pomegranate

fast and flexible probabilistic modelling in python

Jacob Schreiber
Paul G. Allen School of Computer Science & Engineering
University of Washington

jmschreiber91
@jmschrei
@jmschreiber91

Acknowledgements

UNIVERSITY *of* WASHINGTON
eScience Institute
ADVANCING DATA-INTENSIVE DISCOVERY IN ALL FIELDS

Inria
INVENTORS FOR THE DIGITAL WORLD

PARIETAL

scikit
learn

Overview

pomegranate is **more flexible** than other packages, **fast**, is **intuitive to use**, and can do it all **in parallel**

pomegranate supports many models

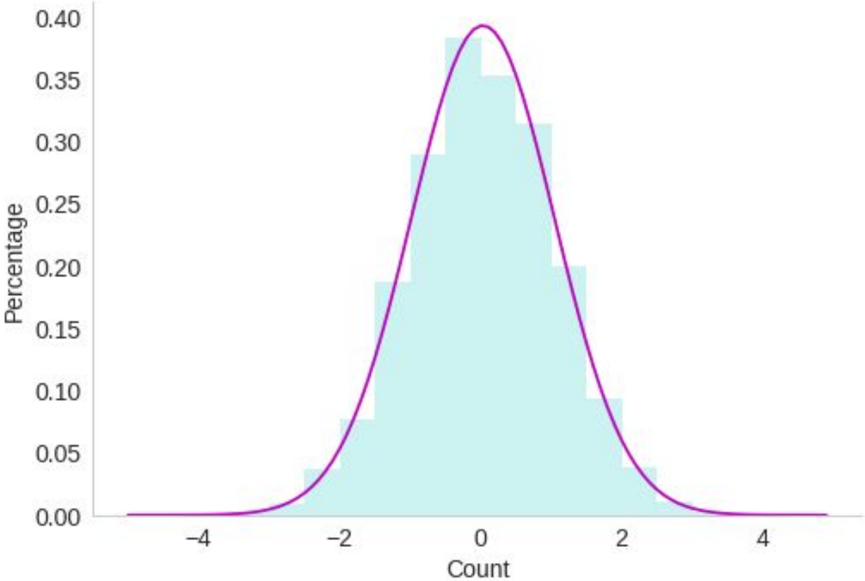
Probability Distributions
General Mixture Models
Hidden Markov Models
Naive Bayes / Bayes' Classifiers
Markov Chains
Bayesian Networks

k-means / kmeans++ / kmeans||
Factor graphs

Models can be made in two ways...

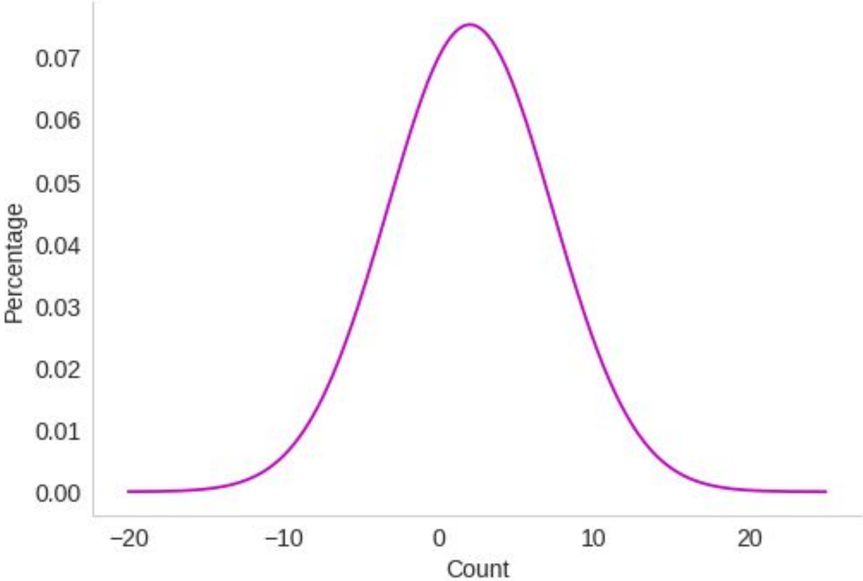
…from data

`d = NormalDistribution.from_samples(X)`



…from known values

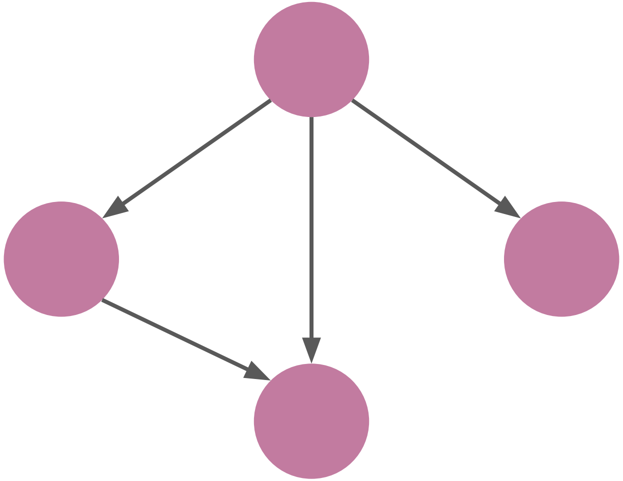
`d = NormalDistribution(5, 2.3)`



Models can be made in two ways...

...from data

```
d = BayesianNetwork.from_samples(X)
```



...from known values

```
n1 = Node(...)
n2 = Node(...)
model = BayesianNetwork()
model.add_nodes(n1, n2...)
model.add_edges(...)
```

The API is common to all models

model.log_probability(X) / model.probability(X)

model.sample()

model.fit(X, weights, inertia)

model.summarize(X, weights)

model.from_summaries(inertia)

Model.from_samples(X, weights)

All models have these methods!

model.predict(X)

model.predict_proba(X)

model.predict_log_proba(X)

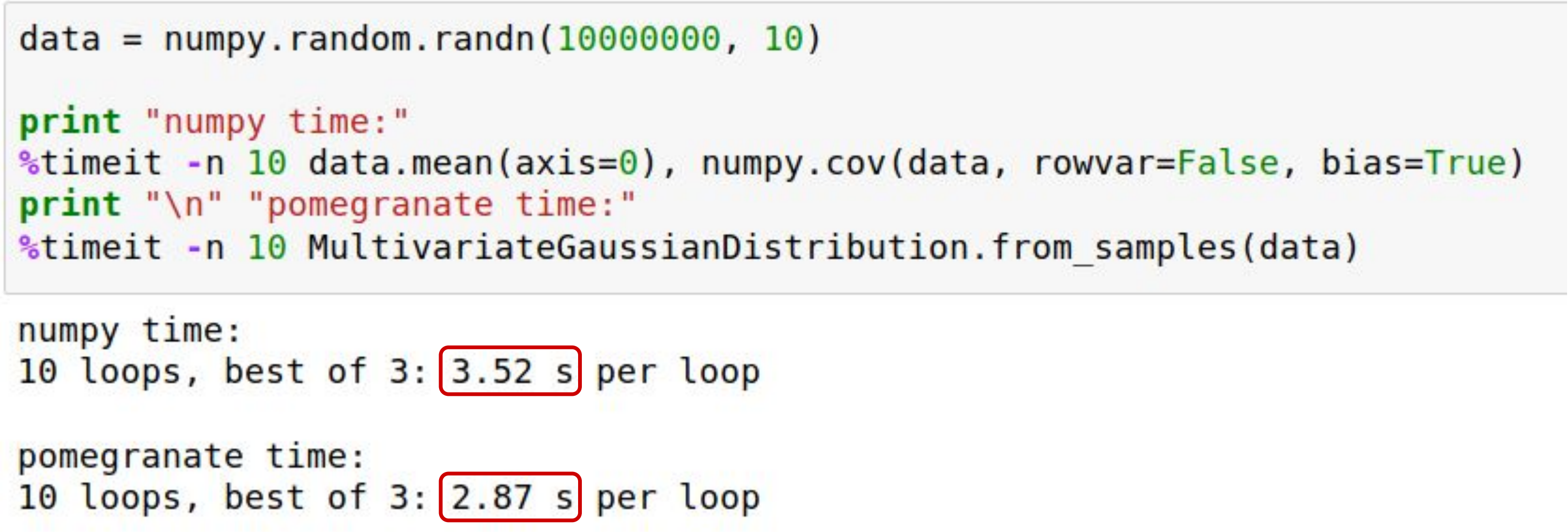
All models composed of distributions (like GMM, HMM...) have these methods too!

Overview: model stacking in pomegranate

```
GeneralMixtureModel.from_samples(NormalDistribution, n_components=3, X=X)

GeneralMixtureModel.from_samples(ExponentialDistribution, n_components=3,
X=X)


BayesClassifier.from_samples(MultivariateGaussianDistribution, X, y)


d1 = GeneralMixtureModel.from_samples...
d2 = GeneralMixtureModel.from_samples...
model = BayesClassifier([d1, d2])
```

pomegranate can be faster than numpy

Fitting Multivariate Gaussian to 10,000,000 samples of 10 dimensions



```
data = numpy.random.randn(10000000, 10)

print "numpy time:"
%timeit -n 10 data.mean(axis=0), numpy.cov(data, rowvar=False, bias=True)
print "\n" "pomegranate time:"
%timeit -n 10 MultivariateGaussianDistribution.from_samples(data)
```

```
numpy time:
10 loops, best of 3: 3.52 s per loop

pomegranate time:
10 loops, best of 3: 2.87 s per loop
```

pomegranate uses additive summarization

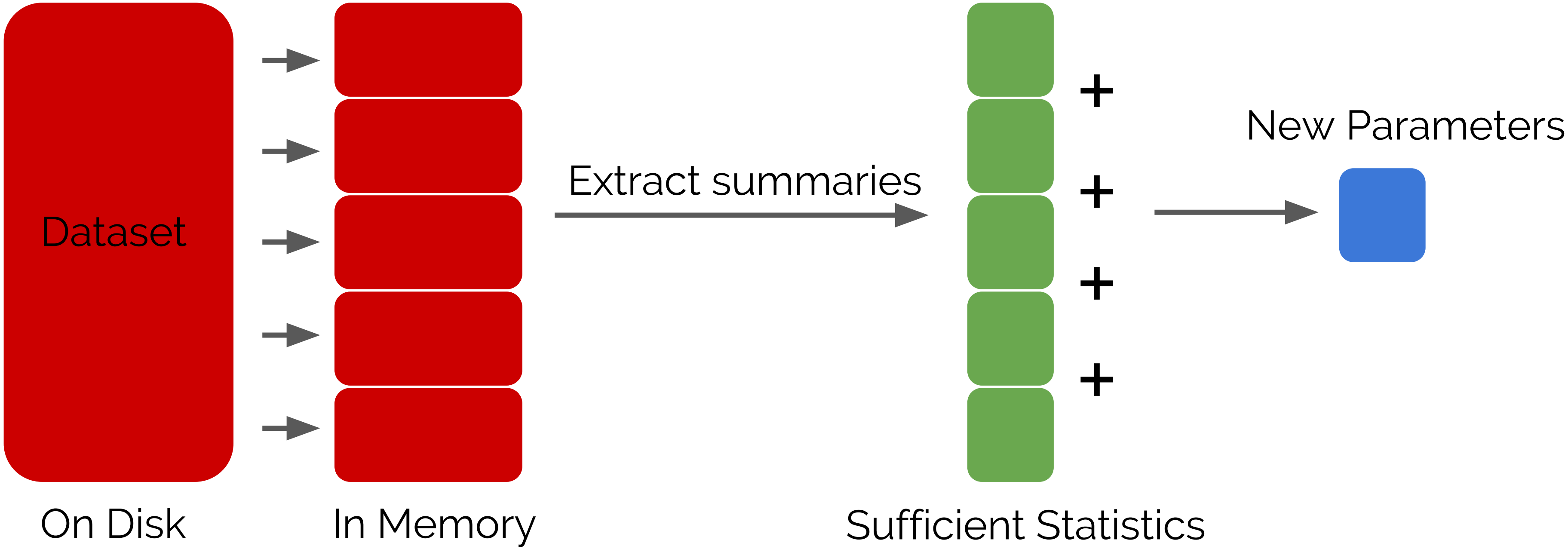
pomegranate reduces data to sufficient statistics for updates and so only has to go datasets once (for all models).

Here is an example of the Normal Distribution sufficient statistics

$$\sum_{i=1}^{n} w_i \quad \sum_{i=1}^{n} w_i x_i \quad \sum_{i=1}^{n} w_i x_i^2 \longrightarrow \begin{aligned} \mu &= \frac{\sum_{i=1}^{n} w_i x_i}{\sum_{i=1}^{n} w_i} \\ \sigma^2 &= \frac{\sum_{i=1}^{n} w_i x_i^2}{\sum_{i=1}^{n} w_i} - \frac{\left(\sum_{i=1}^{n} w_i x_i\right)^2}{\left(\sum_{i=1}^{n} w_i\right)^2} \end{aligned}$$

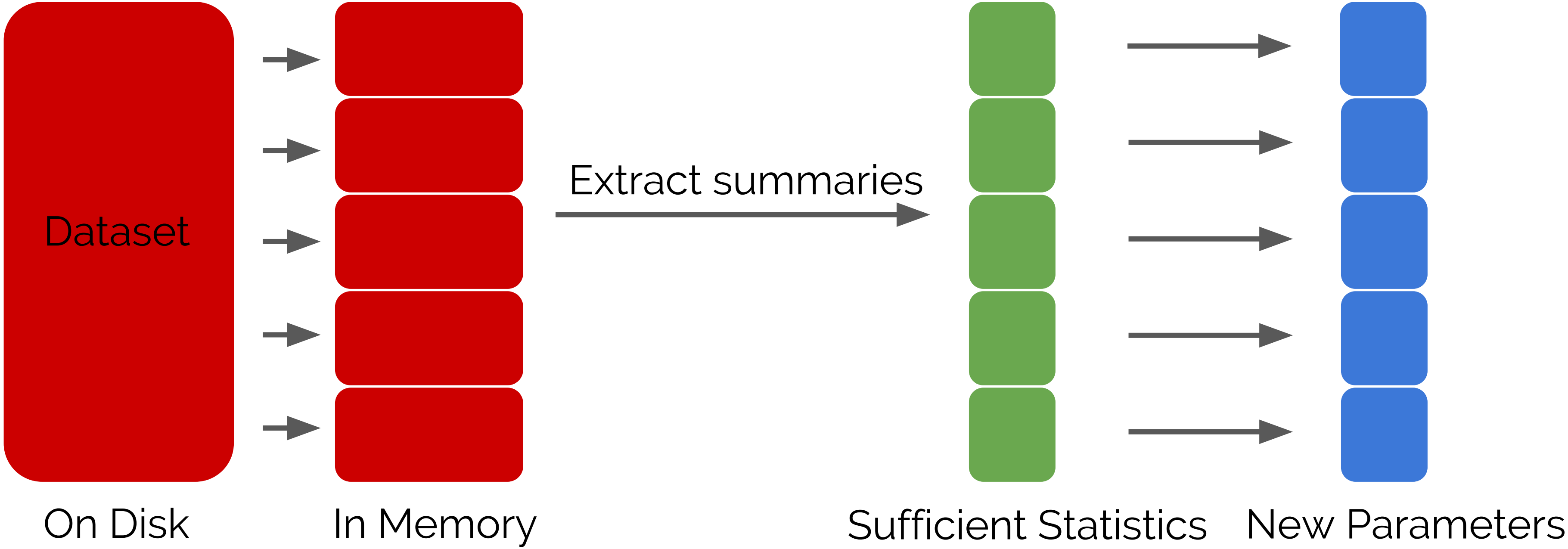
pomegranate supports out-of-core learning

Batches from a dataset can be reduced to additive summary statistics, enabling exact updates from data that can't fit in memory.



