Non-parametric Bayesian Statistics

Graham Neubig

2011-12-22

Overview

- About Bayesian Non-parametrics
  - Basic theory
  - Inference using sampling
  - Learning an HMM with sampling
  - From the finite HMM to the infinite HMM
  - Recent developments (in sampling and modeling)
  - Applications to speech and language processing
- Focus on unsupervised learning for discrete distributions

Non-parametric

The number of parameters is not decided in advance (i.e. infinite)

Bayes

Put a prior on the parameters and consider their distribution

Types of Statistical Models

	Prior on Parameters	# of Parameters (Classes)	Discrete Distribution	Continuous Distribution
Maximum Likelihood	No	Finite	Multinomial	Gaussian
Bayesian Parametric	Yes	Finite	Multinomial+ Dirichlet Prior	Gaussian+ Gaussian Prior
Bayesian Non-parametric	Yes	Infinite	Multinomial+ Dirichlet Process	Gaussian Process

Covered Here

Bayesian Basics

Maximum Likelihood (ML)

- We have an observed sample

  X = 1 2 4 5 2 1 4 4 1 4

- Gather counts $C=\{c_{1,} c_{2,} c_{3,} c_{4,} c_5\}=\{3,2,0,4,1\}$
- Divide counts to get probabilities

$$P(x=i)=\frac{c_i}{\sum_{\tilde{i}} c_{\tilde{i}}}$$

multinomial

$$P(x)=\vec{\theta}=\{0.3,0.2,0,0.4,0.1\}$$

Bayesian Inference

- ML is weak against sparse data
- Don't actually know parameters

if $c(x)=\{3,2,0,4,1\}$

we could have

$$\vec{\theta}=\{0.3,0.2,0,0.4,0.1\}$$

or we could have

$$\vec{\theta}=\{0.35,0.05,0.05,0.35,0.2\}$$

- Bayesian statistics don't pick one probability
  - Use the expectation instead

$$P(x=i)=\int \theta_i P(\vec{\theta}|X)\,d\vec{\theta}$$

Calculating Parameter Distributions

- Decompose with Bayes' law

$$P(\theta|X) = \frac{\overbrace{P(X|\theta)}^{\text{likelihood}}\overbrace{P(\theta)}^{\text{prior}}}{\underbrace{\int P(X|\theta)P(\theta)\,d\theta}_{\text{regularization coefficient}}}$$

- likelihood easily calculated according to the model
- prior chosen according belief about probable values
- regularization requires difficult integration...
  - … but conjugate priors make things easier

Conjugate Priors

- Definition: Product of likelihood and prior takes the same form as the prior

Multinomial Likelihood * Dirichlet Prior = Dirichlet Posterior

Gaussian Likelihood * Gaussian Prior = Gaussian Posterior

Same

- Because the form is known, no need to take the integral to regularize

Dirichlet Distribution/Process

- Assigns probabilities to multinomial distributions

e.g.

$$P(\{0.3, 0.2, 0.01, 0.4, 0.09\}) = 0.000512$$

$$P(\{0.35, 0.05, 0.05, 0.35, 0.2\}) = 0.0000963$$

- Defined over the space of proper probability distributions $\{\theta_1, \ldots, \theta_n\}$

$$\forall_{\theta_i} 0 \leqslant \theta_i \leqslant 1 \qquad \sum_{i=1}^{n} \theta_i = 1$$

- Dirichlet process is a generalization of distribution
  - Can assign probabilities to infinite spaces

Dirichlet Process (DP)

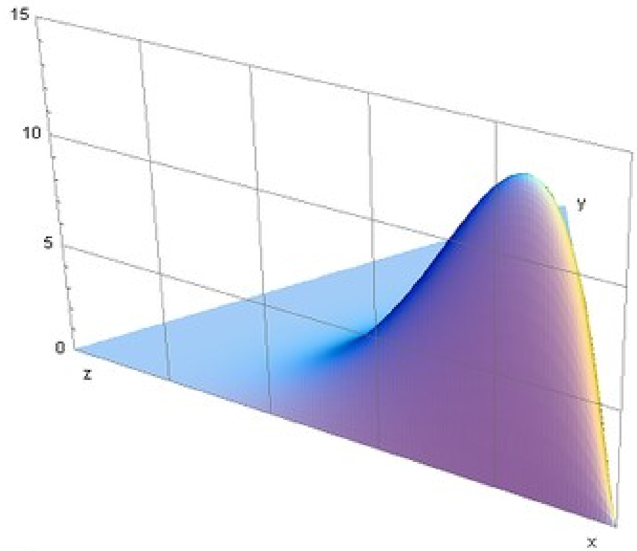
- Eq. $$P(\vec{\theta};\alpha,P_{base})=\frac{1}{Z}\prod_{i=1}^{n}\theta_i^{\alpha P_{base}(x=i)-1}$$
  - α is the "concentration parameter," larger value means more data needed to diverge from prior
  - P_{base} is the "base measure," expectation of θ

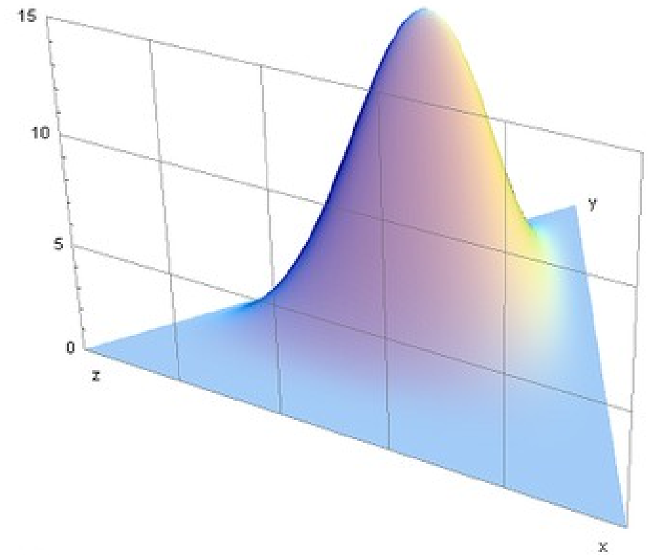
Way of writing in Dirichlet distribution $$\alpha_i=\alpha P_{base}(x=i)$$ Way of writing in Dirichlet process

- Regularization coefficient: (Γ=gamma function) $$Z=\frac{\prod_{i=1}^{n}\Gamma(\alpha P_{base}(x=i))}{\Gamma(\sum_{i=1}^{n}\alpha P_{base}(x=i))}$$

Examples of Probability Densities

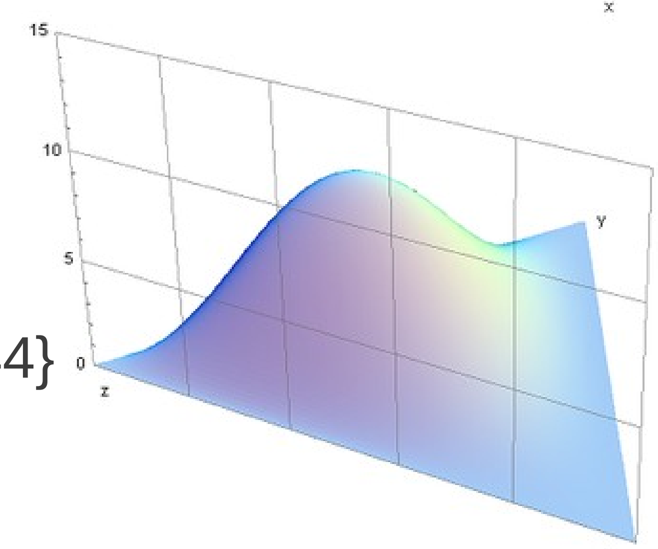
α = 10
P_{base} =
{0.6,0.2,0.2}

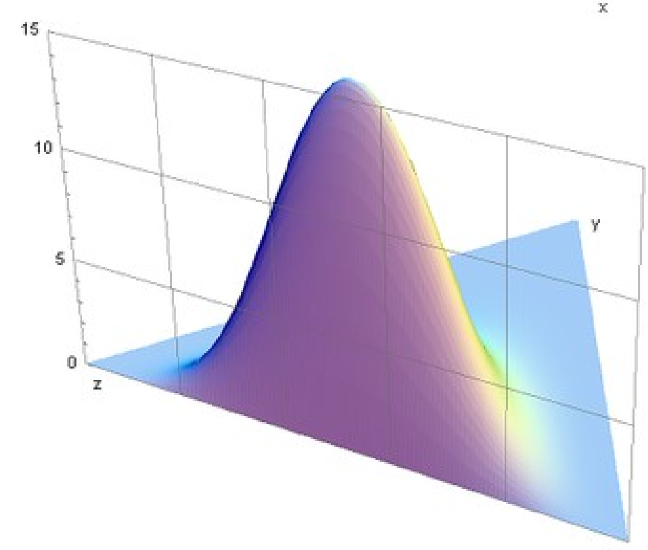




α = 15
P_{base} =
{0.2,0.47,0.33}

α = 9
P_{base} =
{0.22,0.33,0.44}





α = 14
P_{base} =
{0.43,0.14,0.43}

From Wikipedia

Why is the Dirichlet Conjugate?

