CPSC 440: Advanced Machine Learning

EM and KDE

Mark Schmidt

University of British Columbia

Winter 2022

Last Time: Mixture Models

- We discussed mixture models,

$$p(x \mid \pi, \Theta) = \sum_{c=1}^{k} \pi_c p(x \mid \Theta_c),$$

  where PDF is written as a convex combination of simple PDFs.
- We discussed mixture of Gaussians.
  - Unsupervised version of Gaussian discriminant analysis.
  - Universal density estimator for continuous densities.
- We started discussing mixture of Bernoullis.
  - Unsupervised version of naive Bayes (with binary features).
  - Can model dependencies between features.
- We discussed interpreting mixtures in terms of latent variables z^i.
  - Represent mixture/cluster that generated example i.
  - We define responsibility of cluster c for example i as $r_c^i = p(z^i = c \mid x^i, \Theta)$.
- We discussed imputation approach to learning with latent variables:
  - Alternate between finding best value of latent variables, and updating parameters.

Mixture of Independent Bernoullis

- General mixture of independent Bernoullis:

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

  where Θ contains all the model parameters.
  - Θ has k values of π_c and $k \times d$ values of θ_{cj}.

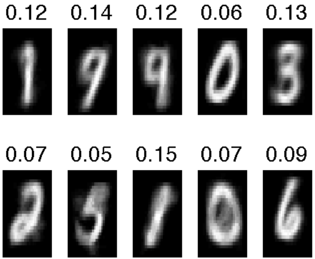
- Mixture of Bernoullis can model dependencies between variables
  - Individual mixtures act like clusters of the binary data.
  - Knowing cluster of one variable gives information about other variables.

- With k large enough, mixture of Bernoullis can model any binary distribution.
  - Hopefully with $k << 2^d$.

Mixture of Independent Bernoullis

- Plotting parameters θ_c with 10 mixtures trained on MNIST digits (with "EM"):

(numbers above images are mixture coefficients π_c)

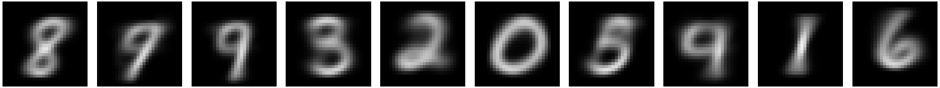


http://pmtk3.googlecode.com/svn/trunk/docs/demoOutput/bookDemos/%2811%29-Mixture_models_and_the_EM_algorithm/mixBerMnistEM.html

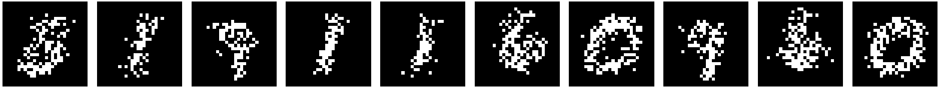
- Remember this is unsupervised: it hasn't been told there are ten digits.
  - You could use this model to "fill in" missing parts of an image.

Mixture of Bernoullis on Digits with $k > 10$

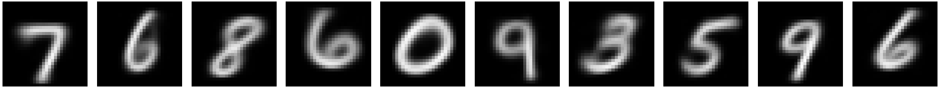
- Parameters of a mixture of Bernoulli model fit to MNIST with $k = 10$:



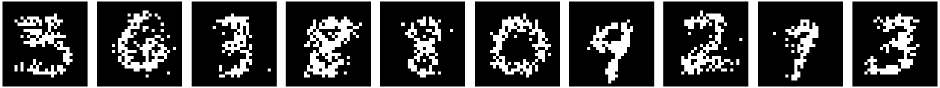
- Samples better than product of Bernoullis (but no within-cluster dependency):



- 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):



Outline

1. Expectation Maximization
2. Advanced Mixtures and KDE

Big Picture: Training and Inference

- Many possible mixture model inference tasks:
  - Generate samples.
  - Measure likelihood of test examples $\tilde{x}^i$.
    - To detect outliers, for example.
  - Compute probability that test example belongs to cluster c.
  - Compute marginal or conditional probabilities.
  - "Fill in" missing parts of a test example.
- Mixture model training phase:
  - Input is a matrix X, number of clusters k, and form of individual distributions.
  - Output is mixture proportions π_c and parameters of components.
    - The θ_c for Bernoulli, and the $\{\mu_c, \Sigma_c\}$ for Gaussians.
    - And maybe the responsibilities r_c^i or cluster assignments z^i.