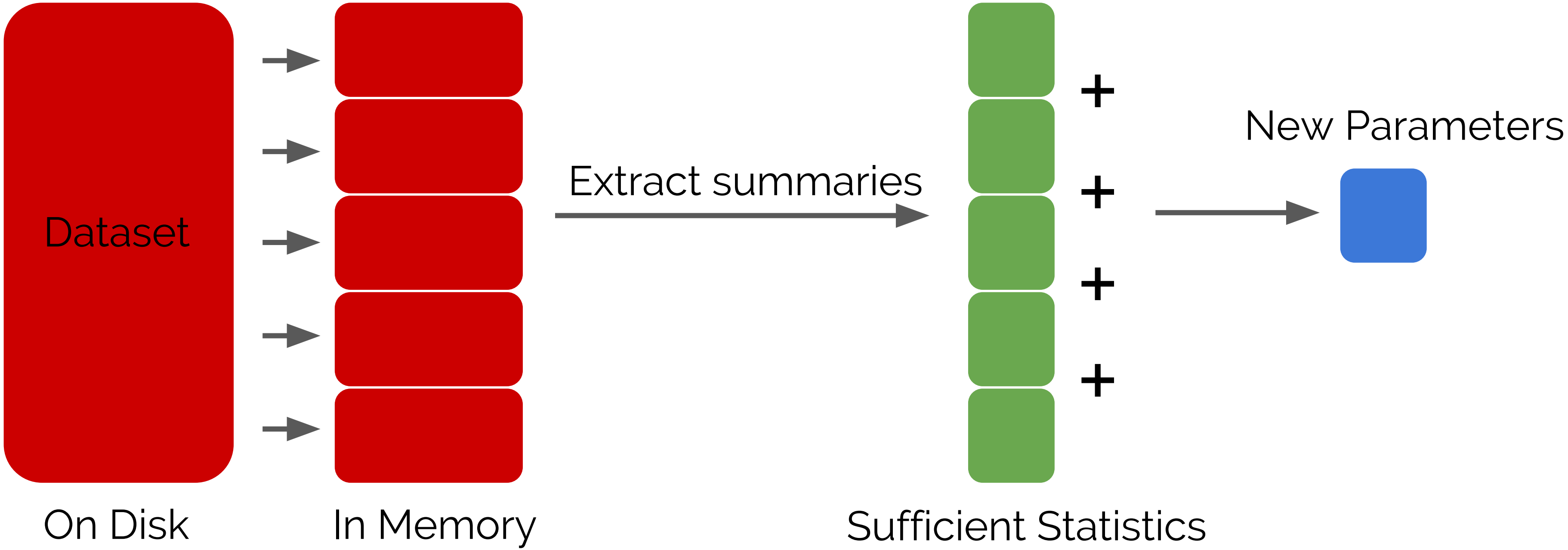
pomegranate supports mini-batching

Instead of going through the full dataset before updating parameters, one could update parameters at each step.



pomegranate supports parallelization

Multiple batches can be loaded at the same time and processed by different threads using `n_jobs` in either fitting or prediction methods



Dataset

Extract summaries

+ + + +

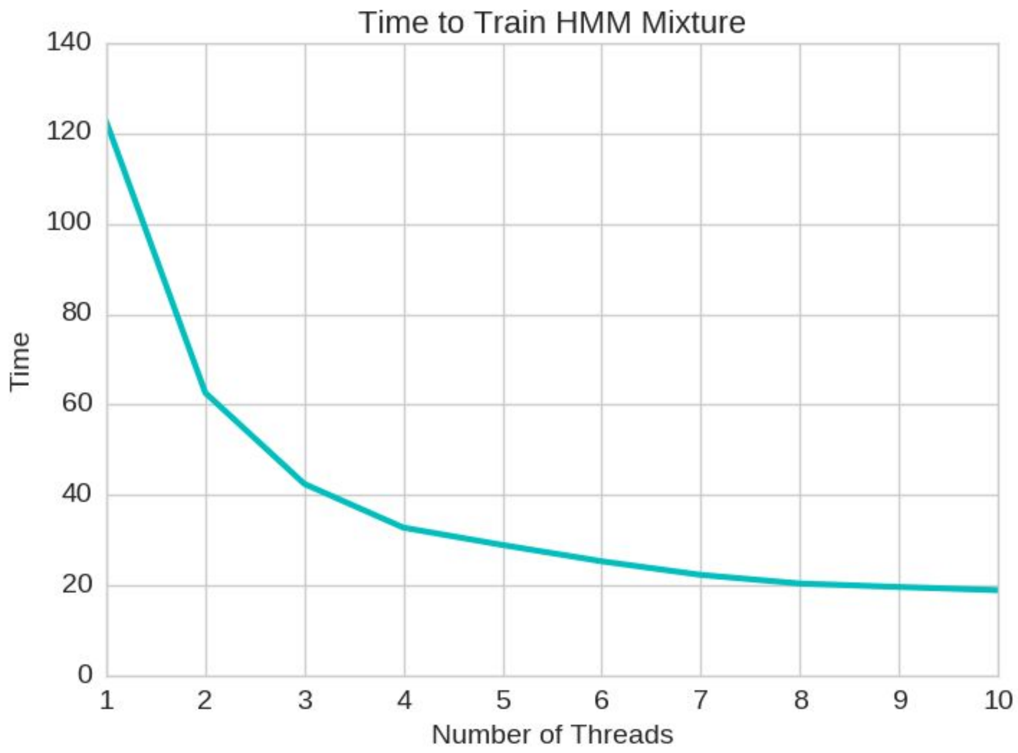
New Parameters

On Disk

In Memory

Sufficient Statistics

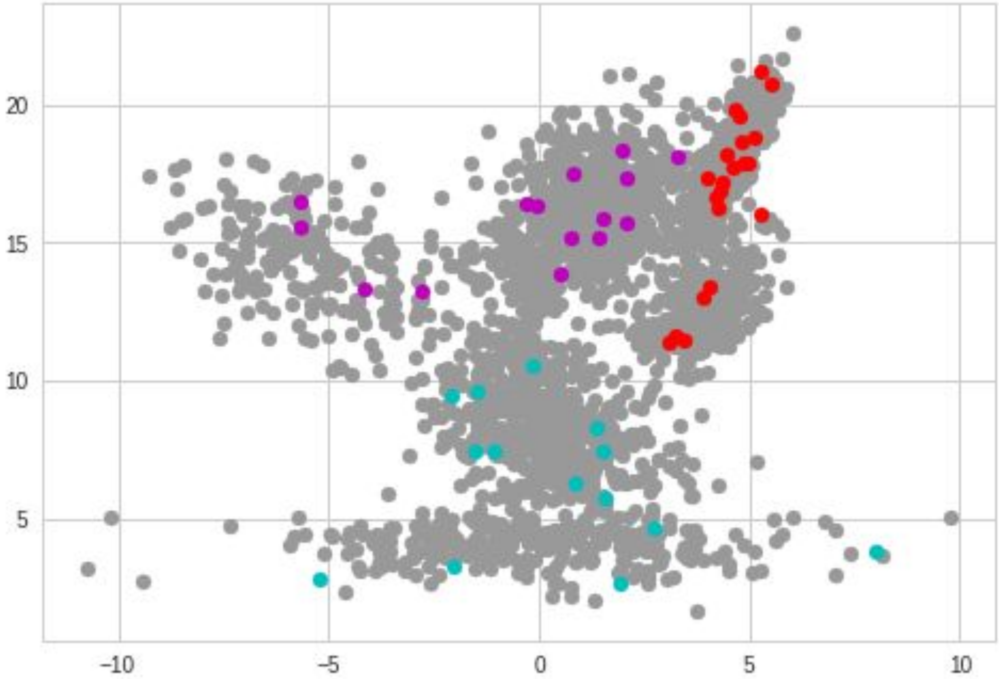
Training a mixture of HMMs in parallel



```
model.fit(X, n_jobs=n)
```

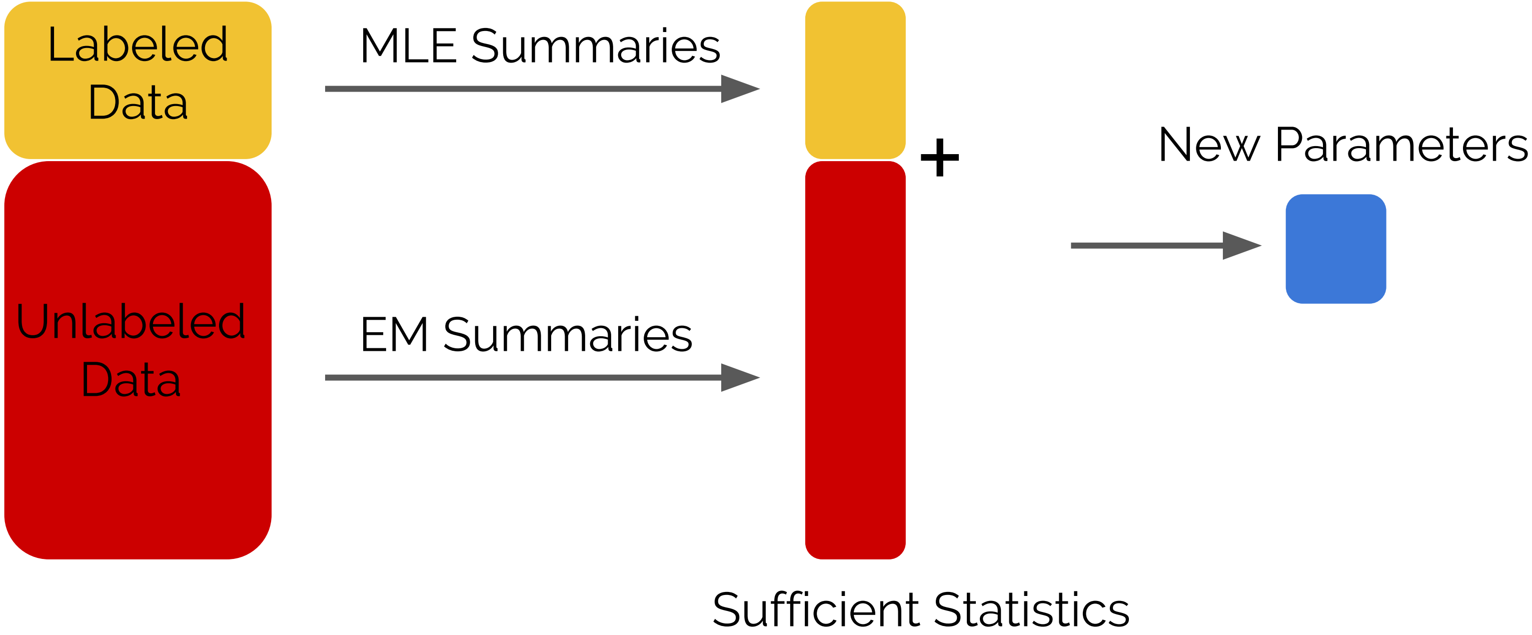
pomegranate allows semisupervised learning

For many tasks, there is limited labeled data but a deluge of unlabeled data, and one wants to utilize both.



pomegranate allows semisupervised learning

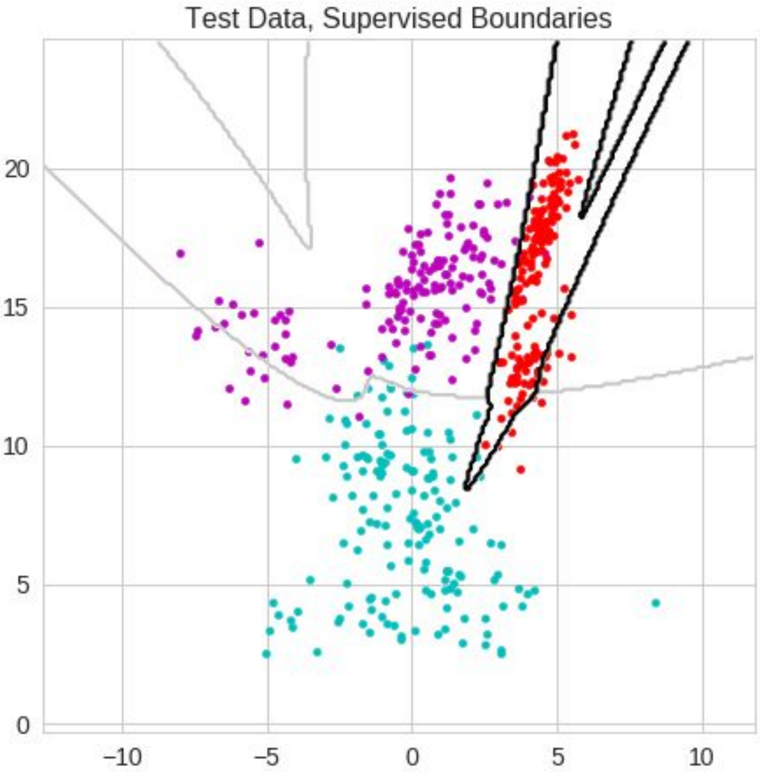
Summaries from MLE on the labeled data can be added to summaries from EM on the unlabeled data

