Machine Learning (Gopakumar) T2.

(After T1 Module 2)

Discriminant function

A classifier can be stated as

$$c(x) = \arg\max_{c} g_c(x)$$

where for each class c, a discriminant function $g_c(\cdot)$ is used to measure up to which degree objects belong to c.

Discriminant function analysis (DFA)

it is a statistical procedure that classifies unknown individuals and the probability of their classification into a certain group.

> Generative vs Discriminant approach

- Generative approches estimate discriminant function by first estimating the probability distribution of data belonging to each class.
- Discriminative approaches estimate the discriminant function explicitly, without assuming probability distribution.

Linear Discriminants

(case of 2 categories)

- Linear discriminant analysis (LDA) or normal discriminant analysis, or discriminant function analysis is generalization of Fisher's linear discriminant, a method used to find a linear combination of features that characterizes or seperates two or more classes of objects or events.

$$LD = \alpha_1 X_1 + \alpha_2 X_2$$

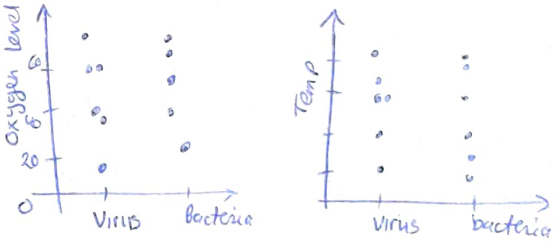
α_1, α_2 = weights

X_1, X_2 are features

eg a person can have either bacterial or viral infection we have his oxygen level & temp.

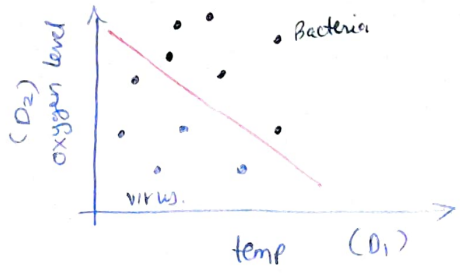
→ also we have old dataset for this.

Now

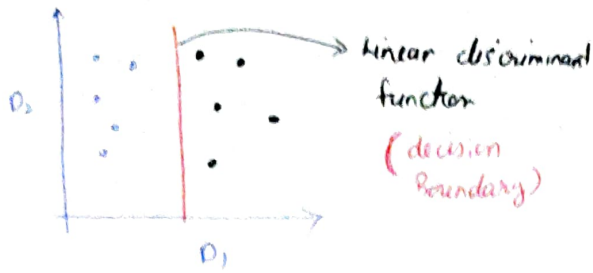


alone both features can't be used for analysis

so LD combines them



we straight, rotate curve to get



So only D_1 can be used

now on basis of D_1 we find discriminant scores

Let got (oxygen)

$LD = 0.11\ CRP + 0.7\ temp$

to get score with center at 0.

$LD = 0.11\ (CRP - \overline{CRP}) + 0.70\ (Temp - \overline{Temp})$ (mean)

⇒ So for viral $LD < 0$

for bacteria $LD > 0$

- we need to find α_1, α_2.

⇒ class definition

A linear discriminant has following form:

$$g(x) = \omega^t x + \omega_0$$
$$= \sum_{i=1}^{d} \omega_i x_i + \omega_0$$

[Decide ω_1 if $g(x) > 0$ & ω_2 if $g(x) < 0$]

if $g(x) = 0$, then x lies on decision boundary & can be assigned to either class.

So end we need to find

ω^t (weight vector), ω_0 (mean)

Best Boundary

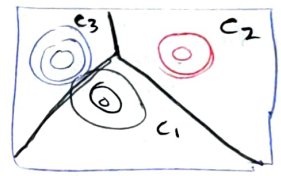
→ Assume 2 classes can be seperated by a linear boundary $I(\theta)$ with some unknown parameters θ.

→ fit the best boundary to data by optimizing over parameters θ.

