CPSC 540: Machine Learning

Independent Component Analysis, Markov Chains

Mark Schmidt

University of British Columbia

Winter 2017

Admin

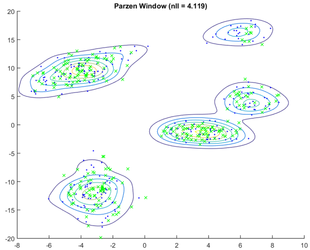
- Assignment 3:
  - Due tonight.
  - 1 late day to hand in Wednesday, 2 for Monday.
- Assignment 4:
  - Due March 20.
- For graduate students planning to graduate in May:
  - Send me a private message on Piazza ASAP.

Last Time: Kernel Density Estimation

- We discussed kernel density estimation,

$$p(x) = \frac{1}{n}\sum_{i=1}^{n} k_R(x - x^i),$$

a mixture of simple densities k_R centered on each example.



- Flexible class of density models, though sensitive to bandwidth R.

Last Time: Probabilistic PCA and Factor Analysis

- PCA is limit of a continuous mixture model under Gaussian assumptions,

$$x|z \sim \mathcal{N}(W^T z, \sigma^2 I), \quad z \sim \mathcal{N}(0, I),$$

as $\sigma \to 0$.

Last Time: Probabilistic PCA and Factor Analysis

- PCA is limit of a continuous mixture model under Gaussian assumptions,

$$x|z \sim \mathcal{N}(W^T z, \sigma^2 I), \quad z \sim \mathcal{N}(0, I),$$

  as $\sigma \to 0$.

- Factor analysis (FA) generalizes to diagonal covariance D,

$$x|z \sim \mathcal{N}(W^T z, D), \quad z \sim \mathcal{N}(0, I),$$

  where W and D are estimated from data.

- Both are 100+ years old with tons of applications.
  - Classic tools for dividing data into "parts" and visualizing high-dimensional data.

- Probabilistic perspective allows us to do things like mixture of factor analyses.

Orthogonality and Sequential Fitting

- The PCA and FA solutions are not unique.

- Common heuristic:
    1. Enforce that rows of W have a norm of 1.
    2. Enforce that rows of W are orthogonal.
    3. Fit the rows of W sequentially.
- This leads to a unique solution up to sign changes.

Orthogonality and Sequential Fitting

- The PCA and FA solutions are not unique.
- Common heuristic:
  1. Enforce that rows of W have a norm of 1.
  2. Enforce that rows of W are orthogonal.
  3. Fit the rows of W sequentially.
- This leads to a unique solution up to sign changes.
- But there are other ways to resolve non-uniqueness (Murphy's Section 12.1.3):
  - Force W to be lower-triangular.
  - Choose an informative rotation.
  - Use a non-Gaussian prior.

Outline

1. Independent Component Analysis
2. Markov Chains
3. Monte Carlo Methods

Motivation for Independent Component Analysis (ICA)

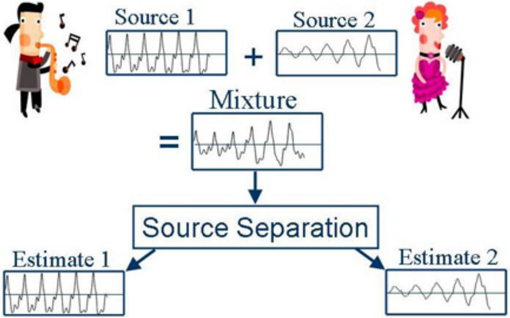
- Factor analysis has found an enormous number of applications.
  - People really want to find the "factors" that make up their data.
- But factor analysis can't even identify factor directions.
  - We can rotate W and obtain the same model.

Motivation for Independent Component Analysis (ICA)

- Factor analysis has found an enormous number of applications.
  - People really want to find the "factors" that make up their data.
- But factor analysis can't even identify factor directions.
  - We can rotate W and obtain the same model.
- Independent component analysis (ICA) is a more recent approach (≈ 30 years).
  - Under certain assumptions, it can identify factors.
- The canonical application of ICA is blind source separation.

Blind Source Separation

- In blind source separation we have microphones recording multiple sources.



http://music.eecs.northwestern.edu/research.php

- Goal is to reconstruct sources (factors) from the measurements.

Independent Component Analysis Applications

- ICA is replacing PCA/FA in many applications.

Some ICA applications are listed below:[1]

- optical Imaging of neurons[17]
- neuronal spike sorting[18]
- face recognition[19]
- modeling receptive fields of primary visual neurons[20]
- predicting stock market prices[21]
- mobile phone communications [22]
- color based detection of the ripeness of tomatoes[23]
- removing artifacts, such as eye blinks, from EEG data.[24]

- Recent work shows that ICA can often resolve direction of causality.

Limitations of Matrix Factorization

- As in PCA/FA, ICA is a matrix factorization method,

$$X \approx ZW.$$

- Let's assume that $X = ZW$ for a "true" W with $k = d$.

Limitations of Matrix Factorization

- As in PCA/FA, ICA is a matrix factorization method,

$$X \approx ZW.$$

- Let's assume that $X = ZW$ for a "true" W with $k = d$.

- The 3 issues stopping us from finding "true" W:
  1. Label switching: get same model if we permute rows of W.
     - We can exchange row 1 and 2 of W (and same columns of Z).
     - Not a problem because we don't care about order of factors.

Limitations of Matrix Factorization

- As in PCA/FA, ICA is a matrix factorization method,

$$X \approx ZW.$$

- Let's assume that $X = ZW$ for a "true" W with $k = d$.

- The 3 issues stopping us from finding "true" W:
  1. Label switching: get same model if we permute rows of W.
     - We can exchange row 1 and 2 of W (and same columns of Z).
     - Not a problem because we don't care about order of factors.
  2. Scaling: get same model if multiply rows of W by constant.
     - If we multiply row 1 of W by α, could multiply column 1 of Z by $1/\alpha$.
     - Can't identify scale/sign, but might hope to identify direction.

Limitations of Matrix Factorization

- As in PCA/FA, ICA is a matrix factorization method,

$$X \approx ZW.$$

- Let's assume that $X = ZW$ for a "true" W with $k = d$.

- The 3 issues stopping us from finding "true" W:
  1. Label switching: get same model if we permute rows of W.
     - We can exchange row 1 and 2 of W (and same columns of Z).
     - Not a problem because we don't care about order of factors.
  2. Scaling: get same model if multiply rows of W by constant.
     - If we multiply row 1 of W by α, could multiply column 1 of Z by $1/\alpha$.
     - Can't identify scale/sign, but might hope to identify direction.
  3. Rotataion: we the get same model if we pre-multiply W by orthogonal Q.
     - Because PCA/FA only depend on W^TW, which equals $(QW)^T(QW)$.

