kappa

Learning happens (i.e. models are fitted) online

Online decision model updates for each new record seen

Model can change structure e.g. new words in TF-IDF or new categories in ‘factor model linear regression’

Lambda with Novell Information

- A ***trained model*** expects structurally the same as training data.
- In linear regression, categorical features are “one-hot-encoded”. A feature with 3 categories expressed as a vector in 2 columns.
- What if a new category pops up?
  - Depends how you program it-
    - ignore the input
    - serve a bad response
- Consider clustering classification on text… new words?
  - Ignore: (probably what you’ll do)
  - Word might be very important...

Kappa with Novell Information

- In Kappa, training happens with each new piece of data
  – Model data can account for structural change in data instantly

- New words can be introduced into TF-IDF
- New categories into a factor variable
- Both examples (and others) causes input vector to change.

Statistical vs. Adversarial

<u>Traditional</u>

Common statistical methods

- Supervised
- Unsupervised

Graded by

- Statistical Fitness Tests
- Out of core testing
- E.g.
  – Confusion Matrix, AuROC
  – MSE, MAPE, R2, MSE

<u>Adversarial</u>

Algorithm Versus Environment

- vs. Spammers
- vs. Hackers
- vs. Nature

Graded by

- Directionally can use some tests
- Really A/B testing
  – Adversaries may get smarter over time
  – Type of test where you automate adversary.

Real-time

- Subjective
- A good buzzword for something that:
  - Doesn't fall into any of the above categories cleanly
  - Doesn't fall into the category you want it to fall into
  - You're not really sure which buzzword to use, so you need a 'safe' word that no one can call you on.
  - Days
  - Weeks?
  - JJs

Table of Contents

- Streaming K-Means
- Streaming Linear Regression
- Why would I ever do with this?

K-Means

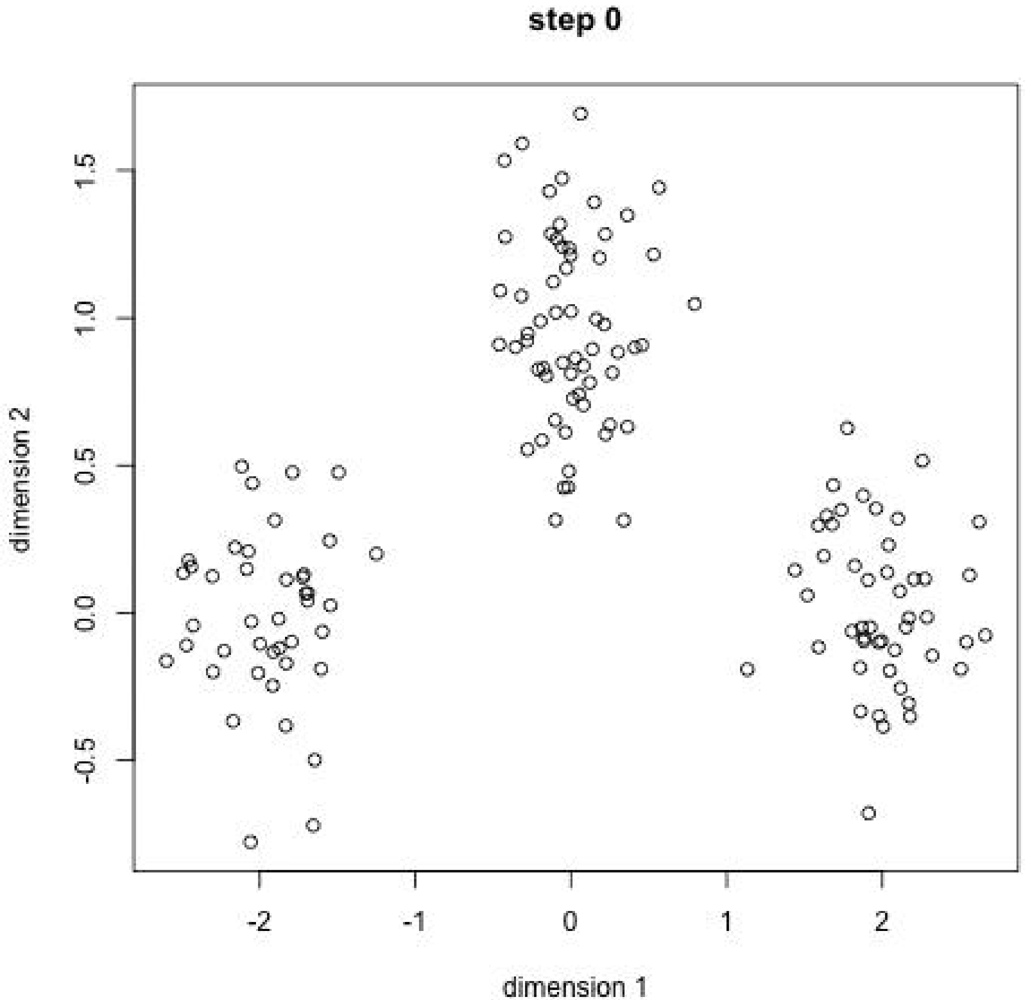
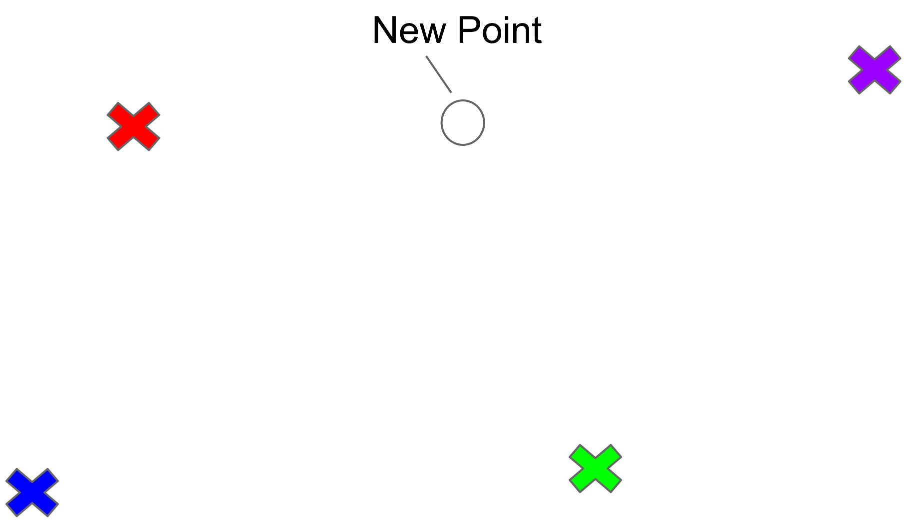


