Bayesian Learning of Classifiers with Stan

Torsten Scholak

twitter @tscholak | github tscholak | blog tscholak.github.io

March 9, 2016

15

http://x.x.x.x:8000

Outline

Classification

Discriminative Approach

Generative Approach

Bayesian Inference in Stan

Demo

http://x.x.x.x:8000

The Classification Problem

Have

D-dimensional, real explanatory variable $\boldsymbol{X} = (X_1, \ldots, X_D)^\intercal$

Class label C, assumes one out of K classes $\{1, \ldots, K\}$

Training set of N observations $(\boldsymbol{x}, c)_{1:N}$

Want

Assign each new observation $\boldsymbol{x}'_{n'}$, $n' = 1, \ldots, N'$, to *one* class

$\boldsymbol{x}'_{n'} \to \boldsymbol{x}'_{n'}, c'_{n'}$

How can we infer the labels $c'_{1:N'}$?

Discriminative Approach

Train a model so as to maximize the probability of getting correct labels

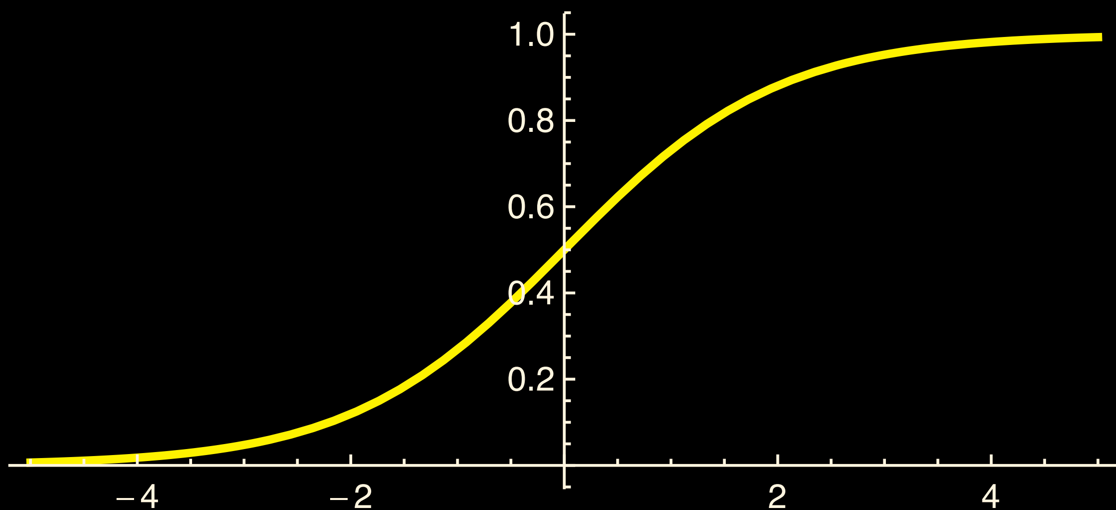
Logistic "Regression"

Logistic regression estimates $p\,(c = k \mid \boldsymbol{x}),\ k = 1, \ldots, K$

Idea: Map the output of a linear model, $b + \boldsymbol{w}^\intercal \boldsymbol{x}$, to a probability

$$p\,(c = k \mid \boldsymbol{x}, \boldsymbol{b}, W) = \mathrm{softmax}_k(\boldsymbol{b} + W\boldsymbol{x}) = \frac{\exp\left(b_k + \boldsymbol{w}_k^\intercal \boldsymbol{x}\right)}{\sum_{k'=1}^{K} \exp\left(b_{k'} + \boldsymbol{w}_{k'}^\intercal \boldsymbol{x}\right)}$$

Softmax is a soft activation function with probability interpretation:



Graph of $y = (1 + \exp(-x))^{-1}$

Training

Maximum Likelihood Estimation

Maximize the conditional log-likelihood of the training data $(\boldsymbol{x}, c)_{1:N}$:

$$\operatorname{argmax}_{\boldsymbol{b},\boldsymbol{W}} \sum_{n=1}^{N} \log p\left(c_n \mid \boldsymbol{x}_n, \boldsymbol{b}, \boldsymbol{W}\right)$$

