Example of Naïve Bayes Classifier

Name	Give Birth	Can Fly	Live in Water	Have Legs	Class
human	yes	no	no	yes	mammals
python	no	no	no	no	non-mammals
salmon	no	no	yes	no	non-mammals
whale	yes	no	yes	no	mammals
frog	no	no	sometimes	yes	non-mammals
komodo	no	no	no	yes	non-mammals
bat	yes	yes	no	yes	mammals
pigeon	no	yes	no	yes	non-mammals
cat	yes	no	no	yes	mammals
leopard shark	yes	no	yes	no	non-mammals
turtle	no	no	sometimes	yes	non-mammals
penguin	no	no	sometimes	yes	non-mammals
porcupine	yes	no	no	yes	mammals
eel	no	no	yes	no	non-mammals
salamander	no	no	sometimes	yes	non-mammals
gila monster	no	no	no	yes	non-mammals
platypus	no	no	no	yes	mammals
owl	no	yes	no	yes	non-mammals
dolphin	yes	no	yes	no	mammals
eagle	no	yes	no	yes	non-mammals

A: attributes

M: mammals

N: non-mammals

$P(A|M)=\frac{6}{7}\times\frac{6}{7}\times\frac{2}{7}\times\frac{2}{7}=0.06$

$P(A|N)=\frac{1}{13}\times\frac{10}{13}\times\frac{3}{13}\times\frac{4}{13}=0.0042$

$P(A|M)P(M)=0.06\times\frac{7}{20}=0.021$

$P(A|N)P(N)=0.004\times\frac{13}{20}=0.0027$

Give Birth	Can Fly	Live in Water	Have Legs	Class
no	yes	no	?	

P(A|M)P(M) > P(A|N)P(N)

=> Mammals

Singular Value Decomposition (SVD)

- For an arbitrary matrix $X \in \mathcal{R}^{m\times n}$ there exists a factorization as follows:

$$X = U\Sigma V$$

- where

$$U \in \mathcal{R}^{m\times m}, V \in \mathcal{R}^{n\times n}, UU^T = U^TU = I, VV^T = V^TV = I$$

diagonal matrix $\Sigma \in \mathcal{R}^{m\times n}$

- If $rank(X) = d$

$$U \in \mathcal{R}^{m\times d}, V \in \mathcal{R}^{d\times n}, U^TU = I, VV^T = I$$

$$\Sigma = \text{diag}(\sigma_1, \sigma_2, \cdots \sigma_d)\ \ \sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_d > 0$$

Minimize the **mean-squared error** :

$$J_n = \frac{1}{n}\sum_{i=1}^{n}(y_i - f(\boldsymbol{x}_i))^2$$

Minimize the **residual sum of squares**

$$RSS(f) = \sum_{i=1}^{n}(y_i - f(\boldsymbol{x}_i))^2$$

Optimizing the MSE Criterion

- Computing the gradient gives:

$$\nabla J_n = -2X(\boldsymbol{y} - X^T\boldsymbol{a})$$

- Setting the gradient to zero,

$$XX^T\boldsymbol{a} = X\boldsymbol{y}$$

$$\boldsymbol{a} = (XX^T)^{-1}X\boldsymbol{y}$$

- Any problems?
- What is the rank of the matrix XX^T?
- The solution for **a** can be obtained uniquely if XX^T is non-singular.
- The fitted values at the training inputs are

$$\hat{\boldsymbol{y}} = X^T\boldsymbol{a} = X^T(XX^T)^{-1}X\boldsymbol{y}$$

$$\boldsymbol{a} = (XX^T)^{-1}X\boldsymbol{y}$$

The solution for **a** can be obtained uniquely if XX^T is non-singular.

If XX^T is singular, overfitting

LASSO: 平方变绝对值

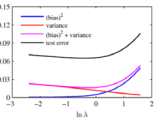
Ridge Regression

$$\boldsymbol{a}^* = \text{argmin}\sum_{i=1}^{n}(y_i - \boldsymbol{x}_i^T\boldsymbol{a})^2 + \lambda\sum_{j=1}^{p}a_j^2$$

$$\boldsymbol{a}^* = (XX^T + \lambda I)^{-1}X\boldsymbol{y}$$

Over-regularized model (large λ) →high bias

Under-regularized model (small λ) → high variance.



