CPSC 440: Advanced Machine Learning

Kernel Density Estimation

Mark Schmidt

University of British Columbia

Winter 2021

Last Time: Bound on Progress of Expectation Maximization

- We have shown the following two bounds for EM:

$$\underbrace{\log p(O \mid \Theta)}_{\text{what we want to maximizie}} \quad \geq \quad \underbrace{Q(\Theta \mid \Theta^t)}_{\text{what EM maximizes}} \quad + \text{entropy}(\alpha)$$

$$\underbrace{\log p(O \mid \Theta^t)}_{\text{what we want to maximize}} \quad = \quad \underbrace{Q(\Theta^t \mid \Theta^t)}_{\text{what EM maximizes}} \quad + \text{entropy}(\alpha).$$

- Subtracting these and using $\Theta = \Theta^{t+1}$ gives a stronger result,

$$\log p(O \mid \Theta^{t+1}) - \log p(O \mid \Theta^t) \geq Q(\Theta^{t+1} \mid \Theta^t) - Q(\Theta^t \mid \Theta^t),$$

  that we improve objective by at least the decrease in Q.
  - This isn't enough for convergence, but EM converges under weak assumptions.
- Inequality holds for any choice of Θ^{t+1}.
  - Approximate M-steps are ok: we just need to decrease Q to improve likelihood.

- Unlike imputation that optimizes MAR values, considers all possible imputations.
  - MAR values "nuissance parameters": there might not be obvious "correct"

EM for MAP Estimation

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

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

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

$$\Theta^{t+1} \in \underset{\Theta}{\text{argmax}}\left\{\underbrace{\sum_H \alpha_H^t \log p(O, H \mid \Theta)}_{Q(\Theta \mid \Theta^t)} + \log p(\Theta)\right\},$$

- Now guarantees monotonic improvement in MAP objective.
  - This still has a closed-form solution for "conjugate" priors (defined later).

- 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).

Digression: Optimizing "Separable" Functions

- Consider an optimization problem of the form

$$\min_{w_1,w_2} f_1(w_1) + f_2(w_2).$$

- This is called a separable function.
  - The variable w_1 only affects the first term, and w_2 only affects second.

- With separable functions, you can optimize each term separately.
  - Gradient with respect to w_1 is: $\nabla f_1(w_1)$ (not affected by w_2).
  - Gradient with repsect to w_2 is: $\nabla f_2(w_2)$ (not affected by w_1).

- Similarly, if you have $\sum_{j=1}^{d} f_j(w_j)$, you optimize each f_j separately.
  - Use this property to simplify your assignment questions.

Digression: Optimizing "Separable" Functions

- Example: product of independent distributions:

$$p(x_1^i, x_2^i, \ldots, x_d^i \mid \Theta) = \prod_{j=1}^{d} p(x_j^i \mid \theta_j).$$

- To compute the MLE:

$$
\begin{aligned}
&\operatorname*{argmin}_{\Theta} -\log \prod_{i=1}^{n} p(x_1^i, x_2^i, \ldots, x_d^i \mid \Theta) && \text{(NLL for IID data)}\\
&\equiv \operatorname*{argmin}_{\Theta} -\sum_{i=1}^{n} \log p(x_1^i, x_2^i, \ldots, x_d^i \mid \Theta) && (\log(\alpha\beta) = \log(\alpha) + \log(\beta))\\
&\equiv \operatorname*{argmin}_{\Theta} -\sum_{i=1}^{n} \log \prod_{j=1}^{d} p(x_j^i \mid \Theta_j) && \text{(product of independent assumption)}\\
&\equiv \operatorname*{argmin}_{\Theta} -\sum_{i=1}^{n} \sum_{j=1}^{d} \log p(x_j^i \mid \Theta_j)) && (\log(\alpha\beta) = \log(\alpha) + \log(\beta))\\
&\equiv \operatorname*{argmin}_{\Theta} -\sum_{j=1}^{d} \sum_{i=1}^{n} \log p(x_j^i \mid \Theta_j)) && \text{(exchanging sums gives separable function: } f_j(\theta_j) = -\sum_{i=1}^{n} \log p(x_j^i \mid \Theta_j)\text{).}
\end{aligned}
$$

- Since the NLL is separable in the Θ_j, you can minimize each f_j separately.