Limitations of Matrix Factorization

- As in PCA/FA, ICA is a matrix factorization method,

$$X \approx ZW.$$

- Let's assume that $X = ZW$ for a "true" W with $k = d$.
- The 3 issues stopping us from finding "true" W:
  1. Label switching: get same model if we permute rows of W.
     - We can exchange row 1 and 2 of W (and same columns of Z).
     - Not a problem because we don't care about order of factors.
  2. Scaling: get same model if multiply rows of W by constant.
     - If we multiply row 1 of W by α, could multiply column 1 of Z by $1/\alpha$.
     - Can't identify scale/sign, but might hope to identify direction.
  3. Rotataion: we the get same model if we pre-multiply W by orthogonal Q.
     - Because PCA/FA only depend on W^TW, which equals $(QW)^T(QW)$.
- If we could address rotation, we could identify the directions.

Another Unique Gaussian Property

- Consider density written as a product of independent factors,

$$p(z) = \prod_{c=1}^{k} p_c(z_c).$$

- If $p(z)$ is rotation-invariant, $p(Qz) = p(z)$, then it must be Gaussian.

Another Unique Gaussian Property

- Consider density written as a product of independent factors,

$$p(z) = \prod_{c=1}^{k} p_c(z_c).$$

- If $p(z)$ is rotation-invariant, $p(Qz) = p(z)$, then it must be Gaussian.

- The (non-intuitive) magic behind ICA:
  - If product of independent factors is non-Gaussian, it isn't rotationally symmetric.

Another Unique Gaussian Property

- Consider density written as a product of independent factors,

$$p(z) = \prod_{c=1}^{k} p_c(z_c).$$

- If $p(z)$ is rotation-invariant, $p(Qz) = p(z)$, then it must be Gaussian.
- The (non-intuitive) magic behind ICA:
  - If product of independent factors is non-Gaussian, it isn't rotationally symmetric.
- Implication: if at most 1 factor is Gaussian, we can identify them.
  - Up to permutation/sign/scaling (other rotations change distribution).

Independent Component Analysis

- In ICA we use the approximation,

$$X \approx WZ$$

  where we want z_j to be non-Gaussian.

- A common strategy is maximum likelihood ICA assuming a heavy-tailed z_j like

$$p(z_j) = \frac{1}{\pi(\exp(z_j) + \exp(-z_j))}.$$

Independent Component Analysis

- In ICA we use the approximation,

$$X \approx WZ$$

where we want z_j to be non-Gaussian.

- A common strategy is maximum likelihood ICA assuming a heavy-tailed z_j like

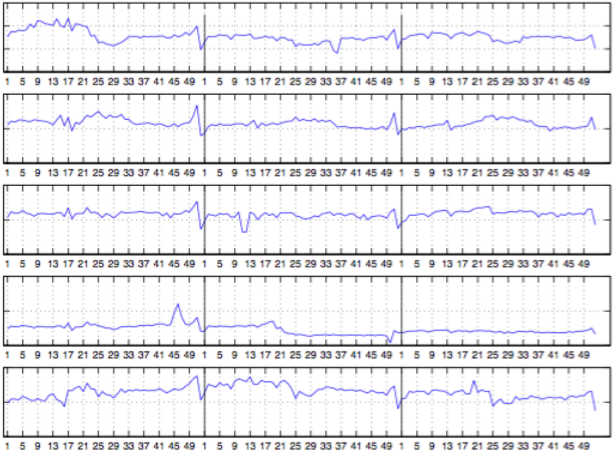
$$p(z_j) = \frac{1}{\pi(\exp(z_j) + \exp(-z_j))}.$$

- Another common strategy fits data while maximizing measure of non-Gaussianity:
  - Maximize kurtosis, which is 0 for Gaussians.
  - Minimimize entropy, which is maximized with Gaussians.

- The fastICA method is a popular Newton-like method maximizing kurtosis.

ICA on Retail Purchase Data

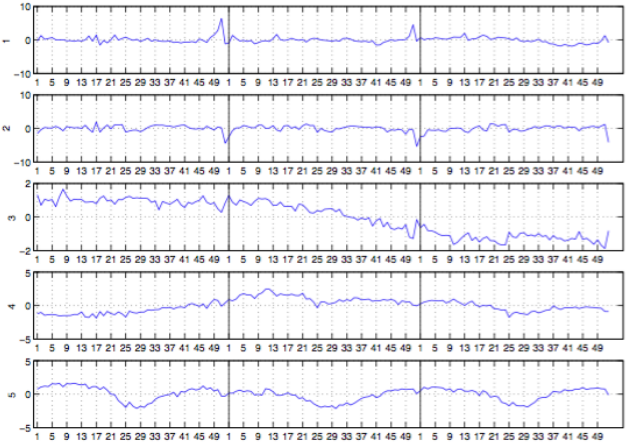
- Cash flow from different stores over 3 years:



http://www.stat.ucla.edu/~yuille/courses/Stat161-261-Spring14/Hyv000-icatut.pdf

ICA on Retail Purchase Data

- Factors found using ICA:



http://www.stat.ucla.edu/~yuille/courses/Stat161-261-Spring14/Hyv000-icatut.pdf

Outline

1 Independent Component Analysis

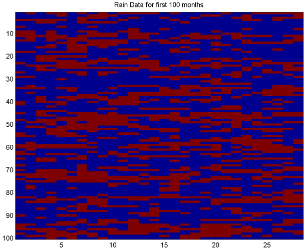
2 Markov Chains

3 Monte Carlo Methods

Example: Vancouver Rain Data

- Consider density estimation on "Vancouver Rain" dataset (first 100 examples):

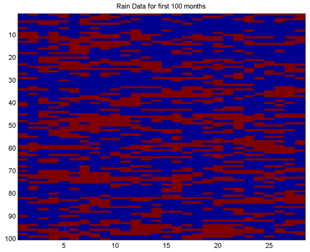
$$X = \left[\quad \right].$$



- Variable x_j^i is whether or not it rained on day j in month i.
  - Each row is a month, each column is a day of the month.
  - Data ranges from 1896-2004.