Logistic Regression

$$P(y_l = \pm 1|\boldsymbol{x}_l, \boldsymbol{a}) = \sigma(y_l\boldsymbol{a}^T\boldsymbol{x}_l) = \frac{1}{1+e^{-y_l\boldsymbol{a}^T\boldsymbol{x}_l}}$$

$$P(D) = \prod_{l\in I}\sigma(y_l\boldsymbol{a}^T\boldsymbol{x}_l)$$

$$l(P(D)) = \sum_{l\in I}\log\left(\sigma(y_l\boldsymbol{a}^T\boldsymbol{x}_l)\right) = -\sum_{l\in I}\log\left(1+e^{-y_l\boldsymbol{a}^T\boldsymbol{x}_l}\right)$$

Objective Function of Linear Regression:

$$E(\boldsymbol{a}) = \sum_{i\in I}(y_i - \boldsymbol{a}^T\boldsymbol{x}_i)^2$$

$$E(\boldsymbol{a}) = \sum_{i\in I}\log\left(1+e^{-y_i\boldsymbol{a}^T\boldsymbol{x}_i}\right) + \lambda\sum_{j=1}^{p}|a_j^2|$$

Two-category Linearly Separable Case

- If
  - $\boldsymbol{w}^T\boldsymbol{x} > 0$ for examples from the positive class.
  - $\boldsymbol{w}^T\boldsymbol{x} < 0$ for examples from the negative class.

$$\min_{\boldsymbol{w},b}\frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n}\xi_i$$

$$y(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1 - \xi_i$$

$$\xi_i \geq 0$$

Hyper plane of maximum margin is *supported* by those points (vectors) on the margin. Those are called Support Vectors. Non-support vectors can move freely without affecting the position of the hyperplane as long as they don't exceed the margin.

Max margin 优点

- Intuitively this feels safest
- If we've made a small error in the location of the boundary, this gives us least chance of causing misclassification
- There's some theory (using VC dimension) that is related to the proposition that this is a good thing.
- Empirically it works very, very well.

KNN:TAKING MAJORITY VOTE

- Examples
  - Linear kernels $\kappa(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x}^T\boldsymbol{x}'$
  - Polynomial kernels $\kappa(\boldsymbol{x}, \boldsymbol{x}') = (\boldsymbol{x}^T\boldsymbol{x}' + 1)^d$
  - RBF kernels $\kappa(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{x}'\|^2}{2\sigma^2}\right)$

The information gain of an attribute a is the expected reduction in entropy caused by partitioning on this attribute.

$$Gain(S, a) = Entropy(S) - \sum_{v\in values(a)}\frac{|S_v|}{|S|}Entropy(S_v)$$

- where S_v is the subset of S for which attribute a has value v

An Illustrative Example (2)

Humidity	Wind	PlayTennis
High	Weak	No
High	Strong	No
High	Weak	Yes
High	Weak	Yes
Normal	Weak	Yes
Normal	Strong	No
Normal	Strong	Yes
High	Weak	No
Normal	Weak	Yes
Normal	Weak	Yes
Normal	Strong	Yes
High	Strong	Yes
Normal	Weak	Yes
High	Strong	No

Humidity: High (3+,4-) E=.985; Normal (6+,1-) E=.592

Wind: Weak (6+2-) E=.811; Strong (3+,3-) E=1.0

9+,5- E=.94

Gain(S,Humidity)= .94 - 7/14 .985 - 7/14 .592= 0.151

Gain(S,Wind)= .94 - 8/14 .811 - 6/14 1.0 = 0.048

$$Gain(S, a) = Entropy(S) - \sum_{v\in values(a)}\frac{|S_v|}{|S|}Entropy(S_v)$$

Outlook: Sunny (2,8,9,11 2+,3- ?); Overcast (3,7,12,13 4+,0- Yes); Rain (4,5,6,10,14 3+,2- ?)

