# CSE 5523: Lecture Notes 4
Bayesian Statistics

## Contents

## 4.1 Bayes' rule

An application of the definition of conditional probability is called **Bayes' rule**

(assuming probability space $\langle P{\times}X, 2^{P\times X}, \mathsf{P}\rangle$, with space of hypotheses $P$ and data $X$):

$$\begin{aligned}
\mathsf{P}(p \mid x) &= \frac{\mathsf{P}(p, x)}{\mathsf{P}(x)} && \text{definition of conditional probability} \\
&= \frac{\mathsf{P}(p) \cdot \mathsf{P}(x \mid p)}{\mathsf{P}(x)} && \text{chain rule decomposition} \\
&\propto \mathsf{P}(p) \cdot \mathsf{P}(x \mid p) && \text{(proportion, ignoring } \mathsf{P}(x)\text{, assuming simple normalization)}
\end{aligned}$$

This comes up often enough in statistics that its components have names:

1. $\mathsf{P}(p)$ is called the **prior**,
2. $\mathsf{P}(x \mid p)$ is called the **likelihood**,
3. $\mathsf{P}(p \mid x)$ is called the **posterior**.

## 4.2 Dirichlet priors and smoothing

We can apply a prior to the multinomial distribution to solve problems with zero counts:

$$\text{Dirichlet}_{h_1,\ldots,h_J}(p_1, \ldots, p_J) = \underbrace{\frac{\Gamma(\sum_j h_j)}{\prod_j \Gamma(h_j)}}_{\text{normalization constant with no } p_j \text{ terms}} \cdot \underbrace{\prod_j (p_j)^{h_j - 1}}_{\text{probability of pseudocounts}}$$

This is called a **Dirichlet** prior. It is the probability of drawing $h_1{-}1, ..., h_J{-}1$ counts from $p_1, ..., p_J$.

It is based on the gamma function (which generalizes the factorial function to reals):

$$\Gamma(n) = \int x^n \, e^{-x} \, dx$$

Dirichlets are **conjugate** to multinomials: substitution as prior, likelihood gives Dirichlet posterior:

$$\begin{aligned}
\mathsf{P}(p_1,\ldots,p_J \,|\, \mathcal{D}) &= \frac{\mathsf{P}(p_1,\ldots,p_J)\cdot\mathsf{P}(\mathcal{D}\,|\,p_1,\ldots,p_J)}{\mathsf{P}(\mathcal{D})} && \text{Bayes' rule}\\
&= \frac{\mathsf{P}(p_1,\ldots,p_J)\cdot\mathsf{P}(\mathcal{D}\,|\,p_1,\ldots,p_J)}{\int \mathsf{P}(p_1,\ldots,p_J)\cdot\mathsf{P}(\mathcal{D}\,|\,p_1,\ldots,p_J)\,dp_1\ldots dp_J} && \text{marginal}\\
&= \frac{\frac{\Gamma(\sum_j h_j)}{\prod_j \Gamma(h_j)}\cdot\prod_j (p_j)^{h_j-1}\cdot\frac{\Gamma(1+\sum_j \mathsf{F}_\mathcal{D}(j))}{\prod_j \Gamma(1+\mathsf{F}_\mathcal{D}(j))}\cdot\prod_j (p_j)^{\mathsf{F}_\mathcal{D}(j)}}{\int \frac{\Gamma(\sum_j h_j)}{\prod_j \Gamma(h_j)}\cdot\prod_j (p_j)^{h_j-1}\cdot\frac{\Gamma(1+\sum_j \mathsf{F}_\mathcal{D}(j))}{\prod_j \Gamma(1+\mathsf{F}_\mathcal{D}(j))}\cdot\prod_j (p_j)^{\mathsf{F}_\mathcal{D}(j)}\,dp_1\ldots dp_J} && \text{substitution}\\
&= \frac{\frac{\Gamma(\sum_j h_j)}{\prod_j \Gamma(h_j)}\cdot\frac{\Gamma(1+\sum_j \mathsf{F}_\mathcal{D}(j))}{\prod_j \Gamma(1+\mathsf{F}_\mathcal{D}(j))}\cdot\prod_j (p_j)^{h_j-1}\cdot\prod_j (p_j)^{\mathsf{F}_\mathcal{D}(j)}}{\frac{\Gamma(\sum_j h_j)}{\prod_j \Gamma(h_j)}\cdot\frac{\Gamma(1+\sum_j \mathsf{F}_\mathcal{D}(j))}{\prod_j \Gamma(1+\mathsf{F}_\mathcal{D}(j))}\cdot\int \prod_j (p_j)^{h_j-1}\cdot\prod_j (p_j)^{\mathsf{F}_\mathcal{D}(j)}\,dp_1\ldots dp_J} && \text{distrib. axiom}\\
&= \frac{\prod_j (p_j)^{h_j-1}\cdot\prod_j (p_j)^{\mathsf{F}_\mathcal{D}(j)}}{\int \prod_j (p_j)^{h_j-1}\cdot\prod_j (p_j)^{\mathsf{F}_\mathcal{D}(j)}\,dp_1\ldots dp_J} && \text{mult. of inverses}\\
&= \frac{\prod_j (p_j)^{\mathsf{F}_\mathcal{D}(j)+h_j-1}}{\int \prod_j (p_j)^{\mathsf{F}_\mathcal{D}(j)+h_j-1}\,dp_1\ldots dp_J} && \text{mult. of powers}\\
&= \frac{\Gamma(\sum_j \mathsf{F}_\mathcal{D}(j)+h_j)}{\prod_j \Gamma(\mathsf{F}_\mathcal{D}(j)+h_j)}\cdot\prod_j (p_j)^{\mathsf{F}_\mathcal{D}(j)+h_j-1} && \text{def. of Dirichlet}
\end{aligned}$$

