pomegranate

fast and flexible probabilistic modelling in python

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

jmschreiber91
@jmschrei
@jmschreiber91

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

Overview: model stacking in pomegranate

Distributions

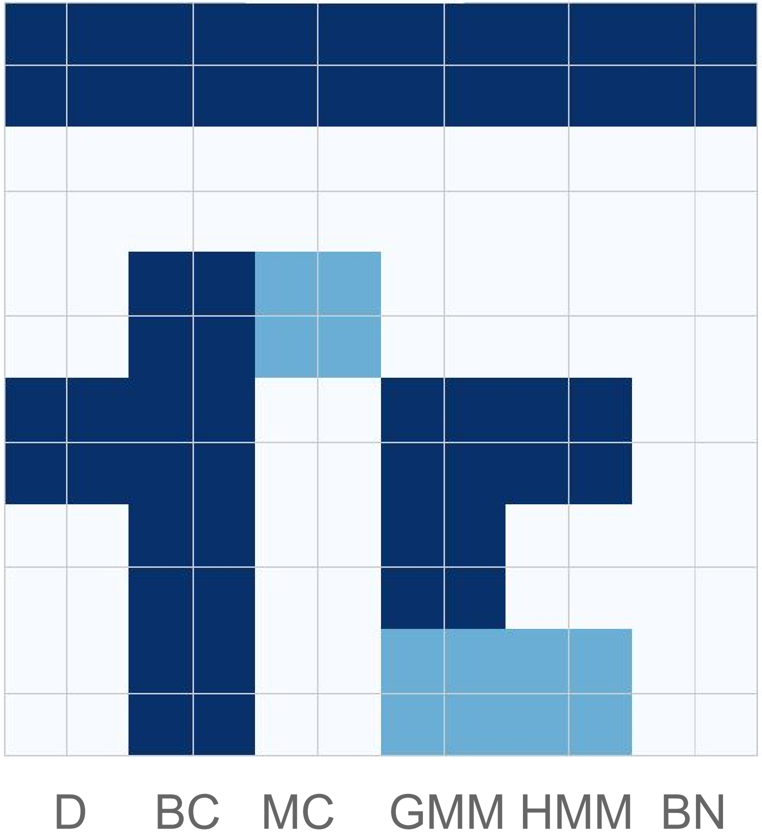
Bayes Classifiers

Markov Chains

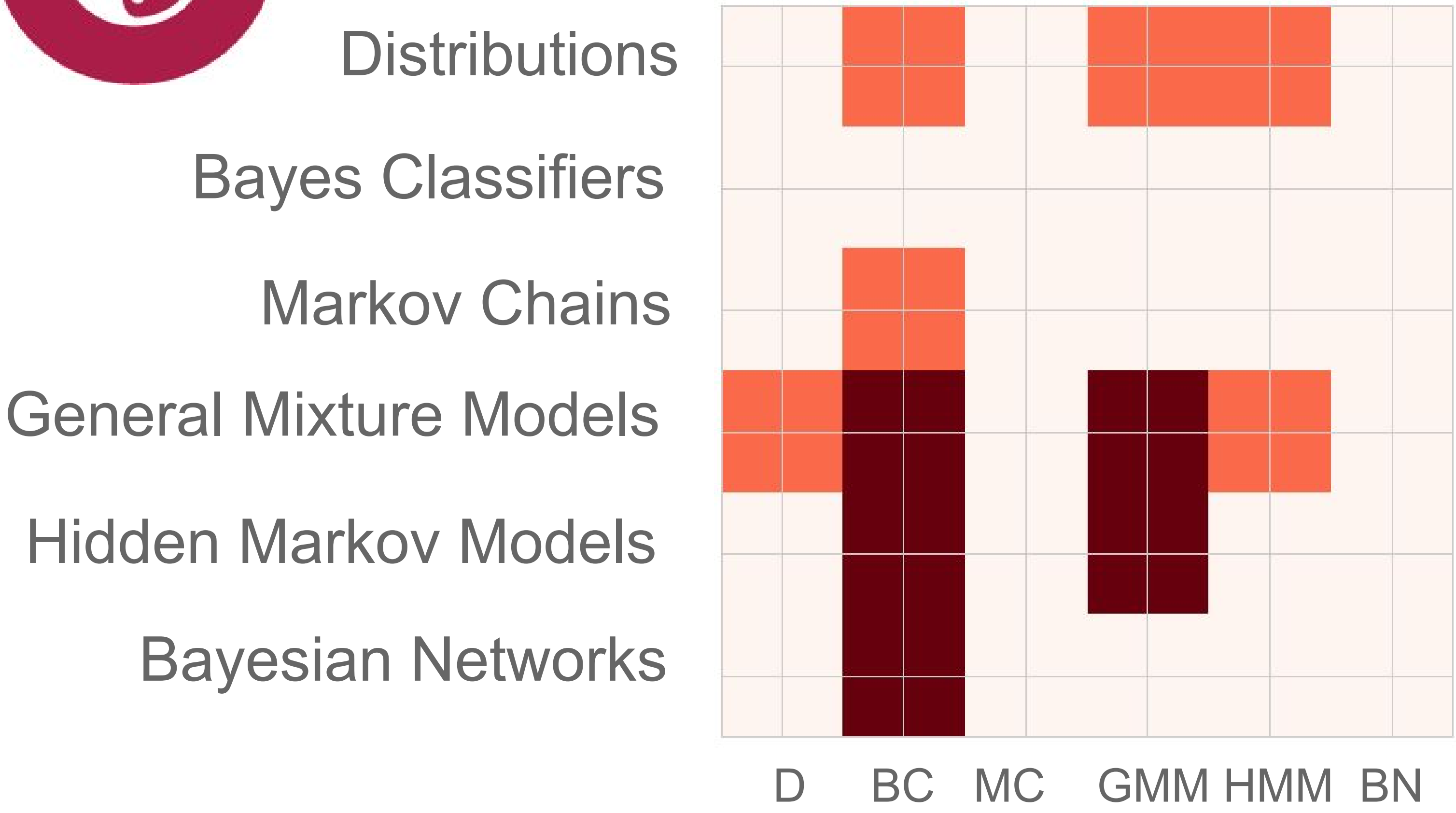
General Mixture Models

Hidden Markov Models

Bayesian Networks



Overview: model stacking in pomegranate



	D	BC	MC	GMM	HMM	BN
Distributions		🟧		🟧	🟧	
Bayes Classifiers						
Markov Chains		🟧				
General Mixture Models	🟧	🟥		🟥	🟧	
Hidden Markov Models		🟥		🟥		
Bayesian Networks		🟥				

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)

All models have these methods!

model.predict(X)

model.predict_proba(X)

model.predict_log_proba(X)

Model.from_samples(X, weights)

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

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

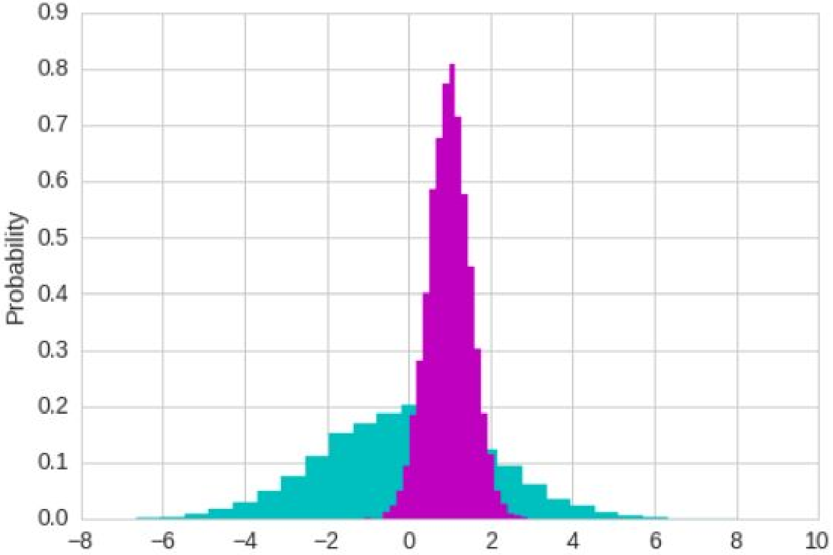
Multivariate Distributions

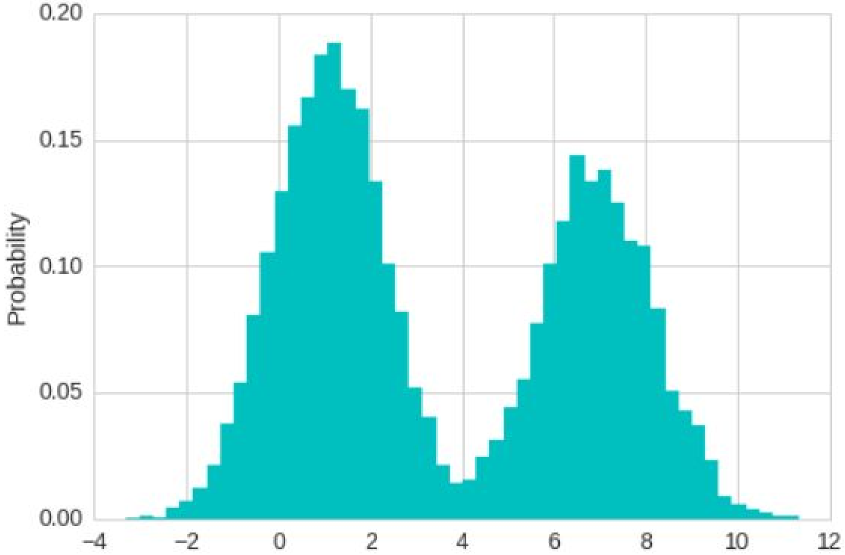
1. IndependentComponentsDistribution
2. MultivariateGaussianDistribution
3. DirichletDistribution
4. ConditionalProbabilityTable
5. JointProbabilityTable

