CUNY School of Professional Studies

SPS.CUNY.EDU

Lecture 03
2020 Spring Data-622
Review of Statistics and Probability with R

Raman Kannan

Instructor Email Address: Raman.Kannan@sps.cuny.edu

Acknowledgements:
Generous support from IBM Power Systems Academic Initiative
IBM PSAI provides computing infrastructure for free

Refresher

In the next two weeks we will set a goal to achieve:
working proficiency in R and
introduce essential concepts from Statistical/Probability/Linear Algebra

Our learning objective is to refresh some essential concepts from Statistics, Linear Algebra, Calculus and Probability using R. Empirical not theoretical.

Linear Algebra
- Matrix: positive definite, semi-positive, rectangular,square, Identity, diagonal
- Multiplication,determinant,Inverse, transpose, Decomposition, cholesky,eigen,svd
- Vector, dot product, cross product,length

Calculus:derivative,partial derivative, integration,log,min,max
Statistics
- Sample (statistic)mean,variance,Sampling distribution of Sample statistic
- Population(parameters)
- Accuracy, precision, bias
- CLT Central Limit Theorem, LLN Law of Large Numbers
- Probability:probablity distribution of discrete/continuous variables,pmf,pdf,cdf.
- Joint/Conditional/Independent events, their probabilities and Bayes Theorem.

We introduce as many of these which are doable in R. All theoretical exercise is left to the learner as a not-for-credit reading exercise.

Stats and Math you need

https://www.datasciencecentral.com/profiles/blogs/math-for-data-science

ML vs Holt-Winters Forecasting - Jan 14 REGISTER TODAY ›

R Exercises

Matrix
- Transpose,inverse,
- Decomposition // factorization

Distributions
- Normal
- Poisson
- binomial

This course favors practice while recognizing the importance of math/stat/probability

Working Environment: R on IBM Cloud

```
> X<-as.matrix(c(1,-1,2))
> X
     [,1]
[1,]    1
[2,]   -1
[3,]    2
> X<-cbind(X,c(1,0,1))
> X
     [,1] [,2]
[1,]    1    1
[2,]   -1    0
[3,]    2    1
> X<-cbind(X,c(1,2,-1))
> X
     [,1] [,2] [,3]
[1,]    1    1    1
[2,]   -1    0    2
[3,]    2    1   -1
> X<-cbind(X,c(1,1,0))
> X
     [,1] [,2] [,3] [,4]
[1,]    1    1    1    1
[2,]   -1    0    2    1
[3,]    2    1   -1    0
```

```
> chol(t(X2)%*%X2)
         [,1]     [,2]      [,3]
[1,] 2.645751 1.133893 0.0000000
[2,] 0.000000 2.171241 0.9211324
[3,] 0.000000 0.000000 1.4668044
> chol(X2%*%t(X2))
         [,1]      [,2]      [,3]          [,4]
[1,] 1.414214 -1.414214 0.7071068 7.071068e-01
[2,] 0.000000  1.000000 1.0000000 2.000000e+00
[3,] 0.000000  0.000000 2.1213203 7.071068e-01
[4,] 0.000000  0.000000 0.0000000 4.080851e-08
> cholX2<-chol(X2%*%t(X2))
> cholt X2<-chol(t(X2)%*%X2)
> cholt X2%*%t(cholt X2)
         [,1]     [,2]     [,3]
[1,] 8.285714 2.461955 0.000000
[2,] 2.461955 5.562771 1.351121
[3,] 0.000000 1.351121 2.151515
> cholt X2%*%cholt X2
     [,1]     [,2]     [,3]
[1,]    7 5.461955 1.044466
[2,]    0 4.714286 3.351121
[3,]    0 0.000000 2.151515
```

https://www.dummies.com/programming/r/how-to-do-matrix-arithmetic-in-r/

For R, you can use Rstudio or Rgui. Or on IBM Cloud as shown above. On IBM, enter R to start R.

Inverting a matrix

```
> mxex<-matrix(c(2,0,4,1,-1,0,0,1,-2),nrow=3)
>  mxex
     [,1] [,2] [,3]
[1,]    2    1    0
[2,]    0   -1    1
[3,]    4    0   -2
> t(mxex)
     [,1] [,2] [,3]
[1,]    2    0    4
[2,]    1   -1    0
[3,]    0    1   -2
> solve(t(mxex))
      [,1]  [,2]  [,3]
[1,] 0.250  0.50  0.50
[2,] 0.250 -0.50  0.50
[3,] 0.125 -0.25 -0.25
>
> #solve to invert a matrix
>  solve(t(mxex))%*%t(mxex)
     [,1] [,2] [,3]
[1,]    1    0    0
[2,]    0    1    0
[3,]    0    0    1
>
> solve(mxex)%*%(mxex)
     [,1] [,2] [,3]
[1,]    1    0    0
[2,]    0    1    0
[3,]    0    0    1
```

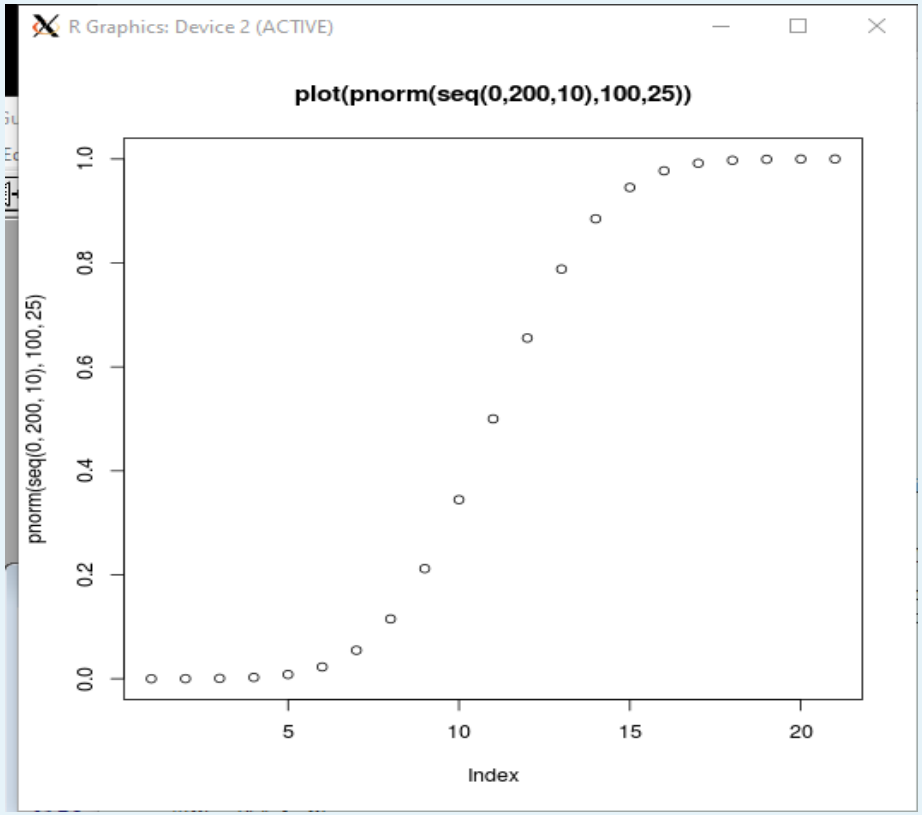
solve is the matrix inverse function.

*inv(M)*M → I*

For R, you can use Rstudio or Rgui. Or on IBM Cloud as shown above.

Probability Distributions and Random Variables

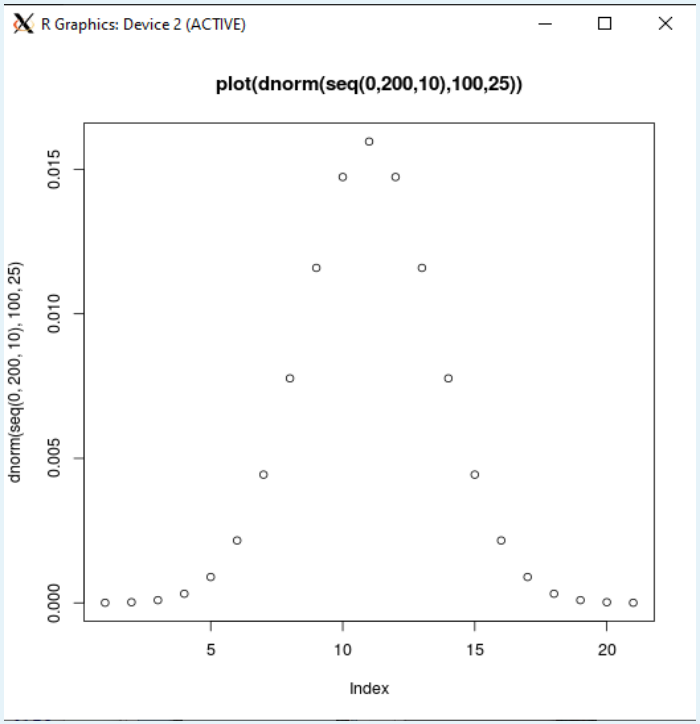
```
> pnorm(13,100,13)
[1] 1.098392e-11
> pnorm(6,100,13)
[1] 2.401329e-13
> pnorm(60,100,13)
[1] 0.001045746
> pnorm(70,100,13)
[1] 0.01050813
> pnorm(80,100,13)
[1] 0.0619679
```

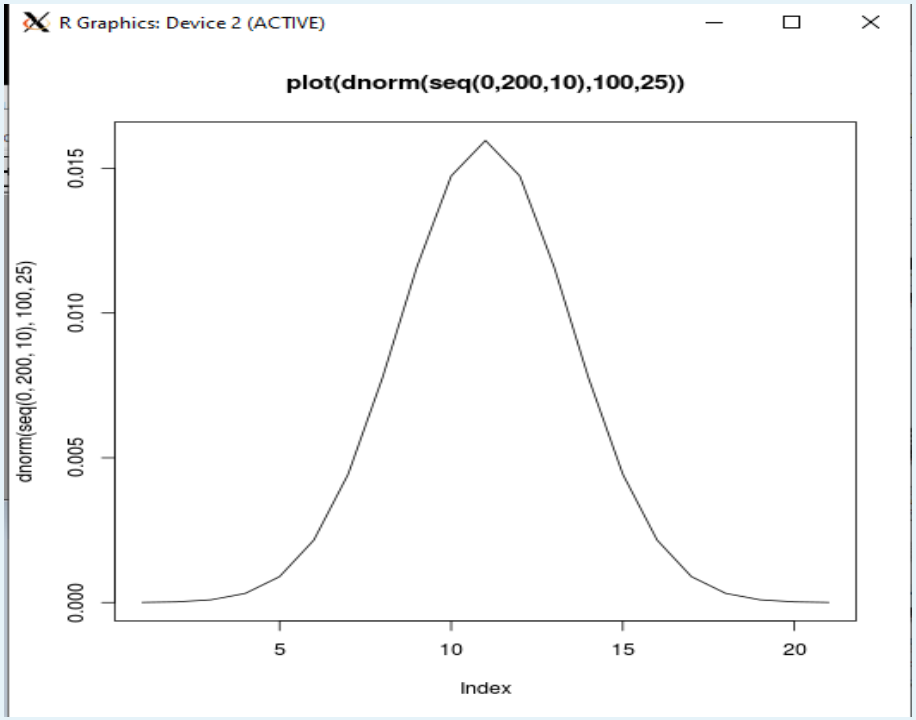


plot(pnorm(seq(0,200,10),100,25),main="plot(pnorm(seq(0,200,10),100,25))")