Fitting a Mixture of Bernoullis: Imputation of z^i

- Imputation approach to ftting mixture of Bernoullis if we view z^i as parameters:
  1. Find the most likely cluster z^i for each example i,
  $$z^i \in \operatorname*{argmax}_c p(z^i = c \mid x^i, \Theta).$$
  2. Update the mixture probabilities as proportion of examples in cluster,
  $$\pi_c = \frac{1}{n}\sum_{i=1}^n I[z^i = c].$$
  3. Update the product of Bernoullis based on examples in cluster,
  $$\theta_{cj} = \frac{1}{n_c}\sum_{i=1}^n I[z^i = c]x_j^i.$$
- You can think of this as doing exact assignments to the z^i variables.

Fitting a Mixture of Bernoullis: Expectation Maximization

- Expectation maximization (EM) approach to ftting mixture of Bernoulli:
  1. Find the responsibility of cluster z^i for each example i
  $$r_c^i = p(z^i = c \mid x^i, \Theta).$$
  2. Update the mixture probabilities as proportion of examples cluster is responsible for,
  $$\pi_c = \frac{1}{n}\sum_{i=1}^{n} r_c^i.$$
  3. Update the product of Bernoullis based on examples cluster is responsible for,
  $$\theta_{cj} = \frac{1}{\sum_{i=1}^{n} r_c^i}\sum_{i=1}^{n} r_c^i x_j^i.$$
- You can think of this as doing probabilistic assignment to the z^i variables.

Fitting a Mixture of Gaussians: Expectation Maximization

- Expectation maximization (EM) approach to ftting mixture of Gaussians:
  1. Find the responsibility of cluster z^i for each example i
  $$r_c^i = p(z^i = c \mid x^i, \Theta).$$
  2. Update the mixture probabilities as proportion of examples cluster is responsible for,
  $$\pi_c = \frac{1}{n}\sum_{i=1}^n r_c^i.$$
  3. Update the Gaussian based on examples cluster is responsible for,
  $$\mu_c = \frac{1}{\sum_{i=1}^n r_c^i}\sum_{i=1}^n r_c^i x^i, \quad \Sigma_c = \frac{1}{\sum_{i=1}^n r_c^i}\sum_{i=1}^n r_c^i (x^i - \mu_c)(x^i - \mu_c)^T.$$
- Video: `https://www.youtube.com/watch?v=B36fzChfyGU`

Expectation Maximization vs. Imputation

- The imputation method is optimizing $p(x^i, z^i \mid \Theta)$ in terms of z^i and Θ.
  - So we are optimizing z^i.
    - $p(x^i, z^i \mid \Theta)$ is called the complete-data likelihood.
- Expectation maximization (EM) is optimizing $p(x^i \mid \Theta)$ in terms of Θ.
  - So we are integrating over z^i values.
    - $p(x^i \mid \Theta)$ is the usual likelihood, marginalizing over the z^i.
- EM is a general algorithm for parameter learning with missing data.
  - For mixtures, the "missing" data is the z^i variables.
  - But EM can be used for any probabilistic model where we have missing data.

Expectation Maximization: General Form

- With data X and hidden values Z, the general EM uses iterations of the form

$$\Theta^{t+1} \in \underset{\Theta}{\text{argmax}} \sum_{Z} p(Z \mid X, \Theta^t) \log p(X, Z \mid \Theta)$$
$$\equiv \underset{\Theta}{\text{argmax}}\, \mathbb{E}_{Z \mid X, \Theta^t}[\log p(X, Z \mid \Theta)].$$

- Summing/integrating over all possible hidden values Z may be hard.
  - But in many cases this simplifies due to conditional independence assumptions.

- For mixture models, the EM iteration simplifies to (see notes on webpage)

$$\sum_{i=1}^{n} \sum_{z^i=1}^{k} \underbrace{p(z^i \mid x^i, \Theta^t)}_{\text{responsibility}} \underbrace{\log p(x^i, z^i \mid \Theta)}_{\text{complete-data log-lik}},$$

so summing over k^n possible clusterings turns into sum over nk terms.

"E-Step" and "M-Step" for Mixture Models

- For mixture models, EM is often written as two steps:
  1. E-step: compute responsibilities r_c^i for all i and c for current Θ^t.
  2. M-step: optimize the weighted "complete-data" log-likelihood

$$\Theta^{t+1} \in \underset{\Theta}{\text{argmax}} \sum_{i=1}^{n} \sum_{z^i=1}^{k} r_c^i \log p(x^i, z^i \mid \Theta).$$

- For other models, there may no separate "E-step" and "M-step".

- EM is most useful when complete-data log-likelihood is easy to optimize.