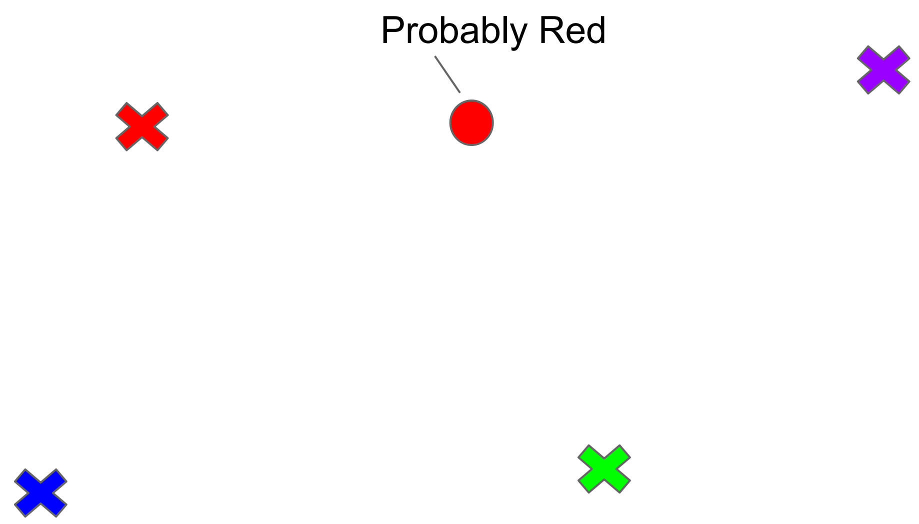
Photo Credit: Simply Statistics http://simplystatistics.org/wp-content/uploads/2014/02/kmeans.gif

