Pattern Recognition 2:
Probability Density Estimation

15-496/782: Artificial Neural Networks
David S. Touretzky

Spring 2004

Reading: Bishop, sections 2.1 - 2.5

1

Pattern Classification Again

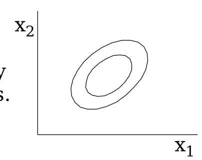
The goal: classify input pattern **x** by assigning it to the most probable class C_k.

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?

2

Approaches to Density Estimation



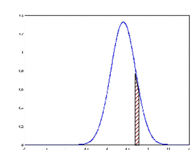
- Parametric Models
  - Assume a distribution defined by some small number of parameters.
  - Gaussian distribution: μ σ
- Non-parametric Models
  - No assumptions about distribution. Use the training set directly to estimate density and classify points.
  - k-Nearest Neighbor classifiers
- Semi-parametric Models
  - Create prototypes, or train feature detectors that are very general functions. Don't retain the training set.
  - Neural nets are semi-parametric models.

3

Parametric Model:
Gaussian Probability Density

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

Function NORMPDF(x,mu,sigma) from Matlab 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]:

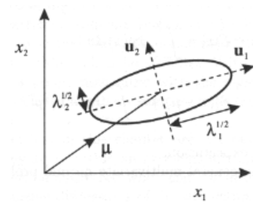
sum(normpdf(x,mu,sigma)*dx) = 1.0

4

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.

5 parameters total: 2 for μ and 3 for Σ
(because Σ is symmetric)

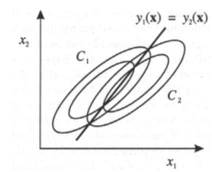
Mahalanobis Distance

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.

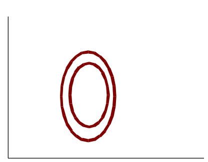


Simplifications

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

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

Ellipse is aligned with the coordinate axes:

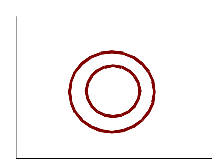


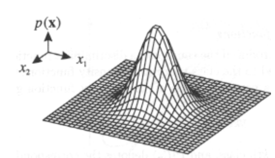
Simplifications

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

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

The 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:

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

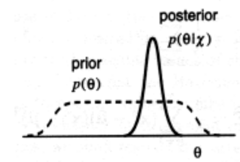
9

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).



10

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.

11

Bayesian Inference

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

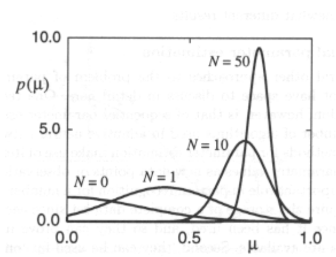
$$p(\theta|\boldsymbol{X}) = \frac{p(\boldsymbol{X}|\theta)p(\theta)}{p(\boldsymbol{X})} = \frac{p(\theta)}{p(\boldsymbol{X})}\prod_{n=1}^{N} p(x^n|\theta)$$

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

12

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 <u>any</u> tractable distribution?

Instead of fitting distribution parameters, we can estimate probability density directly from the training data.

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

Density Estimation Within Region R

$$P = \int_R p(x')\ dx'$$

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

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

$$\text{mean}: E[K/N] = P$$
$$\text{var}: E[(K/N-P)^2] = P(1-P)/N$$

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

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

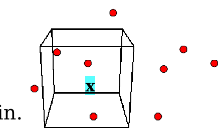
where V is the volume of R, and x is any point in R. So

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

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 $H(\mathbf{u}) = \begin{cases} 1 & \text{if} |u_j| \leq 1/2 \\ 0 & \text{otherwise} \end{cases}$



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

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

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

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

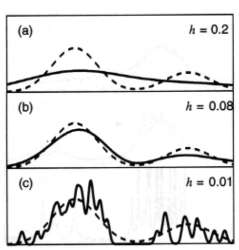
Parzen Windows w/Gaussian Kernels

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

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

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

Varying h changes the amount of smoothing of the estimate.

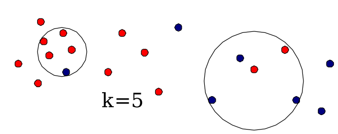


Variable Size Region R

Parzen windows use a fixed kernel size; the number of points falling within the window will vary.

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.



Deriving k-Nearest Neighbor

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

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

$$P(C_k) = \frac{N_k}{N}$$ 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.

High Dimensions Are Strange

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

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

Theorem (Bishop excercise 1.4):

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

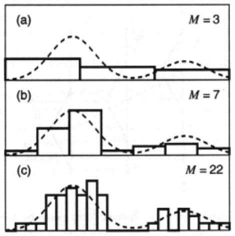
Also, in an n-dim. Gaussian distribution, most of the probability mass is in a thin shell at large radius: a hollow sphere.

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

Semi-Parametric Approaches

Histograms: define a set of bins of fixed size.
Use the histogram to approximate the density.

Brittle: bin edges are arbitrary



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.

Let ξ be an input pattern.

$$\text{Winner } i^* = \operatorname*{argmax}_i \left(\bar{w}_i \cdot \xi\right)$$

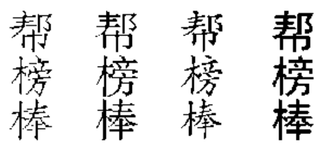
Move winner's weight vector closer to (further from) ξ if winner was correct (incorrect).

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

Example Application: Chinese OCR

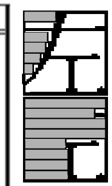
“Simplified” character set has 7000 symbols.

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

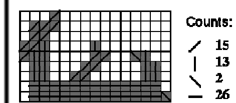


405-Dimensional Feature Set

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



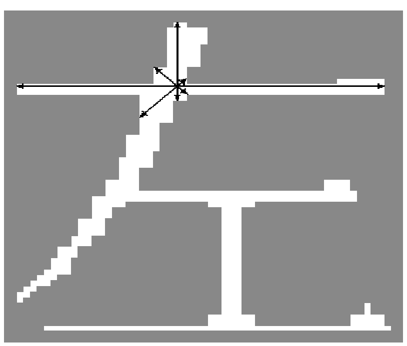
First and second order peripheral features



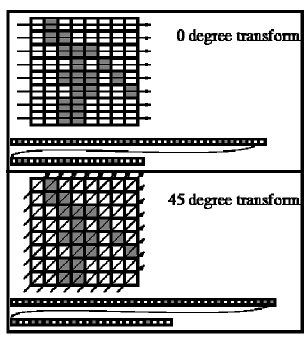
Stroke Proportion

Features from Suchenwirth et al. (1989)

405-Dimensional Feature Set



Local Direction Contributivity



Black Jump Distribution in Balanced Subvectors

Training Strategy

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

1) Collect 20,000 characters of training data.

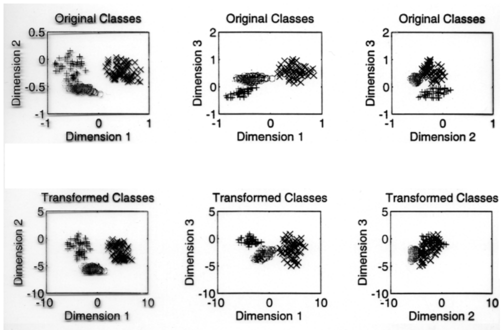
2) Calculate 405 feature values for each character.

3) Extract the top 100 principal components.

4) K-L transform to minimize within-class variance and maximize inter-class variance.

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

Transforming the Feature Space



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.