9.913 Pattern Recognition for Vision
Class VI – Density Estimation
Yuri Ivanov
Road Map

Density Estimation

Parametric
- Max Likelihood
- Bayesian

Non-parametric
- Histograms
- Kernel Methods
- K-NN Method

Semi-parametric
- Mixture Density
Generative vs. Discriminative

There are two schools of thought in Machine Learning:

1. Generative:
   - Estimate class models from data
   - Compute the discriminant function
   - Plug in your data – get the answer

2. Discriminative:
   - Estimate the discriminant from data
   - Plug in your data – get the answer

*Last class*
Density Estimation

Density Estimation is at the core of generative Pattern Recognition

\[
P(a < x < b) = \int_a^b p(x) \, dx
\]

mean: \( E[x] = \int x p(x) \, dx \)

covariance: \( E[(x - E[x])(x - E[x])^T] = \int \left[ (x - E[x])(x - E[x])^T \right] p(x) \, dx \)

function mean: \( E[f(x)] = \int f(x) p(x) \, dx \)

conditional mean: \( E[y \mid x] = \int y p(y \mid x) \, dx \)
Refresher

Minimum expected risk:

\[ R^* = \int \min_\omega [R(\alpha \mid x)] p(x) \, dx \]

… is based on conditional risk:

\[ \omega_i = \arg \min_\omega R(\alpha \mid x) \]

… which is computed from the posterior:

\[ R(\alpha \mid x) = L(\alpha \mid \omega) P(\omega \mid x) \]

… which depends on the likelihood:

\[ P(\omega \mid x) = \frac{p(x \mid \omega)P(\omega)}{p(x)} \]
Setting

Data:

\[ D = \{ D_i \}_{i=1}^{c} \]

Assume that \( D_j \) contains no information about \( \omega_i \), \( \forall i \neq j \)

NOTATIONALLY - we abandon the class label:

\[ p(x | \omega_i) \quad \not\equiv \quad p(x) \]

Keep in mind: \[ p(x | \omega_i) \neq p(x) \]

Goal:
model the probability density function \( p(x) \), given a finite number of data points, \( x_1, x_2, \ldots, x_N \), drawn from it.
Three Methods

1. Parametric
   - Good: small number of parameters
   - Bad: choice of the parametric form

2. Non-parametric
   - Good: data “dictates” the approximator
   - Bad: large number of parameters

3. Semi-parametric
   - Good: combine the best of both worlds
   - Bad: harder to design
   - Good again: design can be subject to optimization
Parametric Density Estimation

Estimate the density from a given functional family

Given: \[ p(x | \theta) = f(x, \theta) \]
Find: \[ \theta \]

Two methods of parameter estimation:

1. **Maximum Likelihood** method
   - Parameters are viewed as unknown but fixed values

2. **Bayesian** method
   - Parameters are random variables that have their distributions
Normal (Gaussian) Density Function

A common assumption - *Gaussian*

\[ p(x \mid \theta) = \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} \exp \left( -\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) \right) \]

\( \theta = (\mu, \Sigma) \)

- Number of dimensions
- “Volume” of the covariance
- Squared Mahalanobis distance

\[ \mu = E[x] \]
- \( d \) parameters

\[ \Sigma = E\left[(x - \mu)^T (x - \mu)\right] \]
- \( d(d + 1)/2 \) parameters
Normal Density

\[ \mu = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \]
\[ \Sigma = \begin{bmatrix} 1 & 0 & .5 \\ 0 & 1 & .3 \\ .5 & .3 & 1 \end{bmatrix} \]

Constant density, \( (x - \mu)^T \Sigma^{-1} (x - \mu) = C \) - quadratic surface

\[ \Sigma \] - Positive semidefinite

\[ \downarrow \]

ellipsoid

Principal axes: eigenvectors of \( \Sigma \)

Length: \( \sqrt{\lambda_i}, \quad \lambda \) - eigenvalues of \( \Sigma \)
Whitening Transform

Define:

\[ \Lambda = \text{diag}\left(\text{eigval}\left(\Sigma\right)\right) \quad - \text{Scaling matrix} \]

\[ \Phi = \text{eigvec}\left(\Sigma\right) \quad - \text{Rotation matrix} \]

