Confidence Interval (CI)

X_n be i.i.d w/ mean μ and variance σ^2

$A_n = \frac{X_1+\dots+X_n}{n}$, Then if 95% CI

$\Pr[\mu \in [A_n - 4.5\frac{\sigma}{\sqrt{n}}; A_n + 4.5\frac{\sigma}{\sqrt{n}}] \geq 95\%$

$\text{Var}(A_n) = \text{Var}(\frac{X_1+\dots+X_n}{n}) = \frac{1}{n}\sigma^2$, $E[A_n] = E[X_i]$

Let X_n be i.i.d. $B(p)$. $A_n = \frac{X_1+\dots+X_n}{n}$

$[A_n - \frac{2.25}{\sqrt{n}}, A_n + \frac{2.25}{\sqrt{n}}]$ is 95%-CI for p

Let X_n be i.i.d. Geom(p)., $A_n = \frac{X_1+\dots+X_n}{n}$

$[\frac{A_n}{1+4.5/\sqrt{n}}, \frac{A_n}{1-4.5/\sqrt{n}}]$ is 95%-CI for $\frac{1}{p}$

Linear Regression

$\text{Cov}(X,Y) = E[(X-E[X])(Y-E[Y])]$
$= E[XY] - E[X]E[Y]$

$\text{Cov}(X,X) = \text{Var}(X) = E[X^2] - E[X]^2$

$\text{Cov}(X,Y) > 0 \Rightarrow$ positively correlated.

Calculate $E[XY]$ in graph $= \sum x \cdot y \cdot \Pr(x,y)$

$\text{Cov}(X,Y) = 0$ if $X \perp Y$ independent

$\text{Cov}(a+X, b+Y) = \text{cov}(X,Y)$

$\text{Cov}(aX+bY, cU+dV) =$
$ac \cdot \text{Cov}(X,U) + ad \cdot \text{cov}(X,V) + 2$ more

$L[Y|X] = \hat{Y} = E[Y] + \frac{\text{Cov}(X,Y)}{\text{Var}(X)}(X - E[X])$

if $E[Y|X]$ is $a+bX$ then $L[Y|X] = E[L|X]$

$Q[Y|X] = a+bX+cX^2$ to min.

$E[(Y-a-bX-cX^2)^2]$

Conditional Expectation CE=MMSE

$E[Y|X] = g(X)$

$E[Y|X=x] = \sum_y y \Pr[Y=y|X=x]$
$= \sum_\omega Y(\omega) \Pr[\omega|X=x]$.

Properties:
(1). X, Y ind. $\Rightarrow E[Y|X] = E[Y]$
(2). $E[aY+bZ|X] = aE[Y|X] + bE[Z|X]$
(3) $E[Yh(X)|X] = h(X)E[Y|X]$
(4) $E[h(X)E[Y|X]] = E[h(X)Y]$

Markov chains

Transition matrix P, $P(i,j)$ = prob. from X_i to X_j.

$\pi_{t+1} = \pi_t P$

irreducible iff there exists some path b/w every pair of states

A state X_i is aperiodic if length of all path starting at X_i ending at X_i has GCD of 1.

A MC is aperiodic if all states are aperiodic

MC only converges if irreducible and aperiodic, periodic would relie on initial distribution.

irreducible means that every state has same aperiodicity.

Invariant distribution unique $\pi = \pi P$

Balance equations $\pi(j) = \sum_i P(i,j)\pi(i)$, $\sum_j \Pr(i,j) = 1$

$\sum \pi(i) = 1$

First step equations: $\alpha(i) = \sum_j \Pr(i,j)\alpha(j)$ ← probability

$\beta(i) = \text{time at } X_i + \sum_j P(i,j)\beta(j)$ hitting time problem

irreducible MC w/ invariant distribution π

$\frac{1}{n}\sum_{m=0}^{n-1} 1\{X_m = i\} \to \pi(i)$ as $n \to \infty$

aperiodic and irreducible

$\pi_n(i) \to \pi(i)$ as $n \to \infty$

$\pi_n = \pi_0 P^n$

$P(i,j) = \Pr[X_{n+1} = j \mid X_0, \dots X_n = i]$

irreducible $\Rightarrow$ one and only one invariant distribution π

irreducible $\Rightarrow$ fraction of time in state i approaches $\pi(i)$

Continuous Probability

CDF: $F(x) = \Pr[X \leq x] = \int_{-\infty}^{x} f(x)\,dx$ cumulative density function

PDF: $f(x) = \frac{d}{dx}F(x)$ = probability density function

$E[X] = \int_{-\infty}^{\infty} x f(x)\,dx$ $\quad P([a,b]) = \int_a^b f_x(x)\,dx = F_x(b) - F_x(a)$

$E[g(x)] = \int_{-\infty}^{\infty} g(x) f(x)\,dx$ $\quad$ 1. $\forall x\ f_x(x) \geq 0$ $\quad F_x(-\infty) = 0$

uniform from 0 to 1 $\quad$ 2. $\int_{-\infty}^{\infty} f_x(x)\,dx = 1$ $\quad F_x(+\infty) = 1$

$f(x) = 1$ $\quad F(x) = x$ $\quad \Pr[x < X < x+\varepsilon] \approx f_x(x)\,\varepsilon$

uniform hitting a circle
$F(x) = \frac{\text{area small circle}}{\text{area of board}} = \frac{\pi x^2}{\pi(1)^2} = x^2$
$f(x) = 2x$

$U[a,b] = a + (b-a)U[0,1]$

Expo answers how many units of time until first success.

Expo(λ) for $\lambda > 0$

$f_x(x) = \lambda e^{-\lambda x} 1\{x \geq 0\}$

$F_x(x) = \begin{cases} 0 & \text{if } x < 0 \\ 1 - e^{-\lambda x} & \text{if } x \geq 0 \end{cases}$

Properties.
1. Memoryless: $\Pr[X > t+s \mid X = s] = \Pr[X > t]$
2. Scaling
$\alpha\,\text{Expo}(\lambda) = \text{Expo}(\lambda/\alpha)$; $\text{Expo}(\lambda) = \frac{1}{\lambda}\text{Expo}(1)$
$X = \text{Expo}(\lambda)$; $Y = \alpha X$
$\Pr[Y > t] = \Pr[X > t/\alpha] = \text{Expo}(\lambda/\alpha)$
3. Scaling uniform. $\Pr(X > x) = e^{-\lambda x}$
$X = U[0,1]$, $Y = a+bX$ $\quad X = \text{Expo}(\lambda)$
$f_Y(y) = 1/b$ f

if X has pdf of $f_x(x)$, $Y = a+bX$
$f_Y(y) = \frac{1}{b} f_x\left(\frac{y-a}{b}\right)$

$X = \text{Expo}(\lambda)$ $\boxed{E[X] = \frac{1}{\lambda},\ \text{var}(X) = \frac{1}{\lambda^2}}$

Joint Continuous RV

$f_{X,Y}(x,y)\,dx\,dy = \Pr[X \in (x, x+dx), Y \in (y, y+dy)]$

$f_{X,Y} = \frac{1}{|A|}$ $1\{(x,y) \in A\}$, if uniform ($|A|$ = area)

if X, Y independent $\Pr[X \in A, Y \in B] = \Pr[X \in A]\Pr[Y \in B]$

$f_{X,Y}(x,y) = f_x(x) f_y(y)$

$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f_{X,Y}(x,y)\,dx\,dy = 1$

$f_x(x) = \int_{-\infty}^{\infty} f_{X,Y}(x,y)\,dy$

$f_{X|Y}(x|y) = \frac{f_{X,Y}(x,y)}{f_Y(y)}$

Max of two exponentials

$X = \text{Expo}(\lambda)$; $Y = \text{Expo}(\mu)$
$Z = \max\{X, Y\}$
$\Pr[Z \leq z] = \Pr[X \leq z, Y \leq z]$
$= 1 - e^{-\lambda z} - e^{-\mu z} + e^{-(\lambda+\mu)z}$
take $\frac{d}{dz}$ to get $f_Z(z)$
$E[Z] = \int_0^\infty f_Z(z) = \frac{1}{\lambda} + \frac{1}{\mu} - \frac{1}{\lambda+\mu}$

Max of n i.i.d. Exponentials

let $X_1, \dots X_n$ be i.i.d Expo(1) and
$Z = \max(X_1, \dots X_n)$
$Z = \min(X_1, \dots X_n) + V$ where
V = max of $n-1$ i.i.d Expo(1)
$A_n = E[\min\{X_1, \dots X_n\}] + A_{n-1}$
$= \frac{1}{n} + A_{n-1}$
$E[Z] = A_n = 1 + \frac{1}{2} + \dots + \frac{1}{n} = H(n) = \ln(n) + \gamma$

minimum of Expo is Expo with the sum of the rates

Min of expos

$X = \min\{X_1, \dots X_n\} \sim \text{Expo}(\lambda_1 + \dots + \lambda_n)$
$P(X > x) = P(X_1 > x, \dots X_n > x)$
$= P(X_1 > x) \dots P(X_n > x)$

Normal (Gaussian) Distribution

$Y = N(\mu, \sigma^2)$ has pdf:
$f_Y(y) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(y-\mu)^2/2\sigma^2}$

Standard normal has $\mu = 0$ $\sigma = 1$
$X = N(0,1)$ $Y = \mu + \sigma X$ then $Y = N(\mu, \sigma^2)$
$E[Y] = \mu$ $\text{var}[Y] = \sigma^2$

Central Limit Theorem

let $X_1, X_2 \dots X_n$ be i.i.d w/ $E[X_i] = \mu$ and $\text{var}(X_i) = \sigma^2$

$S_n = \frac{A_n - \mu}{\sigma/\sqrt{n}} = \frac{X_1 + \dots + X_n - n\mu}{\sigma\sqrt{n}}$

$S_n \to N(0,1)$ as $n \to \infty$

$\Pr[S_n < \alpha] \to \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\alpha} e^{-x^2/2}\,dx$

$[A_n - \frac{\sigma}{\sqrt{n}}, A_n + \frac{\sigma}{\sqrt{n}}]$ is 95%-CI for μ

Coins and normal
$X_1, \dots X_2$ be $B(p)$ then $\mu = p$ $\sigma = \sqrt{p(1-p)}$
$\frac{X_1 + \dots + X_n - np}{\sqrt{p(1-p)n}} \to N(0,1)$ $[A_n - \frac{1}{\sqrt{n}}, A_n + \frac{1}{\sqrt{n}}]$ = 95%-CI

Scanned with CamScanner

Continuous RV and Bayes's rule

Ex 1: find $E[Y|X=x]$

w.p. ½ - X, Y are i.i.d. Expo(1) and w.p ½ they are Expo(3)

let B be event that $X \in [x, x+\delta]$

let A be X, Y = expo(1)

$\Pr[A|B] = \frac{\Pr(A)\Pr(B|A)}{\Pr(A)\Pr(B|A)+\Pr(\bar{A})\Pr(B|\bar{A})} = \frac{e^{-x}}{e^{-x}+3e^{-x}}$

$E[Y|X=x] = E[Y|A]\Pr[A|X=x] + E[Y|\bar{A}]\Pr[\bar{A}|X=x]$

$= \frac{1+e^{2x}}{3+e^{2x}}$

Ex 2:

w.p. ½ Bob good play shoots dirt Uniform in $r=1$, w.p. ½ very good $r=1/2$.

Bob shoot 0.3 from center

(a) prob. very good

(b) Expected distance next dirt

(a) $\Pr[VG|0.3] = \frac{\Pr(VG)\Pr[0.3|VG]}{\Pr[VG]\Pr[0.3|VG]+\Pr[\bar{G}]\Pr[0.3|G]} = 0.8$

use $\Pr[X \in (x, x+\delta)] \approx f_x(x)\delta$

(b) $E[x] = 0.8 \times 0.5 \times \frac{2}{3} + 0.2 \times \frac{2}{3} = 0.4$

Poisson Process

Each lightbulb is i.i.d Expo(λ) and replace one as old one dies.

# of lb we replace by time t is

$P(N_t = n) = \frac{e^{-\lambda t}(\lambda t)^n}{n!}$

Sum of independent Gaussians

If $X \sim N(\mu_1, \sigma_1^2)$ and $Y \sim N(\mu_2, \sigma_2^2)$, and independent, then $X+Y \sim N(\mu_1+\mu_2, \sigma_2^2+\sigma_1^2)$

Normal

$\Pr[|Y-\mu| > 1.65\sigma] = 10\%$

$\Pr[|Y-\mu| > 2\sigma] = 5\%$

Markov Property!

X_{n+1} and X_{n-1} are independent given X_n.

b blue r red, what is expect of # of b until k^{th} r?

$E[X_i] = k/(r+1)$ $E[X] = \frac{bk}{r+1}$

var(non-empty) = var(m-non-empty) = var(empty)

How many evens before seeing odd?

X = # of evens Y = # of odds

$X \sim Bin(Y-1, 1)$

$Y \sim Geom(1/2)$

$E[X] = E[E[X|Y]] = E[Y-1] = 1$

* $a^{p-1} \equiv 1 \bmod p \Rightarrow a^p = a \bmod p$
* $a^{(p-1)(q-1)} \equiv 1 \bmod pq$
* no polynomial deg $> p-1$ in mod p.
* $\Pr[B] = \Pr[A]\Pr[B|A] + \Pr[\bar{A}]\Pr[B|\bar{A}]$
* countably infinite: add an element, map each existed to n-1 and make f(0) be new element.
* chebyshov gives $[A_n - \frac{4.5\sigma}{\sqrt{n}}, A_n + \frac{4.5\sigma}{\sqrt{n}}]$ as 95%-CI for μ

** More distributions

Uniform in $\{1, 2, \ldots, n\}$: $U[1, \ldots, n]$

$\Pr[X=m] = 1/n$, $\mu = \frac{n+1}{2}$ $var = \frac{n^2-1}{12}$

Bin(p)

$\Pr[X=1] = p$; $\Pr[X=0] = 1-p$ $\mu = p$ $var = p(1-p)$

Uniform $U[0,1]$ continous

$f_X(x) = 1\{0 \le x \le 1\}$ Mean $= \frac{1}{2}$, $var = \frac{1}{12}$

Expo(λ)

$f_X(x) = \lambda e^{-\lambda x} 1\{x > 0\}$ $F_X(x) = \{1 - e^{-\lambda x}\}1\{x \ge 0\}$

$\mu = \frac{1}{\lambda}$ $var = \frac{1}{\lambda^2}$

Gaussian $N(\mu, \sigma^2)$

$f_X(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x-\mu)^2/(2\sigma^2)}$ $\mu = \mu$ $var = \sigma^2$

Math

$1+2+\ldots n = \frac{n(n+1)}{2}$

for $|a| < 1$, $1 + a + a^2 + \ldots = \frac{1}{1-a}$

For $a \ne 1$, $n \ge 0$; $1 + a + a^2 + \ldots + a^n = \frac{(1-a^{n+1})}{(1-a)}$

$\exp\{\varepsilon\} \approx 1 + \varepsilon$ for $|\varepsilon| \ll 1$

$\ln(1+\varepsilon) \approx \varepsilon$ $\|\varepsilon\| \ll 1$

* Max. of two rolls, use CDF
* Marbles w/o replacement, pty w/ time use indicator (count success)

Tail Sum Formula

$E[X] = \sum_{x=1}^{\infty} P(X \ge x)$

where $P(X \ge x) = 1 - P(X < x)$

* $E[X^2] = \sum_{i=1}^{n} E[X_i^2] + \sum_{i \ne j} E[X_i X_j] = nP(A_i) + n(n-1)P(A_i \cap A_j)$

$E[1_A] = P(A)$

$E[1_A^2] = E[1_A]$

$X(t+1) = \alpha X(t)$; $X(t) = \alpha^t X(0)$

$X(t+1) = \alpha X(t) + \beta$, $X(t) = \alpha^t X(0) + \beta \frac{1-\alpha^t}{1-\alpha}$

Friend comes in to see, expected lifetime of lightbulb that is still burning is 2/2 even though $E[Expo(\lambda)] = 1/\lambda$

Scanned with CamScanner

Polynomials

$d+1$ points can give a specific polynomial w/ degree d.

degree d polynomials has d roots

Interpolation

$$D_i(x) = \frac{(x-a_1)(x-a_2)\cdots(x-a_{i+1})\cdots}{(a_i-a_1)(a_i-a_2)\cdots(a_i-a_{i+1})\cdots}$$

$$P(x) = \sum_{i=0}^{d} P(a_i) D_i(x)$$

Secret sharing

at least k people, form $k-1$ degree polynomial, let $P(0)$ = secret, give one pair of point to each person.

