pomegranate

fast and flexible probabilistic modelling in python

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

 jmschreiber91

 @jmschrei

 @jmschreiber91

Acknowledgements

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

Overview

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

Overview: supported models

Six Main Models:

1. Probability Distributions
2. General Mixture Models
3. Markov Chains
4. Hidden Markov Models
5. Bayes Classifiers / Naive Bayes
6. Bayesian Networks

Two Helper Models:

1. k-means++/kmeans||
2. Factor Graphs

pomegranate supports many distributions

Univariate Distributions

1. UniformDistribution
2. BernoulliDistribution
3. NormalDistribution
4. LogNormalDistribution
5. ExponentialDistribution
6. BetaDistribution
7. GammaDistribution
8. DiscreteDistribution
9. PoissonDistribution

Kernel Densities

1. GaussianKernelDensity
2. UniformKernelDensity
3. TriangleKernelDensity

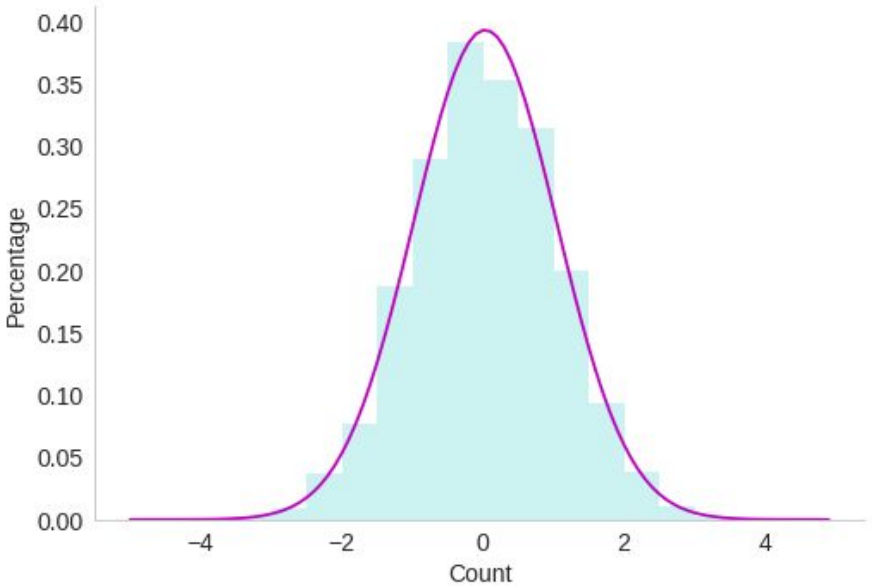
Multivariate Distributions

1. IndependentComponentsDistribution
2. MultivariateGaussianDistribution
3. DirichletDistribution
4. ConditionalProbabilityTable
5. JointProbabilityTable

Models can be made in two ways...

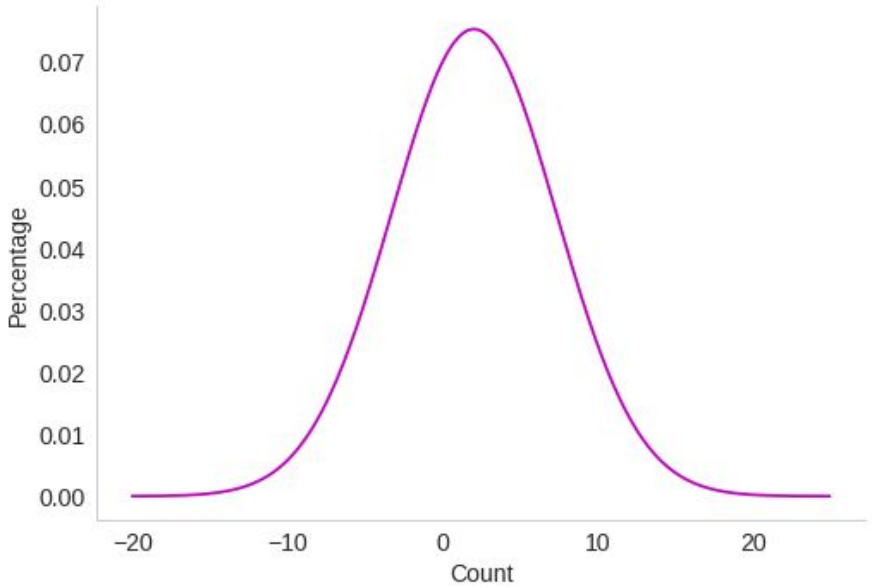
...from data

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



...from known values

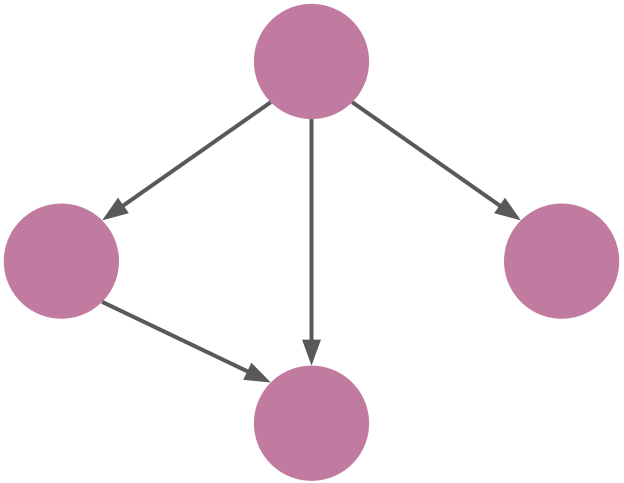
```
d = NormalDistribution(5, 2.3)
```



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)

model.predict(X)

model.predict_proba(X)

model.predict_log_proba(X)

All models have these methods!

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

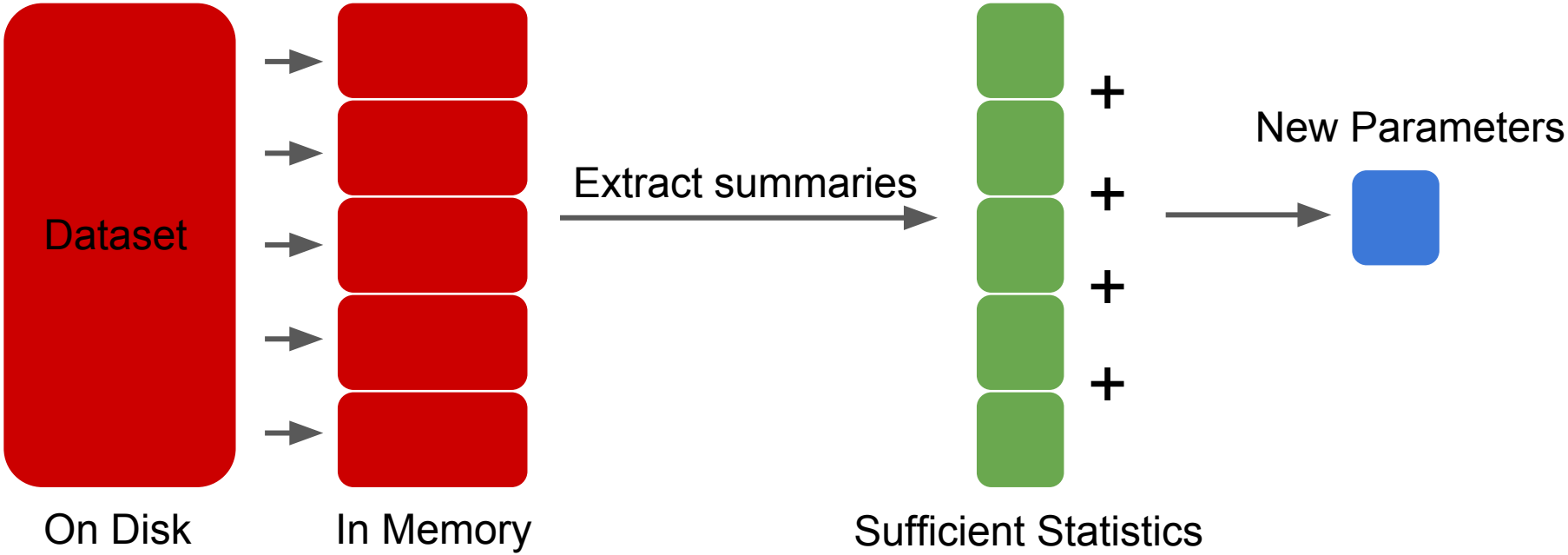
$$\sum_{i=1}^{n} w_i \qquad \sum_{i=1}^{n} w_i x_i \qquad \sum_{i=1}^{n} w_i x_i^2 \longrightarrow$$

$$\mu = \frac{\sum_{i=1}^{n} w_i x_i}{\sum_{i=1}^{n} w_i}$$

$$\sigma^2 = \frac{\sum_{i=1}^{n} w_i x_i^2}{\sum_{i=1}^{n} w_i} - \frac{\left(\sum_{i=1}^{n} w_i x_i\right)^2}{\left(\sum_{i=1}^{n} w_i\right)^2}$$

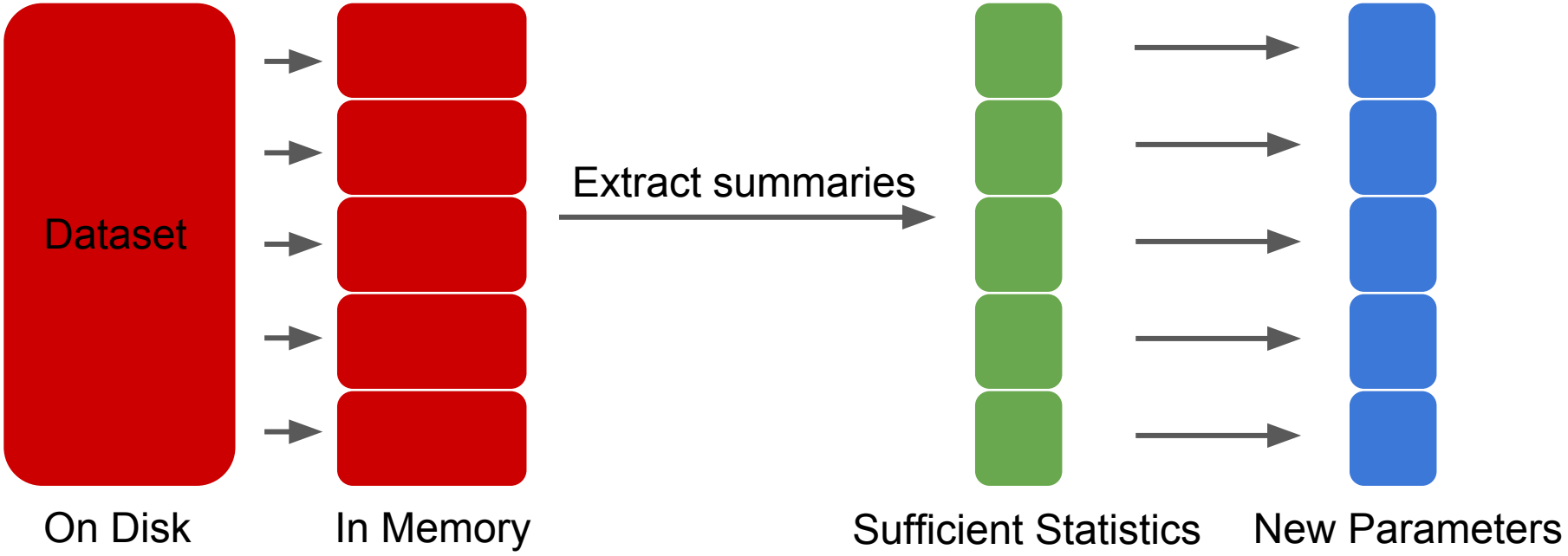
pomegranate supports out-of-core learning