Models can be created from known values

```
mu, sig = 0, 2
a = NormalDistribution(mu, sig)
```

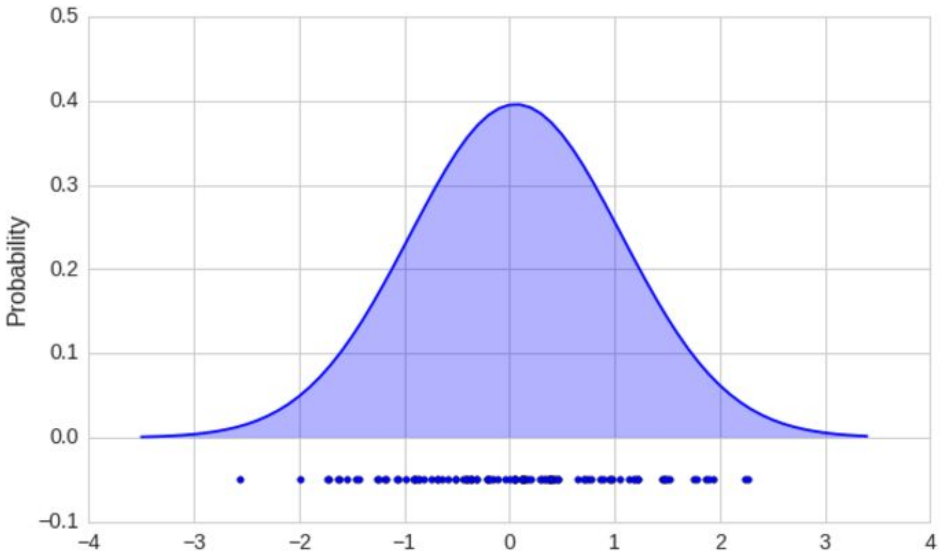
```
X = [0, 1, 1, 2, 1.5, 6, 7, 8, 7]
a = GaussianKernelDensity(X)
```





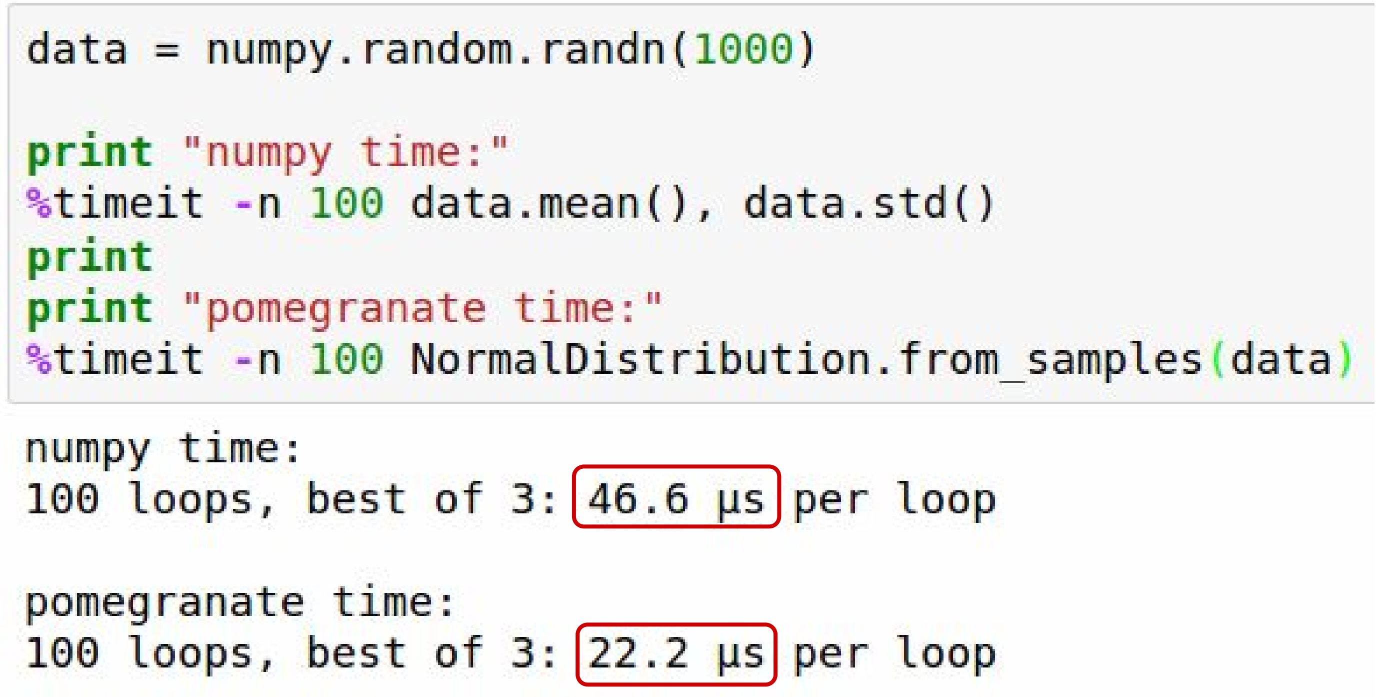
Models can be learned from data

```
X = numpy.random.normal(0, 1, 100)
a = NormalDistribution.from_samples(X)
```



pomegranate can be faster than numpy

Fitting a Normal Distribution to 1,000 samples

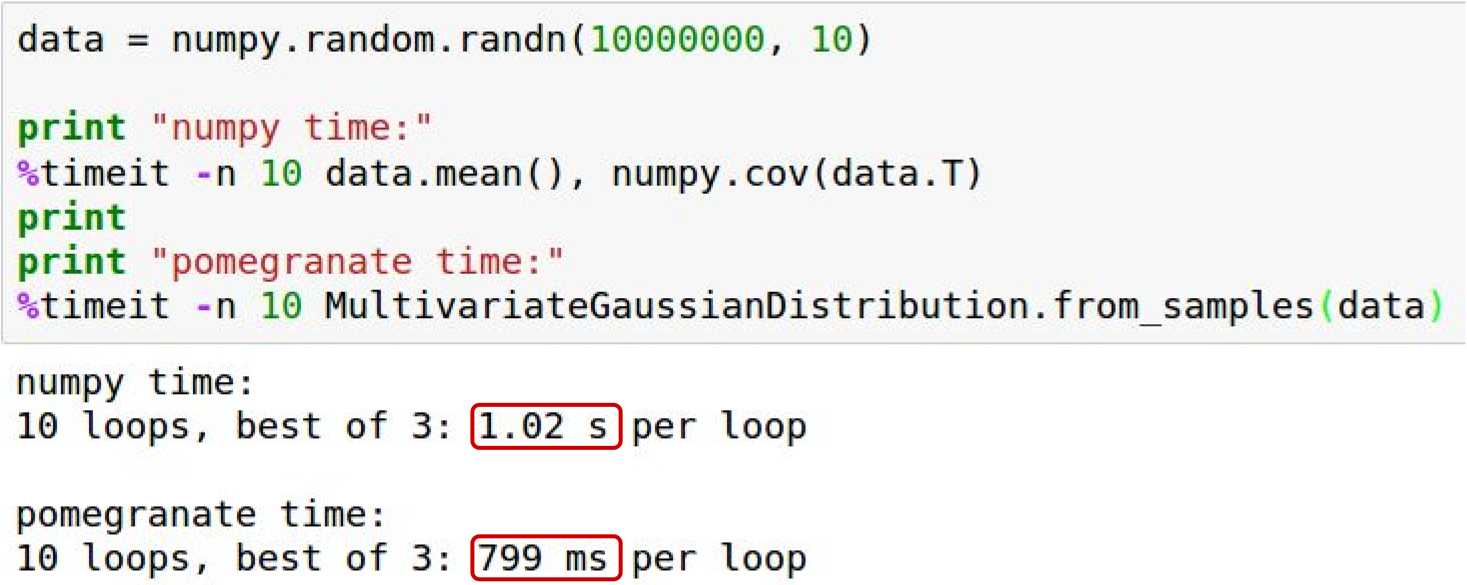


```
data = numpy.random.randn(1000)

print "numpy time:"
%timeit -n 100 data.mean(), data.std()
print
print "pomegranate time:"
%timeit -n 100 NormalDistribution.from_samples(data)
```

```
numpy time:
100 loops, best of 3: 46.6 µs per loop

pomegranate time:
100 loops, best of 3: 22.2 µs per loop
```

pomegranate can be faster than numpy

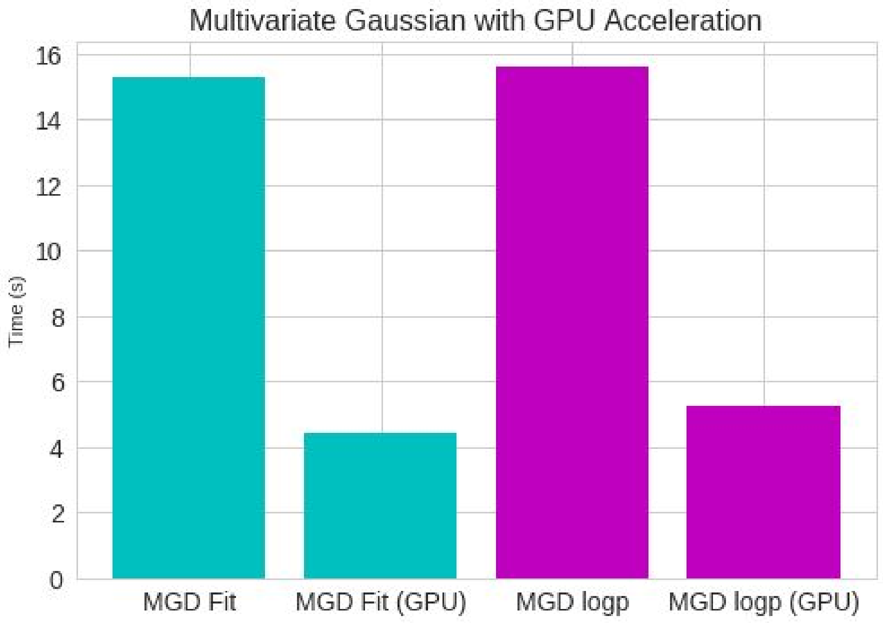
Fitting Multivariate Gaussian to 10,000,000 samples of 10 dimensions