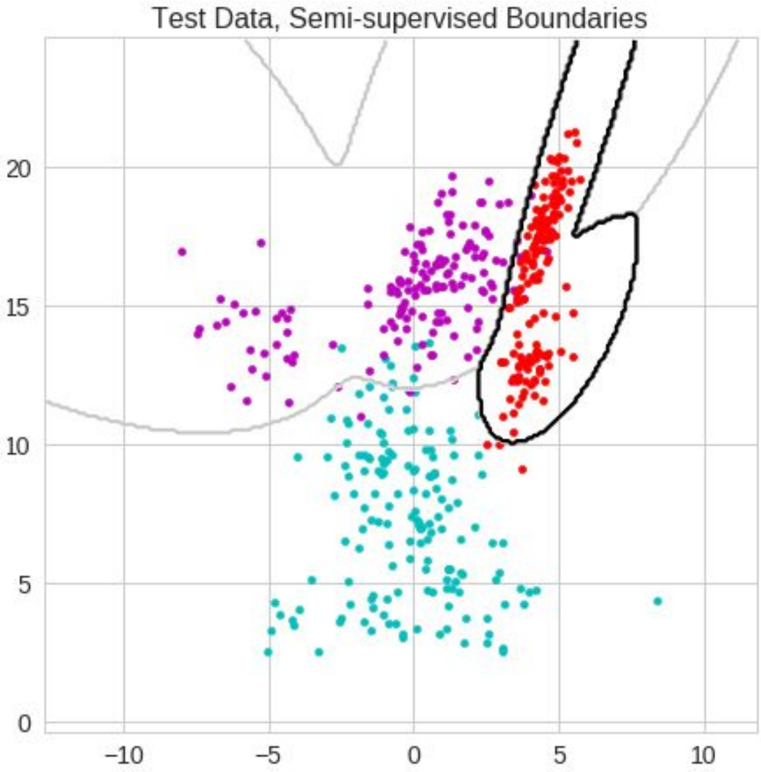


pomegranate allows semisupervised learning

Supervised Accuracy: 0.93

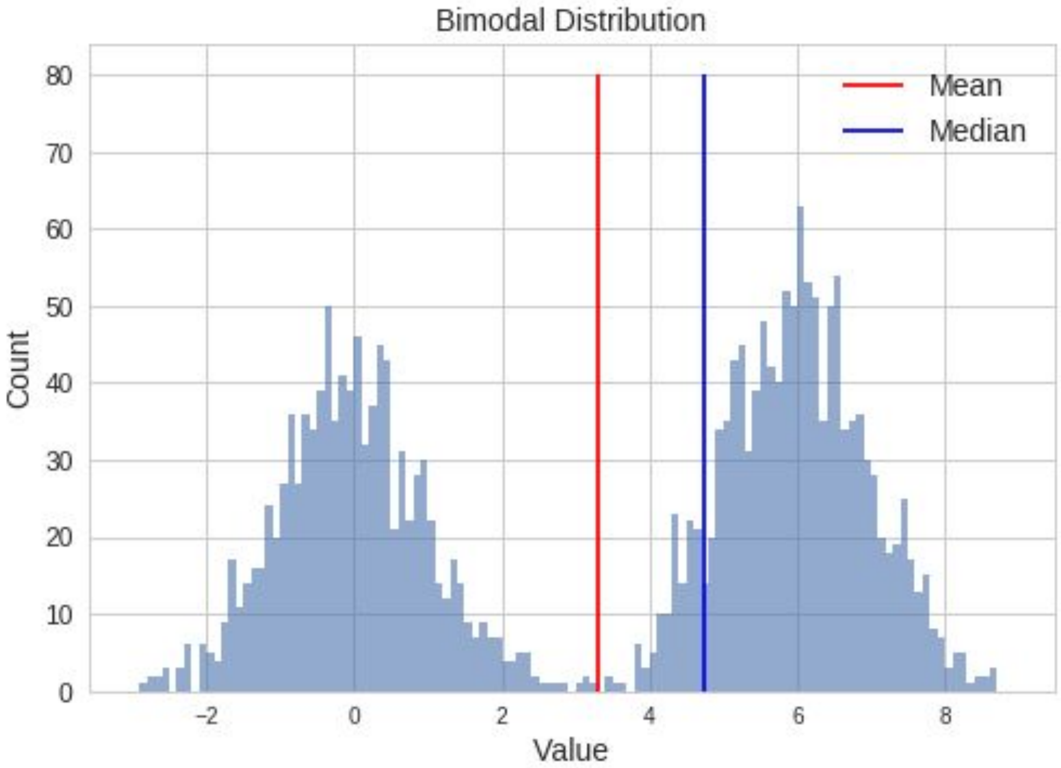
Semisupervised Accuracy: 0.96





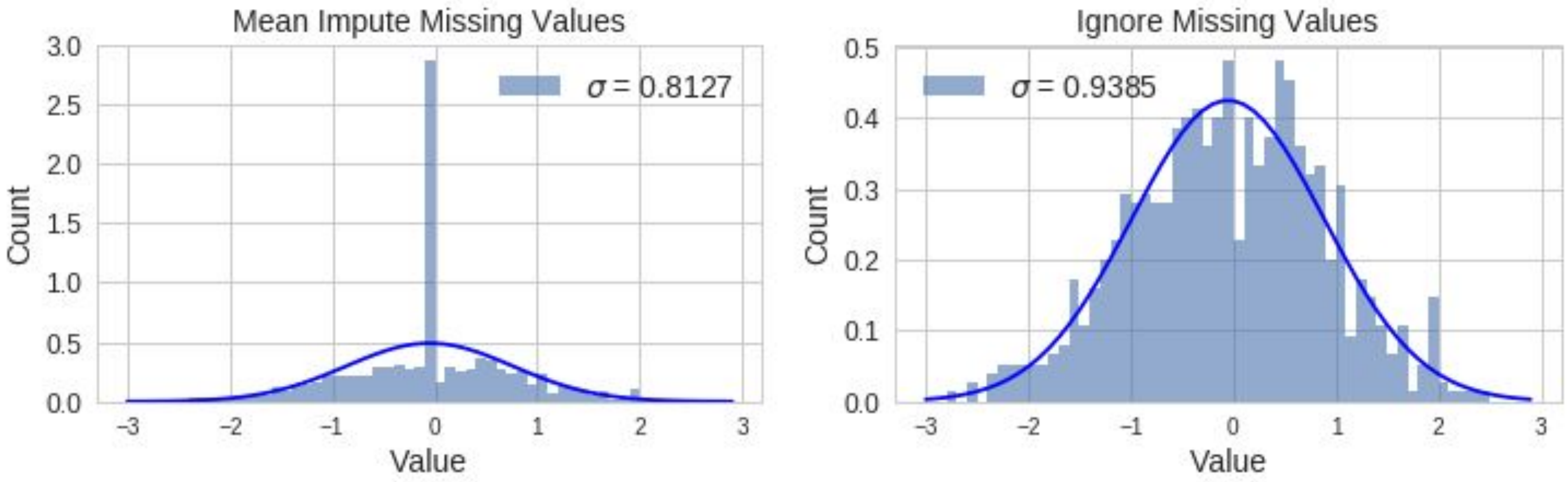
pomegranate supports missing data

Many real world tasks involve missing data, but common approaches aren't sufficient for tackling the problem.



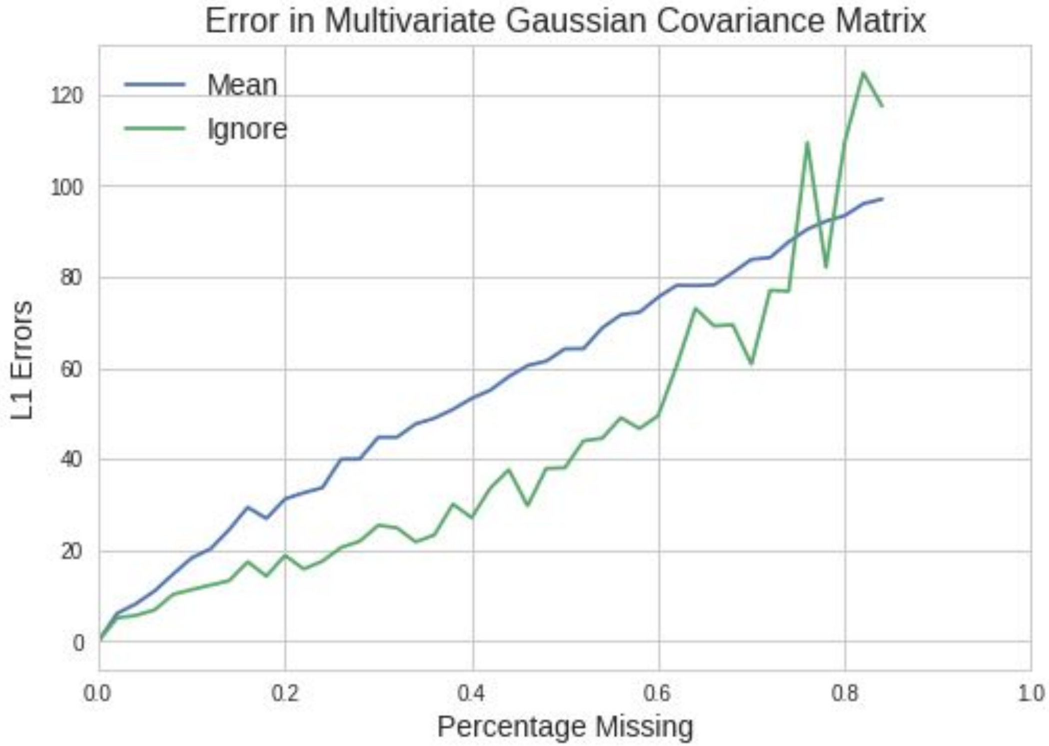
pomegranate supports missing data

Many real world tasks involve missing data, but common approaches aren't sufficient for tackling the problem.



pomegranate supports missing data

Many real world tasks involve missing data, but common approaches aren't sufficient for tackling the problem.



pomegranate supports missing data

Pomegranate supports **model fitting**, **structure learning**, and **model predictions** on data sets that include missing values, no matter how complicated the model or sparse the data set.

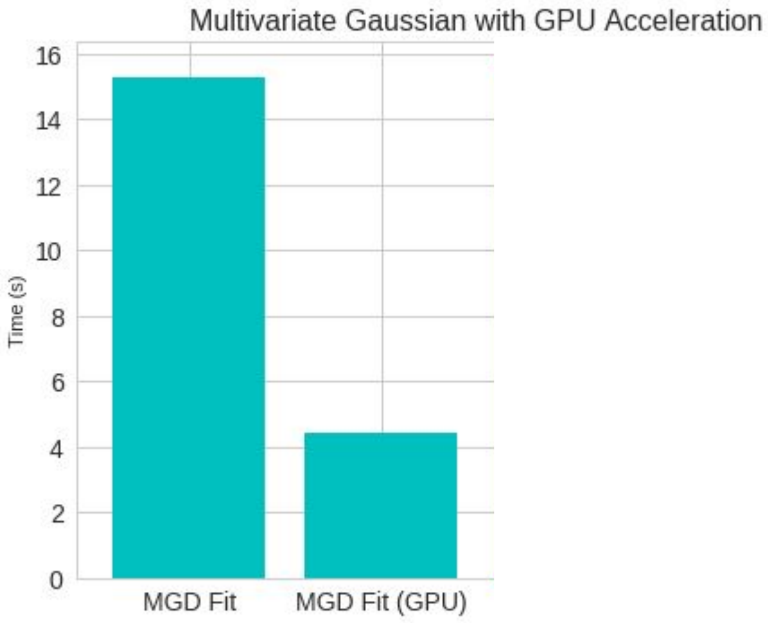
You can **fit a Gaussian mixture model** to incomplete data sets.

You can run the **Viterbi or forward-backward algorithm** using a HMM on incomplete data sets.

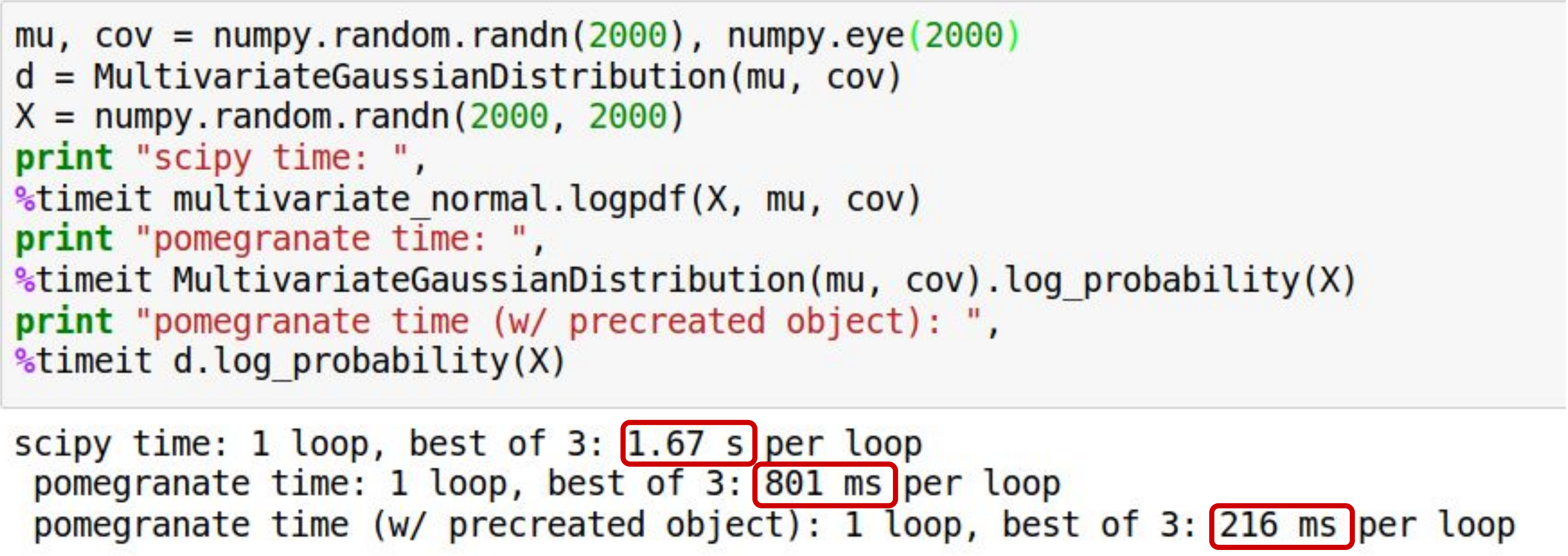
You can **learn the structure of a Bayesian network** on incomplete data sets.

All without having to change your command, simply by including np.nan in the place of the missing value

pomegranate uses Cupy for GPU support



pomegranate can be faster than scipy



```
mu, cov = numpy.random.randn(2000), numpy.eye(2000)
d = MultivariateGaussianDistribution(mu, cov)
X = numpy.random.randn(2000, 2000)
print "scipy time: ",
%timeit multivariate_normal.logpdf(X, mu, cov)
print "pomegranate time: ",
%timeit MultivariateGaussianDistribution(mu, cov).log_probability(X)
print "pomegranate time (w/ precreated object): ",
%timeit d.log_probability(X)
```

```
scipy time: 1 loop, best of 3: 1.67 s per loop
 pomegranate time: 1 loop, best of 3: 801 ms per loop
 pomegranate time (w/ precreated object): 1 loop, best of 3: 216 ms per loop
```

pomegranate uses aggressive caching

$$P(X|\mu,\sigma) = \frac{1}{\sqrt{2\pi\sigma}} exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

$$logP(X|\mu,\sigma) = -\log(\sqrt{2\pi\sigma}) - \frac{(x-\mu)^2}{2\sigma^2}$$

$$logP(X|\mu,\sigma) = \alpha - \frac{(x-\mu)^2}{\beta}$$

GOSSIP GIRL

Example ‘blast’ from Gossip Girl

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.

Example ‘blast’ from Gossip Girl

Why'd she leave? Why'd she return? Send me all the deets. And who am I? That's the secret I'll never tell. The only one. —XOXO. Gossip Girl.

How do we encode these ‘blasts’?

Better lock it down with Nate, B. Clock's ticking.

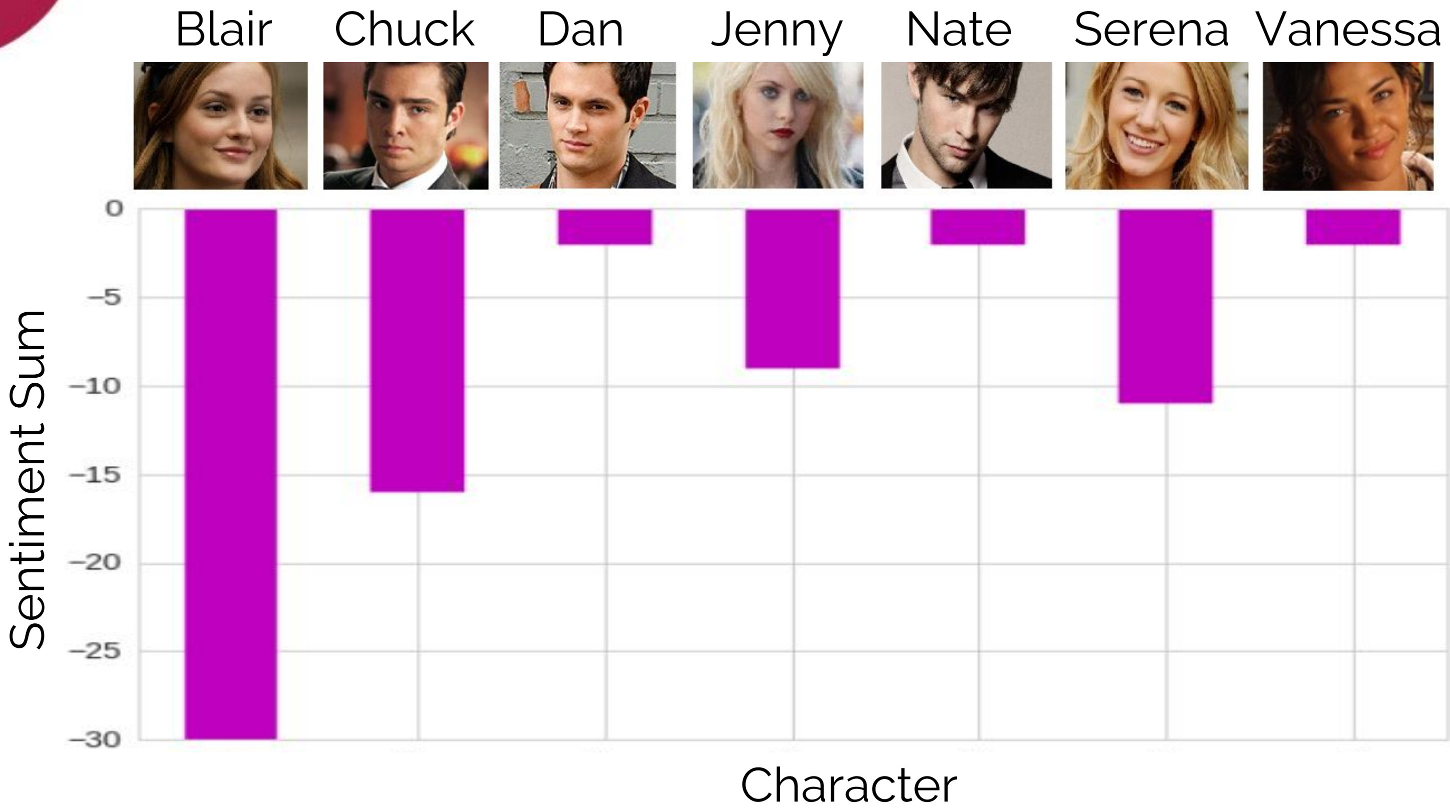
+1 Nate
-1 Blair

How do we encode these ‘blasts’?

