
Density Estimation and Mixture Models

Statistical Machine Learning

Nic Schraudolph

Today's Topics:

Modeling:

- parametric vs. non-parametric models
- semi-parametric and mixture models

Density Estimation:

- non-parametric: histogram, Parzen window (kernel)
- mixture: Expectation-Maximisation (EM) algorithm
- related method: k-means clustering

Parametric Models

A **parametric** model implements a very restricted family of functions $\mathbf{f}(\mathbf{x}; \mathbf{w})$, leaving only a few parameters \mathbf{w} to be learned. It thus expresses a strong presupposition (= prior) about the structure of the data.

Example: parametric density estimation

Assume density is isotropic Gaussian: $f(\mathbf{x}_k; \mathbf{w}) = \mathcal{N}(\mathbf{x}_k; \boldsymbol{\mu}, \sigma^2 \mathbf{I})$
=> need only determine optimal mean $\boldsymbol{\mu}^*$ and variance σ^{2*}

ML quickly gives

$$\boldsymbol{\mu}^* = \frac{1}{n} \sum_k \mathbf{x}_k, \quad \sigma^{2*} = \frac{1}{n} \sum_k \|\mathbf{x}_k - \boldsymbol{\mu}\|^2.$$

Parametric Models

Advantages:

- ❑ only few parameters to fit => learning is easy
- ❑ model is very compact (low memory & CPU usage)
- ❑ excellent results **if** model appropriate (strong prior!)

Disadvantages:

- ❑ inflexible: fails if inappropriate model was chosen
- ❑ requires strong prior – can't express ignorance well
- ❑ appropriate model may be obscure, complicated, ...

Non-Parametric Models

Non-parametric models make only weak, general prior assumptions about the data, such as smoothness. $\mathbf{f}(\mathbf{x}; \mathbf{w})$ is constructed directly over the memorized training data \mathbf{X} ; the construction involves no or few parameters \mathbf{w} to be learned.

Example: k -nearest neighbor methods

The model's output $\mathbf{f}(\mathbf{x}; \mathbf{w})$ for some new datum \mathbf{x} is calculated by combining (in some fixed way) the memorized responses for the k nearest neighbors of \mathbf{x} in the training data. Example:
(regression) interpolate between nearest neighbor responses
(classification) take majority vote of nearest neighbor classes

Non-Parametric Models

Advantages:

- ❑ few or no parameters to fit => “learning” is easy
- ❑ very flexible: can fit (almost) any data well
- ❑ requires virtually no prior knowledge

Disadvantages:

- ❑ expensive in memory and CPU (must store all data)
- ❑ not much opportunity to incorporate prior knowledge

Will cover **Kernel methods** in depth later; today: simple example.

Non-Parametric Density Estimation

Probability that \mathbf{x} lies in region R : $P(\mathbf{x} \in R) = \int_R p(\mathbf{u}) d\mathbf{u}$

R sufficiently small $\Rightarrow p(\mathbf{u}) \simeq \text{const.}$

$\Rightarrow P(\mathbf{x} \in R) \simeq p(\mathbf{x}) V_R$ ($V_R = \text{volume of } R$)

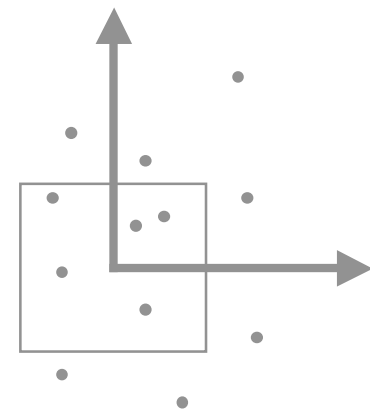
R sufficiently large $\Rightarrow k_R$ of n training points lie in R

$\Rightarrow P(\mathbf{x} \in R) \simeq k_R/n$

Putting the two together: $p(\mathbf{x}) \simeq k_R / (V_R n)$

This gives us two ways to determine $p(\mathbf{x})$:

1. **k -nearest neighbor**: vary V_R until $k_R = k$
2. determine k_R for given set of regions R ,
e.g. **histogram**: fixed (rectangular) grid