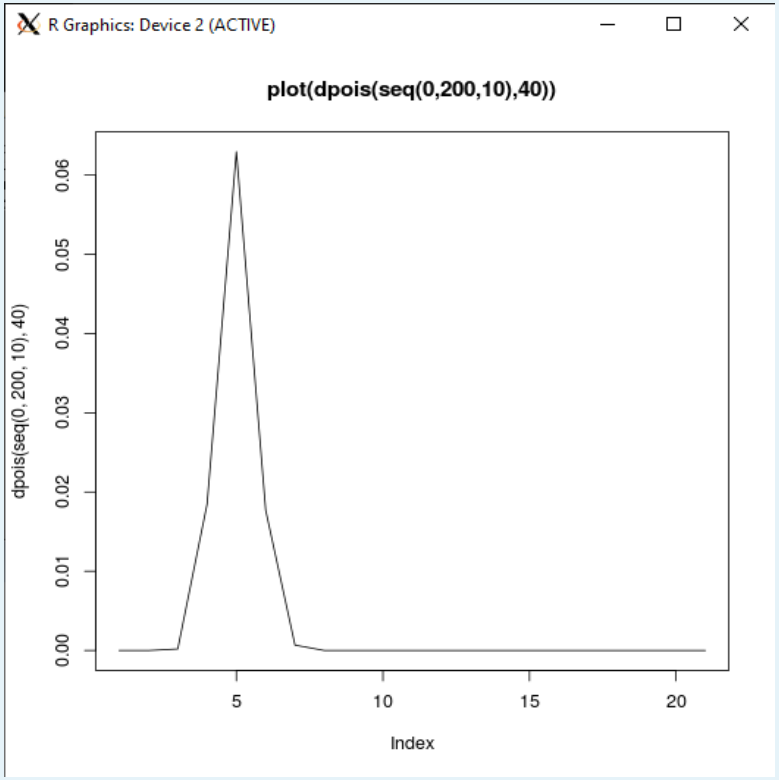
Plotting densities – Gaussian

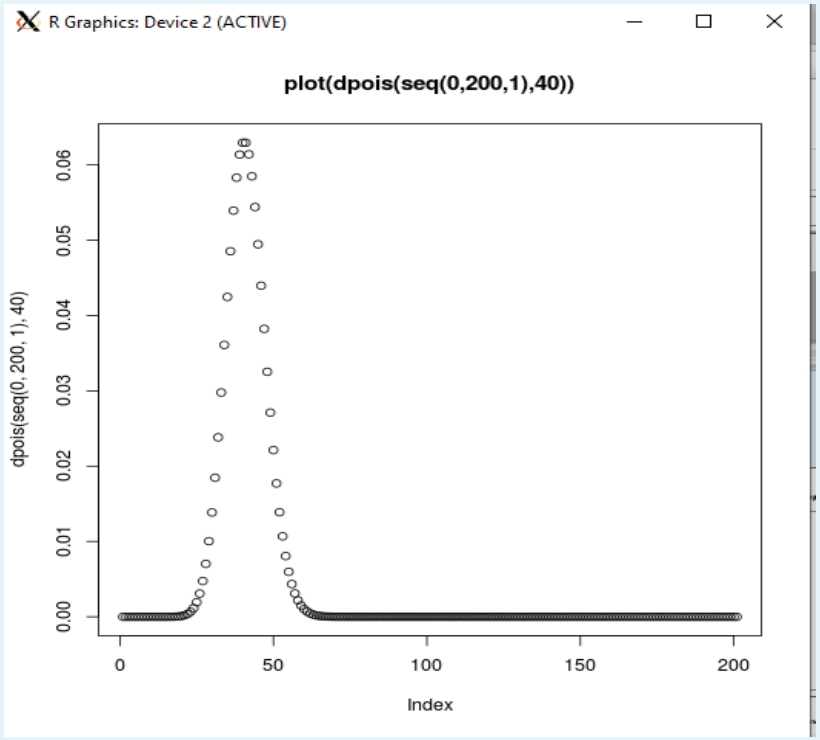
```
plot(dnorm(seq(0,200,10),100,25),
main="plot(dnorm(seq(0,200,10),100,25))",type="l")
plot(dnorm(seq(0,200,10),100,25),main="plot(dnorm(seq(0,200,10),100,25))")
```





poisson






```
plot(dpois(seq(0,200,10),40),main="plot(dpois(seq(0,200,10),40
))",type="l")
plot(dpois(seq(0,200,1),40),main="plot(dpois(seq(0,200,1),40))")
```

Generating densities

```
> dpois(seq(0,20,1),10)
 [1] 4.539993e-05 4.539993e-04 2.269996e-03 7.566655e-03 1.891664e-02
 [6] 3.783327e-02 6.305546e-02 9.007923e-02 1.125990e-01 1.251100e-01
[11] 1.251100e-01 1.137364e-01 9.478033e-02 7.290795e-02 5.207710e-02
[16] 3.471807e-02 2.169879e-02 1.276400e-02 7.091109e-03 3.732163e-03
[21] 1.866081e-03
> dnorm(seq(0,20,1),10)
 [1] 7.694599e-23 1.027977e-18 5.052271e-15 9.134720e-12 6.075883e-09
 [6] 1.486720e-06 1.338302e-04 4.431848e-03 5.399097e-02 2.419707e-01
[11] 3.989423e-01 2.419707e-01 5.399097e-02 4.431848e-03 1.338302e-04
[16] 1.486720e-06 6.075883e-09 9.134720e-12 5.052271e-15 1.027977e-18
[21] 7.694599e-23
> dbinom(seq(0,20,1),10)
Error in dbinom(seq(0, 20, 1), 10) :
  argument "prob" is missing, with no default
> dbinom(seq(0,20,1),10,prob=0.4)
 [1] 0.0060466176 0.0403107840 0.1209323520 0.2149908480 0.2508226560
 [6] 0.2006581248 0.1114767360 0.0424673280 0.0106168320 0.0015728640
[11] 0.0001048576 0.0000000000 0.0000000000 0.0000000000 0.0000000000
[16] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
[21] 0.0000000000
>
```

dpois(seq(0,20,1),10)
dnorm(seq(0,20,1),10)
dnorm(seq(0,20,1),mean=10,sd=3)
dbinom(seq(0,20,1),10,prob=0.4))

Generating Random Variables

```
> dnorm(seq(0,20,1),mean=10,sd=3)
 [1] 0.000514093 0.001477283 0.003798662 0.008740630 0.017996989 0.033159046
 [7] 0.054670025 0.080656908 0.106482669 0.125794409 0.132980760 0.125794409
[13] 0.106482669 0.080656908 0.054670025 0.033159046 0.017996989 0.008740630
[19] 0.003798662 0.001477283 0.000514093
>
> pnorm(20,mean=10,sd=3)
[1] 0.9995709
> rnorm(20,mean=10,sd=3)
 [1]  9.376419  6.627208  8.453833 11.048097  6.502322 11.527968  9.335249
 [8] 10.487730 13.532382  7.236364  7.616700 10.968781 15.649711  6.911608
[15]  9.432571  5.543706 13.311440 10.505815  8.427892 10.887784
```

dnorm is generating probability density for a gaussian of mean ->10 and standard deviation 3 pnorm is generating CDF.

rnorm is returning n variates for the given normal distribution

http://www.stat.umn.edu/geyer/old/5101/rlook.html

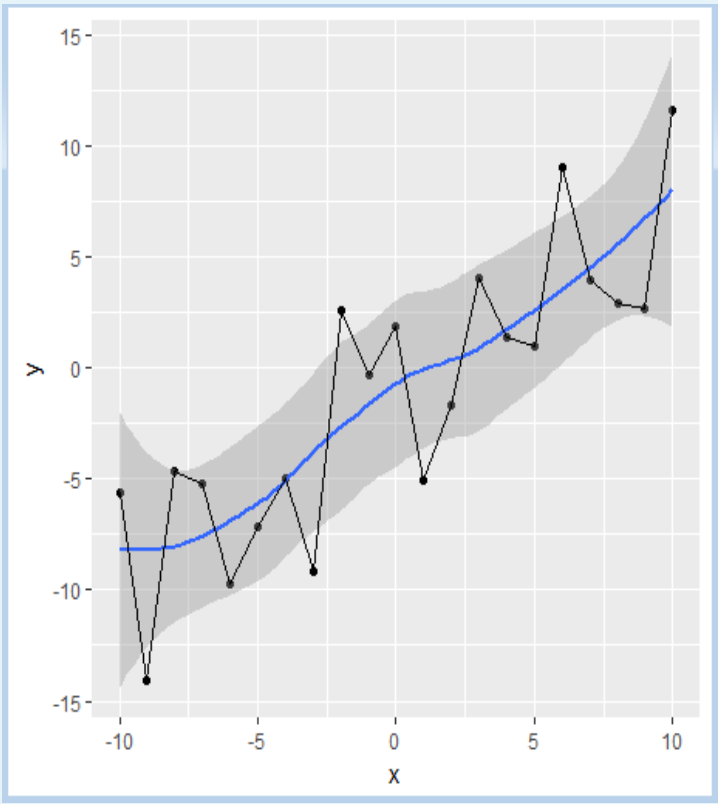
Plotting with ggplot

```
x<- (-10):10
n<-length(x)
y<-rnorm(n,x,4)
# returns n random variates,
# since x is a vector,
# each random variate is
# centered around x and with a sd of 4

ggplot(data.frame(x=x,y=y),aes(x=x,y=y))+
geom_point(aes(x=x,y=y))+
geom_line(aes(x=x,y=y))

ggplot(data.frame(x=x,y=y),aes(x=x,y=y))+
geom_point(aes(x=x,y=y))+geom_smooth()

ggplot(data.frame(x=x,y=y),aes(x=x,y=y))+
geom_point(aes(x=x,y=y))+geom_smooth()+
 geom_line(aes(x=x,y=y))
```



