Submodular optimization for machine learning

Jacob Schreiber
Department of Genetics
Stanford University


jmschreiber91


@jmschrei

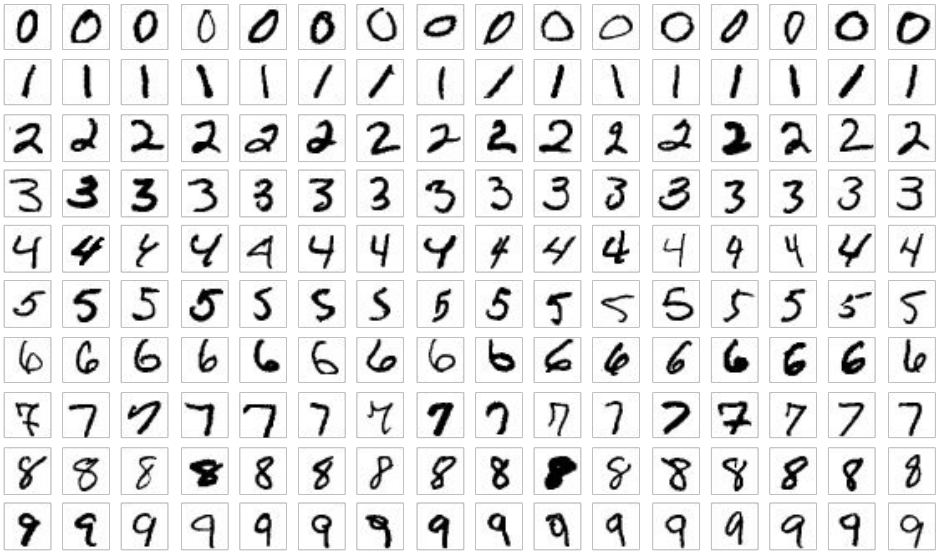

@jmschreiber91

Data sets have been getting increasingly large over time

Iris (1936): 150 examples, 4 features

https://en.wikipedia.org/wiki/Iris_flower_data_set#/media/File:Kosaciec_szczecinkowaty_Iris_setosa.jpg

Data sets have been getting increasingly large over time

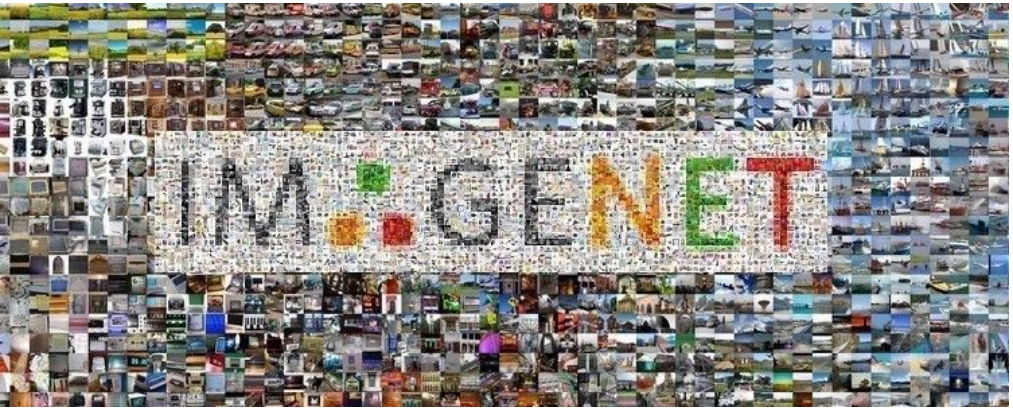


Iris (1936): 150 examples, 4 features

MNIST (1998): 70,000 images, 784 pixels

https://en.wikipedia.org/wiki/MNIST_database#/media/File:MnistExamples.png

Data sets have been getting increasingly large over time



Iris (1936): 150 examples, 4 features

MNIST (1998): 70,000 images, 784 pixels

ImageNet (2010): 14,197,122 images, 50,176 pixels

https://en.wikipedia.org/wiki/MNIST_database#/media/File:MnistExamples.png

Unfortunately, increases in size generally correspond to increases in redundancy

Many duplicates have been found in CIFAR10 and CIFAR100

- Redundant examples in training sets can waste computational resources
- Redundant examples between training and test sets can cause overly optimistic estimates of performance

Barz and Denzler (2020), https://www.mdpi.com/2313-433X/6/6/41

Submodular functions are set functions that exhibit diminishing returns

Specifically, they must take the following form:

$$f(X \cup \{v\}) - f(X) \geq f(Y \cup \{v\}) - f(Y)$$

where $X \subseteq Y$ and $v \notin Y$

The diminishing returns property states that the gain one observes from adding in some specific element *v* to *X* decreases, or stays the same, as other elements are added to X.