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



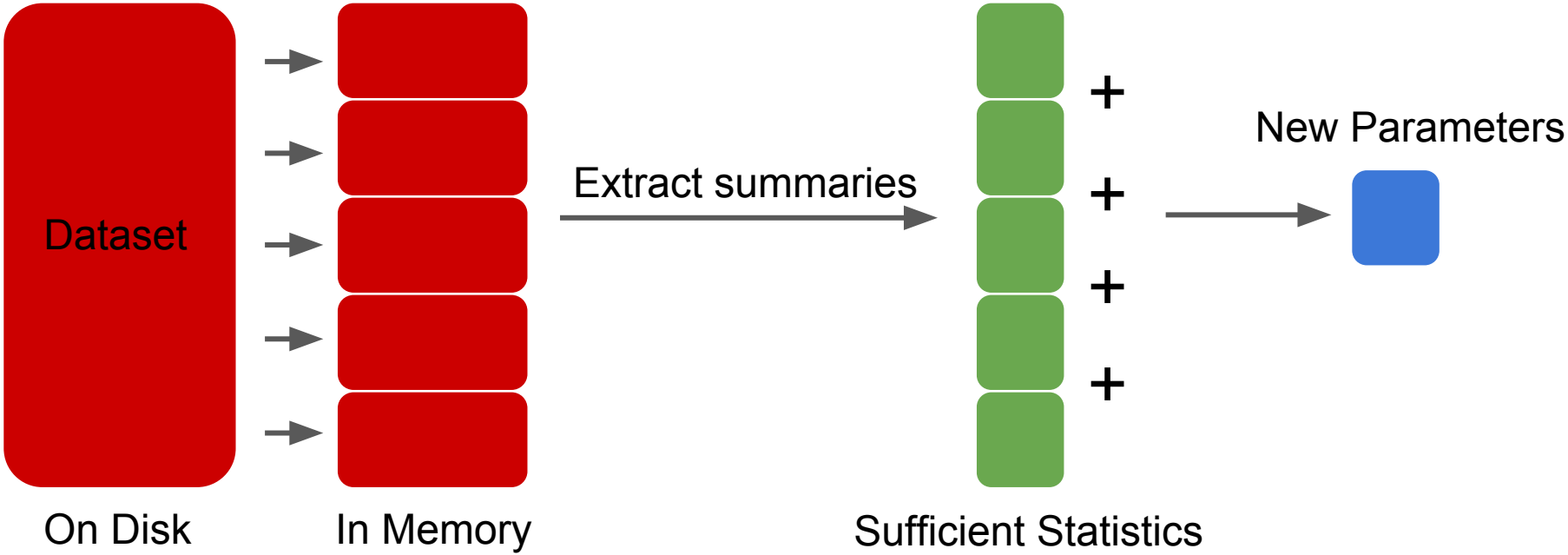
pomegranate supports mini-batching

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

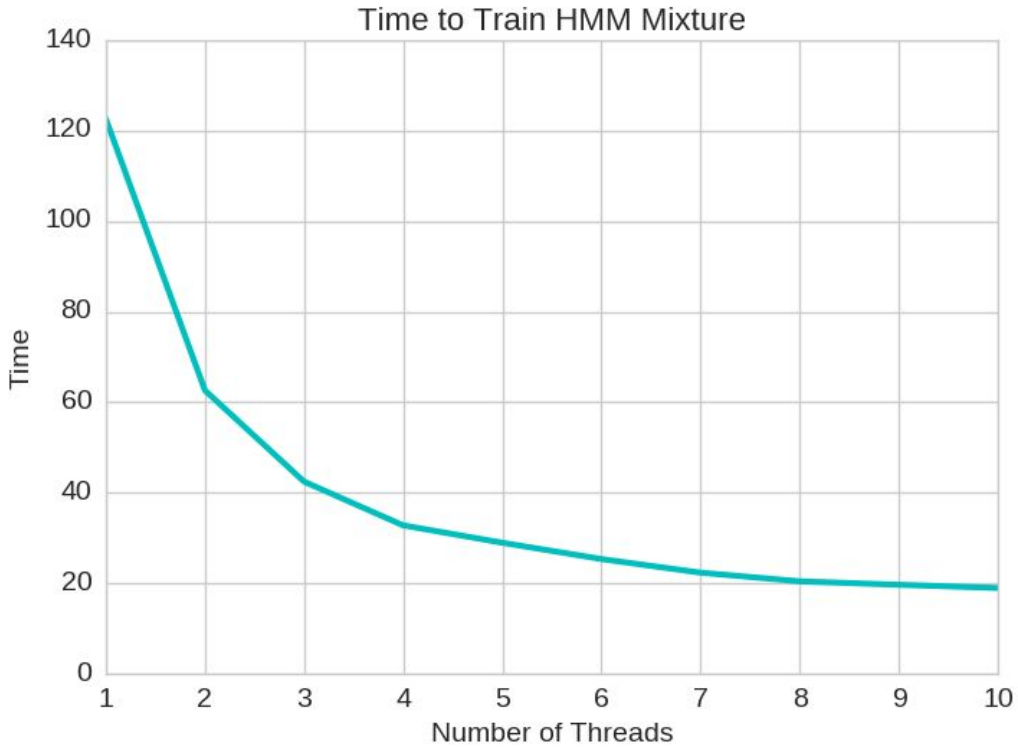


pomegranate supports parallelization

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



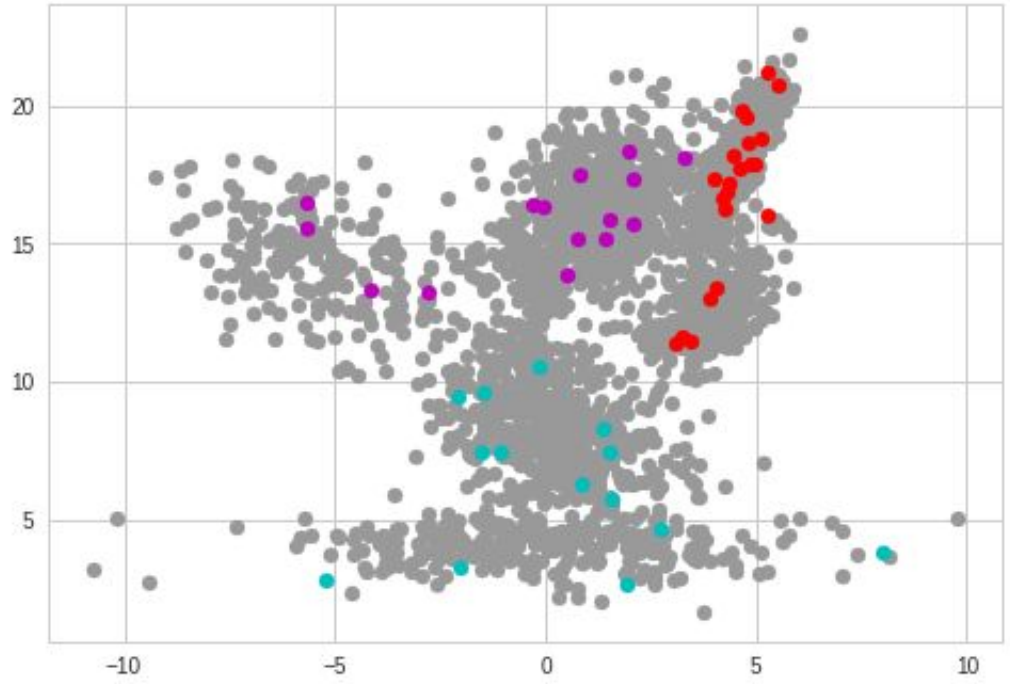
Training a mixture of HMMs in parallel




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

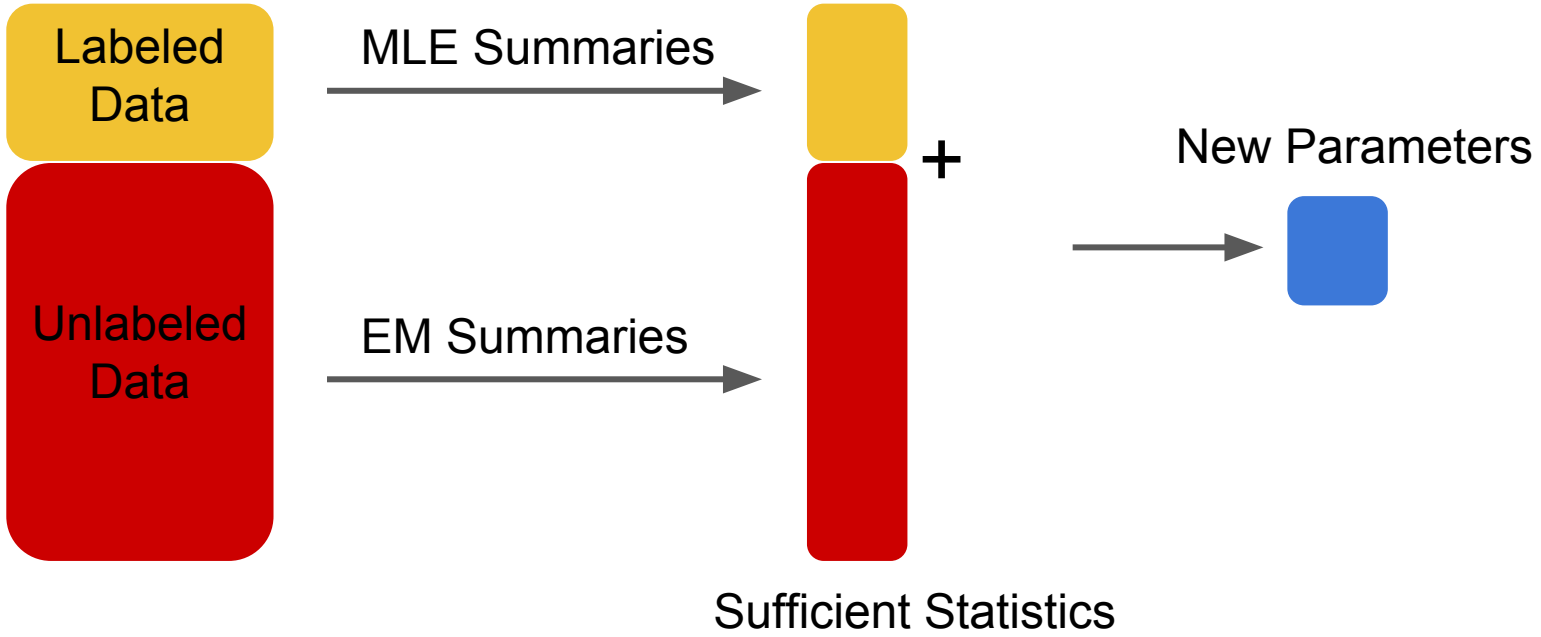
pomegranate supports semisupervised learning

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



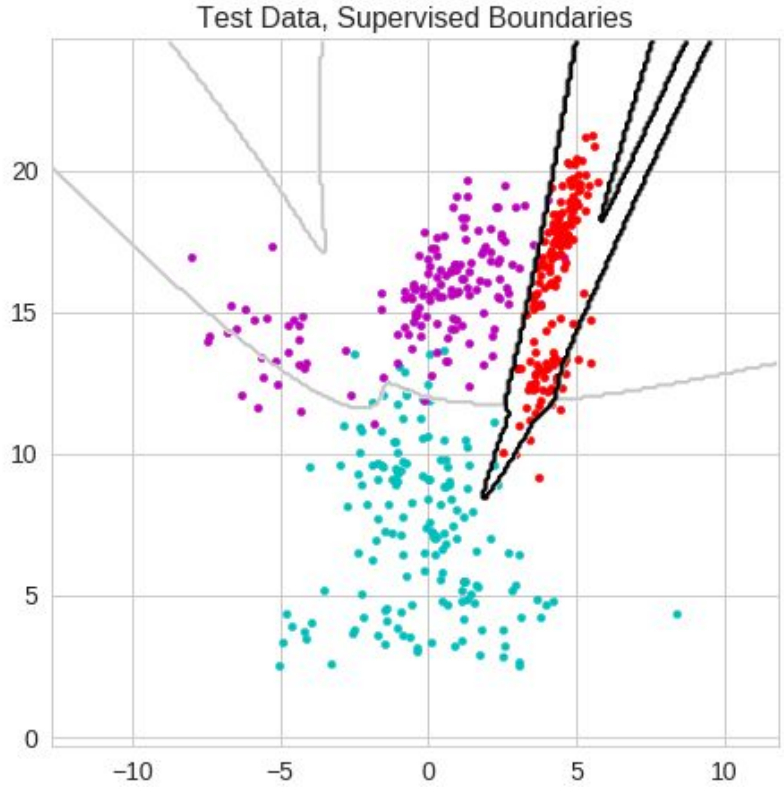
pomegranate supports semisupervised learning