Multiplying in prior now directly maximizes (at zero slope) probability of parameters $p_1,\ldots,p_J$:

$$\begin{aligned}
0 &= \frac{\partial}{\partial p_i}\,\mathsf{P}(p_1,\ldots,p_J \,|\, \mathcal{D})\\
&= \frac{\partial}{\partial p_i}\,\frac{\mathsf{P}(p_1,\ldots,p_J)\cdot\mathsf{P}(\mathcal{D}\,|\,p_1,\ldots,p_J)}{\mathsf{P}(\mathcal{D})} && \text{Bayes' rule}\\
&= \frac{\partial}{\partial p_i}\,\frac{\mathsf{P}(p_1,\ldots,p_J)\cdot\mathsf{P}(\mathcal{D}\,|\,p_1,\ldots,p_J)}{\int \mathsf{P}(p_1,\ldots,p_J)\cdot\mathsf{P}(\mathcal{D}\,|\,p_1,\ldots,p_J)\,dp_1\ldots dp_J} && \text{marginal}\\
&= \frac{\partial}{\partial p_i}\,\frac{\frac{\Gamma(\sum_j h_j)}{\prod_j \Gamma(h_j)}\cdot\prod_j (p_j)^{h_j-1}\cdot\frac{\Gamma(1+\sum_j \mathsf{F}_\mathcal{D}(j))}{\prod_j \Gamma(1+\mathsf{F}_\mathcal{D}(j))}\cdot\prod_j (p_j)^{\mathsf{F}_\mathcal{D}(j)}}{\int \frac{\Gamma(\sum_j h_j)}{\prod_j \Gamma(h_j)}\cdot\prod_j (p_j)^{h_j-1}\cdot\frac{\Gamma(1+\sum_j \mathsf{F}_\mathcal{D}(j))}{\prod_j \Gamma(1+\mathsf{F}_\mathcal{D}(j))}\cdot\prod_j (p_j)^{\mathsf{F}_\mathcal{D}(j)}\,dp_1\ldots dp_J} && \text{substitution}\\
&= \frac{\partial}{\partial p_i}\,\frac{\Gamma(\sum_j \mathsf{F}_\mathcal{D}(j)+h_j)}{\prod_j \Gamma(\mathsf{F}_\mathcal{D}(j)+h_j)}\cdot\prod_j (p_j)^{\mathsf{F}_\mathcal{D}(j)+h_j-1} && \text{conjugacy of Dirichlet}\\
&= \frac{\partial}{\partial p_i}\,\prod_j (p_j)^{\mathsf{F}_\mathcal{D}(j)+h_j-1} && \text{product rule, mult. by constant}\\
&= \frac{\partial}{\partial p_i}\,(p_i)^{\mathsf{F}_\mathcal{D}(i)+h_i-1}\prod_{j\neq i} (p_j)^{\mathsf{F}_\mathcal{D}(j)+h_j-1} && \text{def. limit product}\\
&= \frac{\partial}{\partial p_i}\,(p_i)^{\mathsf{F}_\mathcal{D}(i)+h_i-1}(1-p_i)^{\sum_{j\neq i}\mathsf{F}_\mathcal{D}(j)+h_j-1} && \text{multinomial distrib. sums to one}
\end{aligned}$$