2 methods

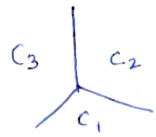
1. Parametric methods

- Assume shape of density for classes is known $p_1(x|\theta_1)$, $p_2(x|\theta_2)$ - - -
- Estimate $\theta_1, \theta_2, - -$ from data
- Use Bayesian classifier to find decision regions



2) Discriminant Functions

- Assume discriminant functions are of known shape $I(\theta_1)$, $I(\theta_2)$ with parameters θ_1, θ_2 - -
- Estimate θ_1, θ_2 - - from data.
- Use discriminant functions for classification



⇒ Estimating accurate density functions is much harder than estimating accurate discriminant functions

⇒ Discriminant functions can be more general than linear

→ Linear discriminant functions are optimal for Gaussian distributions with equal covariance

Decision boundary

$g(x) = \omega^t x + \omega_0 = 0$ is a hyperplane

→ set of vectors x for which for some scalars $\alpha_0, \ldots, \alpha_d$ satisfy $\alpha_0 + \alpha_1 x^{(1)} + \ldots + \alpha_d x^{(d)} = 0$

- A hyperplane is
  - a point in 1D
  - a line in 2D
  - a plane in 3D

Orientation of hyperplane is determined by ω & its location by ω_0

→ ω is normal to hyperplane.

→ if $\omega_0 = 0$, it passes through origin

- use "learning" algorithms to estimate ω & ω_0 from training data x_k.
  - The solution can be found by minimizing an error function, eg the "training error" or "emperical risk".

LDF (Many classes)

- Suppose we have m classes
- Define m linear discriminant functions

$$g_i(x) = \omega_i{}^t x + \omega_0 \qquad i = 1 \ldots m$$

- Given x, assign class c_i if

$$g_i(x) \geq g_j(x)$$

Such classifier is called **linear machine**

→ A linear machine devides the feature space into c decision regions, with $g_i(x)$ being the largest discriminant if x is in the region R.

Estimation

Let us suppose that

1) true class label (given to us)

$$z_k = \begin{cases} +1 & \text{if } x_k \in \omega_1 \\ -1 & \text{if } x_k \in \omega_2 \end{cases}$$

ω_1 = class 1 ω_2 class 2

2) predicted class label

$$\hat{z}_k = \begin{cases} +1 & \text{if } g(x_k) > 0 \\ -1 & \text{if } g(x_k) < 0 \end{cases}$$

The solution can be found by minimizing an error function. eg "training error" or "emperical risk".

$$J(\omega, \omega_0) = \frac{1}{n} \sum_{k=1}^{n} [z_k - \hat{z}_k]^2$$

geometric Interpretation

Using vector algebra.

x can be expressed as.

$$x = x_p + r \frac{\omega}{\|\omega\|}$$

& $g(x) = \omega^t x + \omega_0$

$$= \omega^t \left(x_p + r \frac{\omega}{\|\omega\|} \right) + \omega_0$$

$$= \omega^t x_p + r \frac{\omega^t \omega}{\|\omega\|} + \omega_0$$

$\Rightarrow \omega^t \omega = \|\omega\|^2$ & $\omega^t x_p + \omega_0 = 0$ (projection)

So get

$$g(x) = r \|\omega\|$$

$\Rightarrow$ distance of x from hyperplane is given by

$$r = \frac{g(x)}{\|\omega\|}$$

- distance of plane from origin

$n = 0$

$r = \frac{w_0}{\|w\|}$

- feature augmenting notation

Q Assume training set of samples as :-

x	class
0	1
1	0

for a bias of value 1, what will be weight of linear discriminant function.

⇒ it becomes

$g(x) = y \begin{bmatrix} 0 \\ 1 \end{bmatrix} + 1$

if $y = -2$

for class 1. $-2 \cdot 0 + 1 = 1$.

for class 0 $-2 \cdot 1 + 1 = -1$

both are different side of line with 0 as center

So choose $y = -2$

LDF : Augmented feature vector

Linear discriminant function :

$g(x) = w^t x + w_0$

can rewrite as :

$g(x) = [w_0 \; w^t] \begin{bmatrix} 1 \\ x \end{bmatrix} = a^t y = g(y)$

new weight vector a. | new feature vector y

- y is called augmented feature vector
- Added a dummy dimension to get a completely equivalent new homogenous problem.

old problem

$g(x) = w^t x + w_0$

$\begin{bmatrix} x_1 \\ \vdots \\ x_d \end{bmatrix}$

new problem

$g(y) = a^t y$

$\begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_d \end{bmatrix}$

Notes

- LDF are optimal for gaussian distributions with equal covariance.
- Knowledge of class densities is not required when using linear discriminant functions. it is non-parametric approach.