Then:

\[ W = \Lambda^{-1/2} \Phi^T \quad - \text{“Unscales” and “unrotates” the data} \]

For all:

\[ x \sim N(0, \Sigma) \]

\[ Wx \sim N(0, I) \]
Maximum Likelihood

Parameters are fixed but unknown.

\[ D \equiv \{ x^1, x^2, \ldots, x^N \} \quad \text{- a data set, drawn from } p(x) \]

Notationally, we make density explicitly dependent on parameters:

\[ p(x) \iff p(x \mid \theta) \]

Assuming that the data is drawn independently (i.i.d.):

\[
L(\theta) \equiv p(D \mid \theta) = \prod_{n=1}^{N} p(x^n \mid \theta) \quad \text{- a likelihood function}
\]

To find \( \theta \) Maximize \( L(\theta) \) w.r.t. parameters.
Maximum Likelihood

Maximizing \( L(\theta) \) is equivalent to maximizing log-likelihood function:

\[
l(\theta) \equiv \log L(\theta) = \log \prod_{n=1}^{N} p(x^n | \theta) = \sum_{n=1}^{N} \log p(x^n | \theta)
\]

To find \( \theta \) set the derivative to 0:

\[
\nabla_{\theta} l(\theta) = \sum_{n=1}^{N} \nabla_{\theta} \log p(x^n | \theta) = 0
\]

And solve for \( \theta \)
Quick Summary – ML Parameter Estimation

\[ P(\omega_i | x) = P(\omega_i | x, \theta_i) = \frac{p(x | \omega_i, \theta_i) P(\omega_i | \theta_i)}{p(x | \theta_i)} \]

We pick that

\[ \text{argmax}_\theta p(D | \theta) \]

\[ \prod_{n=1}^{N} p(x^n | \theta) \]
Solving a Maximum Likelihood Problem

Fixed covariance:

$$p(D | \theta)$$

$$\log p(D | \theta)$$

some candidates

\[ \hat{\theta} \]
Maximum Likelihood Example

In d-dimensions:

$$\nabla_\theta l(\theta) = \sum_n \nabla_\theta \left\{ -\frac{d}{2} \log [2\pi] - \frac{1}{2} \log [|\Sigma|] - \frac{1}{2} (x^n - \mu)^T \Sigma^{-1} (x^n - \mu) \right\}$$

Solving for the mean:

$$\nabla_\mu l(\theta) = -\frac{1}{2} \sum_n \Sigma^{-1} (x^n - \hat{\mu}) = 0 \implies$$

$$\hat{\mu} = \frac{1}{N} \sum_{n=1}^{N} x^n$$ - arithmetic average of samples
Maximum Likelihood Example (cont.)

\[ \nabla_\theta l(\theta) = \sum_n \nabla_\theta \left\{ -\frac{d}{2} \log [2\pi] - \frac{1}{2} \log [|\Sigma|] - \frac{1}{2} (x^n - \mu)^T \Sigma^{-1} (x^n - \mu) \right\} \]

Solving for the covariance:

For symmetric \( M \):
\[ \frac{d |M|}{dM} = |M|M^{-1} \quad \text{and} \quad \frac{d (a^T M^{-1} b)}{dM} = M^{-1} ab^T M^{-1} \quad \implies \]

\[ \nabla_\Sigma l(\theta) = -\frac{1}{2} \sum_n \left\{ \hat{\Sigma}^{-1} - \hat{\Sigma}^{-1} (x^n - \hat{\mu})(x^n - \hat{\mu})^T \hat{\Sigma}^{-1} \right\} = 0 \quad \implies \]

\[
\hat{\Sigma} = \frac{1}{N} \sum_{n=1}^{N} (x^n - \hat{\mu})(x^n - \hat{\mu})^T
\]

biased - arithmetic average of indiv. covariances

Fall 2004
Pattern Recognition for Vision
What if data comes one sample at a time?

\[
\hat{\mu}_N = \frac{1}{N} \sum_{n=1}^{N} x^n = \frac{1}{N} \left[ x^N + \sum_{n=1}^{N-1} x^n \right]
\]

\[
= \frac{1}{N} \left[ x^N + (N-1) \hat{\mu}_{N-1} \right] = \hat{\mu}_{N-1} + \frac{1}{N} \left[ x^N - \hat{\mu}_{N-1} \right]
\]

This estimate “stiffens” with more data (as it should).

One idea – fix the fraction. Then the estimate can track a non-stationary process.
Recursive ML

Fix the update rate and retrace the steps:

\[ v_N = v_{N-1} + \gamma \left[ x^N - v_{N-1} \right] = (1 - \gamma) v_{N-1} + \gamma x^N \]

\[ = (1 - \gamma)^2 v_{N-2} + (1 - \gamma) \gamma x^{N-1} + \gamma x^N \]

\[ = (1 - \gamma)^M v_{N-M} + \sum_{k=1}^{M} (1 - \gamma)^{M-k} \gamma x^k \]

Recency weights
Simple Example

Several images from a static camera:

How much noise is in it?

\[ x = \text{vec} (I_t - I_{t-1}) \]
\[ \mu = 0 \]
\[ \sigma = 1.2 \]

Now we can set a threshold that will statistically distinguish pixel noise from an object.
Problems with ML

We are given two estimates:

\[ \mu_1, \Sigma_1 \quad \mu_2, \Sigma_2 \]

Which one do we believe?

ML gives a single solution, regardless of uncertainty.
Density Estimation

Parametric
- Max Likelihood
- Bayesian

Non-parametric
- Histograms
- Kernel Methods
- K-NN Method

Semi-parametric
- Mixture Density
Bayesian Parameter Estimation

In classification our goal so far has been to estimate $P(\omega \mid x)$.

Let’s make the dependency on the data explicit:

$$P(\omega_i \mid x, D) = \frac{p(x \mid \omega_i, D)P(\omega_i \mid D)}{p(x \mid D)}$$

- $P(\omega_i \mid D)$ - this is easy to compute
- $P(x \mid D)$ - this is easy to compute by marginalization

What about $p(x \mid \omega_i, D)$?
Bayesian Parameter Estimation

This is a supervised problem so far:

\[ D = \{ D_1, D_2, \ldots, D_N \} \]

\[
p(x | \omega_i, D) = p\left( x | \omega_i, \{ D_j \}_{j=1}^{N} \right)
\]

\[
= p\left( x | \omega_i, D_i, \{ D_j \}_{j \neq i} \right) = p\left( x | \omega_i, D_i \right)
\]

\[
P(\omega_i | x, D) = \frac{p(x | \omega_i, D_i) P(\omega_i | D)}{p(x | D)}
\]
Bayesian Parameter Estimation

We will assume that we can obtain “labeled” data, so again:

Notationally: \[ p(x \mid \omega_i, D_i) \not\iff p(x \mid D) \]

Now our problem is to compute density for \( x \) given the data \( D \).

We assume the form of \( p(x) \) – the source density for \( D \):

\[ p(x) \iff p(x \mid \theta) \]

… and treat \( \theta \) as a random variable
Bayesian Parameter Estimation

Instead of choosing a value for a parameter, we use them all:

\[ p(x \mid D) = \int p(x, \theta \mid D) d\theta = \int p(x \mid \theta, D) p(\theta \mid D) d\theta \]

*Data predicts the new sample*  
*x is independent of D given \( \theta \)*

\[ = \int p(x \mid \theta) p(\theta \mid D) d\theta \]

*We chose the form of this*  
*What is this?*

Average densities \( p(x \mid \theta) \) for ALL possible values of \( \theta \) weighted by its posterior probability
Computing the posterior probability for \( \theta \):

Using Bayes rule:

\[
p(\theta \mid D) = \frac{p(D \mid \theta) p(\theta)}{\int p(D \mid \theta) p(\theta) d\theta}
\]

Using independence:

\[
p(D \mid \theta) = \prod_{n=1}^{N} p(x^n \mid \theta)
\]

Bayesian method does not commit to a particular value of \( \theta \), but uses the entire distribution.
Quick Summary – Bayesian Parameter Estimation

\[ P(\omega_i \mid x) = P(\omega_i \mid x, D) = \frac{p(x \mid \omega_i, D_i) P(\omega_i \mid D)}{p(x \mid D)} \]

**Easy**

**Hard**

\[ \int p(x \mid \theta) p(\theta \mid D) d\theta \]

we pick that

\[ p(D \mid \theta) p(\theta) \]

we “know” this*, **

\[ \prod_{n=1}^{N} p(x^n \mid \theta) \]

*Non-informative* prior – doesn’t introduce bias

**Conjugate prior** – causes \( p(\theta \mid D) \) have the same functional form as \( p(D \mid \theta) \)
Bayesian Parameter Estimation

For $\theta = \mu$:

Parameter prior

Parameter posterior

ML solution

Bayesian solution

posterior

weighted likelihoods

$\int p(x | \mu) p(\mu | D) \, d\mu$
Bayesian Parameter Estimation - Example

First let’s deal with the parameter:

Likelihood: \( p(x \mid \mu) = \mathcal{N}(\mu, \sigma^2) \) fixed

Parameter prior: \( p(\mu) = \mathcal{N}(\mu_0, \sigma_0^2) \)

Need to find: \( p(\mu \mid D) \)

Bayes rule again:

\[
p_N(\mu \mid D) = \frac{p(D \mid \mu)p(\mu)}{p(D)} = \alpha \left[ \prod_{n=1}^{N} p(x^n \mid \mu) \right] \mathcal{N}(\mu_0, \sigma_0^2) = \mathcal{N}(\mu_N, \sigma_N)
\]

\( N \)-sample parameter posterior

This is a Gaussian

Need these
Bayesian Parameter Estimation - Example

So, the posterior is a Gaussian

$$ p_N(\mu | D) = \mathcal{N}(\mu_N, \sigma_N) $$

After some algebra and identifying the terms:

$$ \frac{1}{\sigma_N^2} = \frac{1}{\sigma^2} N + \frac{1}{\sigma_0^2} \quad - \text{when Gaussians multiply – precisions add} $$

... and

$$ \mu_N = \frac{N\sigma_0^2}{N\sigma_0^2 + \sigma^2} \bar{x} + \frac{\sigma^2}{N\sigma_0^2 + \sigma^2} \mu_0 $$

With increasing $N$ covariance of the posterior decreases and the prior becomes unimportant.
Bayesian Parameter Estimation - Example

Now the integral:

\[
p(x \mid D) = \int p(x \mid \theta)p(\theta \mid D)d\theta
\]

\[
= \int N(\mu, \sigma^2)N(\mu_N, \sigma_N^2)d\mu = N(\mu_N, \sigma^2 + \sigma_N^2)
\]

You can show that it is also a Gaussian

Any guesses about why Gaussian is such a common assumption?
Recursive Bayes

For $N$-point likelihood:

$$p(D^N | \theta) = \prod_{n=1}^{N} p(x^n | \theta)$$

$$= p(x^N | \theta) \prod_{n=1}^{N-1} p(x^n | \theta) = p(x^N | \theta) p(D^{N-1} | \theta)$$

From this the recursive relation for the posterior:

$$p(\theta | D^N) = \frac{p(x^N | \theta) p(D^{N-1} | \theta) p(\theta)}{p(D^N)}$$

$$= \frac{p(x^N | \theta) p(\theta | D^{N-1})}{\int p(x^N | \theta) p(\theta | D^{N-1}) d\theta}$$
Recursive Bayes (cont.)

Again:

\[
p(\theta | D^N) = \frac{p(x^N | \theta) p(\theta | D^{N-1})}{\int p(x^N | \theta) p(\theta | D^{N-1}) \, d\theta}
\]

- 1-point update.

Setting \(N=1\):

\[
\frac{1}{\sigma_n^2} = \frac{1}{\sigma^2} + \frac{1}{\sigma_{n-1}^2}
\]

\[
\mu_n = \frac{\sigma_{n-1}^2}{\sigma_{n-1}^2 + \sigma^2} x + \frac{\sigma^2}{\sigma_{n-1}^2 + \sigma^2} \mu_{n-1}
\]
Recursive Bayes (cont.)

\[ p(\theta | D^N) \]

\[ p(x | \theta) \]

\( N = 10 \)
\( N = 2 \)
\( N = 1 \)
Problems with Bayesian Method

1. Integration is difficult
2. Analytic solutions are only available for restricted class of densities
3. Technicality: If the true $p(x|\theta)$ is NOT what we assume it is, the prior probability of any parameter setting is 0!
4. Integration is difficult
5. Did I mention that the integration is hard?
Relation between Bayesian and ML Inference

\[ p(\theta \mid D) \propto p(D \mid \theta) \cdot p(\theta) \]

peaks at \( \hat{\theta}_{ML} \)

\[
= \left[ \prod_{n} p(x^n \mid \theta) \right] \cdot p(\theta) = L(\theta) \cdot p(\theta)
\]

If the peak is sharp and \( p(\theta) \) is flat, then:

\[
p(x \mid D) = \int p(x \mid \theta) \cdot p(\theta \mid D) d\theta
\]

\[
= \int p(x \mid \hat{\theta}) \cdot p(\theta \mid D) d\theta = p(x \mid \hat{\theta}) \int p(\theta \mid D) d\theta = p(x \mid \hat{\theta})
\]

As \( N \to \infty \), \( p(x \mid D) \leftrightarrow p(x \mid \hat{\theta}) \)
Non-Parametric Methods for Density Estimation

Non-parametric methods do not assume any particular form for $p(x)$

1. Histograms
2. Kernel Methods
3. K-NN method
Density Estimation

Parametric
- Max Likelihood
- Bayesian

Non-parametric
- Histograms
- Kernel Methods
- K-NN Method

Semi-parametric
- Mixture Density
Histograms

\( \hat{P}(x) \) is a discrete approximation of \( p(x) \)

- Count a number of times that \( x \) lands in the \( i \)-th bin

\[
H(i) = \sum_{j=1}^{N} I(x \in R_i), \quad \forall i = 1, 2, ..., M
\]

- Normalize

\[
\hat{P}(i) = \frac{H(i)}{\sum_{j=1}^{M} H(j)}
\]
Histograms

How many bins?

- $M = 3$
  - "Oversmoothing"

- $M = 20$
  - "Overfitting"

- $M = 10$

- $M = 50$

Fall 2004

Pattern Recognition for Vision
Histograms

Good:
• Once it is constructed, the data can be discarded
• Quick and intuitive

Bad:
• Very sensitive to number of bins, $M$
• Estimated density is not smooth
• Poor generalization in higher dimensions
Aside: Curse of dimensionality (Bellman, ‘61):

- Imagine we build a histogram of a 1-d feature (say, *Hue*)
  - 10 bins
  - 1 bin = 10% of the input space
  - need at least 10 points to populate every bin

- We add another feature (say, *Saturation*)
  - 10 bins again
  - 1 bin = 1% of the input space
  - we need at least 100 points to populate every bin

- We add another feature (say, *Value*)
  - 10 bins again
  - 1 bin = 0.1% of the input space
  - we need at least 1000 points to populate every bin

\[
N = b^d
\]

- number of points grows exponentially
Aside: Curse Continues

Volume of a cube in $R^d$ with side $l$:

$$V_l = l^d$$

Volume of a cube with side $l-\varepsilon$:

$$V_\varepsilon = (l - \varepsilon)^d$$

Volume of the $\varepsilon$-shell:

$$\Delta = V_l - V_\varepsilon = l^d - (l - \varepsilon)^d$$

Ratio of the volume of the $\varepsilon$-shell to the volume of the cube:

$$\frac{\Delta}{V_l} = \frac{l^d - (l - \varepsilon)^d}{l^d} = 1 - \left(1 - \frac{\varepsilon}{l}\right)^d \rightarrow 1 \text{ as } d \rightarrow \infty \text{ !!!!!!}$$
Aside: Lessons of the curse

In generative models:
- Use as much data as you can get your hands on
- Reduce dimensionality as much as you can get away with

<End of Digression>
General Reasoning

By definition:

\[ P(x \in R) = P = \int_R p(x')\,dx' \]

If we have \( N \) i.i.d. points drawn from \( p(x) \):

\[ P(|x \in R| = k) = \frac{N!}{k!(N-k)!} P^k (1-P)^{N-k} = B(N, P) \]

- Num. of unique splits \( K \) vs. \((N-K)\)
- Prob that \( k \) of particular \( x \)-es are in \( R \)
- Prob that the rest are not

\( B(N, P) \) is a binomial distribution of \( k \)
General Reasoning (cont.)

Mean and variance of $B(N, P)$:

Mean: $\mu = E[k] = NP \implies P = E[k / N]$

Variance: $\sigma^2 = E[(k - \mu)^2] = NP(1 - P)$

$\implies E\left[\left(\frac{k}{N} - P\right)^2\right] = \left(\frac{\sigma}{N}\right)^2 = \frac{P(1 - P)}{N}$

That is:
- $E[k/N]$ is a good estimate of $P$
- $P$ is distributed around this estimate with vanishing variance

So:

$P \approx k / N$
So:

\[ P = \frac{k}{N} \]

On the other hand, under mild assumptions:

\[ P = \int_{R} p(x') dx' \approx p(x)V \]

Volume of \( R \)
(not \( p(x) \))

… which leads to:

\[ p(x) \approx \frac{k}{NV} \]
Now, given $N$ data points – how do we really estimate $p(x)$?

$$p(x) \approx \frac{k}{NV}$$

- Fix $k$ and vary $V$ until it encloses $k$ points
- Fix $V$ and count how many points ($k$) it encloses

**K-Nearest Neighbors (KNN)**

**Kernel methods**
Kernel Methods of Density Estimation

We choose $V$ by specifying a hypercube with a side $h$:

$$V = h^d$$

Mathematically:

$$H(y) = \begin{cases} 
1 & |y_j| < 1/2 \quad j = 1, \ldots, d \\
0 & \text{otherwise}
\end{cases}$$

kernel function:

$$H(y) \geq 0, \forall y \quad \text{and} \quad \int H(y)dy = 1$$
Parzen Windows

Then

\[ H \left( \left( x - x^n \right) / h \right) \] - a hypercube with side \( h \) centered at \( x^n \)

\( H \) can help count the points in a volume \( V \) around any \( x \):

\[ k(x) = \sum_{n=1}^{N} H \left( \frac{x - x^n}{h} \right) \]

\[ x_1 \]
\[ x_2 \]

\( h \)-neighborhood of \( x \)

No contribution to the count at \( x \)
Rectangular Kernel

So the number of points in h-neighborhood of $x$:

$$k(x) = \sum_{n=1}^{N} H \left( \frac{x - x^n}{h} \right)$$

… is easily converted to the density estimate:

$$\tilde{p}(x) = k(x) \frac{1}{NV} = \frac{1}{N} \sum_{n=1}^{N} \frac{1}{h^d} H \left( \frac{x - x^n}{h} \right)$$

Subtle point:

$$\int \left[ \frac{1}{N} \sum_{n=1}^{N} K \left( x, x^n \right) \right] dx = \frac{1}{N} \sum_{n=1}^{N} \left[ \int K \left( x, x^n \right) dx \right] = 1$$

$$\Rightarrow \int \tilde{p}(x) dx = 1$$
Example

Source

$h=1$

$h=2$

$h=4$
Smoothed Window Functions

The problem is as in histograms – it is discontinuous

We can choose a smoother function, s.t.:

\[ \tilde{p}(x) \geq 0, \quad \forall x \quad \text{and} \quad \int \tilde{p}(x)\,dx = 1 \]

Ensured by kernel conditions

Eg: <loud cheer> a (spherical) Gaussian:

\[ K(x, x^n) = \frac{1}{(\sqrt{2\pi h})^d} \exp \left( -\frac{\|x - x^n\|^2}{2h^2} \right) \]

… so:

\[ \tilde{p}(x) = \frac{1}{N} \sum_{n=1}^{N} \frac{1}{(\sqrt{2\pi h})^d} \exp \left( -\frac{\|x - x^n\|^2}{2h^2} \right) \]
Example

Source

- $h=0.6$
- $h=1.0$
- $h=1.3$
Some Insight

Interesting to look at expectation of the estimate with respect to all possible datasets:

\[
E \left[ \hat{p}(x) \right] = E \left[ \frac{1}{N} \sum_{n=1}^{N} K(x, x^n) \right] = E \left[ K(x, x') \right]
\]

\[
= \int K(x - x') p(x') dx'
\] - convolution with true density

That is:

\[
\hat{p}(x) = p(x) \quad \text{if} \quad K(x, x') = \delta(x, x')
\]

But not for the finite data set!
Conditions for Convergence

How small can we make $h$ for a given $N$?

\[ \lim_{N \to \infty} h_N^d = 0 \quad - \text{It should go to 0} \]

\[ \lim_{N \to \infty} N h_N^d = \infty \quad - \text{But slower than } 1/N \]

Based on the similar analysis of variance of estimates

Eg:

\[ h_N^d = h_1^d / \sqrt{N} \]

\[ h_N^d = h_1^d / \log(N) \]

Note that the choice of $h_1^d$ is still up to us.
Problems With Kernel Estimation

• Need to choose the width parameter, $h$
  
  • Can be chosen empirically
  • Can be adaptive, eg. $h_j = h d_{jk}$ – where $d_{jk}$ the distance from $x_j$ to $k$-th nearest neighbor

• Need to store all data to represent the density

• Leads to Mixture Density Estimation
K-Nearest Neighbors

Recall that:

\[ \tilde{p}(x) = \frac{k}{NV} \]

Now we fix \( k \) (typically \( k = \sqrt{N} \) ) and expand \( V \) to contain \( k \) points.

This is not a true density!

Eg.: choose \( N=1, k=1 \). Then:

\[ \tilde{p}(x) = \frac{1}{1 \cdot \|x - x_1\|} \quad \text{Oops!} \]

BUT it is useful for a number of theoretical and practical reasons.
K-NN Classification Rule

Let’s try classification with K-NN density estimate

Data: \[ N \] - total points
\[ N_j \] - points in class \( \omega_j \)

Need to find the class label for a query, \( x \)

Expand a sphere from \( x \) to include \( K \) points

\[ K \] - number of neighbors of \( x \)
\[ K_j \] - points of class \( \omega_j \) among \( K \)
KNN Classification

Then class priors are given by:  \[ p(\omega_j) = \frac{N_j}{N} \]

We can estimate conditional and marginal densities around any \( x \):
\[
p(x \mid \omega_j) = \frac{K_j}{N_j N V} \quad p(x) = \frac{K}{N V}
\]

By Bayes rule:
\[
p(\omega_j \mid x) = \frac{K_j}{N_j V} \frac{N_j}{N} \frac{N V}{K} = \frac{K_j}{K}
\]

Then for \textit{minimum error rate} classification:
\[
C = \arg \max_j K_j
\]
KNN Classification

Important theoretical result:

In the extreme case, $K=1$, it can be shown that:

\[
N\text{-sample error rate}
\]

for \( P = \lim_{N \to \infty} P_N(error) \)

\[
P^* \leq P \leq P^* \left( 2 - \frac{c}{c-1} P^* \right)
\]

That is, using just a single neighbor rule, the error rate is at most twice the Bayes error!!!
Problems with Non-parametric Methods

- Memory: need to store all data points
- Computation: need to compute distances to all data points every time
- Parameter choice: need to choose the smoothing parameter
Density Estimation

Parametric
- Max Likelihood
- Bayesian

Non-parametric
- Histograms
- Kernel Methods
- K-NN Method

Semi-parametric
- Mixture Density
Mixture Density Model

\[ p(x) = \sum_{j=1}^{M} p(x \mid j)P(j) \]

Uses MUCH less “kernels” than kernel methods
Kernels are parametric densities, subject to estimation
Example

$$p(x) = \sum_{j=1}^{M} p(x | j) P(j)$$
Using ML principle, the objective function is the *log-likelihood*:

\[
l(\theta) \equiv \log \prod_{n=1}^{N} p(x^n) = \sum_{n=1}^{N} \log \left\{ \sum_{j=1}^{M} p(x^n | j) P(j) \right\}
\]

Differentiate w.r.t. parameters:

\[
\nabla_{\theta_j} l(\theta) = \sum_{n=1}^{N} \frac{\partial}{\partial \theta_j} \log \left\{ \sum_{k=1}^{M} p(x^n | k) P(k) \right\}
\]

\[
= \sum_{n=1}^{N} \frac{1}{\sum_{k=1}^{M} p(x^n | k) P(k)} \frac{\partial}{\partial \theta_j} p(x^n | j) P(j)
\]
Mixture Density

Again let’s assume that \( p(x|\omega) \) is a Gaussian

We need to estimate \( M \) priors, and \( M \) sets of means and covariances

\[
\frac{\partial l(\theta)}{\partial \mu_j} = \sum_{n=1}^{N} P(j|x^n) \left[ \Sigma_j^{-1}(x^n - \hat{\mu}_j) \right]
\]

Setting it to 0 and solving for \( \mu_j \):

\[
\hat{\mu}_j = \frac{\sum_{n=1}^{N} P(j|x^n)x^n}{\sum_{n=1}^{N} P(j|x^n)}
\]

- convex sum of all data
Mixture Density

Similarly for the covariances:

\[ \frac{\partial l(\theta)}{\partial \sigma^2_j} = \sum_{n=1}^{N} P(j|\ x^n) \left[ \hat{S}_j^{-1} - \hat{S}_j^{-1} (x^n - \hat{\mu}_j)(x^n - \hat{\mu}_j)^T \hat{S}_j^{-1} \right] \]

Setting it to 0 and solving for \( \Sigma_i \):

\[ \hat{S}_j = \frac{\sum_{n=1}^{N} P(j|\ x^n)(x^n - \hat{\mu}_j)(x^n - \hat{\mu}_j)^T}{\sum_{n=1}^{N} P(j|\ x^n)} \]
Mixture Density

A little harder for $P(j)$ – optimization is subject to constraints:

$$\sum_{j=1}^{M} P(j) = 1 \quad \text{and} \quad P(j) \geq 0, \forall j$$

Here is a trick to enforce the constraints:

$$P(j) = \frac{\exp(\gamma_j)}{\sum_{k=1}^{M} \exp(\gamma_k)}$$

$$\frac{\partial P(i)}{\partial \gamma_j} = \delta(i - j)P(j) - P(i)P(j)$$
Mixture Density

Using the chain rule:

\[
\nabla_{\gamma_j} l(\theta) = \sum_{k=1}^{M} \frac{\partial l(\theta)}{\partial P(k)} \frac{\partial P(k)}{\partial \gamma_j} = \sum_{k=1}^{M} \sum_{n=1}^{N} \frac{p(x^n | k)}{P(x)} \left( \delta_{jk} P(j) - P(j)P(k) \right)
\]

\[
= \sum_{n=1}^{N} \left\{ \frac{p(x^n | j)}{P(x)} P(j) - \sum_{k=1}^{M} \frac{p(x^n | k)}{P(x)} P(j)P(k) \right\}
\]

\[
= \sum_{n=1}^{N} \left\{ P(j | x^n) - P(j) \sum_{k=1}^{M} p(k | x^n) \right\} = \sum_{n=1}^{N} \left\{ P(j | x^n) - P(j) \right\} = 0
\]

The last expression gives the value at the extremum:

\[
P(j) = \frac{1}{N} \sum_{n=1}^{N} P(j | x^n)
\]
Mixture Density

What’s the problem?

\[ P(j) = \frac{1}{N} \sum_{n=1}^{N} P(j \mid x^n) \]

\[ \hat{\mu}_j = \frac{\sum_{n=1}^{N} P(j \mid x^n)x^n}{\sum_{n=1}^{N} P(j \mid x^n)} \]

\[ \hat{S}_j = \frac{\sum_{n=1}^{N} P(j \mid x^n)(x^n - \hat{\mu}_j)(x^n - \hat{\mu}_j)^T}{\sum_{n=1}^{N} P(j \mid x^n)} \]

We can’t compute these directly!

Solution – EM algorithm. We will study it in Clustering.