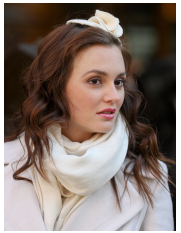
These functions do not need to be continuous or differentiable, so you can be very creative in crafting functions to suit your purpose.

Critically, theory exists showing that greedy algorithms will find a subset within a constant factor of 1 - e^-1 of the optimal subset, and empirical results show that the subset if almost always much closer.

Imagine you need to choose a representative cast of characters from *Gossip Girl*

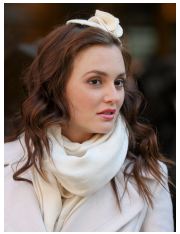
Potentially, you could craft a function that takes in one or more characters and returns the number of scenes that one or more of them appear in, i.e., for each scene, add one if any of the characters in the set appears in it, otherwise add zero.

f() = 10

f() = 17

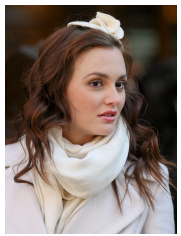
Imagine you need to choose a representative cast of characters from *Gossip Girl*

Potentially, you could craft a function that takes in one or more characters and returns the number of scenes that one or more of them appear in, i.e., for each scene, add one if any of the characters in the set appears in it, otherwise add zero.

f( , ) = 22

Imagine you need to choose a representative cast of characters from *Gossip Girl*

Potentially, you could craft a function that takes in one or more characters and returns the number of scenes that one or more of them appear in, i.e., for each scene, add one if any of the characters in the set appears in it, otherwise add zero.

f( ,  , ) = 23

Imagine you need to choose a representative cast of characters from *Gossip Girl*

Potentially, you could craft a function that takes in one or more characters and returns the number of scenes that one or more of them appear in, i.e., for each scene, add one if any of the characters in the set appears in it, otherwise add zero.

… but after a useful function is identified, how do you use it to select a good subset?

A simple class of submodular functions are feature-based functions

$$f(X) = \sum_{d=1}^{D} \phi \left(\sum_{x \in X} x_d \right)$$

Where **X** is the set of selected examples, **x** is an individual example, **D** is the number of dimensions in the data set, **d** is the index of a particular dimension, and phi is a concave function like sqrt or log