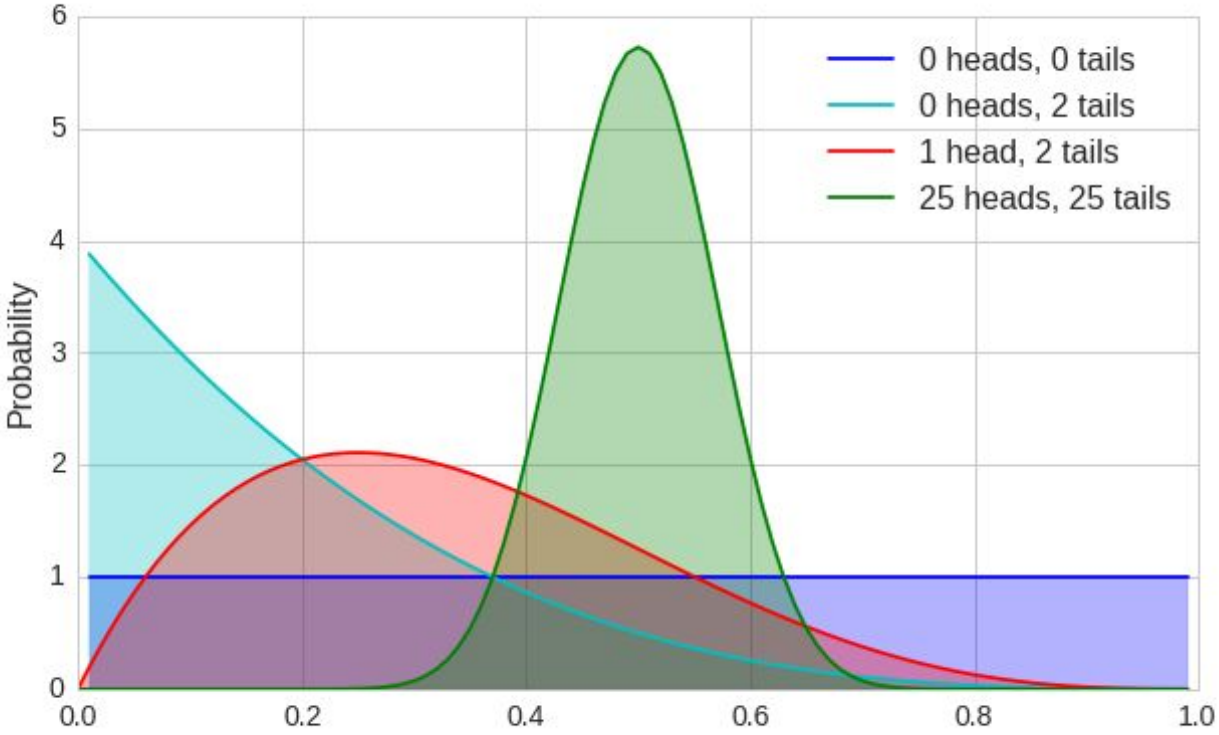
This just in: S and B committing a crime of fashion. Who doesn't love a five-finger discount. Especially if it's the middle one.

-1 Blair
-1 Serena

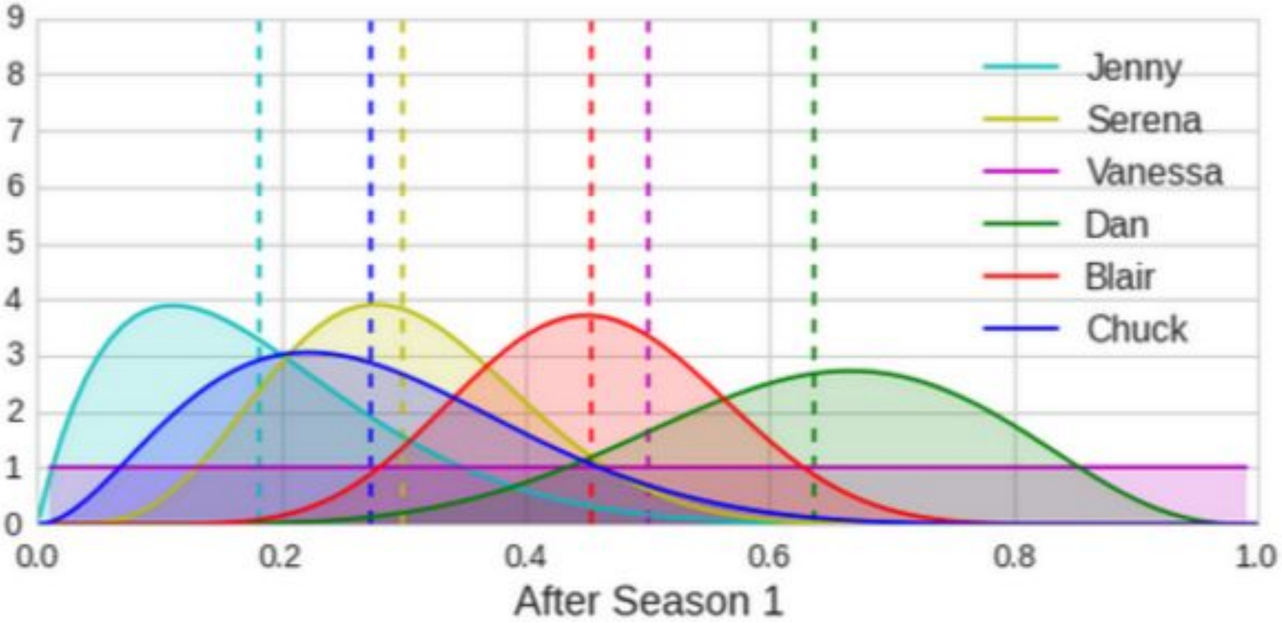
Simple summations don't work well



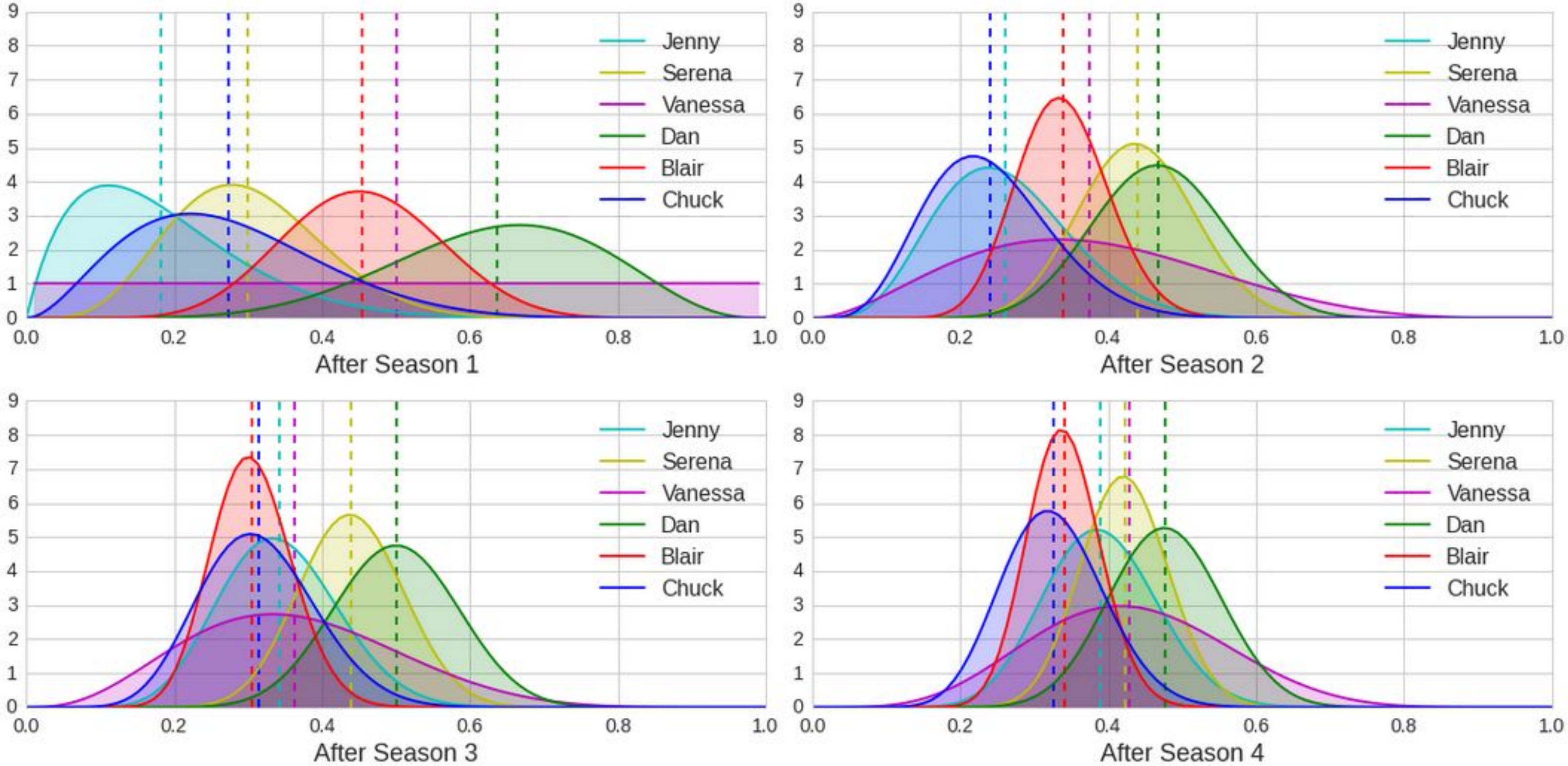
Beta distributions can model uncertainty



Beta distributions can model uncertainty

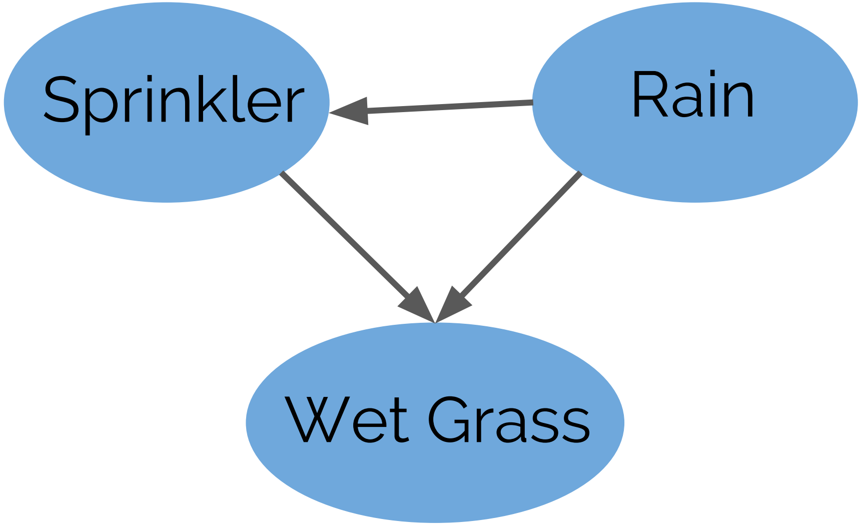


Beta distributions can model uncertainty



Bayesian networks

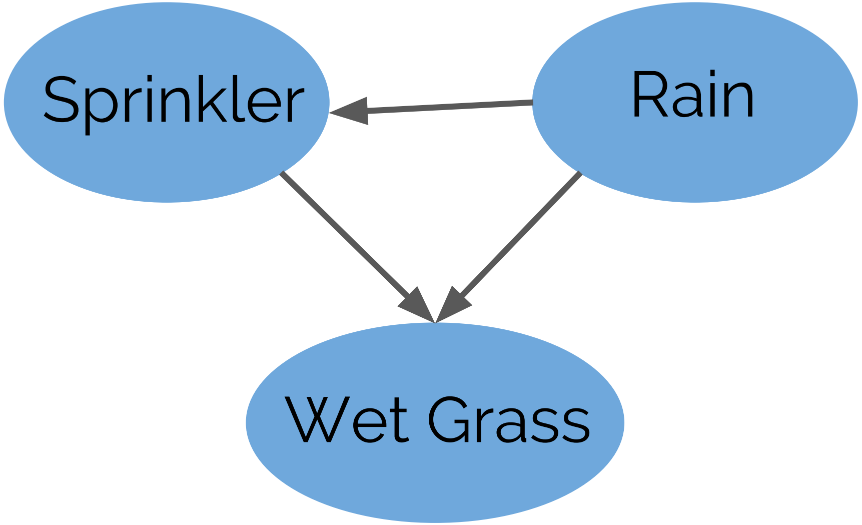
Bayesian networks are powerful inference tools which define a dependency structure between variables.



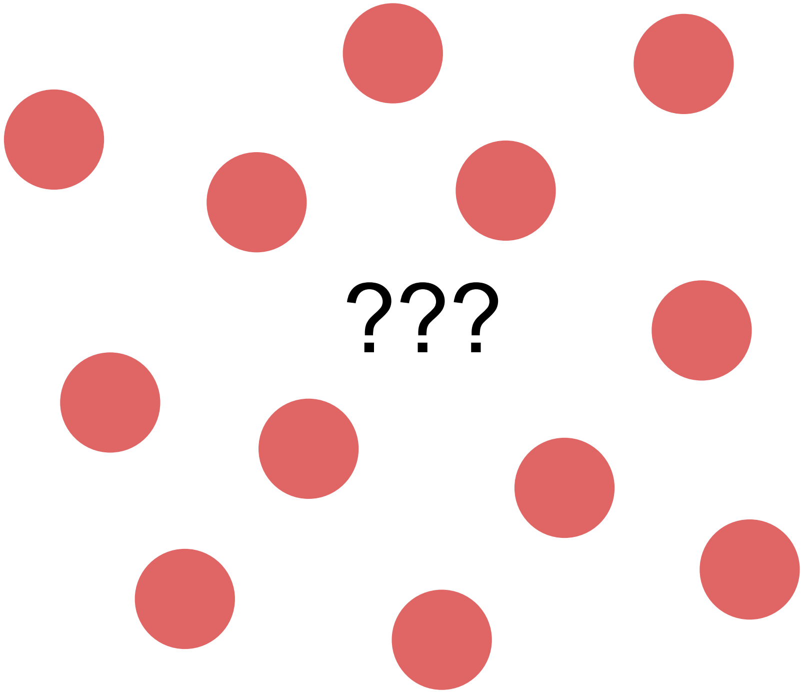
Bayesian networks

Two main difficult tasks:

1. Inference given incomplete information
2. Learning the dependency structure from data



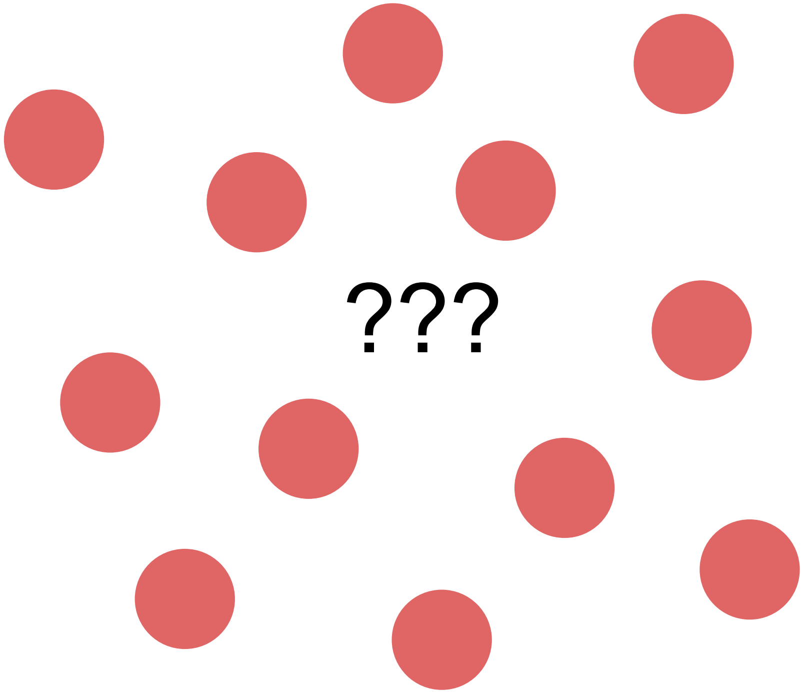
Bayesian network structure learning



Three primary ways:

- “Search and score” / Exact
- “Constraint Learning” / PC
- Heuristics

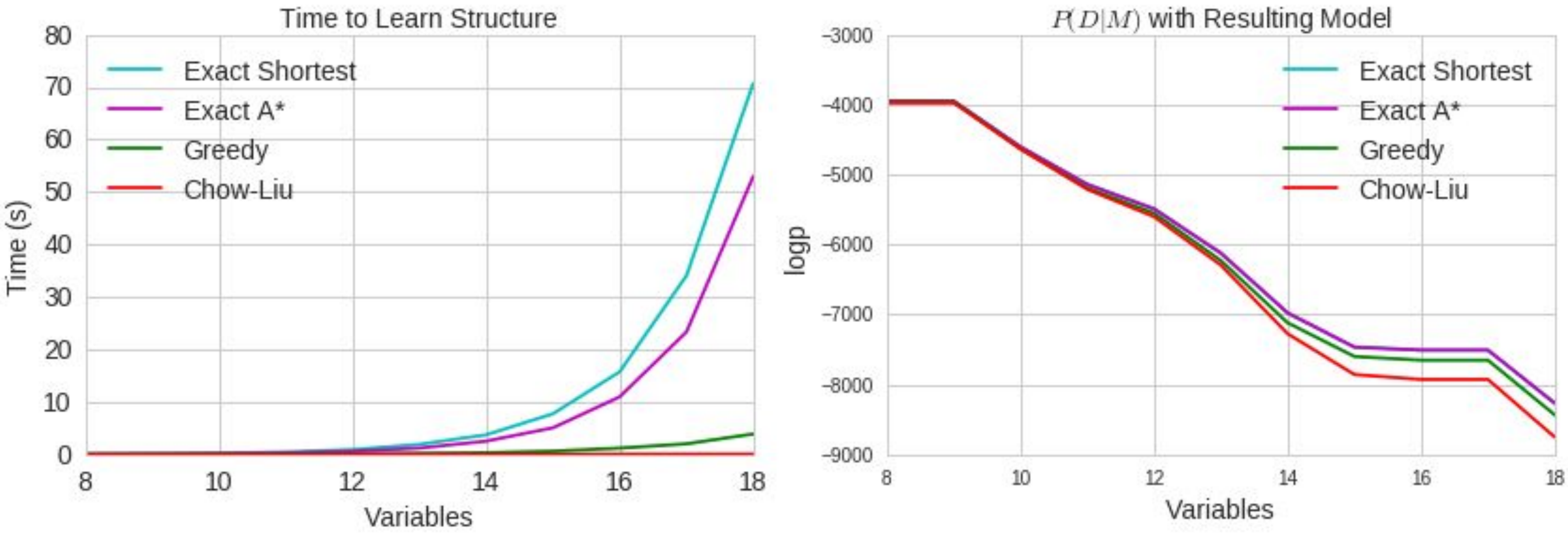
Bayesian network structure learning



pomegranate supports:

- “Search and score” / Exact
- “Constraint Learning” / PC
- Heuristics

pomegranate supports four algorithms



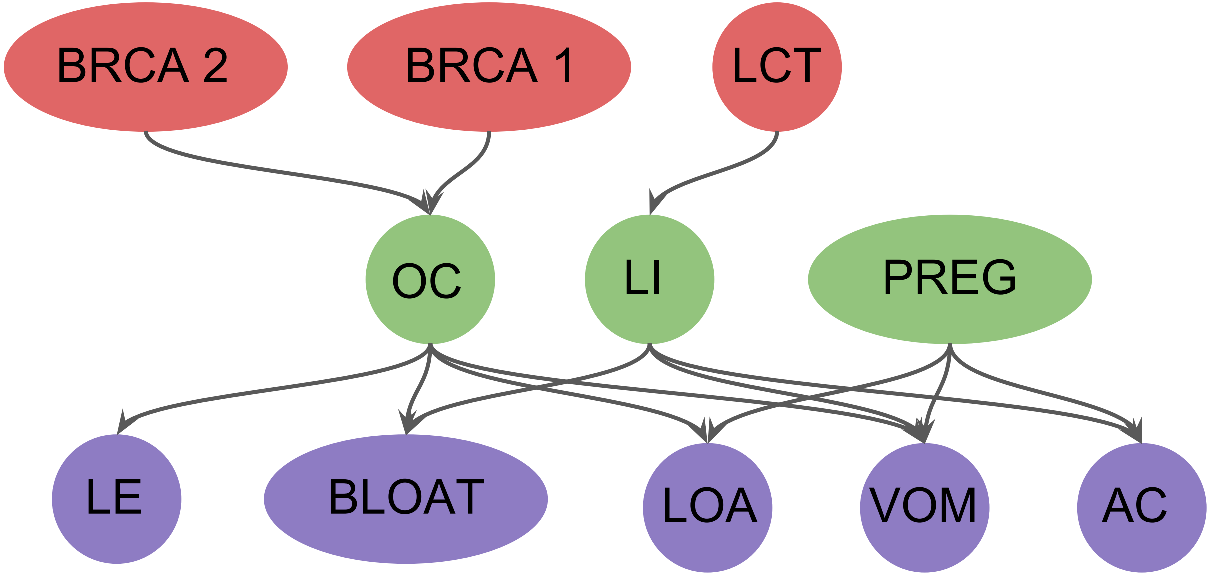
BNSL is hard due to acyclicity requirement

Easy! Tractable!

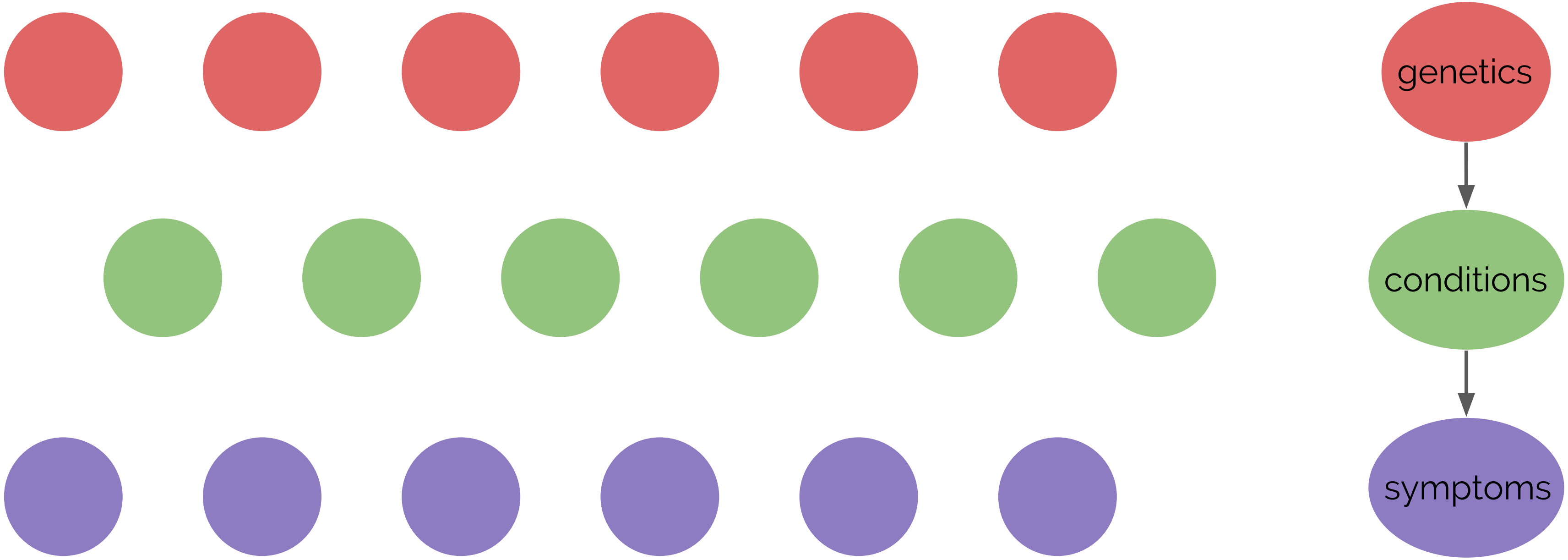
Global Parameter Independence: The parents of some variable A are independent of the parents of some variable B given that they don’t form a cycle in the resulting graph

Hard! Exponential Time!

Medical diagnosis Bayesian network



Constraint graphs merge data + knowledge

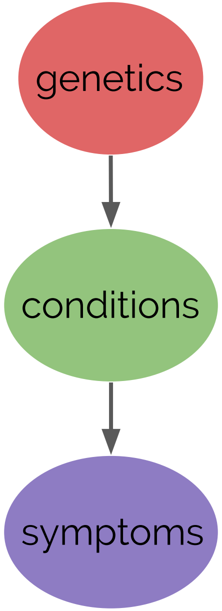


Variables to be Modeled

Constraint Graph

Constraint graphs merge data + knowledge

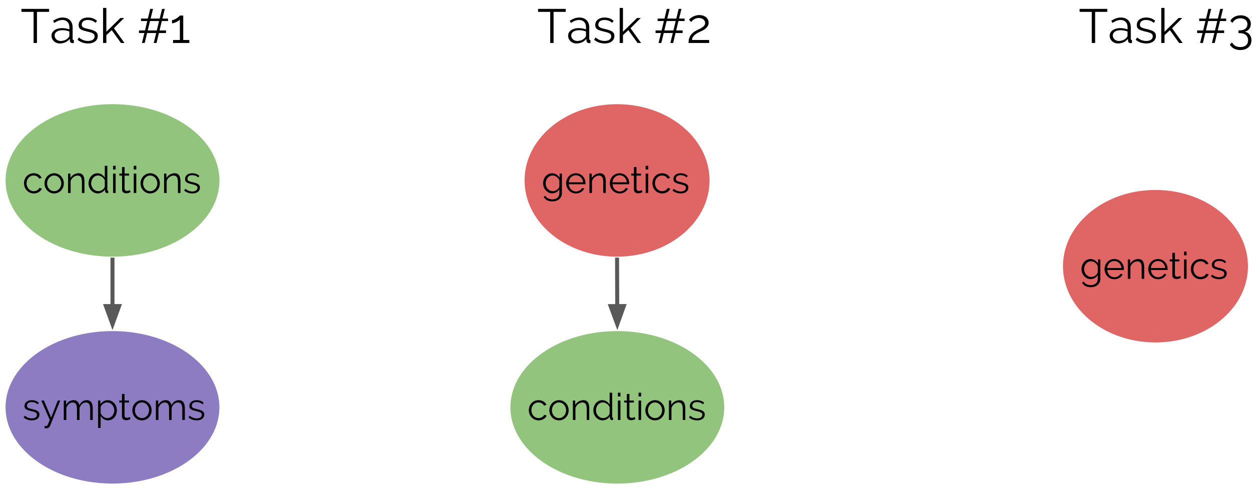
Constraint Graph



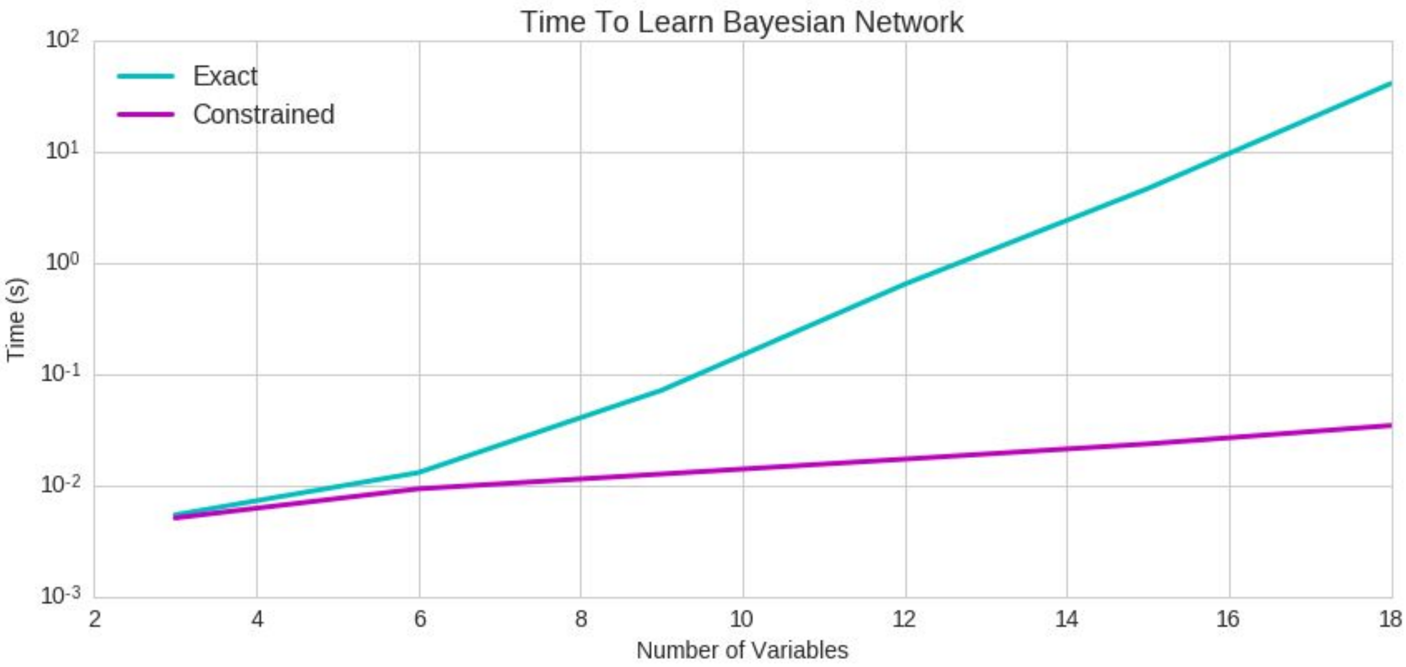
Global Parameter Independence: The parents of some variable A are independent of the parents of some variable B given that they don't form a cycle in the resulting graph

Constraint graphs merge data + knowledge

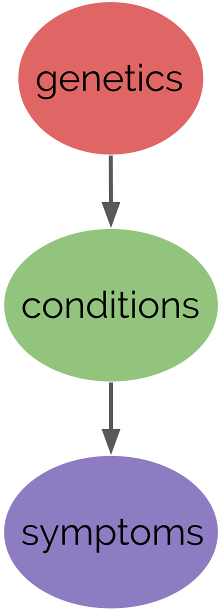
The parents of some variable A are independent of the parents of some variable B



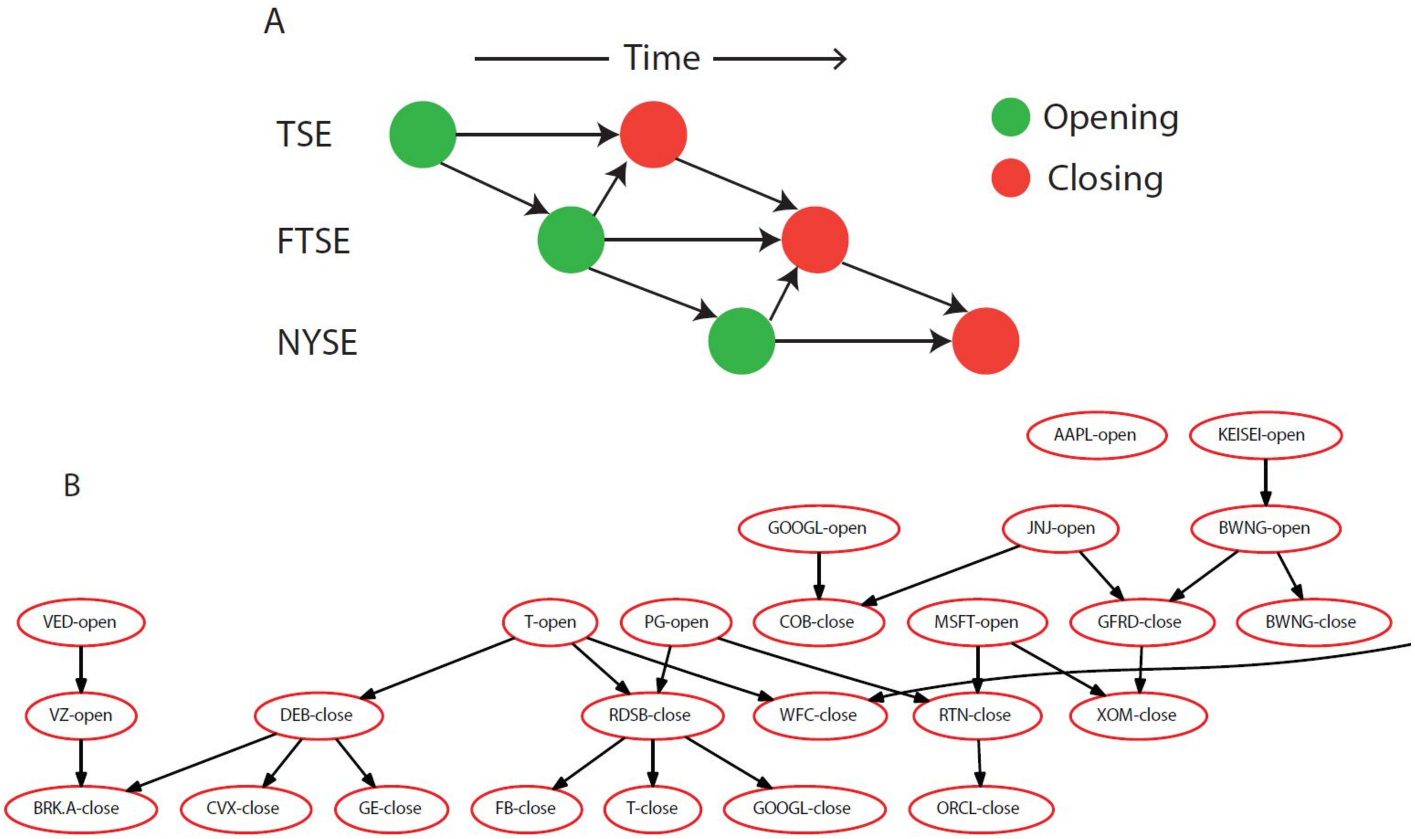
Constraint graphs merge data + knowledge