Analyses for higher dimensions

we can consider line as

$$\omega_1 x_1 + \omega_2 x_2 + \omega_0 = 0$$

& a line it becomes

$$x_2 = -\frac{\omega_1}{\omega_2} x_1 - \frac{\omega_0}{\omega_2}$$

$$y = mx + c$$

& for more dimensions

$$\omega_0 + \omega_1 x_1 + \omega_2 x_2 + \cdots + \omega_d x_d = 0$$

or $\sum_{i=0}^{d} \omega_i x_i = 0$

$$\Rightarrow \underbrace{[\omega_0\ \omega_1\ \omega_2 \cdots \omega_d]}_{g(x),\ \omega^T} \underbrace{\begin{bmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}}_{x} = 0$$

$$g(x) = \omega^T x$$

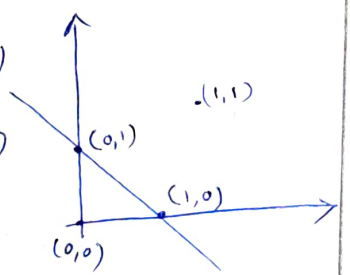
$$\text{eq}^n = \omega^T x = 0.$$

- if we got $\omega^T = [-1\ 1\ 1]$

So $g(x) = [-1\ 1\ 1]\begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix} = x_2 + x_1 - 1$

for line $x_2 + x_1 - 1 = 0$

- $x_2 = 0$ gets $x_1 = 1$ point $\rightarrow (1, 0)$
- $x_1 = 0$ gets $x_2 = 1$ $(0, 1)$



check for $(1,1)$, $g(x) = 1 + 1 - 1 = 1 > 0$

for $(0,0)$ $= 0 + 0 - 1 < 0$

on line $(1,0) = 1 + 0 - 1 = 0$.

Gradient descent

Gradient descent is an optimization algorithm used to find values of parameters (coefficients) of function (f) that minimizes a cost function (cost).

Gradient descent is best used when parameters cannot be calculated analytically (eg using linear algebra) and must be searched for by an optimization algorithm

→ basically used to minimize some error function $J(\alpha)$ with respect to α

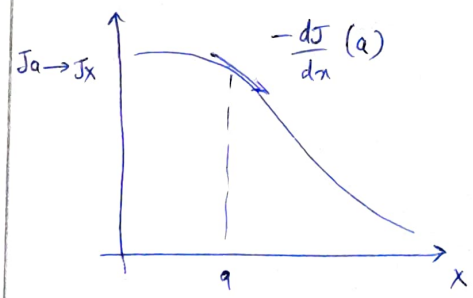
eg $J(\alpha) = \frac{1}{n} \sum_{k=1}^{n} [z_k - \hat{z}_k]^2$.

Gradient descent ⇒

$$\alpha(k+1) = \alpha(k) + \eta(k)\, p_k$$

p_k = search direction
$\eta(k)$ = learning rate

what is p_k



- ∇J points in the direction of steepest decrease.

& $p_k = -\nabla J(\alpha(k))$

& $\alpha(k+1) = \alpha(k) - \eta(k) \nabla J(\alpha(k))$

Learning rate

- if $\eta(k)$ is too small, it takes too many iterations
- if it is too big, it might overshoot the solution (and never find it) possibly leading to oscillations (no convergence)

⇒ Gradient descent is very popular due to its simplicity but can get stuck in local minima.

Newton descent

here

$$\alpha(k+1) = \alpha(k) + \eta(k)\, p_k$$

$$p_k = -H^{-1} \nabla J(\alpha(k))$$

$$\Downarrow$$

$$\alpha(k+1) = \alpha(k) - \eta(k)\, H^{-1} \nabla J.$$

H = Hessain matrix

$$H = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \\ \vdots & & & \\ \frac{\partial^2 f}{\partial x_n x_1} & \cdots & & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