Parzen Window Density Estimation

R = **Parzen** window (or kernel), *e.g.* a Gaussian centered on the point \mathbf{x} for which we want to estimate $p(\mathbf{x})$

without loss of generality, let $V_R = \int_{\mathbf{u}} R(\mathbf{u}) d\mathbf{u} = 1$

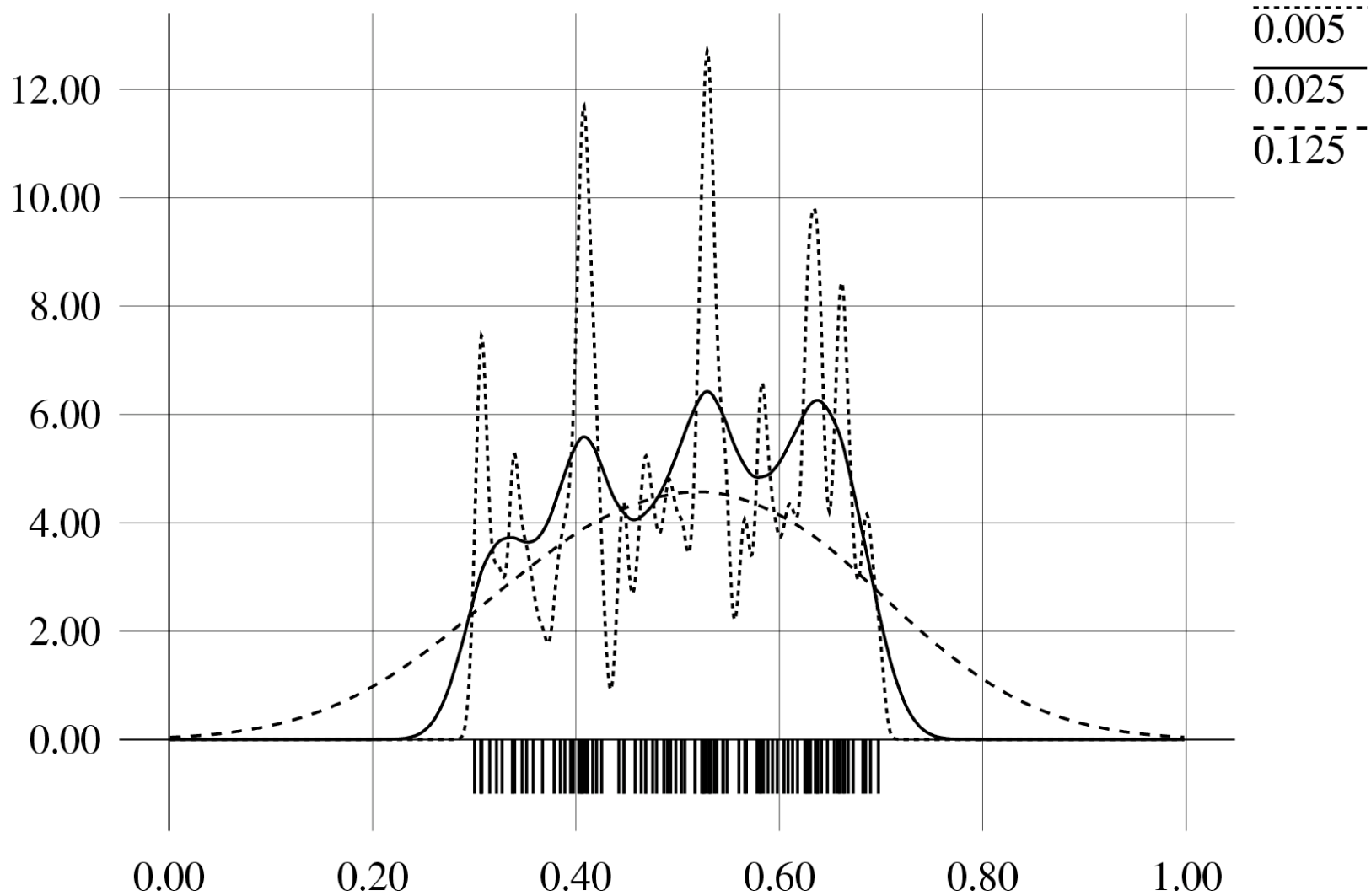
Then $p(\mathbf{x}) \simeq k_R/n = \frac{1}{n} \sum_k R(\mathbf{x} - \mathbf{x}_k)$,

i.e., $p(\mathbf{x})$ is estimated by **convolving** the data with the kernel R .