- Likelihood is product of multinomial probabilities

Data: $x_1=1, x_2=5, x_3=2, x_4=5$

$$P(X|\theta)=p(x=1|\theta)\,p(x=5|\theta)\,p(x=2|\theta)\,p(x=5|\theta)=\theta_1\theta_5\theta_2\theta_5$$

- Combine multiple instances into a single count

$$c(x=i)=\{1,1,0,0,2\}$$

$$P(X|\theta)=\theta_1\theta_2\theta_5^2=\prod_{i=1}^{n}\theta_i^{c(x=i)}$$

- Take product of likelihood and prior

$$\prod_{i=1}^{n}\theta_i^{c(x=i)} * \frac{1}{Z_{prior}}\prod_{i=1}^{n}\theta_i^{\alpha_i-1} \rightarrow \frac{1}{Z_{post}}\prod_{i=1}^{n}\theta_i^{c(x=i)+\alpha_i-1}$$

Expectation of θ in the DP

- When N=2

$$E[\theta_1]=\int_0^1 \theta_1 \frac{1}{Z}\theta_1^{\alpha_1-1}\theta_2^{\alpha_2-1}d\theta_1$$

$$=\frac{1}{Z}\int_0^1 \theta_1^{\alpha_1}(1-\theta_1)^{\alpha_2-1}d\theta_1$$

Integration by Parts

$$u=\theta_1^{\alpha_1} \qquad du=\alpha_1\theta_1^{\alpha_1-1}d\theta_1$$

$$dv=(1-\theta_1)^{\alpha_2-1}d\theta_1$$

$$v=-(1-\theta_1)^{\alpha_2}/\alpha_2$$

$$\int u\,dv=uv-\int v\,du$$

$$=\frac{1}{Z}[-\theta_1^{\alpha_1}(1-\theta_1)^{\alpha_2}/\alpha_2]_0^1-$$

$$\frac{1}{Z}\int_0^1 -(1-\theta_1)^{\alpha_2}/\alpha_2 * \alpha_1\theta_1^{\alpha_1-1}d\theta_1$$

$$=0+\frac{\alpha_1}{\alpha_2}\frac{1}{Z}\int_0^1 \theta_1^{\alpha_1-1}(1-\theta_1)^{\alpha_2}d\theta_1$$

$$=\frac{\alpha_1}{\alpha_2}E[\theta_2]=\frac{\alpha_1}{\alpha_2}(1-E[\theta_1])$$

$$E[\theta_1]=\frac{\alpha_1}{\alpha_1+\alpha_2}$$

Multi-Dimensional Expectation

$$E[\theta_i] = \frac{\alpha_i}{\sum_{i=1}^{n} \alpha_i} = \frac{\alpha P_{base}(x=i)}{\alpha} = P_{base}(x=i)$$

- Posterior distribution for multinomial with DP prior:

$$P(x=i) = \int_0^1 \theta_i \frac{1}{Z_{post}} \prod_{i=1}^{n} \theta_i^{c(x=i)+\alpha_i-1}$$

Observed Counts

$$= \frac{c(x=i) + \alpha * P_{base}(x=i)}{c(\cdot) + \alpha}$$

Base Measure

Concentration Parameter

- Same as additive smoothing

Marginal Probability

- Calculate prob. of observed data using the chain rule

X = 1 2 1 3 1   α=1 P_{base}(x=1,2,3,4) = .25

$$P(x_i)=\frac{c(x_i)+\alpha * P_{base}(x_i)}{c(\cdot)+\alpha}$$

c = { 0, 0, 0, 0 }

$$P(x_1=1)=\frac{0+1*.25}{0+1}=.25$$

c = { 1, 0, 0, 0 }

$$P(x_2=2|x_1)=\frac{0+1*.25}{1+1}=.125$$

c = { 1, 1, 0, 0 }

$$P(x_3=1|x_{1,2})=\frac{1+1*.25}{2+1}=.417$$

c = { 2, 1, 0, 0 }

$$P(x_4=3|x_{1,2,3})=\frac{0+1*.25}{3+1}=.063$$

c = { 2, 1, 1, 0 }

$$P(x_5=1|x_{1,2,3,4})=\frac{2+1*.25}{4+1}=.45$$

Marginal Probability
P(X) = .25*.125*.417*.063*.45

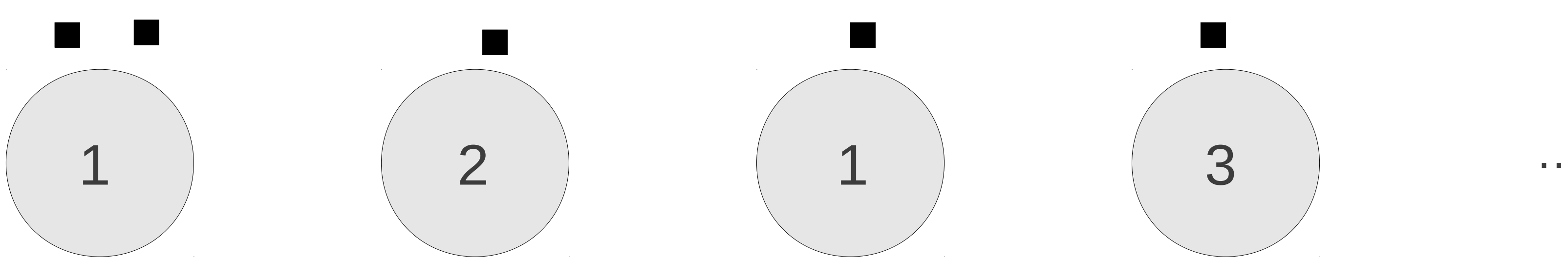
Chinese Restaurant Process

- Way of expressing DP and other stochastic processes
- Chinese restaurant with infinite number of tables
- Each customer enters restaurant and takes action:

$$P(\text{sits at table i}) \propto c(i)$$
$$P(\text{sits at a new table}) \propto \alpha$$

- When the first customer sits at a table, choose the food served there according to P_{base}

X = 1 2 1 3 1 α=1 N=4



Sampling Basics

Sampling Basics

- Generate a sample from probability distribution:

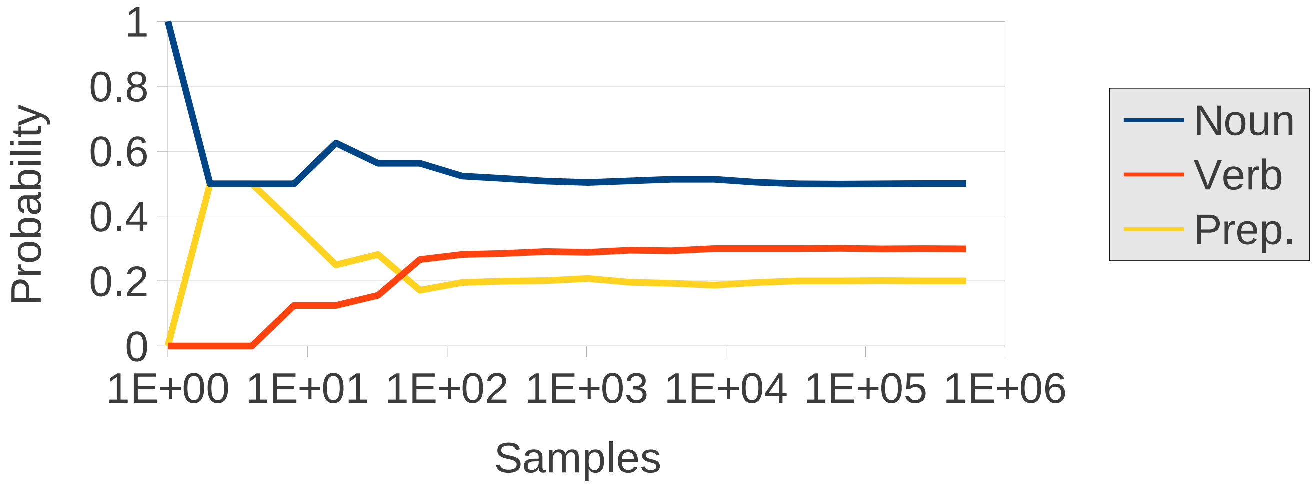
Distribution: P(Noun)=0.5 P(Verb)=0.3 P(Preposition)=0.2

Sample: Verb Verb Prep. Noun Noun Prep. Noun Verb Verb Noun …

- Count the samples and calculate probabilities

P(Noun)= 4/10 = 0.4, P(Verb)= 4/10 = 0.4, P(Preposition) = 2/10 = 0.2

- More samples = better approximation



Actual Algorithm

```
SampleOne(probs[])
    z = sum(probs)
    remaining = rand(z)
    for each i in 1:probs.size
        remaining -= probs[i]
        if remaining <= 0
            return i
```

Calculate sum of probs

Generate number from uniform distribution over [0,z)

Iterate over all probabilities

Subtract current prob. value

If smaller than zero, return current index as answer

Bug check, beware of overflow!