Second derivative

⇒ Gradient descent can be seen as a special case of Newton's method assuming

$H = I$

⇒ If $J(\alpha)$ is quadratic Newton's method converges in one iteration

Difference

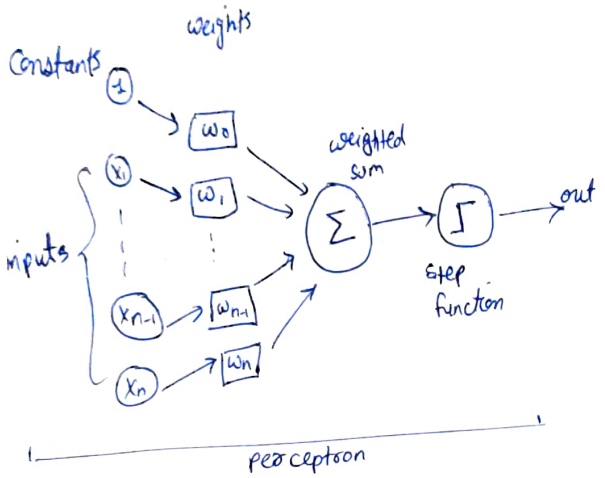
Gradient descent maximizes a function using knowledge of its derivative. Newton's method, a root finding algorithm, maximizes a function using knowledge of its second derivative. That can be faster when second derivative is known and easy to compute.

⇒ Newton's method is much faster than gradient descent.

Perceptron rule

Perceptron → it is a single layer neural network, and a multi-layer perceptron is called a neural network.

Perceptron is a linear classifier (binary). Also is used in supervised learning. It helps to classify given input data.



- Perceptron rule minimizes following error function.

$$J_p(\alpha) = \sum_{y \in Y(\alpha)} (-\alpha^t y)$$

where $Y(\alpha)$ is set of samples misclassified by α

- if $Y(\alpha)$ is empty $J_p(\alpha) = 0$ otherwise $J_p(\alpha) > 0$.
- $J_p(\alpha)$ is $\|a\|$ times sum of distances of misclassified examples to decision boundary.
- $J_p(a)$ is piecewise linear and thus suitable for gradient descent.

Can be used in 2 ways

1) Batch perceptron

here $a \leftarrow a + \eta(k) \sum_{y \in y_k} y$

Batch of misclassified examples.

→ keep updating the orientation of hyperplane until all training samples are on positive side.

2) Fixed increment single-sample perceptron.

if y_k is misclassified by a.

$$\alpha \leftarrow \alpha + \eta(k) y_k$$

→ Update is done using one misclassified example at a time.

Notes

- Batch algorithm leads to smoother trajectory in solution space.
- Single sample is easier to analyze and also concentrates more than necessary on any isolated "noisy" training samples.

Perceptron Convergence theorem

-> if training samples are linearly separable then perceptron algorithm will terminate at a solution vector in finite number of steps

Machine learning

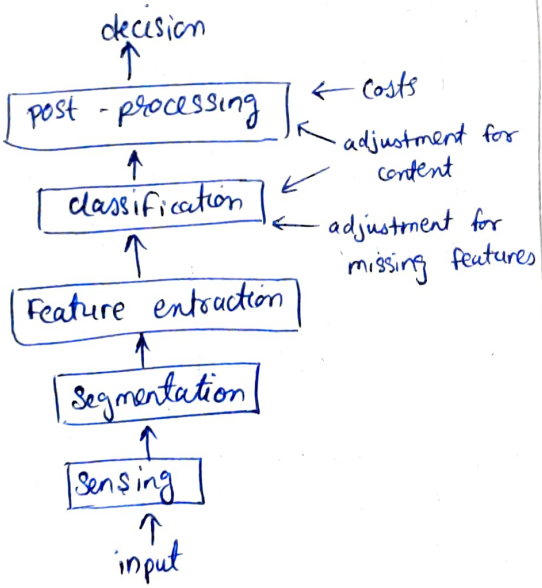
(Gopa Kumar)

- Machine learning algo's work by recognising patterns.

Pattern recognition systems

1) Sensing
- Use of transducer (camera or microphone)
- PR system depends on bandwidth, resolution sensitivity distortion of transducer.

2) Segmentation & grouping
- Patterns should be well separated and should not overlap.



3) Feature extraction
- Discriminative features
- Invariant features wrt translation, rotation & scale.

4) Classification

Use a feature vector provided by feature extractor to assign the object to a category.

5) Post - Processing

Exploit context input dependent information other than from target pattern itself to improve performance.