Example: Vancouver Rain Data

- Consider density estimation on "Vancouver Rain" dataset (first 100 examples):

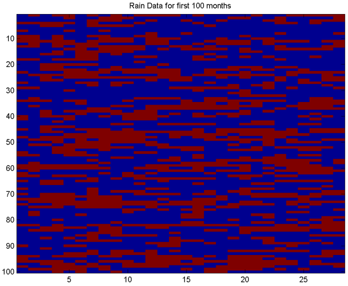
$X =$ 

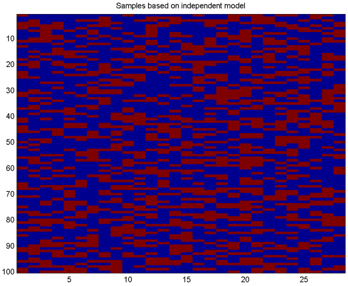
- Variable x_j^i is whether or not it rained on day j in month i.
  - Each row is a month, each column is a day of the month.
  - Data ranges from 1896-2004.

- Without extra information (like "July"), days have simple relationship:
  - If it rained yesterday, it's likely to rain today ($> 50\%$ chance of $(x_j == x_{j-1})$).

Example: Vancouver Rain Data

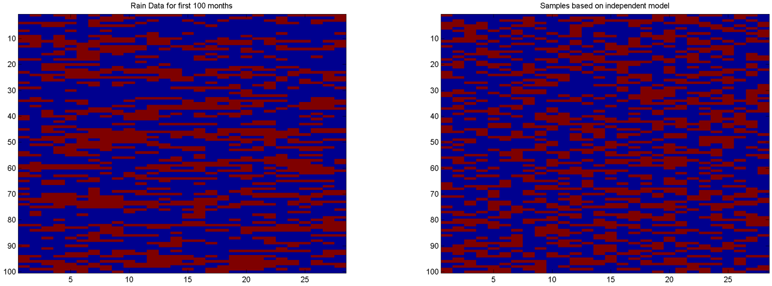
- With independent Bernoullis, we get $p(x_j^i = \text{“rain”}) \approx 0.41$ (sadly).
  - Real data vs. independent Bernoulli model:





Example: Vancouver Rain Data

- With independent Bernoullis, we get $p(x_j^i = \text{"rain"}) \approx 0.41$ (sadly).
  - Real data vs. independent Bernoulli model:



- Independent model misses correlations between days.
- Mixture of Bernoullis could model correlation, but it's inefficient:
  - "Position independence" of correlation would need lots of mixtures.

Markov Chains

- A better density model for this data is a Markov chain.

$$p(x_1, x_2, \ldots, x_d) = p(x_1)p(x_2|x_1)p(x_3|x_2)\cdots p(x_d|x_{d-1})$$

Markov Chains

- A better density model for this data is a Markov chain.

$$p(x_1, x_2, \dots, x_d) = p(x_1)p(x_2|x_1)p(x_3|x_2)\cdots p(x_d|x_{d-1})$$
$$= \underbrace{p(x_1)}_{\text{initial prob.}} \prod_{j=2}^{d} \underbrace{p(x_j|x_{j-1})}_{\text{transition prob.}},$$

  where I'm using x_j as short for x_j^i for a generic i.
- Models dependency of feature on previous feature.
  - Assuming a meaningful ordering of features.

Markov Chains

- A better density model for this data is a Markov chain.

$$\begin{aligned} p(x_1, x_2, \dots, x_d) &= p(x_1)p(x_2|x_1)p(x_3|x_2)\cdots p(x_d|x_{d-1}) \\ &= \underbrace{p(x_1)}_{\text{initial prob.}} \prod_{j=2}^{d} \underbrace{p(x_j|x_{j-1})}_{\text{transition prob.}}, \end{aligned}$$

  where I'm using x_j as short for x_j^i for a generic i.
- Models dependency of feature on previous feature.
  - Assuming a meaningful ordering of features.

- Makes a strong conditional independence assumption ("Markov property"),

$$p(x_j|x_{j-1}, x_{j-2}, \dots, x_1) = p(x_j|x_{j-1}),$$

  that the last "time" x_{j-1} tells us everything we need to know about the "past".
  - What we want for the rain data.

Homogenous Markov Chains

- For rain data it makes sense to use a homogeneous Markov chain:
  - Transition probabilities $p(x_j|x_{j-1})$ are the same for all j.

Homogenous Markov Chains

- For rain data it makes sense to use a homogeneous Markov chain:
  - Transition probabilities $p(x_j|x_{j-1})$ are the same for all j.

- MLE for discrete x_j values is given by

$$\theta_{x_j,x_{j-1}} = \frac{(\text{number of transitions from } x_{j-1} \text{ to } x_j)}{(\text{number of times we saw } x_{j-1} \text{ for } j > 1)},$$

and we use the same $\theta_{x_j,x_{j-1}}$ for all j.

Homogenous Markov Chains

- For rain data it makes sense to use a homogeneous Markov chain:
  - Transition probabilities $p(x_j|x_{j-1})$ are the same for all j.

- MLE for discrete x_j values is given by

$$\theta_{x_j,x_{j-1}} = \frac{(\text{number of transitions from } x_{j-1} \text{ to } x_j)}{(\text{number of times we saw } x_{j-1} \text{ for } j > 1)},$$

  and we use the same $\theta_{x_j,x_{j-1}}$ for all j.

- A special case of the general idea of parameter tieing:
  - "Making different parts of the model use the same parameters."

Homogenous Markov Chains

- For rain data it makes sense to use a homogeneous Markov chain:
  - Transition probabilities $p(x_j|x_{j-1})$ are the same for all j.

- MLE for discrete x_j values is given by

$$\theta_{x_j,x_{j-1}} = \frac{(\text{number of transitions from } x_{j-1} \text{ to } x_j)}{(\text{number of times we saw } x_{j-1} \text{ for } j > 1)},$$

  and we use the same $\theta_{x_j,x_{j-1}}$ for all j.

- A special case of the general idea of parameter tieing:
  - "Making different parts of the model use the same parameters."

- Advantages:
  1. You have more data available to estimate each parameter.
     - Don't need to independently learn $p(x_j|x_{j-1})$ for days 14 and 15.

Homogenous Markov Chains

- For rain data it makes sense to use a homogeneous Markov chain:
    - Transition probabilities $p(x_j|x_{j-1})$ are the same for all j.

- MLE for discrete x_j values is given by

$$\theta_{x_j,x_{j-1}} = \frac{(\text{number of transitions from } x_{j-1} \text{ to } x_j)}{(\text{number of times we saw } x_{j-1} \text{ for } j > 1)},$$

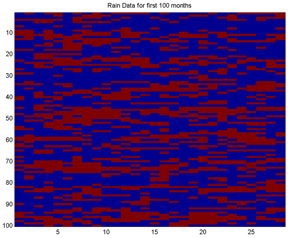
and we use the same $\theta_{x_j,x_{j-1}}$ for all j.