Erasure Errors

send $n+k$ points

General errors

send $n+2k$ points

$Q(x) = P(x)E(x)$ — $Q(i) = R(i)E(i)$ ← received i

$P(x) = Q(x)/E(x)$

$E(x)$ has degree k ← k unknown coefficient

$Q(x)$ has degree $n+k-1$ ← $n+k$ unknown coefficient

$E(x) = x^k + b_{k-1}x^{k-1} + \cdots + b_0$

$Q(x) = a_{n+k-1}x^{n+k-1} + \cdots + a_0$

Random Variable

A function $X: \Omega \to \mathbb{R}$

Expectation

$$E[X] = \sum_{x=1}^{\infty} x \times Pr[X=x]$$

Inequalities

$$Pr[X \geq a] \leq \frac{E[f(X)]}{f(a)}$$ Markov's

$$Pr[|X-E[X]| > a] \leq \frac{Var[X]}{a^2}$$ Chebyshev's

Weak Law of Large Number: $\frac{X_1 + \cdots X_N}{N} \approx E[X]$

Confidence Intervals

$1-\delta$ ← prob. of error (confidence)

ε ← tolerance, n ← # of sample

$$Pr(|X-E(X)| \geq \varepsilon) \leq \frac{Var(X)}{\varepsilon^2}$$

one-side: $Pr(X-E(X) \geq \varepsilon) \leq Pr(|X-E(X)| \geq \varepsilon)$

2-side

* Distributions

Geometric Geo(p)

- p probability success at each trial, # of trials need to succeed?

$Pr[X=k] = (1-p)^{k-1}p$ $E[X] = \frac{1}{p}$ $Var[X] = \frac{1-p}{p^2}$

Binomial bin(n,p)

- p probability success at each trial, prob. that k succeed in n trials?

$Pr[X=k] = \binom{n}{k}p^k(1-p)^{n-k}$ $E[X] = np$ $Var[X] = np(1-p)$

Poisson Distribution

- average occurance = λ

$E[X] = \lambda$

$Pr[X=k] = e^{-\lambda}\left(\frac{\lambda^k}{k!}\right)$

$Var[X] = \lambda$

Uniform

$U[1,\ldots n]$: $Pr[X=m] = \frac{1}{n}$, $m = 1,\ldots,n$.

$E[X] = \frac{n+1}{2}$

$Pois(20) = Pois(10) + Pois(10)$

Indicator

$$X(w) = \begin{cases} 1, & \text{if } w \in A \\ 0, & \text{if } w \notin A \end{cases}$$

$E[X] = Pr[A]$

$X = I_A$ $I_A^2 = I_A$

$E[X] = \sum_{w}^{k} X(w) Pr[w]$

Linearity of Expectations

$E[a_1X_1 + \ldots + a_nX_n] = a_1E[X_1] + \ldots + a_nE[X_n]$

Coupon Collector Problems

X - time to get n coupons, $\frac{1}{n}$ chance for every coupon

X_i - time to get i^{st} coupon $= i$

$Pr[X_2|X_1] = \frac{n-1}{n}$ $Pr[X_i|X_{i-1}] = \frac{n-(i-1)}{n} = \frac{n-i+1}{n}$

$E[X_2] = \frac{1}{p} = \frac{n}{n-1}$ $E[X_i] = \frac{n}{n-i+1}$

$E[X] = E[X_1] + \ldots E[X_n] = \frac{n}{n} + \frac{n}{n-1} + \ldots \frac{n}{1} = nH(n) \approx n(\ln(n))$

If X and Y are independent $E[XY] = E[X]E[Y]$

Collection & Collisions

$Pr[\text{no collision}] \approx e^{-m^2/2n}$

Collection:

$Pr[\text{missing Wilson}] \approx e^{-m/n}$

$Pr[\text{miss at least one}] \leq n e^{-m/n}$

* $P(x)$ = d degree, d solutions to $P(x) = r$
* no polynomial deg $> p-1$ $\overset{\text{in}}{\text{mod } p}$

Variance and Standard Deviation

$\sigma^2(X) = Var[X] = E[X^2] - E[X]^2 = E[(X-E(X))^2]$

$\sigma(X) = \sqrt{Var[X]}$ ≡ standard deviation

Ex: $X = \begin{cases} -1 & \text{w.p. } 0.99 \\ 99 & \text{w.p. } 0.01 \end{cases}$

$E[X] = -1 \times 0.99 + 99 \times 0.01 = 0$

$E[X^2] = (-1)^2 \times 0.99 + 99^2 \times 0.01 = 100.$

$E[X^2] = Var[X] + E[X]^2$

$Var(cX+d) = c^2 Var(X)$

if X and Y independent $Var(X+Y) = Var(X) + Var(Y)$

Uniform

$E[X^2] = \frac{1+3n+2n^2}{6}$

Geometric

$E[X^2] = \frac{(2-p)}{p^2}$

Math tricks

$\ln(1-\varepsilon) \approx \varepsilon$

$e^{-\varepsilon} \approx 1-\varepsilon$

$(a+b)^n = \sum_{m=0}^{n} \binom{n}{m} a^m b^{n-m}$

$1+2+\ldots+n = \frac{n(n+1)}{2}$

Scanned with CamScanner

Counting

X nCr Y = $\binom{X}{Y} = \frac{X!}{Y!(X-Y)!}$

Ordering of letters = $\frac{n!}{m_1! m_2! \dots}$

where n = length of word, m_1, m_2 = repeated letters

Balls and bins stars and bars

total ways = $\binom{\text{stars + bars}}{\text{bars}}$ bars = bins − 1

numbers in 0-qqqqqq whose digit sum = 9: $\binom{14}{5}$

(be careful for overflow)

need to substract them out

Ways = $n_1 \times n_2 \times \dots n_k$ n_i = choices for i

$|B| = \frac{|A|}{k}$

Sampling w/ replacement: n^k.

coin flip: 2^k

Combination proof

LHS counts same as RHS

Probability

$Pr(A) = \frac{|A|}{|\Omega|}$

$Pr(A|B)$ = prop of A given B = $Pr(A \cap B)/Pr(B)$.

if A & B independent $Pr(A|B) = Pr(A)$

$Pr(A \cup B) = Pr(A) + Pr(B) - Pr(A \cap B)$ [inclusion/exclusion prop]

$Pr(A \cap B) = 0$ if A & B are disjointed

$Pr(A \cup \dots \cup A_n) \leq Pr(A) + \dots Pr(A_n)$ [Union bound]

If $A_1, \dots A_n$ are partition of Ω (disjoint),

then $Pr(B) = Pr(B \cap A_1) + Pr(B \cap A_2) + \dots Pr(B \cap A_n)$; $Pr(B \cap A_i) = Pr(B|A_i) Pr(A_i)$

[law of total probability]

$Pr(B) = \sum_{i=0}^{k} Pr(A_i) Pr(B|A_i)$

Bayes' Rule

$Pr(A|B) = \frac{Pr(B|A) Pr(A)}{Pr(B)}$

Product Rule

$Pr(A_1 \cap \dots \cap A_n) = Pr(A_1) Pr(A_2|A_1) \dots Pr(A_n | A_1 \cap \dots \cap A_{n-1})$

Mutually independent

$Pr[\cap_{k \in K} A_k] = \prod_{k \in K} Pr[A_k]$

head has prop of p, flip a head and tail = p(1−p)

Two disjoint A and B w/ $Pr(A) > 0$ and $Pr(B) > 0$ can't be independent.

Ω is always independent of any event A in Ω

$Pr[A \cap \Omega] = Pr[A] \cdot 1 = Pr[A] Pr[\Omega]$

Symmetry

1st card > 2nd card same freq. as 2nd card > 1st card

Positive correlate $Pr[A \cap B] > Pr[A] Pr[B]$

Countability

(x, y) is countable

((x, y), z) also

Natural numbers = pos. integers = countably infinite

Enumeration (put thing in an ordered list) = countable.

Use diagonization to prove uncountably infinite.

If a set can map 1-to-1 to integers or a countable infinite set, it is countable., show bijection. For uncountable, show subjection.

Halt

To prove that a program is impossible, prove that we can use that program to create HALT program.

Turing(P)

if Halting(P,P) = "halts", loop forever, else halts

Turing(Turing) contradiction

Modular Arithmetic

gcd(x, y) = m ≡ ax + by = m

if $X \cong b \bmod mn$, then $X \cong b \bmod m$ and $X \cong b \bmod n$.

$xy = 1 \bmod m \Rightarrow y$ is the multiplicative inverse of x mod m

⇒ do the steps of gcd(x, m) to find y.

The Chinese Remainder Theorem

$X \equiv a_i \pmod{m_i}$, $1 \leq x \leq r$, $M \equiv m_1 \times m_2 \times \dots m_r$

$x \equiv a_1 M_1 y_1 + \dots a_r M_r y_r \pmod M$

where $M_i = M/m_i$ and

$y_i \equiv (M_i)^{-1} \pmod{m_i}$

y_i is the mult. inverse of M_i mod m_i

RSA

p, q large primes

N = pq (message x are sent mod N)

e, small, relatively prime to (p−1)(q−1)

mult. inverse. $d = e^{-1} \bmod (p-1)(q-1)$ Private keys

(N, e) public keys

$E[x] = m^e \bmod N$

$D[E[x]] = (E[x])^d \bmod N = m^{ed} \bmod N = m \bmod N$

Use $ae_1 + be_2 = 1$, egcd, to find original m if there are 2 public keys

Fermat's little theorem

For prime p and $a \neq 0 \bmod p$,

then $a^{p-1} \equiv 1 \bmod p$

$3^{10} \bmod 11 = 1 \bmod 11$

$3^{453} \bmod 11 = 3^{450} \cdot 3^3 \bmod 11 = (3^{10})^{45} \cdot 3^3 \bmod 11$

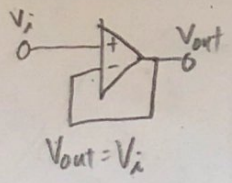
Scanned with CamScanner

* no polynomial deg $> p-1$ in mod p
* The powerset of a set w/ cardinality of int (an countable inf.) is uncountable
* Comput can't print uncountable sets
* $Pr[B] = Pr[A]Pr[B|A] + Pr[\bar{A}]Pr[B|\bar{A}]$
* A degree p polynomial need $p+1$ people to unlock
* $a^{p-1} \equiv 1 \bmod p \Rightarrow a^p = a \bmod p$
* $a^{(p-1)(q-1)} \equiv 1 \bmod pq$
* $x^e \bmod pq$ is bijection if e is co-prime with $p-1$ and $q-1$, like RSA
  The number of #s that are coprime to pq from 1 to pq is $(p-1)(q-1)$
* Countably infinite: add an element, map each existed to $n+1$ and make $f(0)$ be new element

Scanned with CamScanner

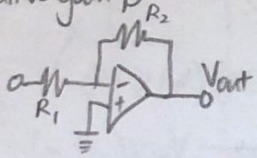
Op-Amps

Key Phrase "Isolate load, zero output/source resistance"
Use buffer



$V_{out} = V_i$

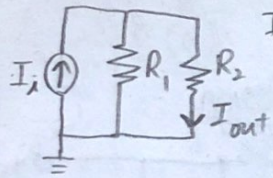
Key Phrase : "negative gain β"
Use inverting amp



$V_{out} = -\frac{R_2}{R_1} V_i$

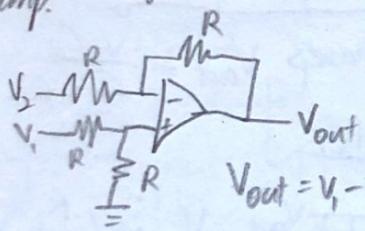
$\beta = -\frac{R_2}{R_1}$

Key Phrase: "divide current by a factor of α, use R"
Use resistive current divider



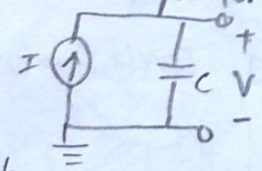
$I_{out} = \frac{R_1}{R_1+R_2} I_i$

Key Phrase "subtract 2 voltage V_1, V_2"
Use difference amp.



$V_{out} = V_1 - V_2$

Key Phrase "creat a voltage that is linear w/ respect to time or a voltage ramp w/ slop β"
Use current source w/ capacitor



$I = C\frac{dV}{dt}$

$I = C\frac{V_1 - V_2}{t_1 - t_2}$ $\quad \frac{I(t_1-t_2)}{C} = V_1 - V_2$

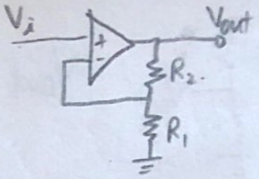
$V = \frac{I}{C}t$ $\quad \beta = \frac{I}{C}$

$P = IV$

$C = \varepsilon_0 \frac{A}{d}$

Current out of a node is positive

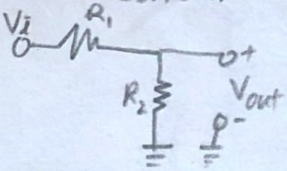
Key Phrase : "positive gain β"
Use noninverting amp.



$V_{out} = (1+\frac{R_2}{R_1})V_i$

$\beta = (1+\frac{R_2}{R_1})$

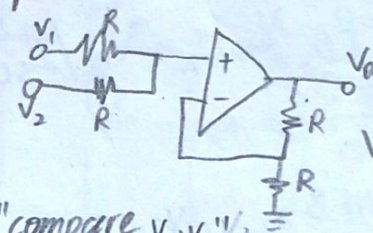
Key Phrase : "divide voltage by factor β"
Use voltage divider (R in series)



$V_{out} = \frac{R_2}{R_1+R_2} V_i$

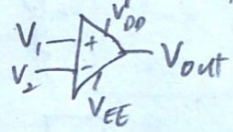
$\beta = \frac{R_2}{R_1+R_2}$

Key Phrase : "add two voltages V_1, V_2"
Use summing amp.