Maximum A Posteriori Estimation

Add ℓ_2-regularization and maximize:

$$\operatorname{argmax}_{\boldsymbol{b},\boldsymbol{W}} \sum_{n=1}^{N} \log p\left(c_n \mid \boldsymbol{x}_n, \boldsymbol{b}, \boldsymbol{W}\right) - \lambda \sum_{k=1}^{K} \left(|b_k|^2 + \|\boldsymbol{w}_k\|^2\right) =$$

$$\operatorname{argmax}_{\boldsymbol{b},\boldsymbol{W}} \prod_{n=1}^{N} p\left(c_n \mid \boldsymbol{x}_n, \boldsymbol{b}, \boldsymbol{W}\right) \times \prod_{k=1}^{K} \left[\mathrm{Normal}\left(b_k \middle| 0, \tfrac{1}{2\lambda}\right) \mathrm{Normal}\left(\boldsymbol{w}_k \middle| \boldsymbol{0}, \tfrac{1}{2\lambda}\right)\right]$$

Prediction

Discrimination is based on hard boundaries:

$$
\begin{aligned}
c'_{n'} &= \operatorname{argmax}_k p\left(c'_{n'} = k \,\middle|\, \boldsymbol{x}'_{n'}, \boldsymbol{b}, \boldsymbol{W}\right) \\
&= \operatorname{argmax}_k \operatorname{softmax}_k\left(\boldsymbol{b} + \boldsymbol{W}\boldsymbol{x}'_{n'}\right)
\end{aligned}
$$

Problems

How to choose the regularization parameter λ so as to avoid overfitting?

How to incorporate prior knowledge?

How to integrate this method into a larger model?

Isn't discrimination hiding the fact that there are uncertainties in the prediction?

Bayesian Generative Approach

Build a full probabilistic model of all variables, not only the class label

Bayesian Learning

1. Introduce a *generative model* of the data, $\boldsymbol{x}_{1:N}$, conditioned on the class labels $c_{1:N}$ and other latent variables θ:

   $$p\left(\boldsymbol{x}_{1:N} | c_{1:N}, \theta\right)$$

2. Model a *prior probability* for the latent variables:

   $$p\left(c_{1:N}, \theta\right)$$

3. Together, that will give us the *posterior probability*:

   $$p\left(c_{1:N}, \theta | \boldsymbol{x}_{1:N}\right) \propto p\left(\boldsymbol{x}_{1:N} | c_{1:N}, \theta\right) p(c_{1:N}, \theta)$$

   (Bayes theorem)

Why Bayesian Learning?

Resistant to noise, avoids overfitting

Takes into account prior knowledge, better results in smaller samples

More flexibility, straightforward integration into larger model

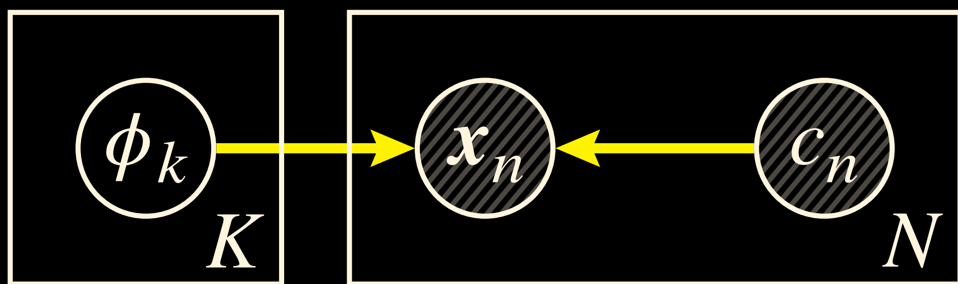
Predictions are probabilistic rather than discriminative

