CPSC 540: Machine Learning

Generative Classifiers

Mark Schmidt

University of British Columbia

Winter 2020

Last Time: Mixture Models

- We discussed mixture models,

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

  where PDF is written as a convex combination of simple PDFs.

- We discussed Gaussian mixture models and Bernoulli mixture models.
  - With k large, can approximate any continuous/discrete PDF.
- More generally, we can have mixtures of any distributions.
  - Mixture of student t, mixture of categorical, mixture of Poisson, and so on.
- Can choose k using test set likelihood.
  - Except if you assign $p(x^i) = \infty$ to a training point that appears in test set.

Big Picture: Training and Inference

- Mixture model training phase:
  - Input is a matrix X, number of clusters k, and form of individual distributions.
  - Output is mixture model: mixture proportions π_c and parameters of each component.
    - And maybe the "responsibilities": probability of each x^i belonging to each cluster.
- Mixture model prediction phase
  - Input is a model, and possibly test data $\tilde{X}$.
  - Many possible inference tasks. For example:
    - Measure likelihood of test examples $\tilde{x}^i$.
    - Compute probability that test example belongs to cluster c.
    - Compute marginal or conditional probabilities.
    - "Fill in" missing parts of a test example.
- There is also a supervised version of mixture models...

Generative Classifiers: Supervised Learning with Density Estimation

- Density estimation can be used for supervised learning:
  - Generative classifiers estimate conditional by modeling joint probability of x^i and y^i,

$$\begin{aligned} p(y^i \mid x^i) &\propto p(x^i, y^i) && \text{(Approach 1: model joint probability of } x^i \text{ and } y^i\text{)} \\ &= p(x^i \mid y^i)p(y^i). && \text{(Approach 2: model marginal of } y^i \text{ and conditional)} \end{aligned}$$

- Common generative classifiers (based on Approach 2):
  - Naive Bayes models $p(x^i \mid y^i)$ as product of independent distributions.
    - Has recently been used for CRISPR gene editing.
  - Linear discriminant analysis (LDA) assumes $p(x^i \mid y^i)$ is Gaussian (shared Σ).
  - Gaussian discriminant analysis (GDA) allows each class to have its own covariance.
- We can think of these as mixture models.

Naive Bayes as a Mixture Model

- In naive Bayes we assume $x^i \mid y^i$ is a product of Bernoullis.

$$p(x^i, y^i = c) = \underbrace{p(y^i)p(x^i \mid y^i)}_{\text{product rule}} = \underbrace{p(y^i = c)}_{cat} \quad \underbrace{p(x^i \mid y^i)}_{\text{Product(Bernoulli)}} \quad = \pi_c \prod_{j=1}^{d} p(x_j^i \mid \theta_{cj}).$$

- If we don't know y^i, this is actually a mixture of Bernollis model:

$$p(x^i) = \sum_{c=1}^{k} p(x^i, y^i = c) = \sum_{c=1}^{k} \pi_c \prod_{j=1}^{d} p(x_j^i \mid \theta_{cj}).$$

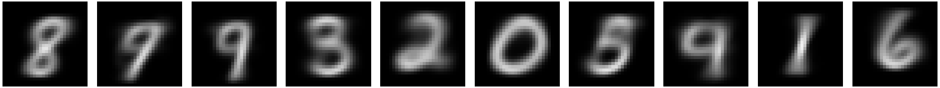
- But since we know which "cluster" each x^i comes from, MLE is simple:

$$\hat{\pi}_c = \frac{n_c}{n}, \quad \theta_{cj} = \frac{1}{n_c} \sum_{y^i = c} x_j^i.$$

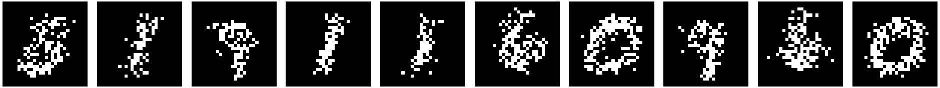
  - "Use the sample statistics for examples in class c".