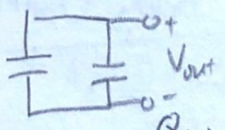
$V_{out} = V_1 + V_2$

Key Phrase : "compare V_1, V_2"

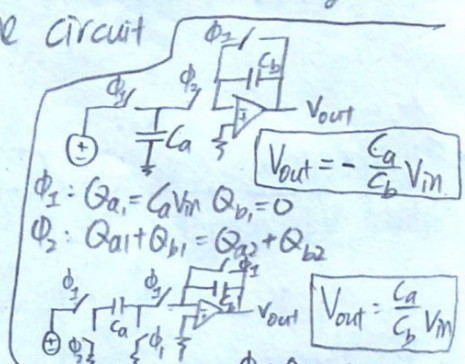


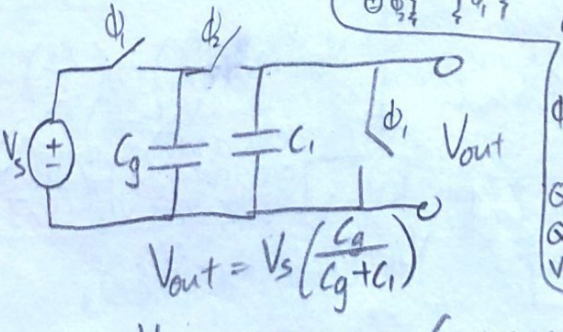
$V_{out} = \begin{cases} V_{DD} & \text{if } V_1 > V_2 \\ V_{EE} & \text{if } V_1 < V_2 \end{cases}$

For circuits, nullspace means a set of valid voltage and current values for the circuit



$V_{out} = \frac{Q_{tot}}{C_{tot}}$





$V_{out} = V_s\left(\frac{C_g}{C_g + C_1}\right)$

ϕ_1: $Q_{a1} = C_a V_{in}$
$Q_{a2} = 0$

ϕ_2: $-Q_{a1} + Q_{b1} = -Q_{a2} + Q_{b2}$

$Q_{a2} - Q_{a1} = Q_{b2} - Q_{b1}$

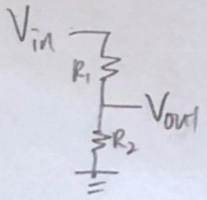
$Q_{a2} = 0$ $\quad V_{out} = -V_b$

$V_{b2} = -\frac{C_a}{C_b} V_{in} = \frac{C_a}{C_b} V_{in}$

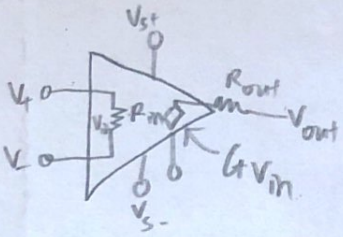
V_{out} increases as C_g increases, vice versa (not linearly)

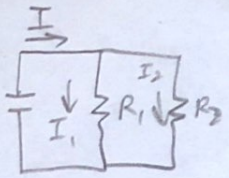
Scanned with CamScanner

Equations: $R = \rho \frac{L}{A}$ $E = \frac{1}{2}CV^2$ $C = \varepsilon \frac{A}{d}$ $P = IV$ $V = IR$ $Q = CV$

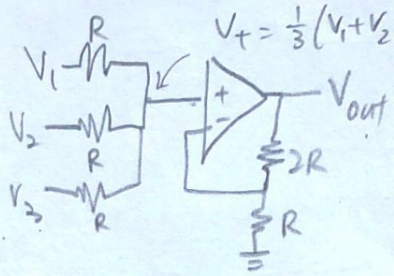


$V_{out} = \frac{R_2}{R_1+R_2} V_{in}$





$I_1 = \frac{R_2}{R_1+R_2} I_{tot}$



$V_+ = \frac{1}{3}(V_1+V_2+V_3)$

$V_{out} = V_1+V_2+V_3$

$G = 3$

$V_{Th} = V_{OC}$

$I_N = I_{SC}$

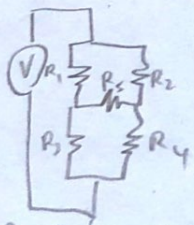
$R_{eq} = R_{Th} = R_N = V_{Th}/I_N = V_{OC}/I_{SC}$

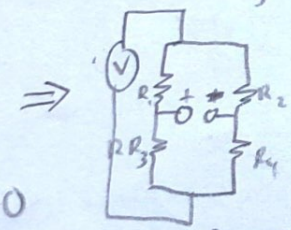
Golden Rules

$I_- = I_+ = 0$

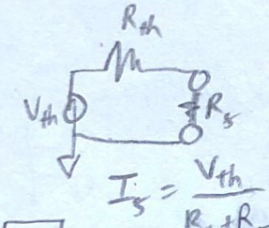
$V_- = V_+$ NFB

Wheatstone Bridge





If $R_1 = R_2$ & $R_3 = R_4$ then $I_S = 0$

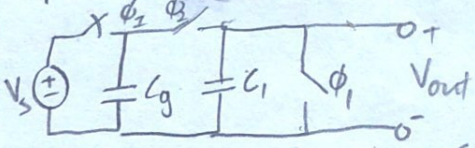


$I_s = \frac{V_{th}}{R_{th}+R_s}$

$V_{th} = V\left(\frac{R_3}{R_1+R_3} - \frac{R_4}{R_2+R_4}\right)$

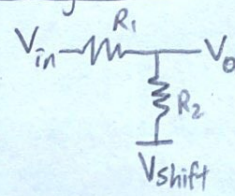
$R_{th} = R_1 \| R_3 + R_3 \| R_4$

Capacitor Design



After 2 phases $V_{out} = \frac{V_s C_g}{C_g + C_1}$

Voltage shifter



$V_{out} = \left(\frac{R_2}{R_1+R_2}\right) V_{in} + \left(\frac{R_1}{R_1+R_2}\right) V_{shift}$

Want $V_{out} = 0$ when $V_{in} = x$, pick R_1, R_2, V_{shift}

Least-square solution corr(a,b) → shift b.

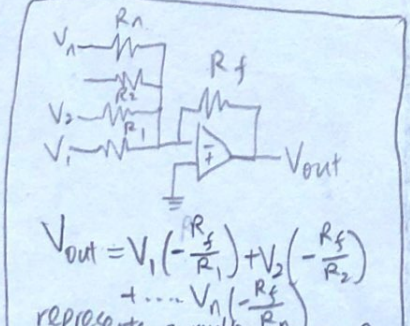
$\hat{x} = (A^T A)^{-1} A^T \vec{y}$

Superposition

short V_s, open I_s

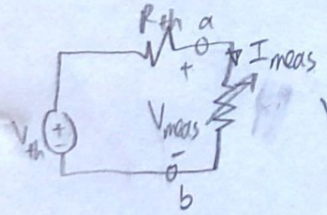
$I = 0$ means open

$V = 0$ means closed



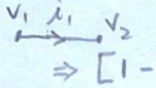
$V_{out} = V_1\left(-\frac{R_f}{R_1}\right) + V_2\left(-\frac{R_f}{R_2}\right) + \dots V_n\left(-\frac{R_f}{R_n}\right)$

represents a multiplication of one row of A w/ voltage vector $x = [V1 \dots V_n]$



KVL: $V_{th} = (I_{meas} \cdot R_{th} + V_{meas}) \cdot i$

$\bar{x}$ ≡ mean of x

Incidence Matrix ex 

⇒ [1 -1]

1) Leaving +, entering −

2) columns are nodes, rows are edges

Left multiply a vector by $\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$ to rotate by θ

$A = P \Lambda P^{-1}$

Λ = eigenvalues

P = eigenvectors

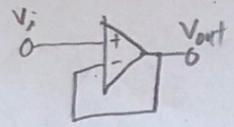
$\text{proj}_u \vec{w} = \frac{\langle \vec{w}, \vec{u} \rangle}{\|u\|} \vec{u} = \frac{\langle \vec{w}, \vec{u} \rangle}{\|u\|^2} \vec{u}$

if $\|u\| = 1$ $\text{proj}_u \vec{w} = \langle \vec{w}, \vec{u} \rangle \vec{u}$

Scanned with CamScanner

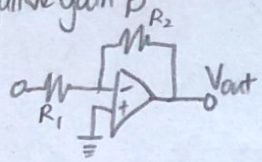
Op-Amps

Key Phrase "Isolate load, zero output/source resistance"
Use buffer



$V_{out} = V_i$

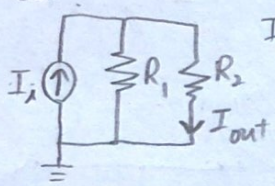
Key Phrase : "negative gain β"
Use inverting amp



$V_{out} = -\frac{R_2}{R_1} V_i$

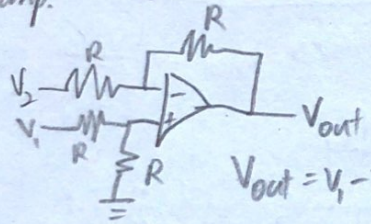
$\beta = -\frac{R_2}{R_1}$

Key Phrase: "divide current by a factor of α, use R"
Use resistive current divider



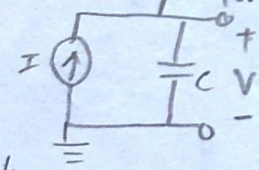
$I_{out} = \frac{R_1}{R_1+R_2} I_i$

Key Phrase "subtract 2 voltage V_1, V_2"
Use difference amp.



$V_{out} = V_1 - V_2$

Key Phrase "creat a voltage that is linear w/ respect to time or a voltage ramp w/ slop β"
Use current source w/ capacitor



$I = C\frac{dV}{dt}$

$I = C\frac{V_1 - V_2}{t_1 - t_2}$ $\quad \frac{I(t_1 - t_2)}{C} = V_1 - V_2$

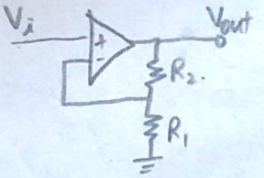
$V = \frac{I}{C}t$ $\quad \beta = \frac{I}{C}$

$P = IV$

$C = \varepsilon_0 \frac{A}{d}$

Current out of a node is positive

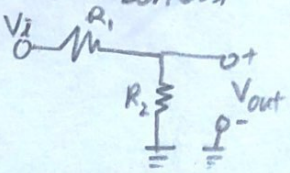
Key Phrase : "positive gain β"
Use nonverting amp.



$V_{out} = (1 + \frac{R_2}{R_1}) V_i$

$\beta = (1 + \frac{R_2}{R_1})$

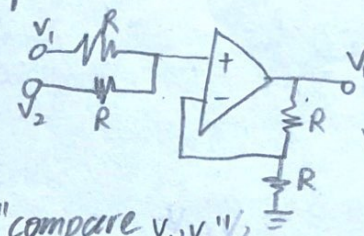
Key Phrase : "divide voltage by factor β"
Use voltage divider (R_n in series)



$V_{out} = \frac{R_2}{R_1+R_2} V_i$

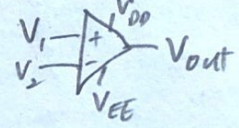
$\beta = \frac{R_2}{R_1+R_2}$

Key Phrase : "add two voltages V_1, V_2"
Use summing amp.



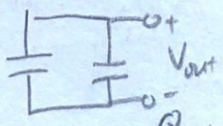
$V_{out} = V_1 + V_2$

Key Phrase : "compare V_1, V_2"

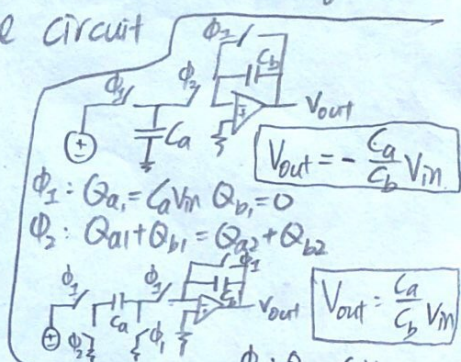


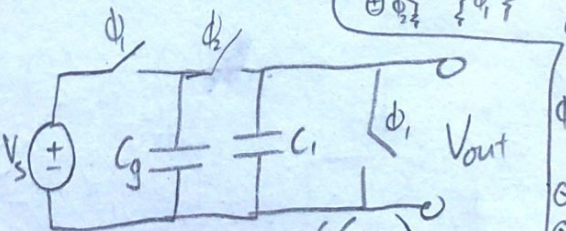
$V_{out} = \begin{cases} V_{DD} & \text{if } V_1 > V_2 \\ V_{EE} & \text{if } V_1 < V_2 \end{cases}$

For circuits, nullspace means a set of valid voltage and current values for the circuit



$V_{out} = \frac{Q_{tot}}{C_{tot}}$





$V_{out} = V_s\left(\frac{C_g}{C_g + C_1}\right)$

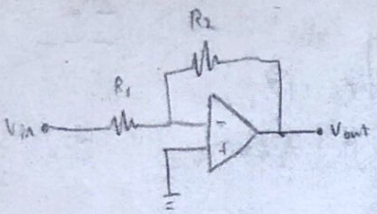
V_{out} increases as C_g increases, vice versa (not linearly)

Scanned with CamScanner

Cheat Sheet

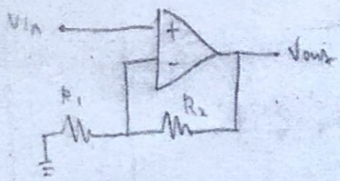
Amplifiers

Inverting



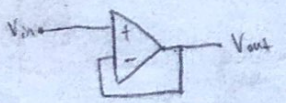
$$V_{out} = \left(-\frac{R_2}{R_1}\right) V_{in}$$

non-inverting



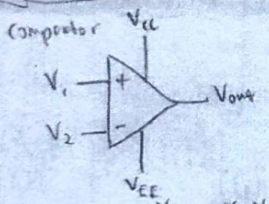
$$V_{out} = \left(1 + \frac{R_2}{R_1}\right) V_{in}$$

Buffer



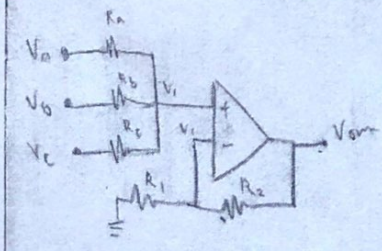
$$V_{out} = V_{in}$$

Comparator



$$V_{out} = \begin{cases} V_{CC} & \text{if } V_1 > V_2 \\ V_{EE} & \text{if } V_1 < V_2 \end{cases}$$

Voltage Summer

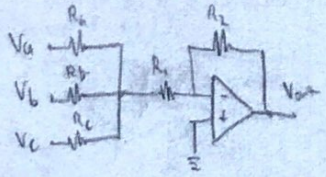


$$\frac{V_1 - V_a}{R_a} + \frac{V_1 - V_b}{R_b} + \frac{V_1 - V_c}{R_c} = 0A$$

$$V_1 = (R_a \| R_b \| R_c)\left(\frac{V_a}{R_a} + \frac{V_b}{R_b} + \frac{V_c}{R_c}\right)$$

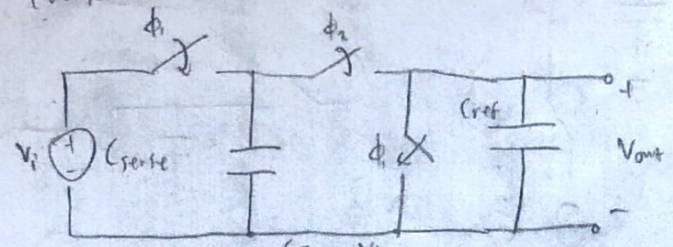
$$\frac{V_1 - 0V}{R_1} + \frac{V_1 - V_{out}}{R_2} = 0A$$

$$V_{out} = \left(1 + \frac{R_2}{R_1}\right)(R_a \| R_b \| R_c)\left(\frac{V_a}{R_a} + \frac{V_b}{R_b} + \frac{V_c}{R_c}\right)$$