$Gain(S_{sunny}, Humidity)$ = .97-(3/5) 0-(2/5) 0 = .97

$Gain(S_{sunny}, Temp)$ = .97- 0-(2/5) 1 = .57

$Gain(S_{sunny}, Wind)$ = .97-(2/5) 1 - (3/5) .92= .02

Avoid Overfitting

Day	Outlook	Temperature	Humidity	Wind	PlayTenn
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

Why Bagging Works: Bias & Variance

$$\left(E_D(f(\boldsymbol{x};D)) - E(y|\boldsymbol{x})\right)^2 + E_D\left\{[f(\boldsymbol{x};D) - E_D(f(\boldsymbol{x};D))]^2\right\}$$

- Let f denote the ground-truth function and $h(\boldsymbol{x})$ denote a learner trained from the bootstrap distribution D_{bs}. The aggregated learner generated by Bagging is

$$H(\boldsymbol{x}) = E_{D_{bs}}[h(\boldsymbol{x})]$$

- Bias: $E_D[h(\boldsymbol{x})] - f \approx H(\boldsymbol{x}) - f$
- Variance:
  - Assume there are T bs samples, if data is $i.i.d.$ and each variance is σ^2. The ensemble variance is $\frac{1}{T}\sigma^2$

$$H(\boldsymbol{x}) = E_{D_{bs}}[h(\boldsymbol{x})] = \frac{1}{T}\sum_{i=1}^{T}h_i(\boldsymbol{x})$$

$$Var(H(\boldsymbol{x})) = Var\left(\frac{1}{T}\sum_{i}^{T}h_i(\boldsymbol{x})\right) = \frac{1}{T^2}Var\left(\sum_{i=1}^{T}h_i(\boldsymbol{x})\right) = \frac{1}{T^2}\sum_{i=1}^{T}Var(h_i(\boldsymbol{x})) = \frac{1}{T}\sigma^2$$

  - In reality, Bootstrap Sampled data is $i.d.$ (not necessarily independent) with correlation ρ

$$\rho\sigma^2 + \frac{1-\rho}{T}\sigma^2$$

Construct Random Forest

- Let N be the number of trees and K be the feature subset size.
- For each N iterations: (building a tree in forest)
  - 1. Select a new bootstrap sample from original training set
  - 2. Growing a tree…
  - 3. At each internal node, randomly select K features from ALL features and then, determine the best split in ONLY the K features.
  - 4. Do not pruning
  - 5. Until Test Error never decrease (here means Validation error in RF)
- At last, overall prediction as the average(or vote) from N trees.

Minimum Error Rate Classification

$$R(\alpha_x \mid \boldsymbol{x}) = 1 - P(\omega_i \mid \boldsymbol{x})$$

$$R(\alpha_i \mid \boldsymbol{x}) = \sum_{j=1}^{c}\lambda(\alpha_i \mid \omega_j)P(\omega_j \mid \boldsymbol{x})$$

- Bayes Risk

$$R = \sum_{\boldsymbol{x}}R(\alpha_i \mid \boldsymbol{x})$$

- Bias-variance Decomposition

Expected prediction error (expected loss) = (bias)²+variance+noise

- Weakness of the Original Model

SVM缺点

- When an outlier appear, the optimal hyperplane may be pushed far away from its original/correct place.
- Assign a slack variable ξ to each data point. That means we allow the point to deviate from the correct margin by a distance of ξ

条件风险 condition risk

$$R(\alpha_i|\boldsymbol{x}) = \sum_{j=1}^{c}\lambda(\alpha_i|\omega_j)P(\omega_j|\boldsymbol{x})$$