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

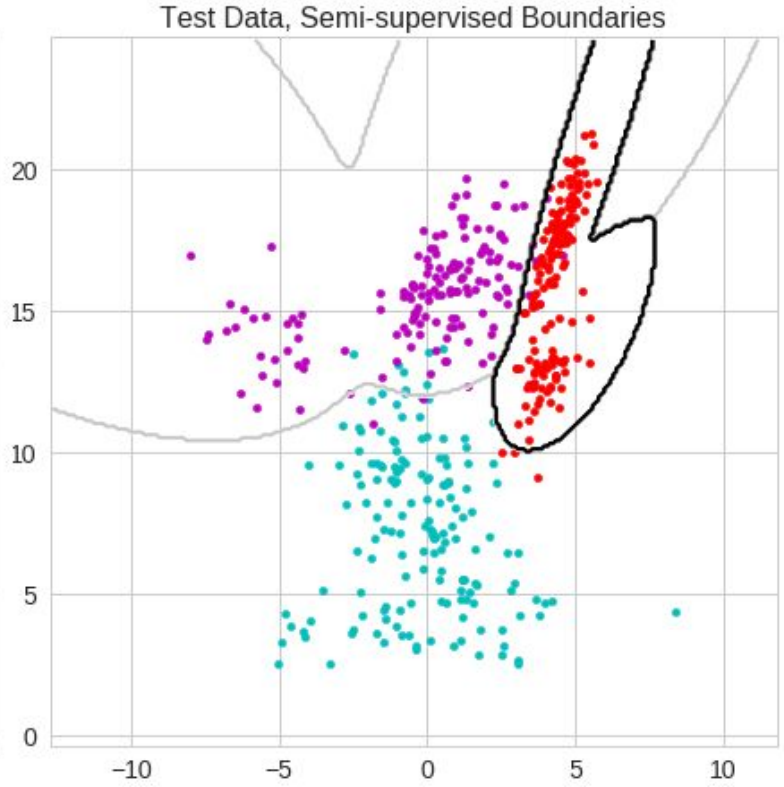


pomegranate supports semisupervised learning

Supervised Accuracy: 0.93

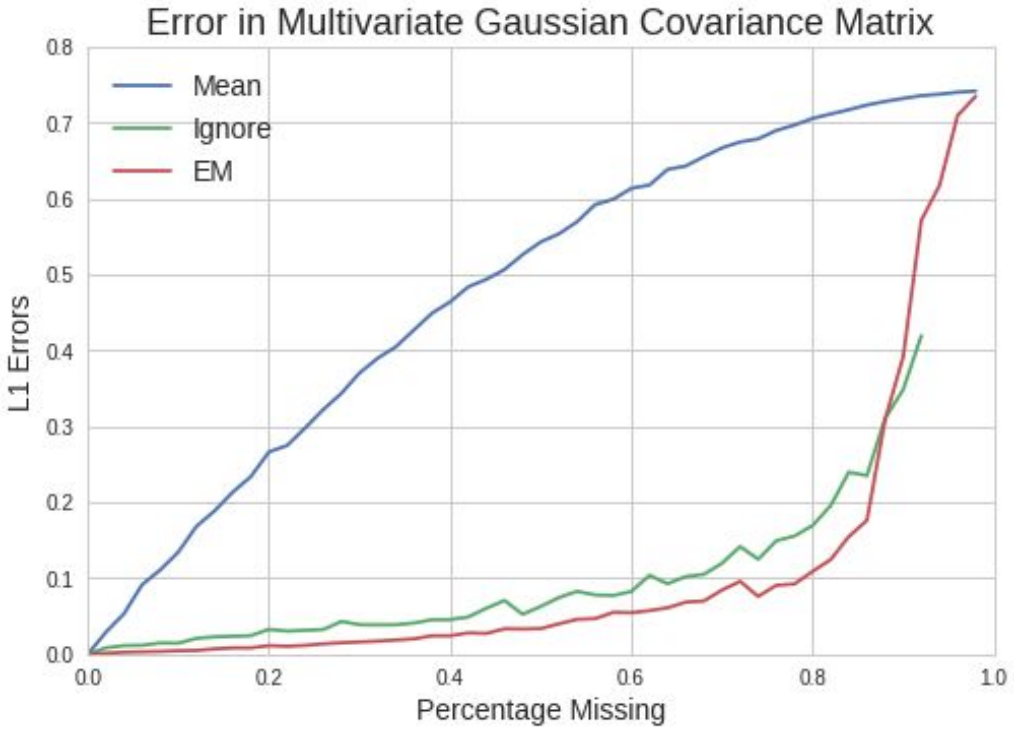


Semisupervised Accuracy: 0.96

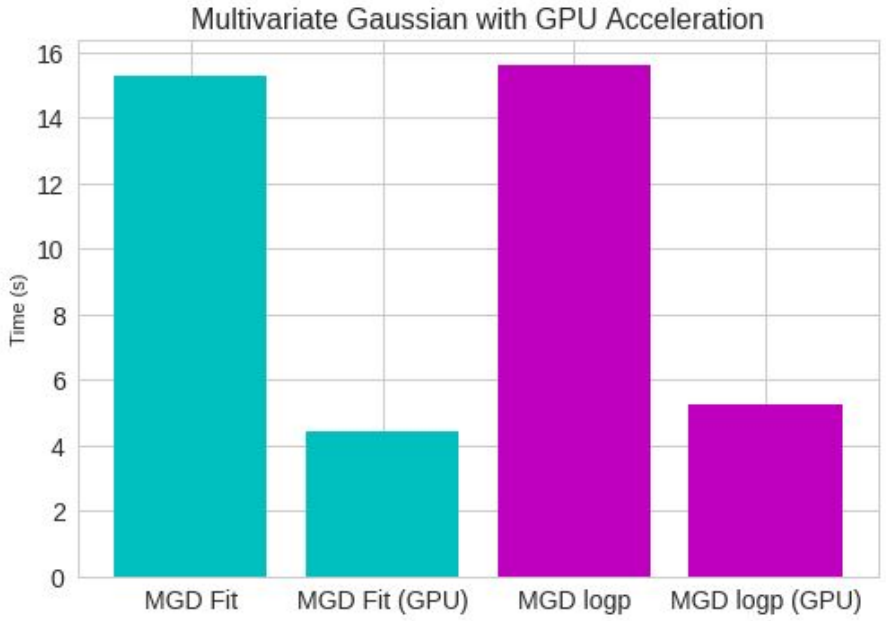


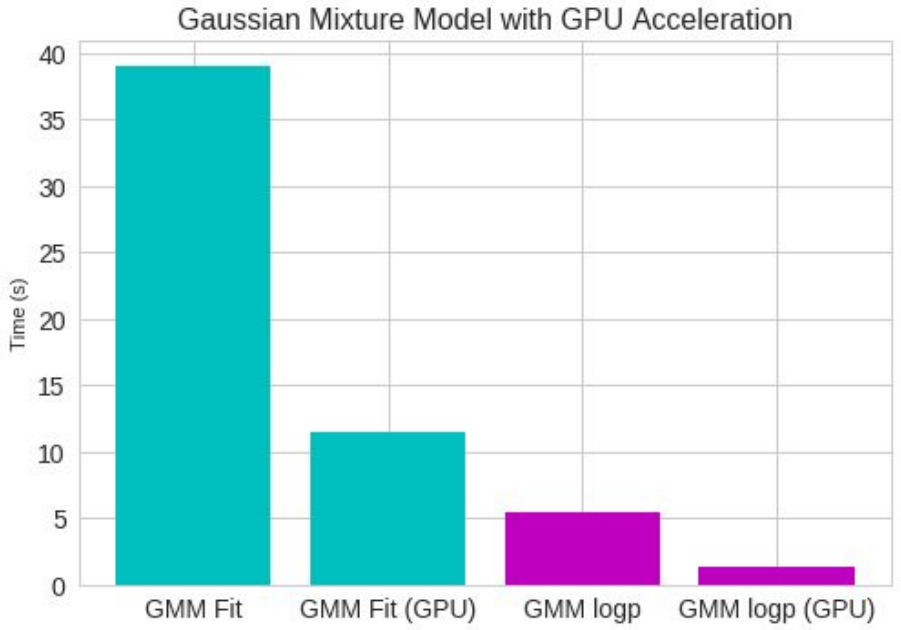
pomegranate will soon support missing data

Many real world tasks involve data sets with missing data. The next version of pomegranate will include handling for all models by ignoring missing data, mean imputation, and EM imputation.



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

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

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

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


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

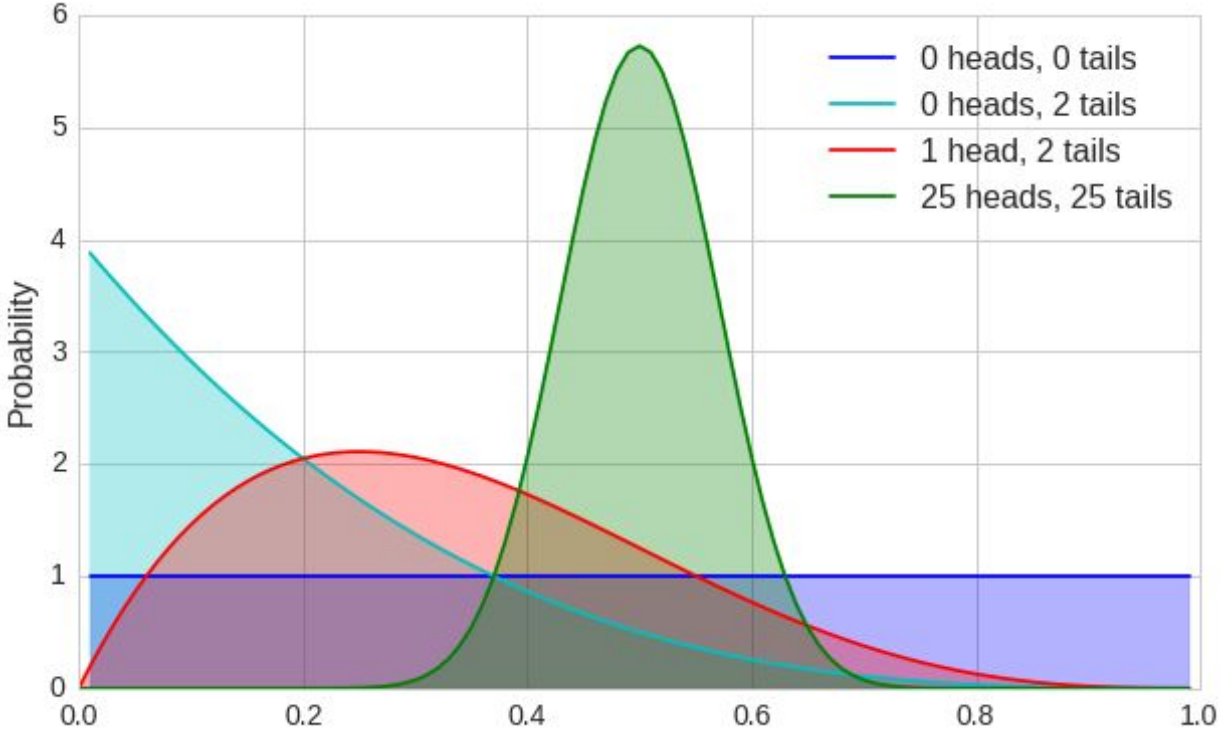
How do we encode these ‘blasts’?

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.

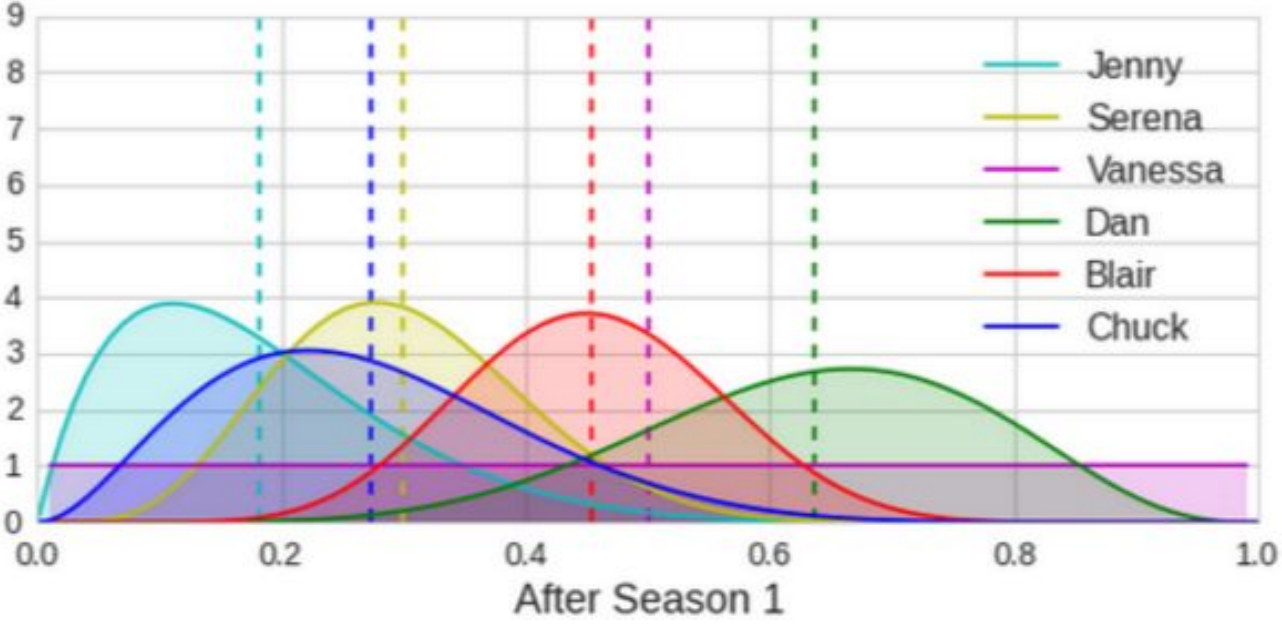
-1 Blair
-1 Serena

Simple summations don’t work well

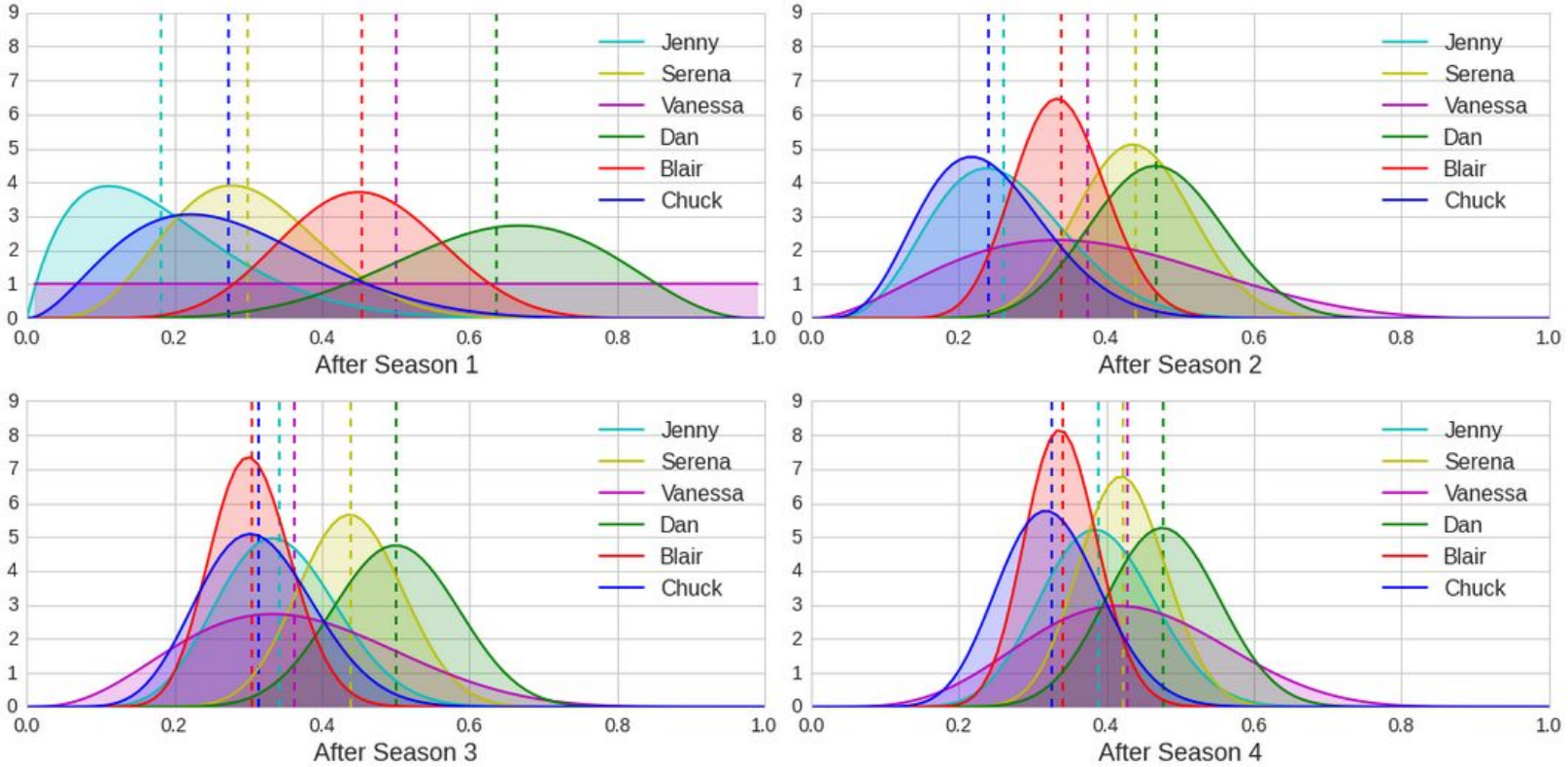
Beta distributions can model uncertainty