Gibbs Sampling

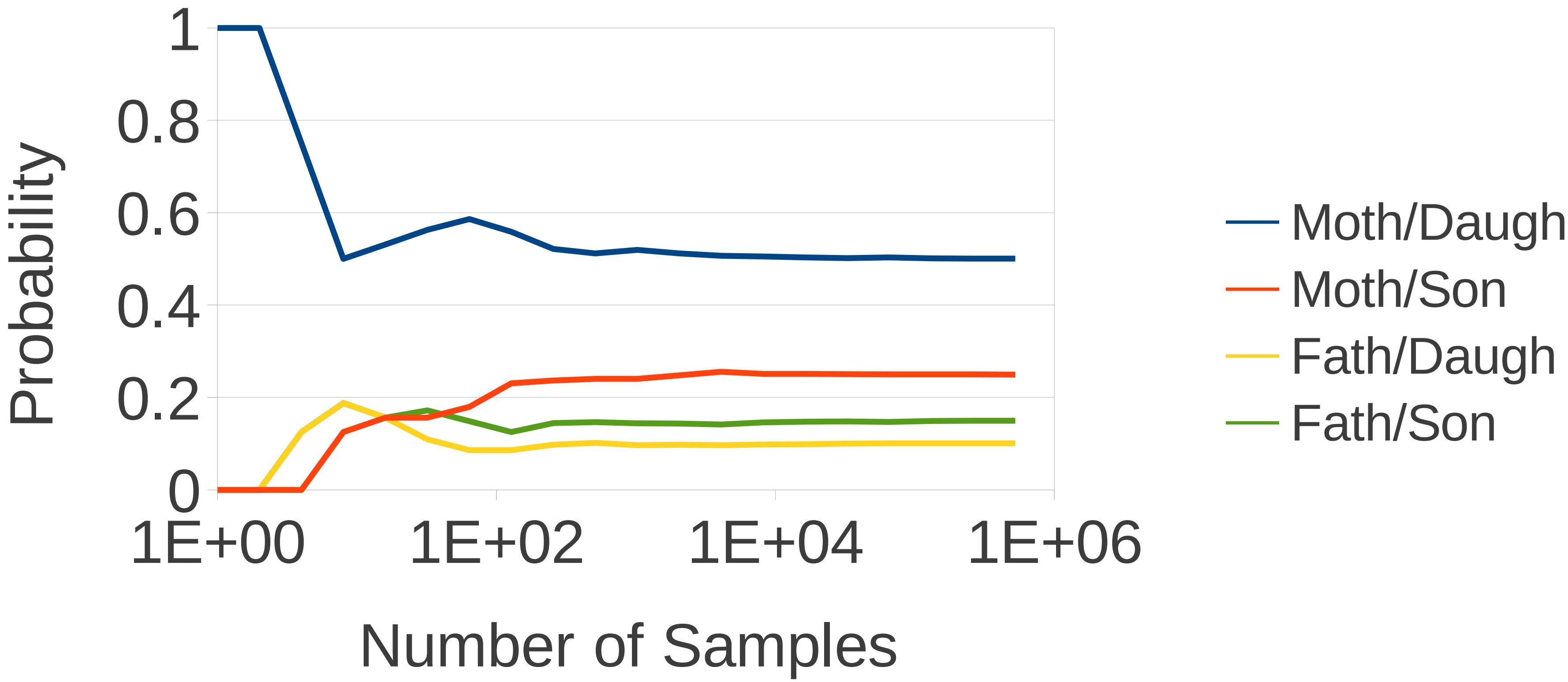
- Want to sample a 2-variable distribution P(A,B)
  - … but cannot sample directly from P(A,B)
  - … but can sample from P(A|B) and P(B|A)
- Gibbs sampling samples variables one-by-one to recover true distribution
- Each iteration:
  Leave A fixed, sample B from P(B|A)
  Leave B fixed, sample A from P(A|B)

Example of Gibbs Sampling

- Parent A and child B are shopping, what sex?
  P(Mother|Daughter) = 5/6 = 0.833
  P(Mother|Son) = 5/8 = 0.625
  P(Daughter|Mother) = 2/3 = 0.667
  P(Daughter|Father) = 2/5 = 0.4
- Original state: Mother/Daughter
  - Sample P(Mother|Daughter)=0.833, chose Mother
  - Sample P(Daughter|Mother)=0.667, chose Son
  - c(Mother, Son)++
  - Sample P(Mother|Son)=0.625, chose Mother
  - Sample P(Daughter|Mother)=0.667, chose Daughter
  - c(Mother, Daughter)++

…

Try it Out:



- In this case, we can confirm this result by hand

Learning a Hidden Markov Model Part-of-Speech Tagger with Sampling

Unsupervised Learning

- Observed Training Data X
  - e.g.: A corpus of natural language text
- Hidden Variables Y
  - e.g.: States of the HMM = Parts of Speech of words
- Unobserved Parameters θ
  - Generally probabilities

Task: Unsupervised POS Induction

- Input: Collection of word strings X

the boats row in a row

- Output: Collection of clusters Y

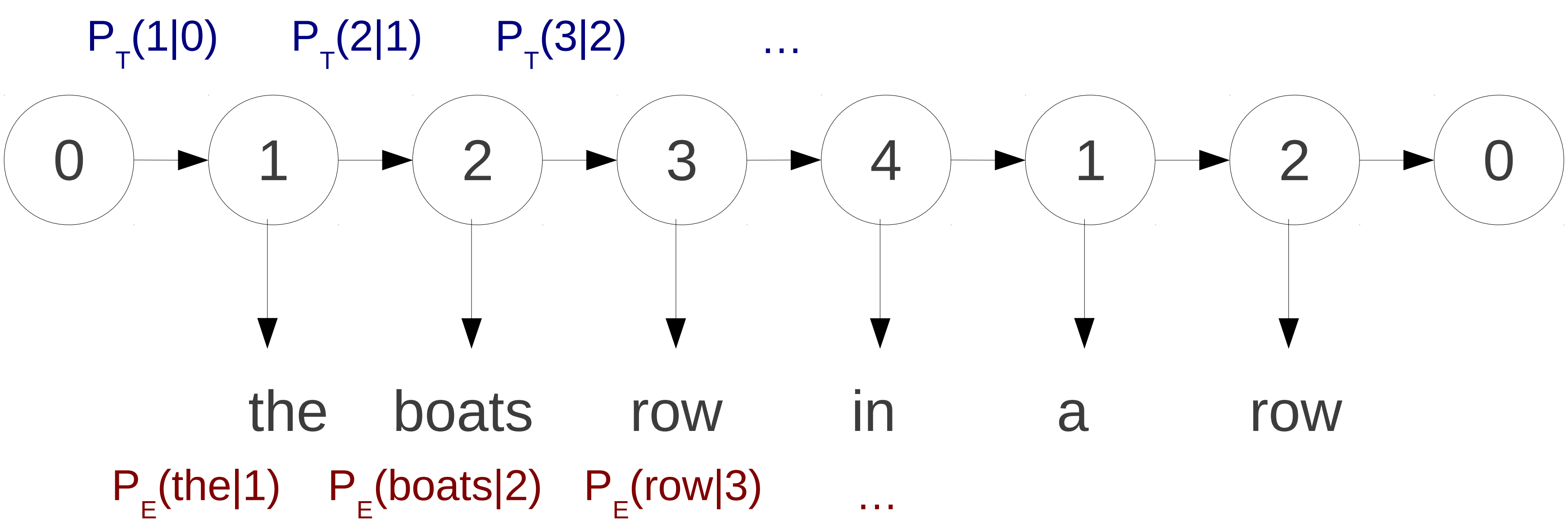
1 2 3 4 1 2

1→Determiner 2→Noun 3→Verb 4→Preposition

the	boats	row	in	a	row
Det	N	V	P	Det	N

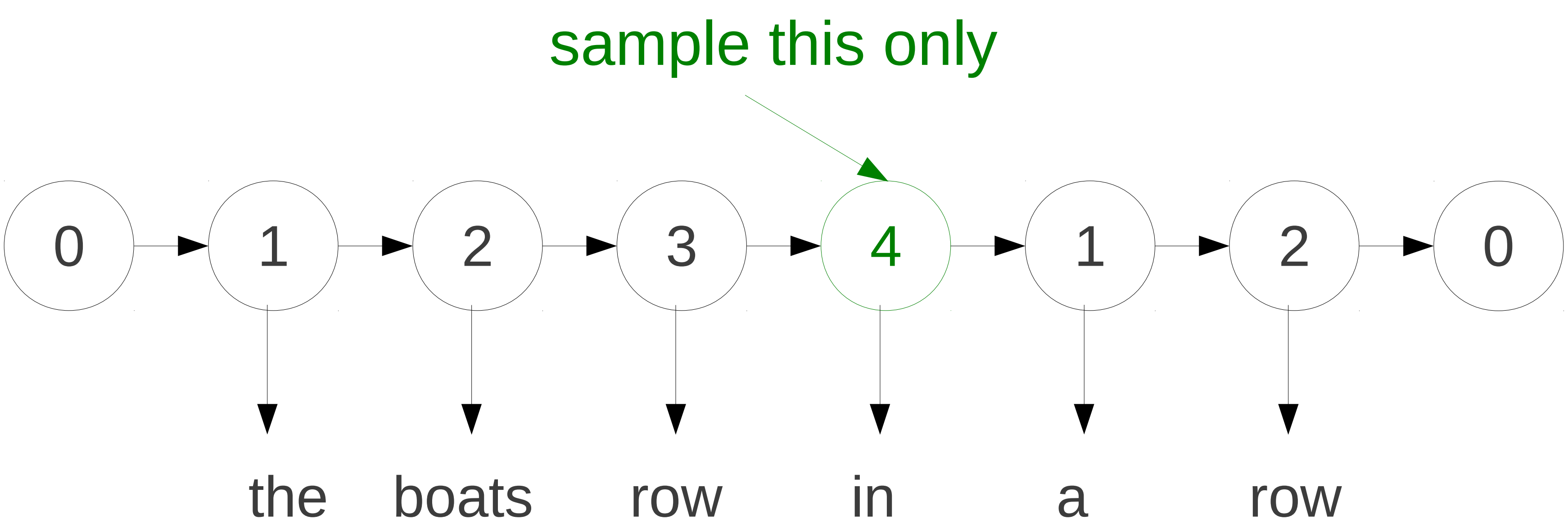
Model: HMM

- Variables Y correspond to hidden states
  - State **transition probability:** $P_T(y_i | y_{i-1}) = \theta_{T, y_i, y_{i-1}}$
- Generate each word from a hidden state
  - Word **emission probability:** $P_E(x_i | y_i) = \theta_{E, y_i, x_i}$