Naive Bayes on Digits

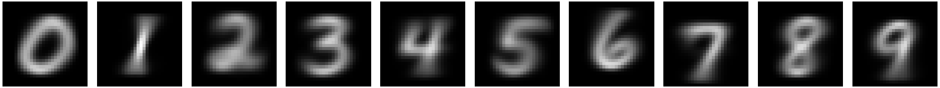
- Parameters of a mixture of Bernoulli model fit to MNIST with $k = 10$:



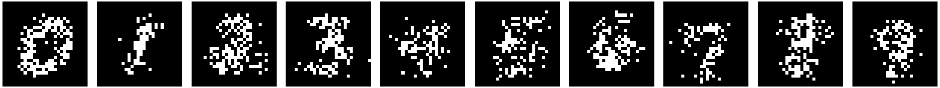
- Shapes of samples are better, but missing within-cluster dependencies:



- For naive Bayes, $\pi_c = 1/10$ for all c and each θ_c corresponds to one class:

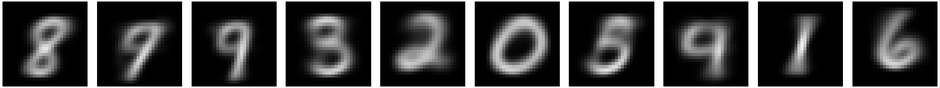


- One sample from each class:

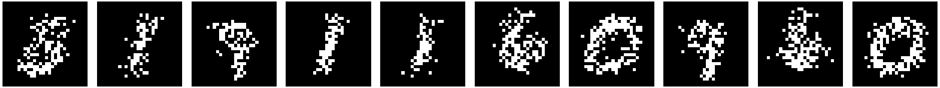


Mixture of Bernoullis on Digits with $k > 10$

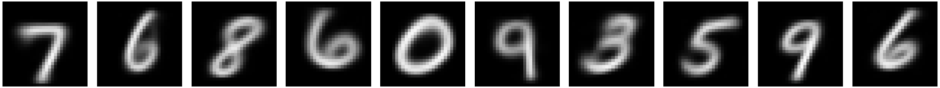
- Parameters of a mixture of Bernoulli model fit to MNIST with $k = 10$:



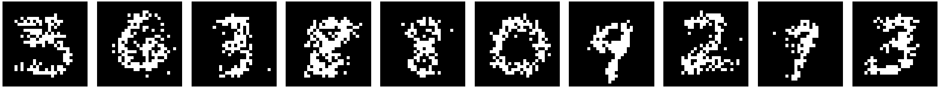
- Shapes of samples are better, but missing within-cluster dependencies:



- You get a better model with $k > 10$. first 10 components with $k = 50$:



- Samples from the $k = 50$ model (can have more than one "type" of a number):



Gaussian Discriminant Analysis (GDA) and Closed-Form MLE

- In Gaussian discriminant analysis we assume $x^i \mid y^i$ is a Gaussian.

$$p(x^i, y^i = c) = \underbrace{p(y^i)p(x^i \mid y^i)}_{\text{product rule}} = \underbrace{\pi_c}_{p(y^i=c)} \underbrace{p(x^i \mid \mu_c, \Sigma_c)}_{\text{Gaussian PDF}}.$$

- If we don't know y^i, this is actually a mixture of Gaussians model:

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

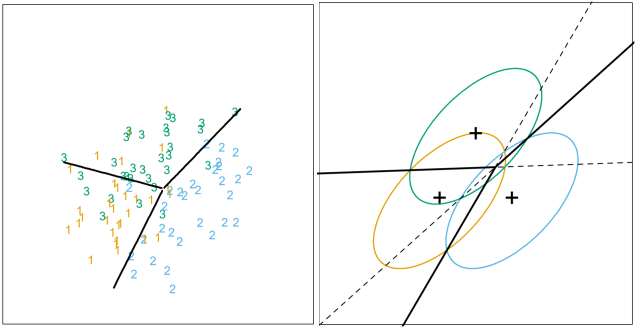
- But since we know which "cluster" each x^i comes from, MLE is simple:

$$\hat{\pi}_c = \frac{n_c}{n}, \quad \hat{\mu}_c = \frac{1}{n_c} \sum_{y^i=c} x^i, \quad \hat{\Sigma}_c = \frac{1}{n_c} \sum_{y^i=c} (x_i - \hat{\mu}_c)(x_i - \hat{\mu}_c)^T,$$

  - "Use the sample statistics for examples in class c".
- In linear discriminant analysis (LDA), we instead use same Σ for all classes.