Outline

A Non-Parametric Mixture Model

- The classic parametric mixture model has the form

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

- A natural way to define a non-parametric mixture model is

$$p(x^i) = \sum_{j=1}^{n} p(z^i = j)p(x^i \mid z^i = j),$$

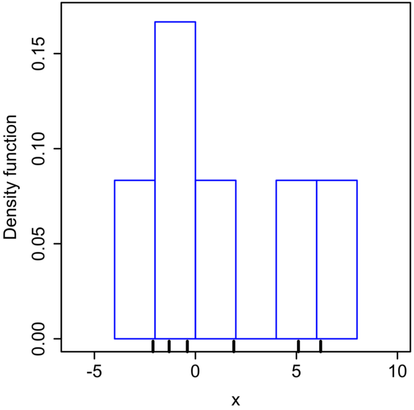
  where we have one mixture for every training example i.
- Common example: z^i is uniform and $x^i \mid z^i$ is Gaussian with mean x^j,

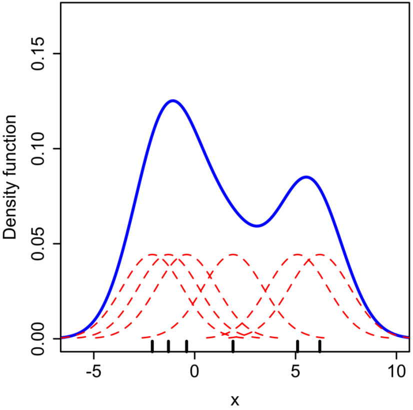
$$p(x^i) = \frac{1}{n}\sum_{j=1}^{n} \mathcal{N}(x^i \mid x^j, \sigma^2 I),$$

  and we use a shared covariance $\sigma^2 I$ (σ can be estimated with validation set).
- This is a special case of kernel density estimation (or Parzen window).

Histogram vs. Kernel Density Estimator

- Think of kernel density estimator as a generalization of a histogram:

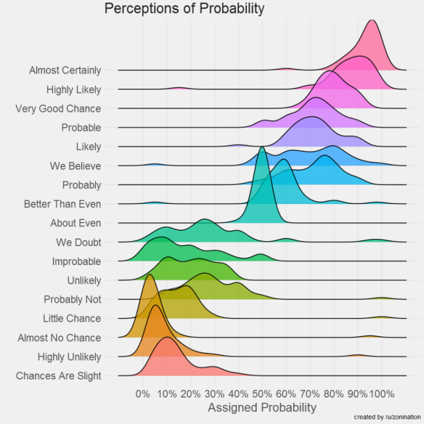




https://en.wikipedia.org/wiki/Kernel_density_estimation

Kernel Density Estimator for Visualization

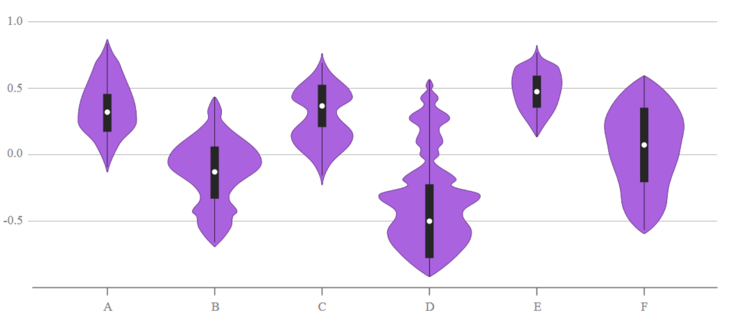
- Visualization of people's opinions about what "likely" and other words mean.



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

Violin Plot: Added KDE to a Boxplot

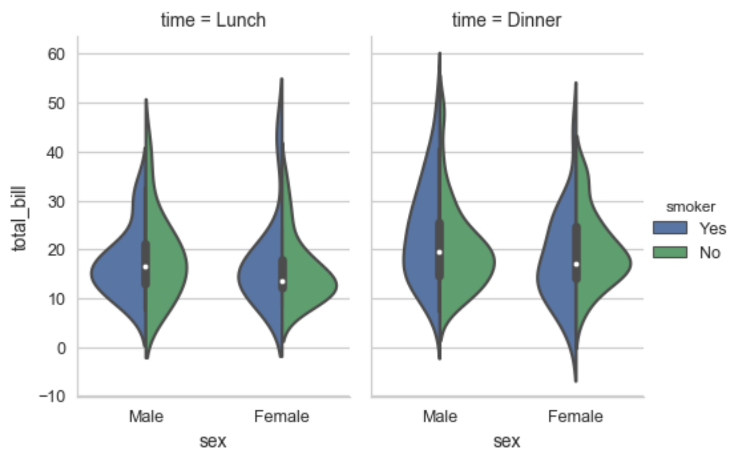
- Violin plot adds KDE to a boxplot:



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

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

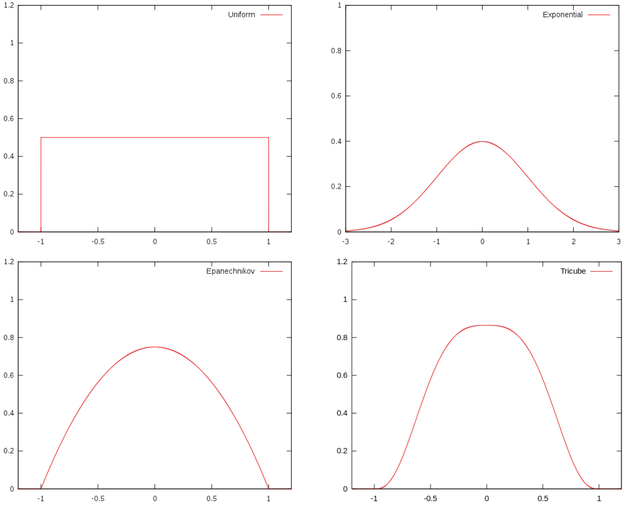
- KDE with the Gaussian kernel is slow at test time:
  - We need to compute distance of test point to every training point.
- A common alternative is the Epanechnikov kernel,

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

- 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

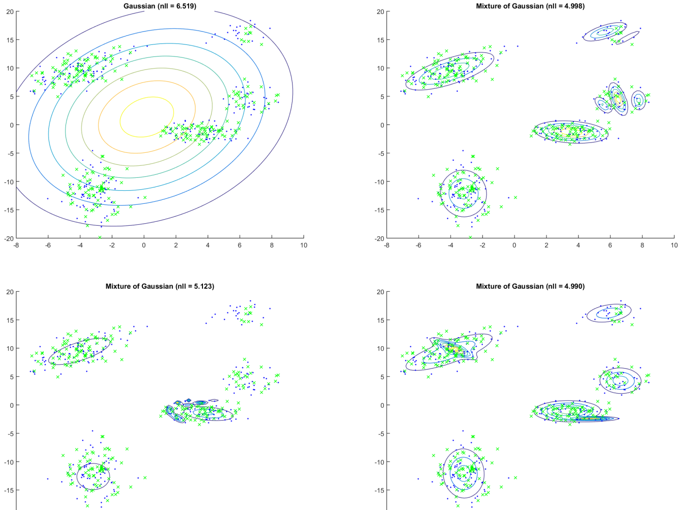
$$k_A(r) = \frac{1}{|A|}k_1(A^{-1}r) \qquad \text{(generalizes } k_\sigma(r) = \frac{1}{\sigma}k_1\left(\frac{r}{\sigma}\right)\text{)},$$

  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.

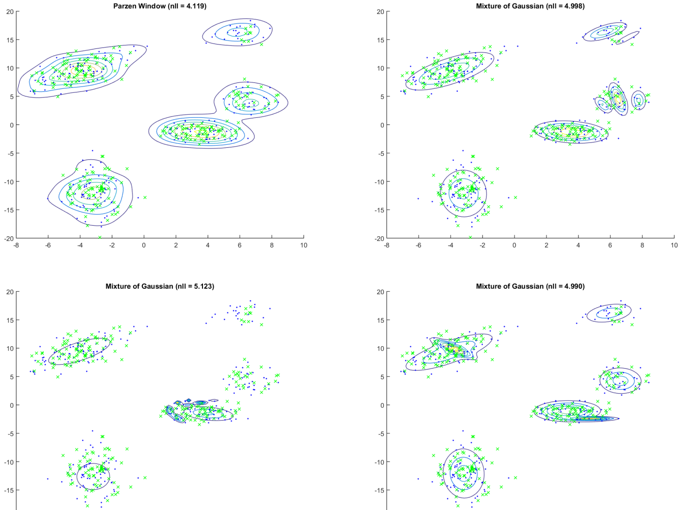
KDE vs. Mixture of Gaussian

- By fixing mean/covariance/k, we don't have to worry about local optima.