- Most common case: complete-data log-likelihood is in exponential-family.
  - Mixture of Bernoullis, mixture of Gaussians, and many other cases.
  - In this case the M-step is a weighted combination of the sufficient statistics.

Expectation Maximization Algorithm: Properties

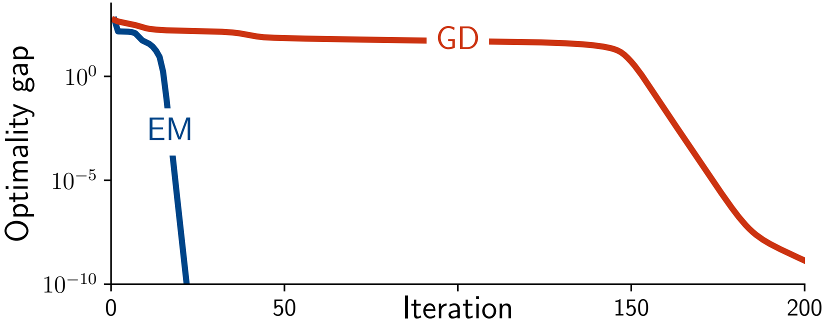
- EM monotonically increases likelihood, $p(X \mid \Theta^{t+1}) \geq p(X \mid \Theta^{t})$.
  - This is useful for debugging: if likelihood decreases you have a bug.
- EM does not need a step size, unlike many learning algorithms.
- EM tends to satisfy constraints automatically.
  - Unlike gradient descent, do not need to worry about constraints on π_c and Σ_c.
    - Assuming you have a prior to avoid degenerate situations where MLE does not exist.
- EM iterations are parameterization independent.
  - Get same performance under any re-parameterization of the problem.
- EM is notorious for converging to bad local optima.
  - But this is not the algorithm's fault: we typically apply EM to hard problems.

Expectation Maximization Algorithm: Properties

- EM converges to a stationary point under weak assumptions.
- EM is at least as fast as gradient descent (with a constant step size).
  - In worst case for differentiable problems.
  - And EM can also be used for non-differentiable likelihoods.
- EM converges faster as entropy of hidden variables decreases.
  - If value of hidden variables is "obvious", it converges very fast.
- And EM can be arbitrarily faster than gradient descent.
- I put a bunch of more-detailed material on the EM algorithm here:
  - `https://www.cs.ubc.ca/~schmidtm/Courses/440-W22/L34.5.pdf`

Expectation Maximization vs. Gradient Descent

- Expectation maximization vs. gradient for fitting mixture of 2 Gaussians:



- Show video.

Outline

1 Expectation Maximization

2 Advanced Mixtures and KDE

Combining Mixture Models with Other Models

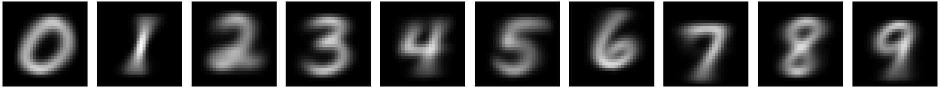
- We can use mixtures in generative classifiers.
  - Model $p(x \mid y)$ as a mixture instead of simple Gaussian or product of Bernoullis.
    - VQNB from Assignment 2 fits a mixture of Bernoullis for each class.

- We can do mixture of more-complicated distributions:
  - Mixture of categoricals (can model arbitrary categorical vectors).
  - Mixture of student t distributions.
    - Not exponential family so no simple closed-form update of parameters.
  - Mixture of Markov chains for rain data (later).
  - Mixture of DAGs/UGMs (could be tree-structured for easy inference).
    - Captures both clusters and dependencies between variables in clusters.

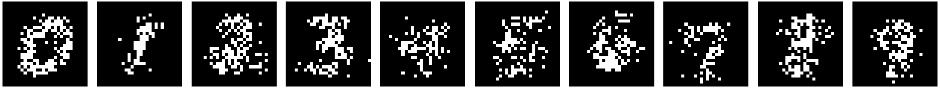
- We can add features to mixture models for supervised learning:
  - Mixture of experts: have k regression/classification models.
    - Each model can be viewed as a "expert" for a cluster of x^i values.

Less-Naive Bayes on Digits

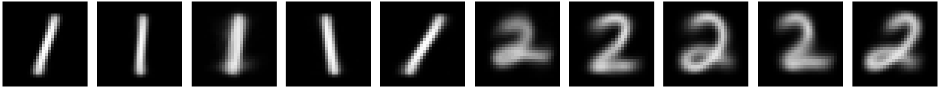
- Naive Bayes θ_c values (independent Bernoullis for each class):



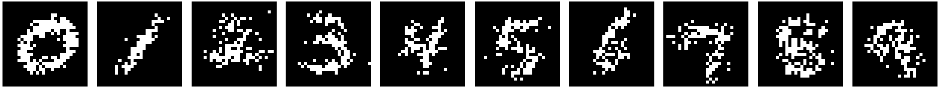
- One sample from each class:



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



- One sample from each class:



- Would get less noisy samples and more variation with mixture of graphical models.