Linear Discriminant Analysis (LDA)

- Example of fitting linear discriminant analysis (LDA) to a 3-class problem:

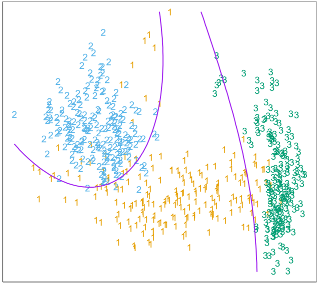


https://web.stanford.edu/~hastie/Papers/ESLII.pdf

- Since variances Σ are equal, class label is determined by nearest mean.
  - Prediction is like a "1-nearest neighbour" or k-means clustering method.
  - This leads to a linear classiifer.

Gaussian Discriminant Analysis (GDA)

- Example of fitting Gaussian discriminant analysis (GDA) to a 3-class problem:



https://web.stanford.edu/~hastie/Papers/ESLII.pdf

- Different Σ_c for each class c leads to a quadratic classifier.
  - Class label is determined by means and variances.

Digression: Generative Models for Structured Prediction

- Consider a structured prediction problem where target y^i is a vector:

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

- Modeling $x^i \mid y^i$ leads to too many y^i potential values.
- But you could model joint probability of x^i and y^i,

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

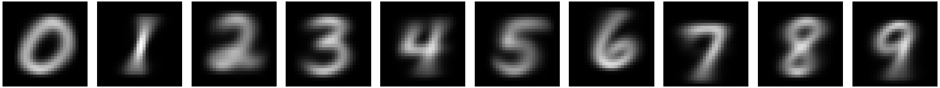
- So any density estimation can be used.
  - Given $p(x^i, y^i)$ use conditioning to get $p(y^i \mid x^i)$ to make predictions.

Digression: Beyond Naive Bayes and GDA

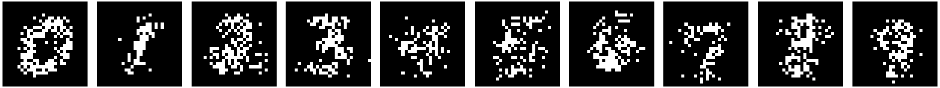
- GDA and naive Bayes make strong assumptions.
  - That features x^i are independent or Gaussian (respectively) given labels y^i.
- You can get a better model of each class by using a mixture model for $p(x^i \mid y^i)$.
  - Or any of the more-advanced methods we'll discuss.
- Generative models were unpopular for a while, but are coming back:
  - Generative adversarial networks (GANs) and variational autoencoders.
    - Deep generative models (later in course).
  - We believe that most human learning is unsupervised.
    - There may not be enough information in class labels to learn quickly.
    - Instead of searching for features that indicate "dog", try to model all aspects of dogs.

Less-Naive Bayes on Digits

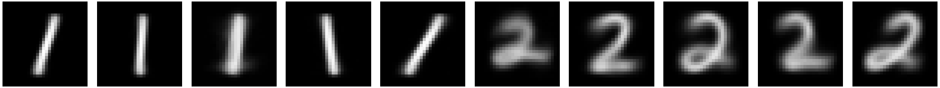
- Naive Bayes θ_c values (independent Bernoullis for each class):



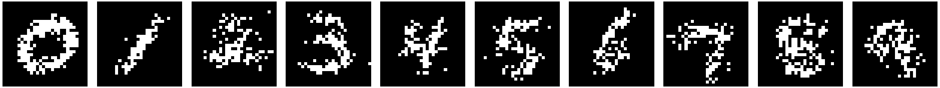
- One sample from each class:



- Generative classifier with mixture of 5 Bernoullis for each class (digits 1 and 2):



- One sample from each class:



Outline

Learning with Hidden Values

- We often want to learn with unobserved/missing/hidden/latent values.
- For example, we could have a dataset like this:

$$X = \begin{bmatrix} N & 33 & 5 \\ L & 10 & 1 \\ F & ? & 2 \\ M & 22 & 0 \end{bmatrix}, y = \begin{bmatrix} -1 \\ +1 \\ -1 \\ ? \end{bmatrix}.$$

- Or we could be fitting a mixture model without knowing the clusters.
- Missing values are very common in real datasets.

- An important issue to consider: why is data missing?