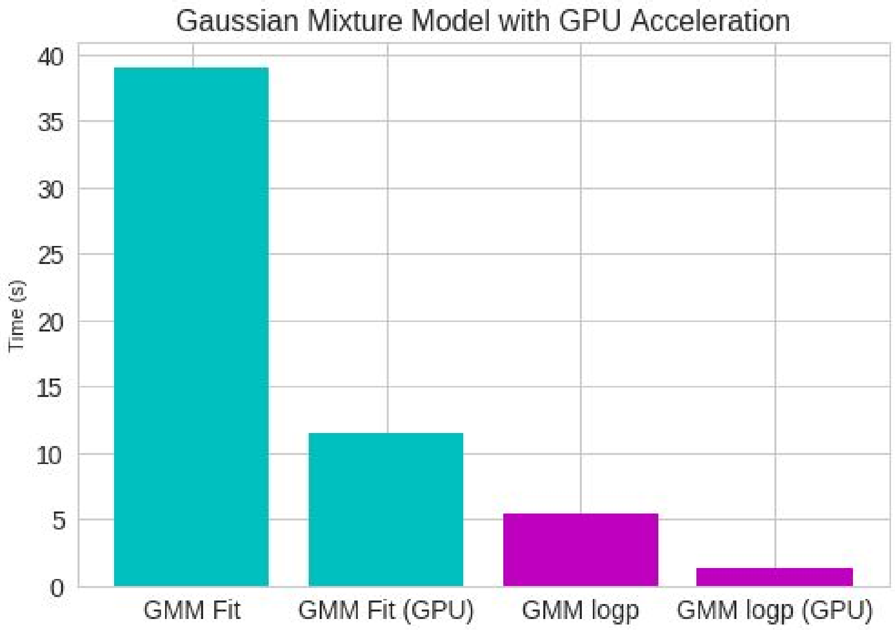


```
data = numpy.random.randn(10000000, 10)

print "numpy time:"
%timeit -n 10 data.mean(), numpy.cov(data.T)
print
print "pomegranate time:"
%timeit -n 10 MultivariateGaussianDistribution.from_samples(data)
```

```
numpy time:
10 loops, best of 3: 1.02 s per loop

pomegranate time:
10 loops, best of 3: 799 ms per loop
```

pomegranate just merged GPU support





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

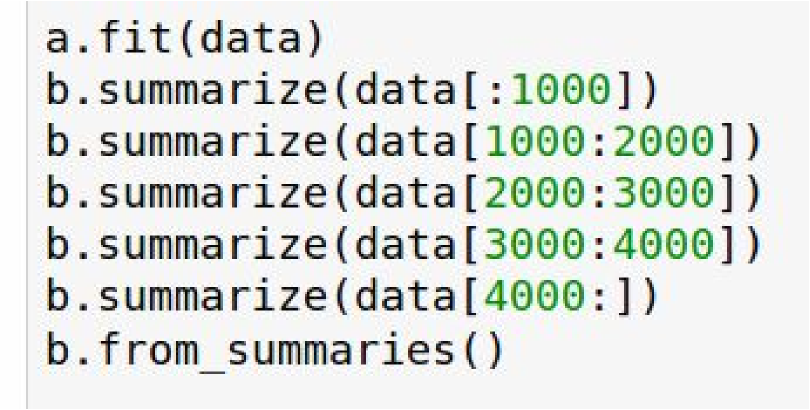
$$\sum_{i=1}^{n} w_i \quad \sum_{i=1}^{n} w_i x_i \quad \sum_{i=1}^{n} w_i x_i^2 \longrightarrow$$

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

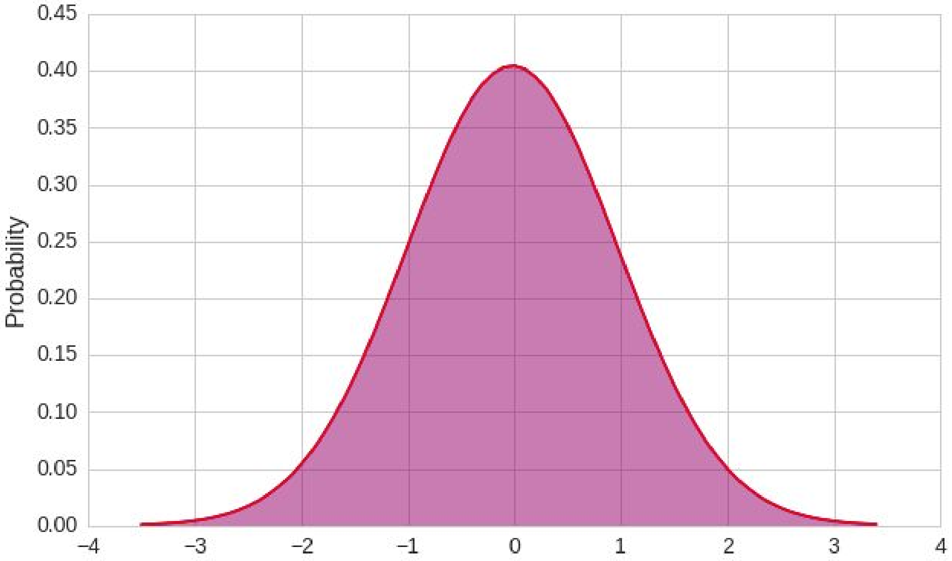
$$\sigma^2 = \frac{\sum_{i=1}^{n} w_i x_i^2}{\sum_{i=1}^{n} w_i} - \frac{\left(\sum_{i=1}^{n} w_i x_i\right)^2}{\left(\sum_{i=1}^{n} w_i\right)^2}$$

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.

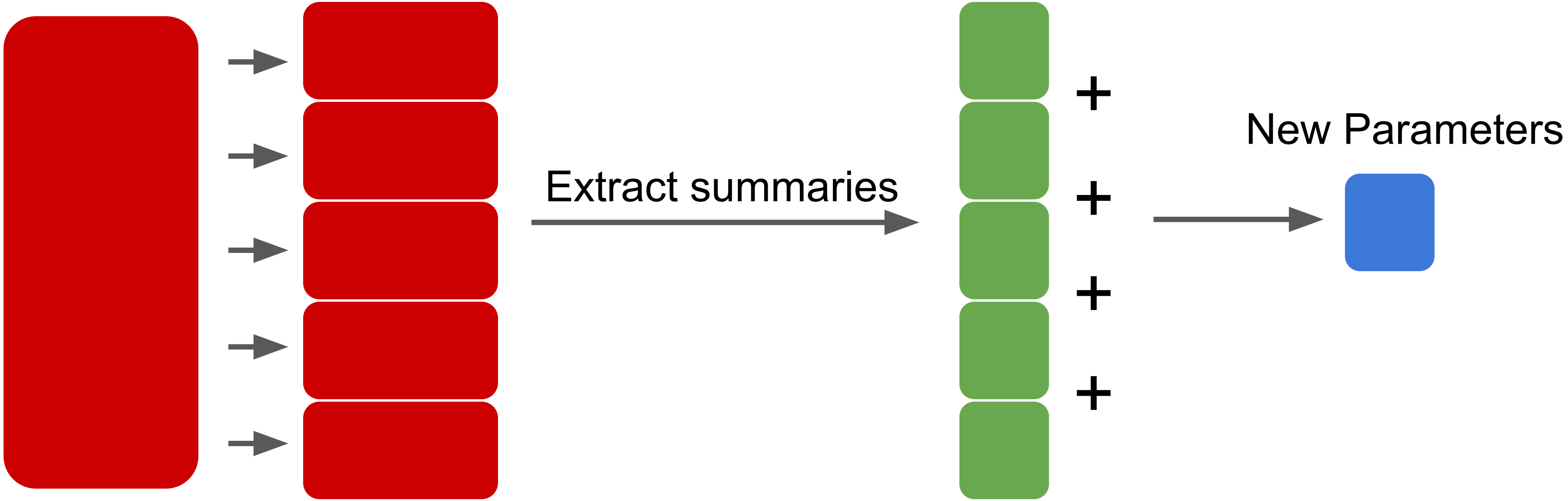


```
a.fit(data)
b.summarize(data[:1000])
b.summarize(data[1000:2000])
b.summarize(data[2000:3000])
b.summarize(data[3000:4000])
b.summarize(data[4000:])
b.from_summaries()
```



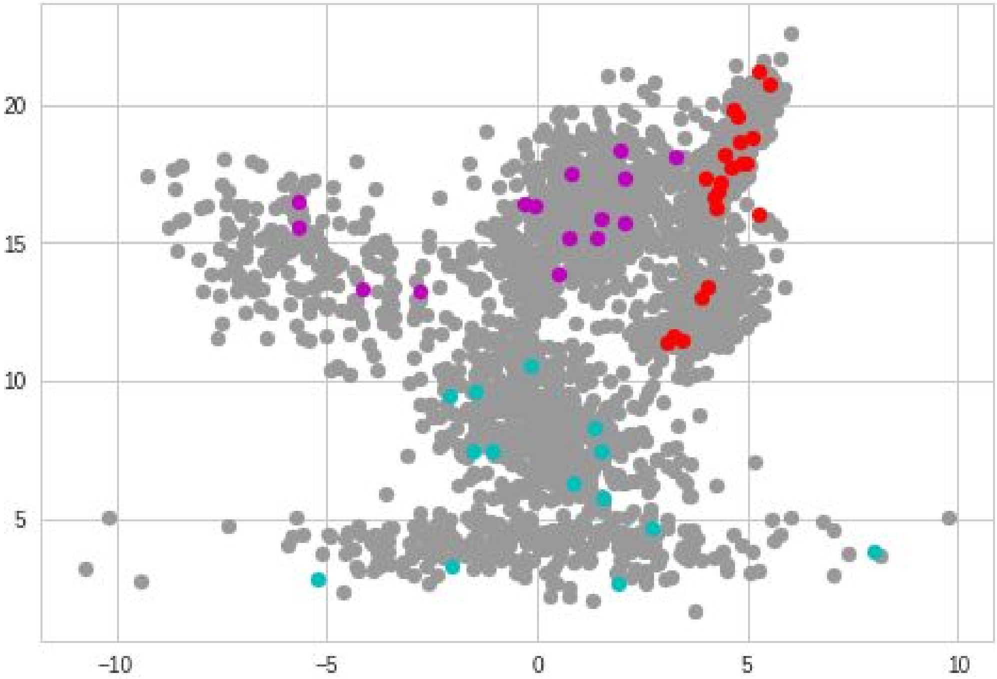
```
Fit Mean:       -0.0174820965846, Fit STD:       0.986767322871
Summarize Mean: -0.0174820965846, Summarize STD: 0.986767322871
```