Beta distributions can model uncertainty



Beta distributions can model uncertainty



Overview: this talk

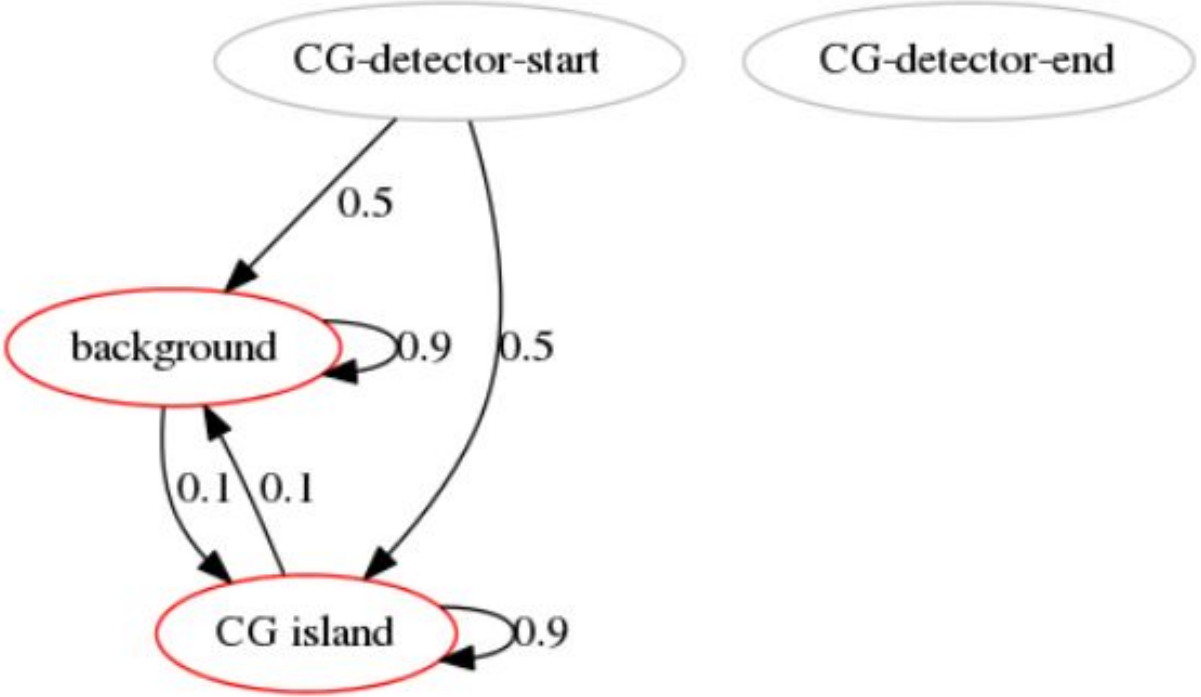
Overview

Major Models/Model Stacks

1. Hidden Markov Models
2. Bayes Classifiers
3. Bayesian Networks

CG enrichment detection HMM

GACTACGACTCGCGCTCGCACGTCGCTCGACATCATCGACA



CG enrichment detection HMM

GACTACGACTCGCGCTCGCACGTCGCTCGACATCATCGACA

```
d1 = DiscreteDistribution({'A': 0.25, 'C': 0.25, 'G': 0.25, 'T': 0.25})
d2 = DiscreteDistribution({'A': 0.10, 'C': 0.40, 'G': 0.40, 'T': 0.10})

s1 = State(d1, name="background")
s2 = State(d2, name="CG island")

hmm = HiddenMarkovModel("CG-detector")
hmm.add_states(s1, s2)
hmm.add_transition(hmm.start, s1, 0.5)
hmm.add_transition(hmm.start, s2, 0.5)
hmm.add_transition(s1, s1, 0.9)
hmm.add_transition(s1, s2, 0.1)
hmm.add_transition(s2, s1, 0.1)
hmm.add_transition(s2, s2, 0.9)
hmm.bake()
```

pomegranate HMMs are feature rich

Feature	pomegranate	hmmlearn
Graph Structure		
Silent States	✓	
Optional Explicit End State	✓	
Sparse Implementation	✓	
Arbitrary Emissions Allowed on States	✓	
Discrete/Gaussian/GMM Emissions	✓	✓
Large Library of Other Emissions	✓	
Build Model from Matrices	✓	✓
Build Model Node-by-Node	✓	
Serialize to JSON	✓	
Serialize using Pickle/Joblib	✓	✓

Algorithms		
Priors		✓
Sampling	✓	✓
Log Probability Scoring	✓	✓
Forward-Backward Emissions	✓	✓
Forward-Backward Transitions	✓	
Viterbi Decoding	✓	✓
MAP Decoding	✓	✓
Baum-Welch Training	✓	✓
Viterbi Training	✓	
Labeled Training	✓	
Tied Emissions	✓	
Tied Transitions	✓	
Emission Inertia	✓	
Transition Inertia	✓	
Emission Freezing	✓	✓
Transition Freezing	✓	✓
Multi-threaded Training	✓	

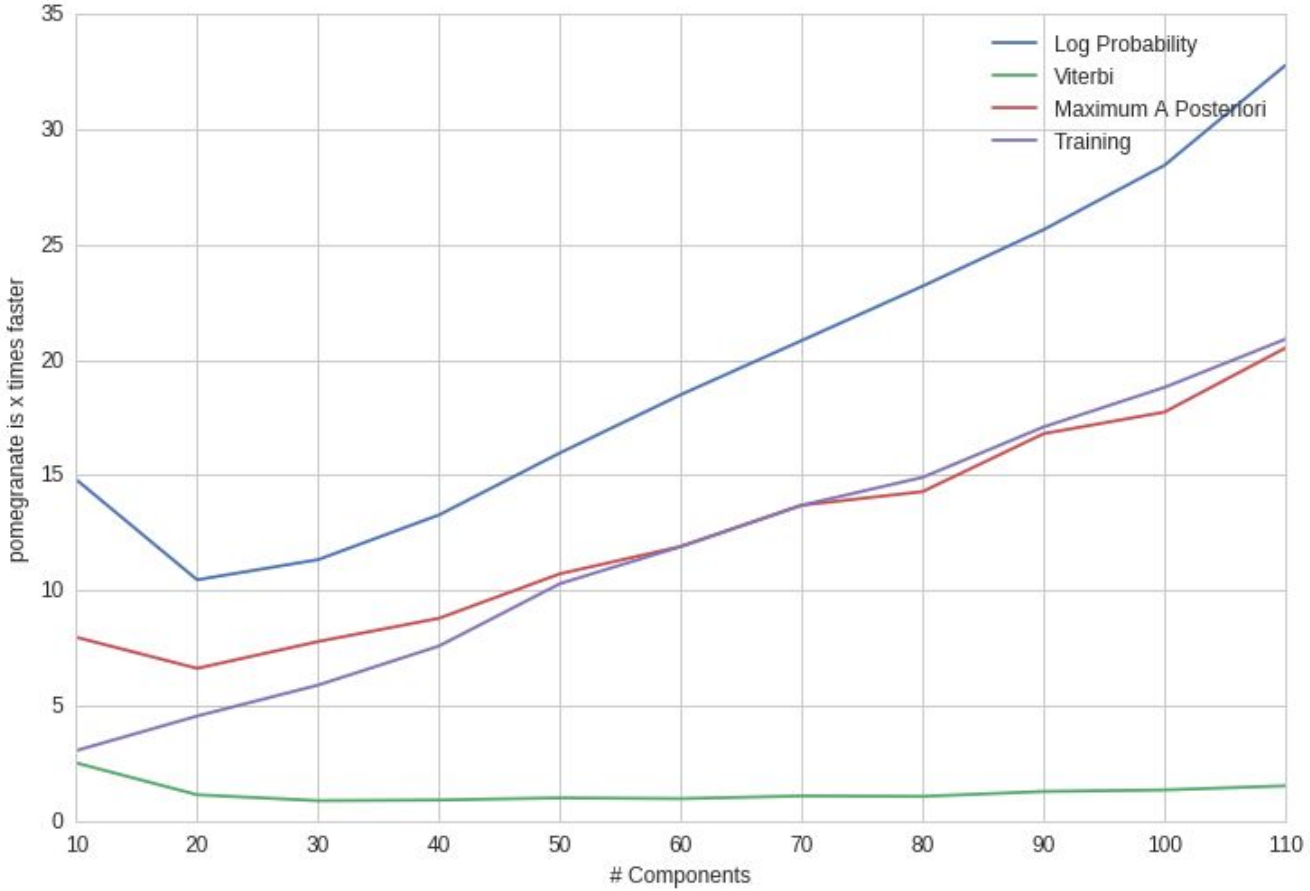
GMM-HMM easy to define

```
d1 = GeneralMixtureModel([NormalDistribution(5, 2), NormalDistribution(5, 4)])
d2 = GeneralMixtureModel([NormalDistribution(15, 1), NormalDistribution(15, 5)])

s1 = State(d1, name="GMM1")
s2 = State(d2, name="GMM2")

model = HiddenMarkovModel()
model.add_states(s1, s2)
model.add_transition(model.start, s1, 0.75)
model.add_transition(model.start, s2, 0.25)
model.add_transition(s1, s1, 0.85)
model.add_transition(s1, s2, 0.15)
model.add_transition(s2, s2, 0.90)
model.add_transition(s2, s1, 0.10)
model.bake()
```