Missing at Random (MAR)

- We'll focus on data that is missing at random (MAR):
  - Assume that the reason ? is missing does not depend on the missing value.
    - Formal definition in bonus slides.
  - This definition doesn't agree with intuitive notion of "random":
    - A variable that is *always* missing would be "missing at random".
    - The intuitive/stronger version is missing completely at random (MCAR).
- Examples of MCAR and MAR for digit data:
  - Missing random pixels/labels: MCAR.
  - Hide the the top half of every digit: MAR.
  - Hide the labels of all the "2" examples: not MAR.
- We'll consider MAR, because otherwise you need to model why data is missing.

Imputation Approach to MAR Variables

- Consider a dataset with MAR values:

$$X = \begin{bmatrix} N & 33 & 5 \\ F & 10 & 1 \\ F & ? & 2 \\ M & 22 & 0 \end{bmatrix}, y = \begin{bmatrix} -1 \\ +1 \\ -1 \\ ? \end{bmatrix}.$$

- Imputation method is one of the first things we might try:
  0. Initialization: find parameters of a density model (often using "complete" examples).
  1. Imputation: replace each ? with the most likely value.
  2. Estimation: fit model with these imputed values.

- You could also alternate between imputation and estimation.

Semi-Supervised Learning

- Important special case of MAR is semi-supervised learning.

$$X = \begin{bmatrix} \quad \end{bmatrix}, \quad y = \begin{bmatrix} \end{bmatrix},$$

$$\bar{X} = \begin{bmatrix} \quad \end{bmatrix}, \quad \bar{y} = \begin{bmatrix} ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix}.$$

- Motivation for training on labeled data (X, y) and unlabeled data $\bar{X}$:
  - Getting labeled data is usually expensive, but unlabeled data is usually cheap.
    - For speech recognition: easy to get speech data, hard to get annotated speech data.

Semi-Supervised Learning

- Important special case of MAR is semi-supervised learning.

$$X = \begin{bmatrix} \\ \\ \end{bmatrix}, \quad y = \begin{bmatrix} \\ \\ \end{bmatrix},$$

$$\bar{X} = \begin{bmatrix} \\ \\ \\ \\ \\ \end{bmatrix}, \quad \bar{y} = \begin{bmatrix} ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix},$$

- Imputation approach is called self-taught learning:
  - Alternate between guessing $\bar{y}$ and fitting the model with these values.

Back to Mixture Models

- To fit mixture models we often introduce n MAR variables z^i.
- Why???
- Consider mixture of Gaussians, and let z^i be the cluster number of example i:
  - So $z^i \in \{1, 2, \cdots, k\}$ tells you which Gaussian generated example i.
  - Given the z^i it's easy to optimize the parameters of the mixture model.
    - Solve for $\{\pi_c, \mu_c, \Sigma_c\}$ maximizing $p(x^i, z^i)$ (learning step in GDA).
  - Given $\{\pi_c, \mu_c, \Sigma_c\}$ it's easy to optimize the clusters z^i:
    - Find the cluster c maximizing $p(x^i, z_i = c)$ (prediction step in GDA).

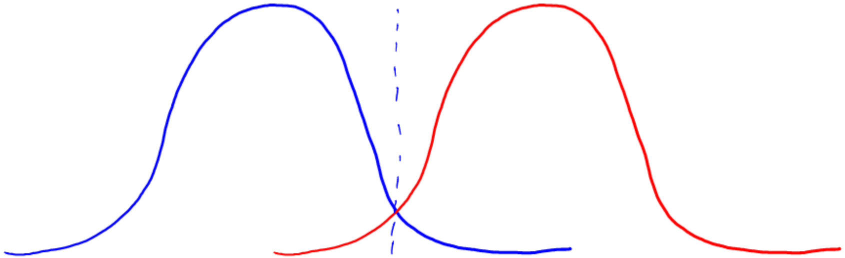
Imputation Approach for Mixtures of Gaussians

- Consider mixture of Gaussians with the choice $\pi_c = 1/k$ and $\Sigma_c = I$ for all c.
- Here is the imputation approach for fitting a mixtures of Gaussian:
  - Randomly pick some initial means μ_c.
  - Assigns x^i to the closest mean (classification rule with same variances).
    - This is how you maximize $p(x^i, z^i)$ in terms of z^i.
  - Set μ_c to the mean of the points assigned to cluster c (Gaussian MLE for cluster).
    - This is how you maximize $p(x^i, z^i)$ in terms of μ_c.
- This is exactly k-means clustering.