$$= \frac{\partial}{\partial p} p^n (1-p)^m \qquad \text{let } p = p_i,\ n = \mathsf{F}_{\mathcal{D}}(i){+}h_i{-}1,\ m = \sum_{j \neq i} \mathsf{F}_{\mathcal{D}}(j){+}h_j{-}1$$

$$= \left(\frac{\partial}{\partial p} p^n\right)(1-p)^m + p^n \left(\frac{\partial}{\partial p}(1-p)^m\right) \qquad \text{product rule}$$

$$= np^{n-1}(1-p)^m + p^n m(1-p)^{m-1}\left(\frac{\partial}{\partial p} 1 - p\right) \qquad \text{power rule}$$

$$= np^{n-1}(1-p)^m + p^n m(1-p)^{m-1}(-1) \qquad \text{power rule}$$

$$= p^{n-1}(1-p)^{m-1}(n(1-p) - mp) \qquad \text{distributive axiom}$$

$$= p^{n-1}(1-p)^{m-1}(n - np - mp) \qquad \text{distributive axiom}$$

$$= \underbrace{p^{n-1}}_{\text{root: } \hat{p} = 0} \underbrace{(1-p)^{m-1}}_{\text{root: } \hat{p} = 1} \underbrace{(n - (n+m)p)}_{\text{root: } \hat{p} = \frac{n}{n+m}} \qquad \text{distributive axiom}$$

So (ignoring 0 and 1 roots, which are minima) the optimal parameters are all $\hat{p}_i = \frac{\mathsf{F}_{\mathcal{D}}(i)+h_i-1}{\sum_j \mathsf{F}_{\mathcal{D}}(j)+h_j-1}$.

This is now a **Dirichlet-multinomial** model.

When $J = 2$ it's called a **Beta-binomial** model.

In training, this gives estimates with 'pseudocounts' $h_j - 1$ added to each multinomial parameter.

These are then added together with real counts and renormalized. No more zero counts!

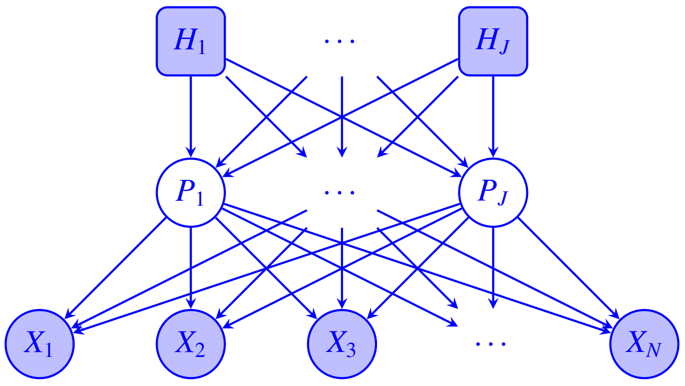
This estimate is called a **maximum a posteriori estimate** because it maximizes the posterior.

(The maximum likelihood estimate is called that because it maximizes the likelihood.)

This general technique of buffing up low or zero counts is called **smoothing**.

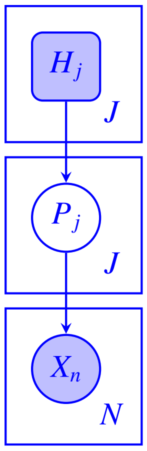
## 4.3 Graphical representations

Bayes net representation of Dirichlet-multinomial (where $O{=}P^J{\times}X^N$, $P{=}\mathbb{R}^1_0$, $X{=}\mathbb{Z}^J_1$) is very busy:



(The rounded boxes are parameters or hyperparameters not in sample space $O$.)