Online K-Means

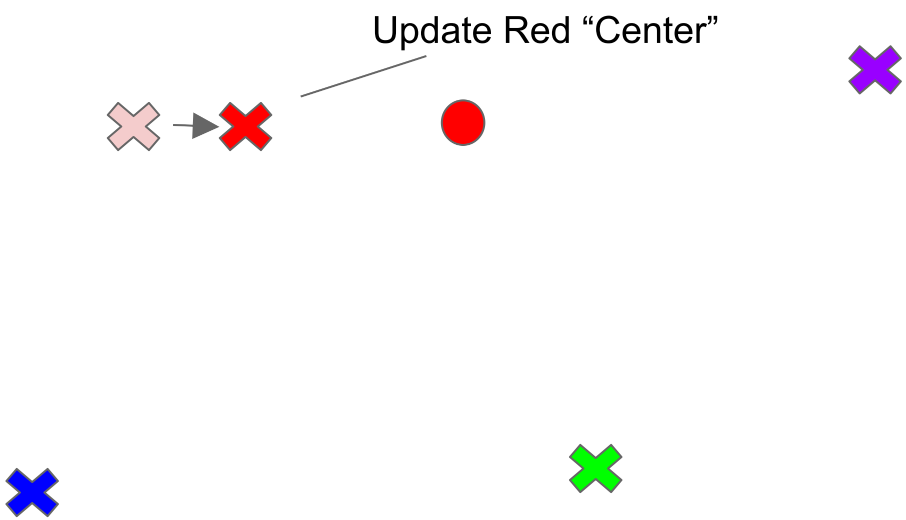
Online K-Means



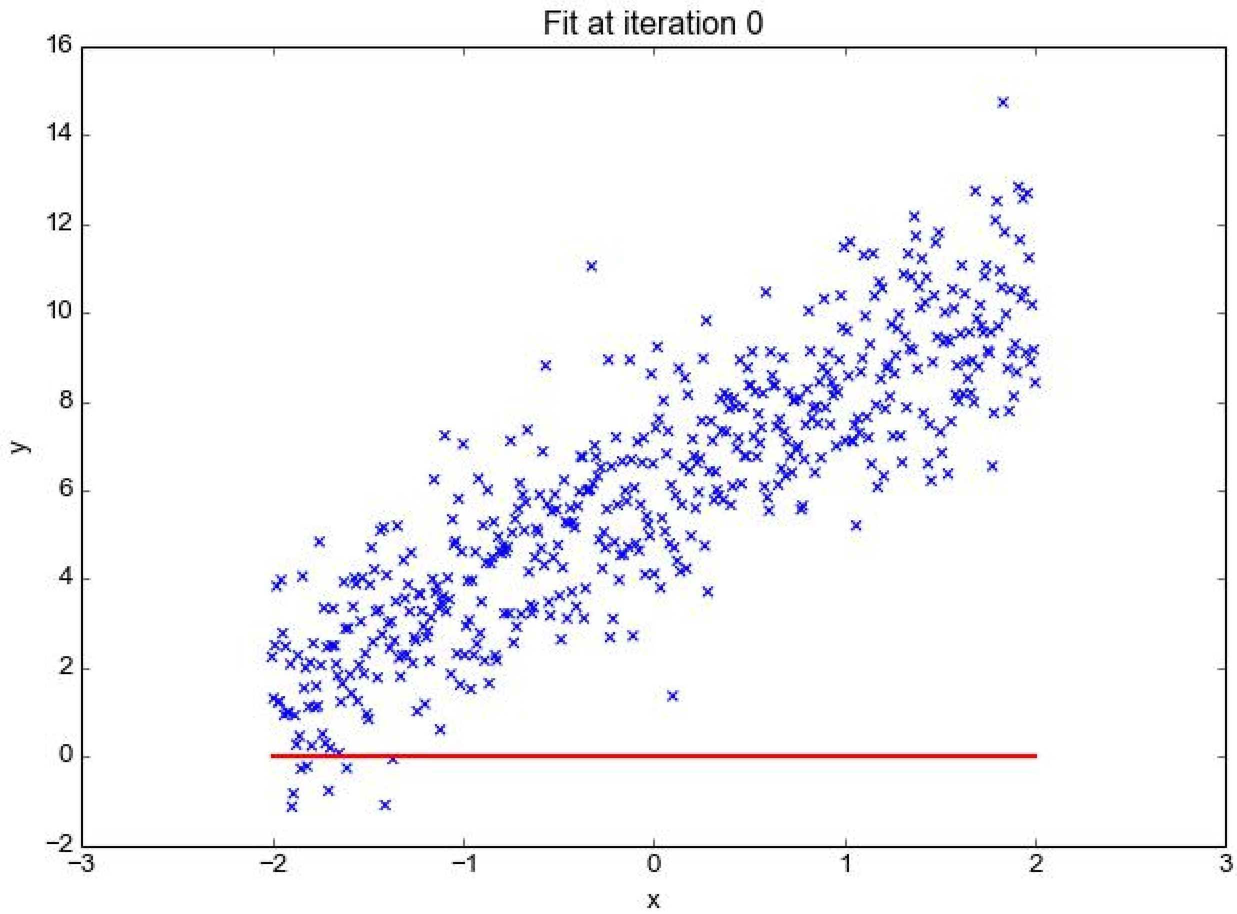
Online K-Means



Online K-Means

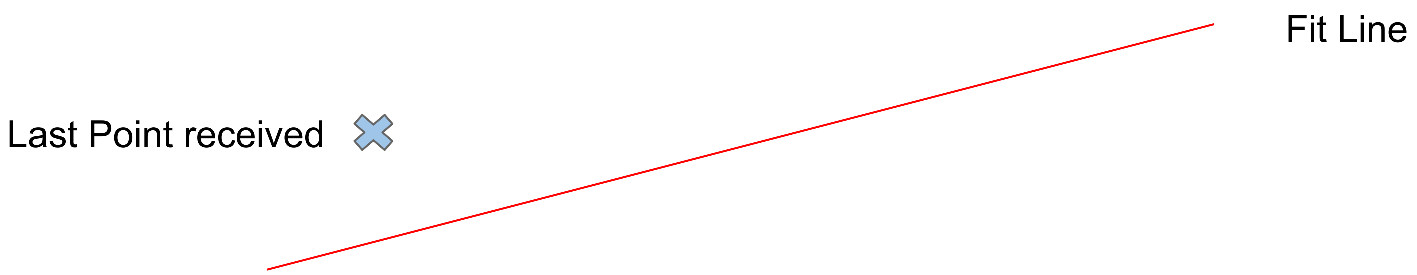


Linear Regression (Stochastic)