A General Generative Model for Classification

We imagine that

$$\boldsymbol{x}_n \mid c_n, \phi_{1:K} \sim f\left(\phi_{c_n}\right)$$

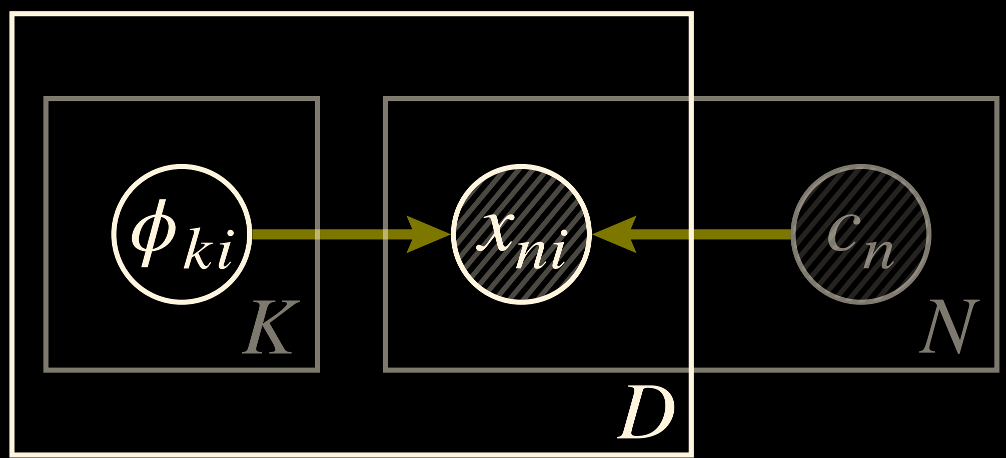
In words: *Given the label c_n, each observed data point $\boldsymbol{x}_n$ is drawn from some distribution with probability density $f\left(\phi_{c_n}\right)$, where the ϕ_k are latent parameters*



A Naïve Assumption

As a simplification, assume that the features factorize:

$$p\left(\boldsymbol{x}_n | c_n, \phi_{1:K}\right) = \prod_{i=1}^{D} f\left(x_{ni} | \phi_{c_n i}\right)$$

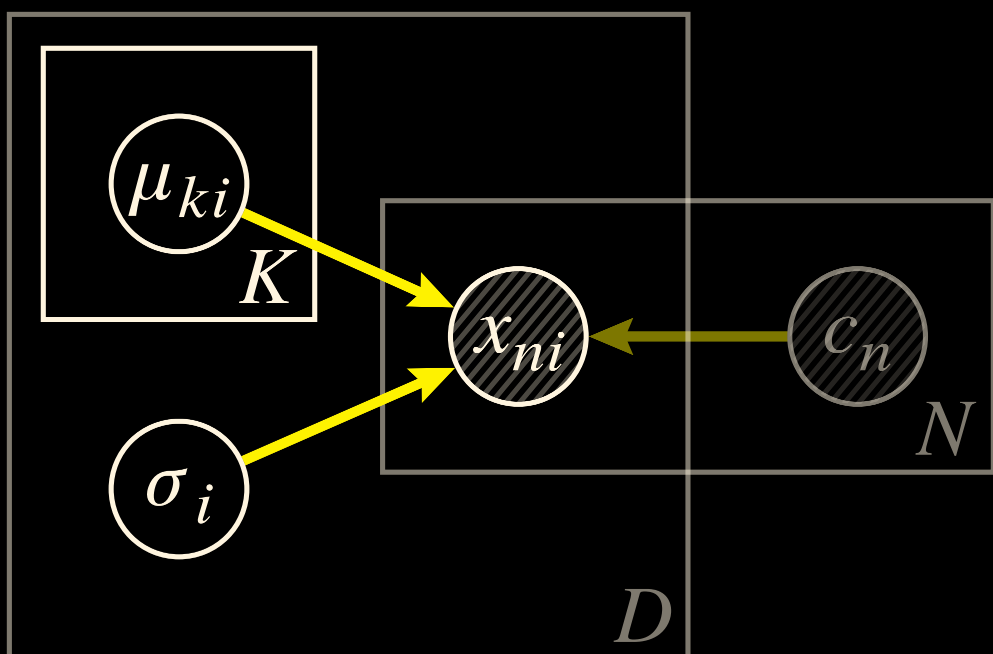


Generate Normally Distributed Samples

For reasons which will become clear soon, choose a normal distribution:

$$\begin{aligned} f\left(x_i \middle| \phi_{ki}\right) &\equiv \mathrm{Normal}\left(x_i \middle| \mu_{ki}, \sigma_i^2\right) \\ &= \frac{1}{\sigma_i \sqrt{2\pi}} \exp\left[-\frac{x_i - \mu_{ki}}{\sigma_i}\right] \end{aligned}$$