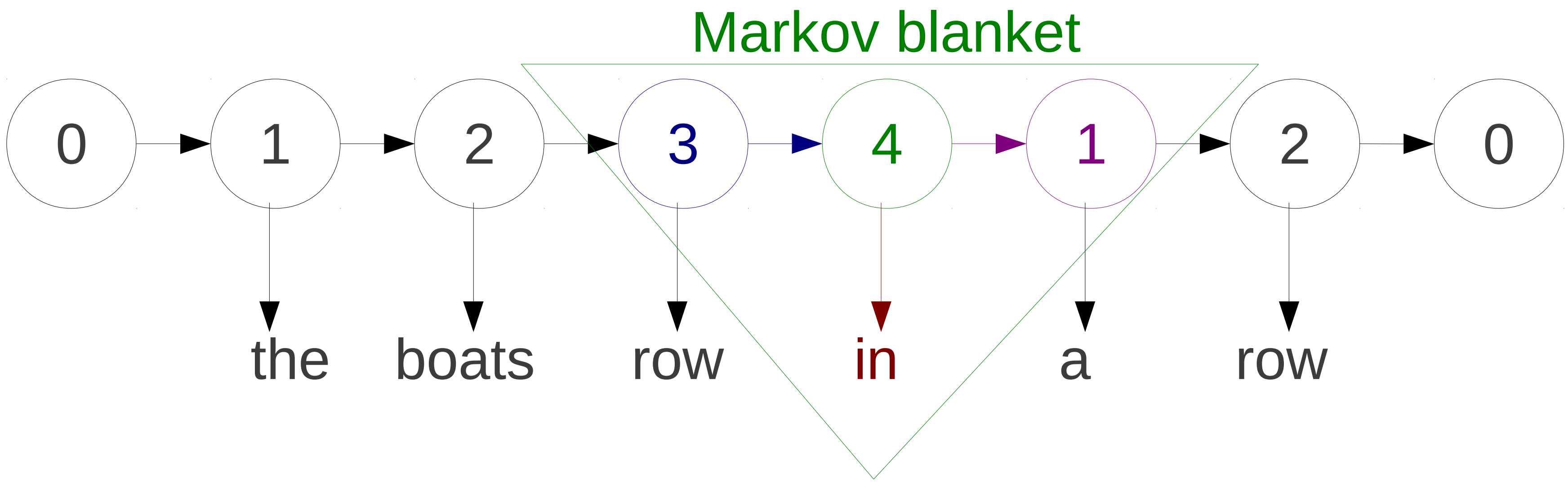


Sampling the HMM

- Initialize Y randomly
- Sample each element of Y using Gibbs sampling



Sampling the HMM



- Probabilities affected by a single tag
  - Transition from previous tag: $P_T(y_i|y_{i-1})$
  - Transition to next tag: $P_T(y_{i+1}|y_i)$
  - Emission probability: $P_E(x_i|y_i)$
- Sample the tag value according to these probabilities
- All variables that have effect are "Markov blanket"

Calculating HMM Probabilities with DP Priors

- Transition probability:

$$P_T(y_i|y_{i-1}) = \frac{c(y_{i-1} y_i) + \alpha_T * P_{baseT}(y_i)}{c(y_{i-1}) + \alpha_T}$$

- Emission probability:

$$P_E(x_i|y_i) = \frac{c(y_i, x_i) + \alpha_E * P_{baseE}(x_i)}{c(y_i) + \alpha_E}$$

Sampling Algorithm for One Tag

SampleTag(y_i)

$c(y_{i-1}\ y_i)$--; $c(y_i\ y_{i+1})$--; $c(y_i \rightarrow x_i)$--

Subtract current tag counts

for each *tag* **in** S (all POS tags)

$p[tag] = P_E(tag|y_{i-1}) * P_E(y_{i+1}|tag) * P_T(x_i|tag)$

Calculate all possible tag probabilities

y_i = **SampleOne**(p)

Choose a new tag

$c(y_{i-1}\ y_i)$++; $c(y_i\ y_{i+1})$++; $c(y_i \rightarrow x_i)$++

Add the new tag counts

Sampling Algorithm for All Tags

SampleCorpus()

initialize Y randomly	Randomly initialize tags
for N iterations	For N iterations
for each y_i **in** the corpus	
SampleTag(y_i)	Sample all the tags
save parameters	Save sample of θ
average parameters	Average parameters θ

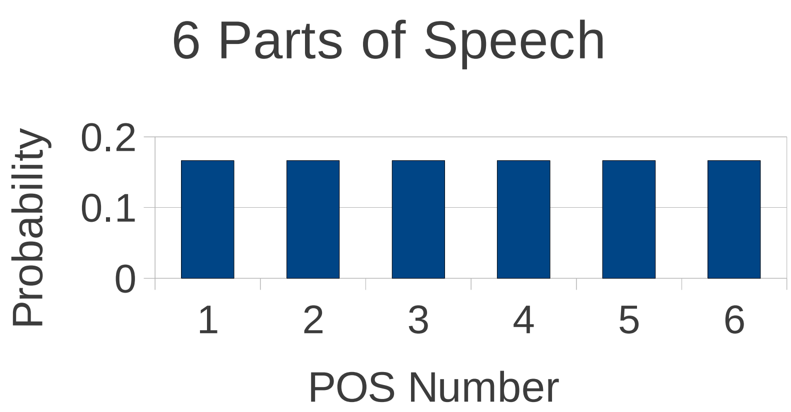
Choosing Hyperparameters

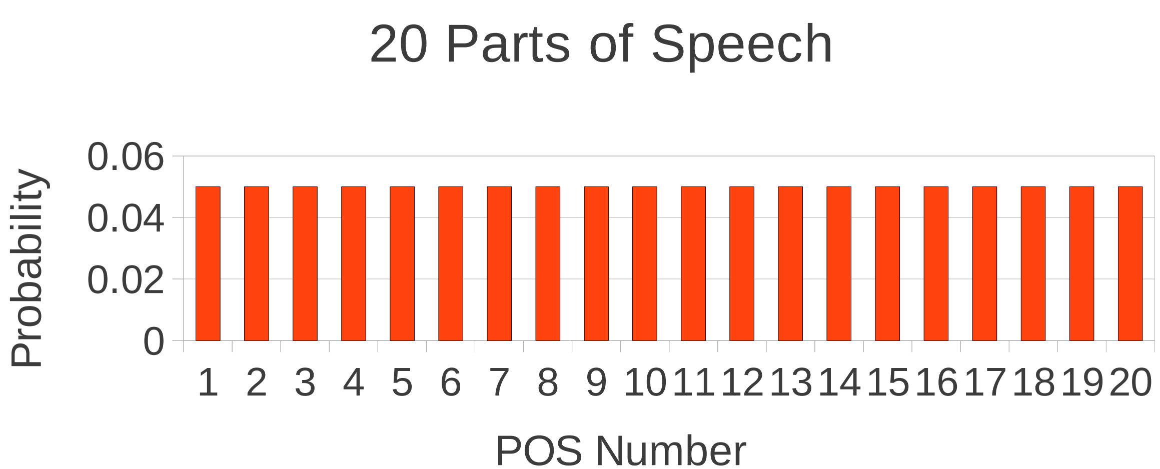
- Must choose α properly to get desired effect
  - Small α(<0.1) creates sparse distributions
    - If we want each word to have one POS tag, we can set α_E of the emission distribution P_e to be small
  - Most distributions are sparse, so often α is set small
- Best to confirm through experiments
- Can also give hyperparameters a prior and sample them as well

From the Finite HMM to the Infinite HMM

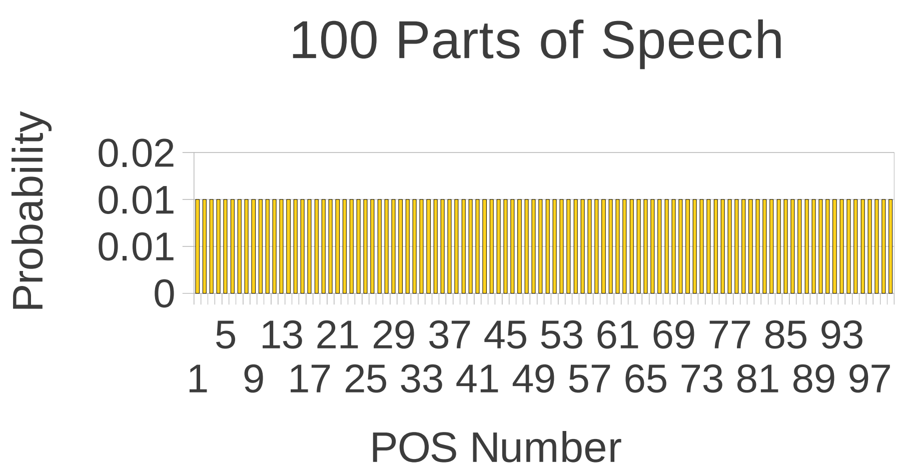
Base Measure and Dimensionality

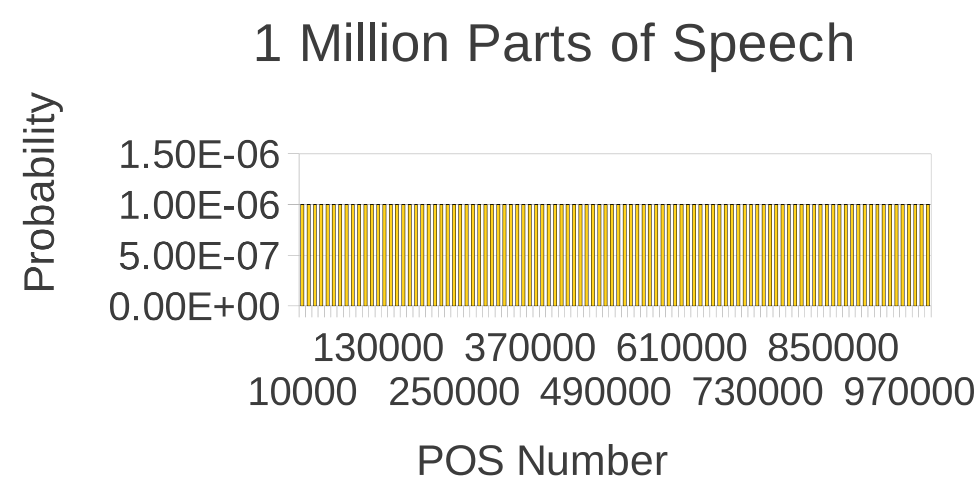
- Using a uniform distribution as the base measure





In the Limit...