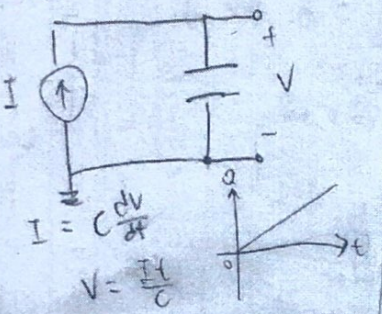


$$V_{out} = \left(-\frac{R_2}{R_1}\right)(R_a \| R_b \| R_c)\left(\frac{V_a}{R_a} + \frac{V_b}{R_b} + \frac{V_c}{R_c}\right)$$

Charge-sharing

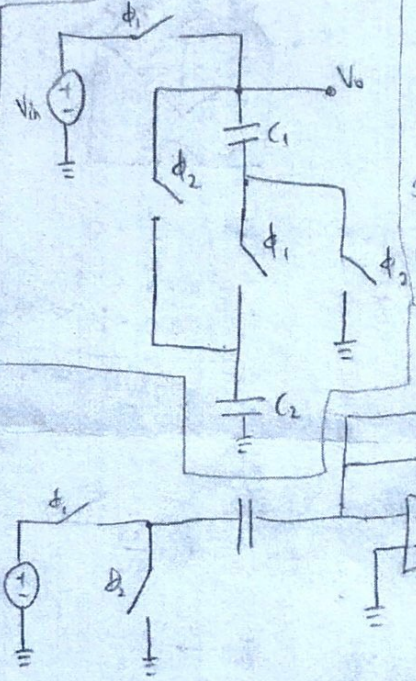


$$V_{out} = \frac{C_{sense} V_i}{C_{sense} + C_{ref}}$$



$$I = C\frac{dV}{dt}$$

$$V = \frac{It}{C}$$



$$C_{total} = \frac{C_1 C_2}{C_1 + C_2}$$

$$V_{o,\phi_1} = V_{in}$$

$$Q_{C_1,\phi_1} = Q_{C_1,\phi_1} = \frac{C_1 C_2 V_{in}}{C_1 + C_2}$$

$$Q_{C_1,\phi_1} + Q_{C_1,\phi_1} = (C_1 + C_2) V_{o,\phi_2}$$

$$V_{o,\phi_2} = \frac{2 C_1 C_2}{(C_1 + C_2)^2} V_{in}$$

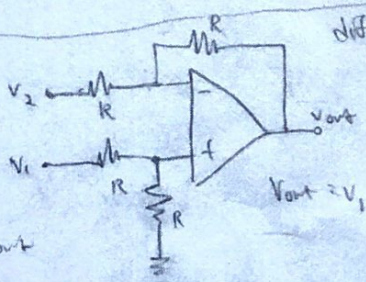
$$Q_{C_1,\phi_1} = V_{in} C_1 \qquad Q_{C_2,\phi_1} = 0V$$

$$V_{C_1,\phi_1} = V_{in} \qquad V_{C_2,\phi_1} = 0V$$

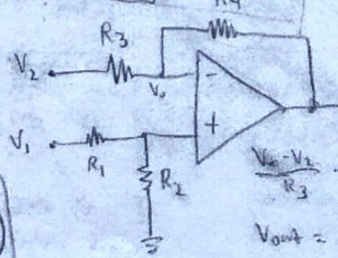
$$Q_{C_1,\phi_2} = 0C \qquad Q_{C_2,\phi_2} = V_{in} C_1$$

$$V_{C_1,\phi_2} = 0V \qquad V_{C_2,\phi_2} = \left(\frac{C_1}{C_2}\right) V_{in} = V_{o,\phi_2}$$

differencial amp



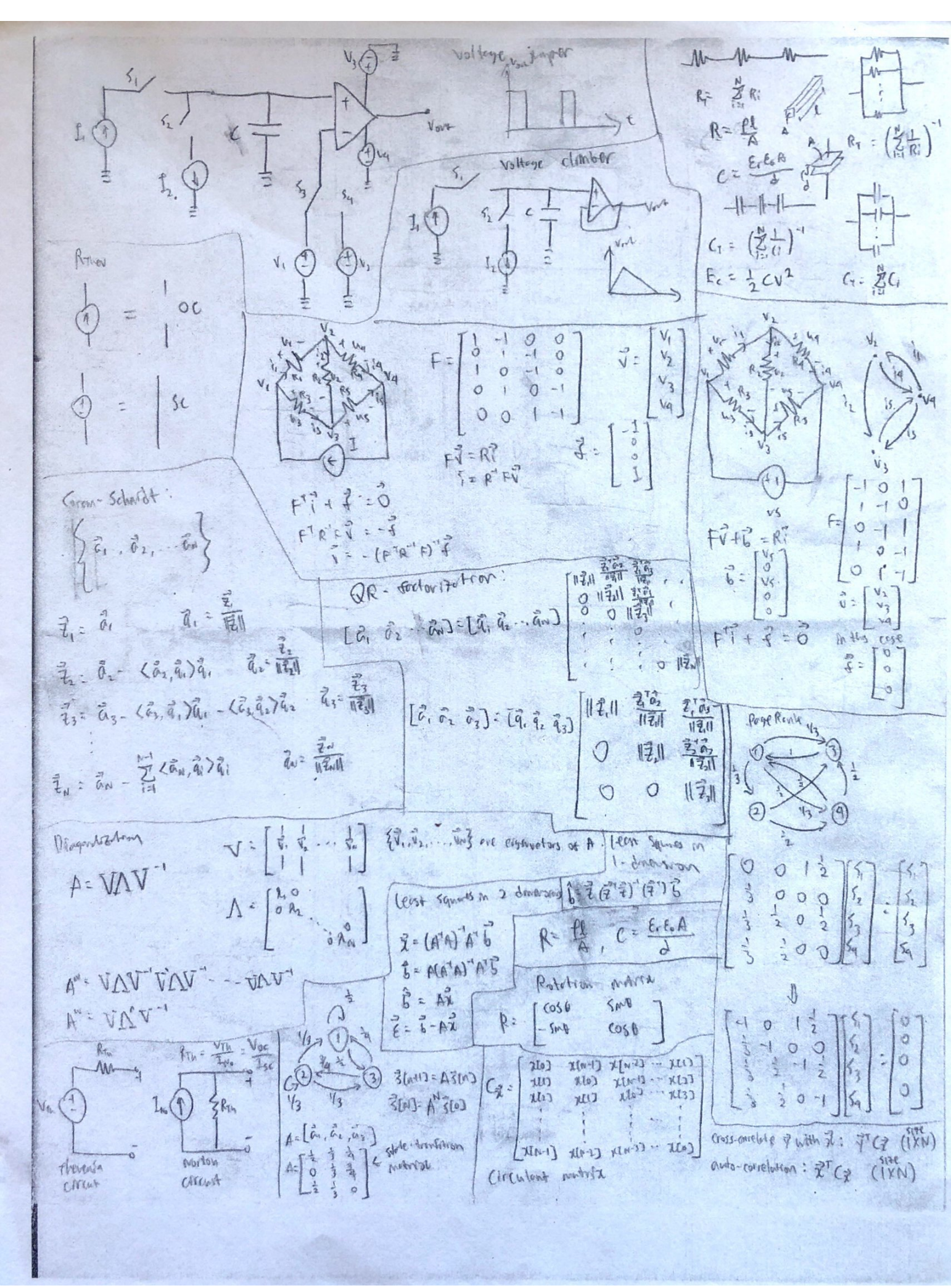
$$V_{out} = V_1 - V_2$$



$$\frac{V_o - V_2}{R_3} + \frac{V_o - V_{out}}{R_4} = 0A$$

$$V_{out} = \left(1 + \frac{R_4}{R_3}\right)\left(\frac{R_2}{R_1 + R_2}\right) V_1 - \frac{V_2}{R_3}$$

Scanned with CamScanner



Scanned with CamScanner

GD

$\theta^{t+1} = \theta^t - \rho(t)\left(\nabla_\theta L(\theta)|_{\theta=\theta^t}\right)$

$\frac{\partial \sigma(x)}{\partial x} = \sigma(x)[1-\sigma(x)]$

SGD

$\nabla_\theta L(\theta) \approx \frac{1}{|B|}\sum_{i\in B}(\sigma(\phi(x_i)^T\theta) - y_i)\phi(x_i)$

$E[X] = \sum_{x\in X} xP(x)$

if independent

$E[XY] = E[X]E[Y]$

$Var(X) = E[(X-E[X])^2]$

$= \sum_{x\in X}(x-E[X])^2 P(x)$

$= E[X^2] - E[X]^2$

if independent

$Var(X+Y) = Var(X) + Var(Y)$

$SD(x) = \sqrt{Var(x)}$

$SD[aX+b] = |a|\,SD[X]$

$X \sim Bern(p)$

$E[X] = p$

$Var[X] = p(1-p)$

$Var[\bar{X}] = \frac{p^*(1-p^*)}{n}$

Linear model

$L_D(\theta) = \frac{1}{n}\sum_{i=1}^{n}(Y_i - f_\theta(x_i))^2$

$\theta = (X^TX)^{-1}X^TY$

One-hot encoding: categorical

Bag-of-words & N-gram: Text data

Custom features: domain knowledge.

Bias: Expected deviation b/w predicted and true

Variance: ← beyond control

Observation variance: variability of random noise in process trying to model.

Estimated model variance

$E[(y - f_\theta(x))^2]$

= "noise" + Bias^2 + Variance

Elastic net = L1 + L2 norm.

optimize θ use training err

best λ use CV error

best degree of poly use CV err.

evalute quality of model use test error

standardization and don't regularize intercept terms

Supervised: regression classification

Unsupervised: dimen. reduction clustering

Error rate

sensitivity: TP rate B %

of + that correctly predicted (Type 1)

specificity: TN rate (type 2)

Confusion matrix

truth \ prediction	F	T
F	n_{FF}	n_{FT}
T	n_{TF}	n_{TT}

Specifity: $n_{FF}/(n_{FF}+n_{FT})$

Sensitity: $n_{TT}/(n_{TF}+n_{TT})$

run bernulli query will give binomial

between 13 and 17 can use

$13 < Hour(t,st) < 17$ and $\exp(-(localHour(t,st)-15)^2$

single-linkage clustering

↑ # sample ≠ ↓ bias

logistic regression

$\hat{P}_\theta(y=1|x) = \sigma(\phi(x)^T\theta)$

$= \frac{1}{1+\exp(-\phi(x)^T\theta)}$

log odd

Scanned with CamScanner

KL Divergence

$$D_{KL}(P \| \hat{P}_\theta) = \sum_{K=1}^{K} P(y=k|x) \log\left(\frac{P(y=k|x)}{\hat{P}_\theta(y=k|x)}\right)$$

argmin.

minimize average cross entropy loss

$$\underset{\theta}{\text{argmin}} \frac{1}{n} \sum_{j=1}^{n} \sum_{k=1}^{K} -P(y_i=k|x_i) \log\left(\hat{P}_\theta(y_i=k|x_i)\right)$$

cross-entropy

$$\underset{\theta}{\text{argmin}} \frac{1}{n} \sum_{i=1}^{n} y_i \phi(x_i)^T\theta + \log\left(\sigma(-\phi(x_i)^T\theta)\right)$$

GD

One-vs-rest: train separate binary classifier for each class

soft-max

$$E[(y-\hat{f}_\theta(x))^2] = E[(y-h(x))^2] + E[(h(x)-\hat{f}_\theta(x))^2]$$

noise

$$+ E[(\hat{f}_\theta(x) - E(\hat{f}_\theta(x))^2]$$

bias

variance.

Boostrap

same # as original sample

Central Limit Theorem

$e \perp \hat{y}$ and

$e \perp X(\hat{\beta}-\beta)$

dummy variable only one is 1

$$0.95 = P(\hat{\theta} - q_{0.975} SE(\hat{\theta}) \le \theta \le \hat{\theta} - q_{0.025} SE(\hat{\theta}))$$

Null model: prob model for explaining chance process

Test Stat: summary, under null model it has sampling dist.

Null hypothesis: what is expected due to chance

Fact table has multiple foreign key

dimension table has primary (only)

Data Lake: store copy of all data, schema on read

record → map → key/value → reduce → key/value

Map: deterministic

reduce: commutative & associative

logistic regression derived from bernoulli likelihood.

Variance refers to variability in prediction due to variability in training data.

↑ # samples ↓ variance

star schemas ↓ redundancy

data warehouse prefer clean over complete, data lake otherwise.

SQL & dataframe replace mapreduce.

stat query pattern:

- eliminate need for end-user to get all data.
- push task closer to storage
- good for MR and SQL
- alternative is to acquire sample on end-user

Two numeric: scatter plot

numeric vs. small int: side-by-side box

histogram: distribution of large # of obs

bar plot: count of categorical

mosacis: 2 categorical.

line: trend over time.

/ don't need to escape

log x, √x, y^2

database is collection of data

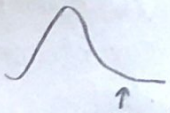
SQL one = sign

address over-plotting: transparency, smoothing, faceting, contour plotting

Scanned with CamScanner

match a string that contains only lowercase letters and numbers
'^[a-z0-9]*$'
*? non-greedy
.unique() for panda Series
.argmax() for Series to get index of max value.

pivot_table
aggr="count"
fill_value=0.0



SRS: random sample where every possible subset of n has same chance to appear in sample

Bayes: $P(A|B) = P(A)P(B|A)/P(B)$
Chain: $P(A_1,A_2,A_3) = P(A_3|A_2,A_1)P(A_2|A_1)P(A_1)$

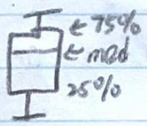
sns.barplot(x=..., y=..., hue=..., data=df)
Groupby makes granularity more coarsed
Series: named column of data w/ index
df["col_name"] returns series
df[["col_name"]] returns df
data cleaning: process of transforming raw data to facilitate analysis
Exploratory Data Analysis (EDA)
- Process of transforming, visualizing and summarizing data to:
  - Build understanding of data
  - id. and address potential issues
  - inform subsequent analysis
  - discover potential hypothesis

TSV and CSV
→ record are delimted by new line
→ fields are delimited by ','
JSON.
→ maps to python dictionaries
→ record can have diff. fields

Quantitative data
withing meaning ratios or intervals (i.e. price)
Categorical data
Ordinal: category w/ order (level of edu)
Nominal: no order (product type)

Granularity
- What does each record represents

Visualizing Univariate Relationships
- Quantitive Data
  - Histograms, box plots, rug plots, smoothed interpolation (KDE)
- Nominal & Ordinal Data.
  - Bar plots (sorted by frequency)

Box chart: ← outlier


Outliers are 1.5 * IQR away from lower or upper quartiles

Bar chart → compare nominal and ordinal data.
Visualizing multivariate relationships
- Conditioning on a range of values and construct side-by-side box or bar

Dot plot focus on comparison of values
Jiggling baseline = baseline shifting across bins
Power transformation effective when max/min > 5

KDE
$\hat{f}(x) = \frac{1}{n}\sum_{i=1}^{n} K_h(x-x_i)$

Smoothing scatter plots
$g(x) = \sum_{i=1}^{n} \frac{K_h(x-x_i)}{\sum K_h(x-x_i)} y_i$

Jitter: add random noise so values aren't plotted on top of each other
Inferential statistics - estimate properties of the population.
Gaussian Kernel: $K_\alpha(r) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{(-r^2/2\sigma^2)}$
↑ α ↑ effect of data from far away.
weighted combination of all y values.
Big n (many rows): aggr & smoothing, transparency
Big p (many col): faceting, create new hybrid columns that summarize multi. columns

Scanned with CamScanner

$Var(x) = p(1-p)$
for binormal X with p for 1
.loc[,]
can be boolean list

hw 3, on weekdays
casual < # registered
registered has bi-modal

open closing
read_html looks for <table></table>
Get - parameters passed in URI
Post - parameters passed in URL and Body

Response status codes
100s informational
200 success
300s Redirection or conditional
400s Client Error
500s Internal Server Error or Broken Request
<light/> ← empty

REST
- Stateless, cacheable, uniform interface.