Dirichlet Process

- Non-parametric Bayesian methods allow us to consider infinite mixture model,

$$p(x \mid \Theta) = \sum_{c=1}^{\infty} \pi_c p(x \mid \Theta_c).$$

- Common choice for prior on π values is Dirichlet process:
  - Also called "Chinese restaurant process" and "stick-breaking process".
  - For finite datasets, only a fixed number of clusters have $\pi_c \neq 0$.
  - But do not need to pick number of clusters, grows with data size.

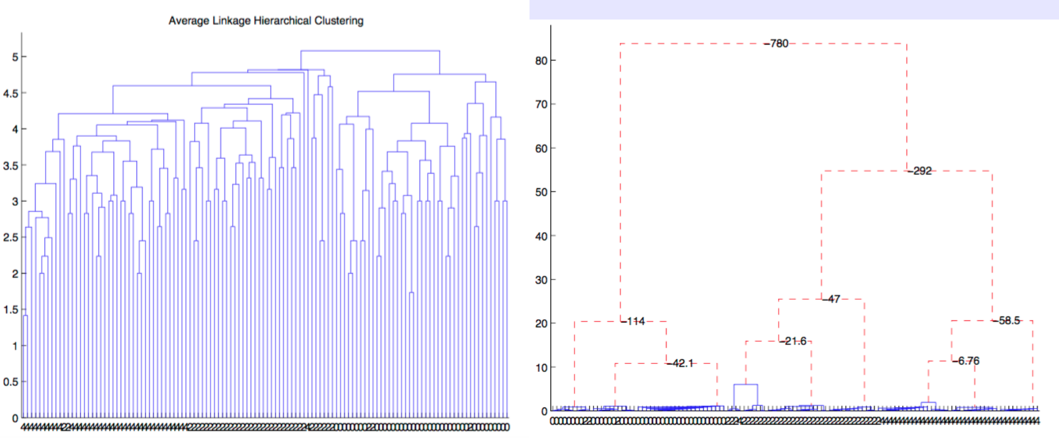
- Gibbs sampling in Dirichlet process mixture model in action:
  `https://www.youtube.com/watch?v=0Vh7qZY9sPs`

Dirichlet Process

- Slides giving more details on Dirichelt process mixture models:
  - `https://www.cs.ubc.ca/labs/lci/mlrg/slides/NP.pdf`
- We could alternately put a prior on number of clusters k:
  - Allows more flexibility than Dirichlet process as a prior.
  - Needs "trans-dimensional" MCMC to sample models of different sizes.
- There a variety of interesting variations on Dirichlet processes
  - Beta process ("Indian buffet process").
  - Hierarchical Dirichlet process.
  - Polya trees.
  - Infinite hidden Markov models.

Bayesian Hierarchical Clustering

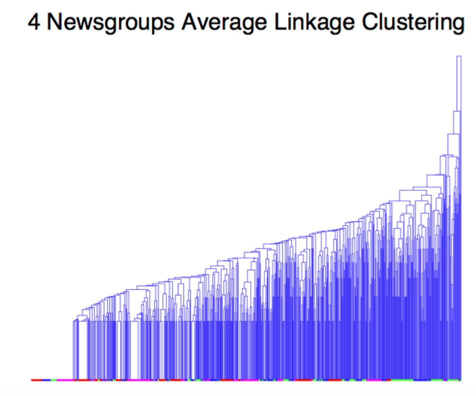
- Hierarchical clustering of $\{0, 2, 4\}$ digits using classic and Bayesian method:

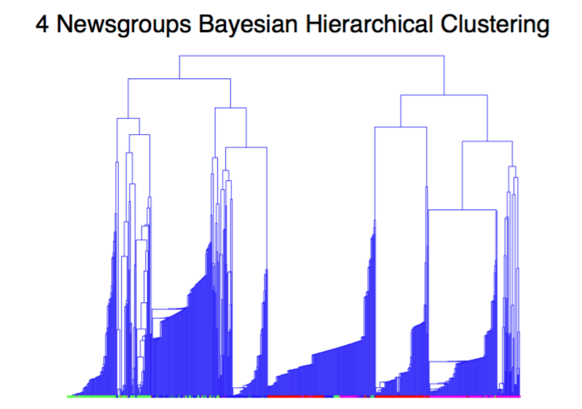


`http://www2.stat.duke.edu/~kheller/bhcnew.pdf` (y-axis represents distance between clusters)

Bayesian Hierarchical Clustering

- Hierarchical clustering of newgroups using classic and Bayesian method:





http://www2.stat.duke.edu/~kheller/bhcnew.pdf (y-axis represents distance between clusters)

Continuous Mixture Models

- We can also consider mixture models where z^i is continuous,

$$p(x^i) = \int_{z^i} p(z^i)p(x^i \mid z^i = c)dz^i.$$

- Unfortunately, computing the integral might be hard.