- As the number of POSs goes to infinity
  - Probabilities of each POS P_{base} goes to zero
  - But total probability of P_{base} is the same

$$P(y_i|y_{i-1}) = \frac{c(y_{i-1} y_i) + \alpha * P_{base}(y_i)}{c(y_{i-1}) + \alpha}$$

N= number of POSs

$$\lim_{N \to \infty} \sum_{i=1}^{N} \frac{1}{N} = 1$$

Finite HMM and Infinite HMM

- Finite HMM
  Probability of emitting POS y_i (after y_{i-1})

$$P(y_i|y_{i-1})=\frac{c(y_{i-1}y_i)+\alpha * P_{base}(y_i)}{c(y_{i-1})+\alpha}$$

- Infinite HMM
  Probability of omitting existing POS y_i (after y_{i-1})

$$P(y_i|y_{i-1})=\frac{c(y_{i-1}y_i)}{c(y_{i-1})+\alpha}$$

  Probability of omitting new POS (after y_{i-1})

$$P(y_i=new|y_{i-1})=\frac{\alpha}{c(y_{i-1})+\alpha}$$

Example

- Assume $c(y_{i-1}=1\ y_i=1)=1\ c(y_{i-1}=1\ y_i=2)=1$

When there are 2 possible POSs

$$P(y_i=1|y_{i-1}=1)=\frac{1+\alpha*1/2}{2+\alpha} \quad P(y_i=2|y_{i-1}=1)=\frac{1+\alpha*1/2}{2+\alpha}$$

$$P(y_i\neq 1,2|y_{i-1}=1)=\frac{\alpha*0}{2+\alpha}$$

When there are 20 possible POSs

$$P(y_i=1|y_{i-1}=1)=\frac{1+\alpha*1/20}{2+\alpha} \quad P(y_i=2|y_{i-1}=1)=\frac{1+\alpha*1/20}{2+\alpha}$$

$$P(y_i\neq 1,2|y_{i-1}=1)=\frac{\alpha*18/20}{2+\alpha}$$

When there are infinite possible POSs

$$P(y_i=1|y_{i-1}=1)=\frac{1+\alpha*1/\infty}{2+\alpha} \quad P(y_i=2|y_{i-1}=1)=\frac{1+\alpha*1/\infty}{2+\alpha}$$

$$P(y_i\neq 1,2|y_{i-1}=1)=\frac{\alpha*1}{2+\alpha}$$

Sampling Algorithm

SampleTag(y_i)

$c(y_{i-1}\ y_i)$--; $c(y_i\ y_{i+1})$--; $c(y_i \rightarrow x_i)$--

Remove counts for current tag

for each *tag* in S (possible POSs)

$p[tag]=P_E(tag|y_{i-1})*P_E(y_{i+1}|tag)*P_T(x_i|tag)$

Calculate existing POS probabilities

$p[|S|+1]=P_E(new|y_{i-1})*P_E(y_{i+1}|new)*P_T(x_i|new)$

Calculate new POS probability

y_i = **SampleOne**(p)

Pick a single value

$c(y_{i-1}\ y_i)$++; $c(y_i\ y_{i+1})$++; $c(y_i \rightarrow x_i)$++

Add the new counts

Non-Uniform Base Measures

- Previous slides assumed uniform base measures, but this is not required
- Example: Language model unknown word model

$$P(\text{word}) = \frac{c(\text{word}) + \alpha * P_{base}(\text{word})}{c(\text{word}) + \alpha}$$

- Split each word into characters, give some probability to all words:

$$P_{base}(\text{word}) = P_{len}(4) P_{char}(\text{w}) P_{char}(\text{o}) P_{char}(\text{r}) P_{char}(\text{d})$$

- Probability is not equal, but gives some probability to each member of an infinite collection

Implementation Tips

- Zero count classes remain → wasted memory
  - When new classes are made, re-use class numbers

$c(y_1)=5\ c(y_2)=0\ c(y_3)=1$

Dumb: $c(y_1)=5\ c(y_2)=0\ c(y_3)=1\ c(y_4)=1$

Smart: $c(y_1)=5\ c(y_2)=1\ c(y_3)=1$

  - When c(y)=0, probability of revival becomes 0
- This model doesn't do well with new POSs
  - New POSs can only appear after 1 type of POS
  - Can fix this with hierarchical model

Transition Prob. → $P_T(y_i|y_{i-1})=DP(\alpha, P_T(y_i))$

POS Prob. → $P_T(y_i)=DP(\alpha, P_{base}(y_i))$

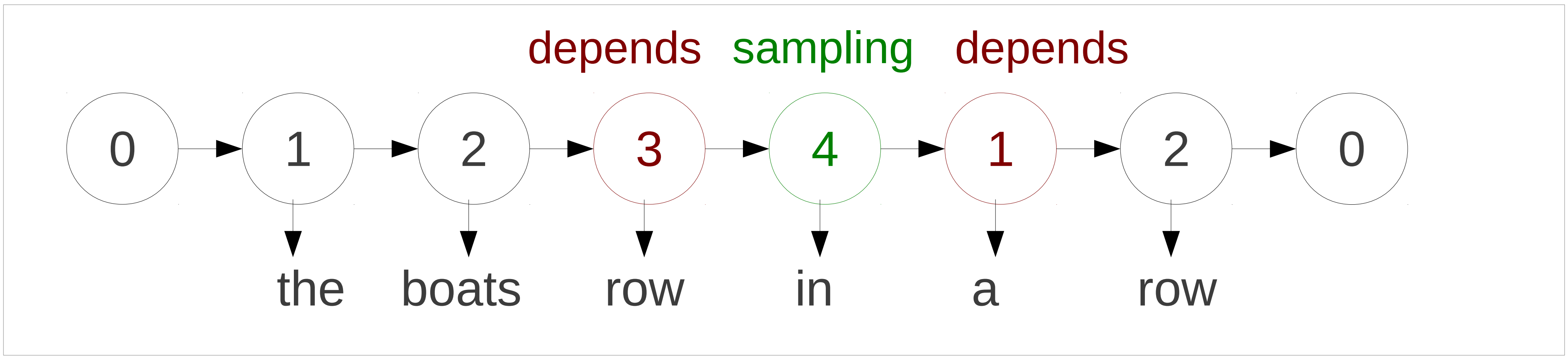
Debugging

- Unit tests! Unit tests! Unit tests!
  - Remove bugs in implementation, and conceptualization
- Create fail-safe function for adding/subtracting counts, terminate if count goes below zero
- When program finishes, remove all samples and make sure the counts are exactly zero
- The likelihood will not always go up, but if it consistently goes down something is probably wrong
- Set the random seed to a single value (srand)

Recent Topics

Block Sampling

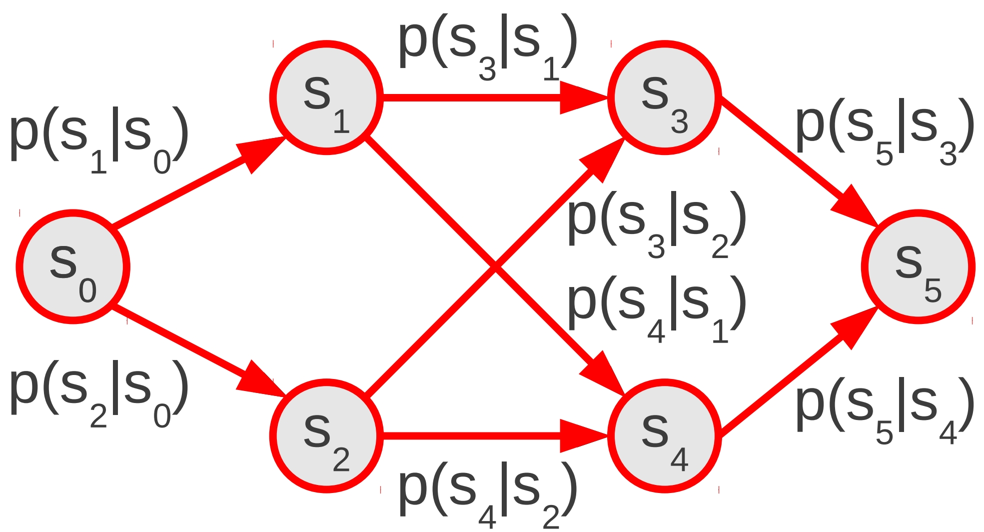
- Often hidden variables depend on each-other strongly