This is also known as **Kernel** density estimation.

The **width** of the kernel (= window) R determines the degree to which the data is smoothed; it can be set optimally by ML.

Influence of Kernel Width



Semi-Parametric Models

Parametric and non-parametric models are tools of classical statistics. Machine learning, by contrast, is often concerned with **semi-parametric** models (*e.g.*, neural networks).

Semi-parametric models are typically composed of multiple parametric components such that in the limit ($\# \text{components} \rightarrow \infty$) they are **universal approximators** capable of fitting any data.

Advantage: by controlling the number of components, we can pick a compromise between the efficiency of parametric and the flexibility of non-parametric methods.

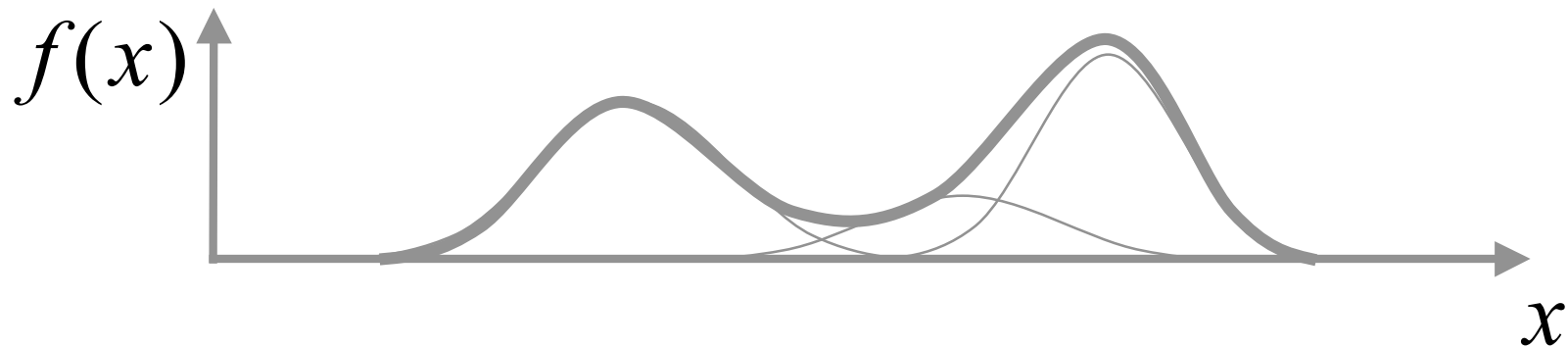
Disadvantage: they have many parameters to fit, which can make learning more difficult, and may require special algorithms. (That's why machine learning exists as a distinct discipline!)

Mixture Models

Mixture models are a major class of semi-parametric model; they output a **weighted sum** of their parametric mixture components. Their parameters comprise:

- the mixture coefficients, plus
- all the parameters of the individual components.