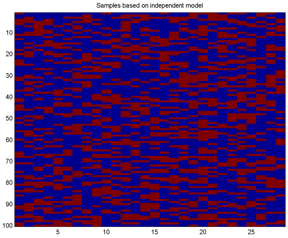
- A special case of the general idea of parameter tieing:
    - "Making different parts of the model use the same parameters."

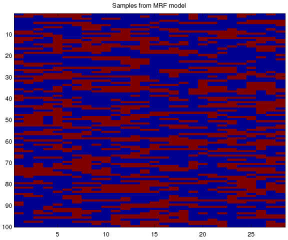
- Advantages:
    1. You have more data available to estimate each parameter.
        - Don't need to independently learn $p(x_j|x_{j-1})$ for days 14 and 15.
    2. You can have models of different sizes.
        - Same model can be used for months with 28, 29, 30, or 31 days.

Homogeneous Markov Chain for Rain Data

- Real vs. independent vs. homogeneous Markov chain:

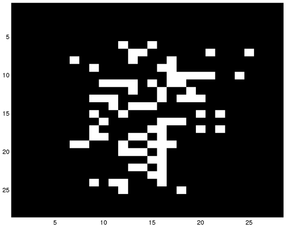


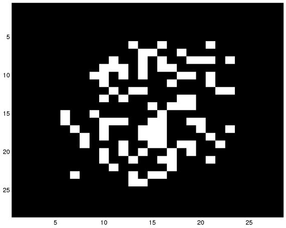


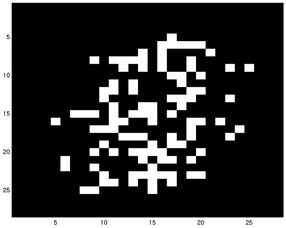


Density Estimation for MNIST Digits

- We've previously considered density estimation for images of digits.
- We saw that independent Bernoullis do terrible

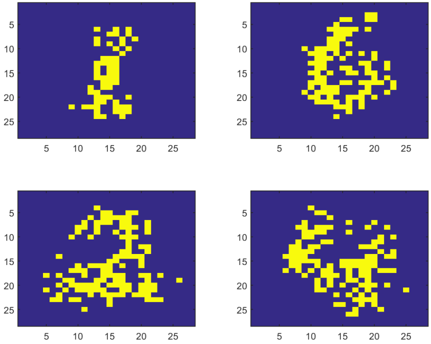






Density Estimation for MNIST Digits

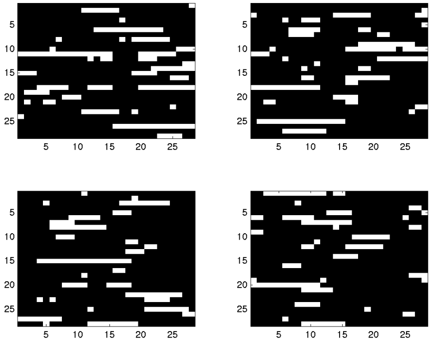
- We can do a bit better with mixture of 10 Bernoullis:



- The shape is looking better, but it's missing correlation between adjacent pixels.
    - Could we capture this with a Markov chain?

Density Estimation for MNIST Digits

- Samples from a homogeneous Markov chain (putting rows into one long vector):



- This captures correlations within rows, but misses dependencies between rows.
    - "Position independence" of homogeneity means it loses position information.

Inhomogeneous Markov Chains

- Markov chains allow a different $p(x_j|x_{j-1})$ for each j.

Inhomogeneous Markov Chains

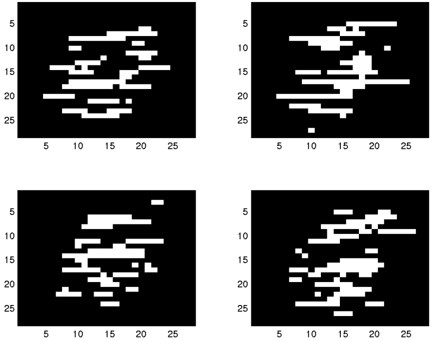
- Markov chains allow a different $p(x_j | x_{j-1})$ for each j.
- MLE for discrete x_j values is given by

$$\theta^{j}_{x_j, x_{j-1}} = \frac{(\text{number of transitions from } x_{j-1} \text{ to } x_j \text{ starting at } (j-1))}{(\text{number of times we saw } x_{j-1} \text{ at position } (j-1))},$$

- Such inhomogeneous Markov chains include independent models as special case:
  - We could set $p(x_j | x_{j-1}) = p(x_j)$.

Density Estimation for MNIST Digits

- Samples from an inhomogeneous Markov chain:



- We now have correlations within rows and position information.
  - But Markov assumption isn't capturing dependency between rows.
  - Next time we'll discuss graphical models which address this.
  - You could alternately consider mixture of Markov chains.

Fun with Markov Chains

- Markov Chains from "Explained Visually":
  `http://setosa.io/ev/markov-chains`

- Modeling Snakes and Ladders as a Markov chain:
  `http://datagenetics.com/blog/november12011/index.html`

- Modeling Candyland as Markov chain:
  `http://www.datagenetics.com/blog/december12011/index.html`

- Modeling Yahtzee as a Markov chain:
  `http://www.datagenetics.com/blog/january42012/`

Outline

1. Independent Component Analysis
2. Markov Chains
3. Monte Carlo Methods

Sampling from a Markov Chains

- Generating samples from a density model allows us to see what it's learned.
- To sample from a mixture model we used:
  - Sample cluster z^i, then sample x^i based on the cluster parameters.

Sampling from a Markov Chains

- Generating samples from a density model allows us to see what it's learned.

- To sample from a mixture model we used:
  - Sample cluster z^i, then sample x^i based on the cluster parameters.

- To sample from a Markov chain we:
  1. Sample x_1 from initial probabilities $p(x_1)$.

Sampling from a Markov Chains

- Generating samples from a density model allows us to see what it's learned.
- To sample from a mixture model we used:
  - Sample cluster z^i, then sample x^i based on the cluster parameters.
- To sample from a Markov chain we:
  1. Sample x_1 from initial probabilities $p(x_1)$.
  2. Given x_1, sample x_2 from transition probabilities $p(x_2|x_1)$.

Sampling from a Markov Chains

- Generating samples from a density model allows us to see what it's learned.

- To sample from a mixture model we used:
  - Sample cluster z^i, then sample x^i based on the cluster parameters.

- To sample from a Markov chain we:
  1. Sample x_1 from initial probabilities $p(x_1)$.
  2. Given x_1, sample x_2 from transition probabilities $p(x_2|x_1)$.
  3. Given x_2, sample x_3 from transition probabilities $p(x_3|x_2)$.