- For example, variables close in time and space
- Block sampling samples multiple hidden variables at a time, considering dependence
- HMMs use forward filtering/backward sampling
  - Context free grammars, etc. also possible

Forward Filtering

- **forward-filtering** adds up probabilities starting from an initial state



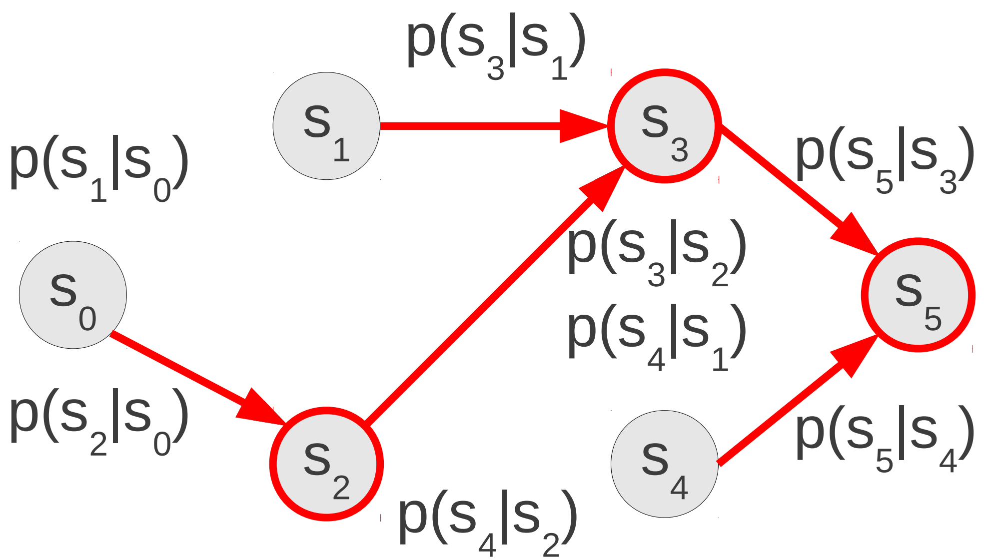
forward filtering

<u>calculate forward probabilities f</u>

$f(s_0) = 1$	$f(s_3) = p(s_3\|s_1)*f(s_1) + p(s_3\|s_2)*f(s_2)$
$f(s_1) = p(s_1\|s_0)*f(s_0)$	$f(s_4) = p(s_4\|s_1)*f(s_1) + p(s_4\|s_2)*f(s_2)$
$f(s_2) = p(s_2\|s_0)*f(s_0)$	$f(s_5) = p(s_5\|s_3)*f(s_3) + p(s_5\|s_4)*f(s_4)$

Backward Sampling

- Backward sampling starts at the acceptance state and samples edges in backwards order



backward sampling
<u>considers edge probs and forward probs</u>

$e(s_5 \rightarrow x)$

$p(x=s_3) \propto p(s_5|s_3)*f(s_3)$

$p(x=s_4) \propto p(s_5|s_4)*f(s_4)$

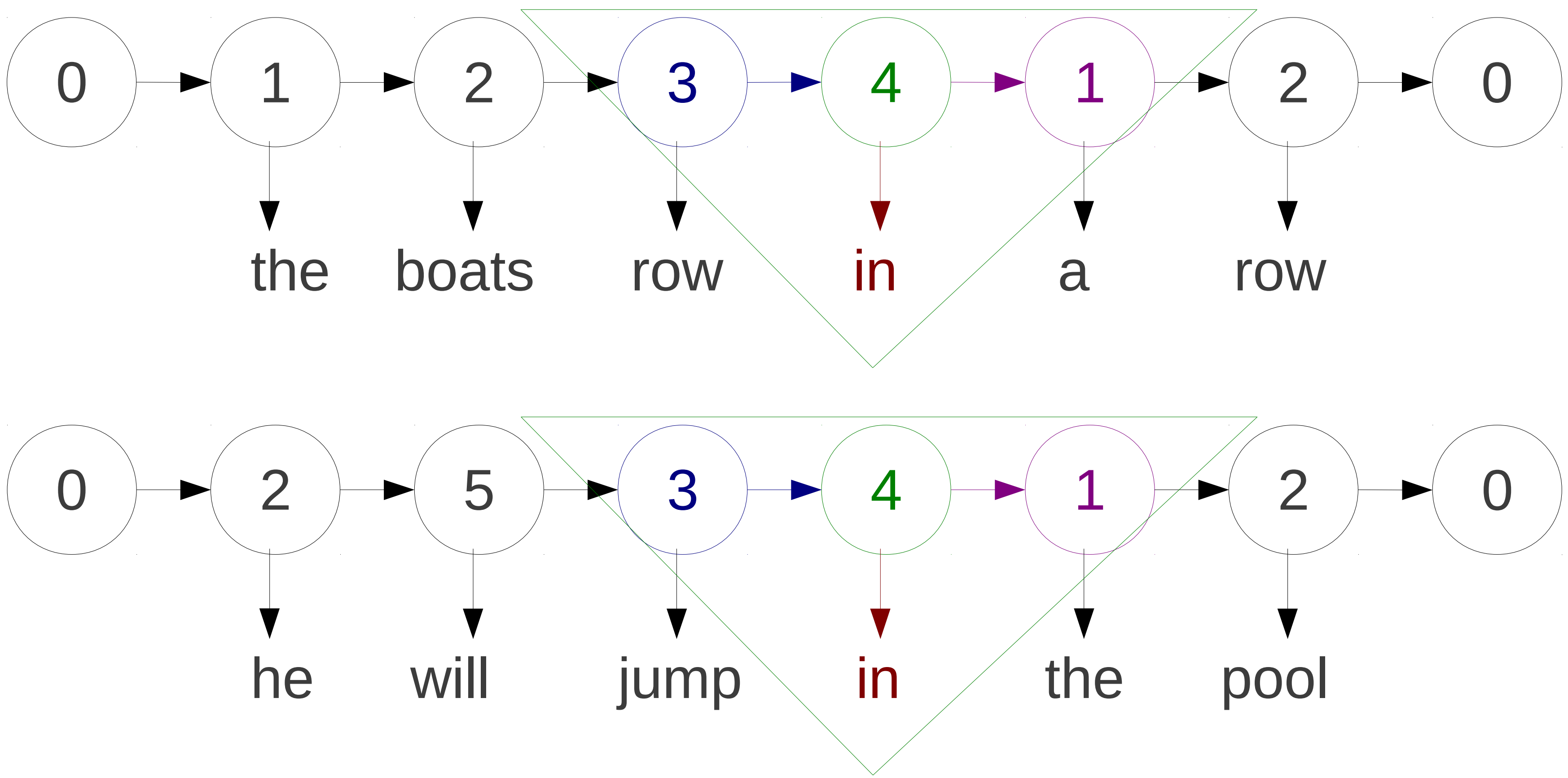
$e(s_3 \rightarrow x)$

$p(x=s_1) \propto p(s_3|s_1)*f(s_1)$

$p(x=s_2) \propto p(s_3|s_2)*f(s_2)$

Type-Based Sampling

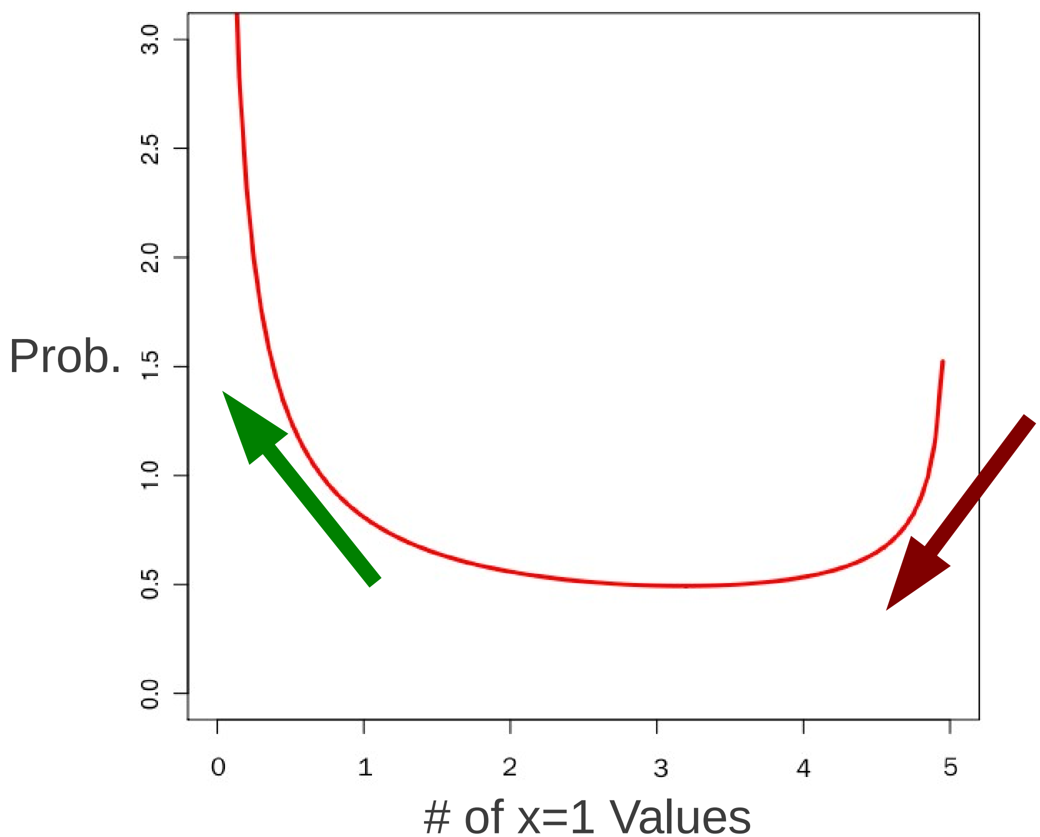
- Sample variables that have the same Markov blanket at once