with latent means μ_{ki} and variances σ_i^2

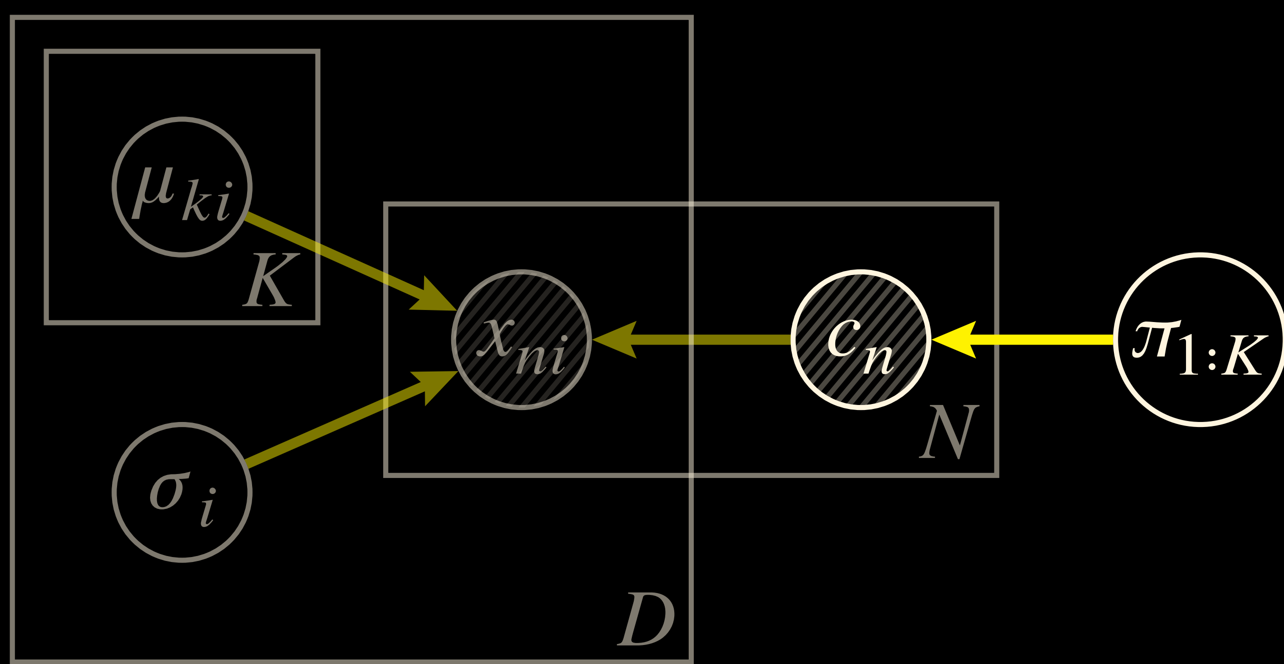


A Class Indicator Prior

We further imagine that

$c_n \mid \pi_{1:K} \sim \text{Categorical}(\pi_{1:K})$

In words: *The indicator variables are distributed according to a categorical distribution defined o event probabilities* π_k *with* $0 \leq \pi_k \leq 1$ *and* $\sum_{k=1}^{K} \pi_k = 1$

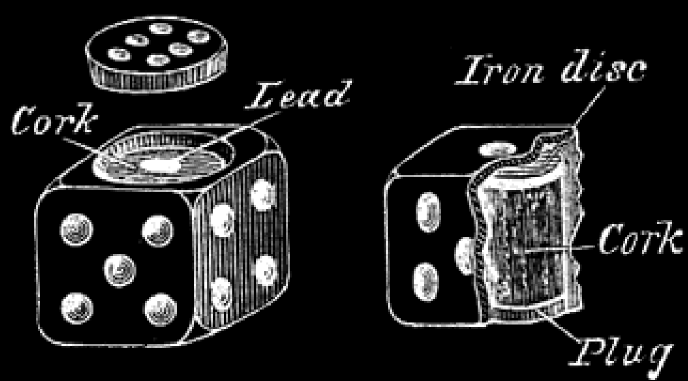


The Categorical Distribution

Probability Mass Function

Loaded K-sided die roll:

$$p\left(c_n = k | \pi_{1:K}\right) = \pi_k$$



The Connection to Logistic Regression

With a naïve Gaussian prior on $\boldsymbol{X}$ and a categorical prior on C, we have:

$$\begin{aligned} p\left(c_n = k \,|\, \boldsymbol{x}_n, \pi_{1:K}, \boldsymbol{\mu}_{1:K}, \boldsymbol{\sigma}\right) &= \frac{\pi_k \;\; \mathrm{Normal}\left(\boldsymbol{x}_n \middle| \boldsymbol{\mu}_k, \mathrm{diag}\left(\boldsymbol{\sigma}^2\right)\right)}{\prod_{k'=1}^{K} \pi_{k'} \;\; \mathrm{Normal}\left(\boldsymbol{x}_n \middle| \boldsymbol{\mu}_{k'}, \mathrm{diag}\left(\boldsymbol{\sigma}^2\right)\right)} \\ &\vdots \\ &= \mathrm{softmax}_k(\boldsymbol{b} + \boldsymbol{W}\boldsymbol{x}_n) \end{aligned}$$