Sampling from a Markov Chains

- Generating samples from a density model allows us to see what it's learned.

- To sample from a mixture model we used:
  - Sample cluster z^i, then sample x^i based on the cluster parameters.

- To sample from a Markov chain we:
  1. Sample x_1 from initial probabilities $p(x_1)$.
  2. Given x_1, sample x_2 from transition probabilities $p(x_2|x_1)$.
  3. Given x_2, sample x_3 from transition probabilities $p(x_3|x_2)$.
  4. ...
  5. Given x_{d-1}, sample x_d from transition probabilities $p(x_d|x_{d-1})$.

Sampling from a Markov Chains

- Generating samples from a density model allows us to see what it's learned.
- To sample from a mixture model we used:
  - Sample cluster z^i, then sample x^i based on the cluster parameters.
- To sample from a Markov chain we:
  1. Sample x_1 from initial probabilities $p(x_1)$.
  2. Given x_1, sample x_2 from transition probabilities $p(x_2|x_1)$.
  3. Given x_2, sample x_3 from transition probabilities $p(x_3|x_2)$.
  4. ...
  5. Given x_{d-1}, sample x_d from transition probabilities $p(x_d|x_{d-1})$.
- This is called ancestral sampling.
  - It's easy if probabilities have nice form, and we know how to sample in 1D...

Sampling from a 1D Discrete Distribution

- Sampling methods assume we can sample uniformly over $[0, 1]$.
  - Usually, a "pseudo-random" number generator is good enough (like Matlab's *rand*).

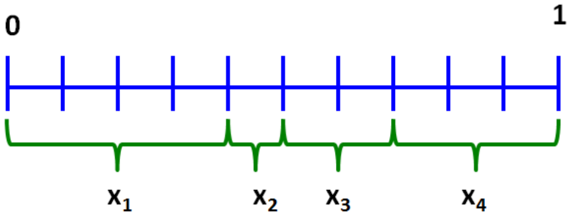
Sampling from a 1D Discrete Distribution

- Sampling methods assume we can sample uniformly over $[0, 1]$.
  - Usually, a "pseudo-random" number generator is good enough (like Matlab's *rand*).

- To sample from a discrete distribution like

$$p(X = 1) = 0.4, \quad p(X = 2) = 0.1, \quad p(X = 3) = 0.2, \quad p(X = 4) = 0.3,$$

we can divide up the $[0, 1]$ interval based on probability values:



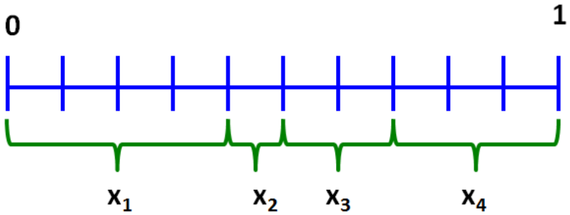
Sampling from a 1D Discrete Distribution

- Sampling methods assume we can sample uniformly over $[0, 1]$.
  - Usually, a "pseudo-random" number generator is good enough (like Matlab's *rand*).

- To sample from a discrete distribution like

$$p(X = 1) = 0.4, \quad p(X = 2) = 0.1, \quad p(X = 3) = 0.2, \quad p(X = 4) = 0.3,$$

we can divide up the $[0, 1]$ interval based on probability values:



- If $u \sim \mathcal{U}(0, 1)$, 40% of the time it lands in x_1 region, 10% of time in x_2, and so on.

Sampling from a 1D Discrete Distribution

- Sampling methods assume we can sample uniformly over $[0, 1]$.
    - Usually, a "pseudo-random" number generator is good enough (like Matlab's *rand*).

- To sample from a discrete distribution like

$$p(X = 1) = 0.4, \quad p(X = 2) = 0.1, \quad p(X = 3) = 0.2, \quad p(X = 4) = 0.3,$$

we can use the following procedure (*sampleDiscrete.m*):

1. Generate $u \sim \mathcal{U}(0, 1)$.

Sampling from a 1D Discrete Distribution

- Sampling methods assume we can sample uniformly over $[0, 1]$.
    - Usually, a "pseudo-random" number generator is good enough (like Matlab's *rand*).

- To sample from a discrete distribution like

$$p(X = 1) = 0.4, \quad p(X = 2) = 0.1, \quad p(X = 3) = 0.2, \quad p(X = 4) = 0.3,$$

we can use the following procedure (*sampleDiscrete.m*):

1. Generate $u \sim \mathcal{U}(0, 1)$.
2. If $u \leq p(X \leq 1)$, output 1.

Sampling from a 1D Discrete Distribution

- Sampling methods assume we can sample uniformly over $[0, 1]$.
  - Usually, a "pseudo-random" number generator is good enough (like Matlab's *rand*).

- To sample from a discrete distribution like

$$p(X = 1) = 0.4, \quad p(X = 2) = 0.1, \quad p(X = 3) = 0.2, \quad p(X = 4) = 0.3,$$

  we can use the following procedure (*sampleDiscrete.m*):

  1. Generate $u \sim \mathcal{U}(0, 1)$.
  2. If $u \leq p(X \leq 1)$, output 1.
  3. If $u \leq p(X \leq 2)$, output 2.

Sampling from a 1D Discrete Distribution

- Sampling methods assume we can sample uniformly over $[0, 1]$.
    - Usually, a "pseudo-random" number generator is good enough (like Matlab's *rand*).

- To sample from a discrete distribution like

$$p(X = 1) = 0.4, \quad p(X = 2) = 0.1, \quad p(X = 3) = 0.2, \quad p(X = 4) = 0.3,$$

we can use the following procedure (*sampleDiscrete.m*):

1. Generate $u \sim \mathcal{U}(0, 1)$.
2. If $u \leq p(X \leq 1)$, output 1.
3. If $u \leq p(X \leq 2)$, output 2.
4. If $u \leq p(X \leq 3)$, output 3.
5. Otherwise, output 4.

Sampling from a 1D Discrete Distribution

- Sampling methods assume we can sample uniformly over $[0, 1]$.
  - Usually, a "pseudo-random" number generator is good enough (like Matlab's *rand*).

- To sample from a discrete distribution like

$$p(X = 1) = 0.4, \quad p(X = 2) = 0.1, \quad p(X = 3) = 0.2, \quad p(X = 4) = 0.3,$$

  we can use the following procedure (*sampleDiscrete.m*):
  1. Generate $u \sim \mathcal{U}(0, 1)$.
  2. If $u \leq p(X \leq 1)$, output 1.
  3. If $u \leq p(X \leq 2)$, output 2.
  4. If $u \leq p(X \leq 3)$, output 3.
  5. Otherwise, output 4.
- With k states, cost to generate a sample is $O(k)$.

