CPSC 440: Advanced Machine Learning

Mixture Models

Mark Schmidt

University of British Columbia

Winter 2022

Last Time: Adding Features to UGMs

- We discussed adding features to UGMs:

$$p(y_1, y_2, \dots, y_k \mid x_1, x_2, \dots, x_k) \propto \exp\left(\sum_{c=1}^{k} y_c w^T x_c + \sum_{(c,c')\in E} y_c y_{c'} v\right),$$

- Common to use log-linear models.
  - Potentials exponentiate a linear function.
  - Gives exponential family model with convex NLL.
  - But gradient requires inference.
- We discussed approximatons for learning:
  - Pseudo-likelihood trains UGMs as if they were a DAG.
  - Variatiational inference methods can be used.
  - Younes algorithm alternates between MCMC and SGD steps.
- You can also have the potentials be the output of a neural network.

End of Part 4 ("Markov Models"): Key Concepts

- We discussed Markov chains:
  - Distribution assuming independence of past given last time (Markov assumption).
  - Common parameterization uses initial probabilities and transition probabilities.
  - Homogeneous Markov chains assume same transition probabilities across time.

- We discussed inference in Markov chains.
  - Ancestral sampling: sample each variable given previous variables in ordering.
  - CK equations: give marginals recursively.
  - Stationary distribution: marginals as time goes to infinity.
  - Viterbi decoding: special case of dynamic programming.
  - Forward backward: computation of all conditionals with two "passes".

End of Part 4 ("Markov Models"): Key Concepts

- We discussed Markov chain Monte Carlo (MCMC):
  - Define a Markov chain that has target distribution as stationary distribution.
  - Use samples from the Markov chain within Monte Carlo method.
    - Possibly with burn in and/or thinning.
  - Most common methods are Metropolis-Hastings.
    - Based on accepting proposals or keeping the same sample.
  - Special case of Metropolis-Hastings is Gibbs sampling.
    - Based on sampling one variable at a time given all others.

End of Part 4 ("Markov Models"): Key Concepts

- We discussed directed acyclic graphical (DAG).
  - Assume independence of previous variables given a set of parent variables.
  - Can be used to visualize models/assumptions.
  - Conditional independences can be tested using d-separation.
    - Are paths blocked by observed chain/fork, or unobserved child?
  - Our standard independence assumptions appear if we add parameters to DAG.
  - Training DAGs decomposes into d supervised learning problems.

- We discussed undirected graphical models (UGMs).
  - Write distribution as product of non-negative potentials over subsets of variables.
  - Log-linear models use $\exp$(linear) potentials.
    - Convex NLL trained with gradient descent, but gradient requires inference.
  - Approximate training methods include pseudo-likelihood and variational methods.
    - Or Younes algorithm which integrates SGD steps within MCMC.
  - Conditional random fields add features to UGMs.
  - Deep structured models learn features in UGMs.

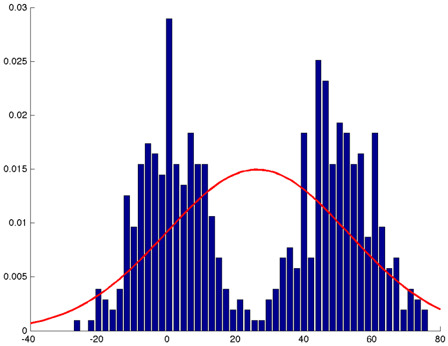
End of Part 4 ("Markov Models"): Key Concepts

- We briefly discussed inference in graphical models.
  - Markov chain inference methods extend to trees for DAGs and UGMs.
  - But for general graphs inference can be hard in DAGs/UGMs.
    - Except unconditional sampling, likelihood, and learning (easy in DAGs).
- We skipped over structured SVMs
  - A generalization of SVMs that can model correlations in labels.
  - Applying SGD requires decoding instead of inference.
  - My slides on this topic are here:
    `https://www.cs.ubc.ca/~schmidtm/Courses/540-W19/L28.5.pdf`

Outline

1 Gaussian for Multi-Modal Data

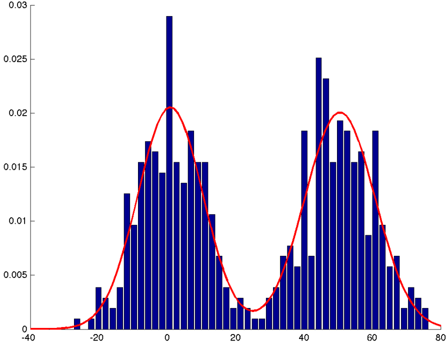
- Major drawback of Gaussian is that it is uni-modal.
  - It gives a terrible fit to data like this:



- If Gaussians are all we know, how can we fit this data?

2 Gaussians for Multi-Modal Data

- We can fit this data by using two Gaussians



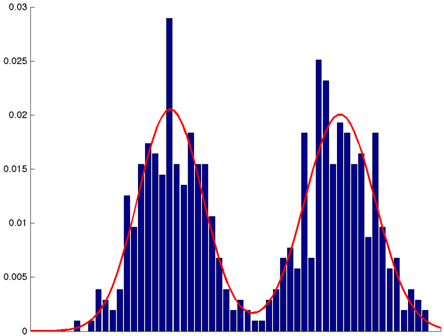
- Half the samples are from Gaussian 1, half are from Gaussian 2.

Mixture of Gaussians

- Our probability density in this example is given by

$$p(x^i \mid \mu_1, \mu_2, \Sigma_1, \Sigma_2) = \frac{1}{2} \underbrace{p(x^i \mid \mu_1, \Sigma_1)}_{\text{PDF of Gaussian 1}} + \frac{1}{2} \underbrace{p(x^i \mid \mu_2, \Sigma_2)}_{\text{PDF of Gaussian 2}},$$

  - We need the (1/2) factors so it still integrates to 1.

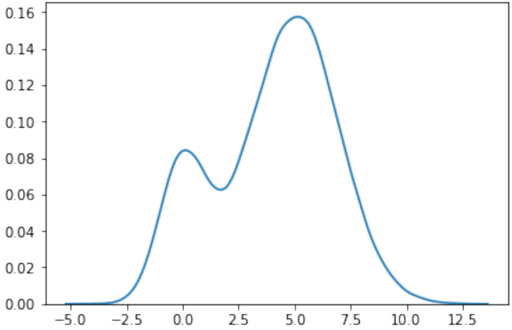


Mixture of Gaussians

- If data comes from one Gaussian more often than the other, we could use

$$p(x^i \mid \mu_1, \mu_2, \Sigma_1, \Sigma_2, \pi_1, \pi_2) = \pi_1 \underbrace{p(x^i \mid \mu_1, \Sigma_1)}_{\text{PDF of Gaussian 1}} + \pi_2 \underbrace{p(x^i \mid \mu_2, \Sigma_2)}_{\text{PDF of Gaussian 2}},$$

  where π_1 and π_2 are non-negative and sum to 1.
  - π_1 represents "probability that we take a sample from Gaussian 1".

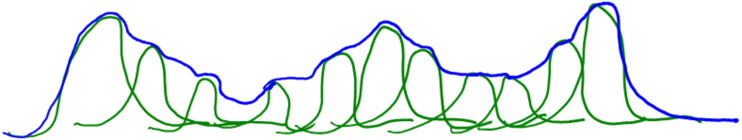


Mixture of Gaussians

- In general we might have a mixture of k Gaussians with different weights.

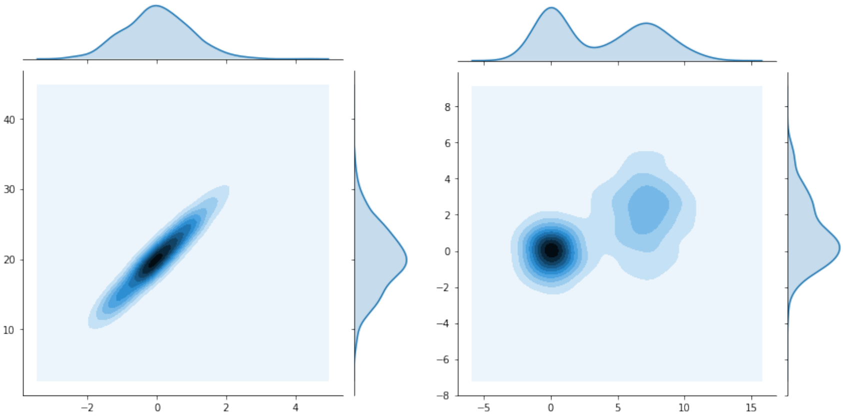
$$p(x \mid \mu, \Sigma, \pi) = \sum_{c=1}^{k} \pi_c \underbrace{p(x \mid \mu_c, \Sigma_c)}_{\text{PDF of Gaussian } c},$$

  - Where π_c are categorical distribution parameters (non-negative and sum to 1).
  - We can use it to model complicated densities with Gaussians (like RBFs).
    - "Universal approximator": can model any continuous density on compact set.



Mixture of Gaussians

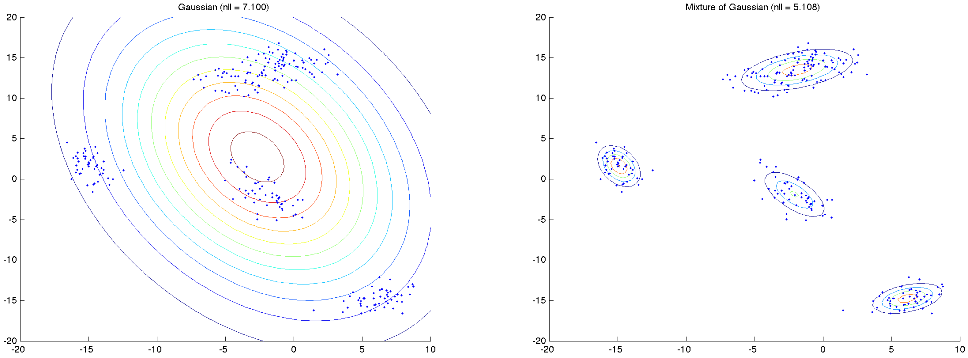
- Gaussian vs. mixture of 2 Gaussian densities in 2D:



- Marginals will also be mixtures of Gaussians.

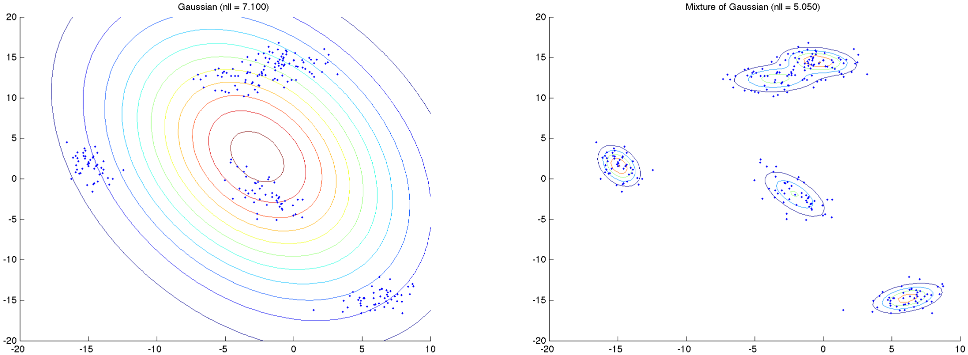
Mixture of Gaussians

- Gaussian vs. Mixture of 4 Gaussians for 2D multi-modal data:



Mixture of Gaussians

- Gaussian vs. Mixture of 5 Gaussians for 2D multi-modal data:



Latent-Variable Representation of Mixtures

- For inference/learning in mixture models, we often introduce variables z^i.
  - Each z^i is a categorical variable in $\{1, 2, \ldots, k\}$ when we have k mixtures.
  - The value z^i represents "what mixture this example came from".
  - We do not observe the z^i values (they are called latent variables).

- Why do mixture have this interpretation of "each x^i comes from one Gaussian"?
  - Consider a model where $p(z^i = c) = \pi_c$, and $x^i \mid z^i = c \sim \mathcal{N}(\mu_c, \Sigma_c)$.
  - Now marginalize over the z^i in this model:

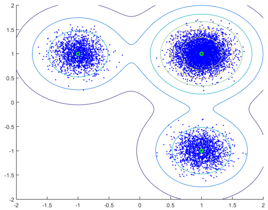
$$p(x \mid \mu, \Sigma, \pi) = \sum_{c=1}^{k} p(x, z = c) = \sum_{c=1}^{k} p(z = c) p(x \mid z = c)$$

$$= \sum_{c=1}^{k} \pi_c \underbrace{p(x \mid \mu_c, \Sigma_c)}_{\text{PDF of Gaussian } c},$$

which is the PDF of the mixture of Gaussians model.

Ancestral Sampling in Mixture of Gaussians

- Generating samples with ancestral sampling in the latent variable representation:
  1. Sample cluster z based on prior probabilities π_c (categorical distribution).
  2. Sample example x based on mean μ_z and covariance Σ_z of Gaussian z.



- Marginalization and computing conditionals is also easy.
- Decoding z or computing marginal $p(z \mid x)$ is easy (next slide).
- Decoding x in Gaussian mixtures is NP-hard.
- We usually fit these models with expectation maximization (EM).
- Choosing k: domain knowledge, test set likelihood, or marginal likellihood.

Inference Task: Computing Responsibilities

- Consider computing probability that example i came from mixture c.
  - We call this the responsibility of mixture c for example i,

$$
\begin{aligned}
r_c^i &= p(z = c \mid x^i) \\
&= \frac{p(z = c, x^i)}{p(x^i)} \\
&= \frac{p(z = c, x^i)}{\sum_{c'=1}^k p(z' = c, x^i} \\
&= \frac{p(z = c)p(x^i \mid z = c))}{\sum_{c'=1}^k p(z' = c)p(x^i \mid z' = c)} \\
&= \frac{\pi_c p(x^i \mid \mu_c, \Sigma_c)}{\sum_{c'=1}^k \pi_{c'} p(x^i \mid \mu_{c'}, \Sigma_{c'})} \qquad \text{(we know all these values)}
\end{aligned}
$$

- If you think the different mixtures as clusters, this is probability of being in cluster.

Notation Alert: π vs. z vs. r (MEMORIZE)

- In mixture models, many people confuse the quantities π, z, and r.
  - Vector π has k elements in $[0, 1]$ and summing up to 1.
    - Number π_c is the "prior" probability that an example is in cluster c.
    - This is a parameter (we learn it from data).
  - Matrix R is $n \times k$ matrix, summing to 1 across rows.
    - Number r_c^i is the "posterior" probability that example i is in cluster c.
    - Computing these values is an inference task (assumes known parameters).
  - Vector z has n elements in $\{1, 2, \ldots, k\}$.
    - Category z^i is the actual mixture/cluster that generated example i.
    - This is a nuissance parameter (an unknown variable that is not a parameter).

Training Mixture Models with Imputation

- Mixture of Gaussian parameters are $\{\pi_c, \mu_c, \Sigma_c\}_{c=1}^k$.
  - Unfortunately, NLL is non-convex and finding MLE is hard.
  - Various optimization methods are used in practice.

- If we treat the z^i as parameters, we get a simple algorithm for decreasing NLL:
  1. Given the clusters z^i, find the most likely parameters.
     - Optimize $p(X \mid \pi, \mu, \Sigma, z)$ in terms of the $\{\pi_c, \mu_c, \Sigma_c\}_{c=1}^k$.
     - Sets π_c based on frequency of seeing $z^i = c$.
     - Sets μ_c to the mean of examples in cluster c.
     - Sets Σ_c to the covariance of examples in cluster c.
  2. Given the parameters, find the most likely clusters.
     - For each example i, compute responsibility $r_c^i = p(z^i = c \mid x^i, \pi_c, \mu_c, \Sigma_c)$.
     - Set z^i to the the argmax of r_c^i over c.

- Connection to Gaussian discriminant analsysis (GDA), using clusters z^i as labels:
  - Step 1 above is the learning step in GDA, Step 2 above is the prediction step in GDA.

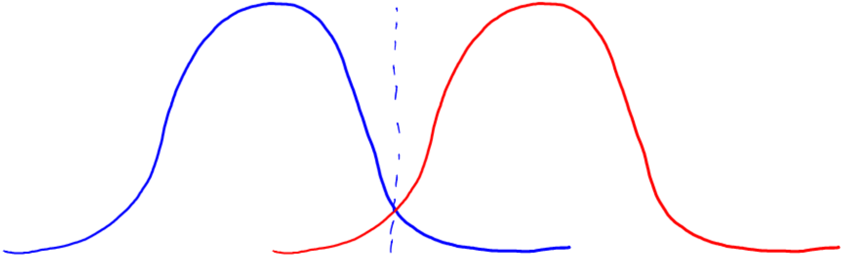
Special Case of K-Means

- Algorithm from the previous slide is a generalization of k-means clustering.
- Apply the algorithm assuming $\pi_c = 1/k$ and $\Sigma_c = I$ for all c:
  1. Given the clusters z^i, find the most likely parameters.
     - Sets μ_c to the mean of examples in cluster c.
  2. Given the parameters, find the most likely clusters.
     - Sets z^i to the closest mean of example i.
- As with k-means, initialization matters for mixture of Gaussians.
  - May need to do multiple random restarts, or clever initializations like k-means++.

K-Means vs. Mixture of Gaussians

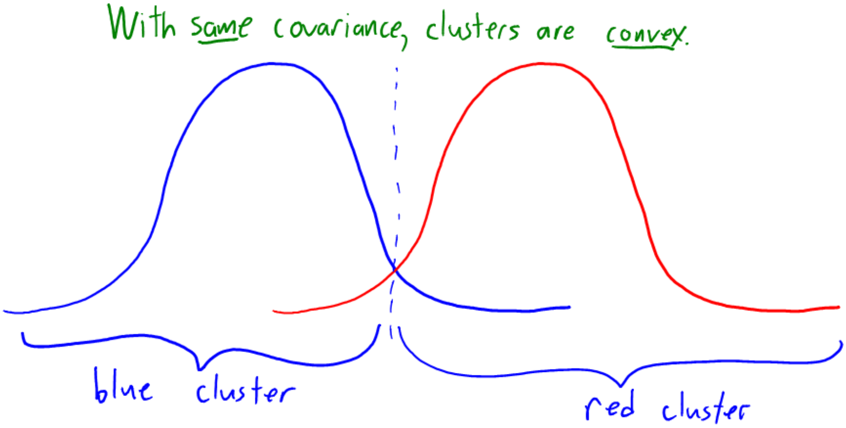
- K-means can be viewed as fitting mixture of Gaussians (same π_c and Σ_c).
  - But variable Σ_c in general mixture of Gaussians allows non-convex clusters.

With same covariance, clusters are convex.



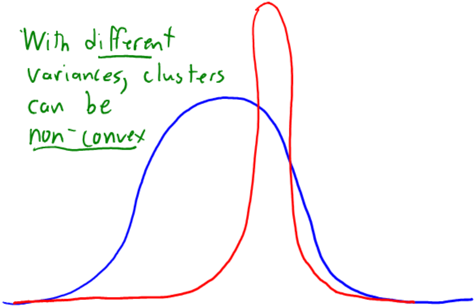
K-Means vs. Mixture of Gaussians

- K-means can be viewed as fitting mixture of Gaussians (same π_c and Σ_c).
  - But variable Σ_c in general mixture of Gaussians allows non-convex clusters.



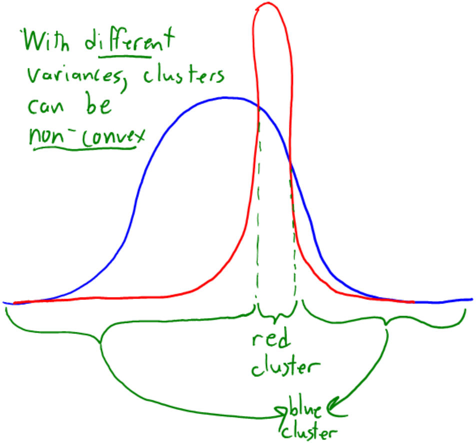
K-Means vs. Mixture of Gaussians

- K-means can be viewed as fitting mixture of Gaussians (same π_c and Σ_c).
  - But variable Σ_c in general mixture of Gaussians allows non-convex clusters.



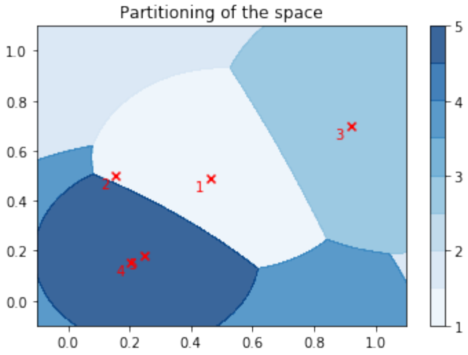
K-Means vs. Mixture of Gaussians

- K-means can be viewed as fitting mixture of Gaussians (same π_c and Σ_c).
  - But variable Σ_c in general mixture of Gaussians allows non-convex clusters.



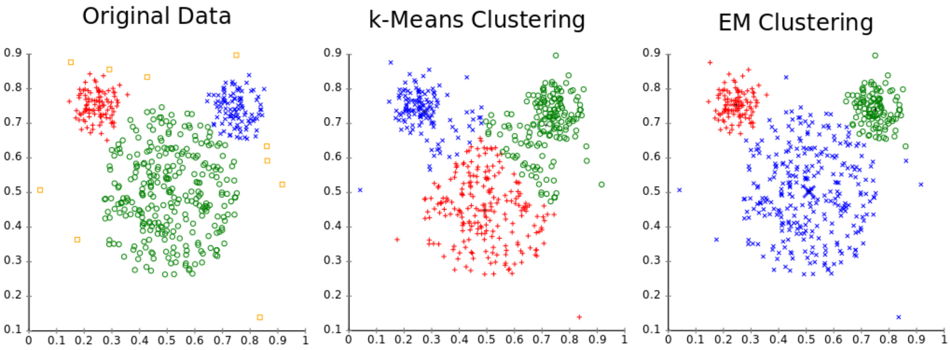
K-Means vs. Mixture of Gaussians

- K-means can be viewed as fitting mixture of Gaussians (same π_c and Σ_c).
  - But variable Σ_c in general mixture of Gaussians allows non-convex clusters.



K-Means vs. Mixture of Gaussians

- K-means can be viewed as fitting mixture of Gaussians (same π_c and Σ_c).
  - But variable Σ_c in general mixture of Gaussians allows non-convex clusters.



https://en.wikipedia.org/wiki/K-means_clustering

Outline

Previously: Product of Bernoullis

- We previously considered density estimation with discrete variables,

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

- We considered a product of Bernoullis:

$$p(x^i \mid \theta) = \prod_{j=1}^{d} p(x^i_j \mid \theta_j).$$

  Easy to fit but strong independence assumption:
  - Knowing x^i_j tells you nothing about x^i_k.

- A more-powerful model is a mixture of Bernoullis.

Mixture of Bernoullis

- Consider a coin flipping scenario where we have two coins:
  - Coin 1 has $\theta_1 = 0.5$ (fair) and coin 2 has $\theta_2 = 1$ (biased).
- Half the time we flip coin 1, and otherwise we flip coin 2:

$$\begin{aligned} p(x^i = 1 \mid \theta_1, \theta_2) &= \pi_1 p(x^i = 1 \mid \theta_1) + \pi_2 p(x^i = 1 \mid \theta_2) \\ &= \frac{1}{2}\theta_1 + \frac{1}{2}\theta_2 = \frac{\theta_1 + \theta_2}{2} \end{aligned}$$

- With one variable this mixture model is not very interesting:
  - It's equivalent to flipping one coin with $\theta = 0.75$.
- But with multiple variables mixture of Bernoullis can model dependencies...

Mixture of Independent Bernoullis

- Consider a mixture of a product of Bernoullis:

$$p(x \mid \theta_1, \theta_2) = \frac{1}{2}\underbrace{\prod_{j=1}^{d} p(x_j \mid \theta_{1j})}_{\text{first set of Bernoullis}} + \frac{1}{2}\underbrace{\prod_{j=1}^{d} p(x_j \mid \theta_{2j})}_{\text{second set of Bernoulli}} .$$

- Conceptually, we now have two sets of coins:
  - Half the time we throw the first set, half the time we throw the second set.

- With $d = 4$ we could have $\theta_1 = \begin{bmatrix} 0 & 0.7 & 1 & 1 \end{bmatrix}$ and $\theta_2 = \begin{bmatrix} 1 & 0.7 & 0.8 & 0 \end{bmatrix}$.
  - Half the time we have $p(x_3^i = 1) = 1$ and half the time it's 0.8.

- Have we gained anything?

Mixture of Independent Bernoullis

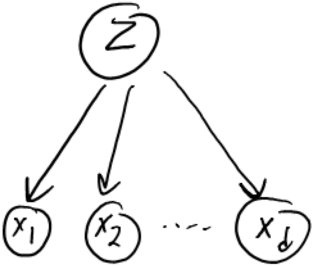
- Example from the previous slide: $\theta_1 = \begin{bmatrix} 0 & 0.7 & 1 & 1 \end{bmatrix}$ and $\theta_2 = \begin{bmatrix} 1 & 0.7 & 0.8 & 0 \end{bmatrix}$.
- Here are some samples from this model:

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

- Unlike product of Bernoullis, notice that features in samples are not independent.
  - In this example knowing $x_1 = 1$ tells you that $x_4 = 0$.
- This model can capture dependencies: $\underbrace{p(x_4 = 1 \mid x_1 = 1)}_{0} \neq \underbrace{p(x_4 = 1)}_{0.5}$.

Mixture of Independent Bernoullis

- Drawing the mixture of Bernoullis as a DAG:



- Since we do not know z, there are dependencies between x_j.
  - But features are independent if we know z.
- This is the same graph as naive Bayes, with cluster z instead of class y.
  - If you see spammy word, it makes other spammy words more likely.

Summary

- Mixture of Gaussians writes probability as convex comb. of Gaussian densities.
  - Can model arbitrary continuous densities.
- Latent-variable representation of mixutres with cluster variables z^i.
  - Allows ancestral sampling by sampling cluster than example.
  - Resonsibility is probability that an example belongs to a cluster.
  - Training by alternating between updating z^i and updating parameters.
- Mixture of Bernoullis can model dependencies between discrete variables.
  - Unsupervised version of naive Bayes.

- Next time: one the top-100 most-cited papers of all time across all fields.

Avoiding Underflow when Computing Responsibilities

- Computing responsibility may underflow for high-dimensional x^i, due to $p(x^i \mid z^i = c, \Theta^t)$.
- Usual ML solution: do all but last step in log-domain.

$$\log r_c^i = \log p(x^i \mid z^i = c, \Theta^t) + \log p(z^i = c \mid \Theta^t) - \log\left(\sum_{c'=1}^{k} p(x^i \mid z^i = c', \Theta^t) p(z^i = c' \mid \Theta^t)\right).$$

- To compute last term, use "log-sum-exp" trick.

Log-Sum-Exp Trick

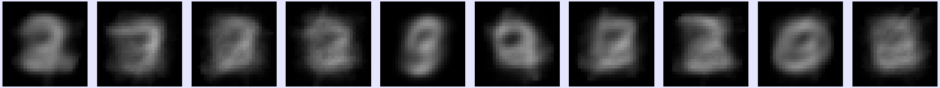
- To compute $\log(\sum_i \exp(v_i))$, set $\beta = \max_i\{v_i\}$ and use:

$$
\begin{aligned}
\log(\sum_c \exp(v_i)) &= \log(\sum_i \exp(v_i - \beta + \beta)) \\
&= \log(\sum_i \exp(v_i - \beta)\exp(\beta)) \\
&= \log(\exp(\beta))\sum_i \exp(v_i - \beta)) \\
&= \log(\exp(\beta)) + \log(\sum_i \exp(v_i - \beta)) \\
&= \beta + \log(\sum_i \underbrace{\exp(v_i - \beta)}_{\leq 1}).
\end{aligned}
$$

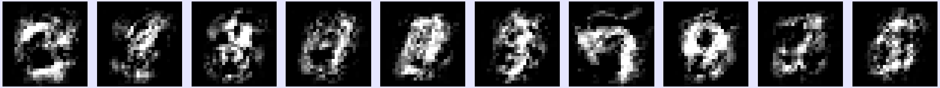
- Avoids overflows due to computing $\exp$ operator.

Mixture of Gaussians on Digits

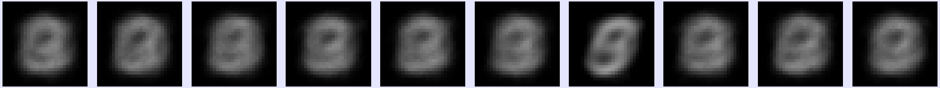
- Mean parameters of a mixture of Gaussians with $k = 10$:



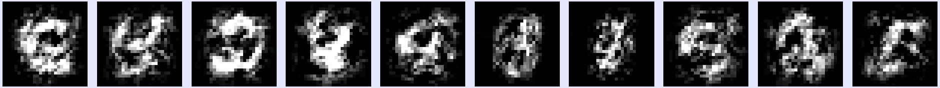
- Samples:



- 10 components with $k = 50$ (I might need a better initialization):



- Samples:



Generative Mixture Models and Mixture of Experts

- Classic generative model for supervised learning uses

$$p(y^i \mid x^i) \propto p(x^i \mid y^i)p(y^i),$$

  and typically $p(x^i \mid y^i)$ is assumed Gaussian (LDA) or independent (naive Bayes).
- But we could allow more flexibility by using a mixture model,

$$p(x^i \mid y^i) = \sum_{c=1}^{k} p(z^i = c \mid y^i)p(x^i \mid z^i = c, y^i).$$

- Another variation is a mixture of discriminative models (like logistic regression),

$$p(y^i \mid x^i) = \sum_{c=1}^{k} p(z^i = c \mid x^i)p(y^i \mid z^i = c, x^i).$$

- Called a "mixture of experts" model:
  - Each regression model becomes an "expert" for certain values of x^i.