- Special case is if both probabilities are Gaussian (conjugate).
  - Leads to probabilistic PCA and factor analysis (OCEAN model in psychology).
  - My old material: `https://www.cs.ubc.ca/~schmidtm/Courses/540-W19/L17.5.pdf`.
- Another special case is scale mixtures of Gaussians
  - Where $p(x^i \mid z^i)$ is Gaussian and $p(z^i)$ is a gamma prior on variance (conjugate).
  - Can represent many distributions in this form, like Laplace and student t.
  - Leads to EM algorithms for fitting Laplace and student t..

Non-Parametric Mixtures: Kernel Density Estimation

- A common non-parametric mixture model centers one cluster on each example:

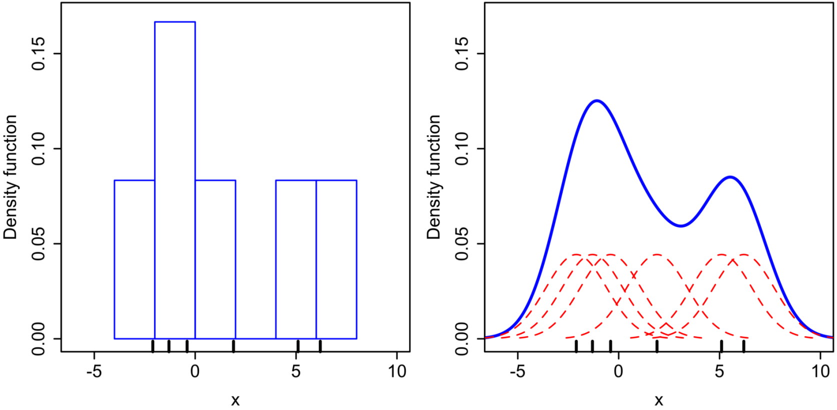
$$p(x^i) = \frac{1}{n}\sum_{j=1}^{n} p(x^i \mid x^j, \sigma^2 I).$$

- This is called kernel density estimation (KDE) or the Parzen window method.
  - Common choice is a Gaussian centered on each example ("mixture of n Gaussians").
  - Scale σ^2 is viewed as a hyper-parameter.

- By fixing mean/covariance/k, no parameters to learn (except σ^2).
  - And most inference tasks (except decoding) are easy but slow (depend on n).
  - Many variations exist, see bonus slides for generalizations.
    - Tends to work great in low dimensions and badly in high dimensions.

Histogram vs. Kernel Density Estimator

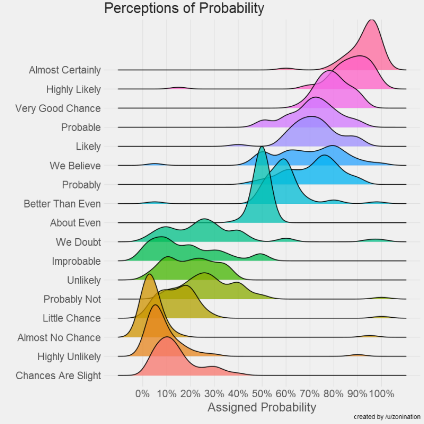
- I think of kernel density estimator as a continuous histogram:



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

Kernel Density Estimator for Visualization

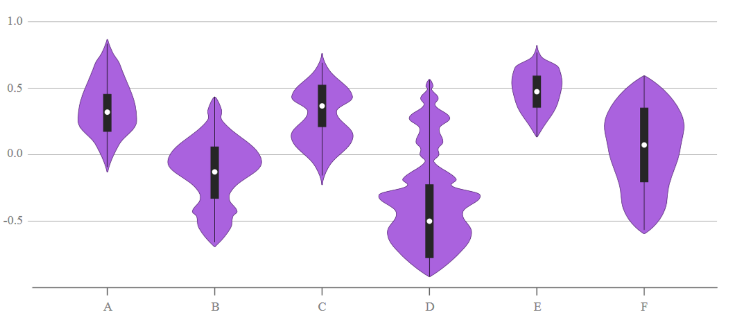
- Visualization of people's opinions about what "likely" and other words mean.