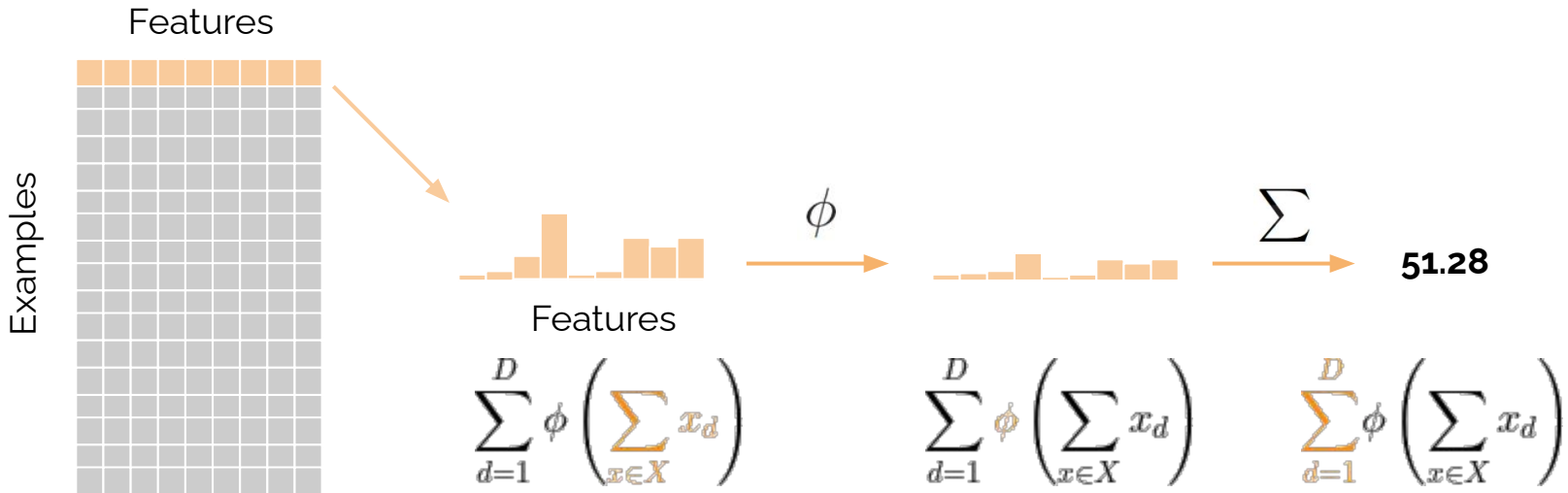
Features

Examples

A simple class of submodular functions are feature-based functions

$$f(X) = \sum_{d=1}^{D} \phi \left(\sum_{x \in X} x_d \right)$$

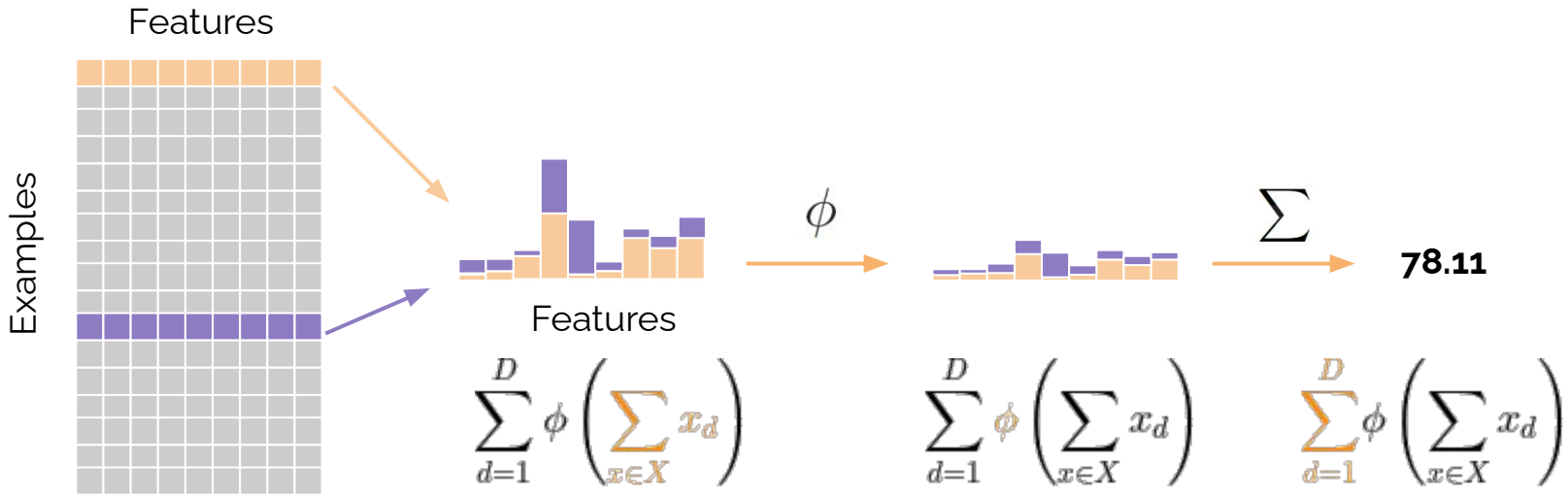
Where **X** is the set of selected examples, **x** is an individual example, **D** is the number of dimensions in the data set, **d** is the index of a particular dimension, and phi is a concave function like sqrt or log