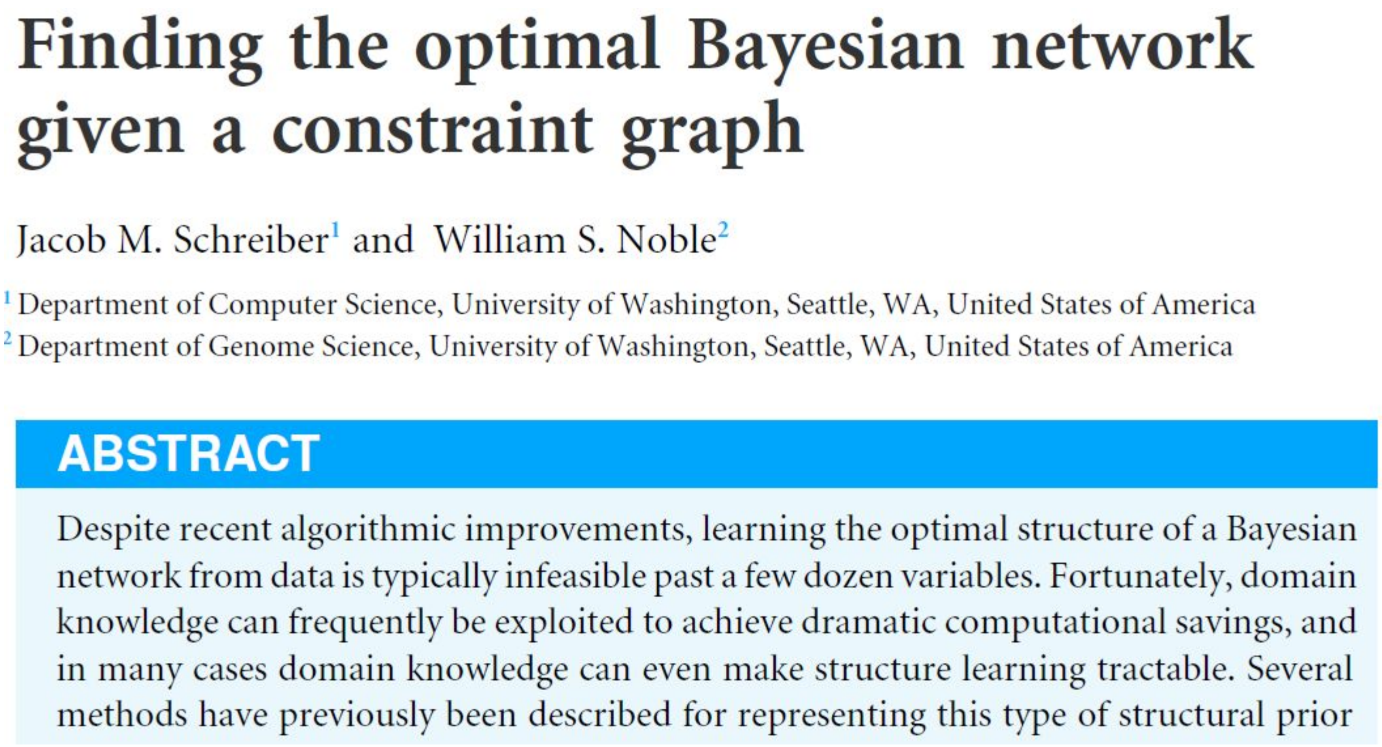
Constraint Graph



Modeling the global stock market



Constraint graph published in PeerJ CS



Finding the optimal Bayesian network given a constraint graph

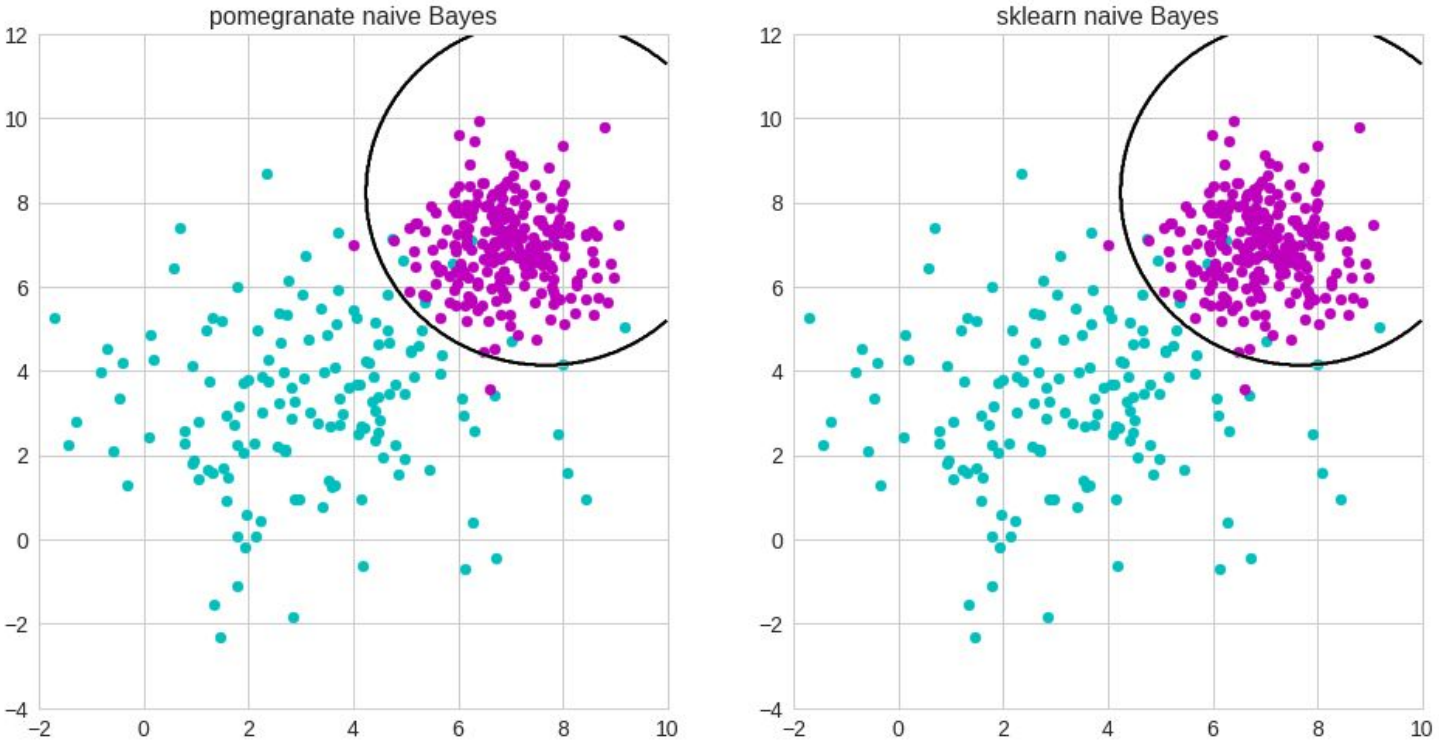
Jacob M. Schreiber[1] and William S. Noble[2]

[1] Department of Computer Science, University of Washington, Seattle, WA, United States of America
[2] Department of Genome Science, University of Washington, Seattle, WA, United States of America

ABSTRACT

Despite recent algorithmic improvements, learning the optimal structure of a Bayesian network from data is typically infeasible past a few dozen variables. Fortunately, domain knowledge can frequently be exploited to achieve dramatic computational savings, and in many cases domain knowledge can even make structure learning tractable. Several methods have previously been described for representing this type of structural prior

Naive Bayes produces ellipsoid boundaries



```
model = NaiveBayes.from_samples(NormalDistribution, X, y)
```

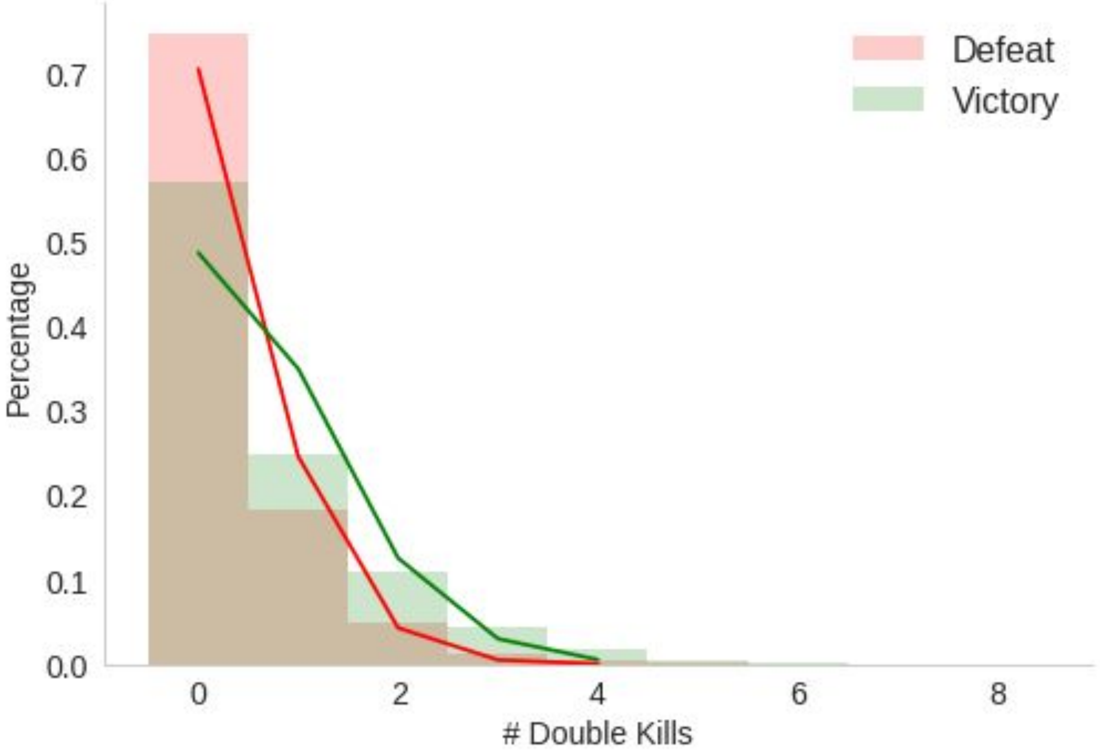
Naive Bayes assumes independent features

$$Posterior = \frac{Likelihood * Prior}{Normalization}$$

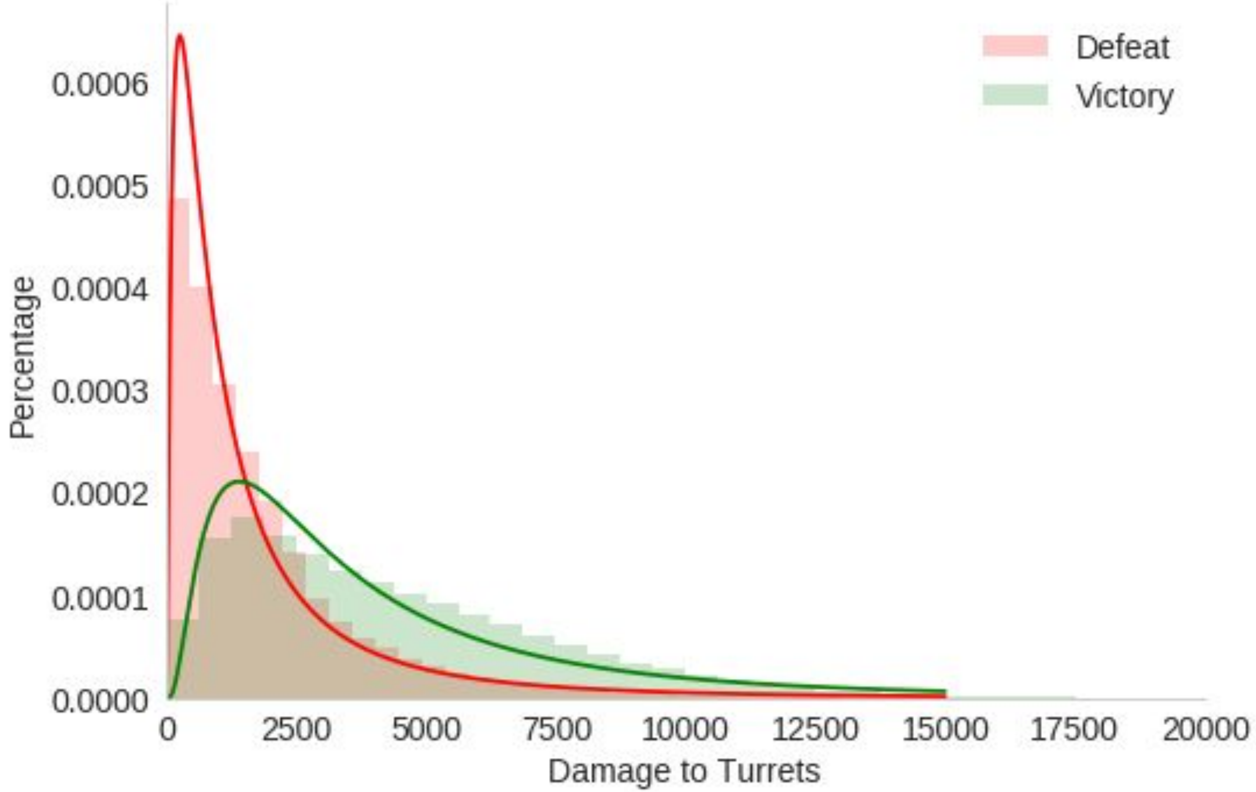
$$P(M|D) = \frac{\prod_{i=1}^{d} P(D_i|M)P(M)}{\sum_{M} \prod_{i=1}^{d} P(D_i|M)P(M)}$$

LEAGUE OF LEGENDS®

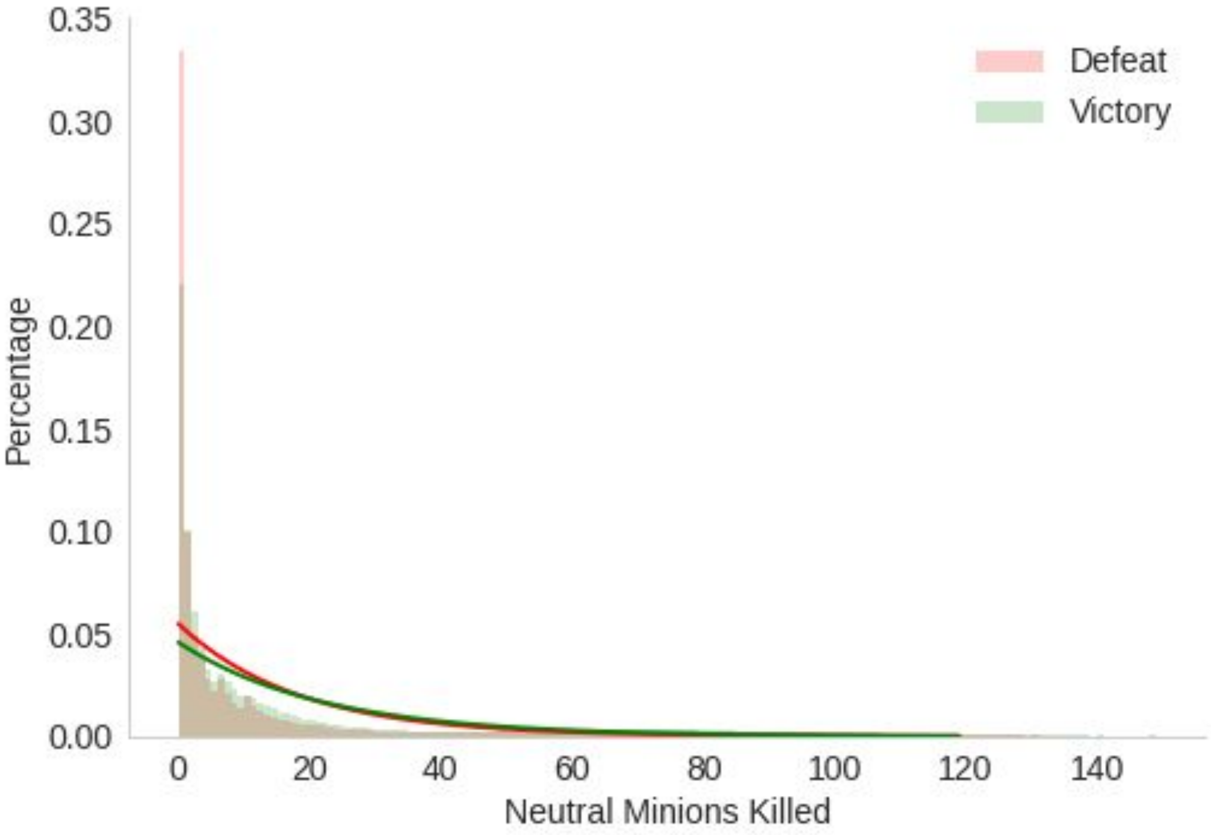
Data can fall under different distributions