Parallelization exploits additive summaries



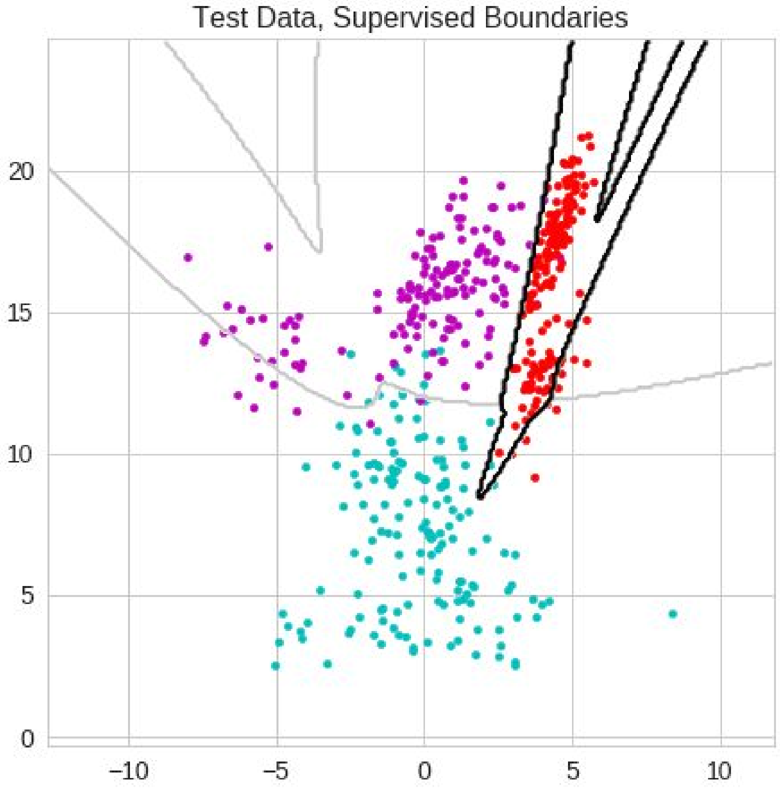
pomegranate supports semisupervised learning

Summary statistics from supervised models can be added to summary statistics from unsupervised models to train a single model on a mixture of labeled and unlabeled data.

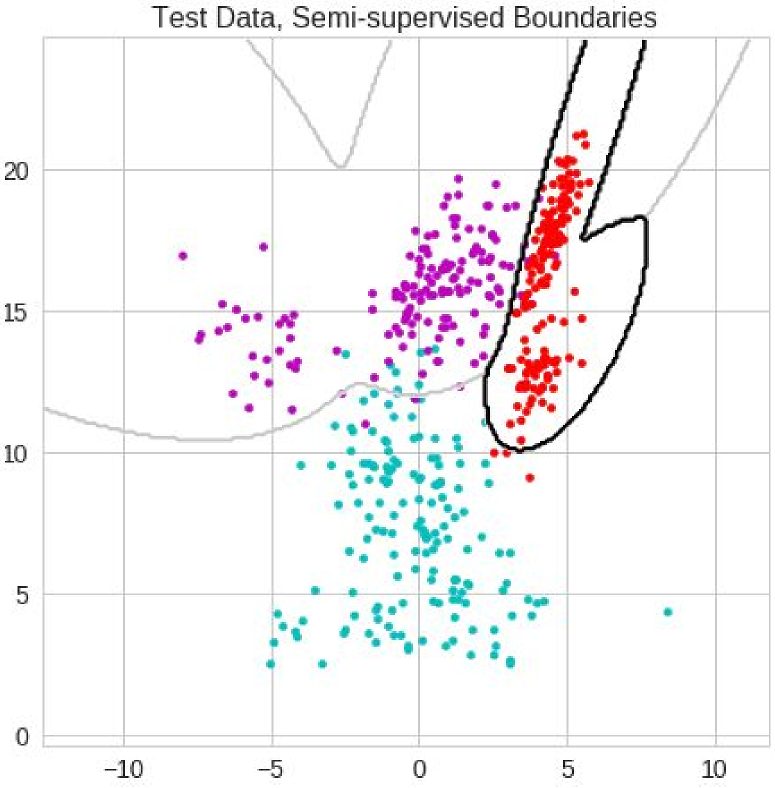


pomegranate supports semisupervised learning

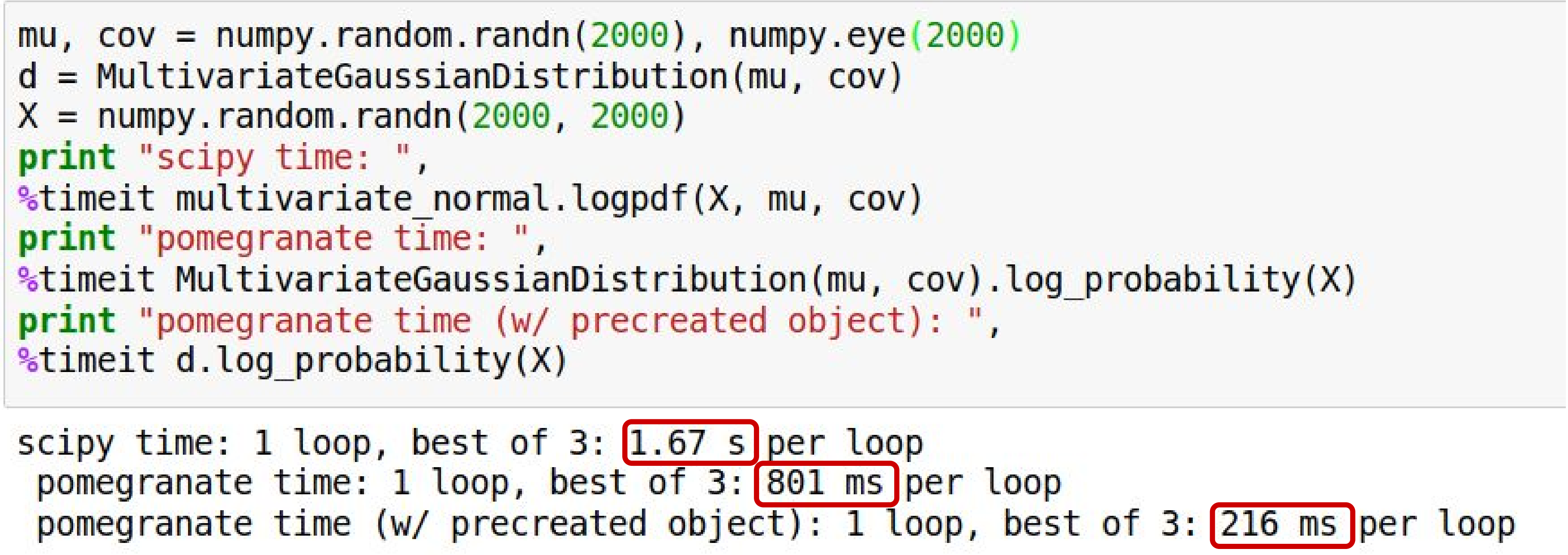
Supervised Accuracy: 0.93


Test Data, Supervised Boundaries

Semisupervised Accuracy: 0.96


Test Data, Semi-supervised Boundaries

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.

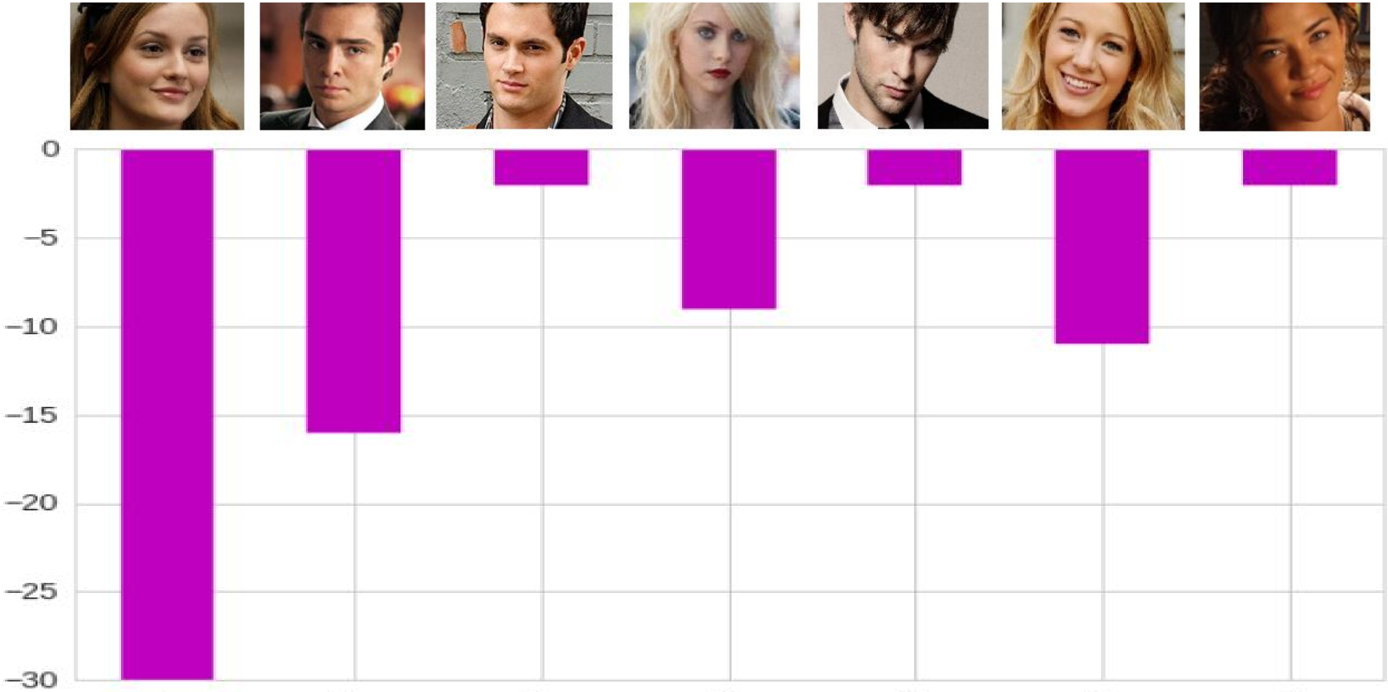
+1 Nate
-1 Blair

How do we encode these ‘blasts’?