Sampling from a 1D Discrete Distribution

- Sampling methods assume we can sample uniformly over $[0, 1]$.
  - Usually, a "pseudo-random" number generator is good enough (like Matlab's *rand*).

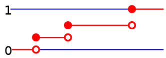
- To sample from a discrete distribution like

$$p(X = 1) = 0.4, \quad p(X = 2) = 0.1, \quad p(X = 3) = 0.2, \quad p(X = 4) = 0.3,$$

  we can use the following procedure (*sampleDiscrete.m*):
  1. Generate $u \sim \mathcal{U}(0, 1)$.
  2. If $u \leq p(X \leq 1)$, output 1.
  3. If $u \leq p(X \leq 2)$, output 2.
  4. If $u \leq p(X \leq 3)$, output 3.
  5. Otherwise, output 4.
- With k states, cost to generate a sample is $O(k)$.
- You can go faster if you're generating multiple samples:
  - One-time $O(k)$ cost to store the $p(X \leq c)$ for all c.
  - Per-sample $O(\log k)$ cost to do binary search for smallest $u \leq p(X \leq c)$.

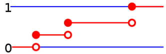
Inverse Transform Method (Exact 1D Sampling)

- Recall that the cumulative distribution function (CDF) F is $p(X \leq x)$.
    - $F(x)$ is between 0 and 1 a gives proportion of times X is below x.

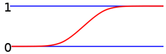


Inverse Transform Method (Exact 1D Sampling)

- Recall that the cumulative distribution function (CDF) F is $p(X \leq x)$.
    - $F(x)$ is between 0 and 1 a gives proportion of times X is below x.



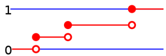
- We can also use the CDF to sample from continuous variables.



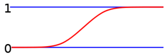
https://en.wikipedia.org/wiki/Cumulative_distribution_function

Inverse Transform Method (Exact 1D Sampling)

- Recall that the cumulative distribution function (CDF) F is $p(X \leq x)$.
  - $F(x)$ is between 0 and 1 a gives proportion of times X is below x.



  - We can also use the CDF to sample from continuous variables.

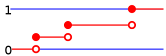


https://en.wikipedia.org/wiki/Cumulative_distribution_function

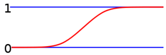
- The inverse CDF (or quantile function) F^{-1} is its inverse:
  - Given a number u between 0 and 1, gives x such that $p(X \leq x) = u$.

Inverse Transform Method (Exact 1D Sampling)

- Recall that the cumulative distribution function (CDF) F is $p(X \leq x)$.
  - $F(x)$ is between 0 and 1 a gives proportion of times X is below x.



  - We can also use the CDF to sample from continuous variables.



https://en.wikipedia.org/wiki/Cumulative_distribution_function

- The inverse CDF (or quantile function) F^{-1} is its inverse:
  - Given a number u between 0 and 1, gives x such that $p(X \leq x) = u$.
- Inverse transfrom method for exact sampling in 1D:
  1. Sample $u \sim \mathcal{U}(0,1)$.
  2. Compute $x = F^{-1}(u)$.

Sampling from a 1D Gaussian

- Consider a Gaussian distribution,

$$x \sim \mathcal{N}(\mu, \sigma^2).$$

- CDF has the form

$$F(x) = p(X \leq x) = \frac{1}{2}\left[1 + \text{erf}\left(\frac{x-\mu}{\sigma\sqrt{2}}\right)\right],$$

where erf the CDF of $\mathcal{N}(0, 1)$.

Sampling from a 1D Gaussian

- Consider a Gaussian distribution,

$$x \sim \mathcal{N}(\mu, \sigma^2).$$

- CDF has the form

$$F(x) = p(X \leq x) = \frac{1}{2}\left[1 + \mathsf{erf}\left(\frac{x-\mu}{\sigma\sqrt{2}}\right)\right],$$

  where erf the CDF of $\mathcal{N}(0,1)$.
- Inverse CDF has the form

$$F^{-1}(u) = \mu + \sigma\sqrt{2}\mathsf{erf}^{-1}(2u-1).$$

- To sample from a Gaussian:
  1. Generate $u \sim \mathcal{U}(0,1)$.
  2. Compute $F^{-1}(u)$.

Inference in Markov Chains

- Given density esimator, we often want probabilistic inferences like computing
  - Marginals: what is the probability that $x_j = c$?
  - Conditionals: if it rains today, what is the probability it will rain in 5 days?

Inference in Markov Chains

- Given density esimator, we often want probabilistic inferences like computing
  - Marginals: what is the probability that $x_j = c$?
  - Conditionals: if it rains today, what is the probability it will rain in 5 days?
- Easy for independent models: we *have* marginals $p(x_j)$ and $p(x_j | x_{j'}) = p(x_j)$.
  - Also easy for mixtures of independent models.
- For Markov chains, it's more complicated...

Inference by Sampling

- Using samples from discrete Markov chain to compute marginals numerically:
  1. Generate a large number of samples x^i from the model.

$$X = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}.$$

Inference by Sampling

- Using samples from discrete Markov chain to compute marginals numerically:
  1. Generate a large number of samples x^i from the model.

$$X = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}.$$

  2. Compute frequency that variable j was equal to c.

$$p(x_2 = 1) = \frac{2}{4} = 0.5, \quad p(x_3 = 0) = \frac{0}{4} = 0.$$

Inference by Sampling

- Using samples from discrete Markov chain to compute marginals numerically:
  1. Generate a large number of samples x^i from the model.

$$X = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}.$$

  2. Compute frequency that variable j was equal to c.

$$p(x_2 = 1) = \frac{2}{4} = 0.5, \quad p(x_3 = 0) = \frac{0}{4} = 0.$$

- This is a special case of a Monte Carlo method.
  - Second most important class of ML algorthms (after numerical optimization).
  - Originally developed to build better atomic bombs :(

Monte Carlo Methods

- Monte Carlo methods approximate expectations of random functions,

$$\mathbb{E}[g(X)] = \underbrace{\sum_{x \in \mathcal{X}} g(x)p(x)}_{\text{discrete } x} \quad \text{or} \quad \underbrace{\mathbb{E}[g(X)] = \int_{x \in \mathcal{X}} g(x)p(x)dx}_{\text{continuous } x}.$$

Monte Carlo Methods

- Monte Carlo methods approximate expectations of random functions,

$$\mathbb{E}[g(X)] = \underbrace{\sum_{x \in \mathcal{X}} g(x)p(x)}_{\text{discrete } x} \quad \text{or} \quad \underbrace{\mathbb{E}[g(X)] = \int_{x \in \mathcal{X}} g(x)p(x)dx}_{\text{continuous } x}.$$