with

$$b_k = \log \pi_k - \sum_{i=1}^{D} \frac{\mu_{ki}^2}{2\sigma_i^2}$$

$$w_{ki} = \frac{\mu_{ki}}{\sigma_i^2}$$

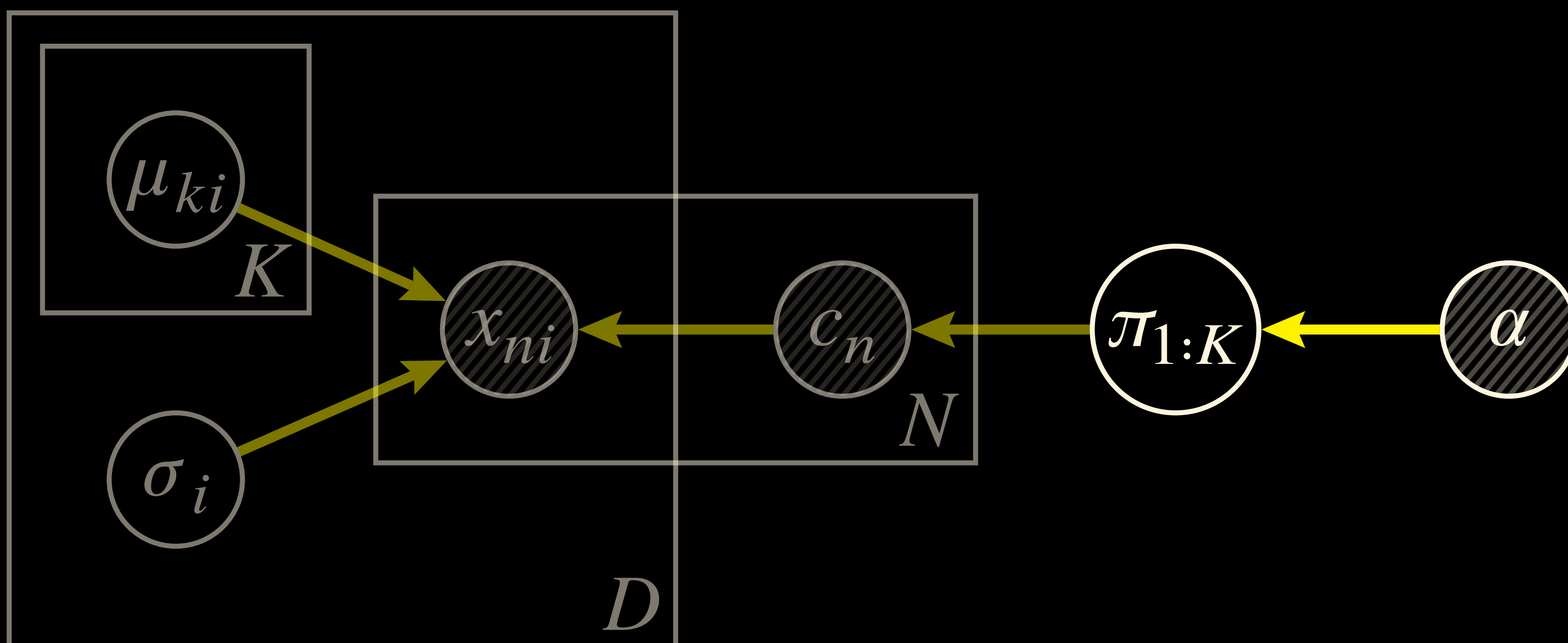
That's the well-known likelihood function of logistic regression!