A simple class of submodular functions are feature-based functions

$$f(X) = \sum_{d=1}^{D} \phi \left(\sum_{x \in X} x_d \right)$$

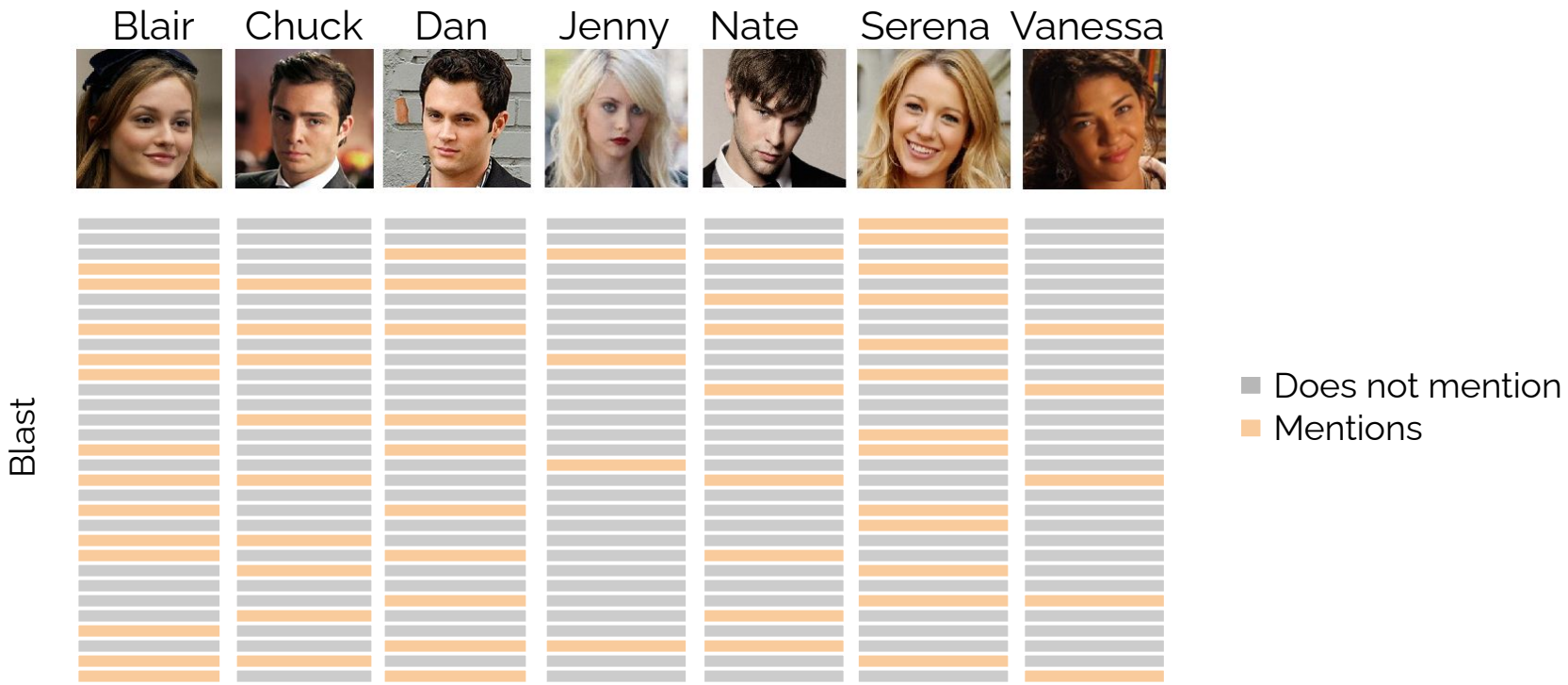
Where **X** is the set of selected examples, **x** is an individual example, **D** is the number of dimensions in the data set, **d** is the index of a particular dimension, and phi is a concave function like sqrt or log



Now imagine you need to choose a representative set of blasts...

Spotted: Lonely Boy. Can't believe the love of his life has returned. If only she knew who he was. But everyone knows Serena. And everyone is talking. Wonder what Blair Waldorf thinks. Sure, they're BFF's, but we always thought Blair's boyfriend Nate had a thing for Serena.

We can encode each blast based on whether it mentions a character



The top two blasts returned by feature-based selection seem to scorch everyone

Not so fast. You're not graduating until I give you my diplomas. Mine are labels, and labels stick. ***Nate Archibald****: [something not nice].* ***Dan Humphrey:*** *The ultimate insider.* ***Chuck Bass:*** *Coward.* ***Blair Waldorf:*** *Weakling. And as for* ***Serena van der Woodsen****, after today, you are officially irrelevant. Congratulations, everyone. You deserve it.*

Poor ***Vanessa****. Even Cinderella was given the courtesy of a stealth getaway. Then again, what's a trio of lovely stepsisters compared to* ***Jenny Humphrey****. Looks like it might turn out to be an "Unhappily Ever After" for everyone.*

Submodular optimization can also be applied to graph-based functions

Facility location is one of the most widely used submodular functions, and has literally been used to specify the locations of new facilities.