- Using n samples x^i from $p(x)$ the Monte Carlo estimate is

$$\mathbb{E}[g(X)] \approx \frac{1}{n}\sum_{i=1}^{n} g(x^i).$$

Monte Carlo Methods

- Monte Carlo methods approximate expectations of random functions,

$$\mathbb{E}[g(X)] = \underbrace{\sum_{x \in \mathcal{X}} g(x)p(x)}_{\text{discrete } x} \quad \text{or} \quad \underbrace{\mathbb{E}[g(X)] = \int_{x \in \mathcal{X}} g(x)p(x)dx}_{\text{continuous } x}.$$

- Using n samples x^i from $p(x)$ the Monte Carlo estimate is

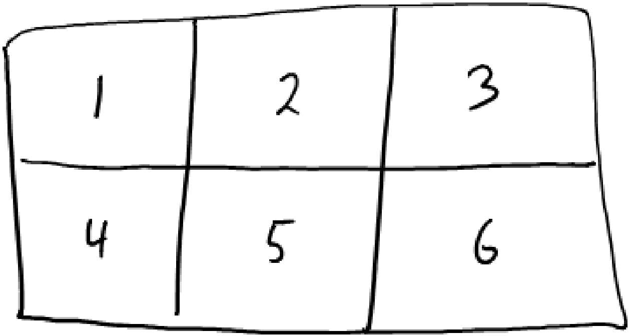
$$\mathbb{E}[g(X)] \approx \frac{1}{n}\sum_{i=1}^{n} g(x^i).$$

- We often take $g(X)$ as indicator function $\mathcal{I}_{\{A\}}$ for some event A so that

$$\mathbb{E}[g(X)] = \mathbb{E}[\mathcal{I}_{\{A\}}] = p(A), \quad \text{and} \quad p(A) \approx \frac{1}{n}\sum_{i=1}^{n} \mathcal{I}_{\{A_i\}},$$

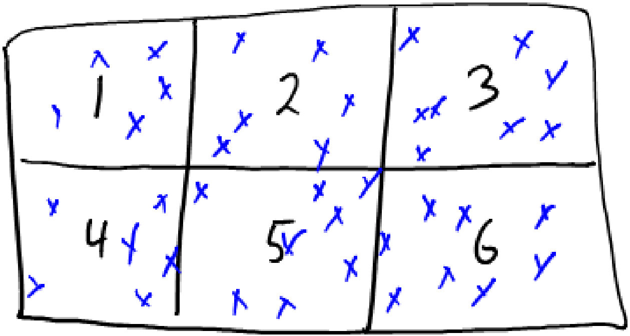
which is a very simple "mixture of indicators" or kernel density estimator model.

Monte Carlo Method for Rolling Di



Probability of event: (number of samples consistent with event)/(number of samples)

Monte Carlo Method for Rolling Di

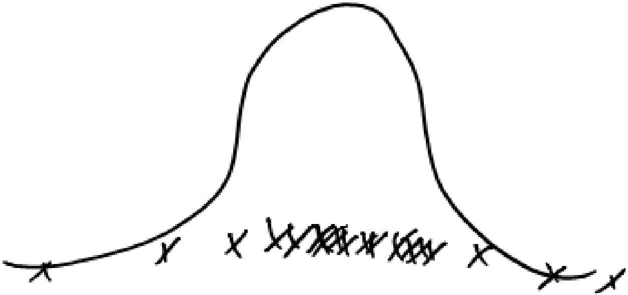


Probability of event: (number of samples consistent with event)/(number of samples)

Monte Carlo Method for Inequalities

Monte Carlo estimate of probability that variable is above threshold,

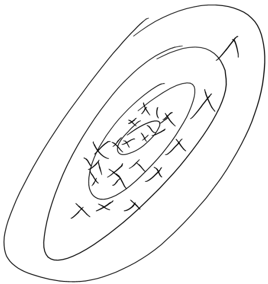
$$g(x) = \mathcal{I}_{x \geq \tau}.$$



Monte Carlo Method for Mean

We could compute mean using $g(x) = x$.

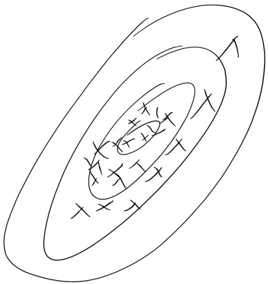
$$E[x] \approx \frac{1}{n}\sum_{i=1}^{n} x^i.$$



Monte Carlo Method for Mean

We could compute mean using $g(x) = x$.

$$E[x] \approx \frac{1}{n}\sum_{i=1}^{n} x^i.$$



How could we sample from a 2D Gaussian?

- Use product rule $p(x, z) = p(z|x)p(x)$ and ancestral sampling:
    - Sample x from marginal $p(x)$, sample z from conditional $p(z|x)$ (both Gaussian).

Monte Carlo Methods

- Monte Carlo estimate is unbiased approximation of expectation,

$$\mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n} g(x^i)\right] = \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}[g(x^i)]$$

Monte Carlo Methods

- Monte Carlo estimate is unbiased approximation of expectation,

$$\mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n} g(x^i)\right] = \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}[g(x^i)]$$
$$= \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}[g(X)] = \mathbb{E}[g(X)],$$

so by law of large numbers it converges (almost surely) to $\mathbb{E}[g(X)]$ as $n \to \infty$.

Monte Carlo Methods

- Monte Carlo estimate is unbiased approximation of expectation,

$$\mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n} g(x^i)\right] = \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}[g(x^i)]$$
$$= \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}[g(X)] = \mathbb{E}[g(X)],$$

  so by law of large numbers it converges (almost surely) to $\mathbb{E}[g(X)]$ as $n \to \infty$.

- Allows computing expectations in Markov chains even if x_j is continuous:
  - $E[x_j]$ is approximated by average of x_j in the samples.

Monte Carlo Methods

- Monte Carlo estimate is unbiased approximation of expectation,

$$\mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n} g(x^i)\right] = \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}[g(x^i)]$$
$$= \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}[g(X)] = \mathbb{E}[g(X)],$$

so by law of large numbers it converges (almost surely) to $\mathbb{E}[g(X)]$ as $n \to \infty$.

- Allows computing expectations in Markov chains even if x_j is continuous:
  - $E[x_j]$ is approximated by average of x_j in the samples.
  - $p(x_j \leq 10)$ is approximate by frequency of x_j being less than 10.
  - $p(x_j \leq 10, x_{j+1} \geq 10)$ is approximated by frequency of joint event.