http://blog.revolutionanalytics.com/2017/08/probably-more-probably-than-probable.html

Violin Plot: Added KDE to a Boxplot

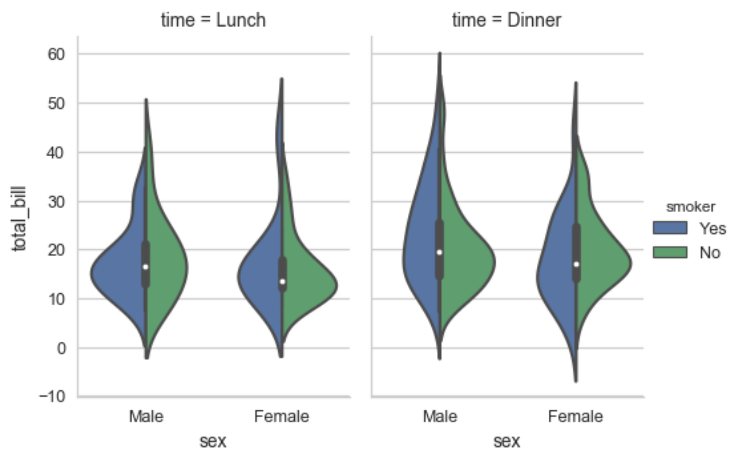
- Violin plot adds KDE to a boxplot:



https://datavizcatalogue.com/methods/violin_plot.html

Violin Plot: Added KDE to a Boxplot

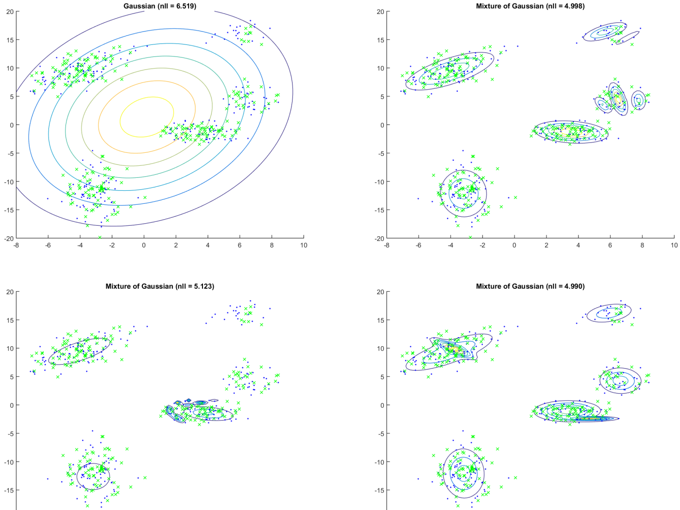
- Violin plot adds KDE to a boxplot:



https://seaborn.pydata.org/generated/seaborn.violinplot.html

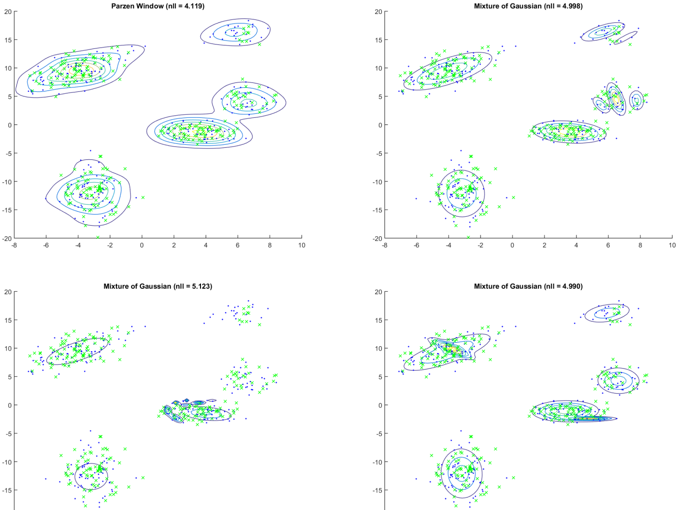
KDE vs. Mixture of Gaussian

- Multivariate vs mixture of Gaussians (different EM initializations):



KDE vs. Mixture of Gaussian

- Kernel density estimation vs mixture of Gaussians (different EM initializations):



Mean-Shift Clustering

- Mean-shift clustering uses KDE for clustering:
  - Define a KDE on the training examples, and then for test example $\hat{x}$:
    - Run gradient descent to maximize $p(x)$ starting from $\hat{x}$.
  - Clusters are points that reach same local minimum.
- https://spin.atomicobject.com/2015/05/26/mean-shift-clustering