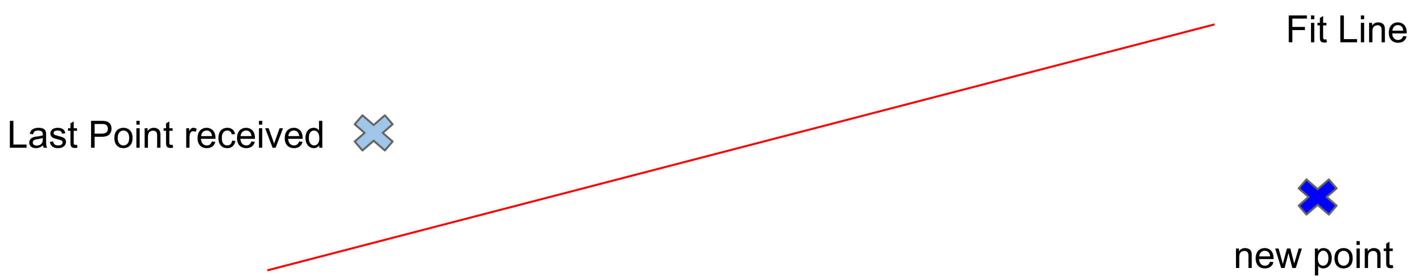


http://eli.thegreenplace.net/images/2016/regressionfit.gif

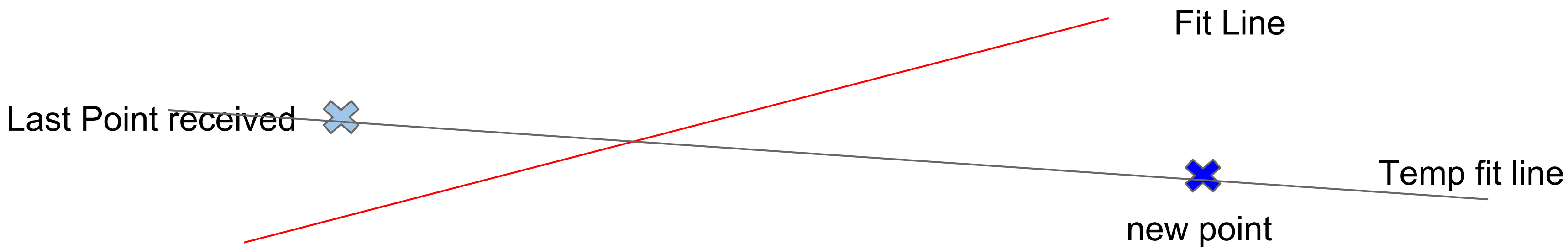
Online Linear Regression



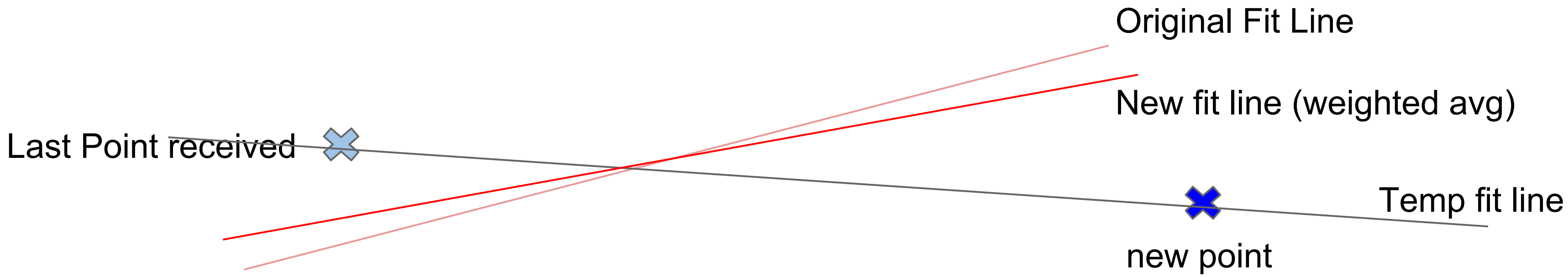
Online Linear Regression



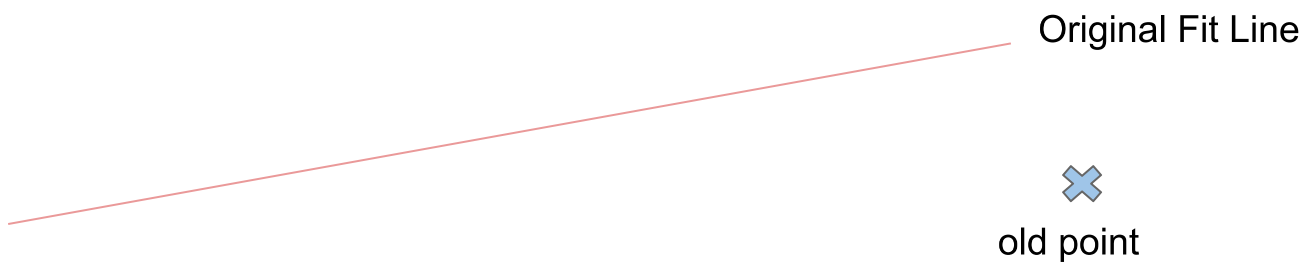
Online Linear Regression



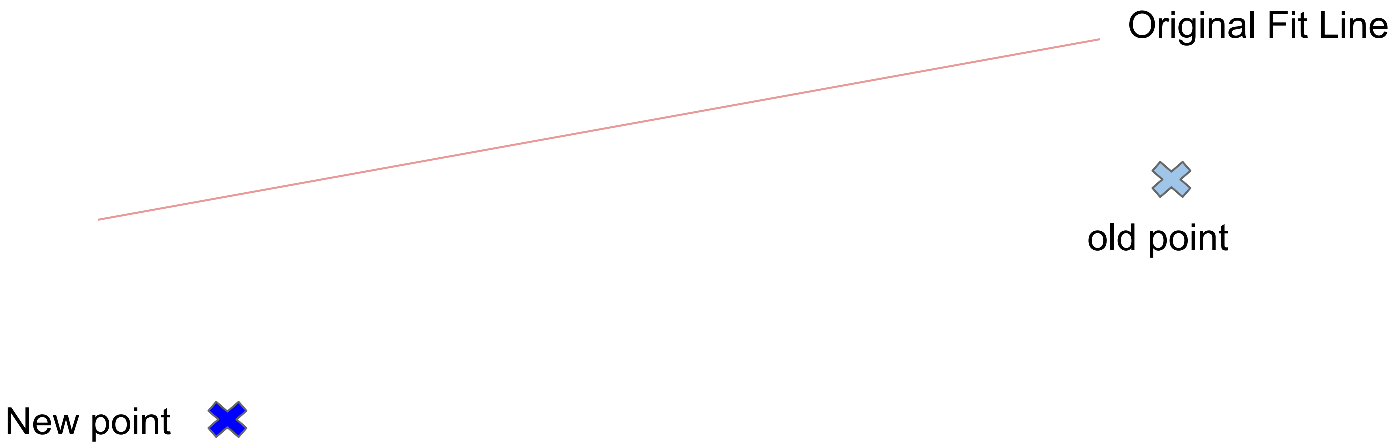