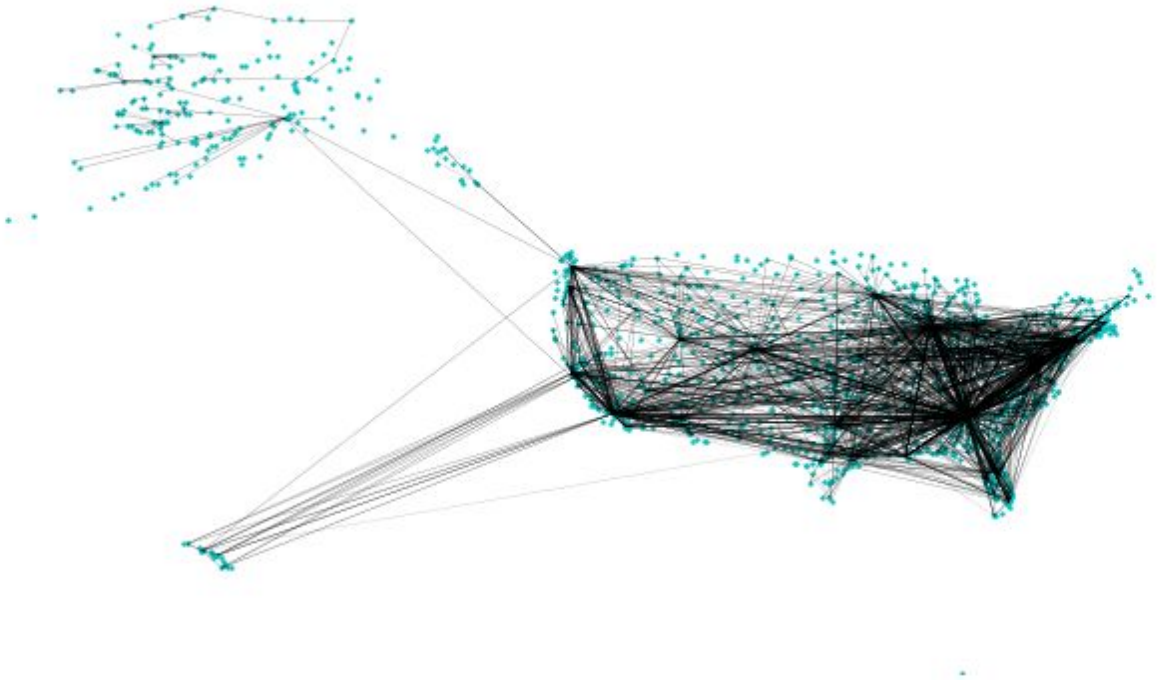
$$f(X) = \sum_{y \in Y} \max_{x \in X} \phi(x, y)$$

Where **Y** is the full set of items, **X** is the set of selected items, phi is a similarity function that returns the similarity between two examples (i.e., a graph) and **y** and **x** are individual examples.