Data can fall under different distributions



Data can fall under different distributions

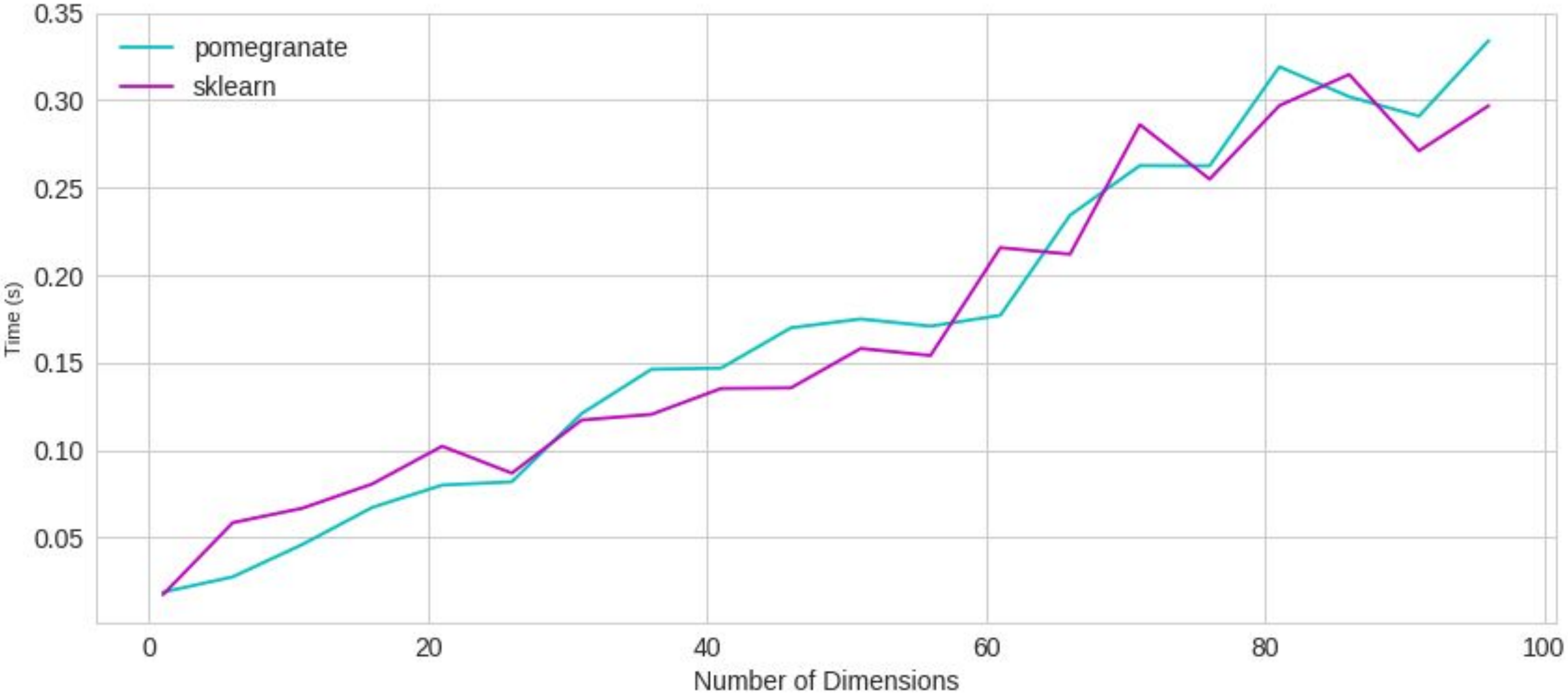


Using appropriate distributions is best

```
dists = [LogNormalDistribution, PoissonDistribution,
ExponentialDistribution, PoissonDistribution]

model1 = NaiveBayes.from_samples(NormalDistribution, X, y)
model2 = NaiveBayes.from_samples(dists, X, y)
model3 = GaussianNB().fit(X, y)
```

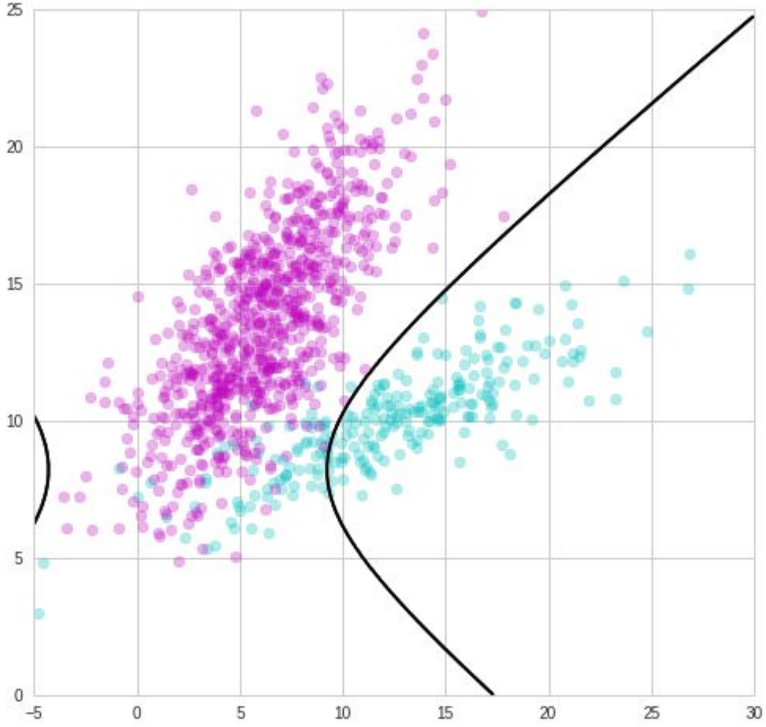
pomegranate Gaussian Naive Bayes: 0.711
sklearn Gaussian Naive Bayes: 0.711
Heterogeneous Naive Bayes: 0.726

This additional flexibility is just as fast

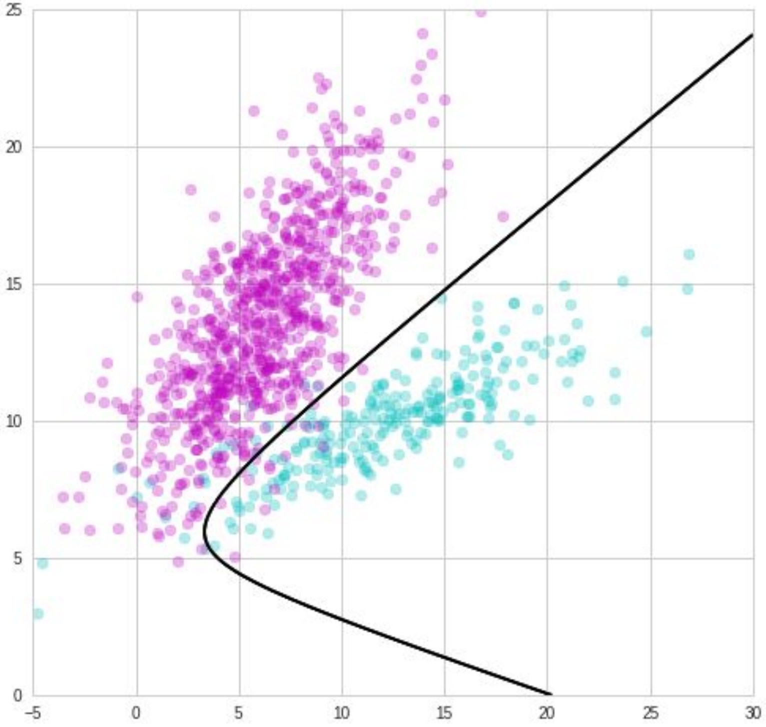


Bayes classifiers don’t require independence

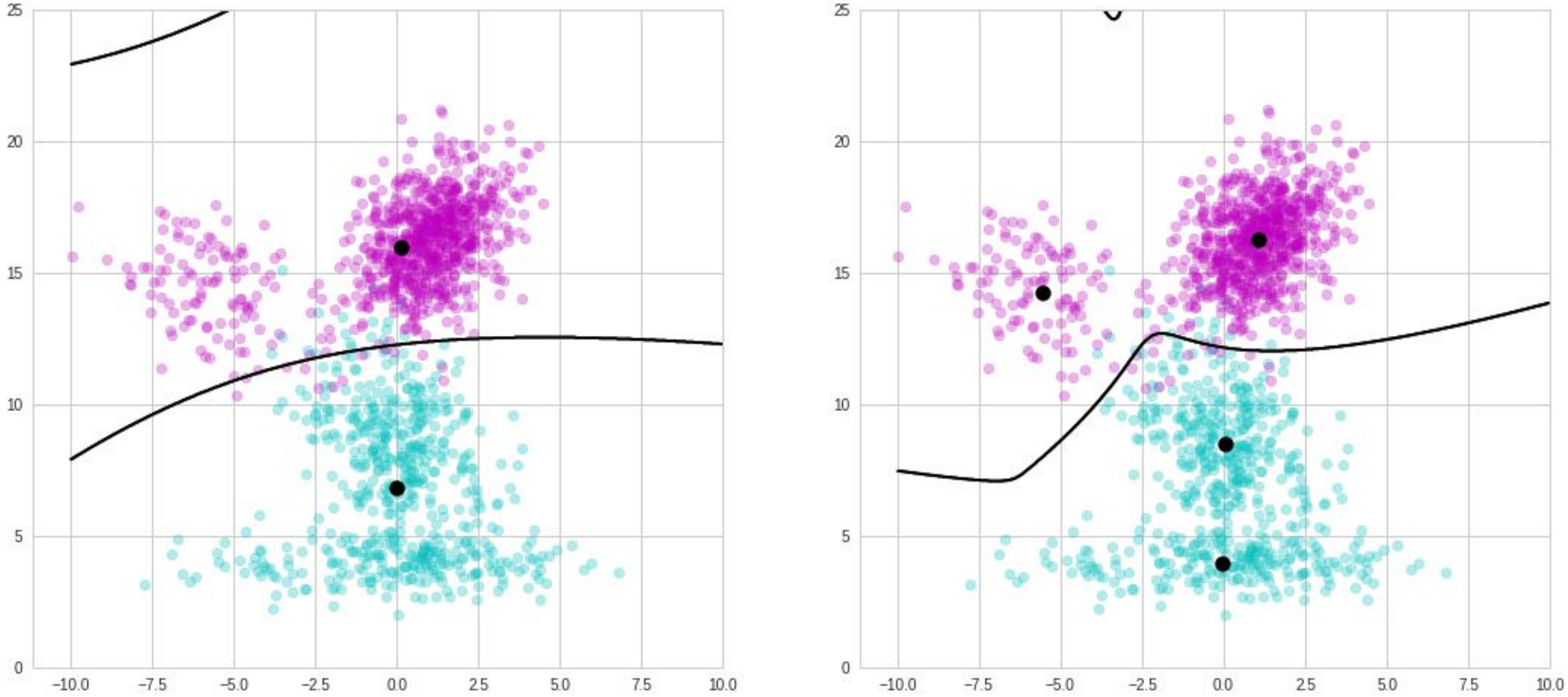
naive accuracy: 0.929



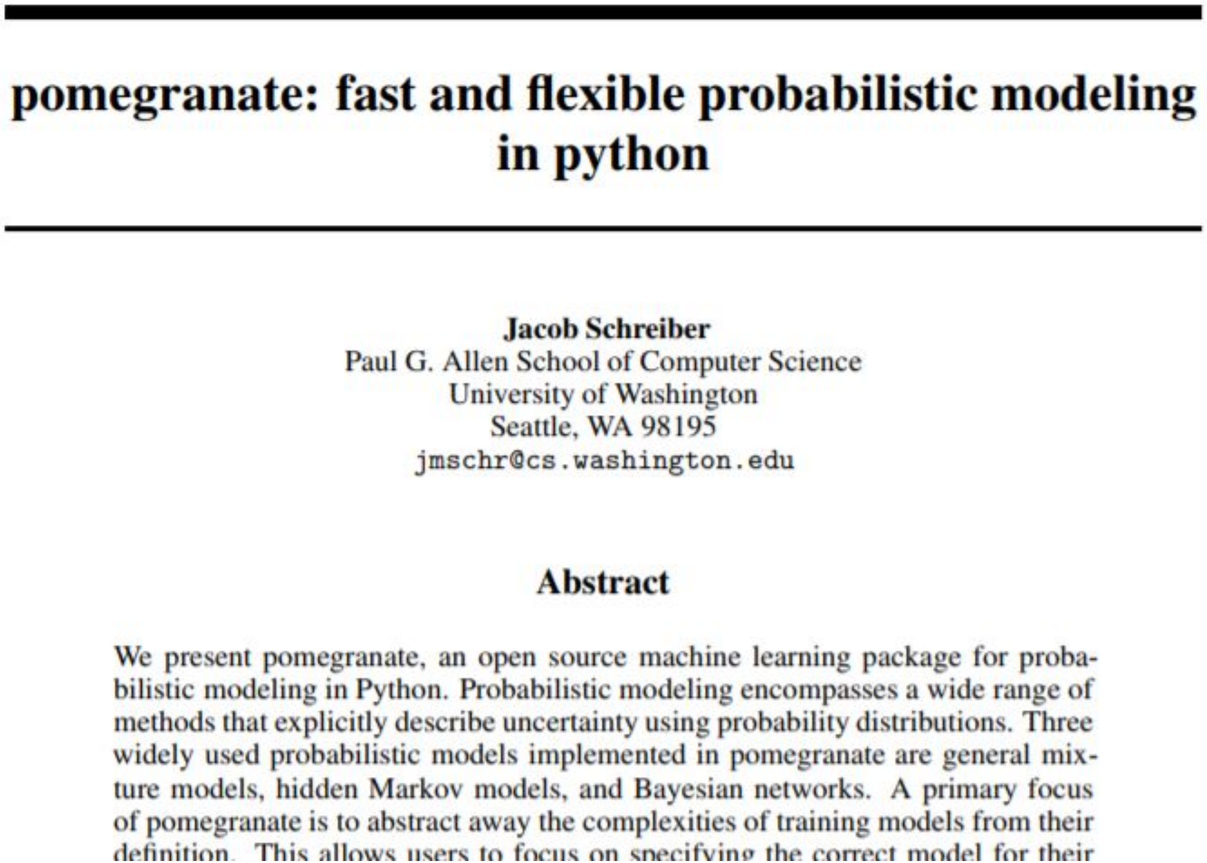
bayes classifier accuracy: 0.966



Gaussian mixture model Bayes classifier



pomegranate paper at JMLR-MLOSS



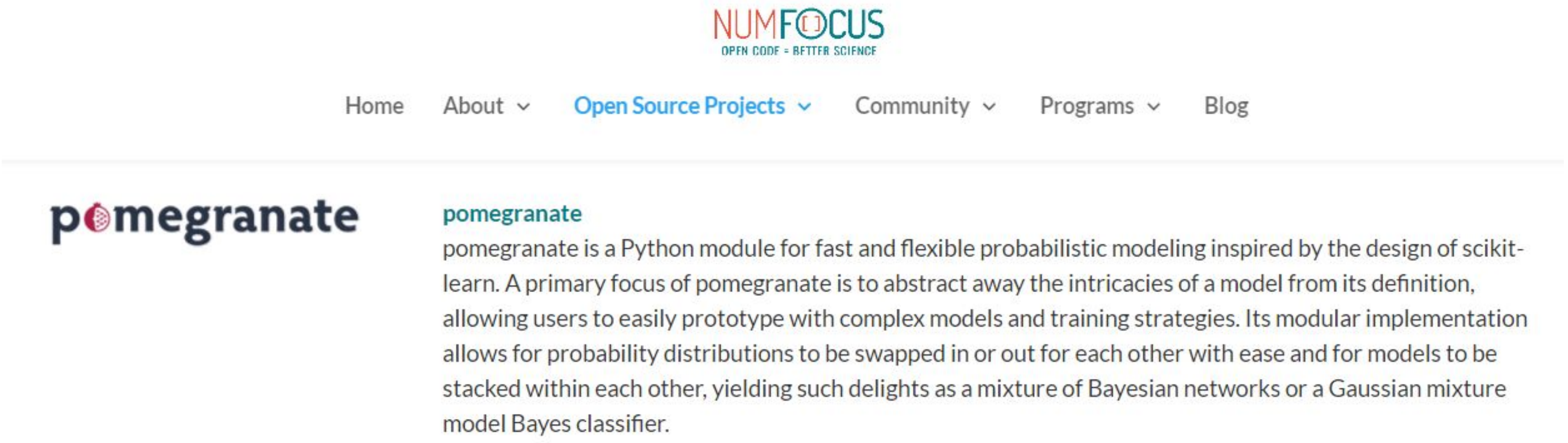
pomegranate: fast and flexible probabilistic modeling in python