Online Linear Regression



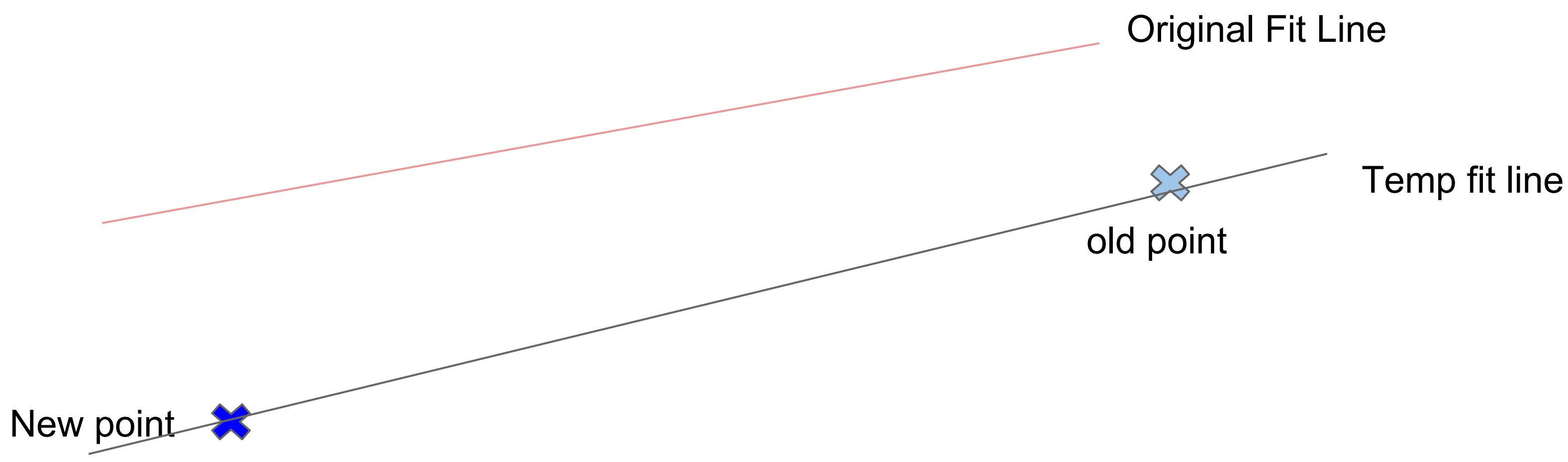
Online Linear Regression



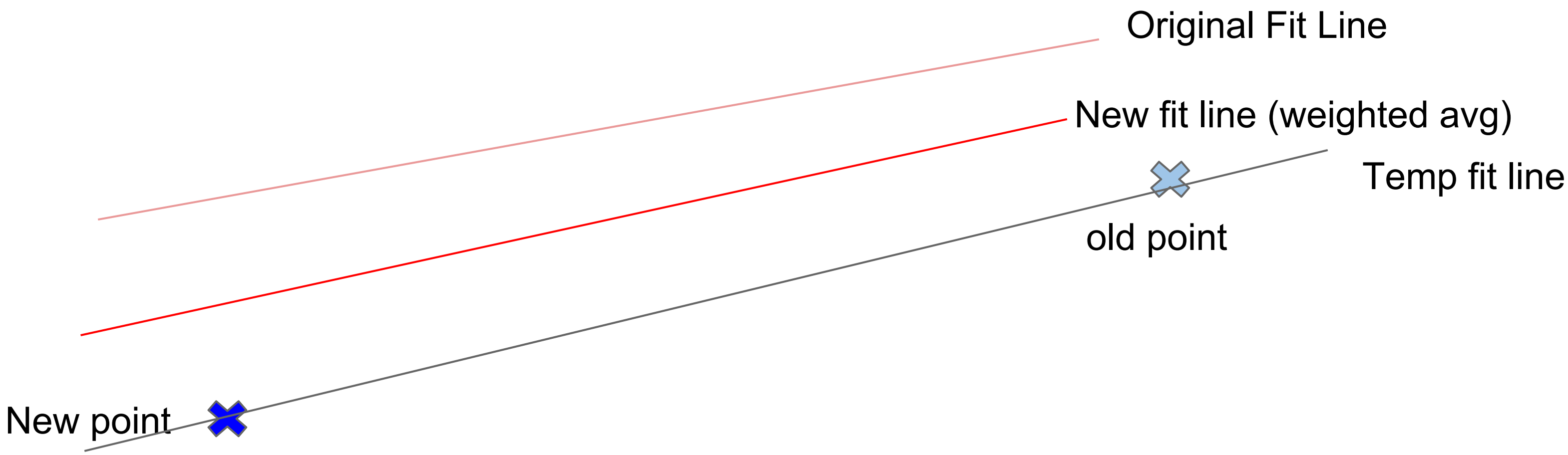
Online Linear Regression



Online Linear Regression



Online Linear Regression



Deep learning

This would work on neural networks too.

Also “Deep Learning” is another buzz word.

Why?