A Class Prevalence Prior

We choose a *conjugate prior* for the probabilities:

$\pi_{1:K} \mid \alpha \sim \text{Dirichlet}(\alpha)$

In words: *The probabilities π_k are distributed according to a symmetric Dirichlet distribution with concentration parameter $\alpha > 0$*

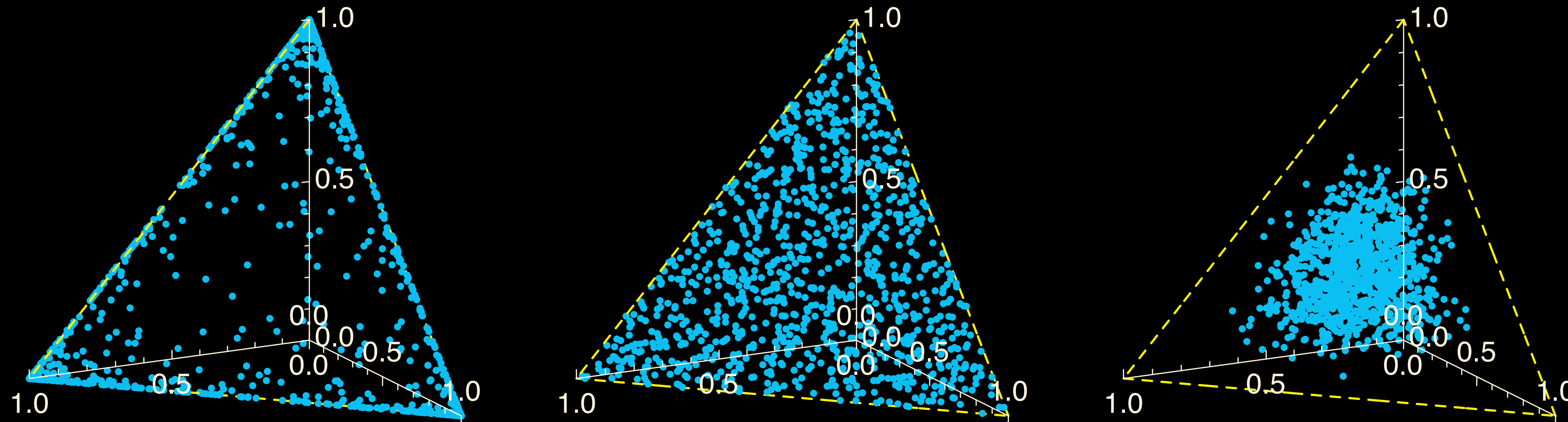


The Dirichlet Distribution I

Probability Density Function

For the symmetric Dirichlet distribution:

$$p\left(\pi_{1:K}|\alpha\right) = \frac{\Gamma\left(K\alpha\right)}{\Gamma(\alpha)^K} \prod_{k=1}^{K} \pi_k^{\alpha-1}$$



If $\alpha \ll 1$, then concentrated around the corners of the simplex

If $\alpha = 1$, then uniformly distributed over the simplex

If $\alpha \gg 1$, then concentrated around the center of the simple

The Dirichlet Distribution II

Pólya's urn

Consider an urn containing balls of K different colors.

Initially, the urn contains each α balls of colors $1, 2, \ldots, K$.

Now perform N draws from the urn, where after each draw, the ball is placed back into the urn with an additional ball of the same color.

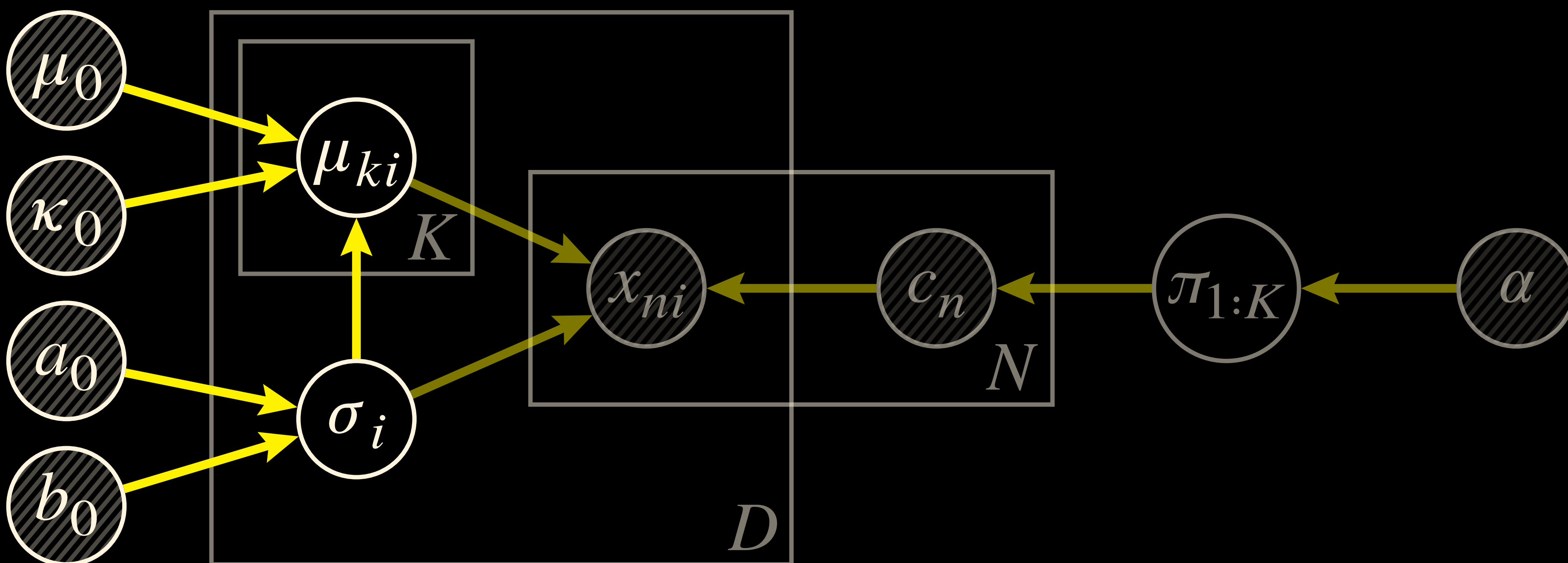
In the limit as $N \to \infty$, the proportions $\pi_{1:K}$ of different colored balls in the urn will be distributed as Dirichlet(α).