Facility location can be used to choose airport hubs that cover the most routes

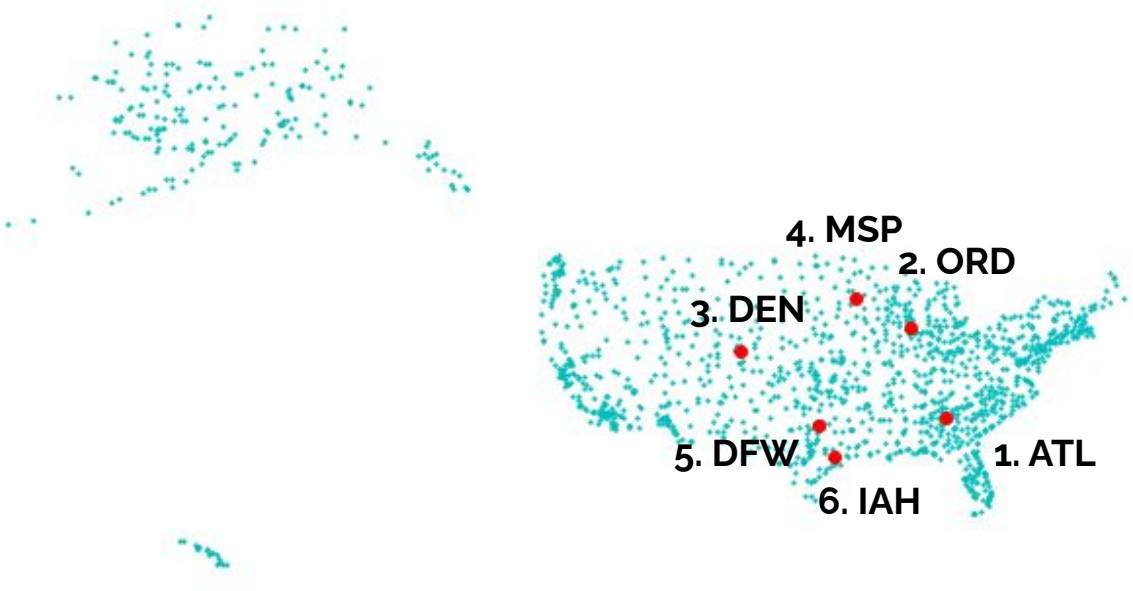
data from openflights.org

Plotting airline routes can show us that hubs form naturally



data from openflights.org

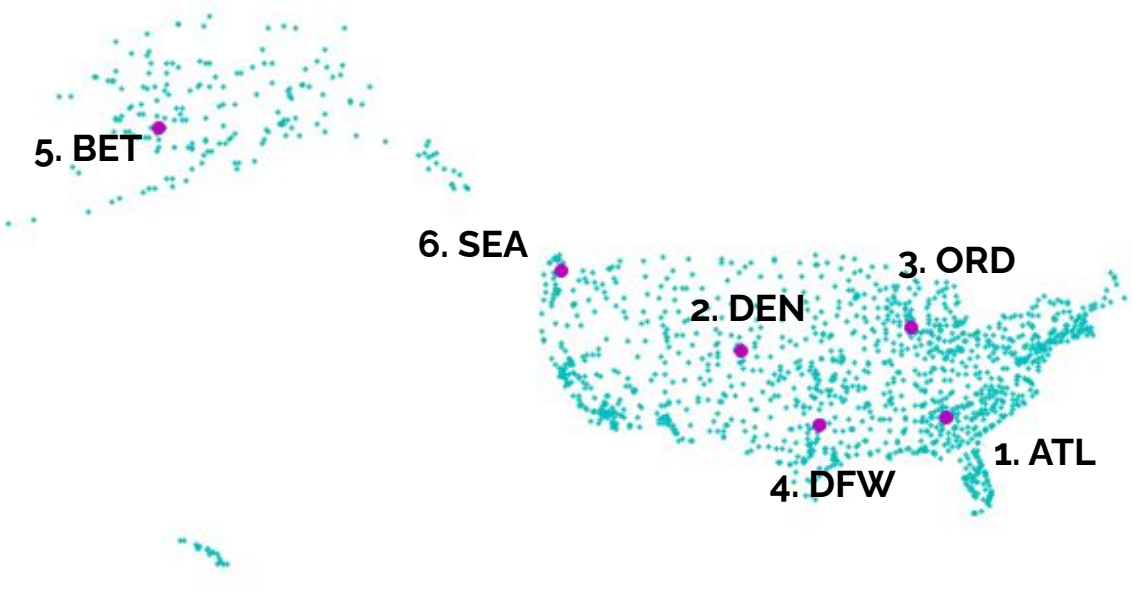
Choosing the airports with the most flights out gives a reasonable set of hubs



These airports service **256** unique routes

data from openflights.org

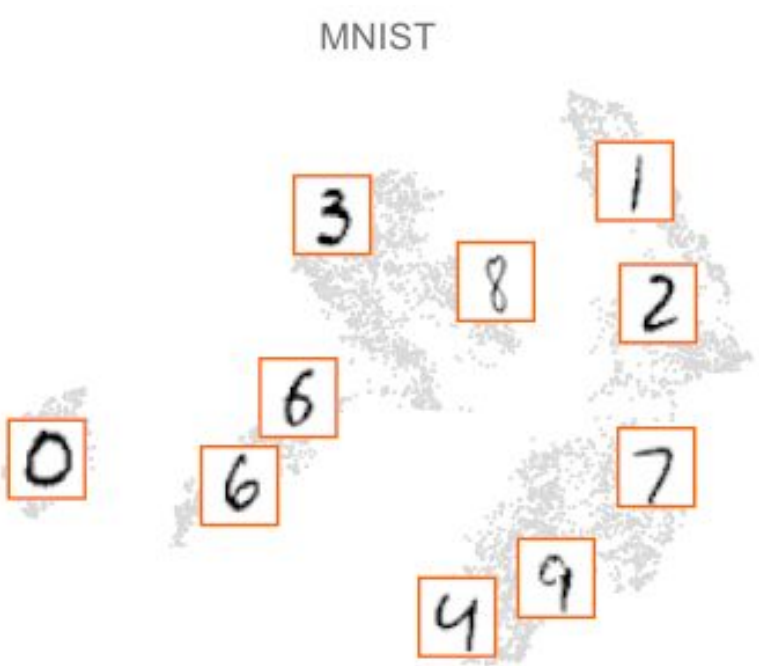
Using facility location gives an even better set of hubs