Jacob Schreiber
Paul G. Allen School of Computer Science
University of Washington
Seattle, WA 98195
jmschr@cs.washington.edu

Abstract

We present pomegranate, an open source machine learning package for probabilistic modeling in Python. Probabilistic modeling encompasses a wide range of methods that explicitly describe uncertainty using probability distributions. Three widely used probabilistic models implemented in pomegranate are general mixture models, hidden Markov models, and Bayesian networks. A primary focus of pomegranate is to abstract away the complexities of training models from their definition. This allows users to focus on specifying the correct model for their

pomegranate is NumFOCUS affiliated

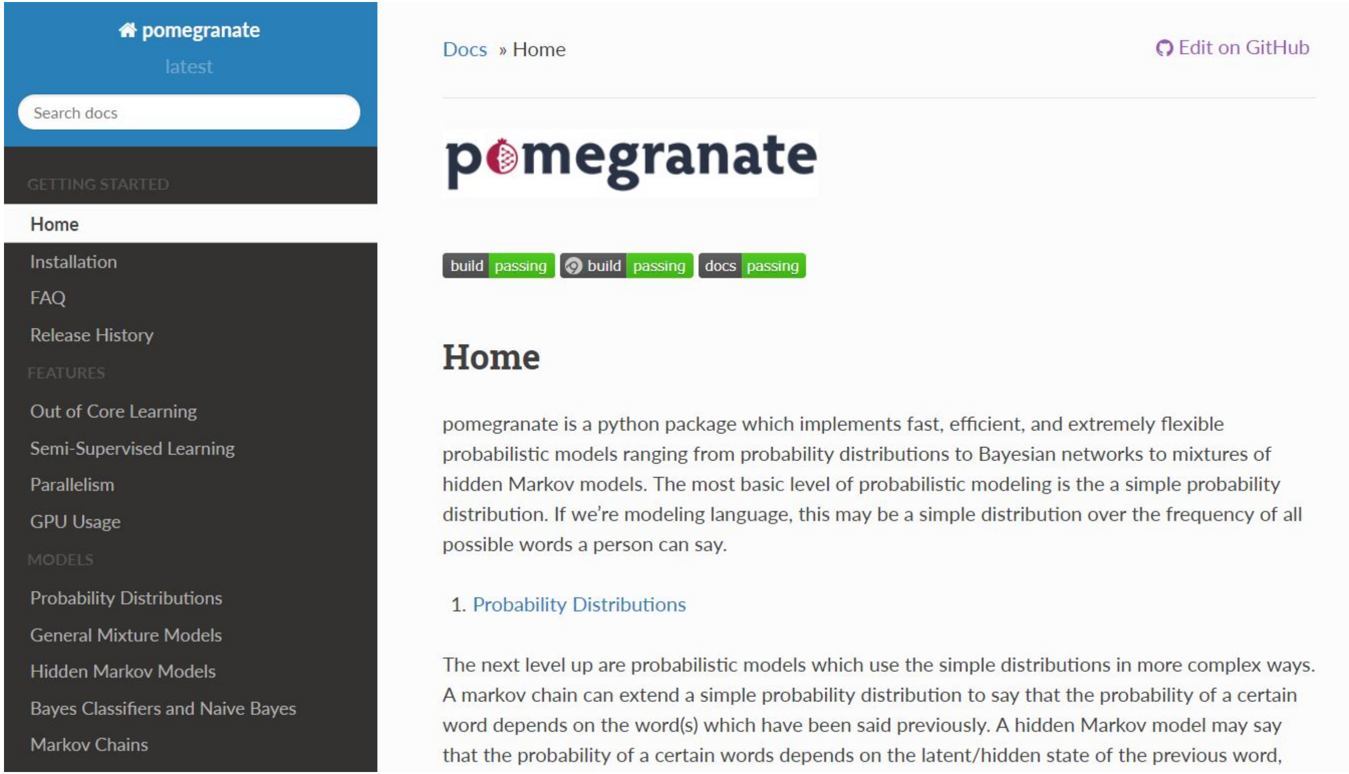


pomegranate

pomegranate is a Python module for fast and flexible probabilistic modeling inspired by the design of scikit-learn. A primary focus of pomegranate is to abstract away the intricacies of a model from its definition, allowing users to easily prototype with complex models and training strategies. Its modular implementation allows for probability distributions to be swapped in or out for each other with ease and for models to be stacked within each other, yielding such delights as a mixture of Bayesian networks or a Gaussian mixture model Bayes classifier.

https://www.numfocus.org/open-source-projects/affiliated-projects/

Documentation available at Readthedocs



Home

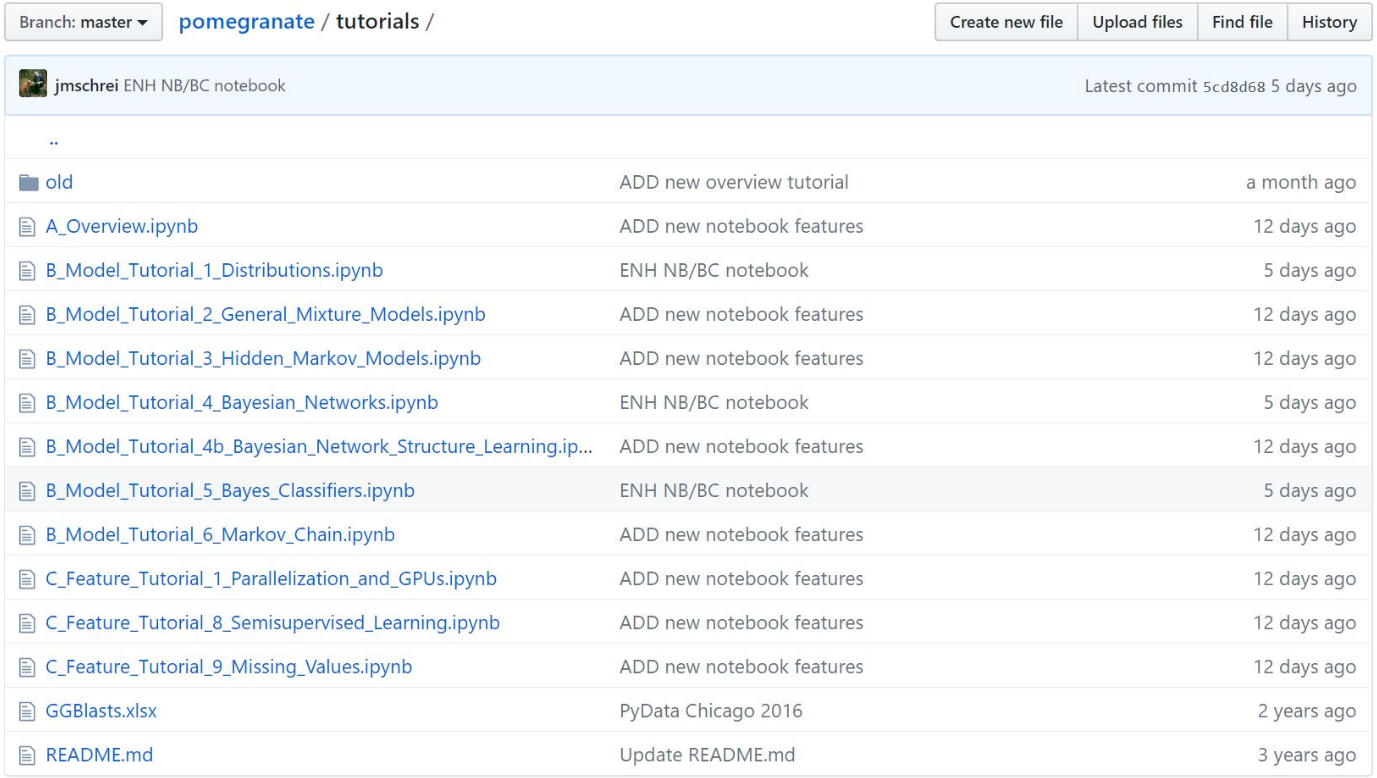
pomegranate is a python package which implements fast, efficient, and extremely flexible probabilistic models ranging from probability distributions to Bayesian networks to mixtures of hidden Markov models. The most basic level of probabilistic modeling is the a simple probability distribution. If we're modeling language, this may be a simple distribution over the frequency of all possible words a person can say.

1. Probability Distributions

The next level up are probabilistic models which use the simple distributions in more complex ways. A markov chain can extend a simple probability distribution to say that the probability of a certain word depends on the word(s) which have been said previously. A hidden Markov model may say that the probability of a certain words depends on the latent/hidden state of the previous word,

https://pomegranate.readthedocs.io/en/latest/

Tutorials available on GitHub

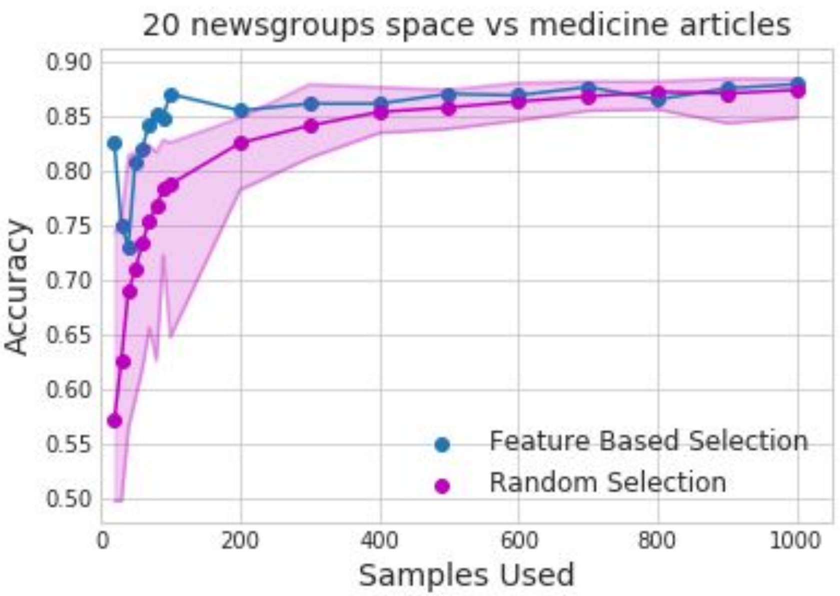


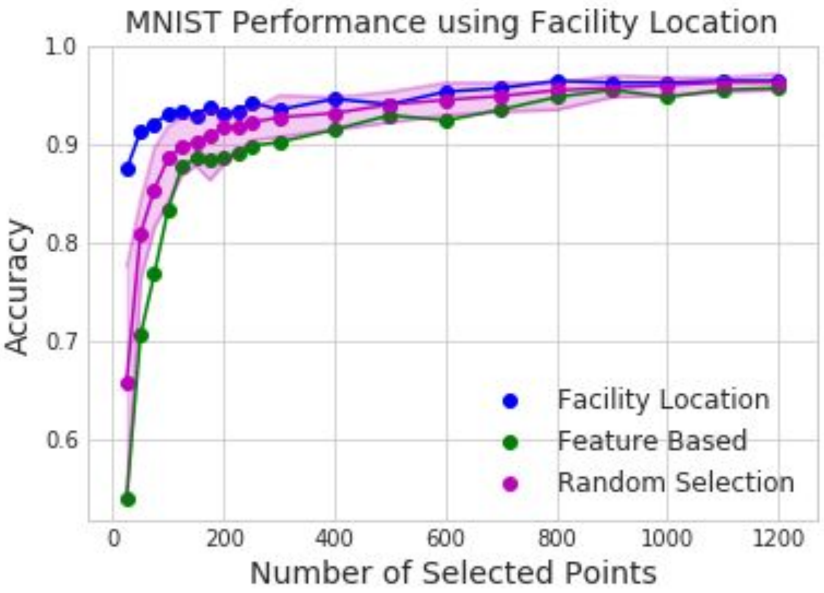
jmschrei ENH NB/BC notebook — Latest commit 5cd8d68 5 days ago

..		
old	ADD new overview tutorial	a month ago
A_Overview.ipynb	ADD new notebook features	12 days ago
B_Model_Tutorial_1_Distributions.ipynb	ENH NB/BC notebook	5 days ago
B_Model_Tutorial_2_General_Mixture_Models.ipynb	ADD new notebook features	12 days ago
B_Model_Tutorial_3_Hidden_Markov_Models.ipynb	ADD new notebook features	12 days ago
B_Model_Tutorial_4_Bayesian_Networks.ipynb	ENH NB/BC notebook	5 days ago
B_Model_Tutorial_4b_Bayesian_Network_Structure_Learning.ip...	ADD new notebook features	12 days ago
B_Model_Tutorial_5_Bayes_Classifiers.ipynb	ENH NB/BC notebook	5 days ago
B_Model_Tutorial_6_Markov_Chain.ipynb	ADD new notebook features	12 days ago
C_Feature_Tutorial_1_Parallelization_and_GPUs.ipynb	ADD new notebook features	12 days ago
C_Feature_Tutorial_8_Semisupervised_Learning.ipynb	ADD new notebook features	12 days ago
C_Feature_Tutorial_9_Missing_Values.ipynb	ADD new notebook features	12 days ago
GGBlasts.xlsx	PyData Chicago 2016	2 years ago
README.md	Update README.md	3 years ago

https://github.com/jmschrei/pomegranate/tree/master/tutorials

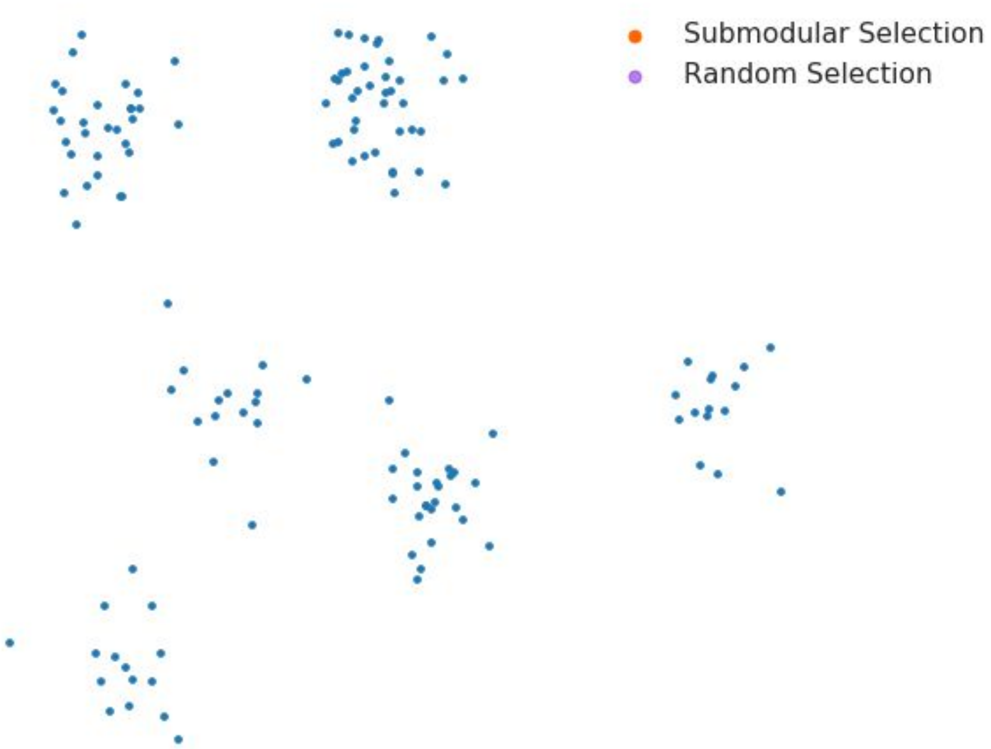
apricot implements submodular selection for training machine learning models faster





https://github.com/jmschrei/apricot

apricot implements submodular selection for training machine learning models faster



https://github.com/jmschrei/apricot

pomegranate

fast and flexible probabilistic modelling in python

Jacob Schreiber
Paul G. Allen School of Computer Science & Engineering
University of Washington

jmschreiber91
@jmschrei
@jmschreiber91