Base Measures

Pick distributions for the μ_{ki} and σ_i:

$$\mu_{ki} \mid \mu_0, \kappa_0 \sim \text{Normal}\left(\mu_0, \sigma_i^2/\kappa_0\right)$$
$$\sigma_i^{-2} \mid a_0, b_0 \sim \text{Gamma}(a_0, b_0)$$

with prior mean μ_0, prior sample size κ_0, shape a_0, and rate b_0

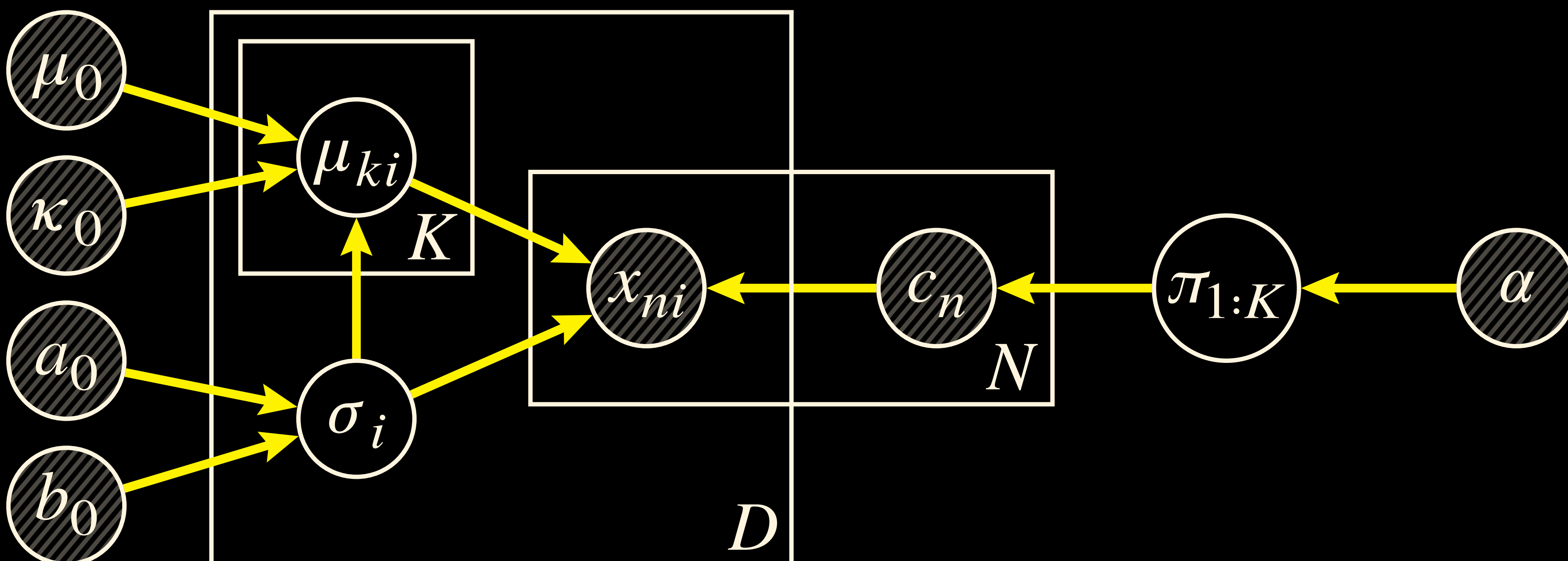


The Posterior of The Complete Model

$$p\left(c_{1:N}, \boldsymbol{\pi}_{1:K}, \boldsymbol{\mu}_{1:K}, \boldsymbol{\sigma} \mid \boldsymbol{x}_{1:N}, \alpha, \mu_0, \kappa_0, a_0, b_0\right)$$

$$\propto \prod_{n=1}^{N} \left[\prod_{i=1}^{D} p\left(x_{ni} \middle| \mu_{c_n i}, \sigma_i^2\right) \right] p\left(c_n \middle| \boldsymbol{\pi}_{1:K}\right)$$

$$\times\, p\left(\boldsymbol{\pi}_{1:K} \middle| \alpha\right) \prod_{i=1}^{D} \left[\prod_{k=1}^{K} p\left(\mu_{ki} \middle| \mu_0, \kappa_0\right) \right] p\left(\sigma_i^{-2} \middle| a_0, b_0\right)$$



Bayesian Inference in Stan

Building, tweaking, enhancing, and enriching models easily without having to think about the implementation

Stan

1. Imperative probabilistic programming language
2. Automatic differentiation for HMC
3. Adaptation routines
4. R, Python, MATLAB, Julia, Stata, and command line interfaces

Program Blocks of A Stan Program

Declare data and parameter variables, define the log-posterior:

```
data { }
transformed data { }
parameters { }
transformed parameters { }
model { }
generated quantities { }
```

The data Block

Reading in information from an external source:

```
data {
    int<lower=1> K;              // number of classes
    int<lower=1> D;              // number of features
    int<lower=0> N;              // number of labelled observations
    int<lower=1,upper=K> c[N];   // classes for labelled observations
    vector[D] x[N];              // features for labelled observations
    vector<lower=0>[K] alpha;    // class concentration
    real mu0;                    // prior mean
    real<lower=0> kappa0;        // prior sample size
    real<lower=0> a0;            // shape