These airports service **270** unique routes

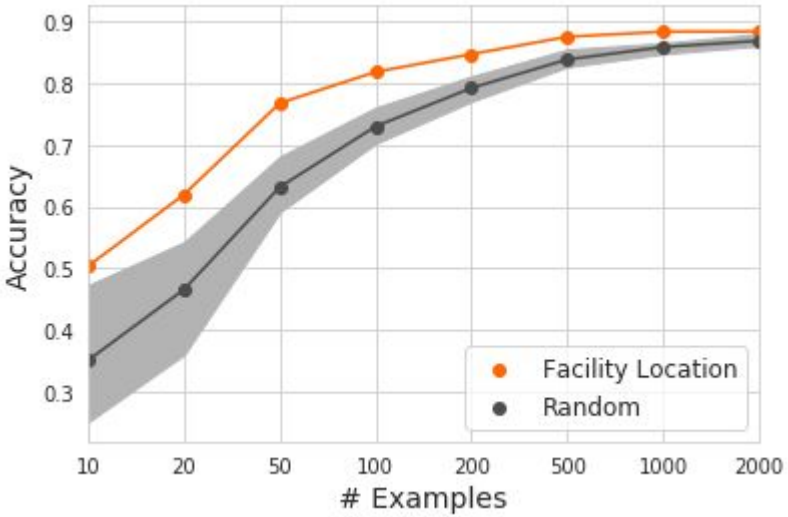
data from openflights.org

Submodular optimization can summarize large data sets into compact sets

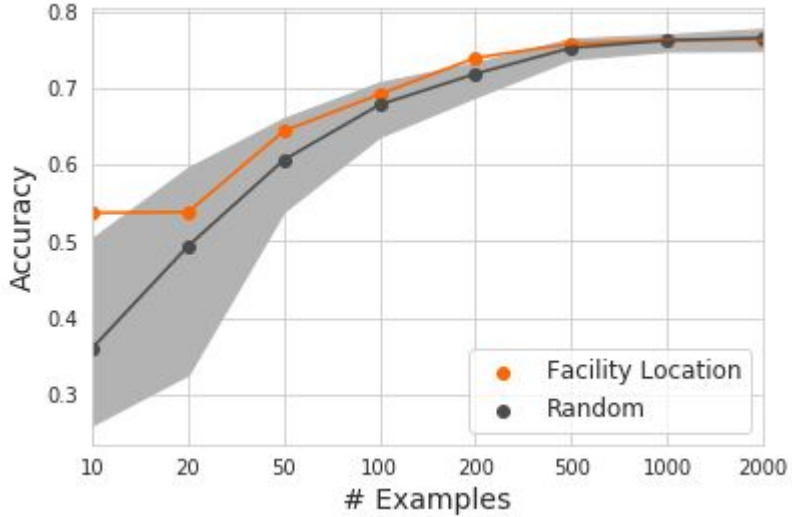


The selected subset can train ML models with a fraction of the examples

MNIST



Fashion MNIST



Choosing the right submodular function is crucial

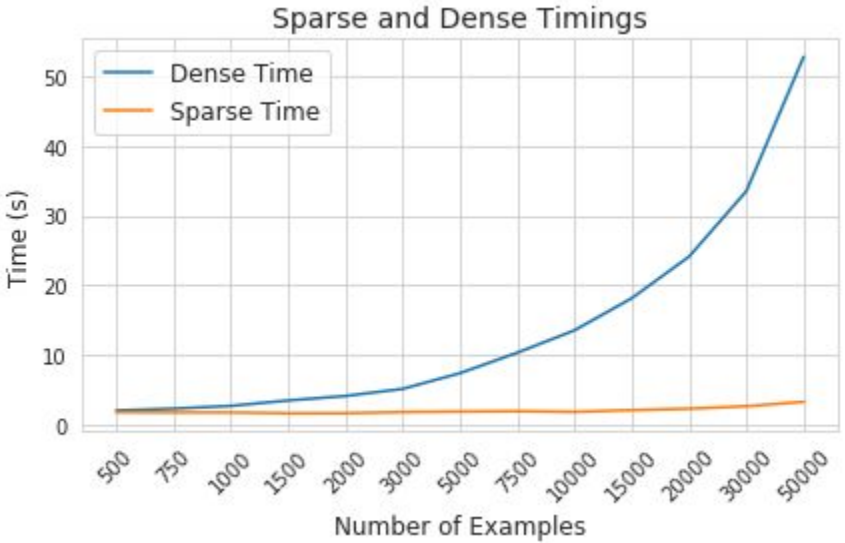
In the same way that choosing the right neural network architecture is critical for modeling data well (e.g., convolutions on images instead of dense layers, transformers with positional encodings for sequences of data), choosing the right submodular function is crucial here:

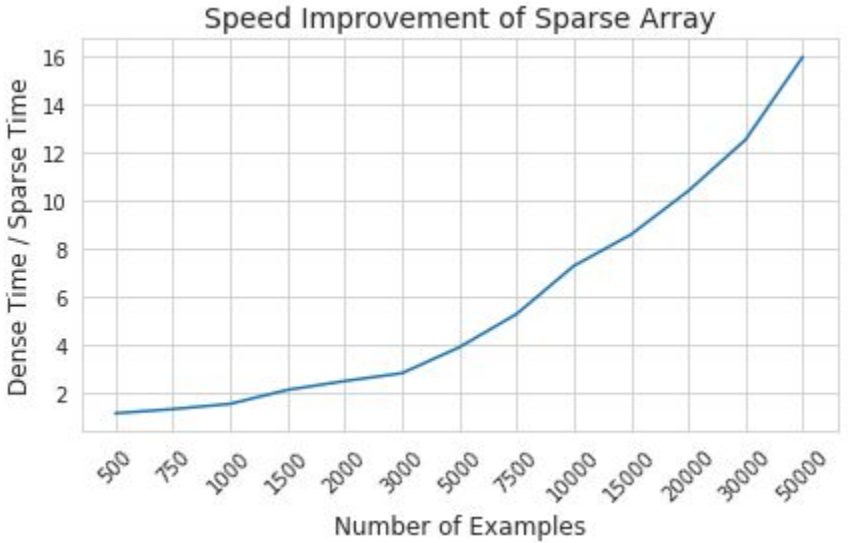
- Feature-based functions work well when each feature is some “quality” of the data and a higher value means more of that quality, i.e., word counts work well, pixel values do not