K-Means vs. Mixture of Gaussians

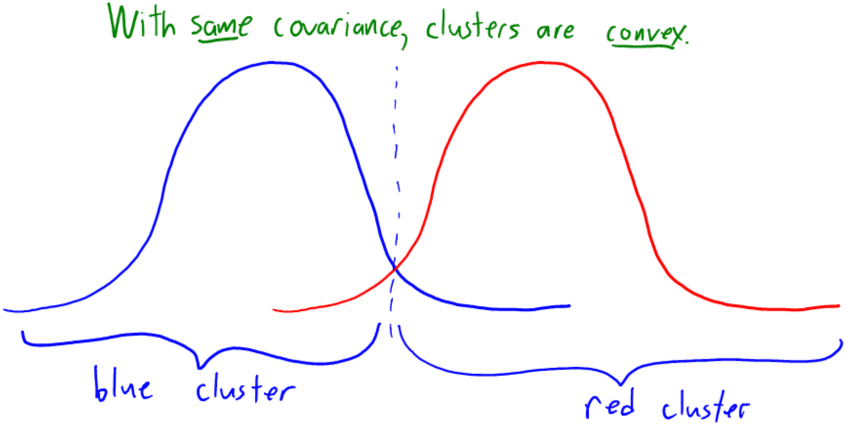
- K-means can be viewed as fitting mixture of Gaussians (common Σ_c).
  - But variable Σ_c in mixture of Gaussians allow non-convex clusters.

With same covariance, clusters are convex.



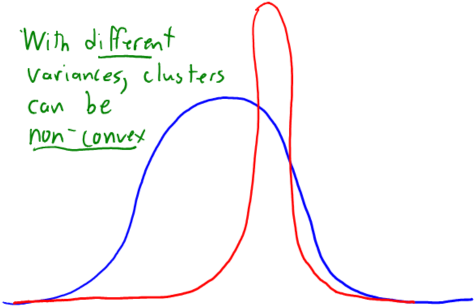
K-Means vs. Mixture of Gaussians

- K-means can be viewed as fitting mixture of Gaussians (common Σ_c).
  - But variable Σ_c in mixture of Gaussians allow non-convex clusters.



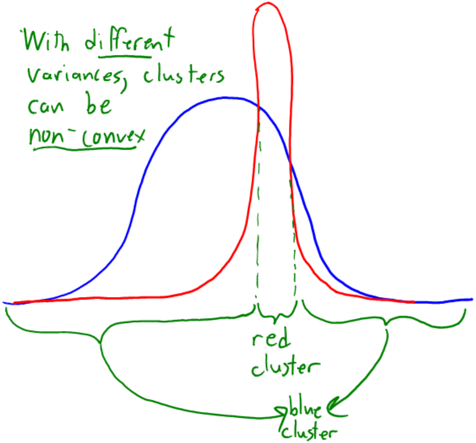
K-Means vs. Mixture of Gaussians

- K-means can be viewed as fitting mixture of Gaussians (common Σ_c).
  - But variable Σ_c in mixture of Gaussians allow non-convex clusters.



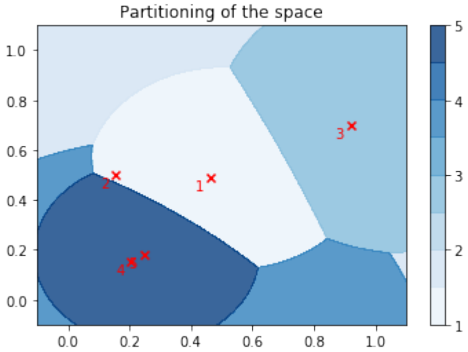
K-Means vs. Mixture of Gaussians

- K-means can be viewed as fitting mixture of Gaussians (common Σ_c).
  - But variable Σ_c in mixture of Gaussians allow non-convex clusters.