KDE vs. Mixture of Gaussian

- By fixing mean/covariance/k, we don't have to worry about local optima.

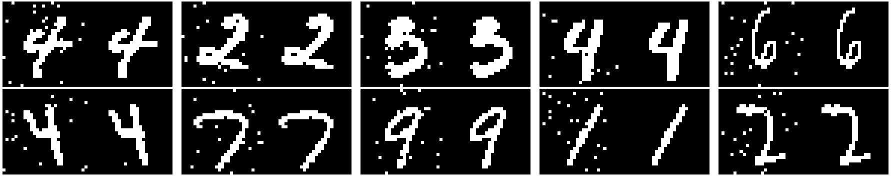


Mean-Shift Clustering

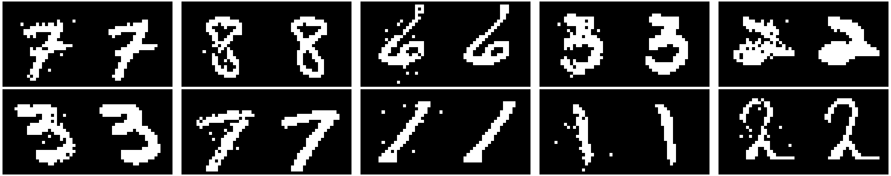
- Mean-shift clustering uses KDE for clustering:
  - Define a KDE on the training examples, and then for test example $\hat{x}$:
    - Run gradient descent to maximize $p(x)$ starting from $\hat{x}$.
  - Clusters are points that reach same local minimum.
- https://spin.atomicobject.com/2015/05/26/mean-shift-clustering
- 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 basically just adds independent noise to the training examples.
  - Usually makes more sense for continuous data that is densely packed.
- A variation with a location-specific variance (diagonal Σ instead of $\sigma^2 I$):



Continuous Mixture Models

- We've been discussing mixture models where z^i is discrete,

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

- We can also consider mixtures 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.
  - But if both probabilities are Gaussian then it's straightforward.

"Component Analysis" Methods

- Probabilistic PCA
  - A continuous mixture where z^i is Gaussian and $x^i \mid z^i$ is Gaussian.
  - Regular PCA is a special case, and so is "fitting a Gaussian to data".
  - Allows you to do things like "mixture of PCAs".
- Factor Analysis
  - Variant of probabilistic PCA with more-flexible covariance matrices.
  - Use in psychology for measuring things like intelligence and personality traits.
    - Like the OCEAN personality model.
  - In practice, performance is similar to PCA.
- Independent Component Analysis (ICA)
  - Variation on PCA where you assume noise is non-Gaussian.
  - Unlike PCA, this lets you identify "true factors".
  - Use in "blind source separation".
    - Record 5 people talking with 5 microphones, and separate sounds.
- I'm not covering these models this year, but you can see my material here: `https://www.cs.ubc.ca/~schmidtm/Courses/540-W19/L17.5.pdf`

End of Part: Basic Density Estimation and Mixture Models

- We discussed mixture models:
  - Write density as a convex combination of densities.
  - Examples include mixture of Gaussians and mixture of Bernoullis.
  - Can model multi-modal densities.

- Commonly-fit using expectation maximization.
  - Generic method for dealing with missing at random data.
  - Can be viewed as a “minimize upper bound” method.

- Kernel density estimation is a non-parametric mixture model.
  - Place on mixture component on each data point.
  - Nice for visualizing low-dimensional densities.

Summary

- Kernel density estimation: Non-parametric density estimation method.
  - Center a mixture on each datapoint.
  - Like a smooth variations on histograms.
  - Used for data visualization and low-dimensional density estimation.
  - Basis of mean-shift clustering.
- We also briefly mentioned "component/factor" analysis methods.
  - Probabilistic PCA, factor analysis, ICA.
- Next time: the sad truth about rain in Vancouver.

Scale Mixture Models

- Another weird mixture model is a scale mixture of Gaussians,

$$p(x^i) = \int_{\sigma^2} p(\sigma^2)\mathcal{N}(x^i \mid \mu, \sigma^2) d\sigma^2.$$

- Common choice for $p(\sigma^2)$ is a gamma distribution (which makes integral work):
  - Many distributions are special cases, like Laplace and student t.
- Leads to EM algorithms for fitting Laplace and student t.