- Mostly Anomaly Detection (moving average, then something deviates)
  - A very popular use case of online/streaming algorithms (more talks today about this)
  - Algorithm learns what is normal (either online or offline)
  - When normality is sufficiently violated- the algorithm sounds an alarm
  - All anomaly detections some flavor of this. Usually referred to as: _______ Anomaly Detection, only to specify what algorithm was used for defining normality (or lack there-of).
  - <u>Architecture:</u> online-offline training choices depend primarily on how fast ‘normality’ changes in your specific use case

Table of Contents

Learning in real-time with supervised methods (challenge)

CHALLENGE:

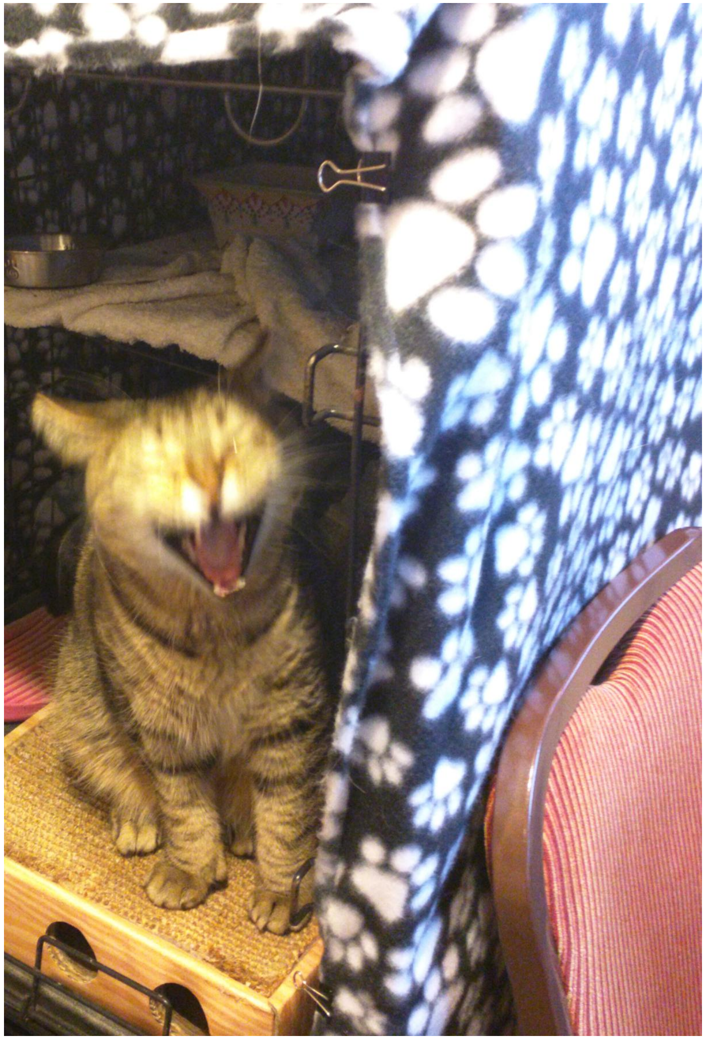
How do you know how far you ‘missed’ prediction? In real life ‘correct’ answers may arrive later.

Corollary: If you have ‘correct’ answer why are you trying to predict it?

Not insurmountable, but prevents ‘one size fits all’ approaches (context dependence).

Latency and Normal Streaming Problems

You've only got so much hardware.



Adversarial Analysis

Simple Adversary-
How well does the algorithm do against “offline” version?

Consider Linear Regression with SGD

- Offline algorithm gets over full data set, then predicts
- Online model gets single pass to train and predict

How much worse is online than offline?

A/B Tests- The gold standard

Online algos are often *interacting* with the environment.

Learning rates, other knobs.

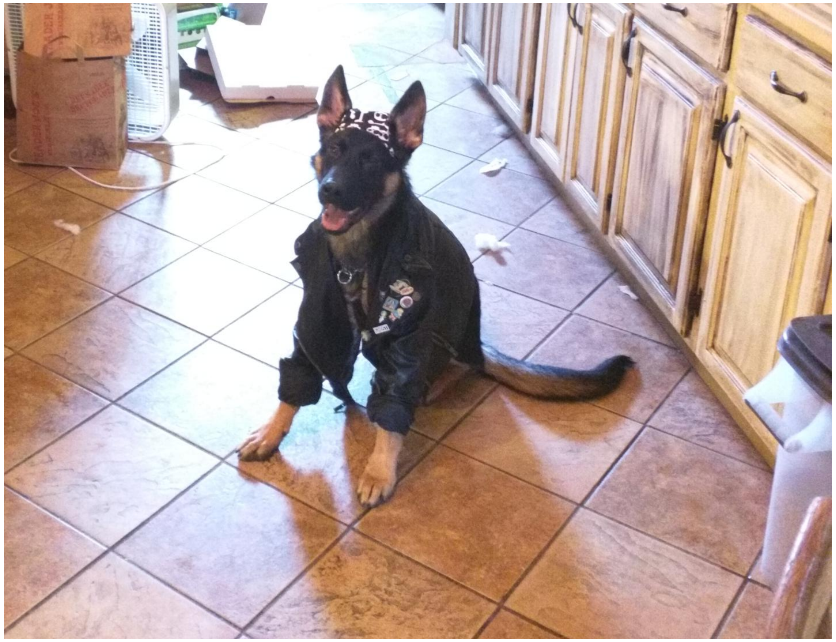


Table of Contents

Correlated Co Occurrence Recommender: Overview / Benefits

- Overview of CCO
  – Collaborative Filtering (Like ALS, etc.)
  – Behavior Based (also like ALS)
  – Uses co-occurrence (no matrix factorization, unlike ALS)
  – Multi-modal: more than one behavior considered (unlike ALS / CO)

- Benefits of CCO
  – Many types of behaviors can be considered at once
  – Can make recommendations for users never seen before.