Concept Tree

Performance
- Loss Function
- argmin
- softmax
- AUC
- ROC
- Accuracy
- Precision
- Sensitivity
- Specificity
- Recall
- AIC/BIC

Probability
- Joint
- Conditional
- Bayes Theorem
- Summation
- Prior
- Likelihood
- Posterior
- Marginal
- Correlation
- Covariance
- Mutlivariate
- Expected value
- parameter

Classification
- Categorical
- Label
- Class
- Separation Margin
- hyperplane – Orion
- Linear
- NonLinear
- TP,TN,FP,FN
- Confusion Matrix

Fundamental
- Bias
- Variance
- Penalization/Regularization
- Generalization
- Overfitting/Underfitting
- Resampling/Bagging/Boosting
- Balanced/Bon Ferroni Correction

Essential Ideas
- NFL No Free Lunch
- KISS – Occam's Razor
- Curse of Dimensionality
- PCA
- Parametric/Non Parametric
- Instance/Logic Based
- Information Theory
  - Entropy

Probability

Probability
- Joint
- Conditional
- Bayes Theorem
- Summation
- Prior
- Likelihood
- Posterior
- Marginal
- Correlation
- Covariance
- Mutlivariate
- Expected value
- parameter

C.3 Conditional probability and independence

Definition C.8 (Conditional probability) *The conditional probability of event A given event B is defined by*

$$\mathbb{P}[A \mid B] = \frac{\mathbb{P}[A \cap B]}{\mathbb{P}[B]}, \tag{C.2}$$

when $\mathbb{P}[B] \neq 0$.

Definition C.9 (Independence) *Two events A and B are said to be* independent *if*

$$\mathbb{P}[A \cap B] = \mathbb{P}[A]\,\mathbb{P}[B]. \tag{C.3}$$

Equivalently, A and B are independent iff $\mathbb{P}[A \mid B] = \mathbb{P}[A]$ when $\mathbb{P}[B] \neq 0$.

A sequence of random variables is said to be *independent and identically distributed (i.i.d.)* when the random variables are mutually independent and follow the same distribution.

The following are basic probability formulae related to the notion of conditional probability. They hold for any events A, B, and $A_1, \ldots, A_n$, with the additional constraint $\mathbb{P}[B] \neq 0$ needed for the Bayes formula to be well defined:

$$\mathbb{P}[A \cup B] = \mathbb{P}[A] + \mathbb{P}[B] - \mathbb{P}[A \cap B] \quad \textit{(sum rule)} \tag{C.4}$$

$$\mathbb{P}[\bigcup_{i=1}^{n} A_i] \leq \sum_{i=1}^{n} \mathbb{P}[A_i] \quad \textit{(union bound)} \tag{C.5}$$

$$\mathbb{P}[A \mid B] = \frac{\mathbb{P}[B \mid A]\,\mathbb{P}[A]}{\mathbb{P}[B]} \quad \textit{(Bayes formula)} \tag{C.6}$$

$$\mathbb{P}[\bigcap_{i=1}^{n} A_i] = \mathbb{P}[A_1]\,\mathbb{P}[A_2 \mid A_1] \cdots \mathbb{P}[A_n \mid \bigcap_{i=1}^{n-1} A_i] \quad \textit{(chain rule)}. \tag{C.7}$$

The sum rule follows immediately from the decomposition of $A \cup B$ as the union of the disjoint sets A and $(B - A \cap B)$. The union bound is a direct consequence of the sum rule. The Bayes formula follows immediately from the definition of conditional probability and the observation that: $\mathbb{P}[A|B]\,\mathbb{P}[B] = \mathbb{P}[B|A]\,\mathbb{P}[A] = \mathbb{P}[A \cap B]$. Similarly, the chain rule follows the observation that $\mathbb{P}[A_1]\,\mathbb{P}[A_2|A_1] = \mathbb{P}[A_1 \cap A_2]$; using the same argument shows recursively that the product of the first k terms of the right-hand side equals $\mathbb{P}[\bigcap_{i=1}^{k} A_i]$.

Finally, assume that $\Omega = A_1 \cup A_2 \cup \ldots \cup A_n$ with $A_i \cap A_j = \emptyset$ for $i \neq j$, i.e., the A_is are mutually disjoint. Then, the following formula is valid for any event B:

$$\mathbb{P}[B] = \sum_{i=1}^{n} \mathbb{P}[B \mid A_i]\,\mathbb{P}[A_i] \quad \textit{(theorem of total probability)}. \tag{C.8}$$

This follows the observation that $\mathbb{P}[B \mid A_i]\,\mathbb{P}[A_i] = \mathbb{P}[B \cap A_i]$ by definition of the conditional probability and the fact that the events $B \cap A_i$ are mutually disjoint.

Foundations of M/L – Mehryar Mohri, Afshin Rostamizadeh, Ameet Talwalkar

Probability

Probability
- Joint
- Conditional
- Bayes Theorem
- Summation
- Prior
- Likelihood
- Posterior
- Marginal
- Correlation
- Covariance
- Mutlivariate
- Expected value
- parameter

MULTIPLICATION RULE FOR INDEPENDENT PROCESSES

If A and B represent events from two different and independent processes, then the probability that both A and B occur can be calculated as the product of their separate probabilities:

$$P(A \text{ and } B) = P(A) \times P(B)$$

Similarly, if there are k events A_1, ..., A_k from k independent processes, then the probability they all occur is

$$P(A_1) \times P(A_2) \times \cdots \times P(A_k)$$

```
> cor(Wage$logwage,Wage$logwage)
[1] 1
> cor(Wage$logwage,Wage$wage)
[1] 0.9506834
> cov(Wage$logwage,Wage$wage)
[1] 13.95427
> cov(Wage$wage,Wage$logwage)
[1] 13.95427
```

https://github.com/jbryer/DATA606Fall2019/blob/master/Textbook/os4.pdf

Joint Probability

Probability
- Joint
- Conditional
- Bayes Theorem
- Summation
- Prior
- Likelihood
- Posterior
- Marginal
- Correlation
- Covariance
- Mutlivariate
- Expected value
- parameter

Note, P(A,B)=P(B,A) always

P(A,B)=P(A)P(B) when A and B are indendent
Otherwise,
P(A,B)=P(A|B)P(B)=P(B|A)P(A)
or
P(A|B)=P(A,B)/P(B) or
P(B|A)=P(A,B)/P(A)

A.2.4 Bayes' Rule

When two random variables are jointly distributed with the value of one known, the probability that the other takes a given value can be computed using *Bayes' rule*:

$$P(y|x) = \frac{P(x|y)P_Y(y)}{P_X(x)} = \frac{P(x|y)P_Y(y)}{\sum_y P(x|y)P_Y(y)} \quad \text{(A.18)}$$

Or, in words

$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{evidence}} \quad \text{(A.19)}$$

Probability Distributions

Probability
- Joint
- Conditional
- Bayes Theorem
- Summation
- Prior
- Likelihood
- Posterior
- Marginal
- Correlation
- Covariance
- Mutlivariate
- Expected value
- parameter

Definition C.2 (Binomial distribution) *A random variable X is said to follow a* binomial distribution *$B(n,p)$ with $n \in \mathbb{N}$ and $p \in [0,1]$ if for any $k \in \{0,1,\ldots,n\}$,*

$$\mathbb{P}[X = k] = \binom{n}{k} p^k (1-p)^{n-k}.$$

Definition C.3 (Normal distribution) *A random variable X is said to follow a* normal *(or* Gaussian*)* distribution *$N(\mu, \sigma^2)$ with $\mu \in \mathbb{R}$ and $\sigma > 0$ if its probability density function is given by,*

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right).$$

The standard normal distribution $N(0,1)$ is the normal distribution with zero mean and unit variance.

The normal distribution is often used to approximate a binomial distribution. Figure C.1 illustrates that approximation.

Definition C.4 (Laplace distribution) *A random variable X is said to follow a* Laplace distribution *with location parameter $\mu \in \mathbb{R}$ and scale parameter $b > 0$ if its probability density function is given by,*

$$f(x) = \frac{1}{2b} \exp\left(-\frac{|x-\mu|}{b}\right).$$

Definition C.5 (Gibbs distributions) *Given a set $\mathcal{X}$ and feature function $\Phi : \mathcal{X} \to \mathbb{R}^N$, a random variable X is said to follow a* Gibbs distribution *with parameter $\mathbf{w} \in \mathbb{R}^N$ if for any $x \in \mathcal{X}$,*

$$\mathbb{P}[X = x] = \frac{\exp(\mathbf{w} \cdot \Phi(x))}{\sum_{x \in \mathcal{X}} \exp(\mathbf{w} \cdot \Phi(x))}.$$

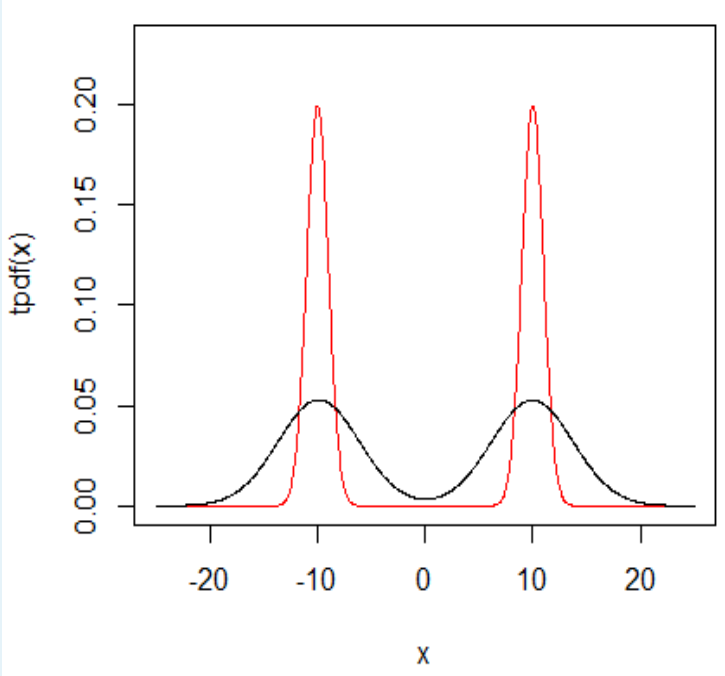
The normalizing quantity in the denominator $Z = \sum_{x \in \mathcal{X}} \exp(\mathbf{w} \cdot \Phi(x))$ is also called the partition function.