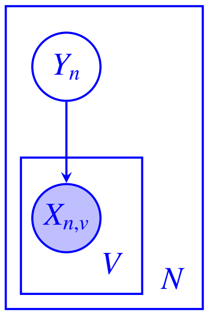
Statisticians represent the same thing using **plate diagrams** — they are more tidy:



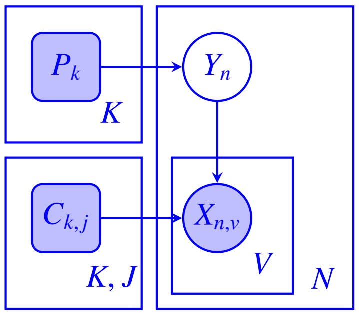
## 4.4 Bayesian (smoothed) Naive Bayes

We can also apply smoothing to our Naive Bayes model.

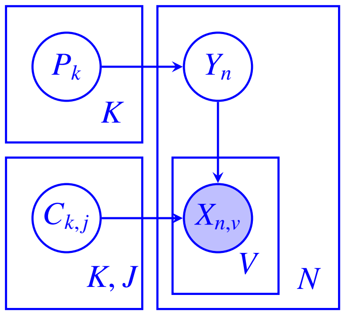
Here's original Naive Bayes (where $O{=}Y^N{\times}X^{N\times V}$, $Y{=}\mathbb{Z}_1^K$, $X{=}\mathbb{Z}_1^J$) as a plate diagram:



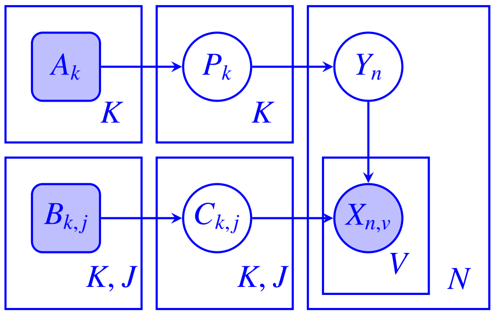
or, showing parameters:



Here's the smoothed version (where $O{=}P^K{\times}Y^N{\times}C^{K\times J}{\times}X^{N\times V}$, $P{=}\mathbb{R}_0^1$, $Y{=}\mathbb{Z}_1^K$, $C{=}\mathbb{R}_0^1$, $X{=}\mathbb{Z}_1^J$):



or with hyperparameters:



The code is exactly the same, except you add your hyperparameters during training.

## 4.5 Probabilistic model comparison using a binomial significance test

Suppose you wrote a new classifier and you want to see if it's better than your old one.

You can find out by running them on some labeled data and comparing their accuracy scores.

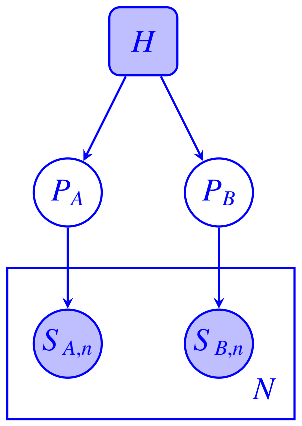
But running them on too few examples may give you the wrong winner. When do you believe it?

Assume parameters $p_A$ and $p_B$ for the old and new systems predicting correct/incorrectly $\{1, 0\}$.

The sample space is: $O{=}P{\times}P{\times}S^N{\times}S^N$, $P{=}\mathbb{R}_0^1$, $S{=}\mathbb{Z}_0^1$ (indicator $[\![\phi]\!] = 1$ if $\phi$ is true, $0$ otherwise).