Example: mixture of Gaussians for density estimation

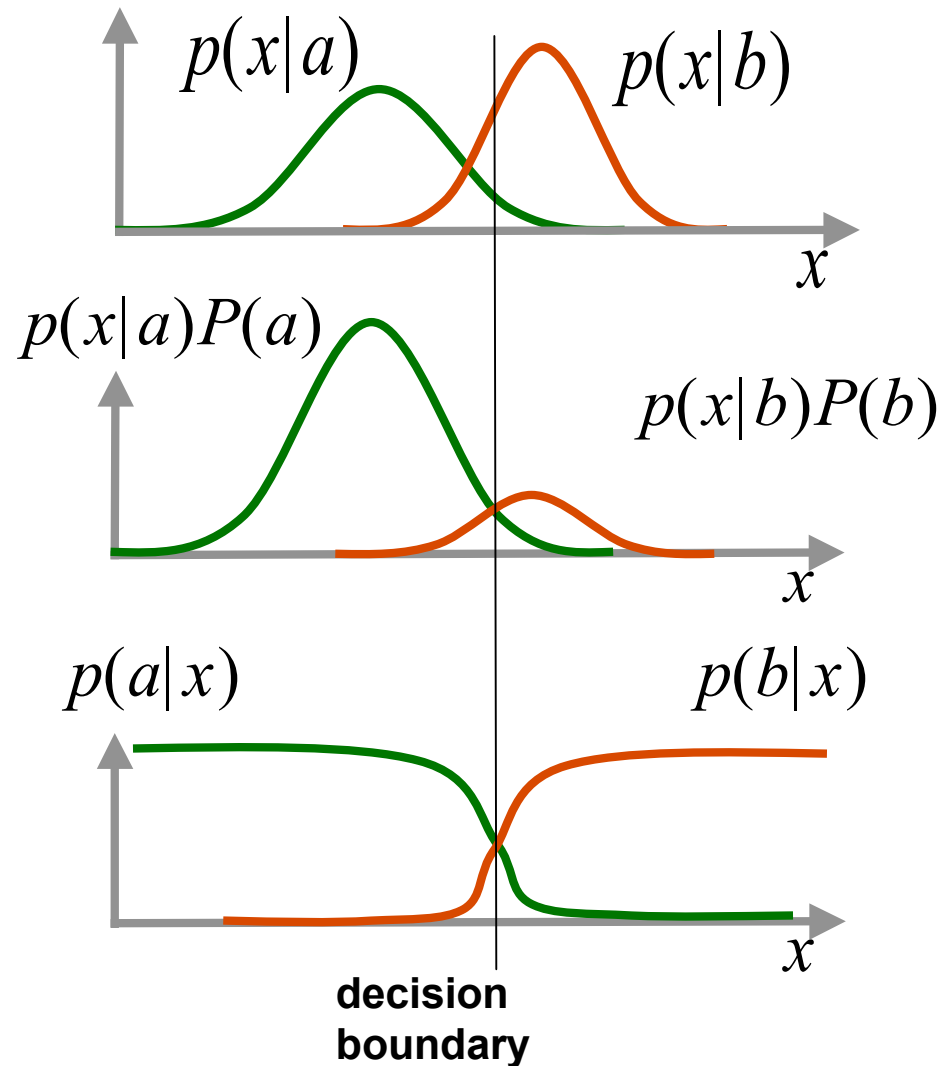


Fitting Mixture Models

The difficulty in learning a mixture model is knowing which mixture component(s) should be responsible for what data.

Imagine our data was nicely **clustered** into k groups, and we were told for each data point which cluster it belongs to. Fitting a mixture density of k Gaussians to that data would then be easy: just fit each cluster separately with a single Gaussian (we know how to do that!), then set each mixture coefficient to the relative size of the corresponding cluster. Note that we could then use a Bayesian classifier (using the mixing coefficients as priors) to assign **new** data points to their most likely cluster.

Chicken-and-Egg Problem



To assign data points to their clusters, we need to have each mixture component fitted to its cluster.

To fit each component to its cluster, we need to know which data points belong to which cluster.

For that, we need to have each mixture component fitted...

k-Means Clustering

To break the vicious circle, start with some initial parameters.

LOOP

- assign the data points to their most likely component
- fit each component to the data points assigned to it

UNTIL no more assignment changes

The **initialization** should ensure that

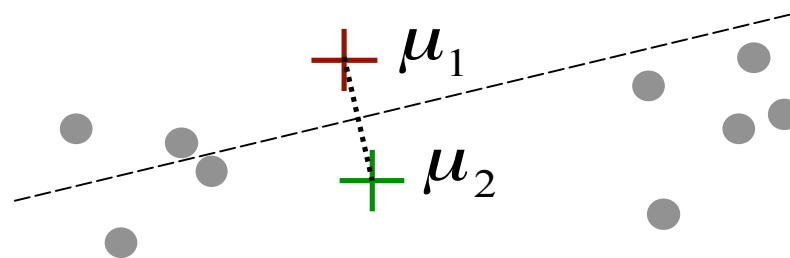
- the components reasonably cover the data, and
- are distinct from each other (**symmetry breaking!**).