HMMs are faster than hmmlearn



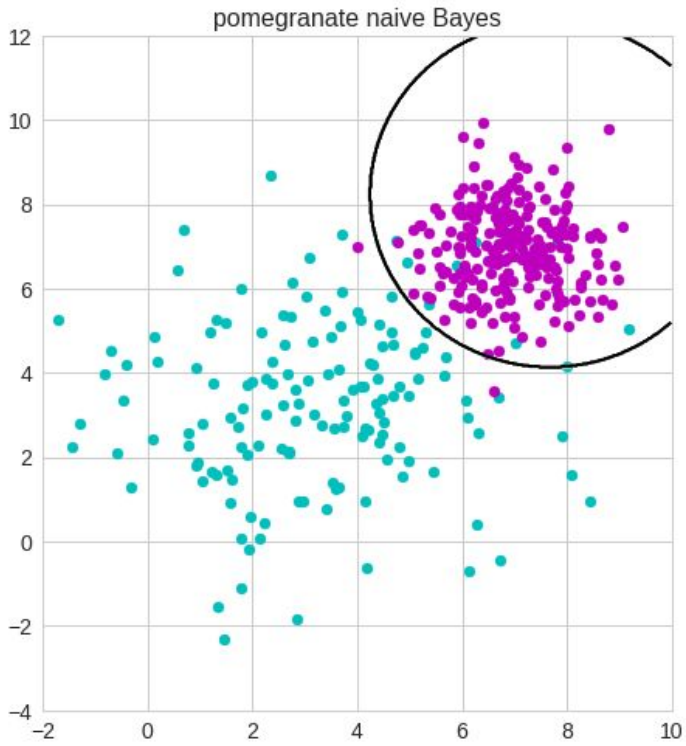
Overview: this talk

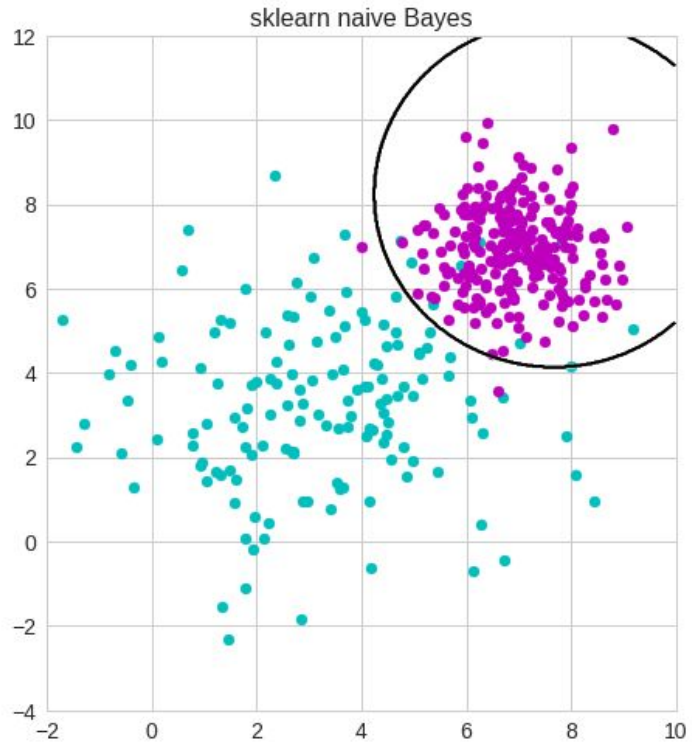
Overview

Major Models/Model Stacks

1. Hidden Markov Models
2. Bayes Classifiers
3. Bayesian Networks

Naive Bayes produces ellipsoid boundaries






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

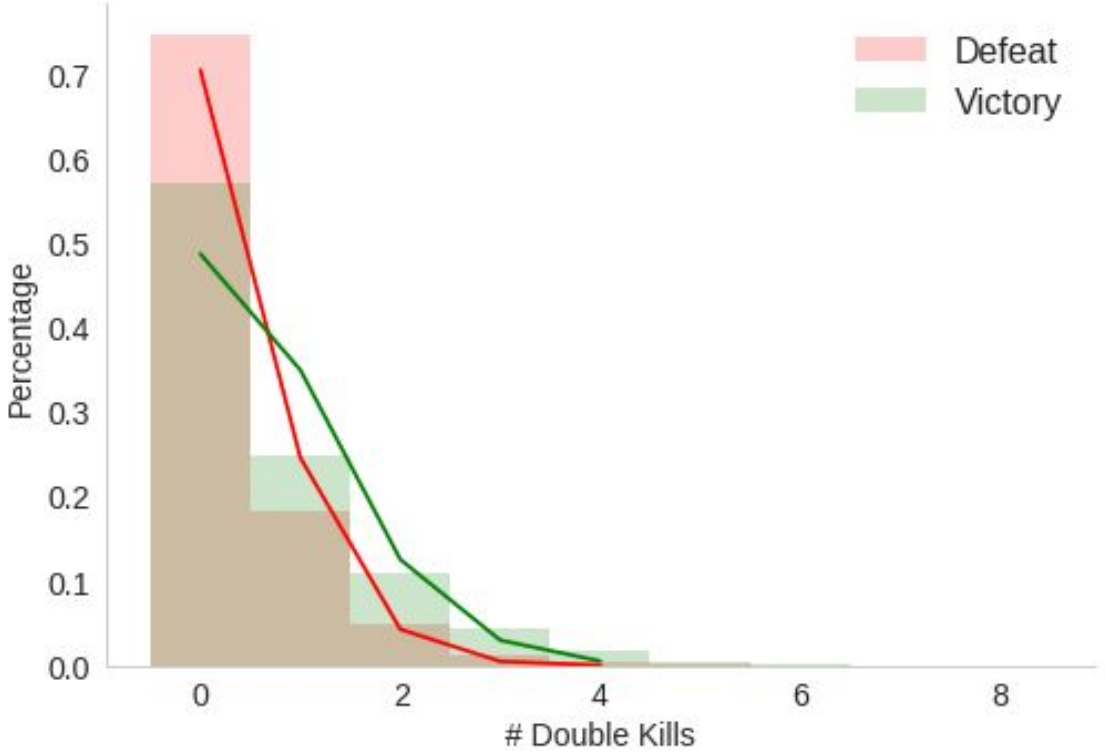
Naive Bayes assumes independent features

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

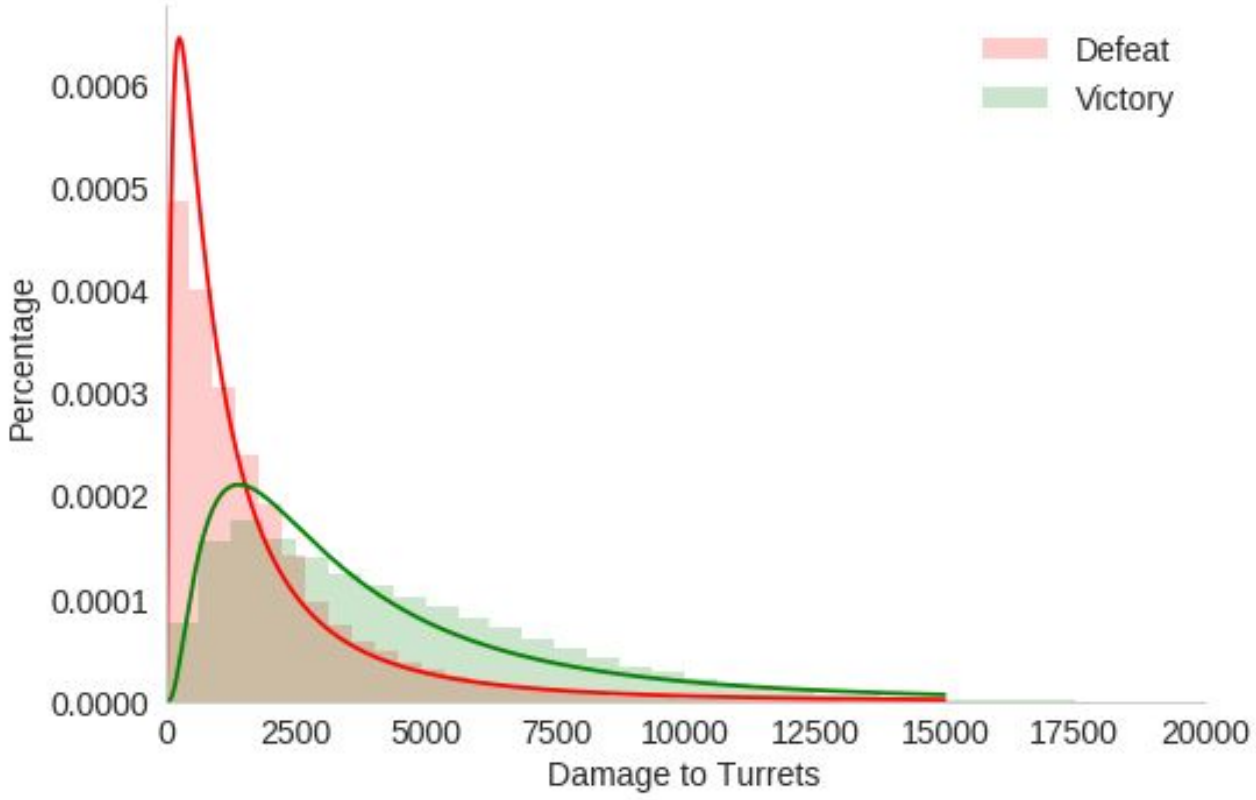
$$P(M|D) = \frac{\prod_{i=1}^{d} P(D_i|M)P(M)}{\sum_{M} \prod_{i=1}^{d} P(D_i|M)P(M)}$$


LEAGUE OF LEGENDS

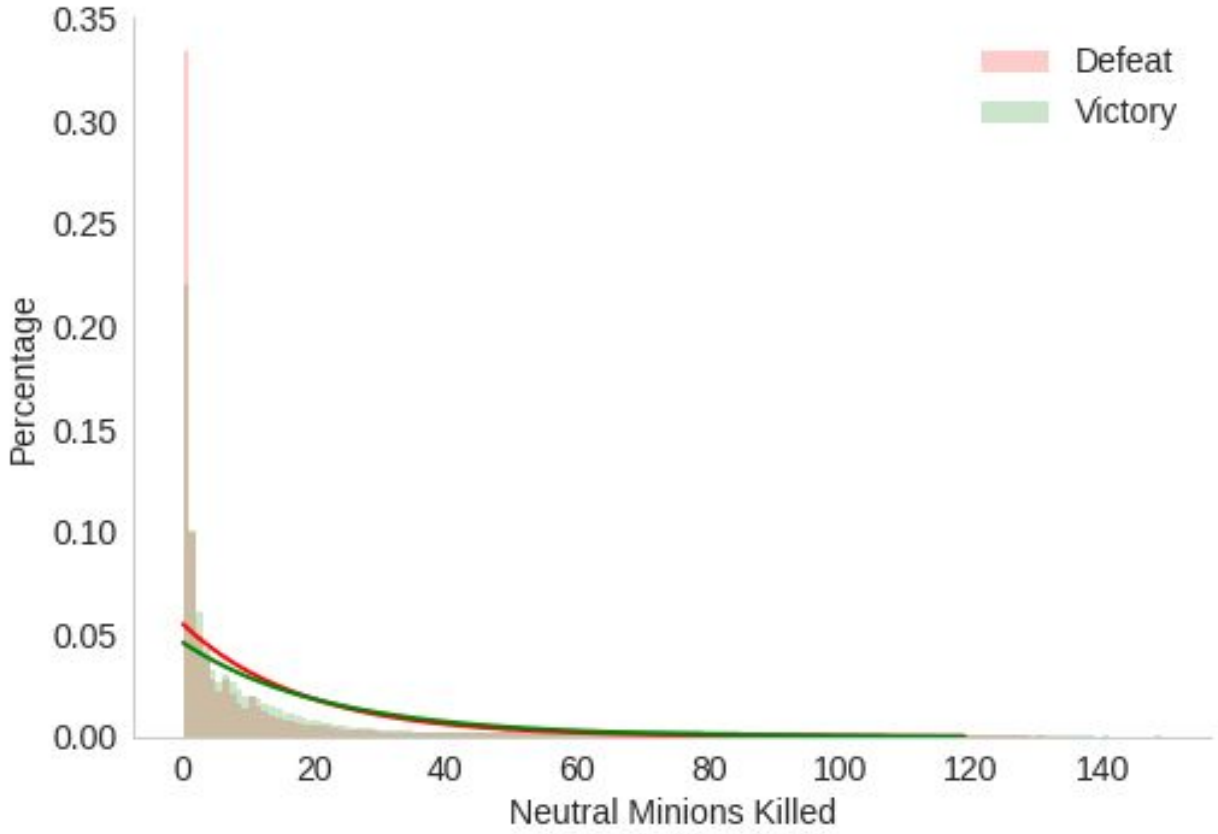
Data can fall under different distributions



Data can fall under different distributions



Data can fall under different distributions



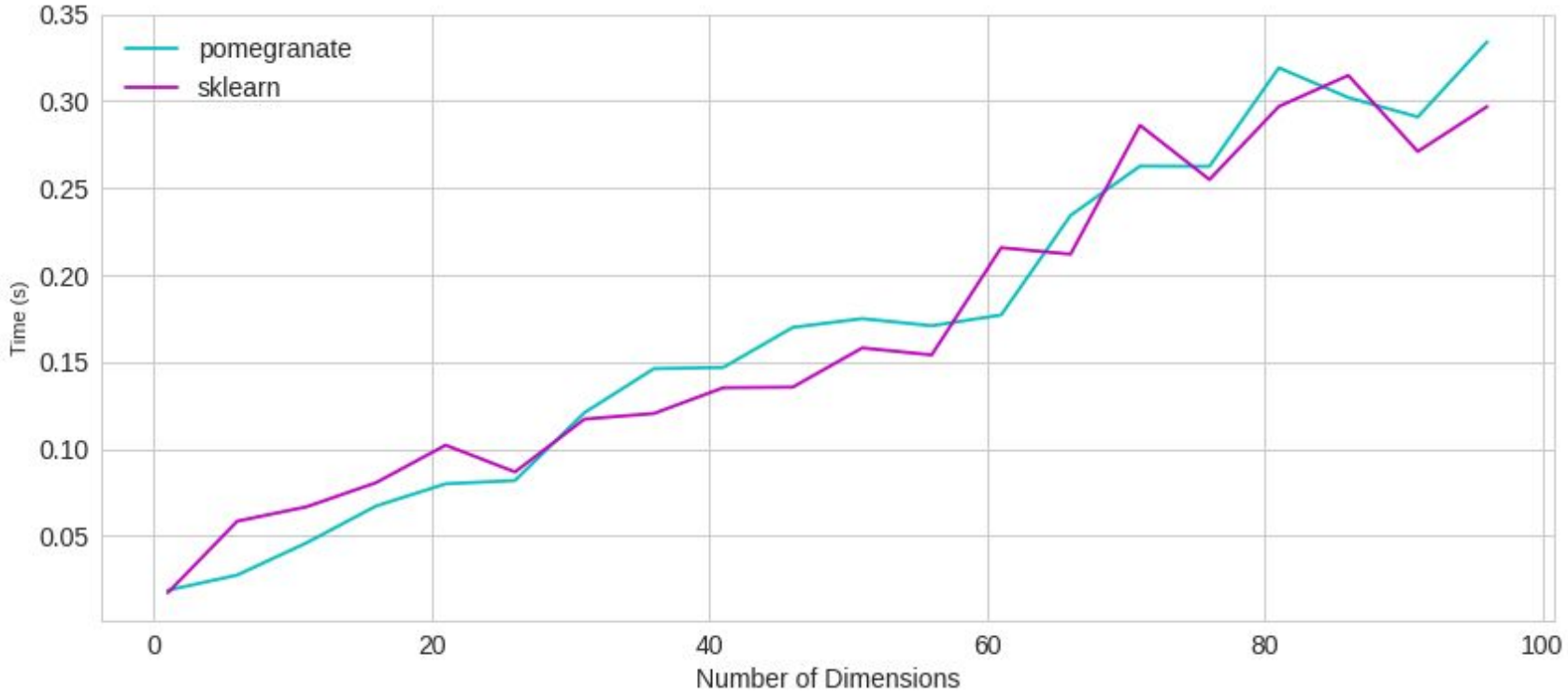
Using appropriate distributions is better