Relational database
tuple (row), attribute (col)
FROM table_name
[INNER | {LEFT | RIGHT | FULL} {OUTER}] JOIN table_name
ON qualification_list
Use IS NULL

NOT

T	F
F	T
N	N

And

	T	F	N
T	T	F	N
F	F	F	F
N	N	F	N

Or

	T	F	N
T	T	T	T
F	T	F	N
N	T	N	N

Squared Loss
$L(\theta, y) = (y-\theta)^2$

Abs. Loss
$L(\theta, y) = |y-\theta|$ less prone to outliers

Huber Loss
$$L_\alpha(\theta, y) = \begin{cases} \frac{1}{2}(y-\theta)^2 & |y-\theta| < \alpha \\ \alpha(|y-\theta| - \frac{\alpha}{2}) & \text{otherwise} \end{cases}$$
extra parameter, α

Average loss
$L(\theta, D) = \frac{1}{n}\sum_{i=1}^{n} L(\theta, y_i)$

Convex
2nd derivative is positive
$t f(a) + (1-t) f(b) \geq f(ta + (1-t)b)$.
$\forall a, \forall b, t \in [0,1]$, line lays above func.
Sum convex = convex
For Abs Loss $\frac{\partial}{\partial \theta} L(\theta, y) = -\frac{1}{n}\sum_{i=1}^{n} \text{sign}(y_i - \theta)$

Gradient Descent Algorithm
$\theta^{t+1} \leftarrow \theta^t - \alpha \nabla L(\theta^t)$

.count()
.sort_value()
plt.xlabel, ylabel, yaxis, x-tick
sns.distplot

Stratified Sample
population is partition into strata and a SRS is taken within each strata.

Cluster Sample
divide population into groups, take an SRS of groups, and elements from each group are selected.

Boostrapping = test that relies on random sampling w/ replacement

Applying power transformation:
- Help visualize highly skewed distribution
- Help straighten relationships b/w pairs of variables

Scanned with CamScanner

C, pass in *int we need to pass in addr
& to get point
* to dereference
accessing members queue→data

fork() returns pid of child in parent
0 for child
-1 for failure.

stack: grows downward, LIFO
heap: grows upward, dynamic allocation
wait: for a child process
exec: UNIX command.
sys call: program request service from OS kernel

File descriptors: 0 stdin, 1 stdout, 2 stderr (close() return 0 if success)
open(const char *path, flg): sys call to open a new file and obtain file descriptor
read(int fildes, void *buf, size_t nbytes): read nbytes data into buf, file offset += nbytes
write(int fildes, void *buf, size_t nbytes): similar to read.
lseek(int fildes, off_t offset, int whence): move offset of file descriptor.
SEEK_SET - The offset is set to offset.
SEEK_CUR - The offset is set to cur offset + offset.
SEEK_END - offset set to size of file + offset
dup2(int fildes, int fildes2): make fildes alias of fildes 2.

Signals: software interrupt, communicate info to a process about others.
signal(int signum, void (*handler)(int)): signal handling func.
SIGINT 2, SIGQUIT 3

threads
share same address space.
pthread_create, create child
pthread_wait, wait on child
pthread_yield, yield to scheduler

atomic operation: must execute to completion.
lock: acquire & release
int test_and_set(int *value) {
  int result = *value;
  *value = 1
  return result;
}

Priority inversion: higher p thread blocked by lower p thread cuz hold resources
Priority donation: Donate priority to lower p thread that has resource to prevent p. inversion.
Hyperthreading: sub-linear speedup, duplicates register state to make "second thread", more inst to run.

Process	Process	thread
Creation	fork()	pthread_create()
Page table	distinct	same
Stack	separate	separate, but accessible
Heap	separate	shared
File descrip	separate	shared
Sync.,	wait(), waitpid()	pthread_join(), semaphores, locks

Multiprocessing = multiple CPUs or core or hyperthreading
multiprogramming = multiple processes
Cooperating threads adv: share resource, speedup, modularity
use lock.acquire() and lock.release() around critical section.
use "guard" value to busy-wait less

Monitors
a lock and 0 or more CV
lock: provides mutex to shared data.
Condition variable: a queue of threads waiting
wait(&lock), signal(), broadcast() ↑ release lock & sleep

Readers/writers
Reader()
Wait until no writers.
access data
wait up one waiting writer.
Writer()
Wait until no active reader or writer
Wake up waiting r or w.
Variables
AR, WR, AW, WW, okToRead/Write

Correctness constraint ex
If queue is full, addToQueue() must wait

Pintos
Each process consist of one thread.
timer interrupts at each tick let kernel preempt
idle thread is placeholder thread allow switch_threads() to run and scheduling
when dis.intempt, handler won't be called
Upon creation of thread, stack initialized 2 dummy frames.
Kernel panics: internal error that can't recover from.
stack in pintos must fit into 4k page

User stack A —(timer interrupt, disable int)→ kernel stack A → kernel stack B —(enable int)→ user stack B

Locks	Semaphores	monitors	send/receive
Load/store	dis. ints	test&set	compare&swap
HW			

Semaphores
semaphore.wait(&lock);
P(), waits for semaphores to become positive, then decrement by 1 - like wait()
V(), increment by 1, waking up a waiting P, like singal().
Mutual exclusion (init value = 1)
Scheduling constraint (init = 0)
i.e. allow thread 1 to wait for signal from thread 2
Order of P important

Monitor
Lock lock;
Condition dataready;
Queue queue;

AddToQueue(item):
lock.Acquire()
queue.enqueue(item)
dataready.signal()
lock.Release()

RemoveFromQueue():
lock.Acquire()
while(queue.isEmpty()) dataready.wait(&lock)
item = queue.dequeue()
lock.released()
return item.

Hoare: if(queue.isEmpty())
Mesa: while() because other thread can run b/f signaled thread

Scheduling
guaranteed to run
FIFO (FCFS)
- schedule as they come
Shortest Remaining Time First.
- preemptively schedule process with shortest remaining time
- optimal, impossible
Strict Priority
- run highest priority
Round-Robin
- run processes in looping order for fixed quanta.
RR - longest thread will have longest wait time and completion time
Lottery scheduling
- give short job more ticket
Earliest Deadline First: for real time, meet deadline
Multi-level feedback: multiple queues of different priorities and algorithm

malloc(sizeof(struct x));
Pinto implementation of semaphore won't work on multiprocessor cuz dis.int wouldn't work
thread_current(): take stack pointer and page align it.
Use pthread_mutex_init for locks and pthread_cond_init for CV

Scanned with CamScanner

Sockets programming

Patrick
agl
Fri 3-4 pm

```
void client(int sockfd) {
  int n;
  char sndbuf[MAXIN]; char rcvbuf[MAXOUT];
  getreq(sndbuf, MAXIN);
  while(strlen(sndbuf)>0) {
    write(sockfd, sndbuf, strlen(sndbuf));   send
    memset(rcvbuf, 0, MAXOUT);   clear
    n=read(sockfd, rcvbuf, MAXOUT-1);   receive
    write(STDOUT_FILENO, rcvbuf, n);   echo
    getreq(sndbuf, MAXIN);   prompt
  }
}
```

```
void server(int sockfd) {
  char reqbuf[MAXREQ];
  int n;
  while(1) {
    memset(reqbuf,0,MAXREQ);
    n = read(consockfd, reqbuf, MAXREQ-1);   Recv
    if (n<=0): return;
    n = write(STDOUT_FILENO, reqbuf, strlen(reqbuf));
    n = write(consockfd, reqbuf, strlen(reqbuf));   echo
  }
}
```

```
char *getreq(char *inbuf, int len) {
  memset(inbuf,0,len);
  return fgets(inbuf, len, stdin);
}
```

- Forms 2-way pipes b/w processes
- Port #: 0-1023 "system" ports
  1024-49151 "registered" ports
  49152-65535 "dynamic" or "private"

Server: call listen() and multiple accept(), #local processes, port
client: call connect()
both creat socket

Connections involves: [client addr, client port, server addr, server port, protocol]

Server protocol

```
listen(lstnsocket, MAXQUEUE);
while(1) {
  consockfd = accept(lstnsockfd, &cli_addr, &clilen);
  cpid = fork()
  if (cpid >0) {close(consockfd)}
  else {
    close(lstnsockfd);
    server(consockfd);
    close(consockfd);
    exit(EXIT_SUCCESS);
  }
}
close(lstnsockfd)
```

init Semaphore
Semaphore x = new semaphore(#)

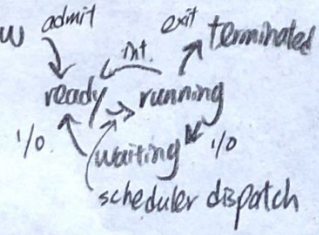
Process Control Block includes

- P. state, P. #, Program Counter
- registers, mem, limit, open files

TCB includes:
CPU registers
execution stack:
parameters, temp variables

Address space

stack
↓
↑
heap
static
code



User mode → kernel mode
exit, exception, interrupt, syscall.
kernel → user
exec, rtn, rfi.

Therac-25
use software for safety interlock and error cause death, race condition.

Requirement for deadlock

mutual exclusion, non-preemptable resources, hold and wait, circular chain waiting.

spinlock (busy wait) is more efficient when CS is short or CS is rarely contested.

semaphore is commutative, CV is not.

Read

character oriented
fputc, fputs, fgetc, fgets
block oriented
fread, fwrite
formatted
fprintf, fscanf

Socket programming

child process handle socket, parent process birth new socket

```
int fd = socket(PF_INET, SOCKET_STREAM, 0);
bind(fd, &addr, sizeof(addr));
listen(fd, 0);                                  Server
while(1): int c = accept(fd); handle(c);
```

The order we declare members of struct thread matters because "magic" member
Threads in same address can access any mem addr.
binary semaphore = lock
need to write flush or close after fputs.
interrupts can be used to implement locks.
wait() in CS releases lock and sleeps
File descriptors are not destoryed by execv()
Dual mode prevents use program from overwritting kernel data
Every process has 1 orinigal thread
test&set return 0 if free
wait(&status) returns pid_t

```
send(sockfd, "message", length, 0);
to kill a connect if read 'q' in socket
n = read(newsockfd, reqbuf, MAXREQ2);
for (int i=0; i<n; i++)
  if (reqbuf[i] == 'q'):
    kill(pid, SIGHUP)
```

Scheduling overhead

time slice / (time slice + context switch time)

put sema_down outside critical section to prevent deadlock

In order for user to use kernel func, kernel need to assign func to free interrupt #.
making I/O call → turn from running to waiting/blocked
execv replace current process

```
execv("/bin/ls", null)
```

gives contents of current directory.

Scanned with CamScanner

Expectation Maximization (EM)

1. initialize θ^0
2. E-step (soft imputation):
   set $q^{t+1} = \arg\max_q F(q, \theta^t)$
   $q^{t+1}(z_i = k | x_i) := p(z_i = k | x_i; \theta^t)$
3. M-step (parameter estimation): set
   $\theta^{t+1} = \arg\max_\theta F(q^{t+1}, \theta)$
   $= \arg\max_\theta E_{q^{t+1}}(L_c(x, z; \theta))$
4. Repeat 2,3 til convergence

EM for MOG $x|z \sim N(\mu_z, \Sigma_z)$, $p(z=k) = \alpha_k$

E step:

$$q^{t+1}(z_i = k | x_i) = \frac{\alpha_k^t p(x_i | z_i = k; \theta^t)}{\sum_{j=1}^K \alpha_j^t p(x_i | z_i = j; \theta^t)}$$

M step:

$$\mu_k^{t+1} = \frac{\sum_{i=1}^N q_{ki}^{t+1} x_i}{\sum_{i=1}^N q_{ki}^{t+1}}$$

$$\Sigma_k^{t+1} = \frac{\sum_{i=1}^N q_{ki}^{t+1} (x_i - \mu_k^{t+1})(x_i - \mu_k^{t+1})^T}{\sum_{i=1}^N q_{ki}^{t+1}}$$

$$\alpha_k^{t+1} = \frac{1}{N} \sum_{i=1}^N q_{ki}^{t+1}$$

Jensen's inequality

$f(E(x)) \le E(f(x))$

$= \alpha E(x) + b = E(ax + b)$

$= E(L(x)) \le E(f(x))$

SVM (discriminative)

find d-1 dimension hyperplane

Hard-margin

$$\max_{m, w, b} m \quad \text{s.t. } y_i \frac{(w^T x_i - b)}{\|w\|_2} \ge m \ \forall i, \ m \ge 0$$

let $w' = \frac{w}{\|w\|_2 m}$ and $b' = \frac{b}{\|w\|_2 m}$

$$\max_{w', b'} \frac{1}{\|w'\|_2} \quad \text{s.t. } y_i (w'^T x_i - b') \ge 1 \ \forall i$$

Soft margin SVM

$$\min_{w, b, \xi_i} \frac{1}{2}\|w\|^2 + C \sum_{i=1}^n \xi_i \quad \text{s.t. } y_i (w^T x_i - b) \ge 1 - \xi_i \ \forall i, \ \xi_i \ge 0 \ \forall i$$

	small C	large C
Desir	max. margin	keep ξ_i small
Danger	underfit	overfitting
Outlier	less sensitive	more sensitive

LDA w/ equal covariance is equivalent to first CCA direction, $\Sigma_c^{-1}(\mu_c - \mu_0) = d^{-2} \varepsilon e_1$, same direction as e_1

soft SVM optimize hinge and ridge

step is not convex

decision tree easily deal w/ missing value.