CCO Math
A Simple Co-Occurrence Recommender

-

$$r = [P^T P] h_p$$

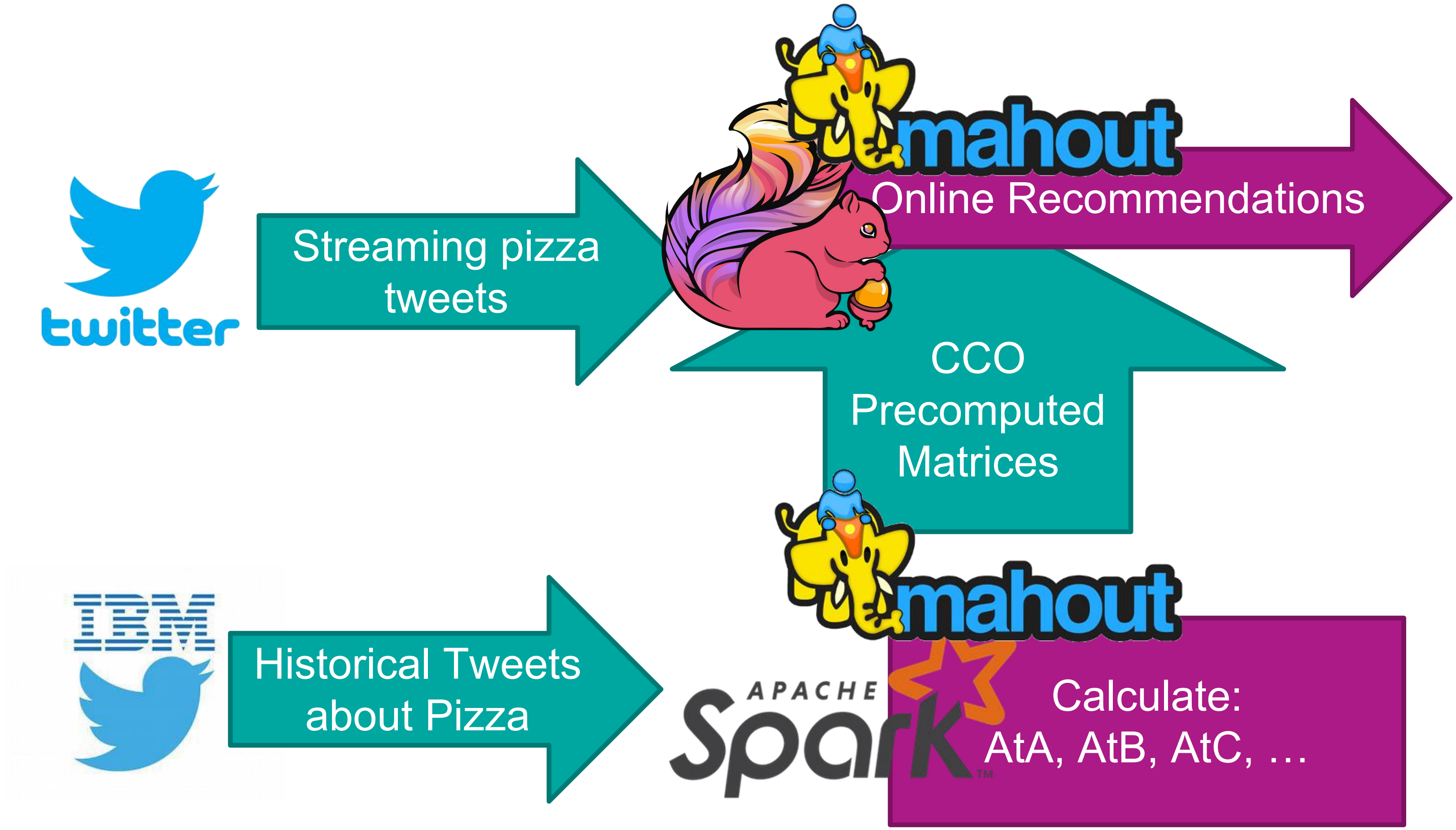
- r – recommendations
- P – history of all users on primary action (e.g. purchases)
  - Rows: user,
  - Columns: “Action” – e.g.(product1, product2, product3)
  - Then Row: Trevor, column: prodcuct2 => Trevor bought product 2
- [PtP] – Log Likelihood based correlation test
- hp- A user’s history on behavior p (could be new user)

CCO Math
Correlated Co-Occurrence Recommender

$$r = [P^T P]h_p + [P^T A]h_a + [P^T B]h_b + \ldots$$

- r – recommendations
- P – history of all users on primary action (e.g. purchases)
- [PtP] – Log Likelihood based correlation test
- A – history of all users on secondary action
  - Must have some rows (e.g. users)
- B – history of all users on tertiary action
  - Must have some rows (e.g. users)
- hp- A user's history on behavior p (could be new user)

Architecture: Lambda CCO
(Logo soup)



Python pulled in historical tweets and did this UserID - HashTag

561918328478785536,None
561918357851897858,None
561909179716481024,pizzagate
561909179716481024,gamergate
561949040011931649,None
561948991777038336,None
561947869805285377,superbowl
561947869805285377,pizzapizza
561918920282476545,None
561926796778565632,gunfriendly
561927577351503873,None

Python pulled in historical tweets and did this UserID - Words

561684486380068865,savethem000
561684486380068865,i
561684486380068865,dunno
561684486380068865,smiles
561684486380068865,want
561684486380068865,to
561684486380068865,get
561684486380068865,some
561684486380068865,pizza
561684486380068865,or
561684486380068865,something
561684441526194176,pizza
561684441526194176,de
561684441526194176,queso
561684441526194176,lista
561684441526194176,para