- Here, the Markov blanket is “3,in,1”

Type-base Sampling

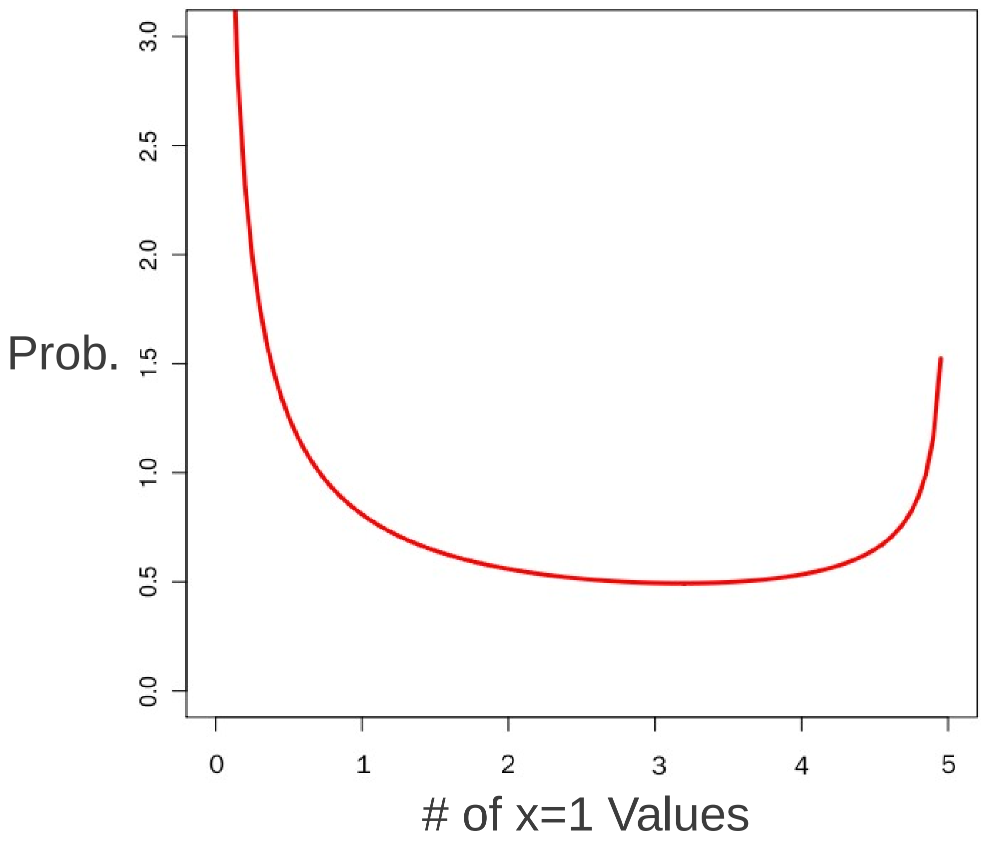
- Models based on Dirichlet distributions tend to assign same tag to similar values (rich-gets-richer)
  - Good for modeling: Induces consistent, compact model
  - Bad for inference: Creates “valleys” in posterior prob



- We are on the right side
- The left side has more probability, but requires several variable changes
- Possible to escape, but takes a very long time

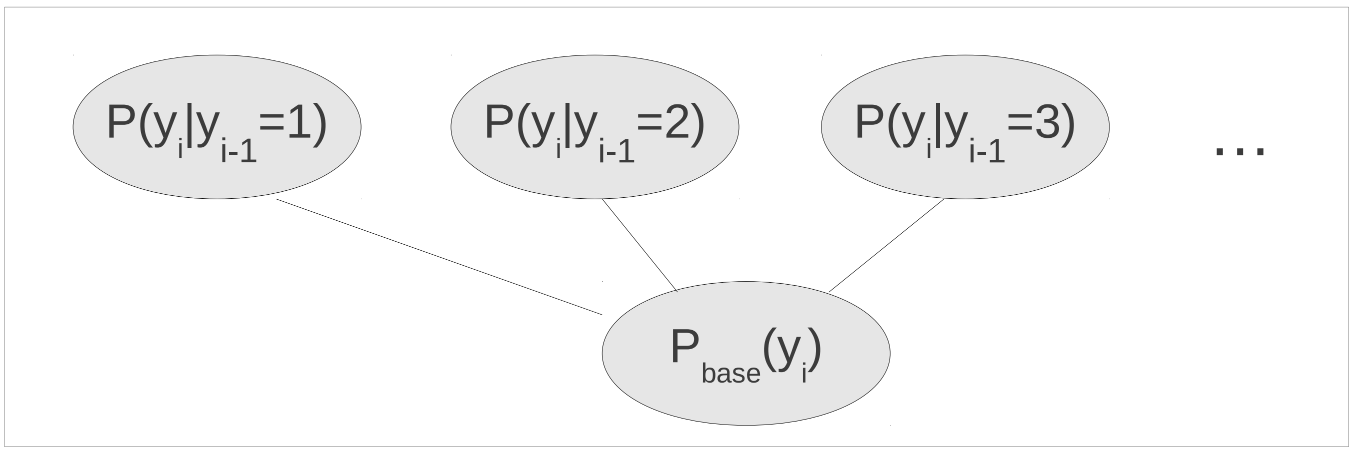
Type-based Sampling

- For each type, sample the number of instances x=1
  - “x=1” has one instance
- Markov blankets are identical, probabilities are also
  - Can set one instance to x=1 randomly, all others to x=2



Hierarchical Models

- Multiple levels using the hierarchical Dirichlet process

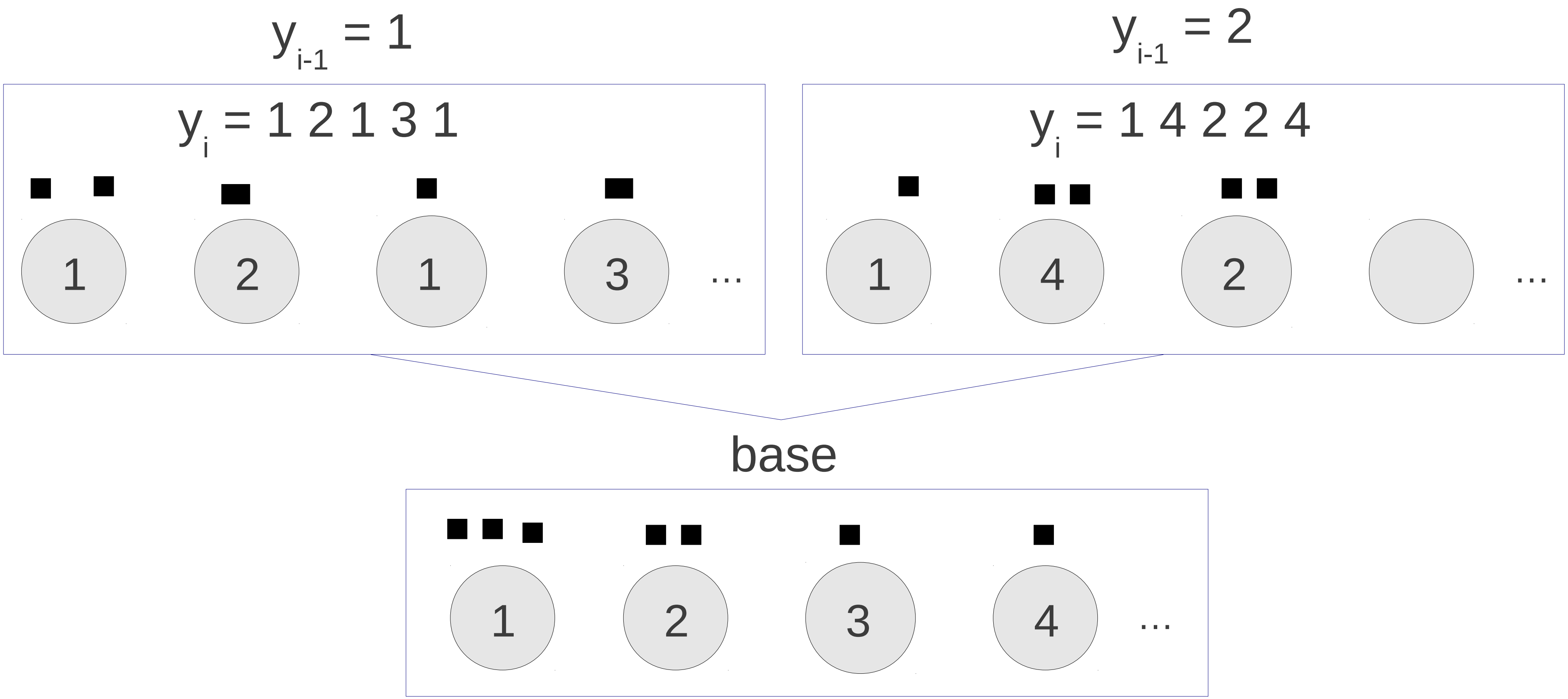


Transition prob: $$P(y_i|y_{i-1}) = \frac{c(y_{i-1} y_i) + \alpha * P_{base}(y_i)}{c(y_{i-1}) + \alpha}$$