MLE 高斯分布

$$\hat{\mu} = \frac{1}{n}\sum_{k=1}^{n}x_k$$

$$\hat{\Sigma} = \frac{1}{n}\sum_{k=1}^{n}(x_k - \hat{\mu})(x_k - \hat{\mu})^t$$

Kmeans 目标函数

$$\mathbf{SSE} = \sum_{i=1}^{K}\sum_{x\in C_i}dist(c_i, x)^2$$

GMM 目标函数

$$\log\prod_{i=1}^{N}p(\boldsymbol{x}^{(i)};\boldsymbol{\theta}) = \sum_{i=1}^{N}\log\left(\sum_{k=1}^{K}\pi_k\mathcal{N}(\boldsymbol{x}^{(i)};\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k)\right)$$

EM算法

(E-step) for each i :

$$Q^i(\boldsymbol{z}^{(i)}) = p(\boldsymbol{z}^{(i)}|\boldsymbol{x}^{(i)};\boldsymbol{\theta})$$

(M-step) Set:

$$\boldsymbol{\theta} := \arg\max_{\boldsymbol{\theta}}\sum_{i=1}^{m}\sum_{\boldsymbol{z}^{(i)}}Q^i(\boldsymbol{z}^{(i)})\log\frac{p(\boldsymbol{x}^{(i)},\boldsymbol{z}^{(i)};\boldsymbol{\theta})}{Q^i(\boldsymbol{z}^{(i)})}$$

SVD与PCA不同，PCA是对 数据的协方差矩阵进行矩阵的分解，而SVD是直接在 原始矩阵上进行的矩阵分解。并且能对非方阵矩阵 分解，得到左奇异矩阵U、sigma矩阵Σ、右奇异矩阵 VT。

LDA

$$\Sigma = \sum_{i=1}^{2}\sum_{j\in\omega_i}\frac{(\boldsymbol{x}_j - \boldsymbol{\mu}_i)(\boldsymbol{x}_j - \boldsymbol{\mu}_i)^T}{N}$$

Knn 目标函数

$$y = arg\ max_{c_j}\sum_{x_i\in N_k(x)}I(y_i = c_j)$$

Clusters

Kmeans: （kndt）

- Partition objects into k nonempty subsets
- Compute seed points as the centroids of the clusters of the current partitioning
- Assign each object to the cluster with the nearest seed point
- Go back to Step 2, stop when he assignment does not change

1、随机选取K个质心的值
2、计算各个点到质心的距离
3、将点的类划分为离他最近的质心，形成K个cluster
4、根据分类好的cluster，在每个cluster内重新计算质心(平均每个点的值)
5、重复迭代2-4步直到满足迭代次数或误差小于指定的值

K-Medoids（kndt）：Instead of taking the mean value of the object in a cluster as a reference point(不在内部), medoids can be used, which is the most centrally located object in a cluster. 1、随机选取K个质心的值（质心必须是某些样本点的值，而不是任意值）2、计算各个点到质心的距离 3、将点的类划分为离他最近的质心，形成K个cluster 4、根据分类好的cluster，在每个cluster内重新计算质心： 4.1 计算cluster内所有样本点到其中一个样本点的曼哈顿距离和(绝对误差) 4.2 选出使cluster绝对误差最小的样本点作为质心 5、重复迭代2-4步直到满足迭代次数或误差小于指定的

值.**PAM(n^2dt):**0. Calculate the pair-wise distance matrix W
1.Initialize: randomly select of the data poi nts as the medoids
2.Associate each data point to the closest medoid
3.For each cluster, compute its medoid
4.Repeat 2-3 until there is no change in the medoids（怎么选？离之心最近的或者distance row sum 最小的）

GMM:

$$p(x;\Theta)=\sum_{k=1}^{K}\pi_k p_k(x;\theta_k)$$

$$\Theta=\{\pi_1,\ldots,\pi_k,\theta_1,\ldots,theta_k\},\sum\pi_k=1$$

$$p_k(x;\theta_k)=\mathcal{N}(x;\mu_k;\Sigma_k)$$

For each point $x^{(i)}$, associate with a hidden variable $z^{(i)}$ denotes which Gaussian $x^{(i)}$ belongs to

Spectral clustering:

Divide vertices into two disjoint groups (A,B). Minimize weight of between-group connections.