Definition C.6 (Poisson distribution) *A random variable X is said to follow a* Poisson distribution *with $\lambda > 0$ if for any $k \in \mathbb{N}$,*

$$\mathbb{P}[X = k] = \frac{\lambda^k e^{-\lambda}}{k!}.$$

Estimating Distribution

```
set.seed(1)
 data = c(rnorm(100,-10,1),rnorm(100,10,1))
 phi = function(x) exp(-.5*x^2)/sqrt(2*pi)
 tpdf = function(x) phi(x+10)/2+phi(x-10)/2
 h = sd(data)*(4/3/length(data))^(1/5)
 Kernel2 = function(x)
mean(phi((x-data)/h)/h)
 kpdf = function(x) sapply(x,Kernel2)
 x=seq(-25,25,length=1000)
 plot(x,tpdf(x),type="l",ylim=c(0,0.23),
col="red")
 par(new=T)
  plot(x,kpdf(x),type="l",ylim=c(0,0.23),
xlab="",ylab="",axes=F)
```



Probability
- Joint
- Conditional
- Bayes Theorem
- Summation
- Prior
- Likelihood
- Posterior
- Marginal
- Correlation
- Covariance
- Mutlivariate
- Expected value
- parameter

https://en.wikipedia.org/wiki/Kernel_density_estimation

Covariance and Correlation

Probability
- Joint
- Conditional
- Bayes Theorem
- Summation
- Prior
- Likelihood
- Posterior
- Marginal
- Correlation
- Covariance
- Mutlivariate
- Expected value
- parameter

Definition C.16 (Covariance) *The covariance of two random variables X and Y is denoted by $\mathrm{Cov}(X,Y)$ and defined by*

$$\mathrm{Cov}(X,Y) = \mathrm{E}\left[(X - \mathrm{E}[X])(Y - \mathrm{E}[Y])\right]. \tag{C.20}$$

Two random variables X and Y are said to be uncorrelated when $\mathrm{Cov}(X,Y) = 0$. It is straightforward to see that if two random variables X and Y are independent then they are uncorrelated, but the converse does not hold in general. The covariance defines a positive semidefinite and symmetric bilinear form:

- symmetry: $\mathrm{Cov}(X,Y) = \mathrm{Cov}(Y,X)$ for any two random variables X and Y;
- bilinearity: $\mathrm{Cov}(X+X',Y) = \mathrm{Cov}(X,Y) + \mathrm{Cov}(X',Y)$ and $\mathrm{Cov}(aX,Y) = a\,\mathrm{Cov}(X,Y)$ for any random variables X, X', and Y and $a \in \mathbb{R}$;
- positive semidefiniteness: $\mathrm{Cov}(X,X) = \mathrm{Var}[X] \geq 0$ for any random variable X.

A **correlation** is a statistic intended to quantify the strength of the relationship between two variables.

Covariance is a measure of the tendency of two variables to vary together.

Think Stats
Exploratory Data Analysis in Python

https://github.com/jbryer/DATA606Fall2019/blob/master/Textbook/os4.pdf

CLT

Probability
- LLN/CLT
- Joint
- Conditional
- Bayes Theorem
- Summation
- Prior
- Likelihood
- Posterior
- Marginal
- Correlation
- Covariance
- Mutlivariate
- Expected value
- parameter

5.1.3 Central Limit Theorem

The distribution in Figure 5.2 looks an awful lot like a normal distribution. That is no anomaly; it is the result of a general principle called the **Central Limit Theorem**.

CENTRAL LIMIT THEOREM AND THE SUCCESS-FAILURE CONDITION

When observations are independent and the sample size is sufficiently large, the sample proportion $\hat{p}$ will tend to follow a normal distribution with the following mean and standard error:

$$\mu_{\hat{p}} = p \qquad SE_{\hat{p}} = \sqrt{\frac{p(1-p)}{n}}$$

In order for the Central Limit Theorem to hold, the sample size is typically considered sufficiently large when $np \geq 10$ and $n(1-p) \geq 10$, which is called the **success-failure condition**.

The Central Limit Theorem is incredibly important, and it provides a foundation for much of statistics. As we begin applying the Central Limit Theorem, be mindful of the two technical conditions: the observations must be independent, and the sample size must be sufficiently large such that $np \geq 10$ and $n(1-p) \geq 10$.

The sampling distribution of the sample mean,
is normal,regardless of the distribution of the underlying data

LLN

Probability
- LLN/CLT
- Joint
- Conditional
- Bayes Theorem
- Summation
- Prior
- Likelihood
- Posterior
- Marginal
- Correlation
- Covariance
- Mutlivariate
- Expected value
- parameter
- statistic

- Theorem: let $(X_n)_{n\in\mathbb{N}}$ be a sequence of independent random variables with the same mean μ and variance $\sigma^2<\infty$ and let $\overline{X}_n=\frac{1}{n}\sum_{i=1}^{n} X_i$, then, for any $\epsilon>0$,

$$\lim_{n\to\infty} \Pr[|\overline{X}_n-\mu|\geq\epsilon]=0.$$

- Proof: Since the variables are independent,

$$\operatorname{Var}[\overline{X}_n]=\sum_{i=1}^{n}\operatorname{Var}\left[\frac{X_i}{n}\right]=\frac{n\sigma^2}{n^2}=\frac{\sigma^2}{n}.$$

For large samples, the sample mean(statistic) is an unbiased statistic, in that it will tend to be the population mean (parameter). the sample variance (statistic) will be 1/n th of the population variance (parameter)

Expectation

Probability
- LLN/CLT
- Joint
- Conditional
- Bayes Theorem
- Summation
- Prior
- Likelihood
- Posterior
- Marginal
- Correlation
- Covariance
- Mutlivariate
- Expected value
- parameter

A.2.5 Expectation

Expectation, *expected value*, or *mean* of a random variable X, denoted by $E[X]$, is the average value of X in a large number of experiments:

$$E[X] = \begin{cases} \sum_i x_i P(x_i) & \text{if } X \text{ is discrete} \\ \int x p(x) dx & \text{if } X \text{ is continuous} \end{cases} \quad \text{(A.20)}$$

It is a weighted average where each value is weighted by the probability that X takes that value. It has the following properties ($a, b \in \Re$):

$$\begin{aligned} E[aX + b] &= aE[X] + b \quad \text{(A.21)} \\ E[X + Y] &= E[X] + E[Y] \end{aligned}$$

For any real-valued function $g(\cdot)$, the expected value is

$$E[g(X)] = \begin{cases} \sum_i g(x_i) P(x_i) & \text{if } X \text{ is discrete} \\ \int g(x) p(x) dx & \text{if } X \text{ is continuous} \end{cases} \quad \text{(A.22)}$$

A special $g(x) = x^n$, called the nth moment of X, is defined as

$$E[X^n] = \begin{cases} \sum_i x_i^n P(x_i) & \text{if } X \text{ is discrete} \\ \int x^n p(x) dx & \text{if } X \text{ is continuous} \end{cases} \quad \text{(A.23)}$$

Mean is the first moment and is denoted by μ.

Variance

Probability
- LLN/CLT
- Joint
- Conditional
- Bayes Theorem
- Summation
- Prior
- Likelihood
- Posterior
- Marginal
- Correlation
- Covariance
- Mutlivariate
- Expected value
- parameter

A.2.6 Variance

Variance measures how much X varies around the expected value. If $\mu \equiv E[X]$, the variance is defined as

$$\text{Var}(X) = E[(X-\mu)^2] = E[X^2] - \mu^2 \quad \text{(A.24)}$$

Variance is the second moment minus the square of the first moment. Variance, denoted by σ^2, satisfies the following property ($a, b \in \Re$):

$$\text{Var}(aX+b) = a^2\text{Var}(X) \quad \text{(A.25)}$$

$\sqrt{\text{Var}(X)}$ is called the *standard deviation* and is denoted by σ. Standard deviation has the same unit as X and is easier to interpret than variance.

As N, the number of points in the sample, gets larger, m deviates less from μ. Let us now check, s^2, the MLE of σ^2:

$$s^2 = \frac{\sum_t (x^t - m)^2}{N} = \frac{\sum_t (x^t)^2 - Nm^2}{N}$$

$$E[s^2] = \frac{\sum_t E[(x^t)^2] - N \cdot E[m^2]}{N}$$

Given that $\text{Var}(X) = E[X^2] - E[X]^2$, we get $E[X^2] = \text{Var}(X) + E[X]^2$, and we can write

$$E[(x^t)^2] = \sigma^2 + \mu^2 \text{ and } E[m^2] = \sigma^2/N + \mu^2$$

Covariance

Probability
- LLN/CLT
- Joint
- Conditional
- Bayes Theorem
- Summation
- Prior
- Likelihood
- Posterior
- Marginal
- Correlation
- Covariance
- Mutlivariate
- Expected value
- parameter

Then, plugging these in, we get

$$E[s^2] = \frac{N(\sigma^2 + \mu^2) - N(\sigma^2/N + \mu^2)}{N} = \left(\frac{N-1}{N}\right)\sigma^2 \neq \sigma^2$$

which shows that s^2 is a biased estimator of σ^2. $(N/(N-1))s^2$ is an unbiased estimator. However when N is large, the difference is negligable. This is an example of an *asymptotically unbiased estimator* whose bias goes to 0 as N goes to infinity.

Covariance indicates the relationship between two random variables. If the occurrence of X makes Y more likely to occur, then the covariance is positive; it is negative if X's occurrence makes Y less likely to happen and is 0 if there is no dependence.

$$\text{(A.26)} \quad \text{Cov}(X,Y) = E[(X-\mu_X)(Y-\mu_Y)] = E[XY] - \mu_X\mu_Y$$

where $\mu_X \equiv E[X]$ and $\mu_Y \equiv E[Y]$. Some other properties are

$$\begin{aligned} \text{Cov}(X,Y) &= \text{Cov}(Y,X) \\ \text{Cov}(X,X) &= \text{Var}(X) \\ \text{Cov}(X+Z,Y) &= \text{Cov}(X,Y) + \text{Cov}(Z,Y) \end{aligned}$$

$$\text{(A.27)} \quad \text{Cov}\left(\sum_i X_i, Y\right) = \sum_i \text{Cov}(X_i, Y)$$

$$\text{(A.28)} \quad \text{Var}(X+Y) = \text{Var}(X) + \text{Var}(Y) + 2\text{Cov}(X,Y)$$

$$\text{(A.29)} \quad \text{Var}\left(\sum_i X_i\right) = \sum_i \text{Var}(X_i) + \sum_i \sum_{j \neq i} \text{Cov}(X_i, X_j)$$

Introduction to Machine Learning – Alpaydin

MLE

Probability

- LLN/CLT
- Joint
- Conditional
- Bayes Theorem
- Summation
- Prior
- Likelihood
- Posterior
- Marginal
- Correlation
- Covariance
- Mutlivariate
- Expected value
- parameter

4.2 Maximum Likelihood Estimation

Let us say we have an independent and identically distributed (iid) sample $\mathcal{X} = \{x^t\}_{t=1}^N$. We assume that x^t are instances drawn from some known probability density family, $p(x|\theta)$, defined up to parameters, θ:

$$x^t \sim p(x|\theta)$$

LIKELIHOOD

We want to find θ that makes sampling x^t from $p(x|\theta)$ as likely as possible. Because x^t are independent, the *likelihood* of parameter θ given sample $\mathcal{X}$ is the product of the likelihoods of the individual points:

$$l(\theta|\mathcal{X}) \equiv p(\mathcal{X}|\theta) = \prod_{t=1}^{N} p(x^t|\theta) \tag{4.1}$$

MAXIMUM LIKELIHOOD ESTIMATION

In *maximum likelihood estimation*, we are interested in finding θ that makes $\mathcal{X}$ the most likely to be drawn. We thus search for θ that maximizes the likelihood, which we denote by $l(\theta|\mathcal{X})$. We can maximize the log of the likelihood without changing the value where it takes its maximum. $\log(\cdot)$ converts the product into a sum and leads to further computational simplification when certain densities are assumed, for example, containing exponents. The *log likelihood* is defined as

LOG LIKELIHOOD

$$\mathcal{L}(\theta|\mathcal{X}) \equiv \log l(\theta|\mathcal{X}) = \sum_{t=1}^{N} \log p(x^t|\theta) \tag{4.2}$$

Introduction to Machine Learning – Alpaydin

Bias and Variance

Probability
- LLN/CLT
- Joint
- Conditional
- Bayes Theorem
- Summation
- Prior
- Likelihood
- Posterior
- Marginal
- Correlation
- Covariance
- Mutlivariate
- Expected value
- parameter
- statistic

4.3 Evaluating an Estimator: Bias and Variance

Let X be a sample from a population specified up to a parameter θ, and let $d = d(X)$ be an estimator of θ. To evaluate the quality of this estimator, we can measure how much it is different from θ, that is, $(d(X) - \theta)^2$. But since it is a random variable (it depends on the sample), we need to

MEAN SQUARE ERROR

average this over possible X and consider $r(d, \theta)$, the *mean square error* of the estimator d defined as

$$r(d, \theta) = E[(d(X) - \theta)^2] \tag{4.9}$$

BIAS

The *bias* of an estimator is given as

$$b_\theta(d) = E[d(X)] - \theta \tag{4.10}$$

UNBIASED ESTIMATOR

If $b_\theta(d) = 0$ for all θ values, then we say that d is an *unbiased estimator* of θ. For example, with x^t drawn from some density with mean μ, the sample average, m, is an unbiased estimator of the mean, μ, because

$$E[m] = E\left[\frac{\sum_t x^t}{N}\right] = \frac{1}{N}\sum_t E[x^t] = \frac{N\mu}{N} = \mu$$

This means that though on a particular sample, m may be different from μ, if we take many such samples, X_i, and estimate many $m_i = m(X_i)$, *their* average will get close to μ as the number of such samples increases. m is also a *consistent* estimator, that is, $\text{Var}(m) \to 0$ as $N \to \infty$.

Introduction to Machine Learning – Alpaydin

Error ~ f(Bias , Variance)

Probability
- LLN/CLT
- Joint
- Conditional
- Bayes Theorem
- Summation
- Prior
- Likelihood
- Posterior
- Marginal
- Correlation
- Covariance
- Mutlivariate
- Expected value
- parameter
- statistic

The mean square error can be rewritten as follows—d is short for $d(X)$:

$$\begin{aligned} r(d,\theta) &= E\left[(d-\theta)^2\right] \\ &= E\left[(d-E[d]+E[d]-\theta)^2\right] \\ &= E\left[(d-E[d])^2+(E[d]-\theta)^2+2(E[d]-\theta)(d-E[d])\right] \\ &= E\left[(d-E[d])^2\right]+E\left[(E[d]-\theta)^2\right]+2E\left[(E[d]-\theta)(d-E[d])\right] \\ &= E\left[(d-E[d])^2\right]+(E[d]-\theta)^2+2(E[d]-\theta)E[d-E[d]] \\ &= \underbrace{E\left[(d-E[d])^2\right]}_{variance}+\underbrace{(E[d]-\theta)^2}_{bias^2} \end{aligned} \tag{4.11}$$

VARIANCE

The two equalities follow because $E[d]$ is a constant and therefore $E[d]-\theta$ also is a constant, and because $E[d-E[d]] = E[d]-E[d] = 0$. In equation 4.11, the first term is the *variance* that measures how much, on average, d_i vary around the expected value (going from one dataset to another), and the second term is the *bias* that measures how much the expected value varies from the correct value θ (figure 4.1). We then write error as the sum of these two terms, the variance and the square of the bias:

$$r(d,\theta) = \mathrm{Var}(d) + (b_\theta(d))^2 \tag{4.12}$$

Introduction to Machine Learning – Alpaydin

Classification

Classification
- Categorical
- Label
- Class
- Separation Margin
- hyperplane – Orion
- Linear boundary
- NonLinear
- TP,TN,FP,FN
- Confusion Matrix

- **The learner's input:** In the basic statistical learning setting, the learner has access to the following:
  - **Domain set:** An arbitrary set, $\mathcal{X}$. This is the set of objects that we may wish to label. For example, in the papaya learning problem mentioned before, the domain set will be the set of all papayas. Usually, these domain points will be represented by a vector of *features* (like the papaya's color and softness). We also refer to domain points as *instances* and to $\mathcal{X}$ as instance space.
  - **Label set:** For our current discussion, we will restrict the label set to be a two-element set, usually $\{0, 1\}$ or $\{-1, +1\}$. Let $\mathcal{Y}$ denote our set of possible labels. For our papayas example, let $\mathcal{Y}$ be $\{0, 1\}$, where 1 represents being tasty and 0 stands for being not-tasty.
  - **Training data:** $S = ((x_1, y_1) \ldots (x_m, y_m))$ is a finite sequence of pairs in $\mathcal{X} \times \mathcal{Y}$: that is, a sequence of labeled domain points. This is the input that the learner has access to (like a set of papayas that have been

Understanding Machine Learning, © 2014 by Shai Shalev-Shwartz and Shai Ben-David
Published 2014 by Cambridge University Press.
Personal use only. Not for distribution. Do not post.
Please link to http://www.cs.huji.ac.il/~shais/UnderstandingMachineLearning

- **The learner's output:** The learner is requested to output a *prediction rule*, $h : \mathcal{X} \to \mathcal{Y}$. This function is also called a *predictor*, a *hypothesis*, or a *classifier*. The predictor can be used to predict the label of new domain points.

Regression is also Supervised Learning, appropriate when the dependent variable (DV) is a Real/continuous variable. Called classification when the DV is categorical. The domain of DV values is the set of labels. In both Regression and Classification, we are seeking to determine the DV given independent variables (IV).

Classification

Classification
- Categorical
- Label
- Class
- Separation Margin
- hyperplane – Orion
- Linear boundary
- NonLinear
- TP,TN,FP,FN
- Confusion Matrix

Classification

R. J. Henery
University of Strathclyde[1]

2.1 DEFINITION OF CLASSIFICATION

Classification has two distinct meanings. We may be given a set of observations with the aim of establishing the existence of classes or clusters in the data. Or we may know for certain that there are so many classes, and the aim is to establish a rule whereby we can classify a new observation into one of the existing classes. The former type is known as Unsupervised Learning (or Clustering), the latter as Supervised Learning. In this book when we use the term classification, we are talking of Supervised Learning. In the statistical literature, Supervised Learning is usually, but not always, referred to as discrimination, by which is meant the establishing of the classification rule from given correctly classified data.

http://www1.maths.leeds.ac.uk/~charles/statlog/

Probabilistic Classification

Probability
- Joint
- Conditional
- Bayes Theorem
- Summation
- Prior
- Likelihood
- Posterior
- Marginal
- Correlation
- Covariance
- Mutlivariate
- Expected value
- parameter
- statistic

By definition,Classification → determine the class given data. Because, P(C|D), can be computed from either P(C,D) or P(D|C)

Generative Classifiers compute P(C|D) using P(C,D)/P(D) – Joint Probability

Discriminative Classifiers compute P(C|D) using P(D|C) P(C)/P(D) – Conditional Probability

parametric classifiers assume there exists an underlying probability, and try to estimate that probability, aka probabilistic classifiers.

non-parametric classifiers make such an assumption and attempt to determine the classes without assuming any distributional property, directly from the given data, aka instance based (kNN), logic/rule based classifiers (decision Trees).

Classification: Loss Function

We will consider the following spaces: a space of examples $\mathcal{X}$ a space of labels $\mathcal{Y}$ and a space of hypothesis $\mathcal{H}$ that contains functions mapping $\mathcal{X}$ to $\mathcal{Y}$. We will also consider a loss function $L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ and a probability measure P over the space $\mathcal{X} \times \mathcal{Y}$.

Definition 1. *For any hypothesis $h \in \mathcal{H}$ we define the risk of h with respect to L and P to be:*

$$\mathcal{L}_P(h) = \mathop{\mathrm{E}}_{P}(L(h(x), y))$$

The general problem of machine learning can be then cast as finding the hypothesis $h \in \mathcal{H}$ that solves the following optimization problem:

$$\min_{h \in \mathcal{H}} \mathcal{L}_P(h)$$

By having access only to a labeled sample $S = (x_i, y_i)_{i=1}^n \in \mathcal{X} \times \mathcal{Y}$. A simplification of this scenario is given when the distribution P is given only over the example space $\mathcal{X}$ and there exists a labeling function $f : \mathcal{X} \to \mathcal{Y}$ in that case we are trying to find a hypothesis h such that it solves the optimization problem:

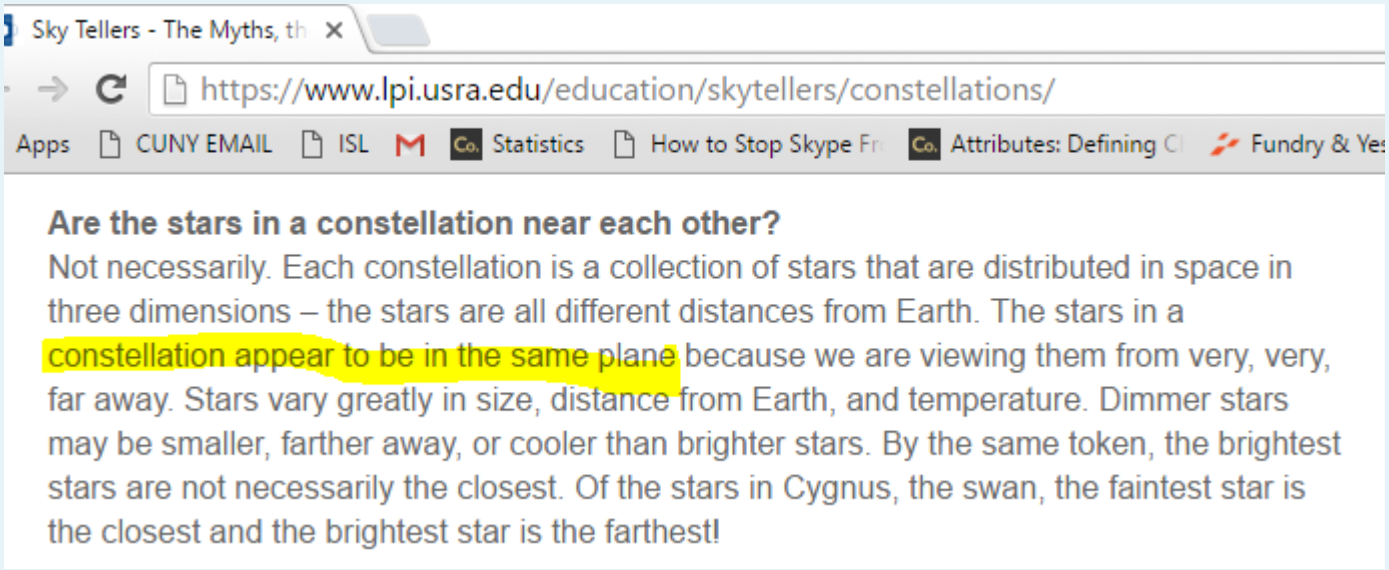
$$\min_{h \in \mathcal{H}} \mathop{\mathrm{E}}_{P}(L(h(x), f(x))$$

https://cims.nyu.edu/~munoz/files/ml_optimization.pdf

- Loss function: $L : Y \times Y \to \mathbb{R}$.
  - $L(\widehat{y}, y)$: cost of predicting $\widehat{y}$ instead of y.
  - binary classification: 0-1 loss, $L(y, y') = 1_{y \neq y'}$.
  - regression: $Y \subseteq \mathbb{R}$, $l(y, y') = (y' - y)^2$.

http://jeffreyheinz.net/classes/18S/materials/Mohri Rostamizadeh Talwalk2012 Foundations of Machine Learning Chap 1.pdf

Hyperplane: Constellation



Do such patterns exist? Several forces are at work here:
Our brain makes up known patterns when none exists – desert mirage
Star light though coming from deep space, are projected on a two dimensional space that patterns begin to emerge. Reduced dimension from high dimension, yields additional patterns – hyperplane...
Lastly, when there are gazillions of stars blinking, patterns emerge randomly – Bon Ferroni

hyperplane

9.1.1 What Is a Hyperplane?

In a p-dimensional space, a *hyperplane* is a flat affine subspace of dimension $p - 1$.[1] For instance, in two dimensions, a hyperplane is a flat one-dimensional subspace—in other words, a line. In three dimensions, a hyperplane is a flat two-dimensional subspace—that is, a plane. In $p > 3$ dimensions, it can be hard to visualize a hyperplane, but the notion of a $(p - 1)$-dimensional flat subspace still applies.

The mathematical definition of a hyperplane is quite simple. In two dimensions, a hyperplane is defined by the equation

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 = 0 \tag{9.1}$$

for parameters β_0, β_1, and β_2. When we say that (9.1) "defines" the hyperplane, we mean that any $X = (X_1, X_2)^T$ for which (9.1) holds is a point on the hyperplane. Note that (9.1) is simply the equation of a line, since indeed in two dimensions a hyperplane is a line.

ISLR

Separation Margin

342 9. Support Vector Machines

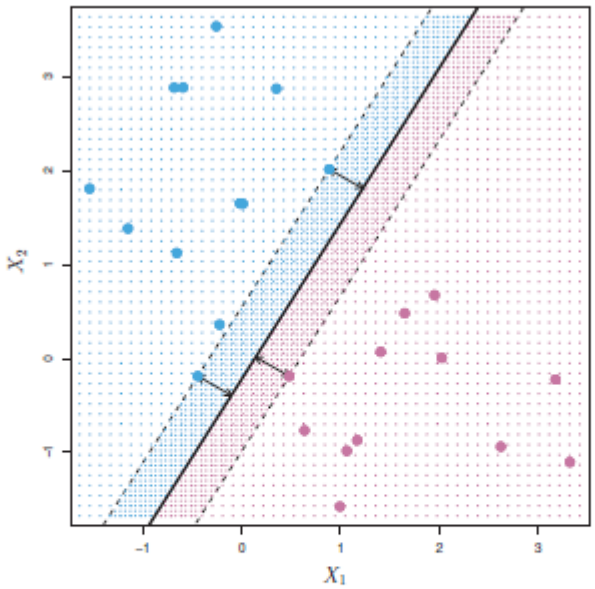
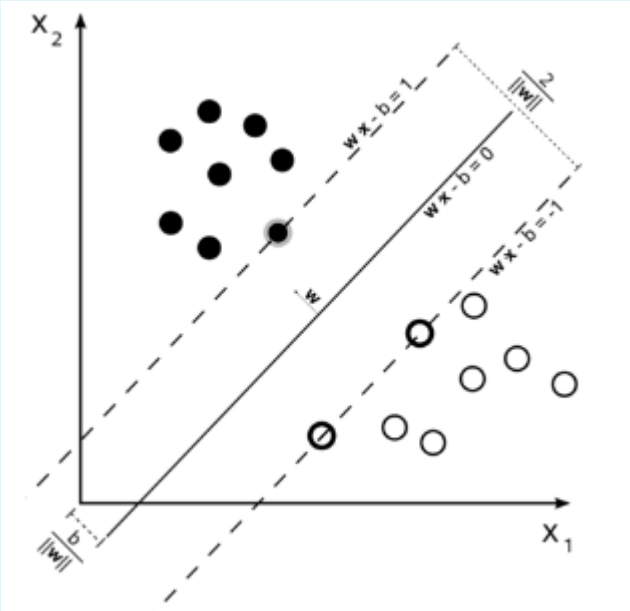


FIGURE 9.3. *There are two classes of observations, shown in blue and in purple. The maximal margin hyperplane is shown as a solid line. The margin is the distance from the solid line to either of the dashed lines. The two blue points and the purple point that lie on the dashed lines are the support vectors, and the distance from those points to the hyperplane is indicated by arrows. The purple and blue grid indicates the decision rule made by a classifier based on this separating hyperplane.*

When linearly separable
– Maximum Margin
Otherwise
– SVMs implement
Soft Max

ISLR
https://en.wikipedia.org/wiki/Support-vector_machine

Separation Margin



https://math.stackexchange.com/questions/1305925/why-is-the-svm-margin-equal-to-frac2-mathbfw

Let $\mathbf{x}_0$ be a point in the hyperplane $\mathbf{w}\mathbf{x} - b = -1$, i.e., $\mathbf{w}\mathbf{x}_0 - b = -1$. To measure the distance between hyperplanes $\mathbf{w}\mathbf{x} - b = -1$ and $\mathbf{w}\mathbf{x} - b = 1$, we only need to compute the perpendicular distance from $\mathbf{x}_0$ to plane $\mathbf{w}\mathbf{x} - b = 1$, denoted as r.

Note that $\frac{\mathbf{w}}{\|\mathbf{w}\|}$ is a unit normal vector of the hyperplane $\mathbf{w}\mathbf{x} - b = 1$. We have

$$\mathbf{w}(\mathbf{x}_0 + r\frac{\mathbf{w}}{\|\mathbf{w}\|}) - b = 1$$

since $\mathbf{x}_0 + r\frac{\mathbf{w}}{\|\mathbf{w}\|}$ should be a point in hyperplane $\mathbf{w}\mathbf{x} - b = 1$ according to our definition of r.

Expanding this equation, we have

$$\begin{aligned}
&\mathbf{w}\mathbf{x}_0 + r\frac{\mathbf{w}\mathbf{w}}{\|\mathbf{w}\|} - b = 1 \\
\Longrightarrow\ &\mathbf{w}\mathbf{x}_0 + r\frac{\|\mathbf{w}\|^2}{\|\mathbf{w}\|} - b = 1 \\
\Longrightarrow\ &\mathbf{w}\mathbf{x}_0 + r\|\mathbf{w}\| - b = 1 \\
\Longrightarrow\ &\mathbf{w}\mathbf{x}_0 - b = 1 - r\|\mathbf{w}\| \\
\Longrightarrow\ &-1 = 1 - r\|\mathbf{w}\| \\
\Longrightarrow\ &r = \frac{2}{\|\mathbf{w}\|}
\end{aligned}$$

https://math.stackexchange.com/questions/1305925

TP,FP,TN,FN

T. Fawcett / Pattern Recognition Letters 27 (2006) 861–874

Hypothesized class	True class: p	True class: n		
Y	True Positives	False Positives	fp rate $= \frac{FP}{N}$	tp rate $= \frac{TP}{P}$
N	False Negatives	True Negatives	precision $= \frac{TP}{TP+FP}$	recall $= \frac{TP}{P}$
Column totals:	P	N	accuracy $= \frac{TP+TN}{P+N}$	
			F-measure $= \frac{2}{1/\text{precision}+1/\text{recall}}$	

Fig. 1. Confusion matrix and common performance metrics calculated from it.

Given a classifier and an instance, there are four possible outcomes. If the instance is positive and it is classified as positive, it is counted as a *true positive*; if it is classified as negative, it is counted as a *false negative*. If the instance is negative and it is classified as negative, it is counted as a *true negative*; if it is classified as positive, it is counted as a *false positive*. Given a classifier and a set of instances (the test set), a two-by-two *confusion matrix* (also called a contingency table) can be constructed representing the dispositions of the set of instances. This matrix forms the basis for many common metrics.

The **true positive rate**[1] (also called *hit rate* and *recall*) of a classifier is estimated as

$$tp\ rate \approx \frac{\text{Positives correctly classified}}{\text{Total positives}}$$

The **false positive rate** (also called *false alarm rate*) of the classifier is

$$fp\ rate \approx \frac{\text{Negatives incorrectly classified}}{\text{Total negatives}}$$

Additional terms associated with ROC curves are

$$\text{sensitivity} = \text{recall}$$

$$\text{specificity} = \frac{\text{True negatives}}{\text{False positives} + \text{True negatives}} = 1 - fp\ rate$$

$$\text{positive predictive value} = \text{precision}$$

ROC

Algorithm 2. Calculating the area under an ROC curve
Inputs: L, the set of test examples; $f(i)$, the probabilistic classifier's estimate that example i is positive; P and N, the number of positive and negative examples.
Outputs: A, the area under the ROC curve.
Require: $P > 0$ and $N > 0$

```
1: L_sorted ← L sorted decreasing by f scores
2: FP ← TP ← 0
3: FP_prev ← TP_prev ← 0
4: A ← 0
5: f_prev ← −∞
6: i ← 1
7: while i ≤ |L_sorted| do
8:    if f(i) ≠ f_prev then
9:       A ← A + TRAPEZOID_AREA(FP, FP_prev,
             TP, TP_prev)
10:      f_prev ← f(i)
11:      FP_prev ← FP
12:      TP_prev ← TP
13:   end if
14:   if i is a positive example then
15:      TP ← TP + 1
16:   else      /* i is a negative example */
17:      FP ← FP + 1
18:   end if
19:   i ← i + 1
20: end while
21: A ← A + TRAPEZOID_AREA(N, FP_prev, N, TP_prev)
22: A ← A/(P × N)   /* scale from P × N onto the unit
    square */
23: end
1: function TRAPEZOID_AREA(X1, X2, Y1, Y2)
2: Base ← |X1 − X2|
3: Height_avg ← (Y1 + Y2)/2
4: return Base × Height_avg
5: end function
```

classifiers simply by graphing them in ROC space and seeing which ones dominate. This is misleading; it is analogous to taking the maximum of a set of accuracy figures from a single test set. Without a measure of variance we cannot compare the classifiers.

Averaging ROC curves is easy if the original instances are available. Given test sets $T_1, T_2, \ldots, T_n$, generated from cross-validation or the bootstrap method, we can simply merge sort the instances together by their assigned scores into one large test set T_M. We then run an ROC curve generation algorithm such as algorithm 1 on T_M and plot the result. However, the primary reason for using multiple test sets is to derive a measure of variance, which this simple merging does not provide. We need a more sophisticated method that samples individual curves at different points and averages the samples.

ROC space is two-dimensional, and any average is necessarily one-dimensional. ROC curves can be projected onto a single dimension and averaged conventionally, but this leads to the question of whether the projection is appropriate, or more precisely, whether it preserves characteristics of interest. The answer depends upon the reason for averaging the curves. This section presents two methods for averaging ROC curves: vertical and threshold averaging.

Fig. 9a shows five ROC curves to be averaged. Each contains a thousand points and has some concavities. Fig. 9b shows the curve formed by merging the five test sets and computing their combined ROC curve. Fig. 9c and d shows average curves formed by sampling the five individual ROC curves. The error bars are 95% confidence intervals.

8.1. Vertical averaging

Vertical averaging takes vertical samples of the ROC curves for fixed FP rates and averages the corresponding

Confusion Matrix

		True default status		
		No	Yes	Total
Predicted default status	No	9,644	252	9,896
	Yes	23	81	104
	Total	9,667	333	10,000

TABLE 4.4. *A confusion matrix compares the LDA predictions to the true default statuses for the 10,000 training observations in the* `Default` *data set. Elements on the diagonal of the matrix represent individuals whose default statuses were correctly predicted, while off-diagonal elements represent individuals that were misclassified. LDA made incorrect predictions for 23 individuals who did not default and for 252 individuals who did default.*

False Negative is Type I error -- reject when it should not be rejected
False Positive is Type II error – NOT rejecting when it should be rejected

– The *p value* is the probability of observing a sample value as extreme as, or more extreme than, the value actually observed, given that the null hypothesis is true.
– *p value* represents the risk of rejecting a true null hypothesis.
– *p-value* is the probability of a Type I error if the null hypothesis is rejected.

Overfitting:

ISLR 6th Edition
When a given method yields a small training MSE but a large test MSE, we are said to be *overfitting* the data. This happens because our statistical learning procedure is working too hard to find patterns in the training data, and may be picking up some patterns that are just caused by random chance rather than by true properties of the unknown function f. When we overfit the training data, the test MSE will be very large because the supposed patterns that the method found in the training data simply don't exist in the test data.

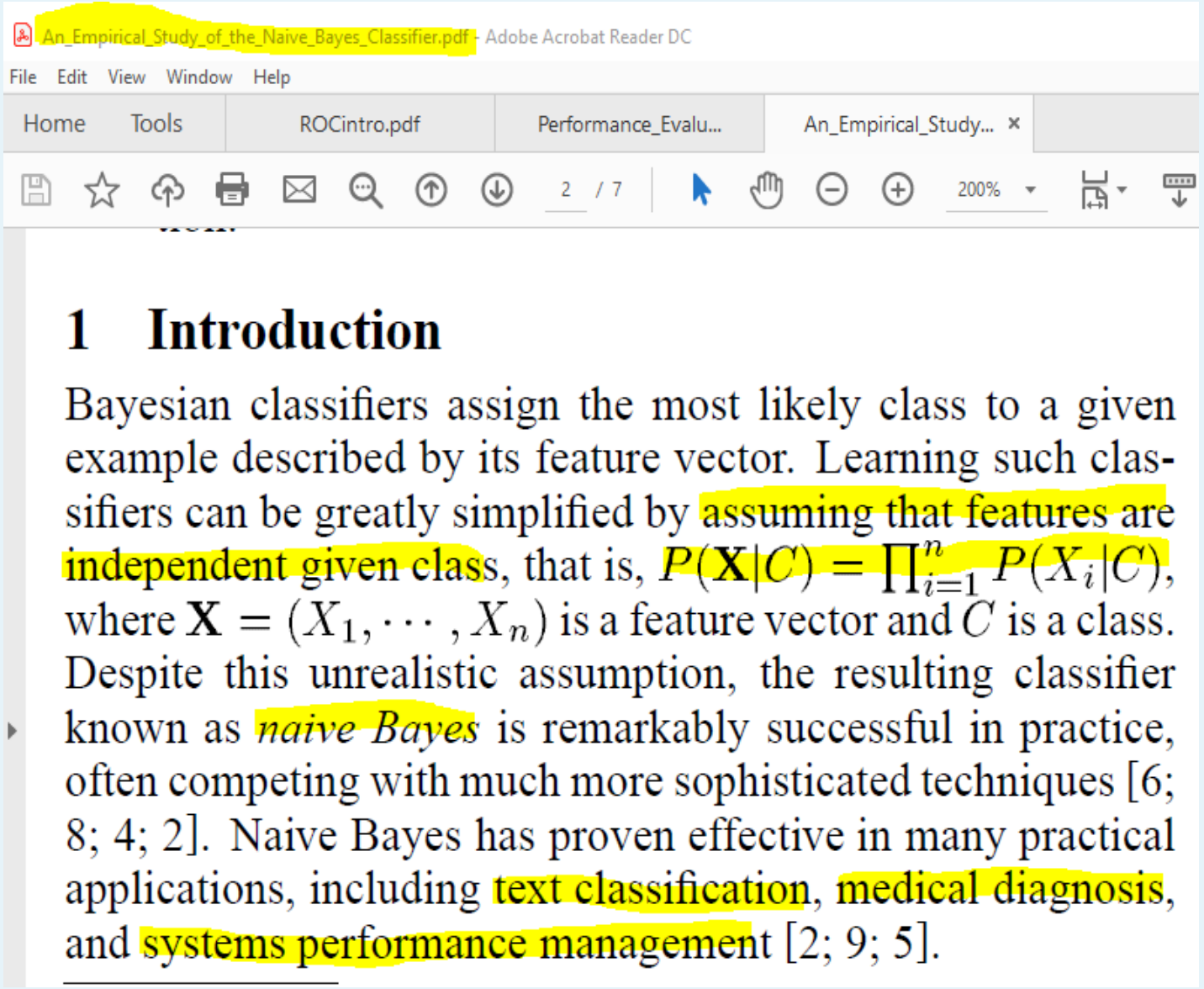
Overfitting refers specifically to the case in which a less flexible model would have yielded a smaller test MSE. Indulge in Occam's Razor – choose a simpler model that does not capture all the noise in the training set.

Expect the error in training set to be ALWAYS lesser than the error in test set.

That more complex models result in lower training error is a fundamental property of statistical learning that holds regardless of the particular data set at hand
and regardless of the statistical method being used. What matters is the error in test or online data – never seen before data.

Occam's Razor → Simplest Model that produces acceptable results.

NB



1 Introduction

Bayesian classifiers assign the most likely class to a given example described by its feature vector. Learning such classifiers can be greatly simplified by assuming that features are independent given class, that is, $P(\mathbf{X}|C) = \prod_{i=1}^{n} P(X_i|C)$, where $\mathbf{X} = (X_1, \cdots, X_n)$ is a feature vector and C is a class. Despite this unrealistic assumption, the resulting classifier known as *naive Bayes* is remarkably successful in practice, often competing with much more sophisticated techniques [6; 8; 4; 2]. Naive Bayes has proven effective in many practical applications, including text classification, medical diagnosis, and systems performance management [2; 9; 5].

NB:Definition

2 Definitions and Background

Let $\mathbf{X} = (X_1, ..., X_n)$ be a vector of observed random variables, called *features*, where each feature takes values from its *domain* D_i. The set of all feature vectors (*examples*, or *states*), is denoted $\Omega = D_1 \times ... \times D_n$. Let C be an unobserved random variable denoting the *class* of an example, where C can take one of m values $c \in \{0, ..., m-1\}$. Capital letters, such as X_i, will denote variables, while lower-case letters, such as x_i, will denote their values; boldface letters will denote vectors.

A function $g : \Omega \rightarrow \{0, ..., m-1\}$, where $g(\mathbf{x}) = C$, denotes a *concept* to be learned. Deterministic $g(x)$ corresponds to a concept without noise, which always assigns the same class to a given example (e.g., disjunctive and conjunctive concepts are deterministic). In general, however, a concept can be *noisy*, yielding a random function $g(x)$.

A classifier is defined by a (deterministic) function $h : \Omega \rightarrow \{0, ..., m-1\}$ (a *hypothesis*) that assigns a class to any given example. A common approach is to associate each class i with a discriminant function $f_i(\mathbf{x})$, $i = 0, ..., m-1$, and let the classifier select the class with maximum discriminant function on a given example: $h(\mathbf{x}) = \arg\max_{i \in \{0,...,m-1\}} f_i(\mathbf{x})$.

An_Empirical_Study_of_the_Naive_Bayes_Classifier.pdf - Adobe Acrobat Reader DC
File Edit View Window Help
Home Tools ROCintro.pdf Performance_Evalu...
3 / 7

$\arg\max_{i \in \{0,...,m-1\}} f_i(\mathbf{x})$.

The *Bayes* classifier $h^*(\mathbf{x})$ (that we also call *Bayes-optimal* classifier and denote $BO(\mathbf{x})$), uses as discriminant functions the class posterior probabilities given a feature vector, i.e. $f_i^*(\mathbf{x}) = P(C = i|\mathbf{X} = \mathbf{x})$. Applying Bayes rule gives $P(C = i|\mathbf{X} = \mathbf{x}) = \frac{P(\mathbf{X}=\mathbf{x}|C=i)P(C=i)}{P(\mathbf{X}=\mathbf{x})}$, where $P(\mathbf{X} = \mathbf{x})$ is identical for all classes, and therefore can be ignored. This yields Bayes discriminant functions

$$f_i^*(\mathbf{x}) = P(\mathbf{X} = \mathbf{x}|C = i)P(C = i), \quad (1)$$

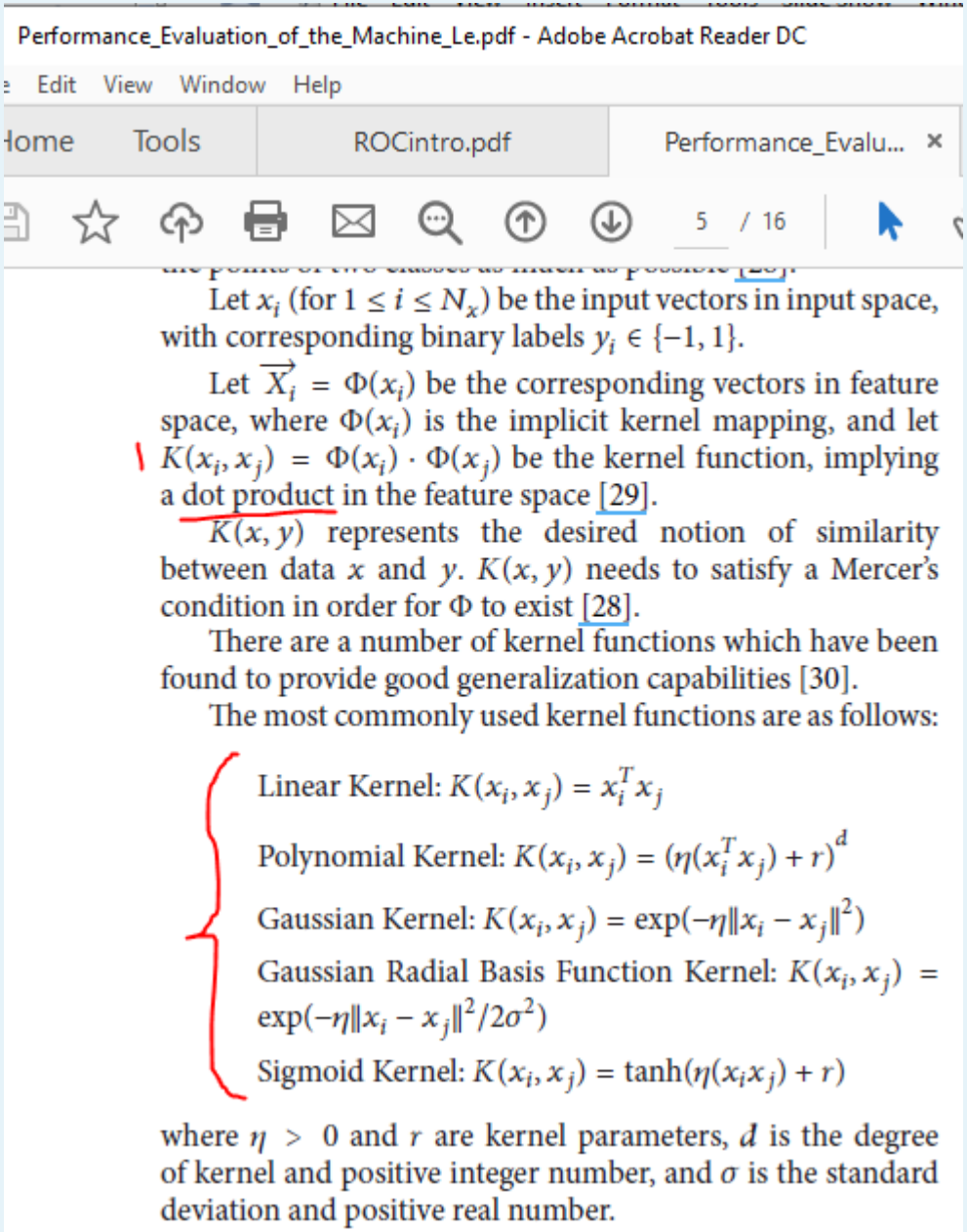
where $P(\mathbf{X} = \mathbf{x}|C = i)$ is called the *class-conditional probability distribution (CPD)*. Thus, the Bayes classifier

$$h^*(\mathbf{x}) = \arg\max_i P(\mathbf{X} = \mathbf{x}|C = i)P(C = i) \quad (2)$$

finds the *maximum a posteriori probability* (MAP) hypothesis given example $\mathbf{x}$. However, direct estimation of $P(\mathbf{X} = \mathbf{x}|C = i)$ from a given set of training examples is hard when the feature space is high-dimensional. Therefore, approximations are commonly used, such as using the simplifying assumption that features are independent given the class. This yields the *naive Bayes* classifier $NB(\mathbf{x})$ defined by discriminant functions

$$f_i^{NB}(\mathbf{x}) = \Pi_{j=1}^n P(X_j = x_j|C = i)P(C = i). \quad (3)$$

Kernel Trick



Performance_Evaluation_of_the_Machine_Le.pdf - Adobe Acrobat Reader DC

Edit View Window Help

Home Tools ROCintro.pdf Performance_Evalu... ×

5 / 16

Let x_i (for $1 \le i \le N_x$) be the input vectors in input space, with corresponding binary labels $y_i \in \{-1, 1\}$.

Let $\overrightarrow{X_i} = \Phi(x_i)$ be the corresponding vectors in feature space, where $\Phi(x_i)$ is the implicit kernel mapping, and let $K(x_i, x_j) = \Phi(x_i) \cdot \Phi(x_j)$ be the kernel function, implying a dot product in the feature space [29].

$K(x, y)$ represents the desired notion of similarity between data x and y. $K(x, y)$ needs to satisfy a Mercer's condition in order for Φ to exist [28].

There are a number of kernel functions which have been found to provide good generalization capabilities [30].

The most commonly used kernel functions are as follows:

Linear Kernel: $K(x_i, x_j) = x_i^T x_j$

Polynomial Kernel: $K(x_i, x_j) = (\eta(x_i^T x_j) + r)^d$

Gaussian Kernel: $K(x_i, x_j) = \exp(-\eta\|x_i - x_j\|^2)$

Gaussian Radial Basis Function Kernel: $K(x_i, x_j) = \exp(-\eta\|x_i - x_j\|^2/2\sigma^2)$

Sigmoid Kernel: $K(x_i, x_j) = \tanh(\eta(x_i x_j) + r)$

where $\eta > 0$ and r are kernel parameters, d is the degree of kernel and positive integer number, and σ is the standard deviation and positive real number.

Instance Based Learning

6 Instance-based learning

Another category under the header of statistical methods is Instance-based learning. Instance-based learning algorithms are lazy-learning algorithms (Mitchell, 1997), as they delay the induction or generalization process until classification is performed. Lazy-learning algorithms require less computation time during the training phase than eager-learning algorithms (such as decision trees, neural and Bayes nets) but more computation time during the classification process. One of the most straightforward instance-based learning algorithms is the *nearest neighbour* algorithm. Aha (1997) and De Mantaras and Armengol (1998) presented a review of instance-based learning classifiers. Thus, in

Supervised Machine Learning: A Review of Classification Techniques
S. B. Kotsiantis

In general, instances can be considered as points within an n-dimensional instance space where each of the n-dimensions corresponds to one of the n-features that are used to describe an instance. The absolute position of the instances within this space is not as significant as the relative distance between instances. This relative distance is determined by using a distance metric. Ideally, the distance metric must minimize the distance between two similarly classified instances, while maximizing the distance between instances of different classes. Many different metrics have been presented. The most significant ones are presented in Table 3.

Minkowsky: $D(x,y)=\left(\sum_{i=1}^{m}\lvert x_i-y_i\rvert^{r}\right)^{1/r}$
Manhattan: $D(x,y)=\sum_{i=1}^{m}\lvert x_i-y_i\rvert$
Chebychev: $D(x,y)=\max_{i=1}^{m}\lvert x_i-y_i\rvert$
Euclidean: $D(x,y)=\left(\sum_{i=1}^{m}\lvert x_i-y_i\rvert^{2}\right)^{1/2}$
Camberra: $D(x,y)=\sum_{i=1}^{m}\frac{\lvert x_i-y_i\rvert}{\lvert x_i+y_i\rvert}$
Kendall's Rank Correlation: $D(x,y)=1-\frac{2}{m(m-1)}\sum_{i=j}^{m}\sum_{j=1}^{i-1}sign(x_i-x_j)sign(y_i-y_j)$

References

– An introduction to ROC analysis,Tom Fawcett
– An empirical study of the naive Bayes classifier, I. Rish
– An Analysis of Bayesian Classifiers, Pat Langley, Wayne Iba, Kevin Thompson
– Performance Evaluation of the Machine Learning Algorithms Used in Inference Mechanism of a Medical Decision Support System, Mert Bal, M. Fatih Amasyali, Hayri Sever, Guven Kose, and Ayse Demirhan
– Supervised Machine Learning: A Review of Classification Techniques, S. B. Kotsiantis
– OpenIntro Statistics,Fourth Edition,David Diez, Mine Cetinkaya-Rundel, Christopher D Barr
– https://en.wikipedia.org/wiki/Support-vector_machine
– http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf
– https://math.stackexchange.com/questions/1305925/why-is-the-svm-margin-equal-to-frac2-mathbfw
– Macihne Learning, Ethem Alpaydin