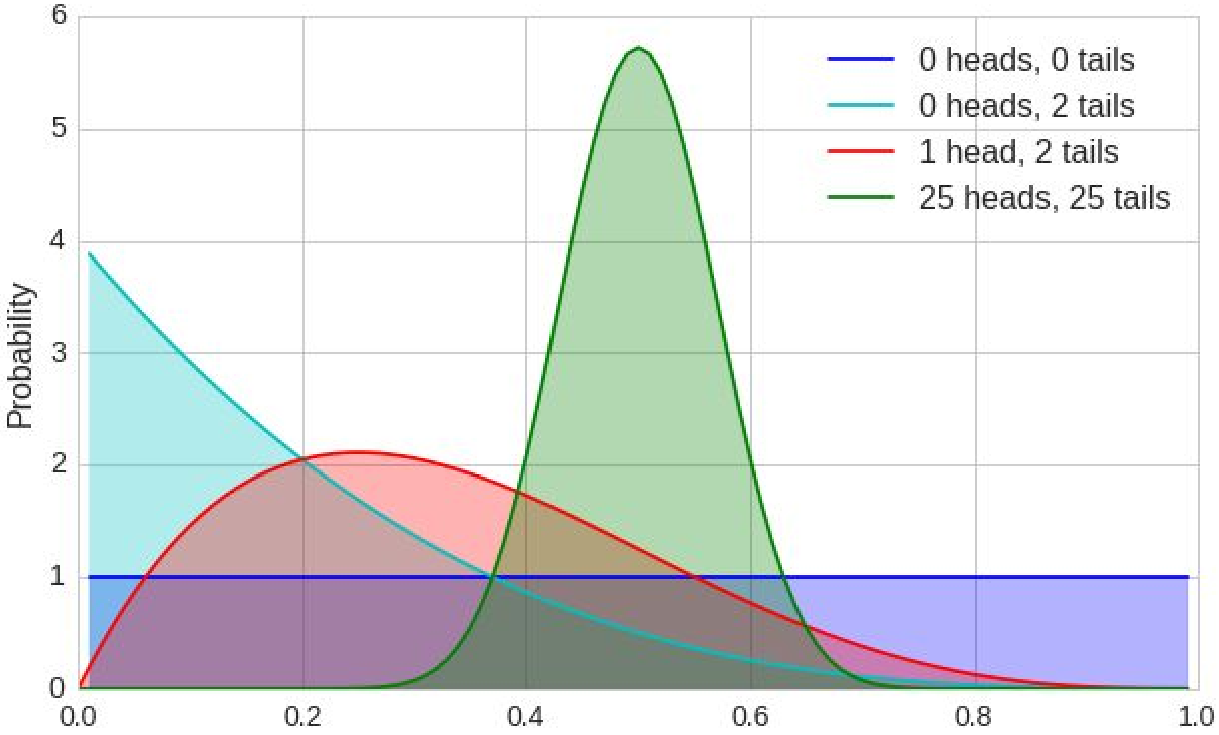
This just in: S and B committing a crime of fashion. Who doesn't love a five-finger discount. Especially if it's the middle one.

-1 Blair
-1 Serena

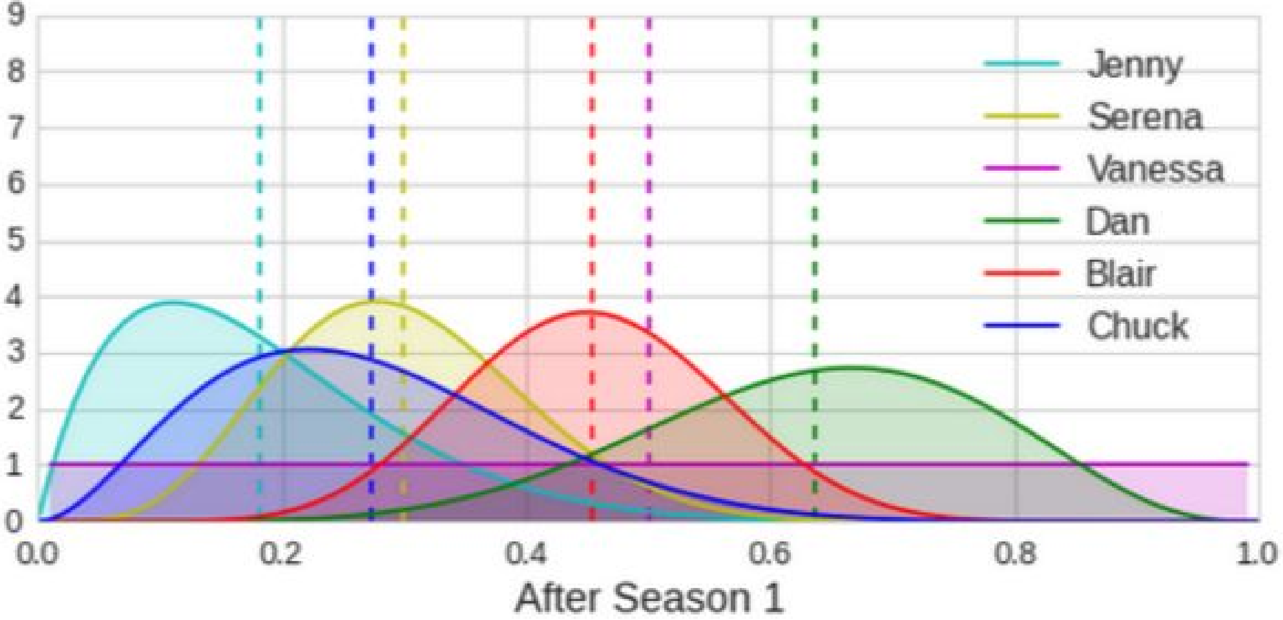
Simple summations don’t work well



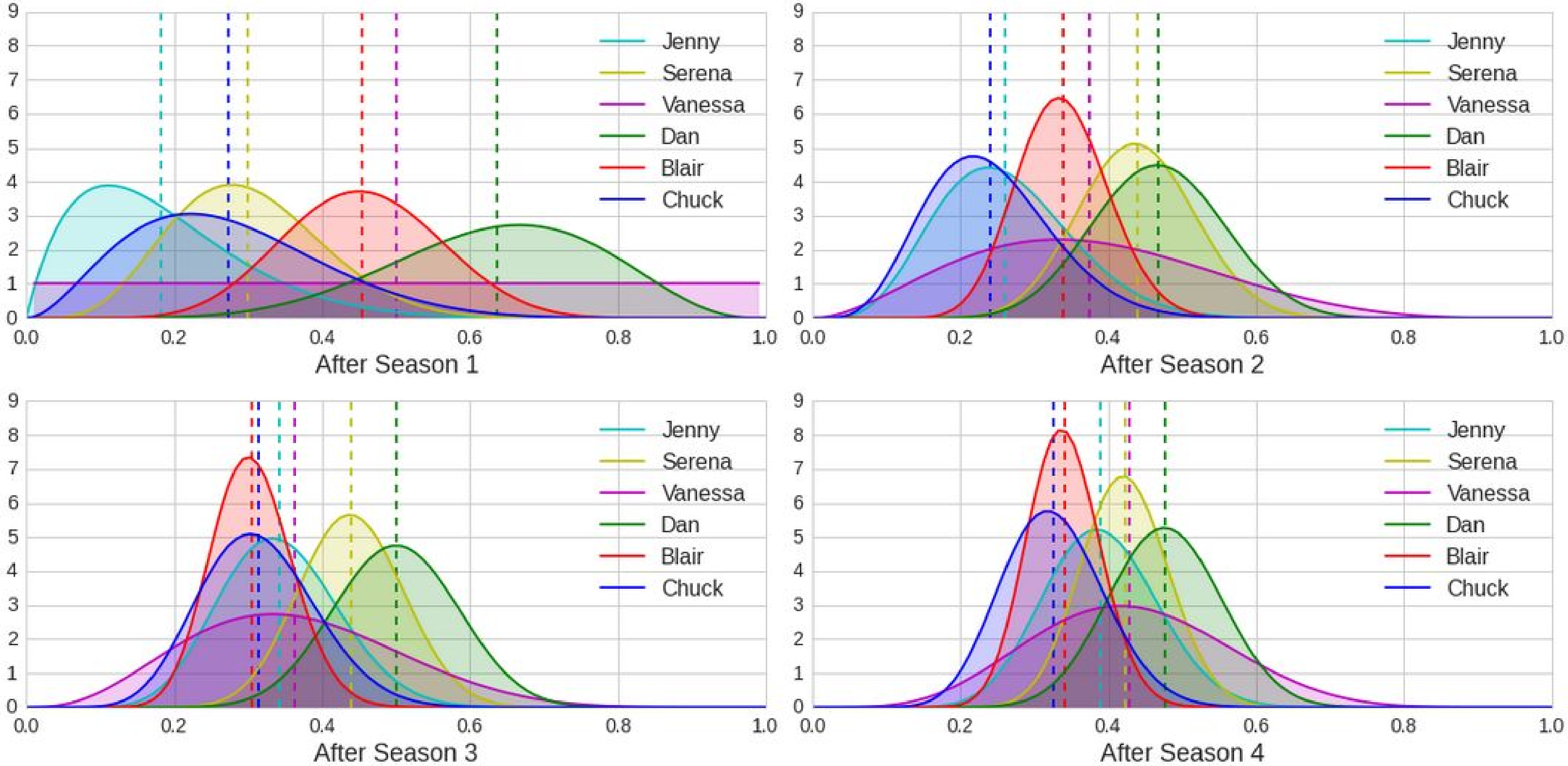
Beta distributions can model uncertainty



