ENGR-E 511; ENGR-E 399

# Machine Learning for Signal Processing

Module 09:

# Undirected Graphical Models

**Minje Kim**

Department of Intelligent Systems Engineering

Email: minje@indiana.edu

Website: http://minjekim.com

Research Group: http://saige.sice.indiana.edu

Meeting Request: http://doodle.com/minje

INDIANA UNIVERSITY

SCHOOL OF INFORMATICS, COMPUTING, AND ENGINEERING

# A Labeling Problem

## - Image segmentation

- You've got an image of 401 X 600 = 240,600 pixels

- Let's assume that there are four segments
- How many candidate solutions to this problem?
  - $4^{240600}$
- We need to do something more than a random guess

# A Labeling Problem
## - Image segmentation

- We've seen a way to give a structure to the problem
  - Clustering: GMM, kMeans, etc
  - Classification: naïve Bayes, neural networks, etc
- The labeling problem:
  - Given a data sample
  - Estimate the posterior prob of the latent variable (class label)

$$\underbrace{P(\boldsymbol{C}_{ij}|\boldsymbol{X}_{ij})}_{\text{Posterior}} \propto \underbrace{P(\boldsymbol{X}_{ij}|\boldsymbol{C}_{ij})}_{\text{Likelihood}}\underbrace{P(\boldsymbol{C}_{ij})}_{\text{Prior}}$$

  - Prior: your prior knowledge about the classes themselves
    - e.g. What's the probability of seeing a "red leaves" class in general?
    - But there could be some other kinds as well (e.g. transition probability)
  - Likelihood: probability of seeing a particular observation given the class
    - e.g. What's the probability of seeing this particular red-ish pixel given the "red leaves" class?
  - Posterior: For your data sample, you calculate the posterior probability for all (four) different classes

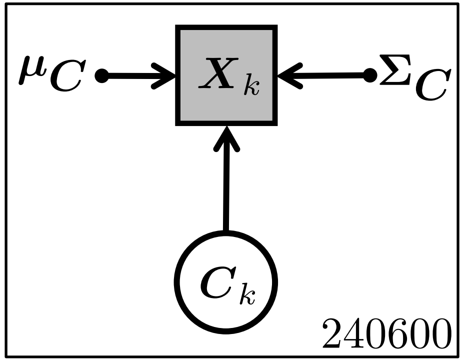
# A Labeling Problem

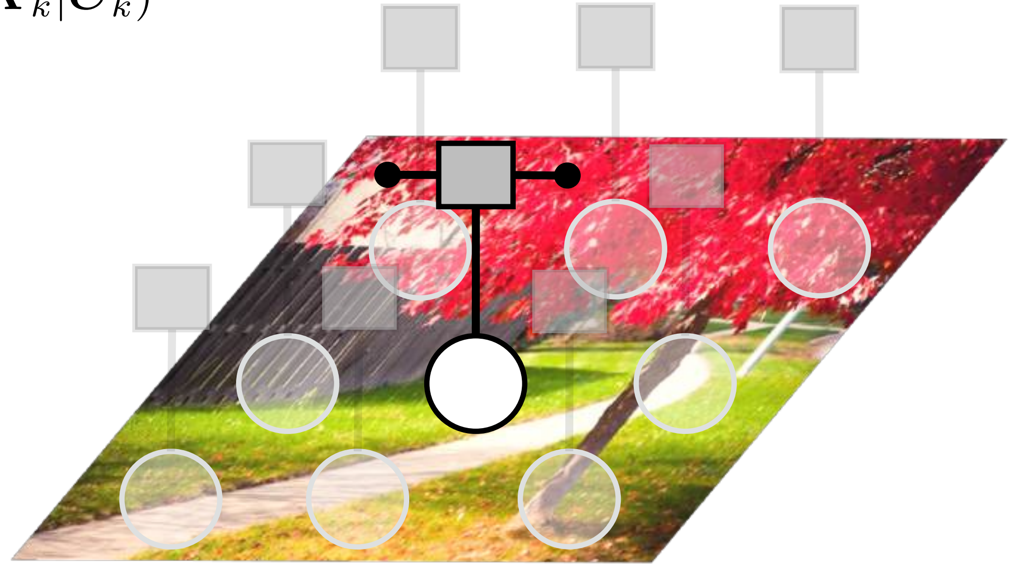
## - Image segmentation

- If we have no prior information about these labels
  - In other words, your prior distribution over the classes is uniform

$$P(\boldsymbol{C}_{ij}|\boldsymbol{X}_{ij}) \propto P(\boldsymbol{X}_{ij}|\boldsymbol{C}_{ij})$$

- Let me simplify a bit more
  - No need to keep two indices if samples are independent from each other
    - i.e. vectorization $P(\boldsymbol{C}_k|\boldsymbol{X}_k) \propto P(\boldsymbol{X}_k|\boldsymbol{C}_k)$



# A Labeling Problem

## - Image segmentation

- I prepared a set of good model parameters for your naïve Bayes classification

$$\boldsymbol{\mu}_1 = [0.55, 0.63, 0.10]^\top \qquad \boldsymbol{\Sigma}_1 = \begin{bmatrix} 0.05 & 0.04 & 0.02 \\ 0.04 & 0.03 & 0.01 \\ 0.02 & 0.01 & 0.01 \end{bmatrix}$$

$$\boldsymbol{\mu}_2 = [0.90, 0.15, 0.24]^\top \qquad \boldsymbol{\Sigma}_2 = \begin{bmatrix} 0.01 & 0.01 & 0.01 \\ 0.01 & 0.04 & 0.03 \\ 0.01 & 0.03 & 0.02 \end{bmatrix}$$

$$\boldsymbol{\mu}_3 = [0.27, 0.23, 0.23]^\top \qquad \boldsymbol{\Sigma}_3 = \begin{bmatrix} 0.01 & 0.01 & 0.01 \\ 0.01 & 0.01 & 0.01 \\ 0.01 & 0.01 & 0.01 \end{bmatrix}$$

$$\boldsymbol{\mu}_4 = [0.92, 0.84, 0.77]^\top \qquad \boldsymbol{\Sigma}_4 = \begin{bmatrix} 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 \end{bmatrix}$$

- Based on this, I can form a classifier

$$\boldsymbol{C}_k = \underset{\boldsymbol{C}_k}{\arg\max}\, P(\boldsymbol{C}_k | \boldsymbol{X}_k) = \underset{\boldsymbol{C}_k}{\arg\max}\, \mathcal{N}(\boldsymbol{X}_k; \mu_{\boldsymbol{C}_k}, \Sigma_{\boldsymbol{C}_k})$$

  - Note that I ignore the prior ('cause it's uniform)

# A Labeling Problem

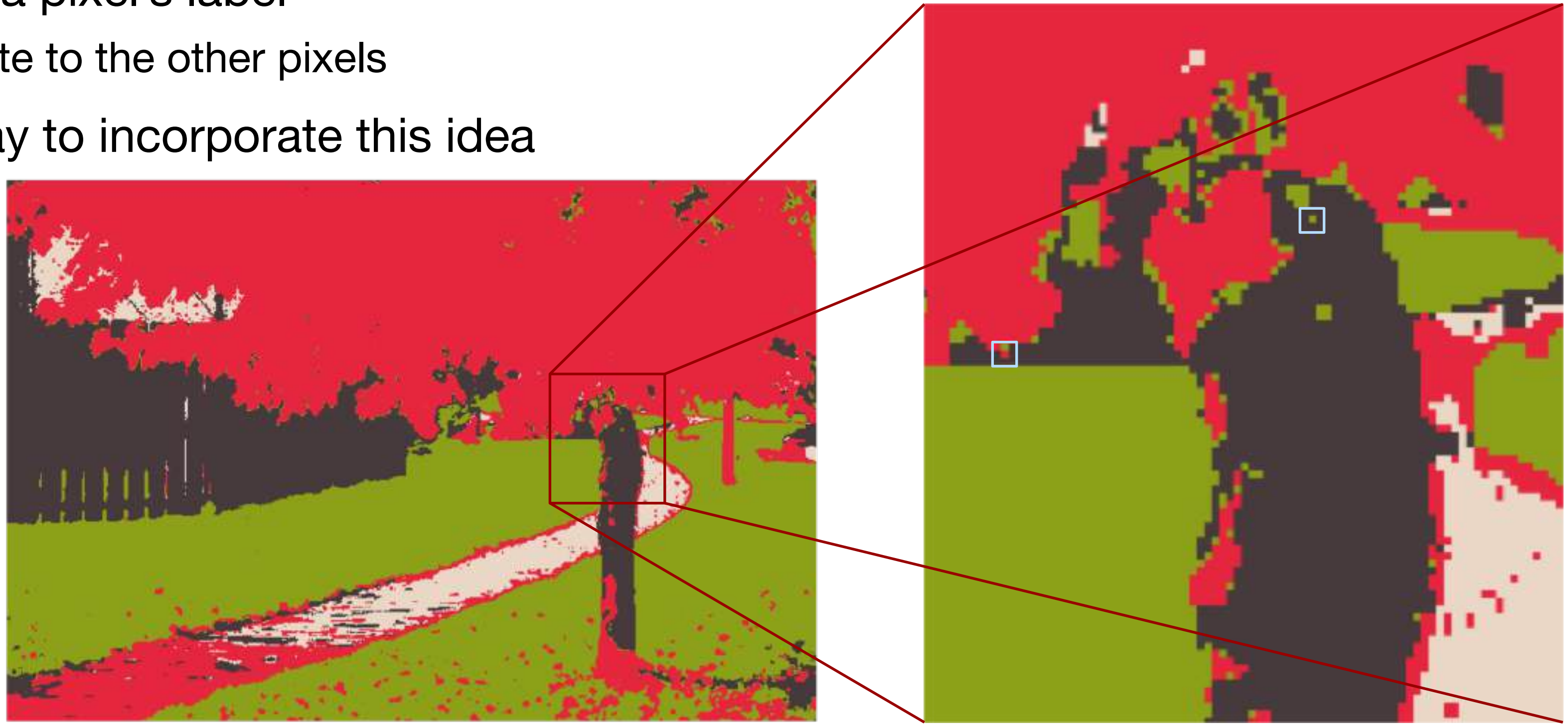
## - Image segmentation

- So far everything should be straightforward
- But I don't quite like the result (as always), because..
  - There are some pixels that seem to be classified into a wrong class
- We'll address this problem today

# A Labeling Problem

## - Image segmentation

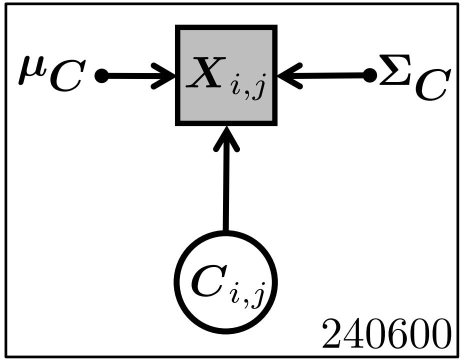
- If all its surrounding pixels are grey
  - It's more probable that the pixel should belong to the grey class, too
- If you make a change on a pixel's label
  - The decision can propagate to the other pixels
- We need a systematic way to incorporate this idea into an optimization problem

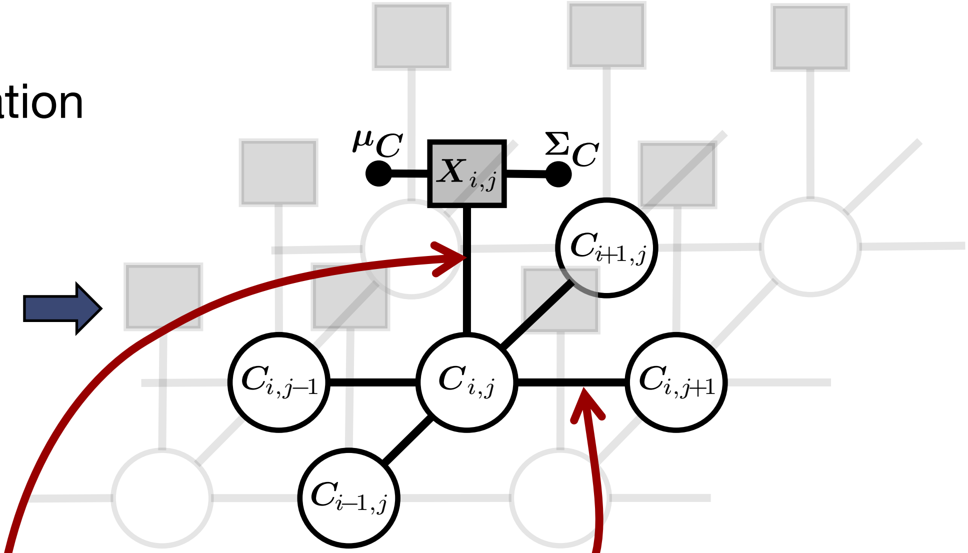


# Markov Random Fields

## - A smoothing mechanism

- Now we incorporate the prior information




$$P(\boldsymbol{C}|\boldsymbol{X}) \propto \prod_{i,j} P(\boldsymbol{X}_{i,j}|\boldsymbol{C}_{i,j}) \prod_{k,l \in \mathcal{N}_{i,j}} P(\boldsymbol{C}_{i,j}|\boldsymbol{C}_{k,l})$$

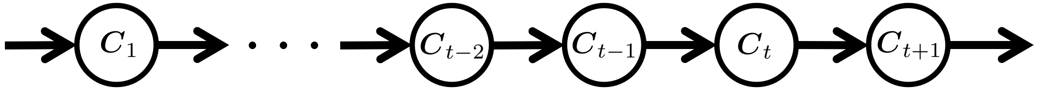
$$P(\boldsymbol{C}_{i,j}|\boldsymbol{C}_{k,l}) \propto \exp^{-\left\{f\left(\boldsymbol{C}_{i,j},\boldsymbol{C}_{k,l}\right)\right\}^2/\sigma^2_{\mathcal{N}}} \quad f(\boldsymbol{C}_{i,j},\boldsymbol{C}_{k,l}) = \begin{cases} 0 & \text{if } \boldsymbol{C}_{i,j} = \boldsymbol{C}_{k,l} \\ a & \text{otherwise} \end{cases}$$

  - If the labels of a neighboring pair agree, the prior probability becomes high
  - Disagreement demotes the posterior probability

# Markov Random Fields

## - A smoothing mechanism

- Why the name "Markov"?
- The Markov chain we've seen so far

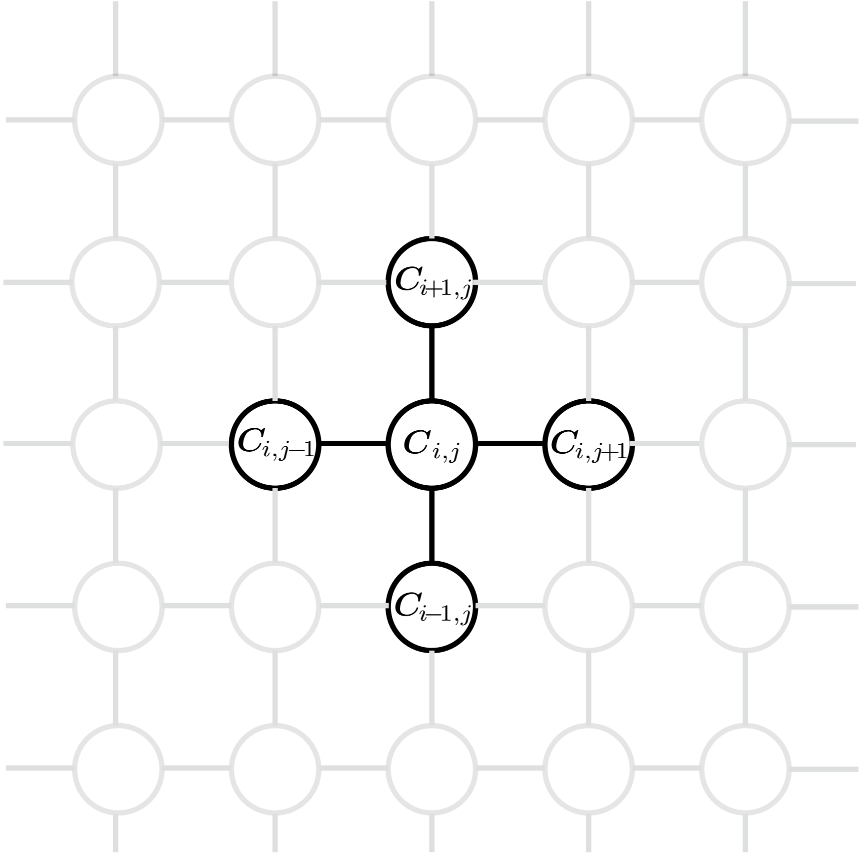



$$P(\boldsymbol{C}_t|\boldsymbol{C}_1, \boldsymbol{C}_2, \cdots, \boldsymbol{C}_{t-1}) = P(\boldsymbol{C}_t|\boldsymbol{C}_{t-1})$$

  - If you know the adjacent label, all the previous labels don't matter

- Markov blanket
  - An undirected graph version
  - In 2D labeling problem..
    - If you know the labels of all your neighbors that block the path from all the other nodes

$$P(\boldsymbol{C}_{i,j}|\boldsymbol{C}_{\backslash i,j}) = P(\boldsymbol{C}_{i,j}|\boldsymbol{C}_{\mathcal{N}_{i,j}})$$



# Markov Random Fields

## - Iterated Conditional Mode (ICM)

- ICM can be seen as a coordinate-wise gradient descent method

- Initialize the labels
  - Using naïve Bayes in our case

- (Randomly) choose a node (repeat)
  - Check out the latent value that maximize the post prob
    - By using the likelihood and prior prob
    - While all the other labels are fixed
  - Replace the label with the best one

# Markov Random Fields

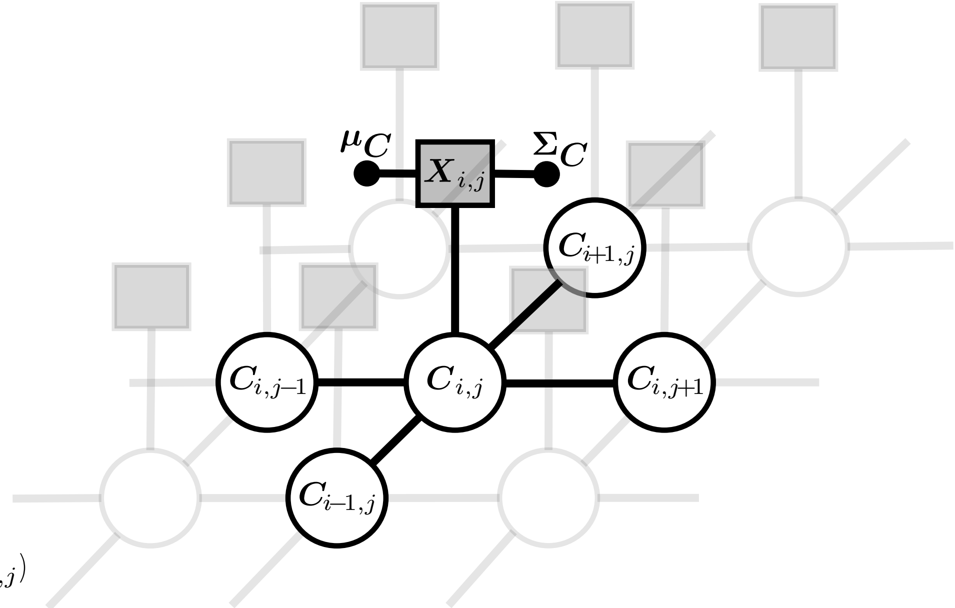
## - Iterated Conditional Mode (ICM)

- Update the label $\boldsymbol{C}_{i,j}$ based on..

$$\underset{\boldsymbol{C}_{i,j}}{\arg\max}\, P(\boldsymbol{X}_{i,j}|\boldsymbol{C}_{i,j}) \prod_{k,l \in \mathcal{N}_{i,j}} P(\boldsymbol{C}_{i,j}|\boldsymbol{C}_{k,l})$$

- $P(\boldsymbol{X}_{i-1,j}|\boldsymbol{C}_{i-1,j})P(\boldsymbol{C}_{i-1,j}|\boldsymbol{C}_{i,j})\xi_{i-1,j}$
- $P(\boldsymbol{X}_{i+1,j}|\boldsymbol{C}_{i+1,j})P(\boldsymbol{C}_{i+1,j}|\boldsymbol{C}_{i,j})\xi_{i+1,j}$
- $P(\boldsymbol{X}_{i,j-1}|\boldsymbol{C}_{i,j-1})P(\boldsymbol{C}_{i,j-1}|\boldsymbol{C}_{i,j})\xi_{i,j-1}$
- $P(\boldsymbol{X}_{i,j+1}|\boldsymbol{C}_{i,j+1})P(\boldsymbol{C}_{i,j+1}|\boldsymbol{C}_{i,j})\xi_{i,j+1}$

The other priors
(nothing to do with $\boldsymbol{C}_{i,j}$)




- Repeat this for all latent nodes
- Once you visit all nodes, that's an epoch
- Do as many epoch as you want (until convergence)

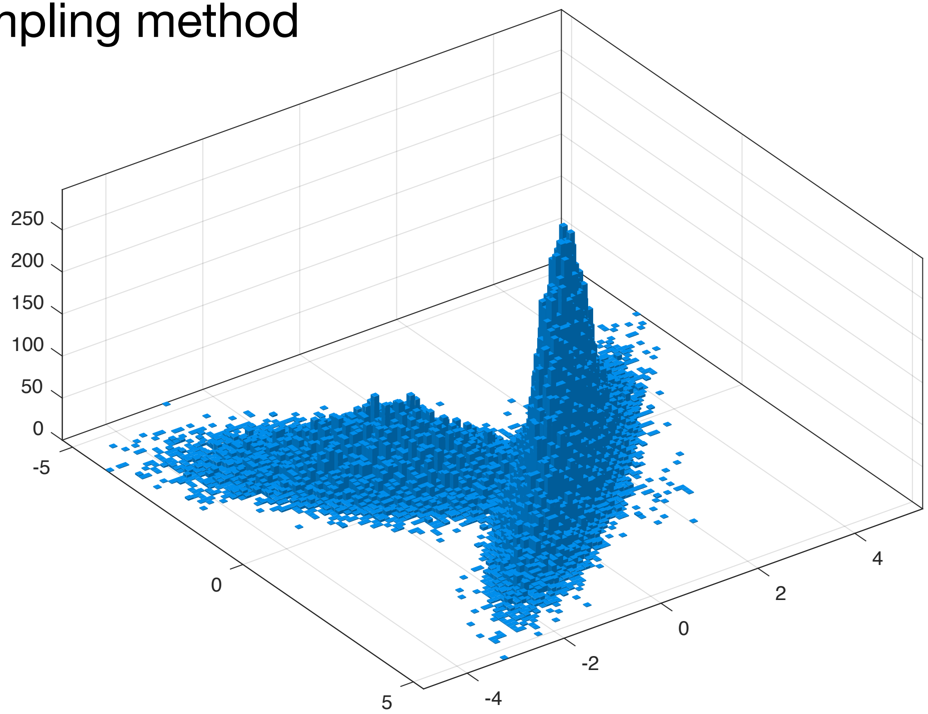
# Markov Random Fields

- Iterated Conditional Mode (ICM)

# Sampling Methods

## - For estimating distributions

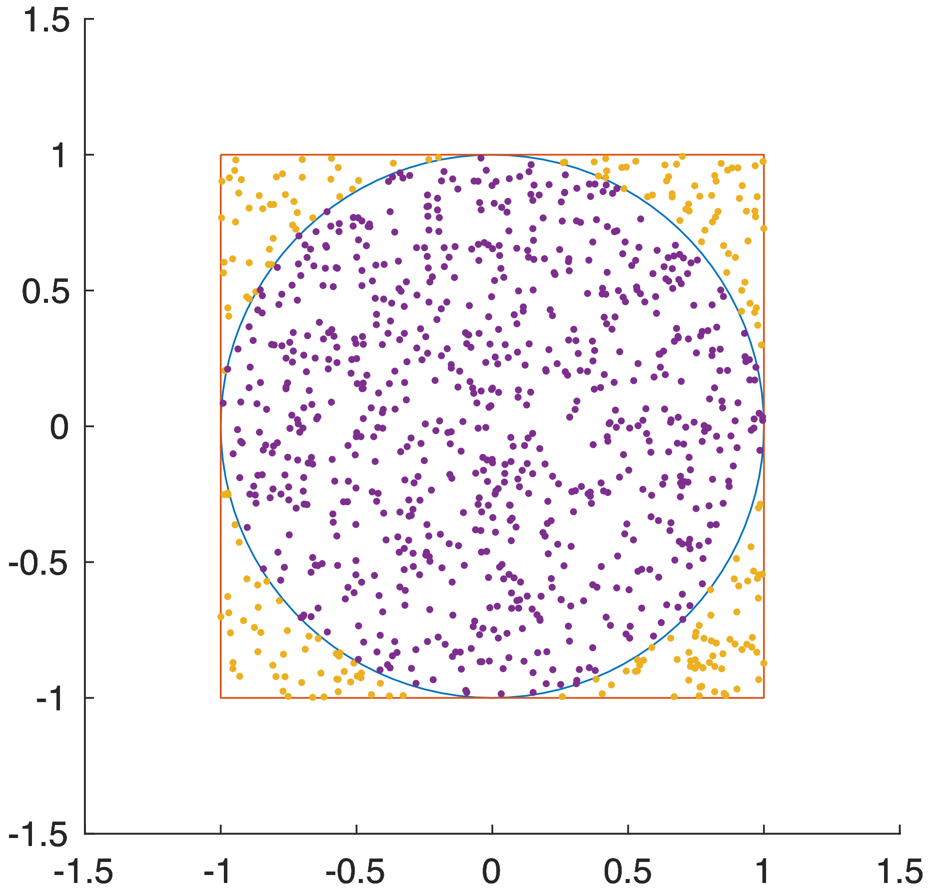
- So far so good, but I feel that ICM is a bit too naïve
- What we want to do is to estimate the posterior distribution
- There are a lot of ways, but today we'll see a sampling method
- Basic idea:
  If you have a good sample distribution,
  then you can estimate the original distribution
  - How?
  - Histogram
    (for continuous variables)
  - Counting and normalizing
    (for discrete variables)



# Sampling Methods

## - Rejection sampling

- You're a mathematician in a secluded tribe
- You don't know what $\pi$ is
- But you're asked to calculate the area of your client's land
  - Which looks like a circle
- How would you calculate the size of the area?
- You draw a square that's larger than the land
  - 'Cause you do know
    how to calculate the size of a square
- You throw 1,000 painted stones
  in the square
  - As randomly as possible
- Count the number of stones inside the circle
  - It turned out that it's 795
- Since the size of the square land is 4 sq. miles,
  - The estimated circular area is 4 X 0.795 = 3.18
  - $\pi r^2 = 3.14 \times 1 = 3.14$



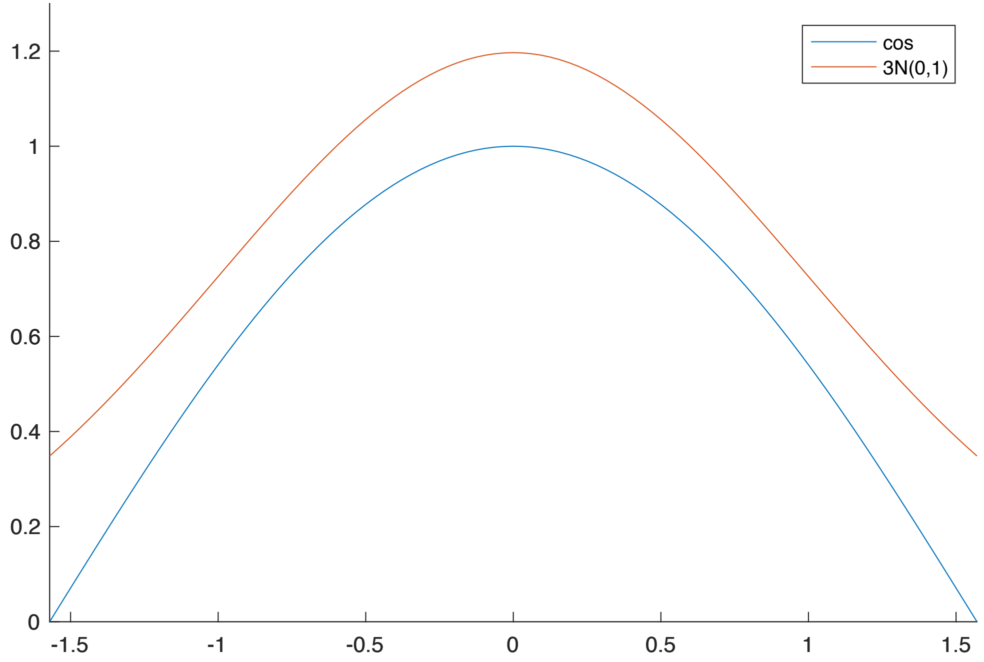
# Sampling Methods

## - Rejection sampling

- There are distributions easier to sample from
  - And, those that are not
  - e.g. Uniform and Gaussian are easier
- For example..

$$P(x) = \begin{cases} \frac{1}{Z}\cos(x) & \text{if } -\frac{\pi}{2} \le x \le \frac{\pi}{2} \\ 0 & \text{otherwise} \end{cases}$$

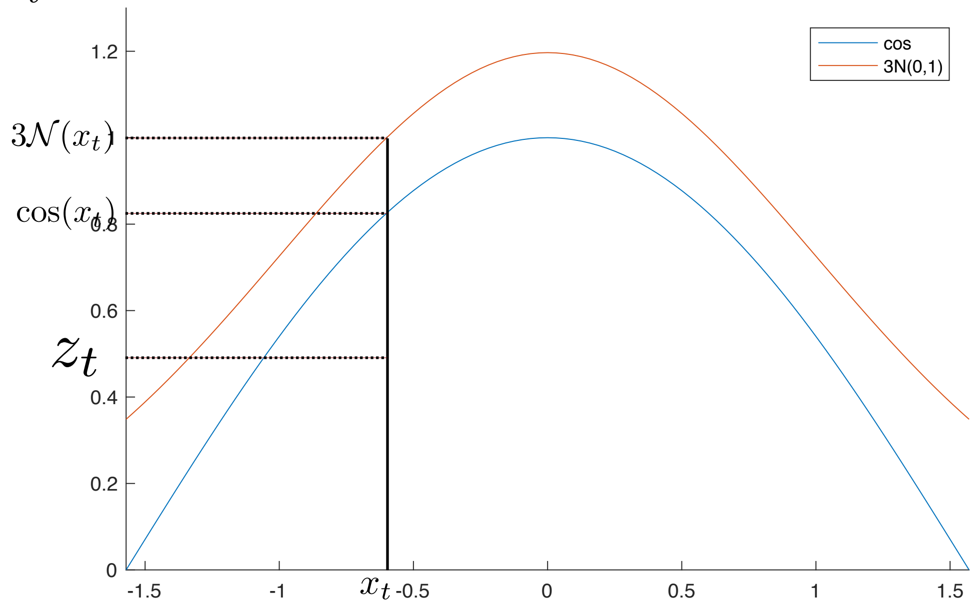
  - What's the mean and the standard deviation?
- We're going to use a **proposal distribution** for the sampling job
  - Hoping that it's easier to sample from there
  - It should be larger than the original distribution at every point



# Sampling Methods

## - Rejection sampling

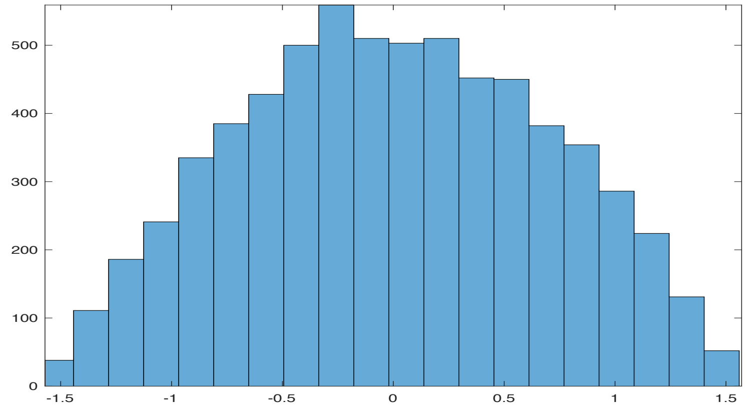
- We assume that it’s easier to sample from a Gaussian
- Rejection sampling
  - Sample from the proposal distribution $x_t \sim kq(x) = 3\mathcal{N}(x; 0, 1)$
  - Sample another value from a uniform distribution $z_t \sim [0, 3\mathcal{N}(x_t; 0, 1)]$
  - If $z_t > \cos(x_t)$, reject $x_t$
  - (Repeat)

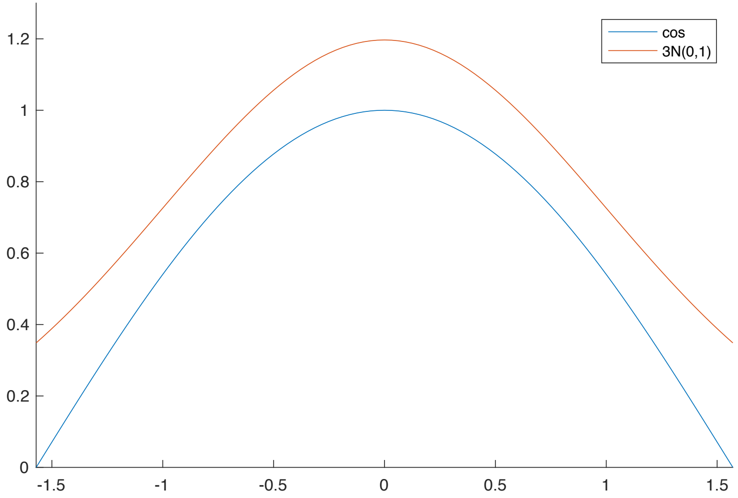


# Sampling Methods

## - Rejection sampling

- Histogram of the samples




- Mean $\frac{1}{N}\sum_{t=1}^{N} x_t = 0.000473$

- Does it always work like this?
  - What if we can't come up with a proposal distribution?

# Gibbs Sampling

## - A Markov Chain Monte Carlo (MCMC) method

- The posterior distribution is difficult to sample from $P(\boldsymbol{C}|\boldsymbol{X})$
  - What's the dimensionality? $401 \times 600 \times 4$
  - It can be very easily high dimensional and difficult to come up with a proposal distribution
  - We don't even know how to correctly calculate this
- With the Markov blanket assumption we can simplify this calculation
  - You've already seen it

$$
\begin{aligned}
P(\boldsymbol{C}|\boldsymbol{X}) &\propto \prod_{i,j} P(\boldsymbol{X}_{i,j}|\boldsymbol{C}_{i,j}) \prod_{k,l \in \mathcal{N}_{i,j}} P(\boldsymbol{C}_{i,j}|\boldsymbol{C}_{k,l}) \\
&= \prod_{i,j} P(\boldsymbol{X}_{i,j}|\boldsymbol{C}_{i,j}) \prod_{k,l \in \setminus i,j} P(\boldsymbol{C}_{i,j}|\boldsymbol{C}_{k,l})
\end{aligned}
$$

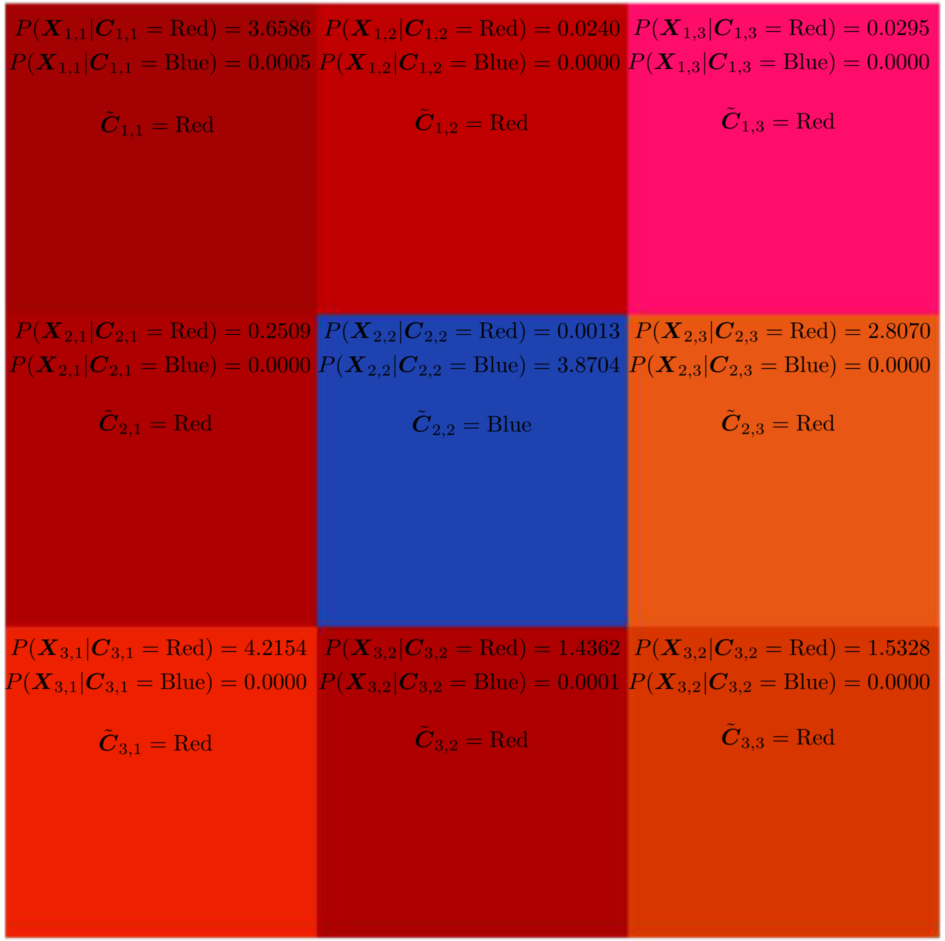
$$
P(\boldsymbol{C}_{i,j}|\boldsymbol{C}_{k,l}) \propto \exp^{-\left\{f\left(\boldsymbol{C}_{i,j},\boldsymbol{C}_{k,l}\right)\right\}^2/\sigma_{\mathcal{N}}^2}
$$

$$
f(\boldsymbol{C}_{i,j}, \boldsymbol{C}_{k,l}) = \begin{cases} 0 & \text{if } \boldsymbol{C}_{i,j} = \boldsymbol{C}_{k,l} \\ a & \text{otherwise} \end{cases}
$$

  - If we observe the labels of the neighboring nodes, we don't have to know the other labels of non-neighbors
  - But then, how do we know the labels of the neighboring nodes?

# Gibbs Sampling

## - A Markov Chain Monte Carlo (MCMC) method




$$
\begin{aligned}
&P(\boldsymbol{C}_{2,2} = \text{Red}|\boldsymbol{X}_{2,2}) = \\
&P(\boldsymbol{X}_{2,2}|\boldsymbol{C}_{2,2} = \text{Red})\cdot \\
&P(\boldsymbol{C}_{2,2} = \text{Red}|\tilde{\boldsymbol{C}}_{1,2} = \text{Red})\cdot \\
&P(\boldsymbol{C}_{2,2} = \text{Red}|\tilde{\boldsymbol{C}}_{3,2} = \text{Red})\cdot \\
&P(\boldsymbol{C}_{2,2} = \text{Red}|\tilde{\boldsymbol{C}}_{2,1} = \text{Red})\cdot \\
&P(\boldsymbol{C}_{2,2} = \text{Red}|\tilde{\boldsymbol{C}}_{2,3} = \text{Red})\cdot \\
&= 0.0013 \times 1 \times 1 \times 1 \times 1 = 0.0013
\end{aligned}
$$

$$
\begin{aligned}
&P(\boldsymbol{C}_{2,2} = \text{Blue}|\boldsymbol{X}_{2,2}) = \\
&P(\boldsymbol{X}_{2,2}|\boldsymbol{C}_{2,2} = \text{Blue})\cdot \\
&P(\boldsymbol{C}_{2,2} = \text{Blue}|\tilde{\boldsymbol{C}}_{1,2} = \text{Red})\cdot \\
&P(\boldsymbol{C}_{2,2} = \text{Blue}|\tilde{\boldsymbol{C}}_{3,2} = \text{Red})\cdot \\
&P(\boldsymbol{C}_{2,2} = \text{Blue}|\tilde{\boldsymbol{C}}_{2,1} = \text{Red})\cdot \\
&P(\boldsymbol{C}_{2,2} = \text{Blue}|\tilde{\boldsymbol{C}}_{2,3} = \text{Red})\cdot \\
&= 3.8704 \times 0.0498^4 = 0.00002378
\end{aligned}
$$

$$P(\boldsymbol{C}_{2,2} = \text{Blue}|\boldsymbol{X}_{2,2}) = \frac{0.00002378}{0.0013 + 0.00002378} = 0.018$$

$$P(\boldsymbol{C}_{2,2} = \text{Red}|\boldsymbol{X}_{2,2}) = \frac{0.0013}{0.0013 + 0.00002378} = 0.982$$

# Gibbs Sampling

## - For image segmentation

- For $t$-th iteration, scan all pixels
  - At $(i, j)$-th pixel, sample a new label, based on its posterior probability
    - Which can be calculated based on the samples of the neighboring labels

$$\tilde{\boldsymbol{C}}_{i,j}^{(t)} \sim P(\boldsymbol{C}_{i,j}|\boldsymbol{X}_{i,j}, \boldsymbol{C}_{\setminus i,j}) = P(\boldsymbol{X}_{i,j}|\boldsymbol{C}_{i,j}) \prod_{k,l \in \mathcal{N}_{i,j}} P(\boldsymbol{C}_{i,j}|\tilde{\boldsymbol{C}}_{k,l}^{(t-1)})$$

    - This sample for $(i, j)$-th pixel will be used to calculate the prior for it's neighboring nodes
  - Move on to the other pixel
- In practice, you may want to sample from the entire matrix using the matrix of labels sampled in the previous round (for a speed-up)
- Burn-in
  - Once you sample a lot in this way, you can accumulate many sampled labels
  - You want to consolidate the samples near the end
  - e.g. Your C matrix in the end is with 401 X 600 X N samples
    - You keep 0.1N final samples and do a majority vote among them

# Gibbs Sampling

- For image segmentation

# MRF on Unsupervised Labeling
## - EM

- So far we saw that the likelihood part is fixed
  - It's also called node potential or data cost
    - FYI, the prior probabilities are called edge potential or smoothness cost
  - Because we were doing naïve Bayes, which is a supervised task

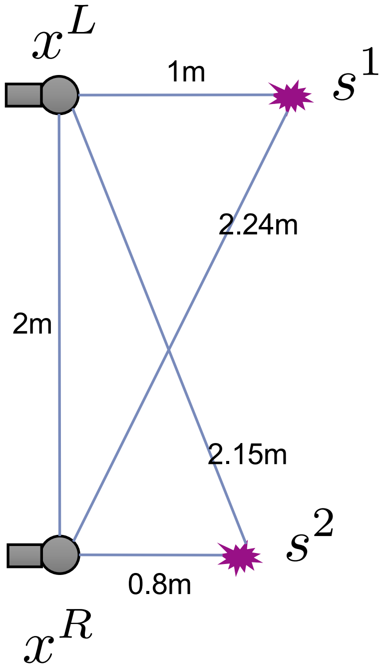
- What if we don't know the model parameters?
  - We have to estimate them using the EM algorithm
    - kMeans or GMM
  - Can we still do MRF smoothing?

- To discuss this problem
  - I'll introduce another labeling problem

# Multichannel Source Separation with MRF

## - SPL and the geometry of the sources and sensors

- Sound Pressure Level (SPL) is inverse-proportional to the distance from the source



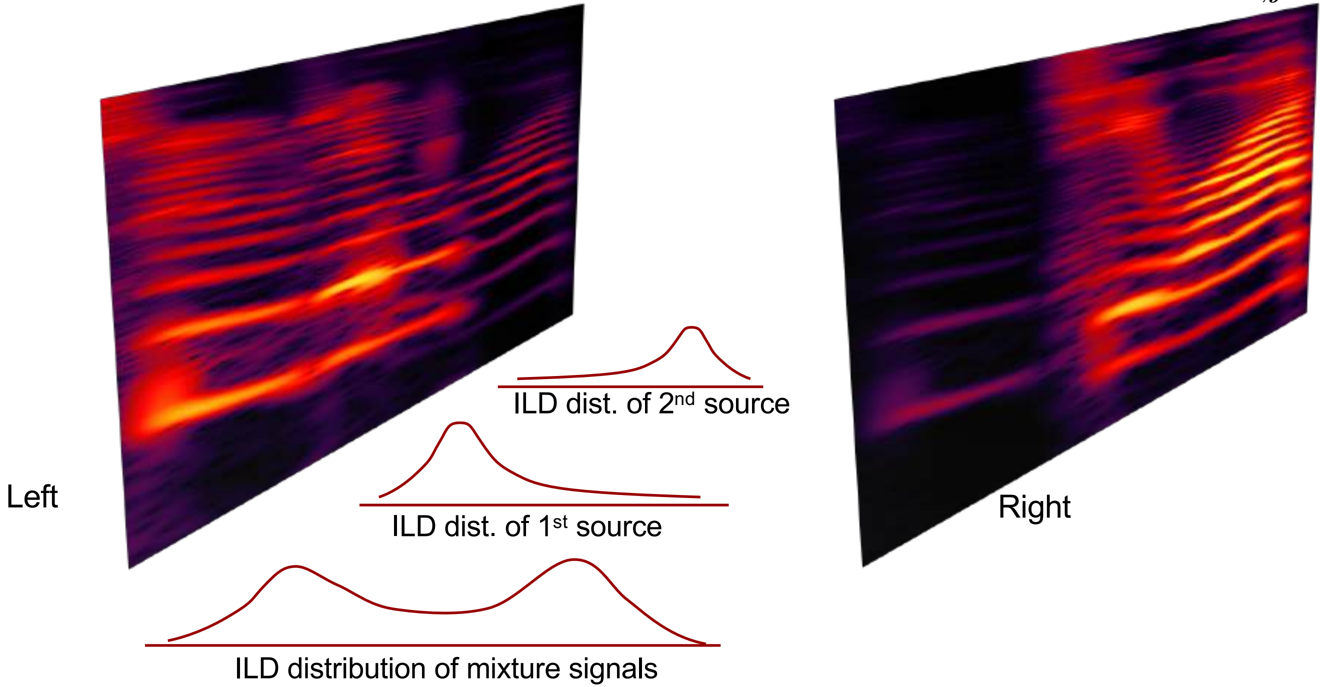

$$a_1 = \frac{\mathrm{SPL}(\hat{\boldsymbol{S}}^{1,L})}{\mathrm{SPL}(\hat{\boldsymbol{S}}^{1,R})} = \frac{\mathrm{dist}(\boldsymbol{X}^R, \boldsymbol{S}^1)}{\mathrm{dist}(\boldsymbol{X}^L, \boldsymbol{S}^1)} = \frac{2.24}{1} = 2.24$$

$$a_2 = \frac{\mathrm{SPL}(\hat{\boldsymbol{S}}^{2,L})}{\mathrm{SPL}(\hat{\boldsymbol{S}}^{2,R})} = \frac{\mathrm{dist}(\boldsymbol{X}^R, \boldsymbol{S}^2)}{\mathrm{dist}(\boldsymbol{X}^L, \boldsymbol{S}^2)} = \frac{0.8}{2.15} = 0.37$$

# Multichannel Source Separation with MRF

## - A clustering approach

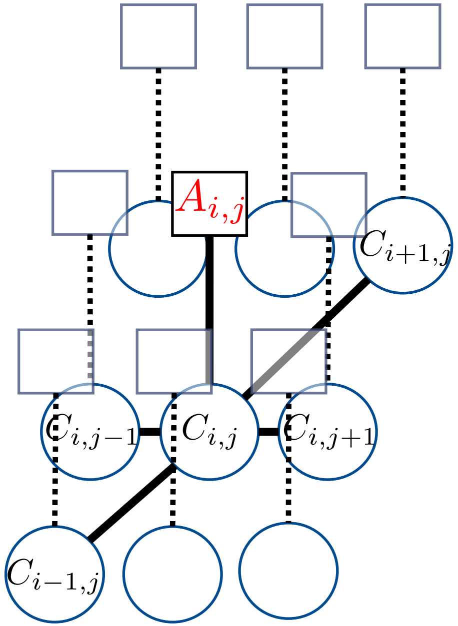
- Inter-channel Level Differences (ILD) can serve as a feature $A_{i,j} = 20 \log \frac{X^R_{i,j}}{X^L_{i,j}}$

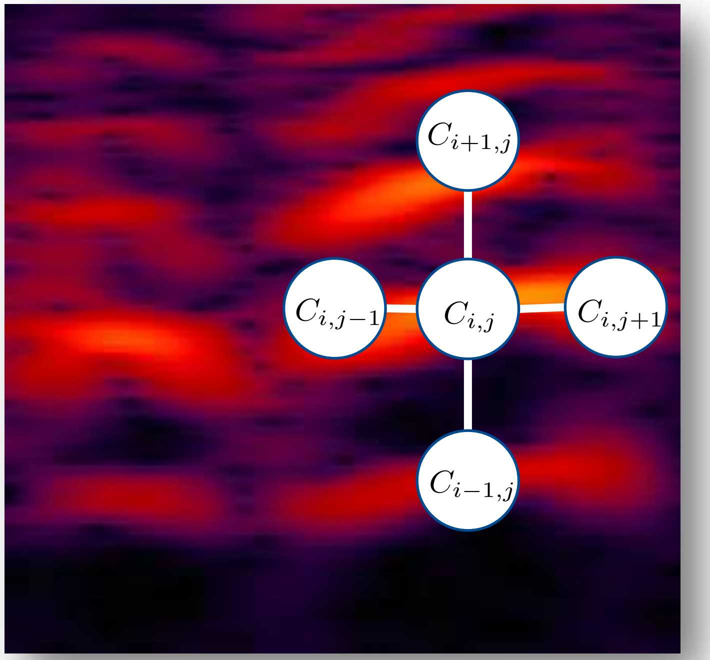



- The goal is to estimate source-wise distributions from their mixture
  - What kind of problem is it?
  - Clustering!

# Multichannel Source Separation with MRF

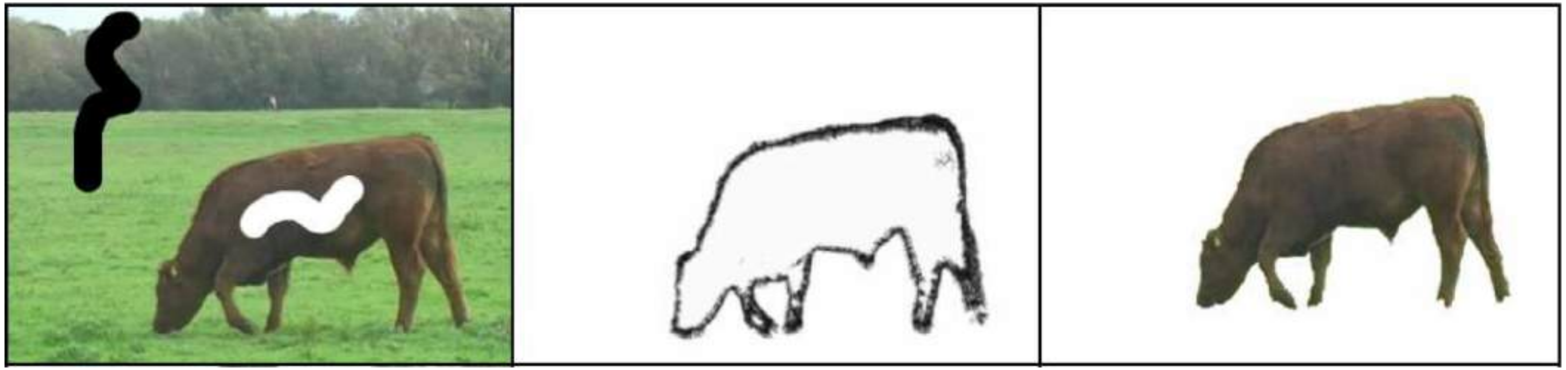
- The same pairwise MRF design



# Multichannel Source Separation with MRF
## - The difference b/w image segmentation. and audio separation

- In the traditional usual image segmentation tasks
  - A user marks pixels to make a guess about the model parameters
    - Means and variances of foreground class and background class



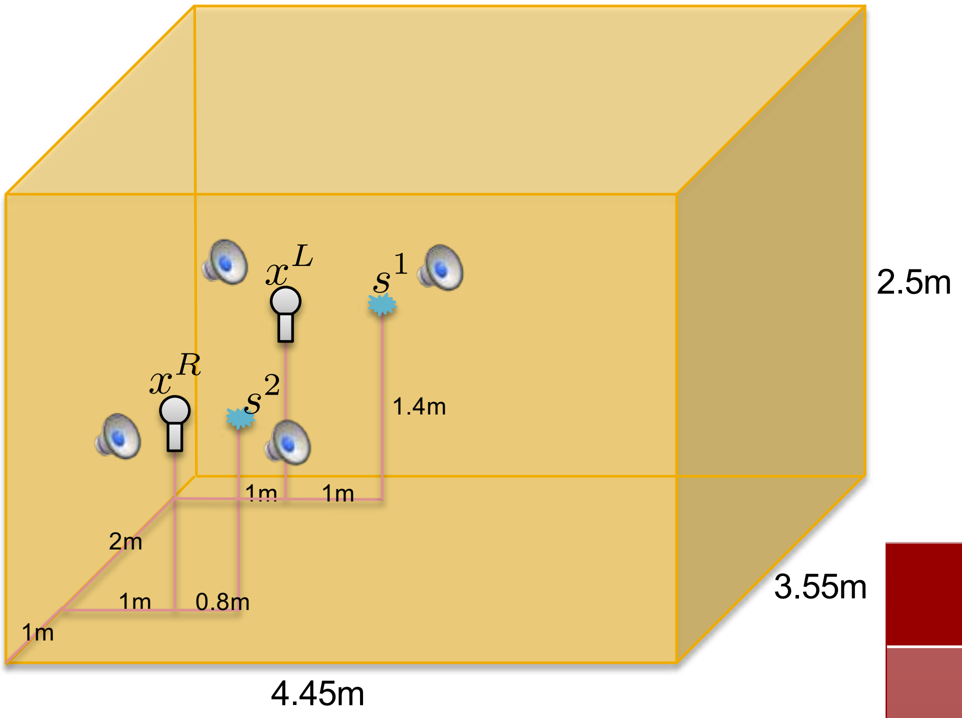
- What if we cannot do that, i.e. in the spectrogram segmentation?
  - That’s why we need EM!



“Dynamic graph cuts for efficient inference in markov random fields,” P Kohli - Pattern Analysis and Machine Intelligence, 2007

# Multichannel Source Separation with MRF

- Two sources

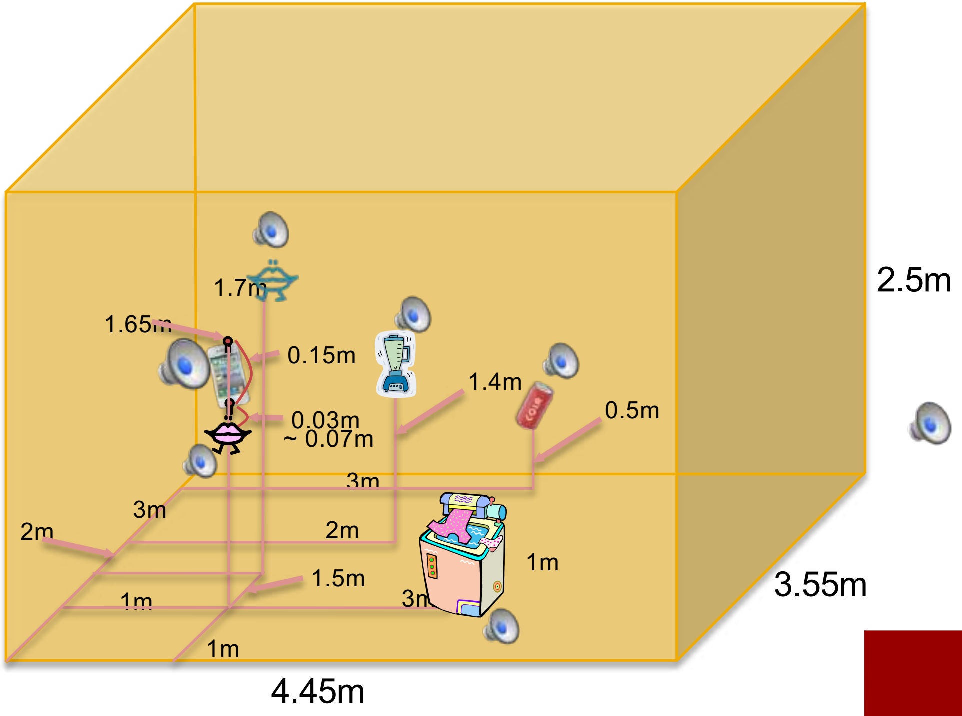



+Improvement from mixture
+Improvement by MRF smoothing

| | | Mixture | Vanilla GMM | MRF Smoothing |
|---|---|---|---|---|
| SDR | S1 | -1.40 | 7.32<br>+8.72 | 8.14<br>+9.54<br>+0.82 |
| | S2 | 1.40 | 8.72<br>+7.32 | 9.54<br>+8.14<br>+0.82 |

# Multichannel Source Separation with MRF

## - The mixing environment (multiple sources)

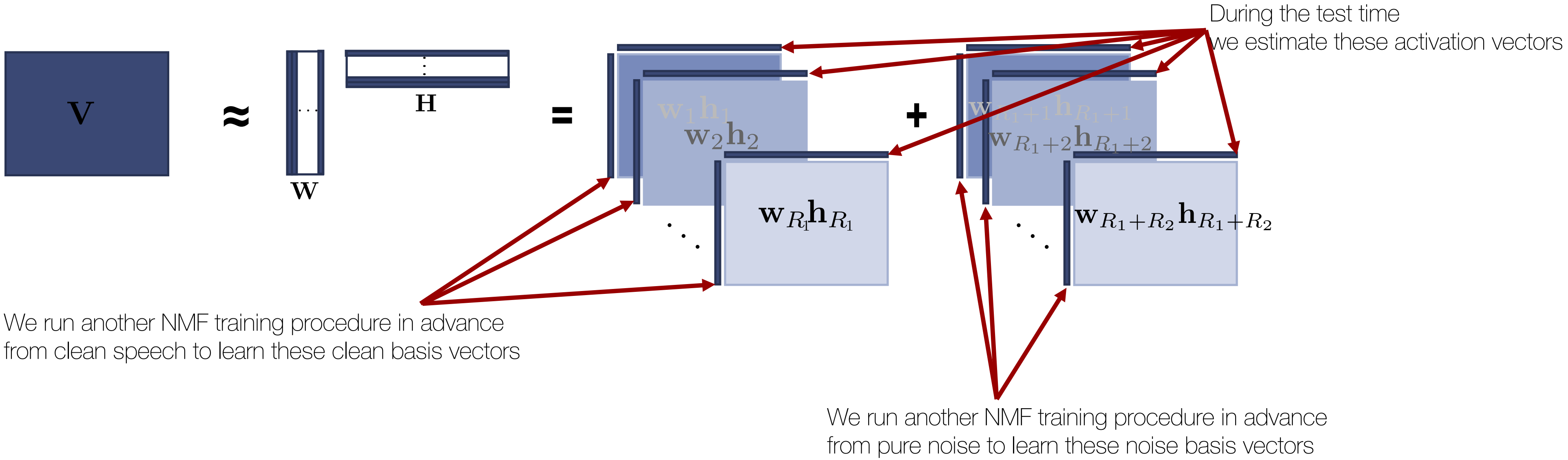



+Improvement from mixture
+Improvement by MRF smoothing

| | Mixture | Vanilla GMM | MRF Smoothing |
|---|---|---|---|
| SDR | 0.06 | 8.08<br>+8.02 | 10.42<br>+10.36<br>+2.34 |

# Monaural Separation Using NMF and MRF
## - NMF

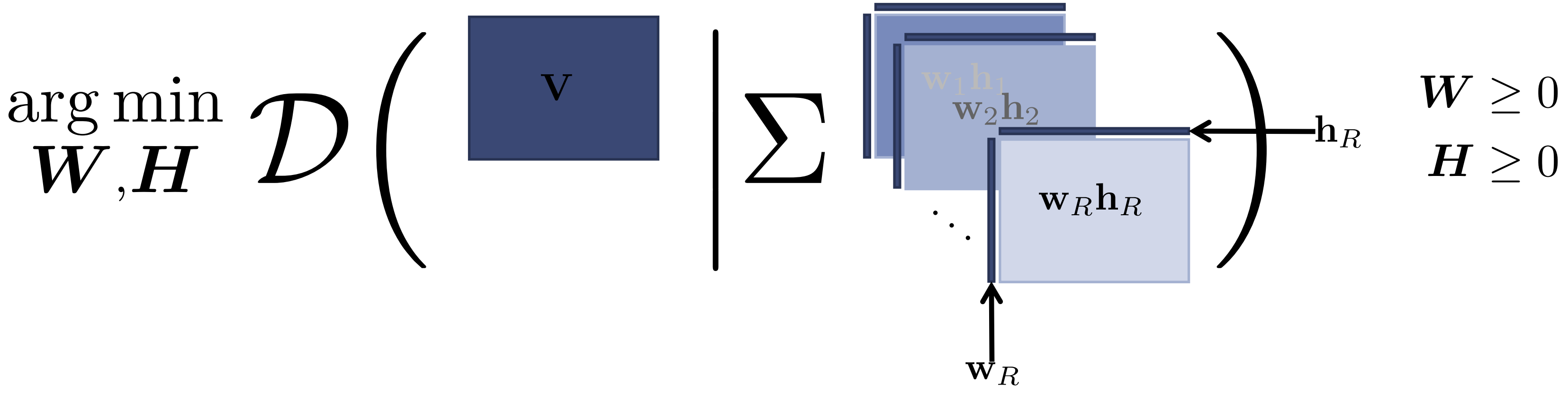
- Nonnegative Matrix Factorization (NMF) for latent components analysis




- Smooth NMF with MRF
  - We see this NMF-based separation problem as a posterior estimation problem
  - Then, we can incorporate MRF smoothing

# Monaural Separation Using NMF and MRF

## - NMF as a Bayesian labeling problem




- ○ NMF with a certain divergence metric (e.g. KL divergence)

$$\mathcal{D}(x|y) = x(\log x - \log y) + (y - x)$$

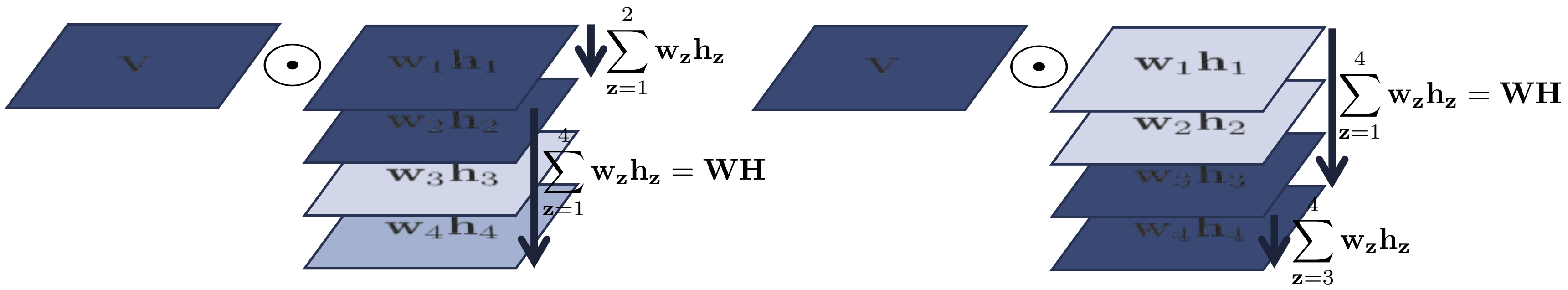
- ○ Multiplicative update rules

$$\boldsymbol{w_z} \leftarrow \boldsymbol{w_z} \odot \frac{\left\{\frac{\boldsymbol{V}}{\boldsymbol{WH}}\right\}\boldsymbol{h_z}^\top}{\mathbf{1}^{M\times N}\boldsymbol{h_z}^\top}, \qquad \boldsymbol{h_z} \leftarrow \boldsymbol{h_z} \odot \frac{\boldsymbol{w_z}^\top\left\{\frac{\boldsymbol{V}}{\boldsymbol{WH}}\right\}}{\boldsymbol{w_z}^\top\mathbf{1}^{M\times N}}.$$

# Monaural Separation Using NMF and MRF

## - NMF as a Bayesian labeling problem

- Let me explain this in the source separation context…
  - We first learn some basis vectors from training (clean) signals of source 1 (target speech) and source 2 (interferences)
  - Let's say that I learned two from source 1 and two from source 2
  - Then, the reconstructions are: $\mathbf{V} = \mathbf{V}^s + \mathbf{V}^n$




$$\mathbf{V}^s \approx \mathbf{V} \odot \frac{\sum_{\mathbf{z}=1}^{2}\mathbf{w_z h_z}}{\mathbf{WH}} = \mathbf{V} \odot \sum_{\mathbf{z}=1}^{2}\mathbf{P_z} \qquad \mathbf{V}^n \approx \mathbf{V} \odot \frac{\sum_{\mathbf{z}=3}^{4}\mathbf{w_z h_z}}{\mathbf{WH}} = \mathbf{V} \odot \sum_{\mathbf{z}=3}^{4}\mathbf{P_z}$$

# Monaural Separation Using NMF and MRF

## - NMF as a Bayesian labeling problem

- We can further break down the update rules into the EM-like formation

$$\boldsymbol{P_z} \leftarrow \frac{\boldsymbol{w_z h_z}}{\boldsymbol{WH}}, \quad \longleftarrow \text{E-step}$$

$$\boldsymbol{w_z} \leftarrow \frac{\left\{\boldsymbol{V} \odot \boldsymbol{P_z}\right\}\mathbf{1}^{N\times 1}}{\mathbf{1}^{M\times N}\boldsymbol{h_z}^\top}, \quad \longleftarrow \text{M-step}$$

$$\boldsymbol{h_z} \leftarrow \frac{\mathbf{1}^{1\times M}\left\{\boldsymbol{V} \odot \boldsymbol{P_z}\right\}}{\boldsymbol{w_z}^\top\mathbf{1}^{M\times N}}. \quad \longleftarrow \text{M-step}$$

Recall:

$$\mathbf{w_z} \leftarrow \mathbf{w_z} \odot \frac{\left\{\frac{\mathbf{V}}{\mathbf{WH}}\right\}\mathbf{h_z}^\top}{\mathbf{1}^{M\times N}\mathbf{h_z}^\top},$$

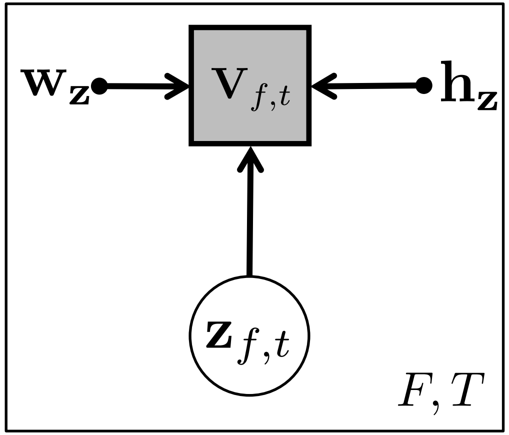
$$\mathbf{h_z} \leftarrow \mathbf{h_z} \odot \frac{\mathbf{w_z}^\top\left\{\frac{\mathbf{V}}{\mathbf{WH}}\right\}}{\mathbf{w_z}^\top\mathbf{1}^{M\times N}}.$$