$$cut(A,B)=\sum_{i\in A,j\in B}w_{ij}$$

$$assoc(A,V)=\sum_{i\in A,j\in V}w_{ij}$$

$$\min Ncut(A,B)=\frac{cut(A,B)}{assoc(A,V)}+\frac{cut(A,B)}{assoc(B,V)}$$

Max cluster间的，min cluster之间的

Generalized Eigen-problem

$$\min_{\boldsymbol{y}}\frac{\boldsymbol{y}^T(D-W)\boldsymbol{y}}{\boldsymbol{y}^TD\boldsymbol{y}},\boldsymbol{y}\in\mathcal{R}^n,\boldsymbol{y}^TD\mathbf{1}=0$$

Obj of NCut

$$(D-W)\boldsymbol{y}=\lambda D\boldsymbol{y}$$

- Eigenvector corresponding to the smallest eigenvalue.
- Vector $\mathbf{1}$ is the eigenvector corresponding to the eigenvalue 0.

$$(D-W)\boldsymbol{y}=\lambda D^{\frac{1}{2}}D^{\frac{1}{2}}\boldsymbol{y}$$
$$D^{-\frac{1}{2}}(D-W)D^{-\frac{1}{2}}D^{\frac{1}{2}}\boldsymbol{y}=\lambda D^{\frac{1}{2}}\boldsymbol{y}$$
$$D^{-\frac{1}{2}}(D-W)D^{-\frac{1}{2}}\boldsymbol{z}=\lambda\boldsymbol{z}$$

- $\boldsymbol{z}_1^T\boldsymbol{z}_2=0\ \rightarrow\ (D^{\frac{1}{2}}\boldsymbol{y}_1)^T(D^{\frac{1}{2}}\boldsymbol{y}_2)=0\ \rightarrow\ \boldsymbol{y}_1^TD\boldsymbol{y}_2=0$
- The eigenvector corresponding to the 2nd small eigenvalue.

Spectral Clustering Algorithm

1. Graph construction
   - Heat kernel $w_{ij}=\exp\left\{-\frac{\|\boldsymbol{x}_i-\boldsymbol{x}_j\|}{2\sigma^2}\right\}$
   - k-nearest neighbor graph
2. Eigen-problem
   - Compute eigenvalues and eigenvectors of the matrix L
   - Map each point to a lower-dimensional representation based on one or more eigenvectors.
3. Conventional clustering schemes, e.g. K-Means
   - Assign points to two or more clusters, based on the new representation.

- Rayleigh quotient

$$\min_{\boldsymbol{y}}\frac{\boldsymbol{y}^TL\boldsymbol{y}}{\boldsymbol{y}^TD\boldsymbol{y}},\boldsymbol{y}\in\mathcal{R}^n,\boldsymbol{y}^TD\mathbf{1}=0$$
$$\Rightarrow(D-W)\boldsymbol{y}=\lambda D\boldsymbol{y}$$

```
D = EuDist2(X);
W = (D <= max(mink(D, k+1)));
W = W & (D <= threshold) ;
W = W - eye(size(D)); % clear the distance to itself
W = double(W | W'); % merge the two-direction matrix
----------------------
D = eye(size(W));
D = D .* sum(W,2);
[Y, ~] = eigs(D - W, D, 2, 'smallestabs');
idx = kmeans(Y(: 2), k);
```

- Perceptron

$$f(x)=w^Tx+w_0$$
$$y=sign(f(x))$$

  - Learn
    - If correct, do nothing
    - If wrong, $w_t=w_{t-1}+xy$

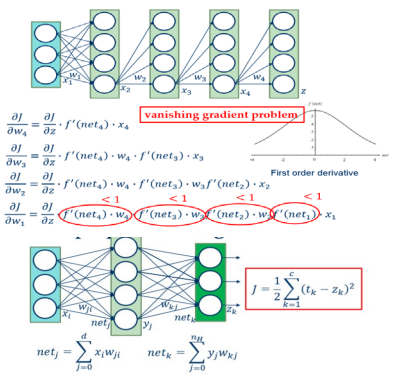
- Perceptron Criterion Function

$$E(w)=-\sum_{i\in I_M}w^Tx_iy_i$$

- Gradient Descent (Steepest descent)