Some Spark Code

```
import org.apache.mahout.sparkbindings.indexeddataset.IndexedDatasetSpark
 import org.apache.mahout.math.cf.SimilarityAnalysis

val baseDir = "/home/rawkintrevo/gits/ffsf17-twitter-recos/data"
// We need to turn our raw text files into RDD[(String, String)]

val userFriendsRDD = sc.textFile(baseDir + "/user-friends.csv")
 .map(line => line.split(",")).filter(_.length == 2).map(a => (a(0), a(1)))
val userFriendsIDS = IndexedDatasetSpark.apply(userFriendsRDD)(sc)

val userHashtagsRDD = sc.textFile(baseDir + "/user-ht.csv")
 .map(line => line.split(",")).filter(_.length == 2).map(a => (a(0), a(1)))
val userHashtagsIDS = IndexedDatasetSpark.apply(userHashtagsRDD)(sc)

val userWordsRDD = sc.textFile(baseDir + "/user-words.csv")
 .map(line => line.split(",")).filter(_.length == 2).map(a => (a(0), a(1)))
val userWordsIDS = IndexedDatasetSpark.apply(userWordsRDD)(sc)

val hashtagReccosLlrDrmListByUser = SimilarityAnalysis.cooccurrencesIDSs(
 Array(userHashtagsIDS, userWordsIDS, userFriendsIDS),
 maxInterestingItemsPerThing = 100,
 maxNumInteractions = 500,
 randomSeed = 1234)
```


CCO Math
Spark+Mahout just Calculated these:

$$r = \boxed{[P^T P]} h_p + \boxed{[P^T A]} h_a + \boxed{[P^T B]} h_b + \ldots$$

- r – recommendations
- P – history of all users on primary action (e.g. purchases)
- [PtP] – Log Likelihood based correlation test
- A – history of all users on secondary action
  - Must have some rows (e.g. users)
- B – history of all users on tertiary action
  - Must have some rows (e.g. users)
- hp- A user's history on behavior p (could be new user)

Some Flink Code

```
streamSource.map(jsonString => {
 val result = JSON.parseFull(jsonString)

 val output = result match {
  case Some(e) => {
   /*************************************************************************************
    * Some pretty lazy tweet handling
   val tweet: Map[String, Any] = e.asInstanceOf[Map[String, Any]]
   val text: String = tweet("text").asInstanceOf[String]
   val words: Array[String] = text.split("\\s+").map(word => word.replaceAll("[^A-Za-z0-9]", "").toLowerCase())

   val entities = tweet("entities").asInstanceOf[Map[String, List[Map[String, String]]]]
   val hashtags: List[String] = entities("hashtags").toArray.map(m => m.getOrElse("text","").toLowerCase()).toList
   val mentions: List[String] = entities("user_mentions").toArray.map(m => m.getOrElse("id_str", "")).toList

   /************************************************************************************
    * Mahout CCO
   val hashtagsMat = sparse(hashtagsProtoMat.map(m => svec(m, cardinality = hashtagsBiDict.size)):_*)
   val wordsMat = sparse(wordsProtoMat.map(m => svec(m, cardinality= wordsBiDict.size)):_*)
   val friendsMat = sparse(friendsProtoMat.map(m => svec(m, cardinality = friendsBiDict.size)):_*)

   val userWordsVec = listOfStringsToSVec(words.toList, wordsBiDict)
   val userHashtagsVec = listOfStringsToSVec(hashtags, hashtagsBiDict)
   val userMentionsVec = listOfStringsToSVec(mentions, friendsBiDict)

   val reccos = hashtagsMat %*% userHashtagsVec + wordsMat %*% userWordsVec + friendsMat %*% userMentionsVec

   /*************************************************************************************
    * Sort and Pretty Print
```

CCO Math
Flink+Mahout just Calculated these:

$$r = [P^T P]\boxed{h_p} + [P^T A]\boxed{h_a} + [P^T B]\boxed{h_b} + \ldots$$

- r – recommendations
- P – history of all users on primary action (e.g. purchases)
- [PtP] – Log Likelihood based correlation test
- A – history of all users on secondary action
  - Must have some rows (e.g. users)
- B – history of all users on tertiary action
  - Must have some rows (e.g. users)
- hp- A user's history on behavior p (could be new user)

Tweets

text: joemalicki josephchmura well i can make a pizza i bet he cant so there
userWordsVec: so a well i there can he make pizza cant
hashtags used: List()
hashtags reccomended:
(**ruinafriendshipin5words** : 13.941270843461098)
(**worstdayin4words** : 8.93444123705558)
(recipes : 8.423061768672596)

text: people people dipping pizza in milk im done
userWordsVec: people in im done pizza
hashtags used: List()
hashtags reccomended:
(None : 18.560367273335828)
(**vegan** : 10.84782189800353)
(fromscratch : 10.84782189800353)

*Results were cherry picked- no preprocessing, this was a garbage in-garbage out algo for illustration purposes only.

Buzzword Soup

Cazognitive AI

Hybrid Lambda Architecture

Online Recommendations

GPU Accelerated

Adversarial

Algorithm

Also...

Don’t do this in real life, probably. (you would use a service)

Table of Contents

- Trevor attempts to tie everything together into a cohesive thought
- Audience members asks easy questions
- Audience members buy speaker beer at after party

Final Thoughts

A lot of buzzwords have been flying around especially with respect to machine learning and streaming.

- Online
- Lambda / Kappa architecture
- Streaming machine learning
- Real time predictive model
- machine learning
- artificial/machine/cognitive intelligence
- cognitive
- blah- ^^ pick 2.

  10 April 2017

Final Thoughts

Now that you've sat through this talk hopefully you can:

1. Call people out for trying to make their product/service/open source project/startup sound like a bigger deal than it is
2. Church up your product/service/open source project/startup to get clients/VC dummies excited about it without *technically* lying

Questions?

Buy trevor beers.

https://github.com/rawkintrevo/fsf17-twitter-recos