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

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

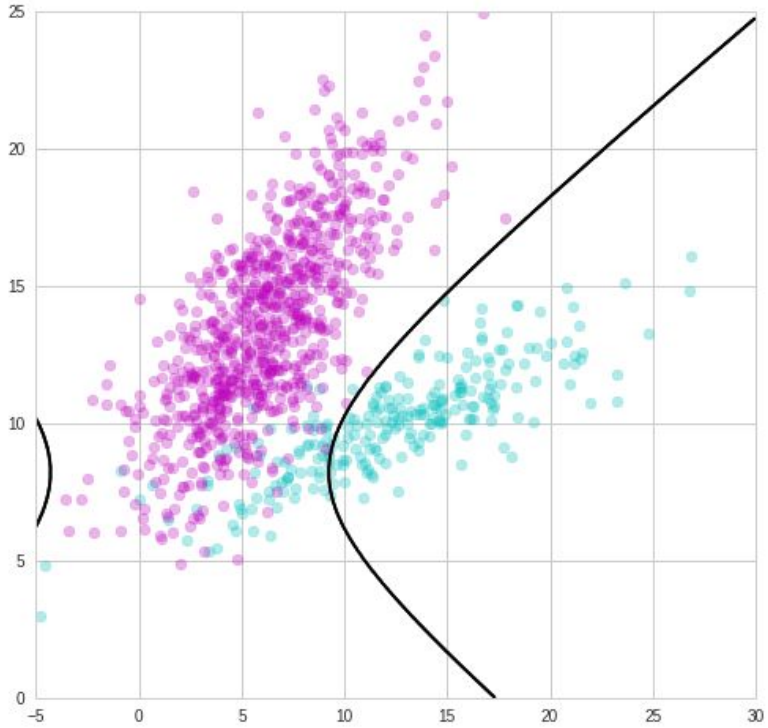
Gaussian Naive Bayes: 0.711
sklearn Gaussian Naive Bayes: 0.711
Heterogeneous Naive Bayes: 0.726

This additional flexibility is just as fast

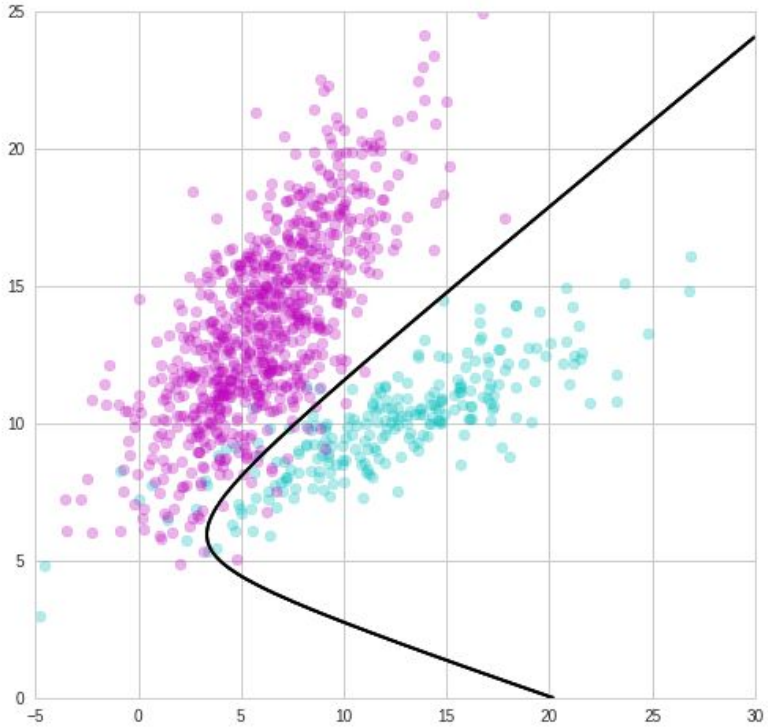


Bayes classifiers don’t require independence

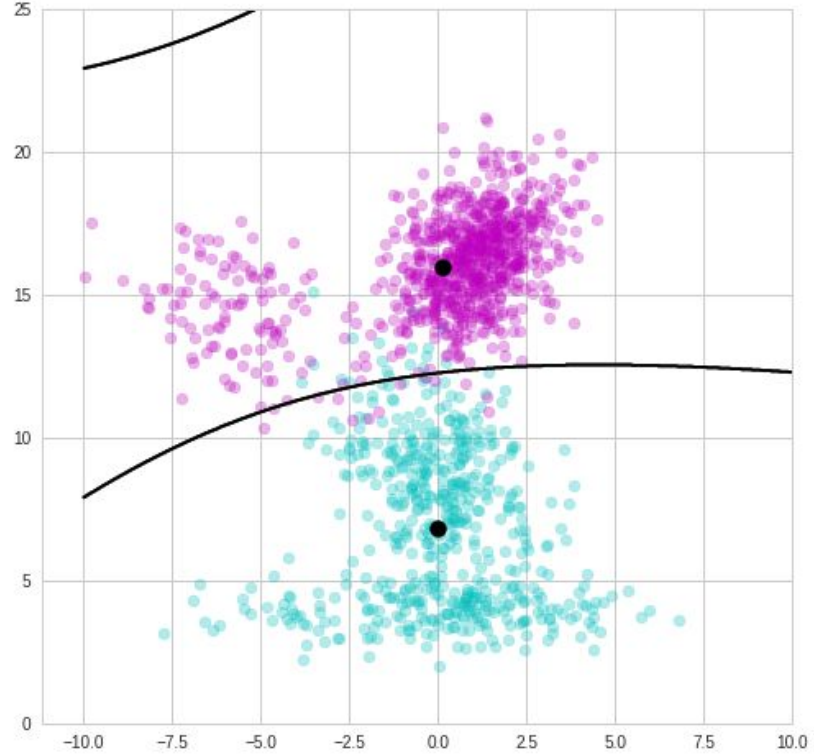
naive accuracy: 0.929

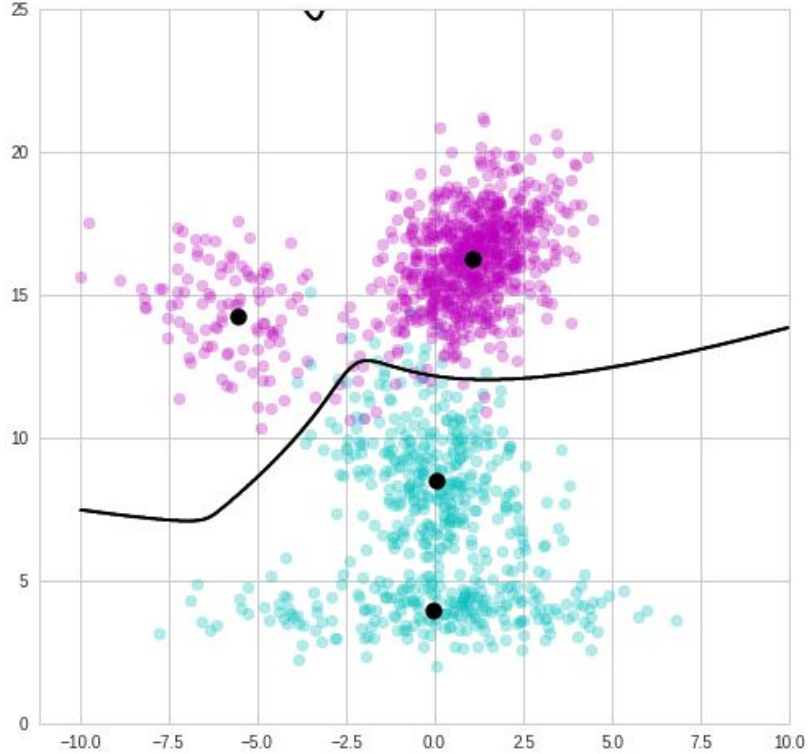


bayes classifier accuracy: 0.966



Gaussian mixture model Bayes classifier





Overview: this talk

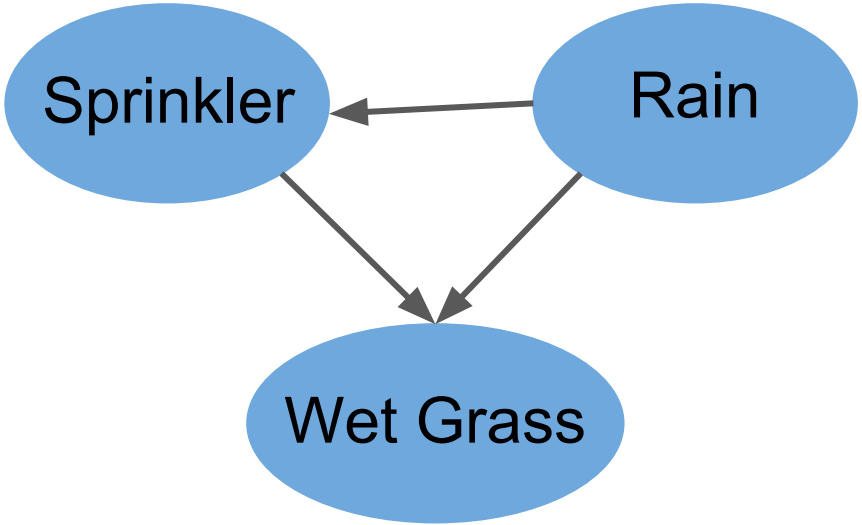
Overview

Major Models/Model Stacks

1. Hidden Markov Models
2. Bayes Classifiers
3. Bayesian Networks

Bayesian networks

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

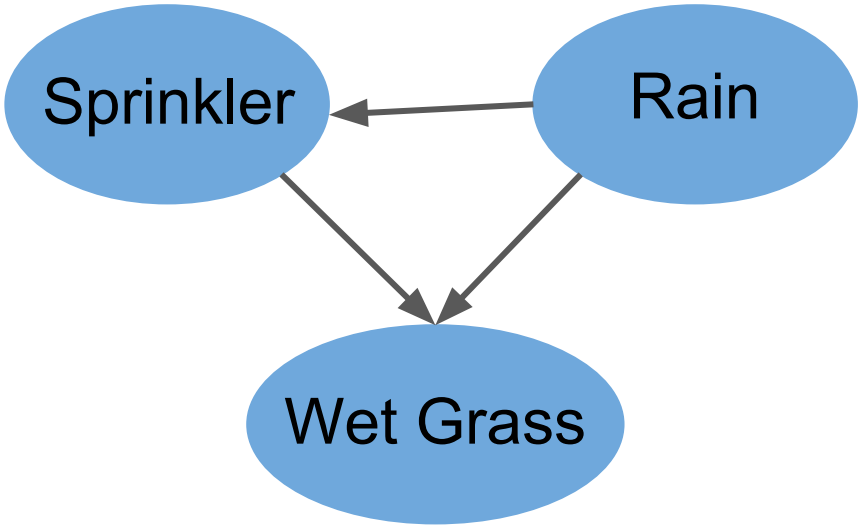


Bayesian networks

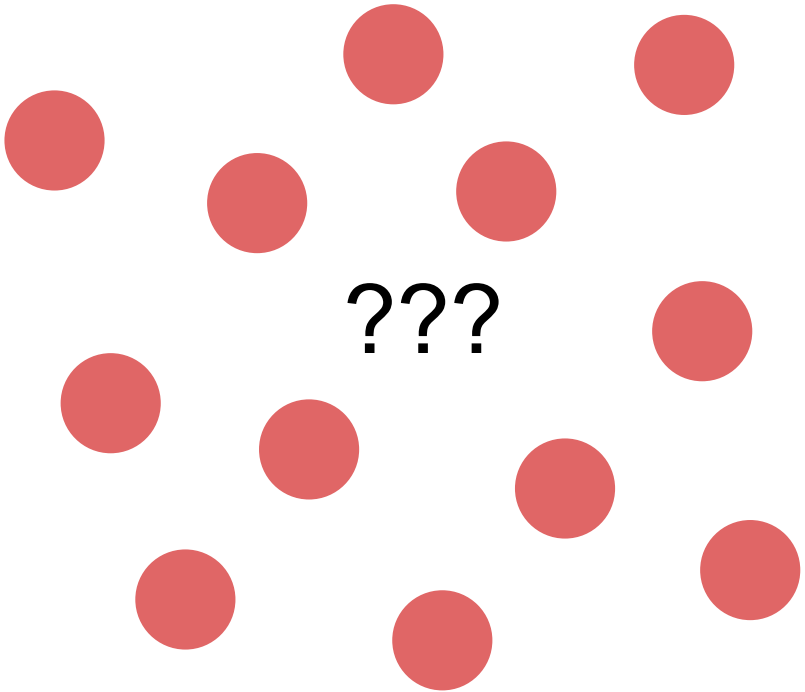
Two main difficult tasks:

(1) Inference given incomplete information

(2) Learning the dependency structure from data



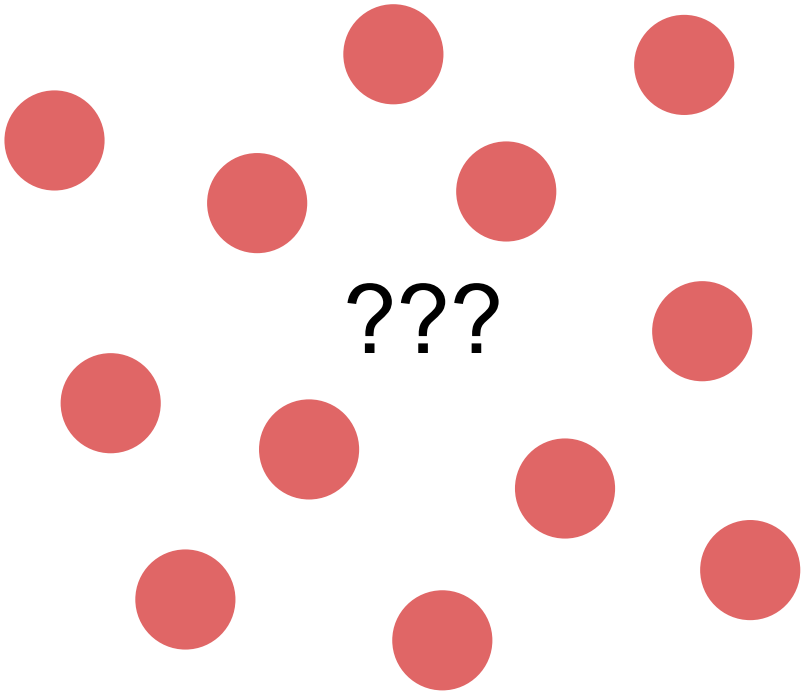
Bayesian network structure learning



Three primary ways:

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

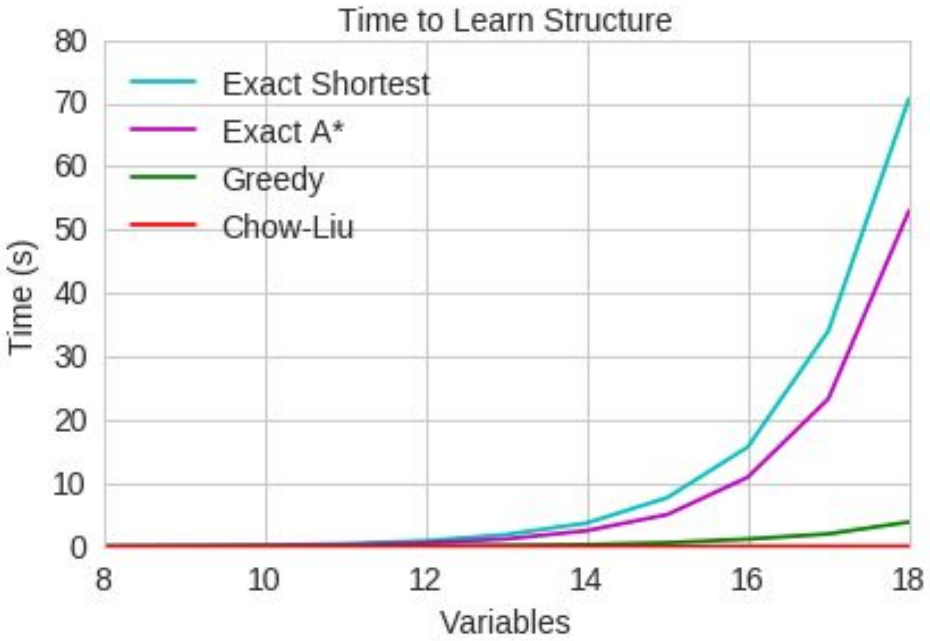
Bayesian network structure learning

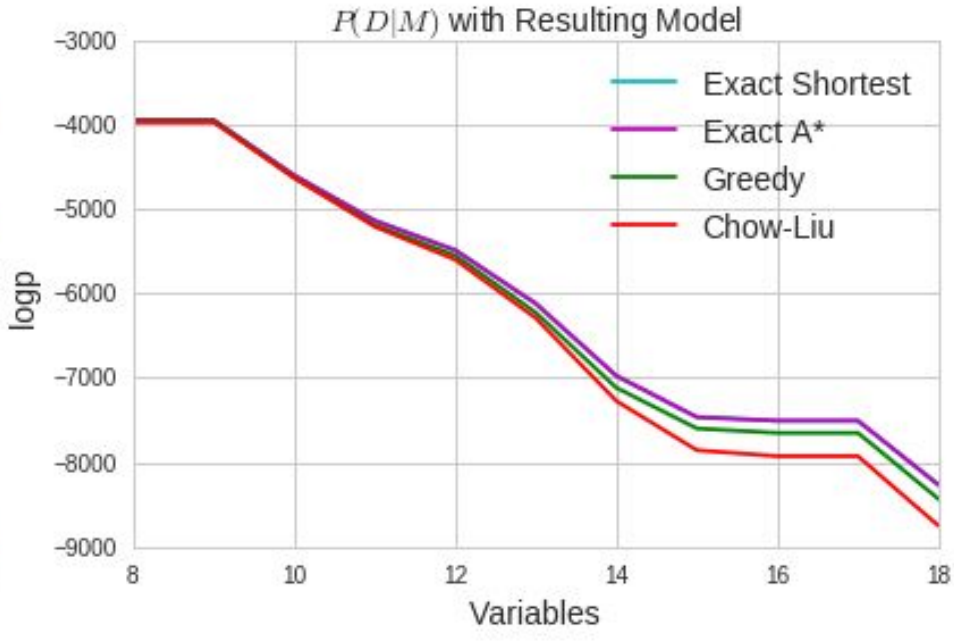


pomegranate supports:

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

pomegranate supports four algorithms





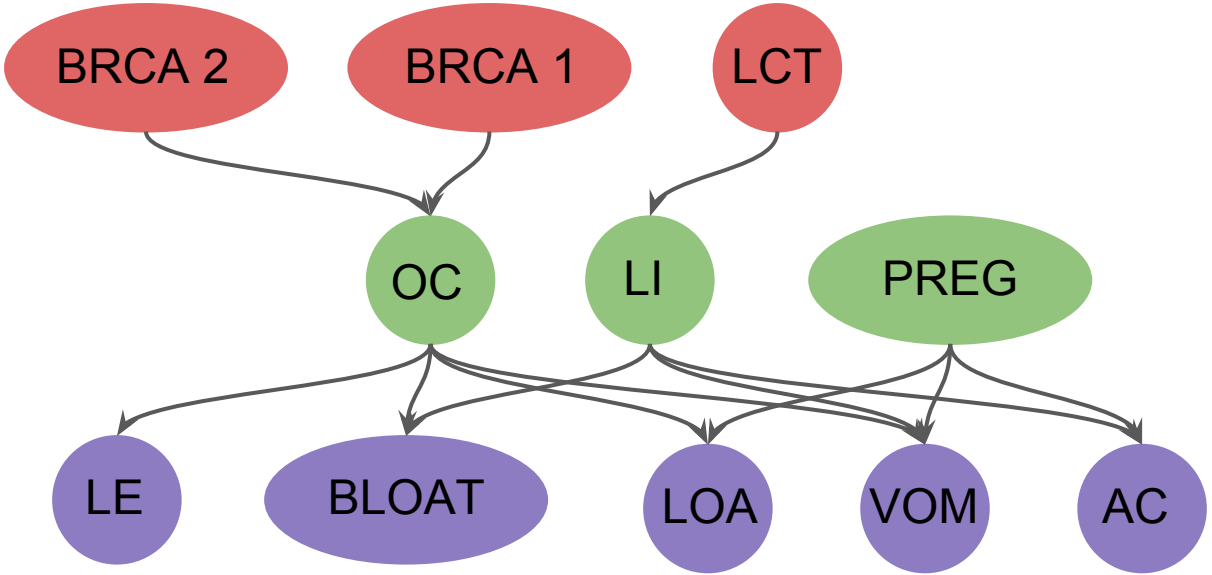
BNSL is hard due to acyclicity requirement

Easy! Tractable!

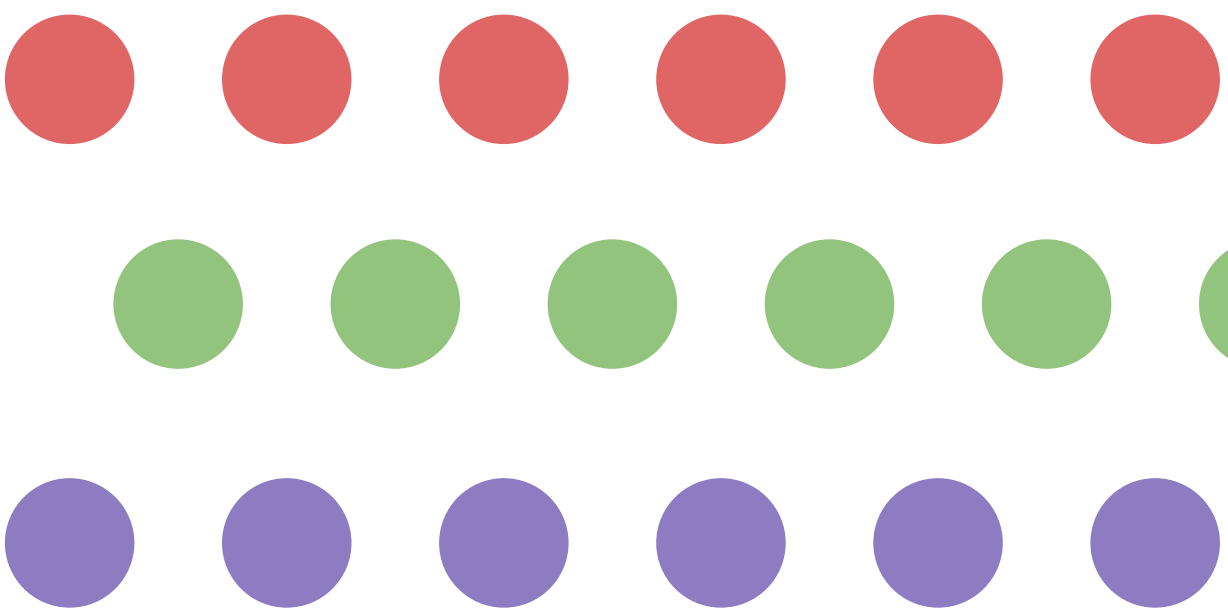
Global Parameter Independence: The parents of some variable A are independent of the parents of some variable B given that they don't form a cycle in the resulting graph

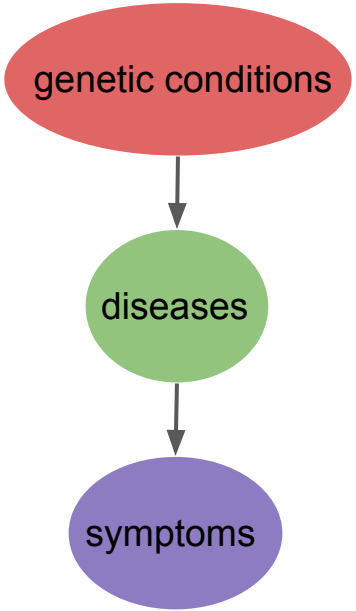
Hard! Exponential Time!

Medical diagnosis Bayesian network



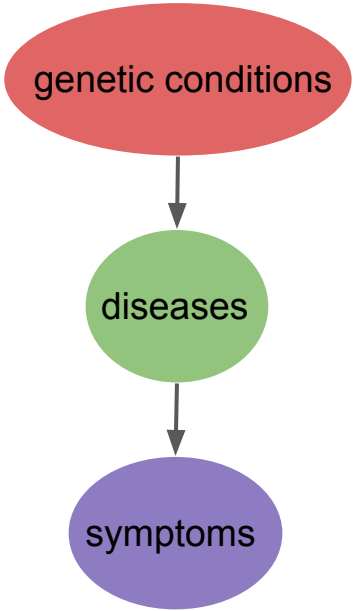
Constraint graphs merge data + knowledge





Constraint graphs merge data + knowledge

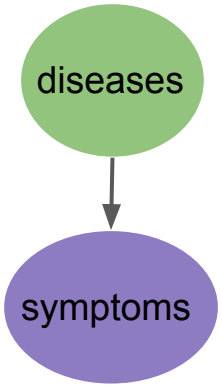
Global Parameter Independence: The parents of some variable A are independent of the parents of some variable B given that they don't form a cycle in the resulting graph



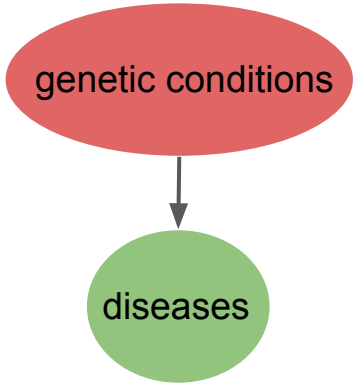
Constraint graphs merge data + knowledge