For a mixture of Gaussians, a typical initialization would be:

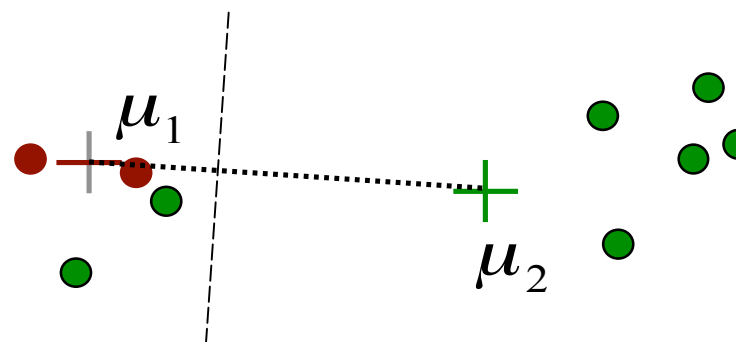
- all mixture coefficients equal,
- component variances equal to that of data: $(\forall j) \Sigma_j = \Sigma_X$,
- component means drawn from Gaussian: $\mu_j \sim N(\mu_X, \Sigma_X)$

Example: k-means Clustering

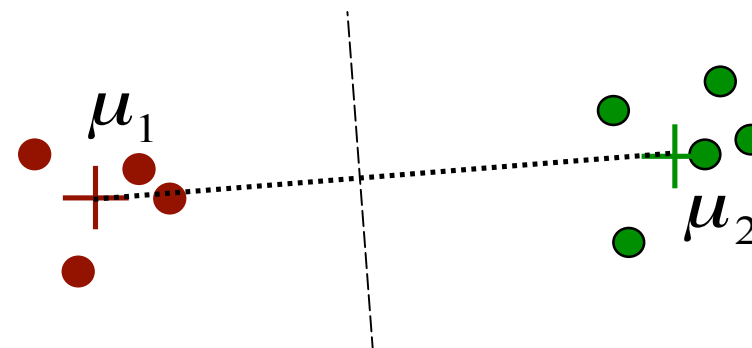
□ “arbitrary” initial state:



□ after one iteration:



□ final state after 2 iterations:
(no further changes)



Expectation-Maximization (EM)

We can do better still: instead of making a hard assignment of data points to mixture components, we can use Bayes' Rule to calculate for each component j and datum \mathbf{x}_i the **expectation** p_{ij} that j is responsible for \mathbf{x}_i (soft assignment):

$$p_{ij} = P(j|\mathbf{x}_i) = P(\mathbf{x}_i|j) P(j) / \sum_k P(\mathbf{x}_i|k) P(k) \quad (\mathbf{E}\text{-step})$$

We then ML-fit each component to all the data weighted by its responsibility for it, as indicated by the p_{ij} :

mixture coefficients: $P(j) = \frac{1}{n} \sum_i p_{ij}$ ($\mathbf{M}\text{-step}$)

component means: $\boldsymbol{\mu}_j = \sum_i p_{ij} \mathbf{x}_i / \sum_i p_{ij}$ for Gaussian mix

component variances: $\boldsymbol{\Sigma}_j = \sum_i p_{ij} (\mathbf{x}_i - \boldsymbol{\mu}_j)(\mathbf{x}_i - \boldsymbol{\mu}_j)^T / \sum_i p_{ij}$

and **loop** until convergence.

More on EM

EM is **the** learning algorithm for models where the M-step is simple, such as mixtures of Gaussians or hidden Markov models (HMMs). It can be used for both clustering (= unsupervised classification) and density estimation.

EM references:

1. A.P. Dempster, N.M. Laird, D.B. Rubin, Maximum-Likelihood from incomplete data via EM algorithm, In Journal Royal Statistical Society, Series B. Vol 39, 1977 (**the inventors**)
2. Jeff A. Bilmes, A Gentle Tutorial of the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models, TR-97-021, ICSI, U.C. Berkeley, CA, USA (on our web page; **read** at least pages 1-3)

Summary

Modeling:

- ❑ parametric vs. non-parametric models
- ❑ semi-parametric and mixture models

Density Estimation:

- ❑ non-parametric: histogram, Parzen window (kernel)
- ❑ mixture: Expectation-Maximisation (EM) algorithm
- ❑ related method: k-means clustering