NN w/ one layer of step can express everything except piecewise linear function

SVM as Tikhonov Regularization

optimize 0-1 step loss:

$$L_{step}(y, w^T x - b) = \begin{cases} 1 & y(w^T x - b) < 0 \\ 0 & y(w^T x - b) \ge 0 \end{cases}$$

Hinge loss $= \max(1 - y(w^T x - b), 0))$

$$\min_{w,b} \frac{1}{2}\|w\|^2 + C \sum_{i=1}^n \max(1 - y_i(w^T x_i - b), 0)$$

Duality primal

$$p^* = \min_x \max_{\lambda \ge 0, v} f_0(x) + \sum_{i=1}^m \lambda_i f_i(x) + \sum_{j=1}^n v_j h_j(x)$$

$$\text{s.t. } f_i(x) \le 0, \ h_j(x) = 0, \ i = 1 \dots m, \ j = 1 \dots n$$

$= L(x, \lambda, v)$ Langrangian

dual $d^* = \max_{\lambda \ge 0, v} \min_x L(x, \lambda, v)$

$g(x) = \min_x L(x, \lambda, v)$

weak duality $p^* \ge d^*$

strong duality $p^* = d^*$

KKT conditions

$f_i(x) \le 0, \ \forall i \in \{1, \dots m\}$

$h_j(x) = 0, \ \forall j \in \{1, \dots n\}$

$\lambda_i \ge 0, \ \forall i \in \{1, \dots m\}$

complementary slackness

$\lambda_i f_i(x) = 0, \ \forall i \in \{1, \dots m\}$

stationary

$$\nabla_x f_0(x) + \sum_{i=1}^m \lambda_i \nabla_x f_i(x) + \sum_{j=1}^n v_j \nabla_x h_j(x) = 0$$

Dual of SVM

Lagrangian of SVM

$$L(w, t, \xi, \alpha, \beta) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^n \alpha_i y_i (w^T x_i - t) + \sum_{i=1}^n \alpha_i + \sum_{i=1}^n (C - \alpha_i - \beta_i) \xi_i$$

Dual is $\max_{\alpha \ge 0, \beta \ge 0} g(\alpha, \beta)$

$g(\alpha, \beta) = \min_{w, t, \xi} L(w, t, \xi, \alpha, \beta)$

KKT condition $\frac{\partial L}{\partial w_i} = \frac{\partial L}{\partial x} = \frac{\partial L}{\partial \xi_i} = 0$

$\frac{\partial L}{\partial w_i} = 0 \Rightarrow w^* = \sum_{i=1}^n \alpha_i^* y_i x_i$

$\frac{\partial L}{\partial x} = 0 \Rightarrow \sum_{i=1}^n \alpha_i^* y_i = 0$

$\frac{\partial L}{\partial \xi_i} = 0 \Rightarrow C - \alpha_i^* - \beta_i^* = 0 \Rightarrow 0 \le \alpha_i^* \le C$

$L(w, t, \xi, \alpha, \beta) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^n \alpha_i^* y_i (w^T x_i) + \sum_{i=1}^n \alpha_i^*$

$g(\alpha^*, \beta^*) = \alpha^{*T} 1 - \frac{1}{2} \alpha^{*T} Q \alpha^*$

$Q_{ij} = y_i (x_i^T x_j) y_j \quad Q = (\text{diag } y) X X^T (\text{diag } y)$

$$\max_\alpha \alpha^T 1 - \frac{1}{2} \alpha^T Q \alpha \quad \text{s.t. } \sum_{i=1}^n \alpha_i y_i = 0, \ 0 \le \alpha_i \le C, \ i = 1, \dots n$$

$\alpha_i^* = 0 \Rightarrow \beta_i^* = C \Rightarrow \xi_i = 0$, i on or outside margin.

$\alpha_i^* = C \Rightarrow \beta_i^* = 0 \Rightarrow \xi_i^* = 0$ or nonzero, i on or inside margin

$0 < \alpha_i^* < C, \ \xi_i = 0$, on margin

k-Nearest Neighbor (Lazy training)

$k \uparrow \Rightarrow$ underfit, $\uparrow$ bias, $\downarrow$ variance

$\text{Var}[h(z)] = \frac{\sigma^2}{k}$

Non-parametric method

parameters grow w/ n.

1-NN upper bound by $2\epsilon^*$, ϵ^* = bayes error

as $\frac{k}{n} \to 0$, kNN error $\Leftrightarrow$ bayes error

Curse of dimensionality

$$\frac{(r-\epsilon)^d}{r^d} \approx e^{-ed/r} \xrightarrow{d \to \infty} 0,$$

need to look at larger ball

Improving k-NN

- More training data
- reduce dimensions

Minkowski distance

$$D_p(x, z) = \left(\sum_{i=1}^n |x_i - z_i|^p\right)^{1/p}$$

$\|\Phi(x) - \Phi(z)\|_2^2 = k(x,x) - 2k(x,z) + k(z,z)$

$D(x, z) = \sqrt{k(x,x) - 2k(x,z) + k(z,z)}$ kernelize

CNN 2

$$\frac{\partial L[i,j]}{\partial G_c[x,y]} = I_c[i+x, j+y]$$

$$\frac{\partial L[i,j]}{\partial I_c[x,y]} = G_c[x-i, y-j]$$

Perceptron has 0 training error on linearly-separable data.

Kernel matrix is symmetric

CNN

$$(I * G)[x,y] = \sum_{a=0}^{w-1} \sum_{b=0}^{h-1} \sum_{c \in \{1,2,3\}} I_c[x+a, y+b] \cdot G_c[a,b]$$

output $L = I * G$ is $(W - w + 1) \times (H - h + 1) \times 1$ value

extract feature $\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$ extract vertical lines

receptive field increases

Pooling: Cross-channel pooling makes transformational invariance

Spatial pooling makes translational invariance

Scanned with CamScanner

Nonlinear Least Square

MLE formulation

$$\hat{\theta}_{MLE} = \arg\max_\theta \ell(\theta; X, y)$$
$$= \arg\max_\theta \sum_{i=1}^{n} \log p(y_i | x_i; \theta)$$
$$= \arg\min_\theta \sum_{i=1}^{n} (y_i - f(x_i;\theta))^2$$

$$\nabla_\theta G = J(\theta)^T (y - F(\theta)) = 0$$

where

$$F(\theta) = \begin{bmatrix} f(x_1;\theta) \\ \vdots \\ f(x_n;\theta) \end{bmatrix}, J(\theta) = \begin{bmatrix} \nabla_\theta f(x_1;\theta)^T \\ \vdots \\ \nabla_\theta f(x_n;\theta)^T \end{bmatrix}$$

J is Jacobian of F

Gauss-Newton Method

$$F(\theta) \approx \tilde{F}(\theta) = F(\theta^{(k)}) + J(\theta^{(k)})\Delta\theta$$
$$\Delta\theta = (J^T J)^{-1} J^T \Delta y$$
$$\theta^{k+1} = \theta^k + (J^T J)^{-1} J^T \Delta y$$

Neural Network

Multilayer perceptron

softmax $\sigma(x)_i = \frac{e^{x_i}}{\sum_{j=1}^{k} e^{x_j}}$

$$\frac{\partial \sigma(x)_i}{\partial x_j} = \sigma(x)_i (1 - \sigma(x_i))$$ if $i = j$

$$\frac{\partial \sigma(x)_i}{\partial x_j} = -\sigma(x_i)\sigma(x_j)$$ if $i \neq j$

$$\text{rank}(\tilde{W}) \leq \min_{\ell \in \{1,\dots,L\}} \text{rank}(W_\ell) \leq \min_{\ell \in \{0,\dots,L\}} n_\ell$$

Step function

$$\sigma(x) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases}$$

for universal func approximator

ReLu

$$\sigma(x) = \begin{cases} x & x \geq 0 \\ 0 & x < 0 \end{cases} = \max\{0, x\}$$

Back propagation

DP + topological order

$$z_j = \sum_i w_{ji} a_i$$
$$\frac{\partial z_j}{\partial a_i} = a_i$$
$$\frac{\partial \ell}{\partial w_{ji}} = \frac{\partial \ell}{\partial z_j} a_i$$
$$\frac{\partial z_j}{\partial a_i} = w_{ji}$$
$$\frac{\partial \ell}{\partial a_i} = \sum_{j=1}^{k} \frac{\partial \ell}{\partial z_j} w_{ji}$$

Generative classification

$$\hat{y} = \arg\max_k p(x, y=k)$$

need $P(k) = P(\text{class} = k)$

$$f_k(x) = f(x | \text{class} = k)$$
$$\hat{y} = \arg\max_k P(k) f_k(x) = \arg\max_k p(y=k|x)$$

Discriminative model

- Directly learn decision boundary, ex: SVM.

LS-SVM (not so good)

$$\arg\min_w \sum_{i=1}^{n} \|y_i - \text{sign}(w^T x_i)\|^2 + \lambda \|w\|^2$$

too hard so

$$\arg\min_w \sum_{i=1}^{n} \|y_i - w^T x_i\|^2 + \lambda \|w\|^2$$

feature extension or NN extension

$$\arg\min_w \sum_{i=1}^{n} \|y_i - w^T \phi(x_i)\|^2 + \lambda \|w\|^2$$

Multi-class extension

use one-hot vector encoding

Logistic Regression (discriminative), convex

use Sigmod $s(w^T x) = \frac{1}{1+e^{-w^T x}}$

$$\frac{\partial s}{\partial x} = s(1-s)$$ log odd

$$\arg\min_w \sum_{i=1}^{n} \|y_i - s(w^T x_i)\|^2 + \lambda \|w\|^2$$

cross-entropy (log loss)

$$p_i = s(w^T x_i)$$
$$L(w) = -\sum_{i=1}^{n} y_i \ln p_i + (1-y_i)\ln(1-p_i)$$

if $p_i = y_i$ then loss = 0

$$P(\hat{Y}_i = y_i) = p_i^{y_i} (1-p_i)^{(1-y_i)}$$

$$\hat{w}_{LR} = \arg\min_w -\sum_{i=1}^{n} y_i \ln p_i + (1-y_i)\ln(1-p_i)$$

KL Divergence

$$D_{KL}(P \| Q) = \sum_x P(x) \ln \frac{P(x)}{Q(x)}$$

$D_{KL}(P\|Q) = 0$ when $P = Q$, not then positive

$$D_{KL}(Y_i \| \hat{Y}_i) = H(Y_i, \hat{Y}_i) - H(Y_i)$$

$$= \arg\min_w \sum_{i=1}^{n} H(Y_i, \hat{Y}_i)$$

Multiclass Logistic regression

use softmax

$$\hat{y} = \max_k \sigma(Wx)_k$$
$$\sigma(Wx) = [\sigma(Wx)_1, \dots \sigma(Wx)_k]$$
$$\hat{w}_{MCLR} = \arg\min_w -\sum_{i=1}^{n} \sum_{j=1}^{k} \delta_{j, y_i} \ln\left(\frac{e^{w_j^T x_i}}{\sum_{k=1}^{K} e^{w_k^T x_i}}\right)$$
$$= \arg\min_w \sum_{i=1}^{n} H(Y_i, \hat{Y}_i)$$

Training

binary: $\nabla_w L(w) = -\sum_{i=1}^{n} (y_i - p_i) x_i$

$$w^{t+1} = w^t + \epsilon \sum_{i=1}^{n} (y_i - p_i) x_i$$

multi: $\nabla_w = \sum_{i=1}^{n} (\delta_{\ell, y_i} - P(\hat{Y}_i = \ell)) x_i$

QDA: Generative

$$\hat{\mu}_k = \frac{1}{n_k} \sum_{x_i = k} x_i \; ; \; \hat{\Sigma}_k = \frac{1}{n_k} \sum_{x_i = k} (x_i - \hat{\mu}_k)(x_i - \hat{\mu}_k)^T$$

$$\hat{y} = \arg\max_k Q_k(x)$$

where $Q_k(x) = \ln\left((\sqrt{2\pi})^d P(k) f_k(x)\right)$

$$\hat{y} = \arg\max_k \ln(P(k)) - \frac{1}{2}(x - \hat{\mu}_k)^T \hat{\Sigma}_k^{-1} (x - \hat{\mu}_k) - \frac{1}{2}\ln(|\hat{\Sigma}_k|)$$

LDA

$$\hat{\Sigma} = \frac{1}{n} \sum_{i=1}^{n} (x_i - \hat{\mu}_{x_i})(x_i - \hat{\mu}_{x_i})^T$$

share among all class

$$\hat{\Sigma} = \sum_{k=1}^{K} \frac{n_k}{n} \Sigma_k$$

decision boundary when $P(\text{class} = A | x) = P(\text{class} = B | x)$

identical distribution $\Sigma_A = \Sigma_B = \sigma^2 I$

$$(x - \hat{\mu}_A)^T (x - \hat{\mu}_A) = (x - \hat{\mu}_B)^T (x - \hat{\mu}_B)$$

w/ Prior

$$x^T(\hat{\Sigma}^{-1}(\hat{\mu}_A - \hat{\mu}_B)) + \left(\ln\left(\frac{P(A)}{P(B)}\right) - \frac{\hat{\mu}_A^T \hat{\mu}_A - \hat{\mu}_B^T \hat{\mu}_B}{2}\right) = 0$$

Nonidentical with Prior

$$\ln(P(A)) - \frac{1}{2}(x - \hat{\mu}_A)^T \hat{\Sigma}_A^{-1} (x - \hat{\mu}_A) =$$
$$\ln(P(B)) - \frac{1}{2}(x - \hat{\mu}_B)^T \hat{\Sigma}_B^{-1} (x - \hat{\mu}_B)$$

Clustering (unsupervised learning)

K-mean clustering solves for

$$\arg\min_{\{C_k\}_{k=1}^{K}, \{c_k\}_{k=1}^{K}} \sum_{k=1}^{K} \sum_{i: x_i \in C_k} \|x_i - c_k\|^2$$

$X = C_1 \cup \dots \cup C_k$

K-means algorithm

initialize c_k

while NOT converage

update partition $C_1, \dots C_k$ given c_k by assign $x \in X$ to closest one

update centroids $c_k = \frac{1}{|C_k|} \sum_{x \in C_k} x$

Soft k-means

let $r_i(k) = \sigma(z)_k$

then

$$\hat{c}_k = \arg\min_{c_k} \sum_{i=1}^{N} r_i(k) \|x_i - c_k\|^2$$
$$= \frac{\sum_{i=1}^{N} r_i(k) x_i}{\sum_{i=1}^{N} r_i(k)}$$

Mixture of Gaussians

draw z from $\{1, \dots K\}$ then draw x from $N(\mu_z, \Sigma_z)$

$$\lambda_i(\theta; x_i) = \sum_{k=1}^{K} p(x_i | z_i = k; \theta) p(z_i = k; \theta)$$
$$\ell(\theta; x) = \sum_{i=1}^{N} \log \sum_{k=1}^{K} \lambda_i(\theta; x_i)$$

Scanned with CamScanner

OLS ⇒ MLE

$\min_w \|Xw-y\|^2$

$y = Xw+\varepsilon$

$\varepsilon \sim N(0, I)$

$w = (X^TX)^{-1}X^Ty$

Ridge regression ⇒ MAP

$\min_w \|Xw-y\|^2 + \lambda\|w\|^2$ ↑λ, ↑bias

$y = Xw+\varepsilon$ ↑λ, ↓variance in MAP

$\varepsilon \sim N(0, \lambda I)$

$w = (X^TX+\lambda I)^{-1}X^Ty$

TLS

$\min_{E_x, E_y} \|[E_x E_y]\|_F^2$

$(X+E_x)w = y+E_y$

$w = (X^TX - \sigma_{d+1}^2 I)^{-1}X^Ty$

where σ_{d+1} is the least singular value of $[X\ y]$

PCA

$\max_{\|u\|=1} u^TX^TXu$

$X = U\Sigma V^T$

$u = V_1$ where V_1 is 1st col of V.