Beta distributions can model uncertainty



Beta distributions can model uncertainty

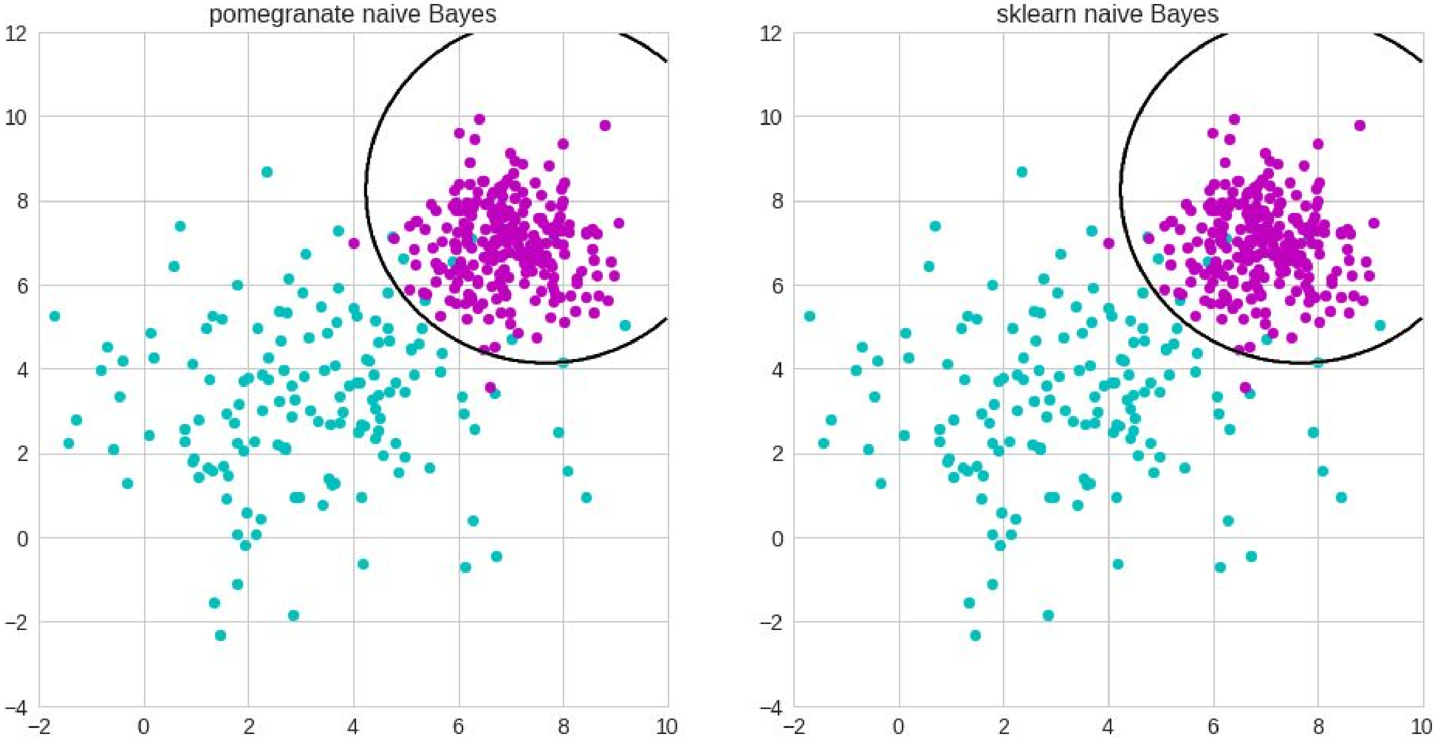


Naive Bayes assumes independent features

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

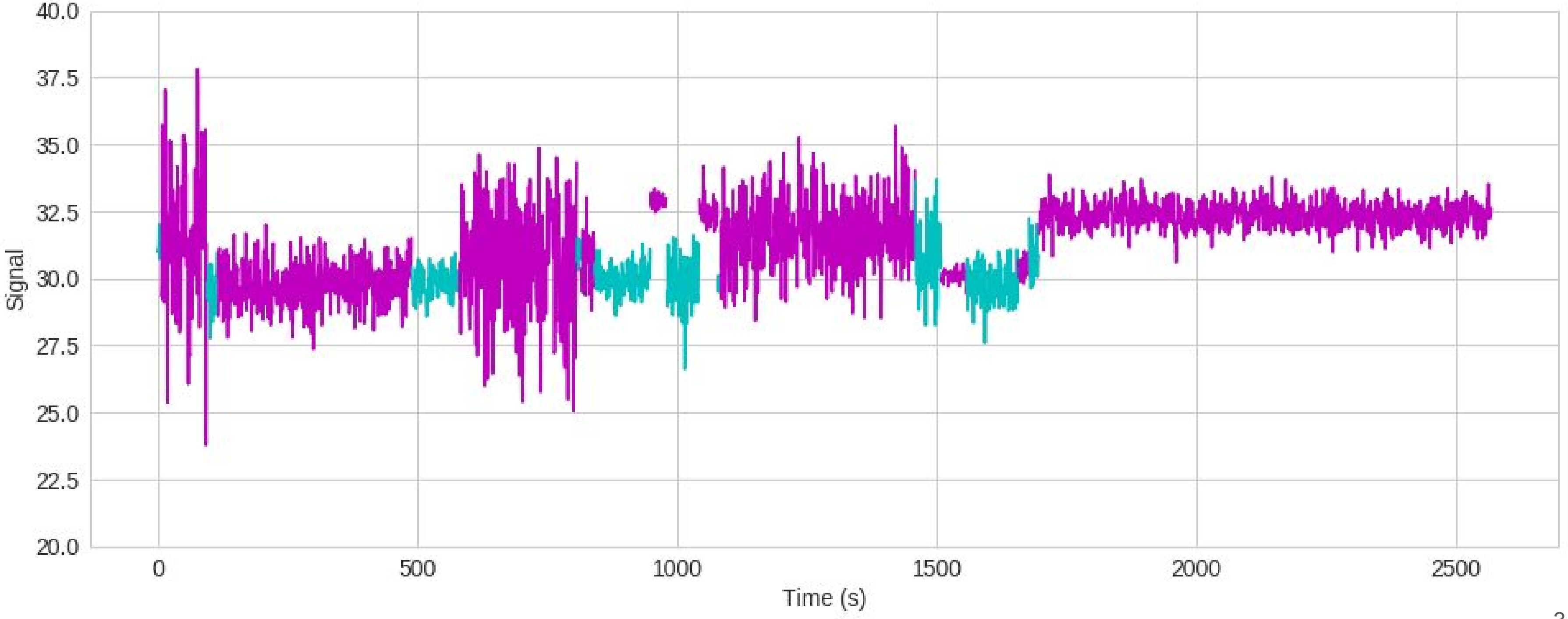
$$Posterior = \frac{Likelihood * Prior}{Normalization}$$

Naive Bayes produces ellipsoid boundaries

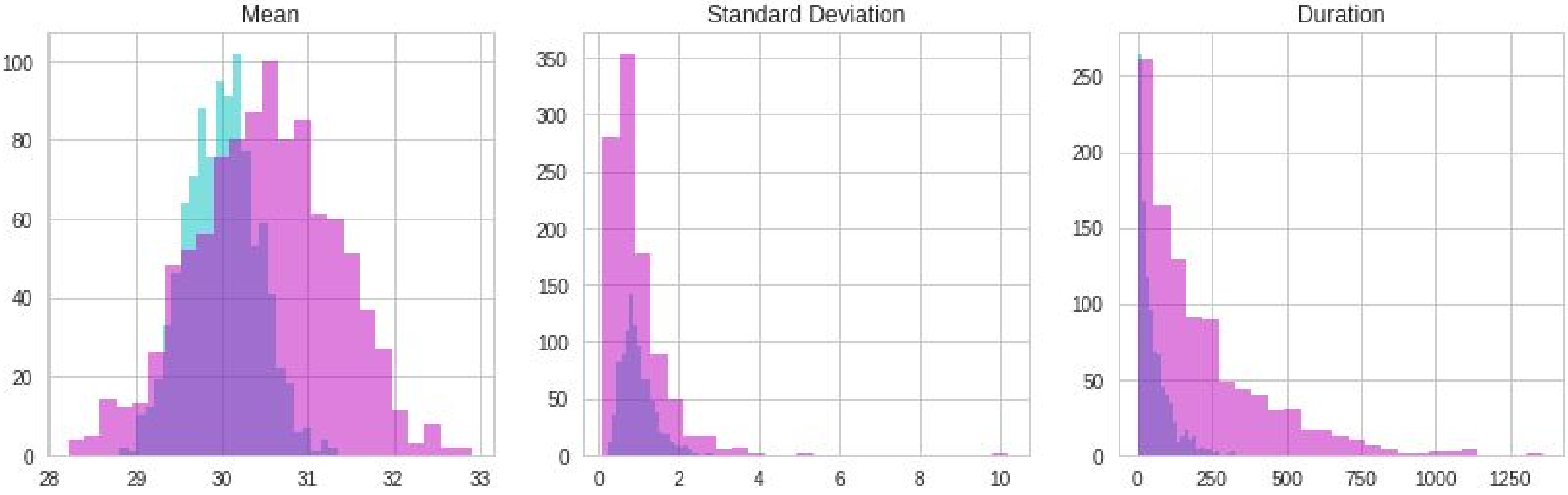


```
model = NaiveBayes.from_samples(NormalDistribution, X, y)
```