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

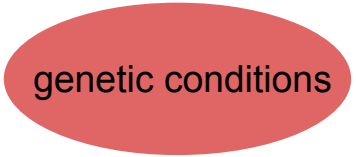
Task #1



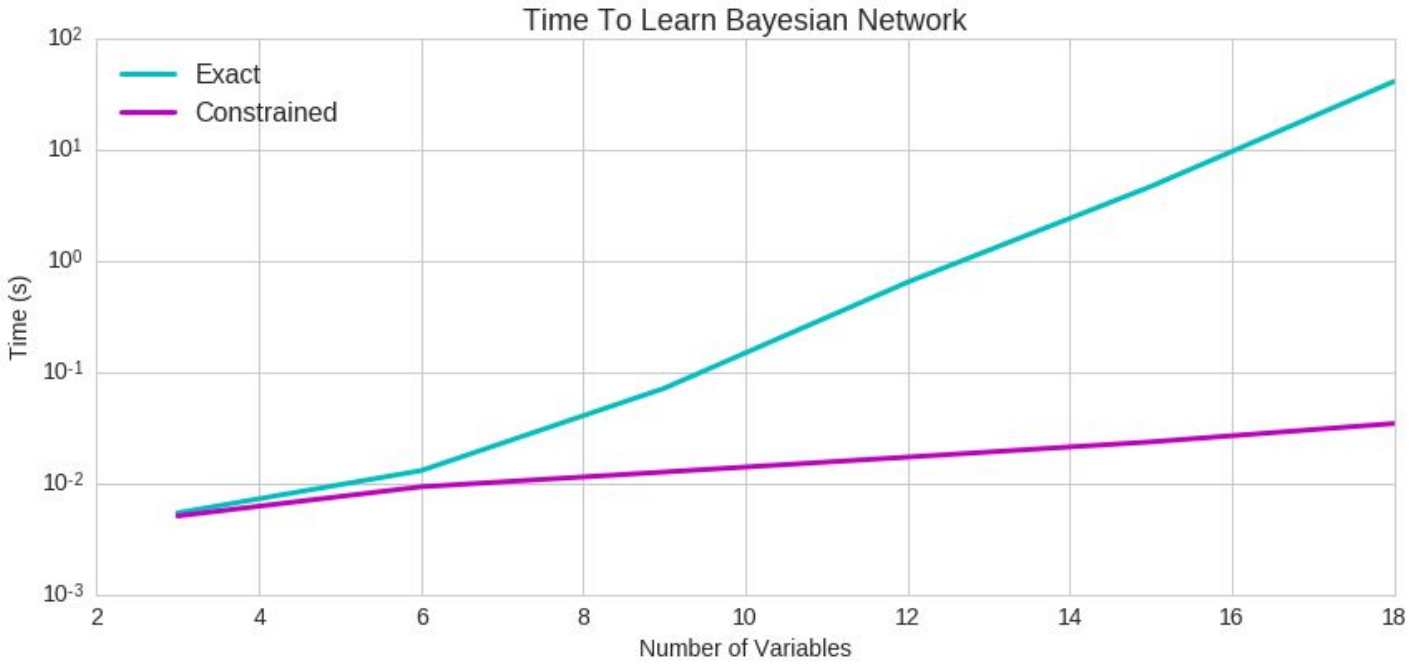
Task #2

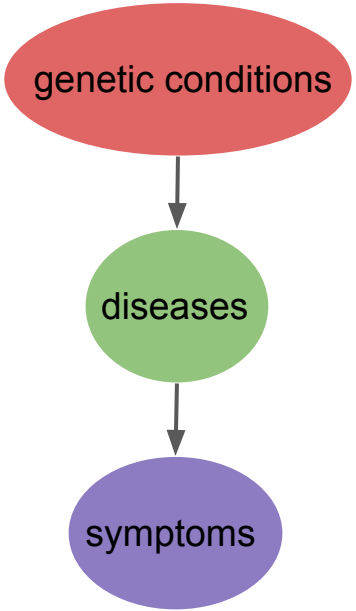


Task #3

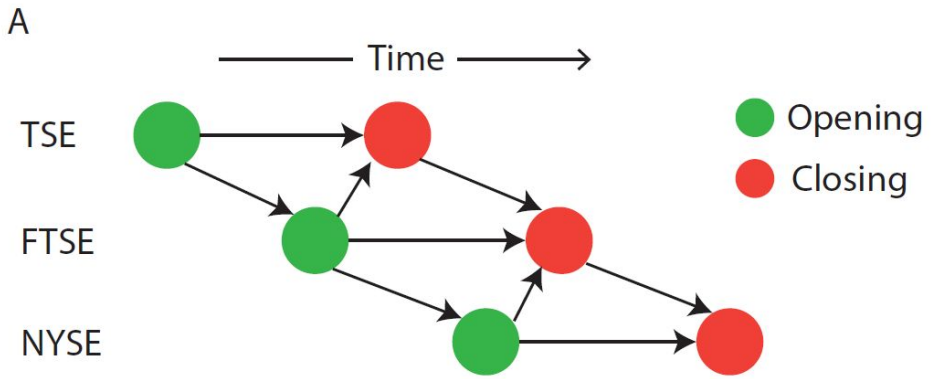


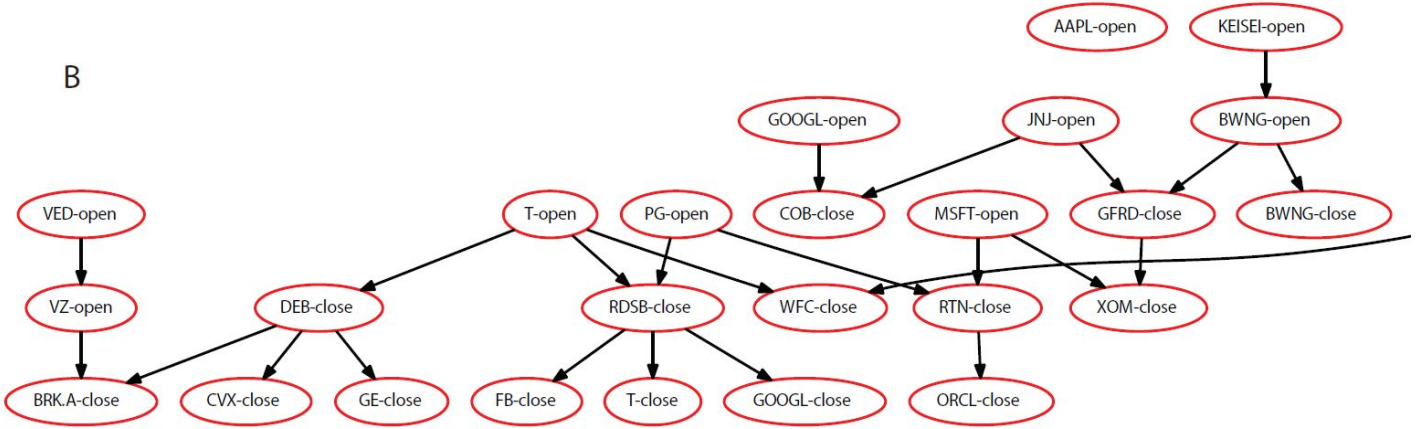
Constraint graphs merge data + knowledge





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

Overview

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

What's next?

Missing value support

Conversion from Cython to numba

Linear Gaussian Bayesian networks

Research in ancestral constraints for Bayesian network structure learning

Paper preprint available on arxiv!

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 now NumFOCUS affiliated

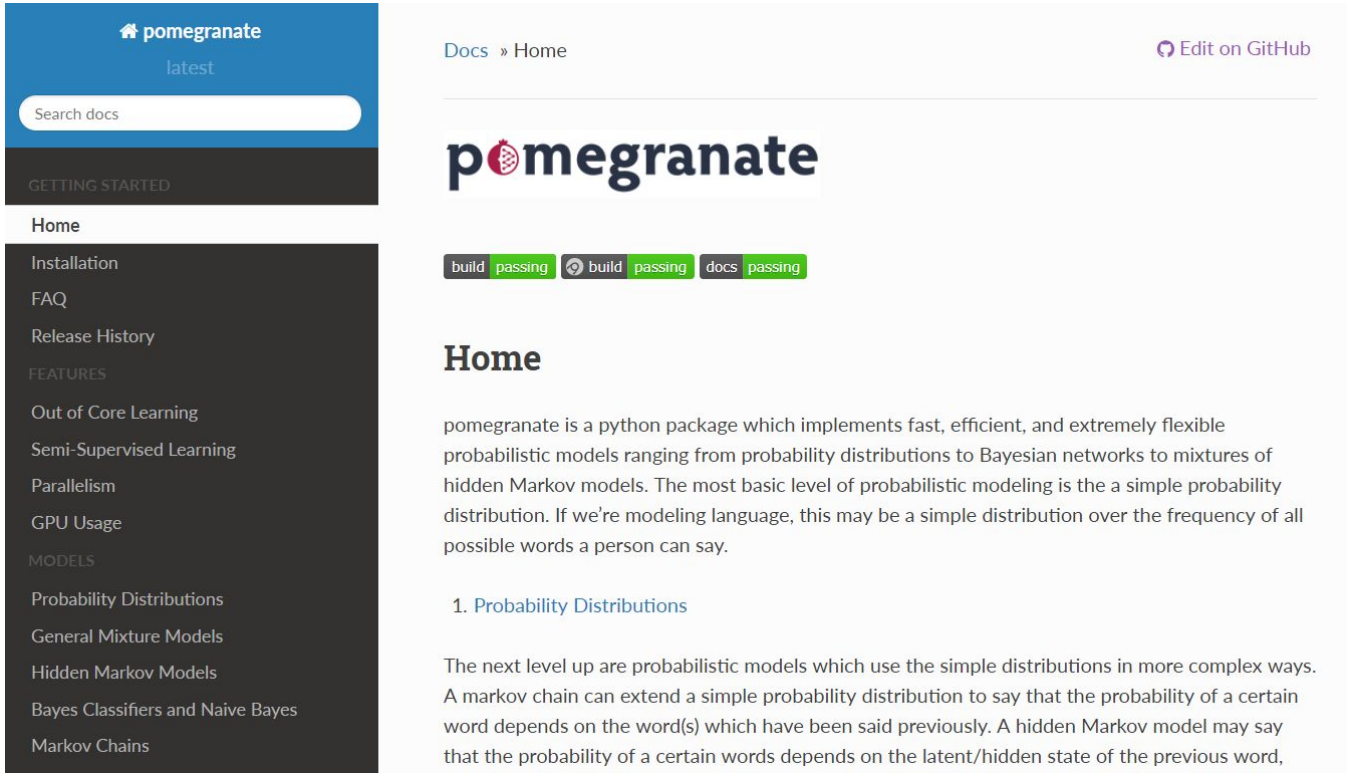
pomegranate

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.

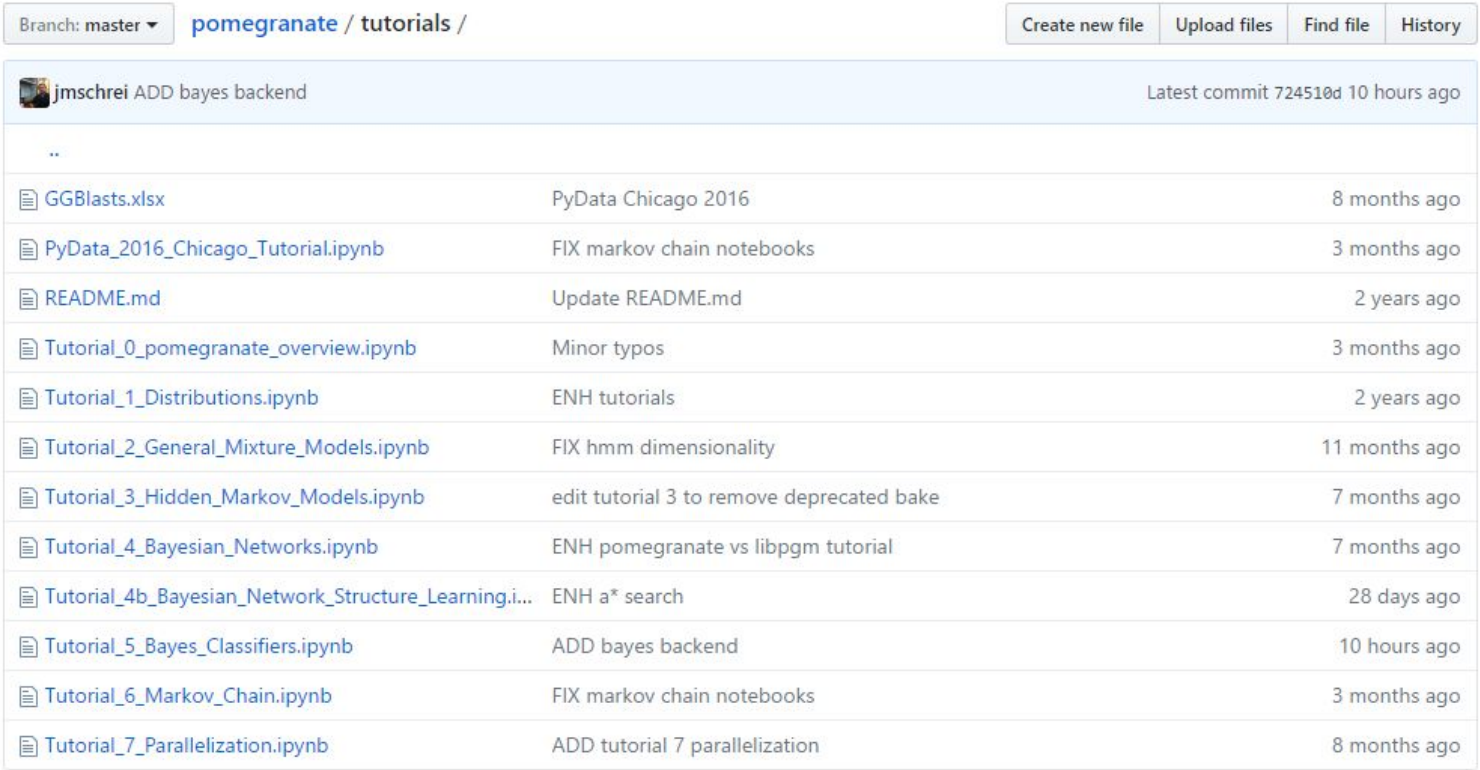
https://www.numfocus.org/open-source-projects/affiliated-projects/

Documentation available at Readthedocs



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

Tutorials available on github



Branch: master ▾ pomegranate / tutorials / Create new file | Upload files | Find file | History

jmschrei ADD bayes backend — Latest commit 724510d 10 hours ago

..		
GGBlasts.xlsx	PyData Chicago 2016	8 months ago
PyData_2016_Chicago_Tutorial.ipynb	FIX markov chain notebooks	3 months ago
README.md	Update README.md	2 years ago
Tutorial_0_pomegranate_overview.ipynb	Minor typos	3 months ago
Tutorial_1_Distributions.ipynb	ENH tutorials	2 years ago
Tutorial_2_General_Mixture_Models.ipynb	FIX hmm dimensionality	11 months ago
Tutorial_3_Hidden_Markov_Models.ipynb	edit tutorial 3 to remove deprecated bake	7 months ago
Tutorial_4_Bayesian_Networks.ipynb	ENH pomegranate vs libpgm tutorial	7 months ago
Tutorial_4b_Bayesian_Network_Structure_Learning.i...	ENH a* search	28 days ago
Tutorial_5_Bayes_Classifiers.ipynb	ADD bayes backend	10 hours ago
Tutorial_6_Markov_Chain.ipynb	FIX markov chain notebooks	3 months ago
Tutorial_7_Parallelization.ipynb	ADD tutorial 7 parallelization	8 months ago

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

pomegranate

fast and flexible probabilistic modelling in python

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

 jmschreiber91

 @jmschrei

 @jmschreiber91

PyMC3, Edward, PyStan?

Pomegranate implements probabilistic models that do not require samplers perform inference with, whereas these packages focus on the implementation of efficient samplers

Model hyperparameters in pomegranate are numbers, whereas they are typically distributions in these other packages. This allows uncertainty in model parameters to be explicitly captured.

Pomegranate focuses on discrete latent state (but discrete/continuous observed state) whereas these focus on continuous latent state