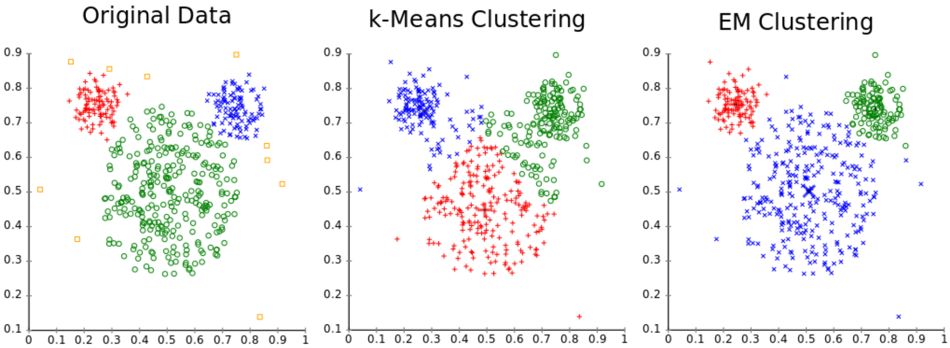
K-Means vs. Mixture of Gaussians

- K-means can be viewed as fitting mixture of Gaussians (common Σ_c).
  - But variable Σ_c in mixture of Gaussians allow non-convex clusters.



K-Means vs. Mixture of Gaussians

- K-means can be viewed as fitting mixture of Gaussians (common Σ_c).
  - But variable Σ_c in mixture of Gaussians allow non-convex clusters.



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

Drawbacks of Imputation Approach

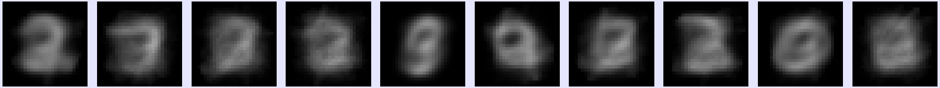
- The imputation approach to MAR variables is simple:
  - Use density estimator to "fill in" the missing values.
  - Now fit the "complete data" using a standard method.
- But "hard" assignments of missing values lead to propagation of errors.
  - What if cluster is ambiguous in k-means clustering?
  - What if label is ambiguous in "self-taught" learning?
- Ideally, we should use probabilities of different assignments ("soft" assignments):
  - If the MAR values are obvious, this will act like the imputation approach.
  - For ambiguous examples, takes into account probability of different assignments.
- Expectation maximization (EM) considers probability of all imputations of ?.

Summary

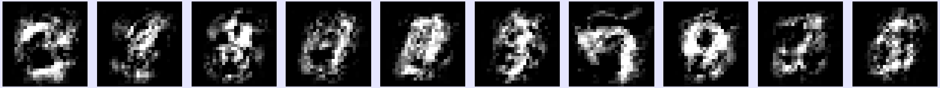
- Generative classifiers turn supervised learning into density estimation.
  - Naive Bayes and GDA are popular, but make strong assumptions.
  - Can be used for structured prediction.
- Missing at random: fact that variable is missing does not depend on its value.
- Imputation approach to handling missing data.
  - Guess values of hidden variables, then fit the model (and usually repeat).
  - K-means is a special case, if we introduce "cluster number" as MAR variables.
- Next time: one of the most cited papers in statistics.

Mixture of Gaussians on Digits

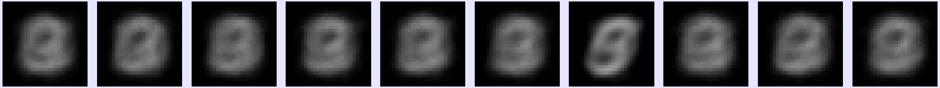
- Mean parameters of a mixture of Gaussians with $k = 10$:



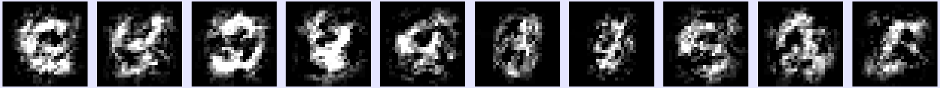
- Samples:



- 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.

Missing at Random (MAR) Formally

- Let's formally define MAR in the context of density estimation.
- Our "observed" data would be a matrix X containing ? values.
- Our "complete" data would be the matrix X the ? values "filled in".
  - Let x_j^i be the value in this matrix, which may be a ? in the observed data.
- Use $z_j^i = 1$ if x_i^j is ? in the "observed" data.
- We say that data is MAR in the observed data X if

$$z_j^i \perp x_j^i,$$

that the fact that x_j^i is missing (z_j^i) is independent of the value of x_j^i.
  - Specific values of the variables are not being hidden.