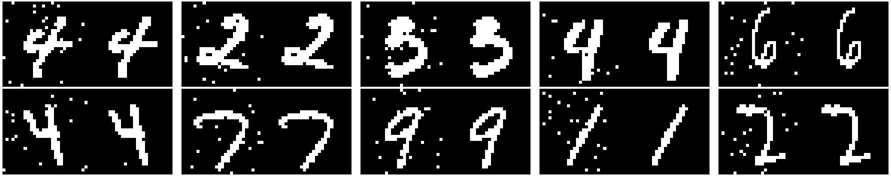
- Not sensitive to initialization, no need to choose k, can find non-convex clusters.

- Similar to density-based clustering from 340.
  - But doesn't require uniform density within cluster.
  - And can be used for vector quantization.

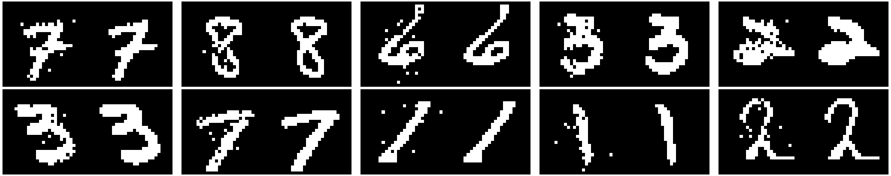
- "The 5 Clustering Algorithms Data Scientists Need to Know":
  - https://towardsdatascience.com/the-5-clustering-algorithms-data-scientists-need-to-know-a36d136ef68

Kernel Density Estimation on Digits

- Samples from a KDE model of digits:
  - Sample is on the left, right is the closest image from the training set.



- KDE just samples a training example then adds noise.
  - Usually makes more sense for continuous data that is densely packed.
- A variation with a location-specific variance (diagonal Σ instead of $\sigma^2 I$):



Summary

- Mixture of Bernoullis can model dependencies between discrete variables.
  - Can model arbitrary binary densities.
- Expectation maximization: algorithm for optimization with hidden variables.
  - Instead of imputation, works with "soft" assignments to nuisance variables.
  - Maximizes log-likelihood, weighted by all imputations of hidden variables.
  - Simple and intuitive updates for fitting mixtures models.
  - Appealing properties as an optimization algorithm, but only finds local optimum.
- Kernel density estimation: Non-parametric density estimation method.
  - Center a mixture on each datapoint (smooth variation on histograms).
  - Used for data visualization and low-dimensional density estimation.
  - Basis of mean-shift clustering.

- Next time: measuring defense in the NBA.

Digression: MLE does not exist

- For mixture of Gaussian, there is no MLE.

- You can make the likelihood arbitrarily large:
  - Set $\mu_c = x^i$ for a particular i and c, and make $\Sigma_c \to 0$.
  - It is common for optimizers to converge to models with degenerate clusters.
    - Empty or covariance is not positive definite.

- It is common to remove empty clusters and use a regularized update,

$$\Sigma_c = \frac{1}{\sum_{i=1}^n r_c^i} \sum_{i=1}^n r_c^i (x^i - \mu_c)(x^i - \mu_c)^T + \lambda I,$$

which corresponds to MAP estimation with an L1-regularizer on Θ diagonals.
  - The MAP estimate exists under this and other usual priors on Σ_c.

EM for MAP Estimation

- We can also use EM for MAP estimation. With a prior on Θ our objective is:

$$\underbrace{\log p(X \mid \Theta) + \log p(\Theta)}_{\text{what we optimize in MAP}} = \log\left(\sum_Z p(X, Z \mid \Theta)\right) + \log p(\Theta).$$

- EM iterations take the form of a regularized weighted "complete" NLL,

$$\Theta^{t+1} \in \underset{\Theta}{\operatorname{argmax}} \left\{ \underbrace{\sum_Z p(Z \mid X, \Theta^t) \log p(X, Z \mid \Theta)} + \log p(\Theta) \right\},$$

- Now guarantees monotonic improvement in MAP objective.
  - Has a closed-form solution for mixture of exponential families with conjugate priors.

- For mixture of Gaussians with $-\log p(\Theta_c) = \lambda \text{Tr}(\Theta_c)$ for precision matrices Θ_c:
  - Closed-form solution that satisfies positive-definite constraint (no $\log|\Theta|$ needed).