CCA

$\max_{u,v} \rho(X_{rv}^T u, Y_{ru}^T v)$

$u = W_x D_x u_d$

$v = W_y D_y v_d$

where whitening gives $W_x = V_x \Sigma_x^{-1} V_x^T$

Decorrelation steps gives $D_x = U_r$ and $D_y = V_r$

$X_w^T Y_w = U_r \Sigma_r V_r^T$, after decorrelation

$(X_w D_x)^T (Y_w D_y) = I$

Polynormal features

$\binom{\lambda+d}{d}$ terms

# of terms $= O(d^k)$

ρ and λ are hyperparameters

training err: bias²

testing err: bias² + variance

MLE

Gaussian $p(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$

$\mathrm{Var}(x) = E[(X-E[X])^2]$

$h^*_{MLE} = \arg\max_{h\in H} \log[P(y_1,\dots y_n | x_1,\dots x_n, h)]$

$\operatorname{argmax}_h \sum_{i=1}^n \log[P(y_i|x_i,h)]$

MAP

$h^*_{MAP} = \operatorname{argmax}_{h\in H} \left(\sum_{i=1}^n \log(P(y_i|x_i,h))\right) + \log[P(h)]$

↑ n, ↓ effect of bad prior. MAP ~ MLE when $n \to \infty$

Bias-Variance

$E(x;h) = \underbrace{(E[h(x;D)] - f(x))^2}_{\text{bias}^2} + \mathrm{Var}(h(x;D)) + \mathrm{Var}(N)$

WLS

$w^*_{WLS} = \arg\min_w \left(\sum_{i=1}^n w_i(y_i - X_i^Tw)^2\right)$

$= ((\Omega^{1/2}X)^T(\Omega^{1/2}X))^{-1}(\Omega^{1/2}X)^T\Omega^{1/2}y$

$= (X^T\Omega)^{-1}X^T\Omega y$

where $\Omega_{ii} = w_i$

$w^*_{WLS} = (X^T\Sigma_z^{-1}X)^{-1}X^T\Sigma_z^{-1}y$

$\Sigma_z = \begin{bmatrix}\sigma_1^2 & 0\\ 0 & \sigma_n^2\end{bmatrix}$ $w_i = \frac{1}{\sigma_i^2}$

Multivariate Gaussians

$Z = RU$ where $R \in R^{n\times n}$ and $U \in R^n$, and $U_i \overset{iid}{\sim} N(0,1)$

$\Sigma = \begin{bmatrix}\Sigma_{xx} & \Sigma_{xy}\\ \Sigma_{yx} & \Sigma_{yy}\end{bmatrix}$ $\Sigma_{xy} = \Sigma_{yx}$ if $\Sigma_{xy} > 0$ then pos. correlation, need eigenvalues $\neq 0$

$\Sigma = E[(Z-\mu)(Z-\mu)^T] = RR^T$ = cov. matrix of Z

MLE with dependent noise

$w^* = (X^T\Sigma_z^{-1}X)^{-1}X^T\Sigma_z^{-1}y$

MAP with colored Noise

$\hat{w} = \mu_w + (X^TX + \Sigma_w^{-1})^{-1}X^T(y - X\mu_w)$

Kernel Ridge Regression

$w^* = (\Phi^T\Phi + \lambda I)^{-1}\Phi^Ty$

when $n \ll d$

Kernels

$\Phi\Phi^T_{ij} = \langle\phi_i, \phi_j\rangle = \langle\phi(x_i), \phi(x_j)\rangle = k(x_i, x_j)$

$K(x,z) = (x^Tz+1)^2$

$\hat{y} = \langle\phi(z); w^*\rangle = \phi(z)^T\Phi^T(\Phi^T\Phi+\lambda I)^{-1}y = [k(x_1,z)\dots k(x_n,z)]^T(K+\lambda I)^{-1}y$

Non-kernelized $\hat{y} = \langle\phi(z), w^*\rangle = \phi(z)^T(\Phi^T\Phi+\lambda I)^{-1}\Phi^Ty$

dccn, non-kernelized method.

TLS $w^* = (X^TX - \sigma_{d+1}^2 I)^{-1}X^Ty$

$y = wx - wz_x + z_y$ $\sim N(0, w^2+1)$

$P(x,y;w) = \frac{1}{\sqrt{2\pi(w^2+1)}} e^{\left(-\frac{1}{2}\frac{(y-wx)^2}{w^2+1}\right)}$

$[X+E_x\ y+E_y]\begin{bmatrix}w\\-1\end{bmatrix} = 0$. $E[X^TX] = X_{true}^TX_{true} + E[Z^TZ]$

Eckart-Young theorem

if suppose $A \in R^{m\times n}$ has rank $r \le \min(m,n)$, let $A = U\Sigma V^T = \sum_{i=1}^r \sigma_i u_i v_i^T$

Then $A_k = \sum_{i=1}^k \sigma_i u_i v_i^T = U\begin{bmatrix}\sigma_1 & & 0\\ & \ddots & \\ 0 & & \sigma_k\\ & 0 & \end{bmatrix}V^T$

then $\|A - A_k\|_F \le \|A - \hat{A}\|_F$ for $\hat{A}$ such that $\mathrm{rank}(\hat{A}) \le F$

PCA

projection

proj $\vec{a}$ on $\vec{u} = \frac{x^Tu}{\|u\|}$

top k eigenvalues

min. reconstruction err

$\tilde{X}_k = \sum_{i=1}^k \sigma_i u_i v_i^T$

complete the square

$w^TAw + w^Tb + c = (w-h)^TA(w-h) + k$

$h = -\frac{1}{2}A^{-1}b$

$k = c - \frac{1}{4}b^TA^{-1}b$

True about Multivariate Gaussian.

- ISOcontours are ellipses
- PDF through mean is Gaussian
- PDF parallel to an axis is Gaussian.

The left singular vector of A is the eigenvectors of AA^T

Finding SVD

compute AA^T, find V by finding orthonormal of eigenvectors of AA^T

$\det(AA^T - \lambda I) = 0$ $V = [v_1\ v_2\ \dots\ v_n]$

find U as $u_i = \frac{1}{\sigma} Av_i$

As λ↑, regularization strength ↑, model complexity ↓.

A matrix times a standard Gaussian vector is jointly Gaussian.

CCA regression

predict Y from X.

$Y_c \approx X_cA$ $X_c = XU$, $Y_c = YV$.

$\hat{Y}_c = \hat{X}_cA = \hat{X}U(U^TX^TXU)^{-1}U^TX^TYV$

$\hat{Y} = \hat{Y}_c(V^TV)^{-1}V^T$

$= \hat{X}U(U^TX^TXU)^{-1}(U^TX^TYV)(V^TV)^{-1}V^T$

$A_{eq} = U(U^TX^TXU)^{-1}(U^TX^TYV)(V^TV)^{-1}V^T$

proj | whitening | decorrelate | proj back

adding $\lambda\|\mu_i\|_2^2$ to k-mean is same as adding λ points at origin at each cluster.

$A = U\Sigma V^T$

$\|a-b\|_2^2 = \langle a-b, a-b\rangle$

whitening change principle component

CCA

Person correlation coefficient $\rho(X,Y)$

$\rho(X,Y) = \frac{\mathrm{cov}(X,Y)}{\sqrt{\mathrm{var}(X)\mathrm{var}(Y)}}$ $\rho(aX+b, cY+d) = \rho(X,Y)$

$\mathrm{Cov}(X,Y) \approx \frac{1}{n}\sum_{i=1}^n (x_i-\bar{x})(y_i-\bar{y})$

Goal: $\max_{u,v} \rho(X_{rv}^Tu, Y_{rv}^Tv)$

$|\rho|$ ↑ when X, y linearly correlated.

$\rho(X,Y) = \frac{\sigma_{xy}}{\sigma_x\sigma_y}$ $\Sigma = \begin{bmatrix}\sigma_x^2 & \sigma_{xy}\\ \sigma_{xy} & \sigma_y^2\end{bmatrix} = \begin{bmatrix}\sigma_x^2 & \rho\sigma_{xy}\\ \rho\sigma_{xy} & \sigma_y^2\end{bmatrix}$

whiten matrix eigenvector decomp

$W_x = U_xS_x^{-1/2}U_x^T$ where $X^TX = U_xS_xU_x^T$

Decorrelate matrix

first SVD: $X_w^TY_w = USV^T$ ← whitening

where $X_w = XW_x$, $Y_w = YW_y$, $u_w = W_x^{-1}u$, $v_w = W_y^{-1}v$.

To decorrelate $D_x = U$ and $D_y = V$

so $(X_wU)^T(Y_wV) = S$.

then $X_d = X_wD_y$, $Y_d = Y_wW_y$,

$u_d = D_x^{-1}u_w = D_x^Tu_w$

$v_d = D_y^Tu_w = D_y^Tv_w$.

$\max u_d^TSv_d$

$\|u_d\| = 1$

$\|v_d\| = 1$

then we have

$u = W_xu_w = W_xD_xu_d$

$v = W_yv_w = W_yD_yv_d$

$U_d = \begin{bmatrix}I_k & \\ & 0_{c-k,k}\end{bmatrix}$ $V_d = \begin{bmatrix}I_k & \\ & 0_{c-k,k}\end{bmatrix}$

$U = W_xD_xU_d$, $V = W_yD_yV_d$

Scanned with CamScanner

Tricks

$\|x\|^2 = x^T x$

Correlation matrix is PSD

$\nabla_x f = \left(\frac{\partial f}{\partial x}\right)^T$

$\frac{\partial (w^T x)}{\partial x} = w$

$\nabla f(x+\Delta) = \nabla f(x) + \nabla^2 f(x)\Delta$

$\frac{\partial (w^T A x)}{\partial x} = w^T A$

$\frac{\partial (w^T A x)}{\partial w} = x^T A^T$

$\frac{\partial (w^T A x)}{\partial A} = x w^T$

if A is a scalar then $A = A^T = \text{trace}(A)$

$\frac{\partial (x^T A x)}{\partial x} = x^T (A^T + A)$

A is symmetric $A = A^T$

A is orthonormal: $AA^T = I$, $A^T = A^{-1}$, $\|Ax\| = \|x\|$

singular value = Eigenvalues

$X = U \Sigma V^T = \sum_{i=1}^{d} \delta_i u_i v_i^T$ (n×d, d×d, d×d)

u_i are left singular vector

v_i are right singular vector

Cauchy-Schwarz: $|a^T b| \le \|a\|_2 \|b\|_2$

$\det(A) = \prod$ eigenvalues

$\text{Cov}(X) = \text{Var}(X)$

$\text{Cov}(X,Y) = E[(X - E[X])(Y - E[Y])]$

$\text{Var}(X) = E[(X - E[X])^2]$

$= E[X^2] - E[X]^2$

$\det\left(\begin{bmatrix} a & b \\ c & d \end{bmatrix}\right) = ad - bc$

$\text{trace}(A) = \sum$ eigenvalues $= \sum_i (A_{ii})$

PCA = degree of freedom.

convex: 2nd derivative > 0

$P(Y|M) = \prod_{i,j} P(Y_{ij} | M_{ij})$

$\min_M \|Y - M\|_F^2$

E-Y theorem: $Y = \sum_{i=1}^{d} \sigma_i u_i v_i^T$

then $\hat{M} = \sum_{i=1}^{k} \sigma_i u_i v_i^T$

$\text{Var}(aX) = a^2 \text{Var}(X)$

$\text{trace}((A^T A)^{-1}) = \sum_{i=1}^{d} \frac{1}{\sigma_i^2}$

$\left(\sum_{j=1}^{d} \sigma_j u_j v_j^T\right)(\sigma_i v_i) = \sigma_i^2 u_i$

trace = sum of diag ↑ if have u_i then $= \sigma_i^2 u_i u_i^T$

PSD:

$\forall v, v^T A v \ge 0$

eigenvalues all ≥ 0

$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$

corr = 0 ⇏ independence

independence ⇒ corr = 0

corr = 0 + JG distribution ⇒ independence

Dual of SVM is quadric, dual depends on Gram matrix, thus can kernelize

adw

Tue 1-2pm

Alvin.

$\sum_{i=1}^{n} x_i x_i^T = X^T X$

Covariance of $X = X^T X$

SVD: $X = U D V^T$

kernel matrix $K = XX^T$

$w \leftarrow w + X^T M X X^T u$

then $a \leftarrow a + M X X^T u$

where a is coefficient of linear comb

LASSO

$\min_w \|Xw - y\|^2 + \lambda \|w\|_1$

$\|w\|_1 = \sum_{i=1}^{n} |w_i|$

assume Laplace prior

Coordinate Descent

while w NOT converge:

pick feature index i

update w_i to $\arg\min_{w_i} L(w)$

find global min if L is jointly convex

$\frac{\partial^2}{\partial x \partial y} f(x,y) \ge 0$.

$$w_i^* = \begin{cases} 0 & \text{if } |a| \le \lambda \\ \frac{-\lambda + a}{b} & \text{if } \frac{-\lambda + a}{b} > 0 \\ \frac{\lambda + a}{b} & \text{if } \frac{\lambda + a}{b} < 0 \end{cases}$$

where $a = -\sum_{j=1}^{n} 2 x_{j,i} c_j(i)$.

$b = \sum_{j=1}^{n} 2 x_{j,i}^2$

Matching Pursuit

initize $r = y$.

initialize $w = 0$

while $\|w\|_0 < k$ do

find index i which r is minized:

$i = \arg\max_j \langle r, x_j \rangle / \|x_j\|$

update i'th entry of weight vector

$w_i = \langle r, x_i \rangle / \|x_i\|^2$

update new residual value

$r = y - Xw$.

OMP

index set

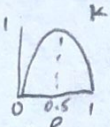
1. initialize $r^0 = y, I^0 = \phi, \hat{w}^0 = 0$
2. repeat $t = 1, \ldots$ until $\|r^t\|_2 <$ threshold or $t > k$ (sparsity)
   - index update
     $r^t = y - X\hat{w}^{t-1}$
     $j^t = \arg\min_i (\min_v \|r^t - v X_i\|_2^2)$
     $I^t = I^{t-1} \cup \{j^t\}$
   - estimate update
     $\hat{w}^t = \arg\min_{w: R^t} \|y - X_t w\|_2^2$
     where $X_t = [X_{j^1}, \ldots X_{j^t}]$

Decision Tree

Surprise: $-\log P(Y=k)$

Entropy: $H(Y) = E[-\log P(Y)]$

$= -\sum_k P(Y=k) \log P(Y=k)$



want to minimize

$\min H(Y|X_{j,v}) :=$

$P(X_{j,v}=1) H(Y|X_{j,v}=1) +$

$P(X_{j,v}=0) H(Y|X_{j,v}=0)$

mutual information

$I(X_{j,v}; Y) := H(Y) - H(Y|X_{j,v}) > 0$

Gini impurity

$G(Y) = \sum_k P(Y=k) \sum_{j \ne k} P(Y=j)$

$= 1 - \sum_k P(Y=k)^2$

Want to

$\min G(Y|X_{j,v}) = P(X_{j,v}=1) \cdot G(Y|X_{j,v}=1) + P(X_{j,v}=0) G(Y|X_{j,v}=0)$

Stopping

- Depth, node purity, info gain criteria,

Prune

Random forest (ensemble learning)

to decrease variance

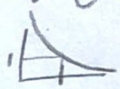
- Per-classifier bagging (bootstrap), choice m<n data points.
- Per-feature randomization, choose k<d feature>

Adaboost

1. initialize $w_i = \frac{1}{n}$ for all points
2. repeat for $m = 1, \ldots M$:
   (a) build classifier G_m according to w_i
   (b) compute $e_m = \frac{\sum_{i, \text{misclassified}} w_i}{\sum_i w_i}$
   (c) reweight $w_i \leftarrow w_i \begin{cases} \sqrt{\frac{1-e_m}{e_m}} & \text{if misclassified} \\ \sqrt{\frac{e_m}{1-e_m}} & \text{otherwise} \end{cases}$
   (d) normalized w_i to sum to 1

Adaboost use exponential loss

$L(y, h(x)) = e^{-yh(x)}$



Gradient Boosting

General version of adaboost

$\min_{\alpha, G} \sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + \alpha G(x_i))$

let $g_i = G(x_i)$, then

cost function

$\approx \sum_{i=1}^{n} L(y_i, F_{m-1}(x_i)) + \alpha \sum_{i=1}^{n} \frac{\partial L}{\partial \hat{y}}(y_i, F_{m-1}(x_i)) g_i$

in case of square loss:

$-\nabla_y L(y, F_{m-1}(x)) = y - F_{m-1}(x)$

$\frac{\partial L}{\partial \hat{y}}(y_i, F_{m-1}(x_i)) = -(y_i - F_{m-1}(x_i))$

as in MP

k-mean, SVM, ridge can be kernelized

$e_m \to 0$, $\frac{1-e_m}{e_m} \to \infty$, so $\alpha_m \to \infty$.