- Graph-based functions, like facility location, are general purpose and can work well on a variety of settings but require quadratic memory to store the similarity matrix

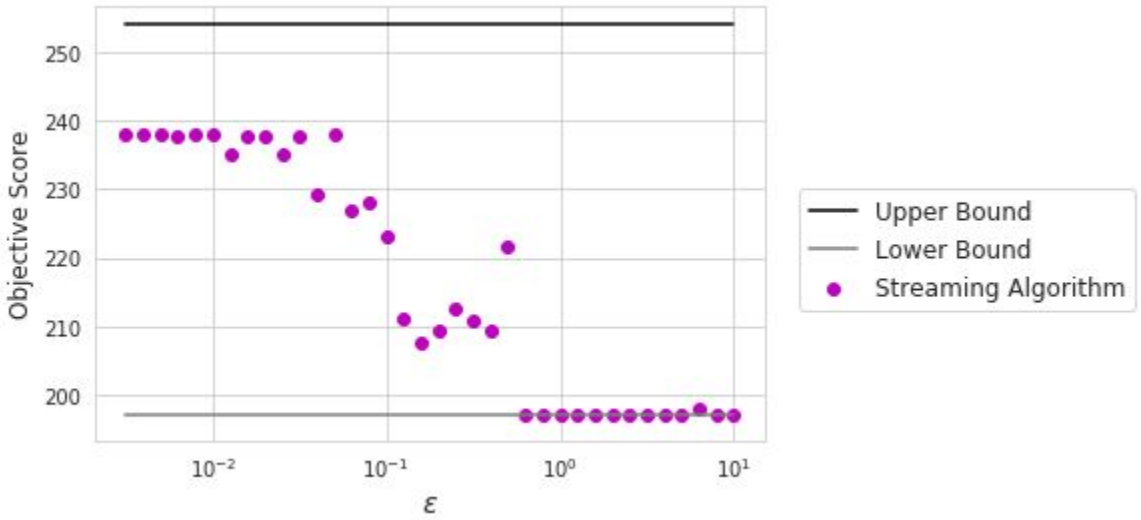
Using sparse data formats can significantly speed up optimization

Consider a random binary matrix that is only 1% non-zeroes, such as a word count matrix, a sparse topic model output, etc..





Streaming submodular optimization can do selection in one pass



As the granularity of the search gets smaller we can find better subsets but they'll rarely be as good as if we ran the original algorithm

So, what's in apricot?

- A variety of built-in submodular functions and optimizers that can be used in a plug-and-play manner with each other, similar to neural network components and optimizers in libraries like PyTorch or TensorFlow

- Several speed optimizations not talked about here, including a numba-based implementation, kNN approximations of similarity matrices, approximate greedy optimizers, and more

- Support for custom functions and optimizers that can work with existing code

- Support for building off of an initial set of selected points to find non-redundant examples with that set

If you'd like to use submodular selection in your own work...

apricot

Submodular optimization for machine learning

downloads 43k | build passing | docs passing

Please consider citing the manuscript if you use apricot in your academic work!

You can find more thorough documentation here.

apricot implements submodular optimization for the purpose of summarizing massive data sets into minimally redundant subsets that are still representative of the original data. These subsets are useful for both visualizing the modalities in the data (such as in the two data sets below) and for training accurate machine learning models with just a fraction of the examples and compute.

MNIST

Fashion MNIST

https://github.com/jmschrei/apricot

apricot: Submodular selection for data summarization in Python

Jacob Schreiber JMSCHR@CS.WASHINGTON.EDU
Paul G. Allen School of Computer Science and Engineering, University of Washington
Seattle, WA 98195-4322, USA

Jeffrey Bilmes BILMES@UW.EDU
Department of Electrical & Computer Engineering, University of Washington
Seattle, WA 98195-4322, USA

William Stafford Noble WILLIAM-NOBLE@UW.EDU
Department of Genome Science, University of Washington
Seattle, WA 98195-4322, USA

https://www.jmlr.org/papers/v21/19-467.html