- In other words, now we can do MRF estimation for NMF

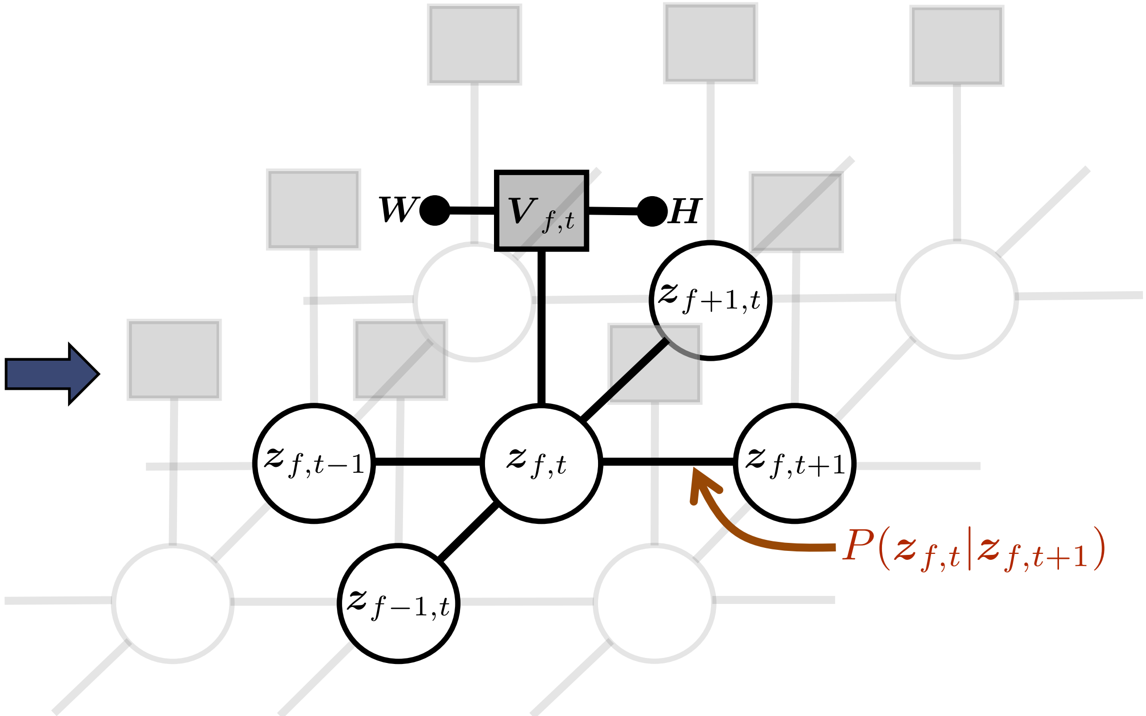
# Monaural Separation Using NMF and MRF

## - NMF as a Bayesian labeling problem

- We can regularize NMF problem by introducing edge potentials (a.k.a. smoothness cost)




NMF as a labeling problem




NMF with MRF smoothing

# Monaural Separation Using NMF and MRF

## - NMF as a Bayesian labeling problem

- The new E-step with smoothing priors

$$
\begin{aligned}
\mathbf{P}_{\mathbf{z}} = P(\mathbf{z}|\mathbf{V}) &\propto \prod_{f,t} P(\mathbf{V}_{f,t}|\mathbf{z}_{f,t}) \prod_{k,l \in \mathcal{N}_{f,t}} P(\mathbf{z}_{f,t}|\mathbf{z}_{k,l}) \\
&= \frac{1}{Z} \prod_{f,t} \phi(\mathbf{z}_{f,t}, \mathbf{V}_{f,t}) \prod_{k,l \in \mathcal{N}_{f,t}} \phi(\mathbf{z}_{f,t}, \mathbf{z}_{k,l}),
\end{aligned}
$$

- Posterior for $(f,t)$-th pixel now involves its neighbors
  - The construction of neighbors is up to applications,
  - but here we assume a Gaussian-like one on the four neighbors once again

$$
P(\boldsymbol{z}_{f,t}|\boldsymbol{z}_{k,l}) \propto \mathrm{e}^{-\left\{f(\boldsymbol{z}_{f,t},\boldsymbol{z}_{k,l})\right\}^2/\sigma_{\mathcal{N}}^2},
$$

$$
f(\boldsymbol{z}_{f,t}, \boldsymbol{z}_{k,l}) = \begin{cases} 0 & \text{if } \boldsymbol{z}_{k,l} \in \mathcal{N}_{f,t} \\ a & \text{otherwise} \end{cases},
$$

- We need an MRF inference routine at every E-step

# Monaural Separation Using NMF and MRF

## - NMF as a Bayesian labeling problem

1. Initialize $W$ and $H$
   - ☐ Fix $W$ with the pre-learned parameters from training signals if you want
2. Repeat until converge
   - ☐ Do this MRF posterior estimation as the E-step using your favorite inference algorithm

$$\max_{\mathbf{z}} \mathbf{P}_{\mathbf{z}} = \max_{\mathbf{z}} \frac{1}{Z} \prod_{f,t} \mathbf{w}_{\mathbf{z}}(f)\mathbf{h}_{\mathbf{z}}(t) \prod_{k,l \in \mathcal{N}_{f,t}} \phi(\mathbf{z}_{f,t}, \mathbf{z}_{k,l}),$$

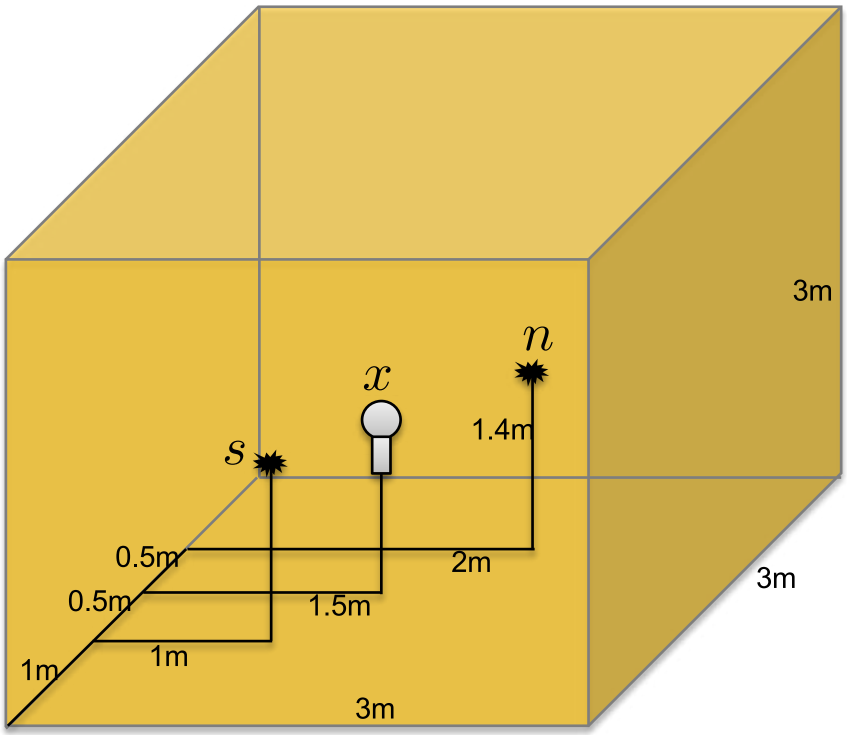
   - ☐ Update parameters as the M-step

$$\boldsymbol{w}_{\boldsymbol{z}} \leftarrow \frac{\left\{\boldsymbol{V} \odot \boldsymbol{P}_{\boldsymbol{z}}\right\}\mathbf{1}^{N\times 1}}{\mathbf{1}^{M\times N}\boldsymbol{h}_{\boldsymbol{z}}^{\top}}, \quad \boldsymbol{h}_{\boldsymbol{z}} \leftarrow \frac{\mathbf{1}^{1\times M}\left\{\boldsymbol{V} \odot \boldsymbol{P}_{\boldsymbol{z}}\right\}}{\boldsymbol{w}_{\boldsymbol{z}}^{\top}\mathbf{1}^{M\times N}}.$$

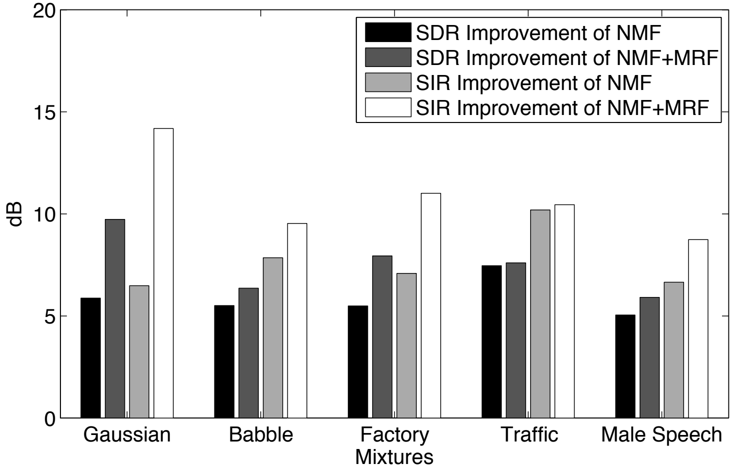
You may want to skip this part if your basis vectors were initialized and fixed with good ones from training

# Monaural Separation Using NMF and MRF

- NMF as a Bayesian labeling problem




| | White | Babble | Factory | Traffic | Male |
|---|---|---|---|---|---|
| Input (0 dB) | | | | | |
| NMF | | | | | |
| NMF +MRF | | | | | |



# Reading

- http://homes.soic.indiana.edu/natarasr/Courses/I590/Papers/MRF.pdf
- Kevin Murphy, “Machine Learning: a Probabilistic Perspective”,
  - Chapter 19: MRF
  - Chapter 24: MCMC
  - Chapter 17: HMM
  - http://site.ebrary.com/lib/iub/detail.action?docID=10597102
- Christopher Bishop: “Pattern Recognition and Machine Learning”
  - Chapter 8: Graphical Models
  - Chapter 11: Sampling Methods

# Thank You!