Shared base measure: $$P_{base}(y_i) = \frac{c_{base}(y_i) + \alpha * 1/N}{c_{base}(\cdot) + \alpha}$$

Counting c_{base}

- Use the Chinese restaurant process

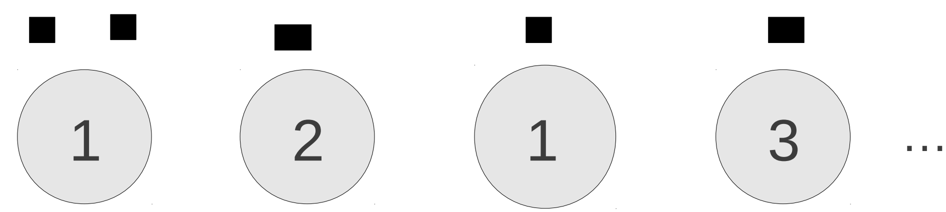


- Add customers to top level for **each data point**, add customers to bottom level for **each table in top level**

Pitman-Yor Process

- Similar to Dirichlet process, but adds table discount d

$$P(x_i) = \frac{c(x_i) - d * t(x_i) + (\alpha + d * t(\cdot)) * P_{base}(x_i)}{c(\cdot) + \alpha}$$



$$P(x_i = 1) = \frac{3 - d * 2 + (\alpha + d * 4) * 0.25}{5 + \alpha}$$

- Similar to absolute discounting for language models
- Able to model power-law distributions, which are common in language

Examples from Speech and Language Processing

Topic Models

- Latent Dirichlet Allocation (LDA) [Blei+ 03]

Collection of Documents	this is a document this is a document this is a document this is a document / this is a document this is a document this is a document this is a document / this is a document this is a document this is a document this is a document
Generate a multinomial topic distribution (with a Dirichlet prior)	Poli. Enter. Sport Econ. Soci. Science { 0.4, 0.05, 0.3, 0.2, 0.01, 0.04}
Generate each word's topic from the topic dist.	1 1 4 3 3 3
Generate each word from the topic's word dist	Bill Clinton buys the Detroit Tigers

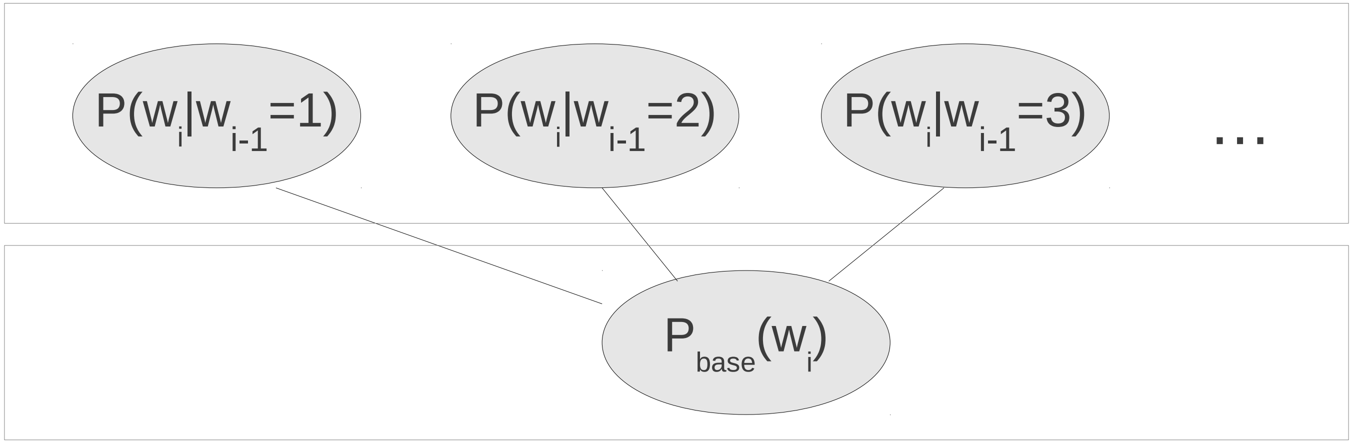
- Infinite topic models [Teh+ 06]
- Applications to computer vision, document clustering, language modeling (e.g.: [Heidel+ 07])

Language Models

- Hierarchical Pitman-Yor language model [Teh 06]

bi-gram

uni-gram



  - Improvements to modeling accuracy by using Pitman-Yor process
  - Similar accuracy to Kneser-Ney
- Used in speech recognition [Huang&Renals 07]

Unsupervised Word Segmentation

- Generate word sequences from 1-gram or 2-gram models [Goldwater+ 09]

Sampling

これ は 単語 で す　　P(単語)

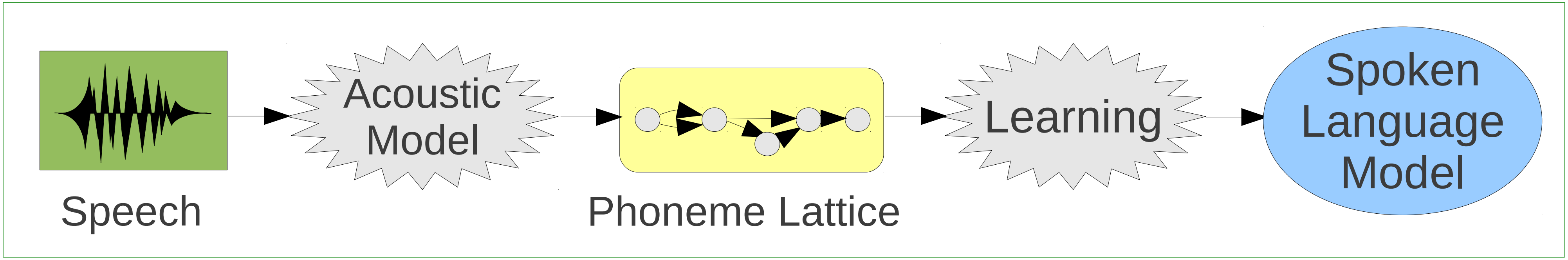
or　　or

これ は 単 語 で す　　P(単)P(語)

- Improvements using block sampling and Pitman-Yor language model [Mochihashi+ 09]

Learning a Language Model from Continuous Speech

- Use Pitman-Yor language model to learn language model and word dictionary from speech [Neubig+ 10]



- Use forward filtering-backward sampling over phoneme lattices
- Can be used for:
  - Learning models for languages with no written text
  - Learning models faithful to spoken language

Learning Various Types of Linguistic Information

- POS using infinite HMM [Beal+ 02]
- CFG [Johnson+ 07] and infinite CFG [Liang+ 07]
- Word and phrase alignment for machine translation [DeNero+ 08, Blunsom+ 09, Neubig+ 11]
- Non-parametric extension of unsupervised semantic parsing [Poon+ 09, Titov+ 11]

References

- M.J. Beal, Z. Ghahramani, and C.E. Rasmussen. 2002. The infinite hidden Markov model. Proceedings of the 16th Annual Conference on Neural Information Processing Systems, 1:577–584.
- David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. Journal of Machine Learning Research, 3:993–1022.
- Phil Blunsom, Trevor Cohn, Chris Dyer, and Miles Osborne. 2009. A Gibbs sampler for phrasal synchronous grammar induction. In Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics, pages 782–790.
- John DeNero, Alex Bouchard-Cˆot´e, and Dan Klein. 2008. Sampling alignment structure under a Bayesian translation model. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, pages 314–323.
- Sharon Goldwater, Thomas L. Griffiths, and Mark Johnson. 2009. A Bayesian framework for word segmentation: Exploring the effects of context. Cognition, 112(1):21–54.
- A. Heidel, H. Chang, and L. Lee. 2007. Language model adaptation using latent Dirichlet allocation and an efficient topic inference algorithm. In Proceedings of the 8th Annual Conference of the International Speech Communication Association (InterSpeech).
- S. Huang and S. Renals. 2007. Hierarchical Pitman-Yor language models for ASR in meetings. In Proceedings of the 2007 IEEE Automatic Speech Recognition and Understanding Workshop, pages 124–129.
- Mark Johnson, Thomas Griffiths, and Sharon Goldwater. 2007. Bayesian inference for PCFGs via Markov chain Monte Carlo. In Proceedings of the Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics, pages 139–146

References

- P. Liang, S. Petrov, M. Jordan, and D. Klein. 2007. The infinite PCFG using hierarchical Dirichlet processes. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, pages 688–697.
- Daichi Mochihashi, Takeshi Yamada, and Naonori Ueda. 2009. Bayesian unsupervised word segmentation with nested Pitman-Yor modeling. In Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics.
- Graham Neubig, Masato Mimura, Shinsuke Mori, and Tatsuya Kawahara. 2010. Learning a language model from continuous speech. In Proceedings of the 11th Annual Conference of the International Speech Communication Association (InterSpeech), Makuhari, Japan, 9.
- Graham Neubig, Taro Watanabe, Eiichiro Sumita, Shinsuke Mori, and Tatsuya Kawahara. 2011. An unsupervised model for joint phrase alignment and extraction. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, Portland, Oregon, USA, 6.
- H. Poon and P. Domingos. 2009. Unsupervised semantic parsing. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, pages 1–10.
- Y.W. Teh, M.I. Jordan, M.J. Beal, and D.M. Blei. 2006. Hierarchical dirichlet processes. Journal of the American Statistical Association, 101(476):1566–1581.
- Yee Whye Teh. 2006. A Bayesian interpretation of interpolated Kneser-Ney. Technical report, School of Computing, National Univ. of Singapore.
- Ivan Titov and Alexandre Klementiev. 2011. A Bayesian model for unsupervised semantic parsing. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics.