Pattern Recognition 2: Probability Density Estimation

15-486/782: Artificial Neural Networks
David S. Touretzky

Fall 2006

Reading: Bishop, sections 2.1 - 2.5

Pattern Classification Again

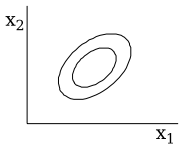
The goal: classify input pattern **x** by assigning it to the most probable class C_k.

In order to do this, we need to measure $p_k(\mathbf{x})$, the probability density of each class in the vicinity of the input pattern.

How do we estimate these probability densities?

Approaches to Density Estimation

- Parametric Models
  - Assume a distribution defined by some small number of parameters.
  - Gaussian distribution: μ, σ

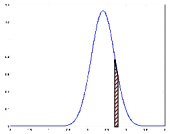


- Non-parametric Models
  - No assumptions about distribution. Use the training set directly to estimate density and classify points.
  - k-Nearest Neighbor classifiers
- Semi-parametric Models
  - Use very general functions for which you can vary the number of parameters (e.g., number of stored prototypes). Don't retain the training set.
  - Neural nets are semi-parametric models.

Parametric Model: Gaussian Probability Density

$$p(x) = \underbrace{\frac{1}{\sqrt{2\pi\sigma^2}}}_{\text{normalization term}} \cdot \exp\left[\frac{-(x-\mu)^2}{2\sigma^2}\right]$$

Function NORMPDF(x,mu,sigma) statistics toolbox:



```
mu = 0.4; sigma = 0.3;
dx = 0.05;  x = -2 : dx : 2;
plot(x, normpdf(x,mu,sigma))
```

normpdf(0.35,mu,sigma) = 1.31 Why is this > 1 ?

Integrate the pdf over [-2,2]:

sum(normpdf(x,mu,sigma)*dx) = 1.0

Multi-Dimensional Gaussian

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

μ is the mean vector: $\mu = E[x]$

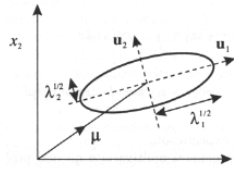
Σ is the covariance matrix:
$\Sigma = E\left[(x-\mu)(x-\mu)^T\right]$

$|\Sigma|$ is the determinant of Σ

2D Gaussian Distribution

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

μ is a mean vector $\langle\mu_1,\mu_2\rangle$
Σ is a 2x2 covariance matrix
$|\Sigma|$ is the determinant of Σ



$p(x)$ is governed by μ and Σ:
$\mu = E[x]$
$\Sigma = E\left[(x-\mu)(x-\mu)^T\right]$
Eigenvectors of Σ are the principal axes of the ellipse.
Eigenvalues are the variances along those directions.

5 parameters total: 2 for μ and 3 for Σ
(not 4, because Σ is symmetric)

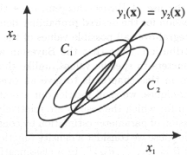
Mahalanobis Distance

Distance from the peak of the probability distribution.
This is the argument to the exp function on the previous page.

$$\Delta^2 = (x-\mu)^T\Sigma^{-1}(x-\mu)$$

Rescale the distance along each dimension based on covariance Σ.

So lines of constant probability are ellipsoids of constant Δ^2
If all gaussians have the same Σ, decision boundaries are linear.

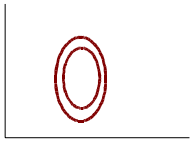


Simplifications

1) Assume no interaction between dimensions.
Then the covariance matrix is diagonal.

$$\Sigma = \begin{bmatrix} \sigma_1^2 & 0 & 0 & \dots \\ 0 & \sigma_2^2 & 0 & \dots \\ 0 & 0 & \sigma_3^2 & \dots \\ \vdots & \vdots & & \end{bmatrix}$$

Ellipse is aligned with the coordinate axes:



Diagonal Covariance Matrix (cont.)

The components of the input vector **x** are statistically independent.

We can simplify p(**x**) as a product of d independent univariate distributions:

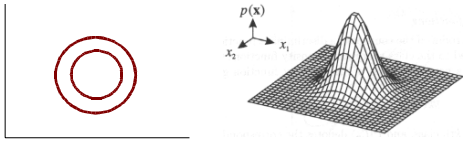
$$p(\boldsymbol{x}) = \prod_{i=1}^{d} p(x_i)$$

Simplifications

2) Assume no interaction, and equal variance σ^2 along all dimensions.

$$\Sigma = \begin{bmatrix} \sigma^2 & 0 & 0 & \dots \\ 0 & \sigma^2 & 0 & \dots \\ 0 & 0 & \sigma^2 & \dots \\ \vdots & \vdots & & \end{bmatrix}$$

This distribution is circular (or a hypersphere).



Finding Optimal Parameters θ by Maximum Likelihood

$$p(\boldsymbol{X}|\theta) = \prod_{n=1}^{N} p(x^n|\theta) \equiv L(\theta)$$

Find the θ that maximizes the likelihood $L(\theta)$
More convenient to minimize negative log likelihood:

$$E = -\ln L(\theta) = -\sum_{n=1}^{N} \ln p(x^n|\theta)$$

Differentiate E to find optimum θ.
For a Gaussian, $\theta = \langle \mu, \Sigma \rangle$, and:

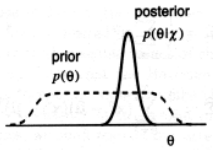
$$\hat{\mu} = \frac{1}{N}\sum_{n=1}^{N} x^n \qquad \hat{\Sigma} = \frac{1}{N}\sum_{n=1}^{N} (x^n - \hat{\mu})(x^n - \hat{\mu})^T$$

Finding Optimal Parameters by Bayesian Inference

Instead of trying to find a single best value for θ, consider the distribution over possible values.

Assume a prior p(θ), in the absence of data.

Then adjust the distribution after seeing the data, giving a posterior p(θ | X).



Bayesian Inference

$$p(x|\boldsymbol{X}) = \int p(x,\theta|\boldsymbol{X})\ d\theta$$

$$p(x,\theta|\boldsymbol{X}) = p(x|\theta,\boldsymbol{X})\,p(\theta|\boldsymbol{X})$$

Independent of $\boldsymbol{X}$

$$p(x|\boldsymbol{X}) = \int p(x|\theta)\,p(\theta|\boldsymbol{X})\ d\theta$$

Performs a weighted average over possible values for θ.

Bayesian Inference

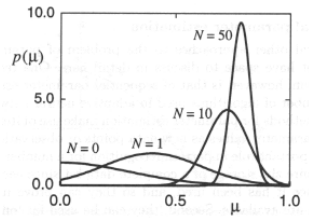
$$p(\boldsymbol{X}|\theta) = \prod_{n=1}^{N} p(x^n|\theta)$$

$$p(\theta|\boldsymbol{X}) = \frac{p(\boldsymbol{X}|\theta)\,p(\theta)}{p(\boldsymbol{X})} = \frac{p(\theta)}{p(\boldsymbol{X})}\prod_{n=1}^{N} p(x^n|\theta)$$

$$where\ p(\boldsymbol{X}) = \int p(\theta')\prod_{n=1}^{N} p(x^n|\theta')\ d\theta'$$

Bayesian Inference Example

Learning a normal distribution.
The prior on μ has zero as the most likely value.
The distribution shifts and tightens as N increases.



Sequential Parameter Estimation: On-Line Learning

Useful for "adaptive systems" that tune their parameters with experience.

Requires only a little storage.

Example: learning the mean of a Gaussian:

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

Non-Parametric Methods

What if the data doesn't fit a Gaussian distribution?

What if it doesn't fit any tractable distribution?

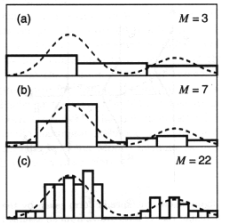
Instead of fitting distribution parameters, we can estimate probability density in each region of feature space directly from the training data.

This is the **non-parametric** approach.

Histograms as Density Estimators

• Divide the feature space into bins and measure the density of points in each class C_k in each bin.

• How many bins should we use?
  – M = 3: fails to capture bimodality
  – M = 7: pretty rough estimate
  – M = 22: bins are "spikier" than the real distribution due to small sample size



• Where should the bins be placed?
  – Source of brittleness.

• Curse of dimensionality: too many bins.

Density Estimation Within Region R

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

Assume N points drawn independently from p(x). The probability that K of them fall within R is:

$$Pr(K) = \frac{N!}{K!(N-K)!} \cdot P^K (1-P)^{N-K}$$

mean number within R: $E[K/N] = P$

variance: $E[(K/N - P)^2] = P(1-P)/N$

Variance drops, distribution sharply peaked as $N \to \infty$

Density Estimation Within Region R

$$P \approx K/N$$

If p(x) doesn't vary much over R, we can estimate

$$P = \int_R p(x')\, dx' \approx p(x)\, V$$

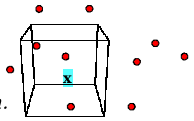
where V is the volume of R, and x is any point in R. So

$$p(x) \approx \frac{P}{V} = \frac{K}{N \cdot V}$$

R should be large so we get a good sample.
R should be small so we don't over-smooth the estimate.
There is a tradeoff in the choice of R.

Fixed R: Kernel-Based Methods (Parzen Windows)`

Kernel function $H(\boldsymbol{u}) = \begin{cases} 1 & \text{if } |u_j| \leq 1/2, \quad j = 1 \ldots d \\ 0 & \text{otherwise} \end{cases}$



$H(\boldsymbol{u})$ *is a unit hypercube centered at the origin.*

$H\left(\frac{x - x^n}{h}\right)$ *is a hypercube w/side h, volume* $V = h^d$, *centered on* $\boldsymbol{x}$.

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

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

Smoothing the Density Estimate: Parzen Windows w/Gaussian Kernels

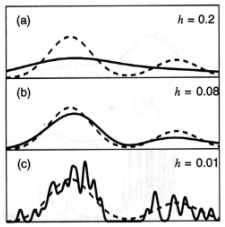
Kernels must satisfy $H(\boldsymbol{u}) \geq 0$ *and* $\int H(\boldsymbol{u})\, d\boldsymbol{u} = 1$

We can use a Gaussian kernel in place of a boolean hypercube.

$$\tilde{p}(x) = \frac{1}{N} \sum_{n=1}^{N} \frac{1}{(2\pi h^2)^{d/2}} \cdot \exp\left(-\frac{\|x - x^n\|^2}{2h^2}\right)$$

Varying h changes the amount of smoothing of the density estimate.

Training set contains 30 points drawn from two normal distributions.



Expectation of the Density Depends On the Kernel Function H

$$\begin{aligned} E[\tilde{p}(\boldsymbol{x})] &= \frac{1}{N} \sum_{n=1}^{N} E\left[\frac{1}{h^d} H\left(\frac{\boldsymbol{x} - \boldsymbol{x}'}{h}\right)\right] \\ &= E\left[\frac{1}{h^d} H\left(\frac{\boldsymbol{x} - \boldsymbol{x}'}{h}\right)\right] \\ &= \int \frac{1}{h^d} H\left(\frac{\boldsymbol{x} - \boldsymbol{x}'}{h}\right) p(\boldsymbol{x}')\, d\boldsymbol{x}' \end{aligned}$$

Expected density $\tilde{p}(\boldsymbol{x})$ is equal to the true density $p(\boldsymbol{x})$ convolved with the kernel H.

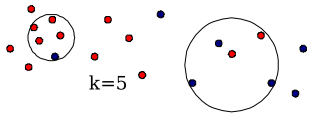
As $h \to \infty$, H approaches a delta function and $\tilde{p}(\boldsymbol{x})$ approaches $p(\boldsymbol{x})$.

Variable Size Volume V

Parzen windows use a fixed kernel size $V = h^d$; the number of points falling within the window varies.

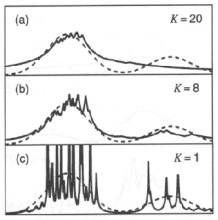
Alternatively, we can choose a fixed number of points k, and scale the window until it's just big enough to admit k points.

This is called k-Nearest Neighbors.



k-NN Density Estimation

- Find the K closest points and divide by the volume to estimate density.
- High K: too much smoothing.
- Low K: too noisy.



Training set: 30 points drawn from two binomial distributions.

Deriving k-Nearest Neighbor Using Bayes' Rule

Choose K nearest points; assume K_k are in class C_k.

$$p(x|C_k) = \frac{K_k}{N_k V} \quad \textit{Class-conditional density}$$

$$p(x) = \frac{K}{NV} \quad \textit{Unconditional density}$$

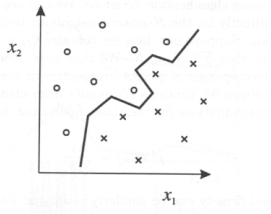
$$P(C_k) = \frac{N_k}{N} \quad \textit{Class priors}$$

$$P(C_k|x) = \frac{p(x|C_k)P(C_k)}{p(x)} = \frac{K_k}{K}$$

So, to classify x with minimum error, let the neighbors vote and follow the majority.

The Nearest Neighbor Rule

- For K=1 we have a very simple classifier:
  - Place **x** in the class of its closest neighbor.
- Decision boundary is piecewise linear.
- Easy to implement, but vulnerable to noise.



Smoothing Parameters for Density Estimation Models

- Histograms: bin width
- Kernel methods: kernel width h
- k-nearest neighbor: value of K

- Over-smoothing means bias is too strong.
- Insufficient smoothing means variance is too high.
- Choosing the right parameter value optimizes the bias/variance tradeoff.

High Dimensions Are Strange

Let S^n= unit hypersphere in n dimensions.
Let C^n= unit cube (sides of length 2) in n dims.

$$\frac{Volume(S^2)}{Volume(C^2)} = \frac{\pi}{4} \approx 0.785$$

Theorem (Bishop excercise 1.4):

$$\lim_{n\to\infty} \frac{Volume(S^n)}{Volume(C^n)} = 0$$

Also, in an n-dim. Gaussian distribution, most of the probability mass is in a thin shell at large radius: a hollow sphere.

Almost no points are at the origin (mean value).
Not what you'd expect from the 1-dimensional case.

Kullback-Leibler Distance Between Two Distributions

$$\begin{aligned} E[-\ln L] &= -\lim_{N\to\infty} \frac{1}{N} \sum_{n=1}^{N} \ln \tilde{p}(\boldsymbol{x}^n) \\ &= -\int p(\boldsymbol{x}) \ln \tilde{p}(\boldsymbol{x})\, d\boldsymbol{x} \end{aligned}$$

When $\tilde{p}(\boldsymbol{x})=p(\boldsymbol{x})$, residual value is the entropy:

$$\int p(\boldsymbol{x}) \ln p(\boldsymbol{x})\, d\boldsymbol{x}$$

Subtract off the residual to give a measure of distance between $p(\boldsymbol{x})$ and $\tilde{p}(\boldsymbol{x})$:

$$L = -\int p(\boldsymbol{x}) \ln \frac{\tilde{p}(\boldsymbol{x})}{p(\boldsymbol{x})}\, d\boldsymbol{x}$$

Measure is asymmetric: should focus on divergence in regions where real distribution lies.

Kohonen's LVQ (Learning Vector Quantizer)

Nearest-neighbor classifier (k=1), but uses a small number of prototypes instead of storing all the training data. Neural net learns the prototypes.

This is a semi-parametric approach.

Let ξ be an input pattern.

Winner $i^* = \underset{i}{argmax}(\bar{w}_i \cdot \xi)$

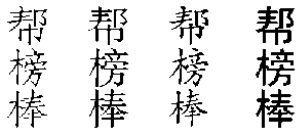
Move winner's weight vector closer to (further from) ξ if winner was correct (incorrect).

$$\Delta \bar{w}_{i^*} = \begin{cases} +\eta(\xi - \bar{w}_{i^*}) & \text{if correct} \\ -\eta(\xi - \bar{w}_{i^*}) & \text{if incorrect} \end{cases}$$

Example Application: Chinese OCR

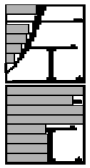
"Simplified" character set has 7000 symbols.

Four main fonts are used: Fangsongti, Heiti, Kaiti, and Songti.

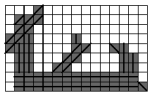


405-Dimensional Feature Set

Name	Dimensions
Blackness (number of black pixels)	1
Stroke Width	1
Total Stroke Length	1
Horizontal Projection	64
Vertical Projection	64
Number of Horizontal Transitions (white-to-black)	1
Number of Vertical Transitions (white-to-black)	1
First Order Peripheral Features	32
Second Order Peripheral Features	32
Stroke Density at 0° and 90°	16
Local Direction Contributivity with four regions and four directions	64
Maximum Local Direction Contributivity	64
Stroke Proportion	32
Black Jump Distribution in Balanced Subvectors	32
Total Feature Dimensions	405



First and second order peripheral features



Stroke Proportion

Features from Suchenwirth et al. (1989)

Training Strategy

Romero, Berger, Thibadeau, and Touretzky (1995):

1) Collect 20,000 characters of training data.

2) Calculate 405 feature values for each character.

3) Extract the top 100 principal components. K-L transform minimizes within-class variance and maximizes inter-class variance.

4) Train neural net classifier using a variant of the LVQ algorithm.

Results

Started with 6992 prototypes.

The system added 40 prototypes during training.

Tested on 40,000 scanned characters.

97.611% correct recognition rate on test set.