    real<lower=0> b0;            // rate
}
```

The `transformed data` Block

Manipulate the external information once:

```
transformed data { }
```

We won't need it

The `parameters` Block

Define the things we are going to sample from:

```
parameters {
    simplex[K] pi;                    // class prevalence
    vector[D] mu[K];                  // means of features
    vector<lower=0>[D] invsigmasqr;  // inverse variances of features
}
```

The `transformed parameters` Block

Process parameters before computing the posterior:

```
transformed parameters {
    vector<lower=0>[D] sigma;    // scales of features
    for (i in 1:D)
        sigma[i] <- inv_sqrt(invsigmasqr[i]);
}
```

Parameters defined here are not sampled by the Markov chain

The `model` Block

Define the posterior:

```
model {
    pi ~ dirichlet(alpha);       // class prevalence prior
    for (n in 1:N)
        c[n] ~ categorical(pi);  // class indicator prior
    for (k in 1:K)
        mu[k] ~ normal(mu0, sigma/sqrt(kappa0));
    for (i in 1:D)
        invsigmasqr[i] ~ gamma(a0, b0);
                                 // base measures
    for (n in 1:N)
        x[n] ~ normal(mu[c[n]], sigma);
                                 // generative model
}
```

The `generated quantities` Block

Produce random samples, e.g. to validate the model with pseudo-data:

```
generated quantities {
    vector[K] z[Np];
    vector[K] sm[Np];
    int<lower=1,upper=K> cp[Np];
    for (np in 1:Np) {
        for (k in 1:K)
            z[np, k] <- normal_log(xp[np], mu[k], sigma) \
                + categorical_log(k, pi);
        sm[np] <- softmax(z[np]);
        cp[np] <- categorical_rng(sm[np]);
    }
}
```

Demo

Further Content Ingestion

Stan User Guide and Reference Manual v2.9.0

- Tons of example programs
- Introduction to Bayesian statistics
- Language reference
- Discussion of HMC and NUTS algorithms

Example program repository

Michael Betancourt's YouTube videos

Andrew Gelman's book