$$x_{n+1}=x_n-\gamma\nabla f(x_n),n\geqslant 0$$

Backpropagation Algorithm



$$\frac{\partial J}{\partial w_4}=\frac{\partial J}{\partial z}\cdot f'(net_4)\cdot x_4$$
$$\frac{\partial J}{\partial w_3}=\frac{\partial J}{\partial z}\cdot f'(net_4)\cdot w_4\cdot f'(net_3)\cdot x_3$$
$$\frac{\partial J}{\partial w_2}=\frac{\partial J}{\partial z}\cdot f'(net_4)\cdot w_4\cdot f'(net_3)\cdot w_3 f''(net_2)\cdot x_2$$
$$\frac{\partial J}{\partial w_1}=\frac{\partial J}{\partial z}\cdot f'(net_4)\cdot w_4\cdot f'(net_3)\cdot w_3 f''(net_2)\cdot w_2 f''(net_1)\cdot x_1$$

$$net_j=\sum_{j=0}^{d}x_iw_{ji}\qquad net_k=\sum_{j=0}^{n_H}y_jw_{kj}$$

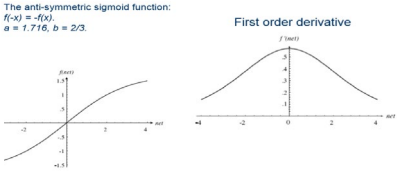
$$\frac{\partial J}{\partial w_{kj}}=\frac{\partial J}{\partial z_k}\cdot\frac{\partial z_k}{\partial net_k}\cdot\frac{\partial net_k}{\partial w_{kj}}=(z_k-t_k)\cdot f'(net_k)\cdot y_j$$

$$\frac{\partial J}{\partial w_{ji}}=\frac{\partial J}{\partial y_j}\cdot\frac{\partial y_j}{\partial net_j}\cdot\frac{\partial net_j}{\partial w_{ji}}$$
$$=\left(\sum_{k=1}^{c}(z_k-t_k)\cdot f'(net_k)\cdot w_{kj}\right)\cdot f'(net_j)\cdot x_i$$

Activation Function

$$f(net)=a\tanh(b\ net)=a\left[\frac{1-e^{b\ net}}{1+e^{b\ net}}\right]=\frac{2a}{1+e^{-b\ net}}-a$$

The anti-symmetric sigmoid function:
f(-x) = -f(x),
a = 1.716, b = 2/3.



Activation function: ReLU(x) = max (0, x)

PCA：

Main steps for computing PCs:

- Form the covariance matrix S.
- Compute its eigenvectors: $\{\boldsymbol{a}_i\}_{i=1}^p$
- Use the first d eigenvectors $\{\boldsymbol{a}_i\}_{i=1}^d$ to form the d PCs.
- The transformation A is given by $A=[\boldsymbol{a}_1,\cdots\boldsymbol{a}_d]$

几种常见的数据挖掘算法

- Naïve Bayesian classifier
- Linear Regression
- Logistic Regression
- SVM
- Perceptron
- Neural Network
- k Nearest Neighbor
- Decision Tree

正太分布：

Multivariate density: $N(\boldsymbol{\mu},\Sigma)$ (with dimension d)

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

- $\boldsymbol{x}=[x_1,\cdots,x_d]^T$
- $\boldsymbol{\mu}=[\mu_1,\cdots,\mu_d]^T$
- Σ: $d\times d$ covariance matrix, $|\cdot|$: determinant

$$P(x|\omega_i)=\frac{1}{\sqrt{2\pi\sigma_i^2}}\exp\left(-\frac{(x-\mu_i)^2}{2\sigma_i^2}\right)$$

MLE：

Maximum-Likelihood Estimation

- Use the information in training samples to estimate $\theta=(\theta_1,\theta_2,\ldots,\theta_c)$; θ_i (i = 1, 2, …, c) is associated with the i-th category
- Suppose sample set D contains n iid samples, $x_1, x_2,\ldots, x_n$

$$p(D|\theta)=\prod_{k=1}^{n}p(x_k|\theta)$$

- $p(D|\theta)$ is called the likelihood of θ w.r.t. the set of samples.
- ML estimate of θ is, by definition, the value θ that maximizes p(D | θ)