$$\begin{aligned}
\mathsf{P}(p_B > p_A) &= \int \begin{array}{l} \mathsf{P}(p_A \mid h_A, h'_A) \cdot \mathsf{P}(s_A{=}1 \mid p_A)^{\mathsf{F}_{\mathcal{D}_A}(1)} \cdot \mathsf{P}(s_A{=}0 \mid p_A)^{\mathsf{F}_{\mathcal{D}_A}(0)} \cdot \\ \mathsf{P}(p_B \mid h_B, h'_B) \cdot \mathsf{P}(s_B{=}1 \mid p_B)^{\mathsf{F}_{\mathcal{D}_B}(1)} \cdot \mathsf{P}(s_B{=}0 \mid p_B)^{\mathsf{F}_{\mathcal{D}_B}(0)} \cdot [\![p_B > p_A]\!]\, dp_A\, dp_B \end{array} \\
&= \int \begin{array}{l} \mathrm{Beta}_{h_A,h'_A}(p_A) \cdot \mathrm{Binomial}_{p_A}(\mathsf{F}_{\mathcal{D}_A}(1), \mathsf{F}_{\mathcal{D}_A}(0)) \cdot \\ \mathrm{Beta}_{h_B,h'_B}(p_B) \cdot \mathrm{Binomial}_{p_B}(\mathsf{F}_{\mathcal{D}_B}(1), \mathsf{F}_{\mathcal{D}_B}(0)) \cdot [\![p_B > p_A]\!]\, dp_A\, dp_B \end{array} \\
&= \int \mathrm{Beta}_{\mathsf{F}_{\mathcal{D}_A}(1)+h_A, \mathsf{F}_{\mathcal{D}_A}(0)+h'_A}(p_A) \cdot \mathrm{Beta}_{\mathsf{F}_{\mathcal{D}_B}(1)+h_B, \mathsf{F}_{\mathcal{D}_B}(0)+h'_B}(p_B) \cdot [\![p_B > p_A]\!]\, dp_A\, dp_B
\end{aligned}$$

This is just two beta-binomial models, side by side:



You can use sampling to estimate the fraction of this space in which the newer model is better:

```
import sys
import numpy
import pandas

SS = pandas.read_csv(sys.argv[1])
numsamples = 1000                ## not N! -- these samples are approximating the integral
numBwins = 0

for i in range(numsamples):
  pA = numpy.random.beta( len(SS[ SS['scoreA']==1 ]) + 1, len(SS[ SS['scoreA']==0 ]) + 1 )
  pB = numpy.random.beta( len(SS[ SS['scoreB']==1 ]) + 1, len(SS[ SS['scoreB']==0 ]) + 1 )
  if pB > pA: numBwins = numBwins + 1

print( 'Fraction where accuracy parameter of B > of A: ' + str(numBwins/numsamples) )
```

Now we can see it work! If we run it on a small test set:

```
scoreA,scoreB
0,0
0,0
0,0
0,1
0,1
0,1
1,0
1,1
```

we get a low probability ('confidence') that the new model is actually better:

```
Fraction where accuracy parameter of B > of A: 0.85
```

If we run it on a larger test set (this is just twice as many of each outcome):

```
scoreA,scoreB
0,0
0,0
0,0
0,0
0,0
0,0
0,1
0,1
0,1
0,1
0,1
0,1
1,0
1,0
1,1
1,1
```

we get a higher probability ('confidence') that the new model is actually better:

```
Fraction where accuracy parameter of B > of A: 0.922
```

Note the model didn't change; we just ran it on a larger test set.

The moral: make sure your test set is big enough to see your effect!

Also: increasing numsamples will boost the fidelity of the estimate, not the confidence itself.

Also also: the confidence is ***not*** the magnitude of the difference. That's this:

```
import sys
import numpy
import pandas

ss = pandas.read_csv(sys.argv[1])

numsamples = 1000
```

```
numBwins0 = 0
numBwins1 = 0
numBwins2 = 0
for i in range(numsamples):

  pA = numpy.random.beta( len(ss[ ss['scoreA']==1 ]) + 1, len(ss[ ss['scoreA']==0 ]) + 1 )
  pB = numpy.random.beta( len(ss[ ss['scoreB']==1 ]) + 1, len(ss[ ss['scoreB']==0 ]) + 1 )

  if pB > pA:      numBwins0 = numBwins0 + 1
  if pB > pA + .1: numBwins1 = numBwins1 + 1
  if pB > pA + .2: numBwins2 = numBwins2 + 1

print( 'Fraction where accuracy parameter of B > of A: '      + str(numBwins0/numsamples) )
print( 'Fraction where accuracy parameter of B > of A + .1: ' + str(numBwins1/numsamples) )
print( 'Fraction where accuracy parameter of B > of A + .2: ' + str(numBwins2/numsamples) )
```

On our larger data set, it gives confidence of different magnitudes of difference:

```
Fraction where accuracy parameter of B > of A: 0.933
Fraction where accuracy parameter of B > of A + .1: 0.803
Fraction where accuracy parameter of B > of A + .2: 0.599
```

On smaller data set:

```
Fraction where accuracy parameter of B > of A: 0.835
Fraction where accuracy parameter of B > of A + .1: 0.689
Fraction where accuracy parameter of B > of A + .2: 0.519
```