Bayesian Decision Theory

eg: catching sea bass and salmon fish

- State of nature, prior
  - → State of nature is random variable (ω)
  - → The catch of salmon & sea bass is equiprobable

$$P(\omega_1) = P(\omega_2) \quad \text{(uniform pairs)}$$

$$P(\omega_1) + P(\omega_2) = 1 \quad \text{(exclusivity \& exhaustivity)}$$

→ Decision rule with only prior information

Decide ω_1 if $P(\omega_1) > P(\omega_2)$ otherwise decide ω_2

Use of class-conditional information
- $P(x/\omega_1)$ and $P(x/\omega_2)$ describe diff in lightness betn populations of sea bas and salmon.

  x → parameter / variable
  here it is lightness

→ Posterior, likelihood, evidence
- $P(\omega_j/x) = P(x/\omega_j)\, P(\omega_j) / P(x)$

  (Bayes Rule)
- where in case of 2 categories

$$P(x) = \sum_{j=1}^{j=2} P(x/\omega_j)\, P(\omega_j)$$

$$P(\omega_j/x) = P(x/\omega_j) \cdot P(\omega_j) / P(x)$$

↓

Posterior = (Likelihood . Prior) / Evidence

Decision given posterior probabilities

X is an observation for which

if $P(\omega_1/x) > P(\omega_2/x)$ → True state of nature = ω_1.

Bayesian Decision Theory
Continuous Features

→ Generalization of preceding ideas

1) Use of more than 1 feature, vector X
2) Use more than 2 states of nature, c classes

Probability of Error

for a given observation x, we would be inclined to let the posterior govern our decision:

$$\omega^* = \arg\max_i P(\omega_i/x)$$

Prob. of error =

$$P(\text{error}/x) = \begin{cases} P(\omega_1/x) & \text{if we decide } \omega_2 \\ P(\omega_2/x) & \text{if we decide } \omega_1 \end{cases}$$

- avg prob of error

$$P(\text{error}) = \int_{-\infty}^{\infty} P(\text{error}/x)\, p(x)\, dx$$

Loss function

- A loss function states exactly how costly each action is
- As earlier, we have c classes $\{\omega_1, \omega_2 \dots \omega_c\}$
- we also have a possible actions $\{\alpha_1 \dots \alpha_a\}$
- The loss function $\lambda(\alpha_i/\omega_j)$ is loss function incurred for taking actions α_i when class is ω_j.

eg: Zero-One loss function

$$\lambda(\alpha_i/\omega_j) = \begin{cases} 0 & i = j \\ 1 & i \neq j \end{cases} \qquad i, j = 1, 2 \dots c$$

It assigns no loss to correct decision & uniform unit loss to an incorrect decision

- expected loss (Conditional Risk)

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

Zero-one conditional risk

$$R(\alpha_i/x) = \sum_{j \neq i} P(\omega_j/x)$$

$$= 1 - P(\omega_i/x)$$

Hence for an observation x, we can minimize expected loss by selecting action that minimize conditional risk

- overall risk is expected loss associated with given decision rule.

$$R = \oint R(\alpha(x)/x)\, p(x)\, dx$$

Pattern Classifiers
Version 1
Discriminant functions

- Discriminant functions are useful way of representing pattern classifiers
- Let $g_i(x)$ is discriminant function for i^{th} class
  This classifier will assign class ω_i to feature vector x if

$$g_i(x) > g_j(x)$$

or equivalently

$$i^* = \arg\max_i g_i(x) \quad \text{decide } \omega_i^*$$

General case with risk

$$g_i(x) = -R(\alpha_i/x)$$
$$= -\sum_{j=1}^{a} \lambda(\alpha_i/\omega_j) P(\omega_j/x)$$

⇒ for zero-one loss fn, Bayes discriminant can be further simplified:

$$g_i(x) = P(\omega_i/x)$$

$$g_i(x) = P(\omega_i/x) = \frac{P(x/\omega_i) P(\omega_i)}{\sum_j P(x/\omega_j) P(\omega_j)}$$

- Decision boundries seperate regions, they are ties among discriminant functions

- 2 category discriminant

$$g(x) = g_1(x) - g_2(x)$$

decide ω_1 if $g(x) > 0$, otherwise decide ω_2

Various manipulations of discriminant

$$g(x) = P(\omega_1/x) - P(\omega_2/x)$$

$$g(x) = \ln \frac{P(x/\omega_1)}{P(x/\omega_2)} + \ln \frac{P(\omega_1)}{P(\omega_2)}$$