"It is the value of θ that best agrees with the actually observed training samples"

Bayes Estimation:

- We assume that the true values of the a priori probabilities are known or obtainable from a trivial calculation;
  - We substitute $P(\omega_i)=P(\omega_i|D)$
- Furthermore, we can separate the training samples by class into c subsets $D_1, D_2,\cdots, D_c$, with the samples in D_i belonging to ω_i

$$P(\omega_i|\mathbf{x},\mathcal{D})=\frac{p(\mathbf{x}|\omega_i,\mathcal{D}_i)P(\omega_i)}{\sum_{j=1}^{c}p(\mathbf{x}|\omega_j,\mathcal{D}_j)P(\omega_j)}.$$

- In essence, we have c separate problems of the following form: use a set D of samples drawn independently according to the fixed but unknown probability distribution $p(\boldsymbol{x})$ to determine

$$P(x|D)$$

- This is the central problem of Bayesian learning

Bayesian Parameter Estimation: General Theory

P(x | D) computation can be applied to any situation in which the unknown density can be parametrized: the basic assumptions are:

- The form of P(x | θ) is assumed known, but the value of θ is not known exactly
- Our knowledge about θ is assumed to be contained in a known prior density P(θ)
- The rest of our knowledge about θ is contained in a set D of n random variables $x_1, x_2, \ldots, x_n$ that follows P(x)

Discriminant Functions for the Normal Density

- The minimum error-rate classification can be achieved by the discriminant function

$$g_i(\boldsymbol{x})=\ln P(\boldsymbol{x}|\omega_i)+\ln P(\omega_i)$$

- In case of multivariate normal densities

$$P(\boldsymbol{x}|\omega_i)=\frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma_i|^{\frac{1}{2}}}\exp\left[-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_i)^T\Sigma_i^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_i)\right]$$

$$g_i(\boldsymbol{x})=-\tfrac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_i)^T\Sigma_i^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_i)-\tfrac{d}{2}\ln 2\pi-\tfrac{1}{2}\ln|\Sigma_i|+\ln P(\omega_i)$$

$$P(error)=\int_{R_2}p(x\mid\omega_1)P(\omega_1)dx+\int_{R_1}p(x\mid\omega_2)P(\omega_2)dx$$

$$\begin{aligned}P(correct)&=\sum_{i=1}^{c}P(\mathbf{x}\in\mathcal{R}_i,\omega_i)\\&=\sum_{i=1}^{c}P(\mathbf{x}\in\mathcal{R}_i|\omega_i)P(\omega_i)\\&=\sum_{i=1}^{c}\int_{\mathcal{R}_i}p(\mathbf{x}|\omega_i)P(\omega_i)\ d\mathbf{x}.\end{aligned}$$

Naïve Bayes Classifier

- Given $\boldsymbol{x}=(x_1,\cdots x_p)^T$
  - Goal is to predict class ω
  - Specifically, we want to find the value of ω that maximizes $P(\omega|\boldsymbol{x})=P(\omega|x_1,\cdots x_p)$

$$P(\omega|x_1,\cdots x_p)\propto P(x_1,\cdots x_p|\omega)P(\omega)$$

- Independence assumption among features

$$P(x_1,\cdots x_p|\omega)=P(x_1|\omega)\cdots P(x_p|\omega)$$

How to Estimate Probabilities from Data?

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

- Class: $P(\omega_k)=\frac{N_{\omega_k}}{N}$
  - e.g., P(No) = 7/10, P(Yes) = 3/10
- For discrete attributes:

$$P(x_i|\omega_k)=\frac{|x_{ik}|}{N_{\omega_k}}$$

  - where $|x_{ik}|$ is number of instances having attribute x_i and belongs to class ω_k
  - Examples:

P(Status=Married | No) = 4/7
P(Refund=Yes | Yes)=0