Exact Marginal Calculation

- Rate of convergence of Monte Carlo measured by variance of estimator.
  - If all samples look the same, it converges quickly.
  - If samples look very different, it can be painfully slow.

Exact Marginal Calculation

- Rate of convergence of Monte Carlo measured by variance of estimator.
  - If all samples look the same, it converges quickly.
  - If samples look very different, it can be painfully slow.

- We can sometimes avoid Monte Carlo and compute univariate marginals exactly:
  - Markov chains with discrete or Gaussian probabilities.

Exact Marginal Calculation

- Rate of convergence of Monte Carlo measured by variance of estimator.
  - If all samples look the same, it converges quickly.
  - If samples look very different, it can be painfully slow.

- We can sometimes avoid Monte Carlo and compute univariate marginals exactly:
  - Markov chains with discrete or Gaussian probabilities.

- In the discrete case, this is given by the recursive Chapman-Kolmogorov equations,

$$p(x_j) = \underbrace{\sum_{x_{j-1}} p(x_j, x_{j-1})}_{\text{marginalization rule}}$$

Exact Marginal Calculation

- Rate of convergence of Monte Carlo measured by variance of estimator.
  - If all samples look the same, it converges quickly.
  - If samples look very different, it can be painfully slow.

- We can sometimes avoid Monte Carlo and compute univariate marginals exactly:
  - Markov chains with discrete or Gaussian probabilities.

- In the discrete case, this is given by the recursive Chapman-Kolmogorov equations,

$$p(x_j) = \underbrace{\sum_{x_{j-1}} p(x_j, x_{j-1})}_{\text{marginalization rule}} = \sum_{x_{j-1}} \underbrace{p(x_j | x_{j-1})p(x_{j-1})}_{\text{product rule}}.$$

- Simple equation that gives probability of all paths leading to $x_j = c$ for all c.

Exact Marginal Calculation

- Recursive Chapman-Kolmogorov (CK) equations:

$$p(x_j) = \sum_{x_{j-1}} p(x_j | x_{j-1}) p(x_{j-1}).$$

- In Markov chains we're given $p(x_1 = c)$ for all c.
  - CK equations give us $p(x_2 = c)$ for all c if we know $p(x_1 = c)$ for all c.

Exact Marginal Calculation

- Recursive Chapman-Kolmogorov (CK) equations:

$$p(x_j) = \sum_{x_{j-1}} p(x_j|x_{j-1})p(x_{j-1}).$$

- In Markov chains we're given $p(x_1 = c)$ for all c.
  - CK equations give us $p(x_2 = c)$ for all c if we know $p(x_1 = c)$ for all c.
  - CK equations give us $p(x_3 = c)$ for all c if we know $p(x_2 = c)$ for all c.
  - ...

Exact Marginal Calculation

- Recursive Chapman-Kolmogorov (CK) equations:

$$p(x_j) = \sum_{x_{j-1}} p(x_j | x_{j-1}) p(x_{j-1}).$$

- In Markov chains we're given $p(x_1 = c)$ for all c.
  - CK equations give us $p(x_2 = c)$ for all c if we know $p(x_1 = c)$ for all c.
  - CK equations give us $p(x_3 = c)$ for all c if we know $p(x_2 = c)$ for all c.
  - ...
- Cost of computing all univarite marginals is $O(dk^2)$ if variable has k states.
  - We repeatedly multiply vector containing marginals by k by k transition matrix.

Exact Marginal Calculation

- Recursive Chapman-Kolmogorov (CK) equations:

$$p(x_j) = \sum_{x_{j-1}} p(x_j|x_{j-1})p(x_{j-1}).$$

- In Markov chains we're given $p(x_1 = c)$ for all c.
    - CK equations give us $p(x_2 = c)$ for all c if we know $p(x_1 = c)$ for all c.
    - CK equations give us $p(x_3 = c)$ for all c if we know $p(x_2 = c)$ for all c.
    - ...
- Cost of computing all univarite marginals is $O(dk^2)$ if variable has k states.
    - We repeatedly multiply vector containing marginals by k by k transition matrix.

- We can also define a continuous version:

$$p(x_j) = \int_{x_{j-1}} p(x_j|x_{j-1})p(x_{j-1}) = \int_{x_{j-1}} p(x_j, x_{j-1})$$

- If $p(x_{j-1})$ and $p(x_j|x_{j-1})$ are Gaussian, then $p(x_j, x_{j-1})$ is Gaussian.
    - Implies $p(x_j)$ is a Gaussian marginal.

Summary

- Independent componenet analysis: allows identifying non-Gaussian latent factors.

Summary

- Independent componenet analysis: allows identifying non-Gaussian latent factors.
- Markov chains model dependencies betwee adjacent features.

Summary

- Independent componenet analysis: allows identifying non-Gaussian latent factors.
- Markov chains model dependencies betwee adjacent features.
- Parameter tieing uses same parameters in different parts of a model.
  - Allows models of different sizes and more data for parameter estimation.

Summary

- Independent componenet analysis: allows identifying non-Gaussian latent factors.
- Markov chains model dependencies betwee adjacent features.
- Parameter tieing uses same parameters in different parts of a model.
  - Allows models of different sizes and more data for parameter estimation.
- Inverse Transform generates samples from simple 1D distributions.

Summary

- Independent componenet analysis: allows identifying non-Gaussian latent factors.
- Markov chains model dependencies betwee adjacent features.
- Parameter tieing uses same parameters in different parts of a model.
  - Allows models of different sizes and more data for parameter estimation.
- Inverse Transform generates samples from simple 1D distributions.
- Ancestral sampling generates samples from a Markov chain.

Summary

- Independent componenet analysis: allows identifying non-Gaussian latent factors.
- Markov chains model dependencies betwee adjacent features.
- Parameter tieing uses same parameters in different parts of a model.
    - Allows models of different sizes and more data for parameter estimation.
- Inverse Transform generates samples from simple 1D distributions.
- Ancestral sampling generates samples from a Markov chain.
- Monte Carlo methods approximate expectations using samples.

Summary

- Independent componenet analysis: allows identifying non-Gaussian latent factors.
- Markov chains model dependencies betwee adjacent features.
- Parameter tieing uses same parameters in different parts of a model.
  - Allows models of different sizes and more data for parameter estimation.
- Inverse Transform generates samples from simple 1D distributions.
- Ancestral sampling generates samples from a Markov chain.
- Monte Carlo methods approximate expectations using samples.
- Chapman-Kolmogorov equations compute exact univariate marginals.
  - For discrete or Gaussian Markov chains.

- Next time: weakening the Markov assumption.