Normal density

1) Continuous univariant normal, or Gaussian density

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

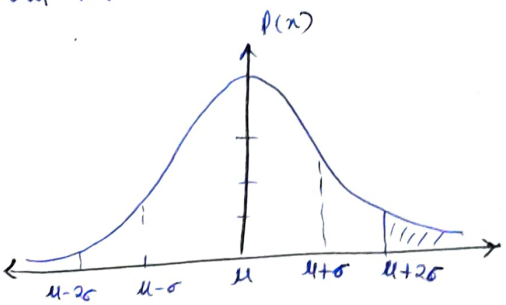
2) mean is expected value of x is

$$\mu \equiv E[x] = \int_{-\infty}^{\infty} x p(x)\, dx$$

3) variance is expected squared division

$$\sigma^2 \equiv E[(x-\mu)^2] = \int_{-\infty}^{\infty} (x-\mu)^2 p(x)\, dx$$

⇒ Samples from normal density tend to cluster around mean and be spread out based on variance



⇒ Normal density is completely specified by mean & variance. These 2 are its sufficient statistics.

$$P(x) \sim N(\mu, \sigma^2)$$

MLE

① Parameter Estimation

MLE provides the parameter value(s) that makes the observed sample the most likely sample among all possible samples.

$\hat{\theta}$ is maximum likelihood estimate if

$$L(\hat{\theta}) > L(\theta_0)$$

for all values of θ_0 in parameter space

eg 1

Pregnancy success for artificial insemination

Let n = no. of couples

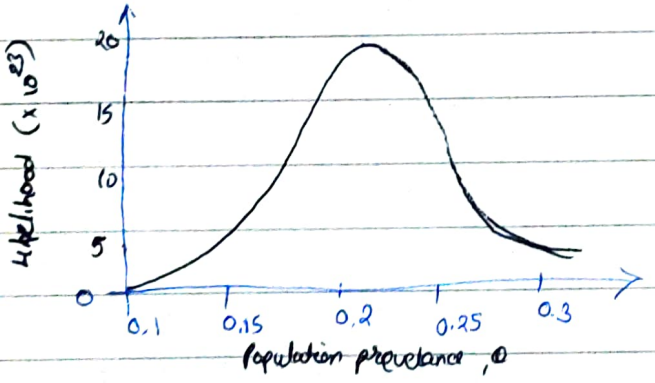
y = total pregnancy attempts by all couples

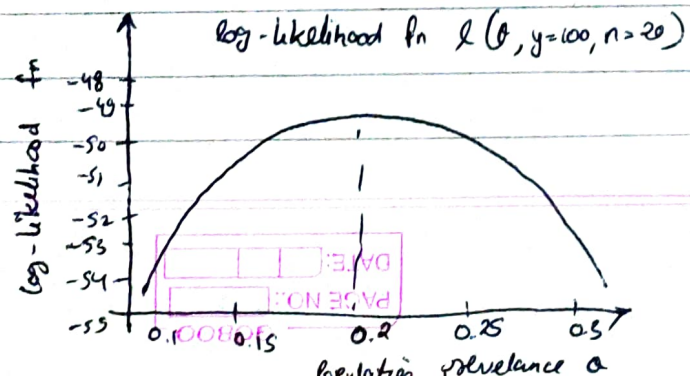
θ = probability of individual success.

$$L(\theta) = \theta^n (1-\theta)^{y-n}$$

$l(\theta) \rightarrow$ log-likelihood function.

Likelihood function $L(\theta, y=100, n=20)$





We want to find peak, and its position

To find MLE, we need to differentiate the log-likelihood fn & set it equal to 0.

$$\frac{d}{d\theta} l(\theta) = 0$$ is called score eq^n

our Confidence in MLE is quantified by the pointedness of log-likelihood,

$$I_o(\theta) = -\frac{d^2}{d\theta^2} l(\theta)$$ This is called observed information

Taking the expected value gives us expected information

$$I(\theta) = E[I_o(\theta; y)]$$

variance of estimates

$$V(\hat{\theta}) \approx I(\theta)^{-1}$$

$$l(\theta) = n \log\theta + (y-n) \log(1-\theta)$$

Q Find MLE with $n = 20$, $y = 100$ & find 95% confidence interval.