Naive Bayes can be heterogeneous



Data can fall under different distributions

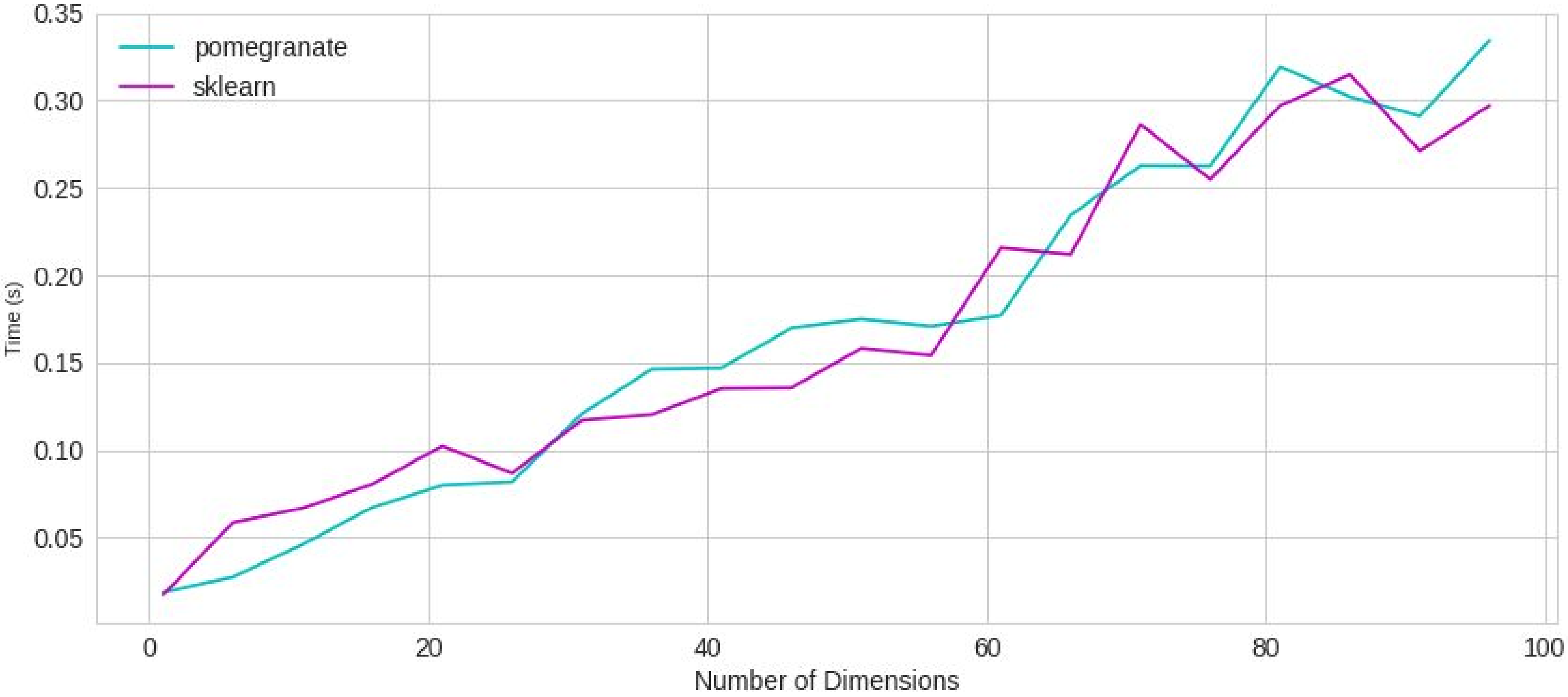


Using appropriate distributions is better

```
model = NaiveBayes.from_samples(NormalDistribution, X_train, y_train)
print "Gaussian Naive Bayes: ", (model.predict(X_test) == y_test).mean()
clf = GaussianNB().fit(X_train, y_train)
print "sklearn Gaussian Naive Bayes: ", (clf.predict(X_test) == y_test).mean()
model = NaiveBayes.from_samples([NormalDistribution, LogNormalDistribution,
ExponentialDistribution], X_train, y_train)
print "Heterogeneous Naive Bayes: ", (model.predict(X_test) == y_test).mean()
```

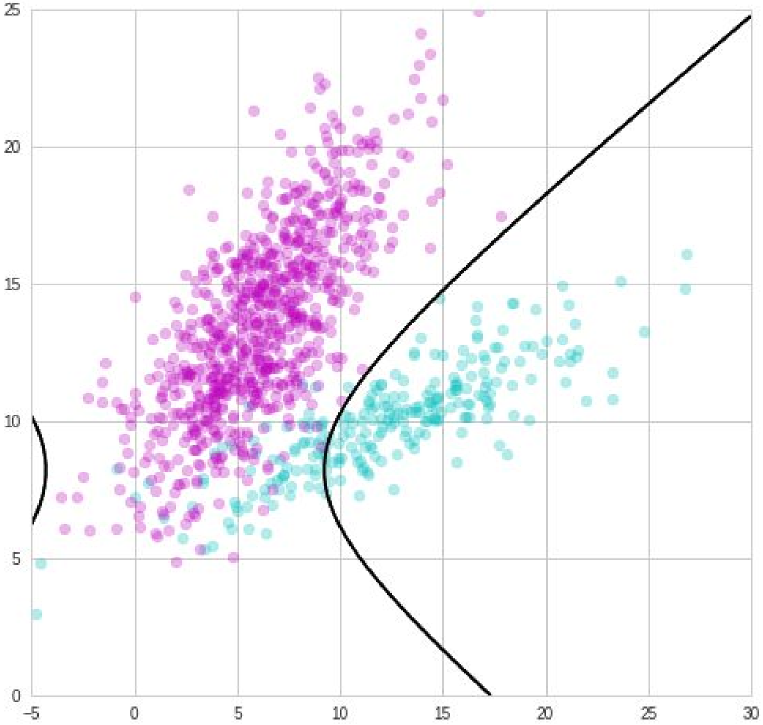
```
Gaussian Naive Bayes:  0.798
sklearn Gaussian Naive Bayes:  0.798
Heterogeneous Naive Bayes:  0.844
```