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,

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

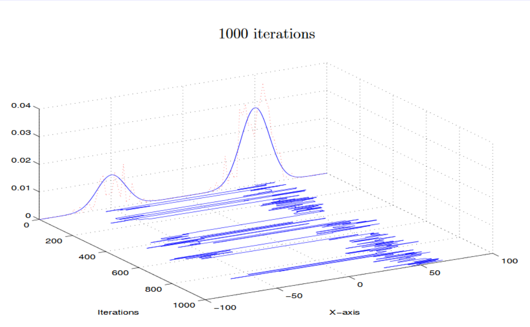
- Another variation is a mixture of discriminative models (like logistic regression),

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

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

Mixtures as Proposals in Metropolis-Hastings

- Suppose we want to sample from a multi-modal distribution:



http://www.cs.ubc.ca/~arnaud/stat535/slides10.pdf

- With random walk proposals, we stay in one mode for a long time.
- We could instead use mixture model as a proposal in Metropolis-Hastings.
  - Proposal could be a mixture between random walk and "mode jumping".

General Kernel Density Estimation

- The 1D kernel density estimation (KDE) model uses

$$p(x^i) = \frac{1}{n}\sum_{j=1}^{n} k_\sigma \underbrace{(x^i - x^j)}_{r},$$

where the PDF k is called the "kernel" and parameter σ is the "bandwidth".

- In the previous slide we used the (normalized) Gaussian kernel,

$$k_1(r) = \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{r^2}{2}\right), \quad k_\sigma(r) = \frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{r^2}{2\sigma^2}\right).$$

- Note that we can add a "bandwith" (standard deviation) σ to any PDF k_1, using

$$k_\sigma(r) = \frac{1}{\sigma}k_1\left(\frac{r}{\sigma}\right),$$

from the change of variables formula for probabilities ($|\frac{d}{dr}\left[\frac{r}{\sigma}\right]| = \frac{1}{\sigma}$).

- Under common choices of kernels, KDEs can model any continuous density.

Efficient Kernel Density Estimation

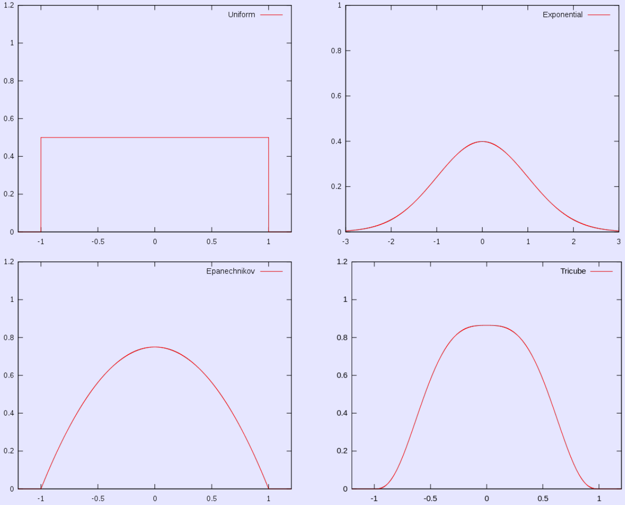
- KDE with the Gaussian kernel is slow at test time:
  - We need to compute distance of test point to every training point.
- A common alternative is the Epanechnikov kernel,

$$k_1(r) = \frac{3}{4}\left(1 - r^2\right)\mathcal{I}\left[|r| \leq 1\right].$$

- This kernel has two nice properties:
  - Epanechnikov showed that it is asymptotically optimal in terms of squared error.
  - It can be much faster to use since it only depends on nearby points.
    - You can use hashing to quickly find neighbours in training data.
- It is non-smooth at the boundaries but many smooth approximations exist.
  - Quartic, triweight, tricube, cosine, etc.
- For low-dimensional spaces, we can also use the fast multipole method.

Visualization of Common Kernel Functions

Histogram vs. Gaussian vs. Epanechnikov vs. tricube:



Multivariate Kernel Density Estimation

- The multivariate kernel density estimation (KDE) model uses

$$p(x^i) = \frac{1}{n}\sum_{j=1}^{n} k_A(\underbrace{x^i - x^j}_{r}),$$

- The most common kernel is a product of independent Gaussians,

$$k_I(r) = \frac{1}{(2\pi)^{\frac{d}{2}}}\exp\left(-\frac{\|r\|^2}{2}\right).$$

- We can add a bandwith matrix A to any kernel using

$$k_A(r) = \frac{1}{|A|}k_1(A^{-1}r) \qquad \text{(generalizes } k_\sigma(r) = \frac{1}{\sigma}k_1\left(\frac{r}{\sigma}\right)\text{)},$$

  and in Gaussian case we get a multivariate Gaussian with $\Sigma = AA^T$.

- To reduce number of parameters, we typically:
  - Use a product of independent distributions and use $A = \sigma I$ for some σ.