$$\frac{d}{d\theta} l(\theta) = \frac{n}{\theta} - \frac{y-n}{1-\theta} = 0 \Rightarrow n - n\theta = y\theta - n\theta$$

$$= \theta = \frac{n}{y} = \frac{20}{100} = 0.2$$

$$I_o(\theta, y) = -\frac{d^2}{d\theta^2} l(\theta) = -\left\{ -\frac{n}{\theta^2} - \frac{y-n}{(1-\theta)^2} \right\}$$

$$I_\theta = E[I_o(\theta, y)] = \frac{n}{\theta^2} + \frac{E(y)-n}{(1-\theta)^2}$$ where $E(y) = \frac{n}{\theta}$

$$I(\theta) = \frac{n}{\theta^2(1-\theta)}$$

95% CI

$$\frac{n}{y} \pm 1.96 \sqrt{\frac{\theta^2(1-\theta)}{n}}$$

$$= 0.2 \pm 1.96 \sqrt{\frac{(0.2)^2(0.8)}{20}}$$

Properties of MLE

Explicit vs Implicit MLE

- If it is possible to solve the score equation to get an expression for MLE, the MLE is explicit
- If software algorithms are required to generate MLE expression, the MLE is implicit

Biasedness & consistency

- The MLE is not an unbiased estimator necessarily.
- MLE is asymptotically biased & consistent
  The asymptotic distribution is normal

$$\hat{\theta} \overset{a}{\sim} N(\theta, I(\hat{\theta})^{-1})$$

Asymptotic efficiency

- MLE is asymptotically efficient. MLE is optimal in all but a handful of unique situations.

Invariance

MLE for $g(\theta) = g(\hat{\theta})$

Adversarial search → Decision Tree + Intro to ML.

Single Criteria & Multiple Criteria Decision Making

Linear programming is used for obtaining most optimal solution for a problem with given conditions

$CR = CI / RI$

$CR < 0.1$ → collection is rejected

$P(B/E) = \frac{P(A \cap B)}{P(A)}, P(A) > 0$

$\chi^2 = \frac{n - \mu}{\sigma}$

its position

COURSE
PAGE NO.
DATE

the

systolic blood pressure

A sample of 5 pregnant women have their SBP taken, which is considered to be normally distributed.

Sample: {135, 123, 120, 102, 110}

find maximum likelihood estimate for μ & σ.

$$L(\mu, \sigma^2) = \prod_{i=1}^{5} f_i(y_i; \mu, \sigma^2) \quad \text{where}$$

$$f(y_i, \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(y_i-\mu)^2}{2\sigma^2}}$$

$$L(\mu, \sigma^2) = \left(\frac{1}{2\pi\sigma^2}\right)^{n/2} \exp\left(-\sum_{i=1}^{n} \frac{(y_i-\mu)^2}{2\sigma^2}\right)$$

$$l(\mu, \sigma^2) = \cdots$$

$$\frac{dl}{d\mu} = \frac{2(\Sigma y - n\mu)}{2\sigma^2} = \frac{\Sigma y - n\mu}{\sigma^2} = 0$$

$$\hat{\mu} = \frac{\Sigma y}{n}$$

$$\frac{dl}{d\sigma^2} = \frac{-n}{2\sigma^2} + \frac{1}{2(\sigma^2)^2}\left(\Sigma y^2 - 2\mu\Sigma y + n\mu^2\right) = 0$$

$$\hat{\sigma}^2 = \left(\frac{1}{n}\right)\left(\Sigma y^2 - 2\hat{\mu}\Sigma y + n\hat{\mu}^2\right)$$

$$= \frac{\Sigma y^2}{n} - \frac{2\Sigma y \Sigma y}{n^2} + \frac{(\Sigma y)^2}{n^2}$$

$$= \frac{\Sigma y^2}{n} - \left(\frac{\Sigma y}{n}\right)^2$$

or

$$\frac{\Sigma (y - \bar{y})^2}{n}$$

Parametric ML algorithms

1) Regression, 2) LDA
3) Perceptron

Advantage	Disadvantage
Simple	constrainted
fast	limited complexity
less data	Poor fit

Non Parametric Algo

1) Decision tree
2) Naive Bayes
3) SVM
4) Neural Network

Advantage	Disadvantage
1) flexibility	Most Data
2) Power	slower
3) Performance	overfitting

- In parametric model, no. of parameters is fixed with respect to sample size.
In non parametric model, the (effective) number of parameters can grow with sample size.