This additional flexibility is just as fast

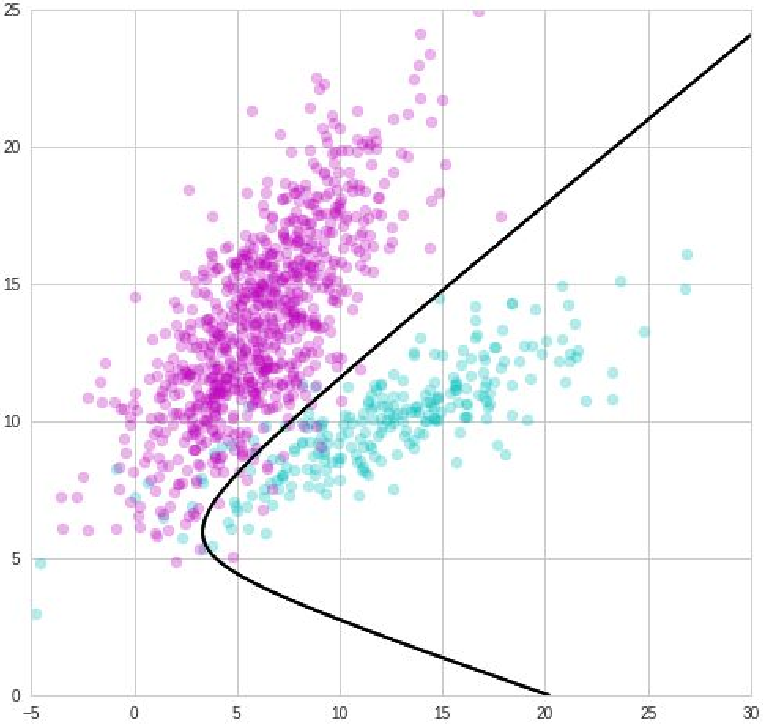


Bayes classifiers don’t require independence

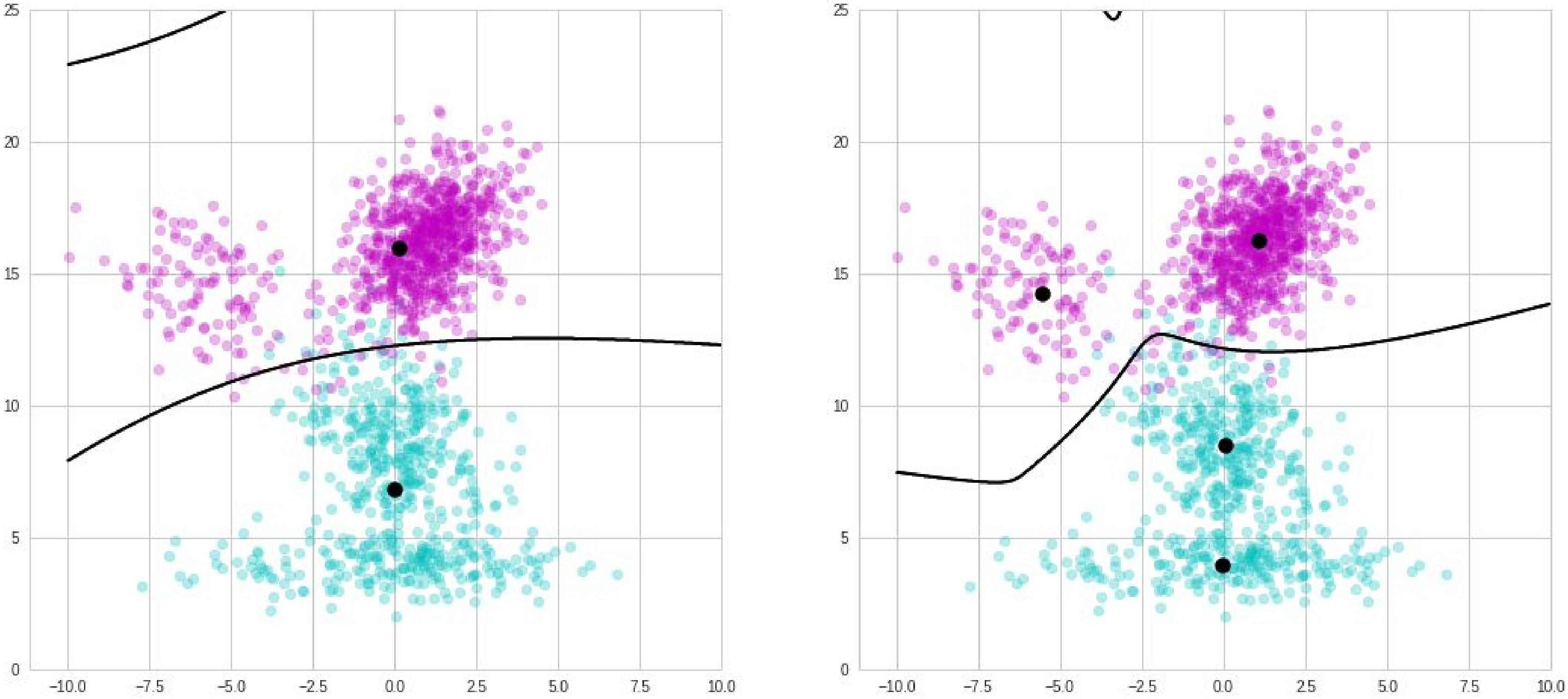
naive accuracy: 0.929



bayes classifier accuracy: 0.966



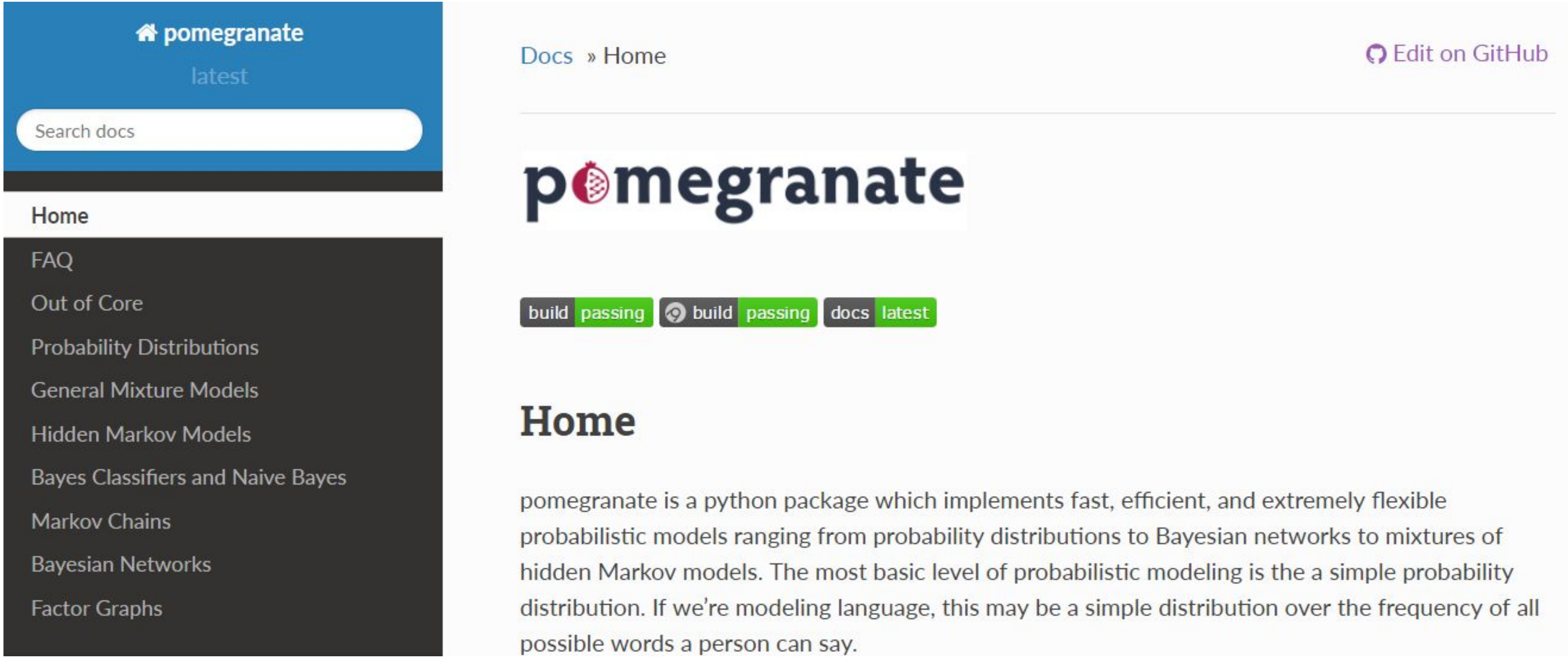
Gaussian mixture model Bayes classifier



Overview

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

Documentation available at Readthedocs



pomegranate
latest
Search docs

- Home
- FAQ
- Out of Core
- Probability Distributions
- General Mixture Models
- Hidden Markov Models
- Bayes Classifiers and Naive Bayes
- Markov Chains
- Bayesian Networks
- Factor Graphs

Docs » Home

Edit on GitHub

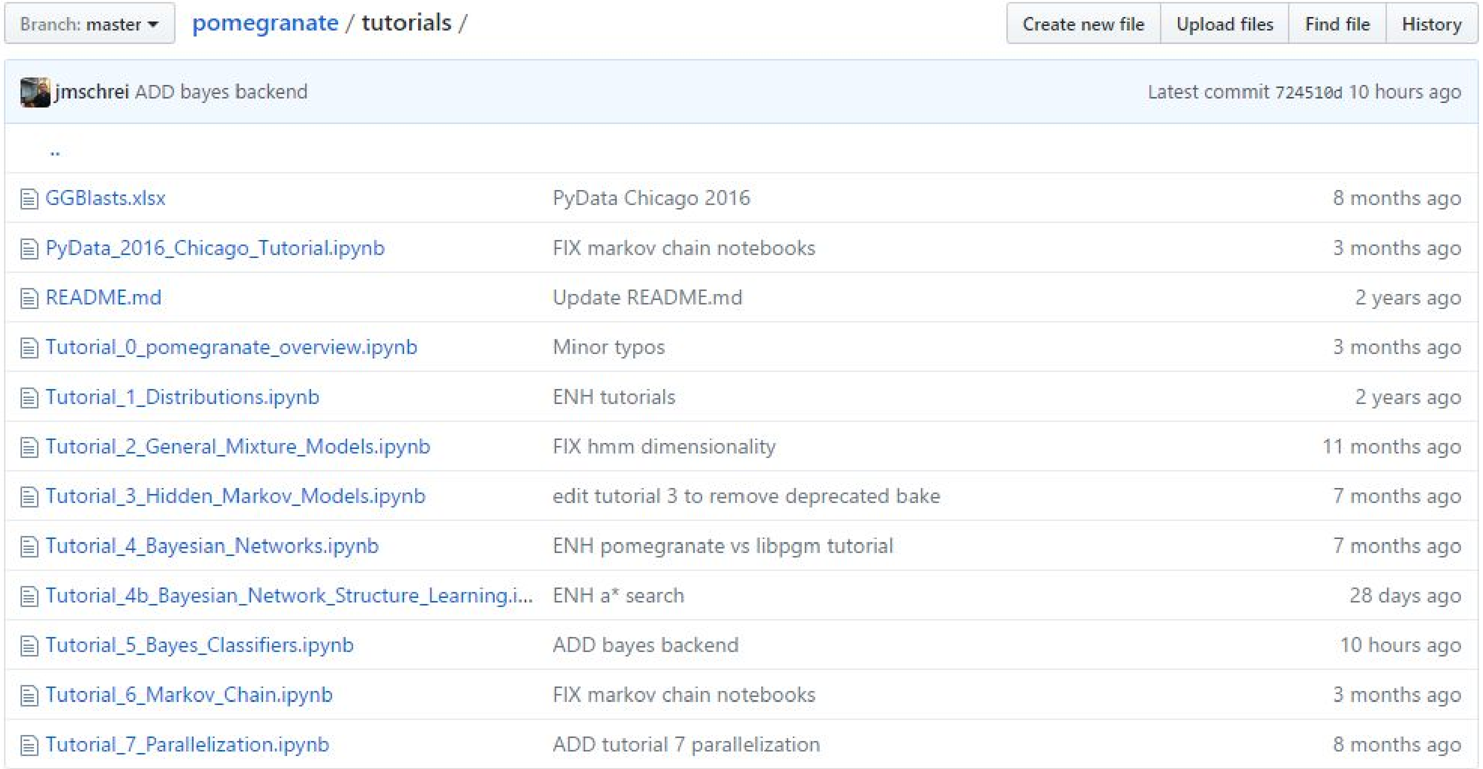
pomegranate

build passing | build passing | docs latest

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.

Tutorials available on github



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

Acknowledgements

Inria

INVENTORS FOR THE DIGITAL WORLD

PARIETAL

UNIVERSITY of WASHINGTON

eScience Institute

ADVANCING DATA-INTENSIVE DISCOVERY IN ALL FIELDS