$e_m \to 1$, $\frac{1-e_m}{e_m} \to 0$, $\alpha_m \to -\infty$

$\alpha_m = \frac{1}{2} \ln\left(\frac{1-e_m}{e_m}\right)$

$h(x) = \text{sign}\left(\sum_{m=1}^{M} \alpha_m G_m(x)\right)$

(misclassified ↑ w_i)

Scanned with CamScanner

SQL

SELECT [Distinct] <columns>
FROM <single table>
[WHERE <predicate>]
[GROUP BY <columns>
  [HAVING <predicate>]] ← ASC, DESC.
[ORDER BY <column list>]; [LIMIT] <number>

SQL Joins (multi-table queries)

SELECT [Distinct] <columns>
FROM <table>
  [INNER | {LEFT | RIGHT | FULL} {OUTER}] JOIN <table> ← preserve left
  ON <predicate>
[WHERE <predicate>]

SQL keywords

- UNION ∪ EXCEPT
- INTERSECT ∩
- IN ← nested query, can have > ANY
- EXIST
- COUNT, ALL, ANY, NOT, NULL

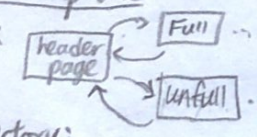
LIKE 'B%' matches strings start w/ B

Create View — or With Name(..,..) AS (query)
AS (query)

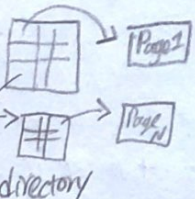
Files and Buffer Management

Files of pages, pages of record

Unordered heap file

as list: 

Page directory:



directory

Directory entries include # free bytes on referenced page

Heap

- full scan good

Sorted

- retrieval in order, range

Clustered files and indexes

- fast lookup and modification

Page Layout

Fixed length record, packed
- record number in page.
- delete need re-arrange.

Fixed length record unpacked
- page header has 'bitmap' denotes "slots" w/ record.

Variable length record, slotted page
- slot directory, footer.
- each slot has pointer to beginning of record and length of record.
- slot grow from end
- need reorgainzation to prevent fragmentation.

Buffer Management

- FramId, PageId, Dirty?, Pin Count

Requestor of page must:
1) indicate whether page written (dirty)
2) unpin page

LRU = Least recently used.
- Pages at the end of reading = last B pages
- Costly (need to find last used) and bad for sequential scan

MRU = Most recently used.
- Good for sequential scan
  - $\approx (B-1)/N$ hit rate

Clock Policy

- Replace, set pin, set ref bit, advance clock

Sorting

Double buffering

- Main thread runs f(x)
- 2nd thread drains/fills unused I/O bufs in parallel
- Same # of I/O but faster.

External merge sort

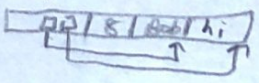
- Pass 0 (conquer a batch):
  - quicksort each page and make $\lceil N/B \rceil$ sorted runs
  - Pass 1, ..., Sort B-1 runs one at a time
  - Num pass = $1 + \lceil \log_{B-1} \lceil N/B \rceil \rceil$
  - Cost = $2N *$ (# of passes)
  - How big a table sorted in k passes > $B(B-1)^{k-1}$

Hashing

- arranged on disk so that no 2 records that are incomparable are separated by a greater or smaller record.

Record Layout

Fixed length: use arithmetic to find field.
Variable length: delimiters (CSV), use ;, bad
Record header, has pointer to end of variable length fields



Scanned with CamScanner

Cost of Operations

	Heap File	Sorted File	Clustered 2/3 fill, Index. $f = 2d+1$
Scan all records	$B*D$	$B*D$	$1.5B*D$
Equality search	$0.5B*D$	$(\log_2 B)*D$	$(\log_F 1.5B+1)*D$
Range search	$B*D$	$((\log_2 B+pages))D$	$((\log_F 1.5B)+pages)D$
Insert	$2D$	$((\log_2 B)+B)*D$	$((\log_F 1.5B)+2)*D$
Delete	$(0.5*B+1)*D$	$((\log_2 B)+B)D$	$((\log_F 1.5B)+2)*D$

B: Number of data blocks
R: Number of records per block
D: Average time to read/write blo

Next block concept
- Sequential blocks on same track
- blocks on same cylinder
- block on adjacent cylinder
→ minimize seek and rotational delay
pre-fetch
- seeker time, ~2-3 ms on average
- rotational delay, ~0-4 ms
- transfer time
~0.25 ms per 64 kB page

Big O

Scan all records	$O(B)$	$O(B)$	$O(B)$
Equality search	$O(B)$	$O(\log_2 B)$	$O(\log_F B)$
Range search	$O(B)$	$O(\log_2 B+pages)$	$O(\log_F B+pages)$
Insert	$O(1)$	$O(B)$	$O(\log_F B)$
Delete	$O(B)$	$O(B)$	$O(\log_F B)$

Indexes

ISAM, insertion and overflow cause linked list

B+ Tree

$d \leq$ # entries $\leq 2d$
$f = 2d+1$
fill-factor $\geq 2/3$

Insert
Overflow: leaf node split, copy mid to inner
inner node move mid to upper inner.
k = height
max elements hold = $2d(2d+1)^{k-1}$

Bulk Loading
- Fill leaf pages to some fill factor
- update parent until full, split parent to fill factor

Composite key
- Lexicographic order, all = except last one can be <, >.
- Alternative 1 Index (B+ Tree)
  - Record contents are stored in index file. (super by design cluster)
- Alternative 2: By reference, <k, rid of match data record>
- Alternative 3: By list of reference, <k, list of rids of matching data records>

Cluster
- Sort heap file
  → Efficient range searches, locality, compression
  → Expansive to maintain, only 2/3 full.

Prefix key compression
- On split, determine minimum splitting prefix to copy up.
- compress so keys are diff w/ shortest letters

Suffix key compression
Move common prefix to header.

T: Single-pass streaming w/ separate input and output disks is nearly all sequential I/O. Single pass required fix amount of RAM. External merge sort make use of one-pass streaming in merges

Scanned with CamScanner

Relational algebra (return set)

π : projection, retain columns ([])
σ : selection, retain row (▭)
ρ : renaming
∧ = and
∪ : union, tuples in r1 or in r2
− : set diff., tuples in r1 but not in r2
× : Cross product, size: |R1|×|R2|
∩ : intersection, tuples in r1 and r2
⋈, $⋈_θ$: joins
ρ(relation_name(1→sid1, 4→sid4), R1×S1)
↑ new table name. ↑ input
∩ ≡ s1 − (s1 − s2).
Theta join ($⋈_θ$): $R ⋈_θ S = σ_θ(R×S)$
Natural join (⋈):
R⋈S : $π_{\text{unique field}} σ_{\text{eq. matching field}} (R×S)$
Division (/):
$A/B = π_x(A) − π_x((π_x(A)×B) − A)$
Group by / Aggregation
$γ_{\text{age, AVG(rating)}, \text{COUNT(*)}>2}$ (Sailors)
Optional HAVING

Join Costs

notations: [R] = # of pages store R,
P_R = # of records per page for R
|R| = cardinality (# of record) = $[R]P_R$

Simple Nested Loop

```
for record R:
    for record S:
        Join r_i, s_i
```
Cost: $([R][R])*[S]+[R]$

Page Oriented Nested Loop Join

```
for each page in R
    for each page in S
        for each record r in page_R
            for each record s in page_S
                join r_i, s_i
```
Cost: [R]*[S] + [R]

Block/batch Nested Loop Join

```
for each block b_R of B-2 pages in R
    for each page in S:
        for each record r in b_R:
            join r_i, s_i
```
Cost: $\lceil \frac{[R]}{B-2} \rceil * [S] + [R]$

Nested Loop Join

cost F = [Access cost C] + [cardinality of C]* [access cost F]

Sort-Merge Join

```
while not done {                      ← merging scan phase
    while (r<s) {advance r}
    while (s<r) {advance s}
    mark s //save start of "block"
    while (r==s) {
        while (r==s) {
            yield <r,s>
            advance s.
        }
        reset s to mark
        advance r
    }
}
```
2 passes of ([R]+[S])
Cost: Sort R + Sort S + ([R]+[S])

Grace Hash Join

Cost: $[R] < \sim B^2$
$R ⋈_θ S = σ_θ(R×S)$
Partition tuples from R and S by join key
Build & Probe a separate hash table for each partition

PseudoCode

```
For Cur in {R,S}
    For page in Cur
        read page into input buffer
        for tup on page
            place tup in output buf hash_p(tup.joinkey)
            if output buf full then flush to disk partition
Flush output bufs to disk partitions
For i in [0...(B-1)]
    For page in R_i
        For tup on page
            build tup in memory hash_r(tup.joinkey)
    For page in S_i
        For tup on page
            Read page into input buf
            Probe memory hash_r(tuple.joinkey) to match
            Send all matches to output buf
            Flush output buf if full.
```
B-1 partitions, need to fit B-2 buffer page for hash table

Hybrid hash join

k = (smallest page size − (B−2))/(B−3)
# page for hash join = B − 2 − k
Cost = hash join − $(\frac{k}{\text{Small}})*2$

n machine, B buffer each, N pages sort and hash
$1 + (1 + \log_{B-1}(N/mB))$
Lossless test
ABDE ∩ BCDF = BD (BD not contain BD→BDE or BD→BCF) either decompose, so not lossless

speed up / scaleup −

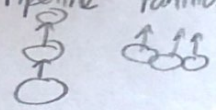
Index Nested Loop Join (index on s)

```
for each tuple r in R:
    for each tuple s in S where r_i == s_i do
        add <r,s> to result.
```
Cost = [R] + ([R] * P_R) * cost to find S tuple matching r.

Alt 1: cost traverse tree from root to leaf (2-4 IOs)
Alt 2 or 3: (plus)
1. cost to look up RID: 2-4 IO's + (alt 2 adds one more)
2. Cost to retrieve record: Clustered: 1 I/O per page of matching S tuple
Unclustered: 1 I/O per matching S tuple

Parallelism

Pipeline · Partition



shared memory: mem + disk
shared disk: disk, share nothing (cluster)

Kinds of Query Parallism

Intra-query
1. inter-operator (pipeline) vs (bushy tree parallism).
2. intra-operator (partition).
inter-query
- each query runs on separate processor
- requires concurrency control

Intra-operator parallism

Data partition
Range: good for equijoins, range queries, group-by
Hash: good for equijoins, group-by.
Round-robin: spreading load.
cost: (2*# of partition phase + 1)([R]+[S])

A is superkey if A is not on RHS of every FD

Parallel Hash Join Grace

- Pass 1 is like hashing.
- Pass 2 is local GHS per node (hash both R and S)

Parallel Sorting

- Range partition first then sort

Parallel Sort Merge

- Pass 0 → n-1 are like parallel sort
- Pass n merge partitions by node

Scanned with CamScanner

Symmetric shuffle; same last page

Asymmetric shuffle: If R already partitioned, just partition S

Broadcast Join:
if R is small, send it to all nodes and join at each node w/ S partitioned

Relational Query Optimization

Query Parser → Query Rewriter → Query Opt. → Query Executor

selection:

$\sigma_{c1 \wedge \dots \wedge cn}(R) \equiv \sigma_{c1}(\dots(\sigma_{cn}(R))\dots)$ cascade

$\sigma_{c1}(\sigma_{c2}(R)) \equiv \sigma_{c2}(\sigma_{c1}(R))$ commute

Projection:

$\pi_{a1}(R) \equiv \pi_{a1}(\dots(\pi_{a1\dots,an-1}(R))\dots)$ cascade

Cartesian

$R \times (S \times T) \equiv (R \times S) \times T$ associative

$R \times S \equiv S \times R$

- Selection cascade and pushdown
  - Apply selections as soon you have relevant columns
- Projection cascade and pushdown
- Avoid Cartesian Product

Materialize inner loops

System R (Selinger) optimizer

1. Plan Space
   - Only consider left-deep plans
2. Cost estimation.
   - consider CPU & I/O costs
3. Search Alg: DP

Translating SQL to "Relational algebra"

1. Join (FROM, WHERE, AND) use $\bowtie$
2. Selection (AND), σ
3. Group By (use GROUP BY)
4. HAVING
5. Projection (SELECT), π.

$\sigma_{R.a=S.b}(R \times S) = R \bowtie_{R.a=S.b} S$

Physical properties: sort and hashed

Cost estimate

cost = #I/O + CPU-factor * # tuples

Catalogs keeps stat of relations

Selectivity (sel) = $\frac{|\text{output}|}{|\text{input}|}$

Also called reduction factor.

Result size estimation

Result Cardinality = Max # tuples * product of RF's

- Term col = value: RF = 1/NKeys(I)
- Term col1 = col2: RF = 1/(MAX(NKeys(I1), NKeys(I2)))
- Term col > value: RF = $\frac{\text{High}(I) - \text{value}}{\text{High}(I) - \text{low}(I) + 1}$
- If missing stat, assume 1/10.

Histogram

Assume uniform distribution in bin and (∧), multiply, or (∨), add, - product of disjuncts

sel for $R \bowtie_p \sigma_q(S) = SpSq|R||S|$

Enumeration of plans

- single-table queries include select, projects, groupBy/agg

Cost estimate for single plan

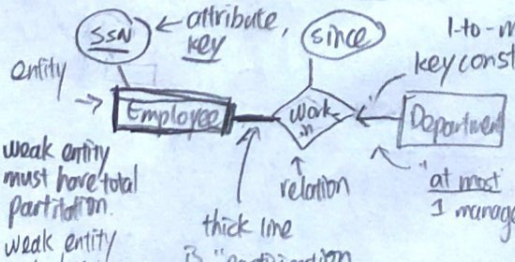
- Index I on primary key match selection
  - Height(I) + 1 for B+ tree
- Clustered index I: I ≡ index
  - (NPages(I) + NPages(R)) * product of RF's of matching selects
- Non-clusted index I:
  - (NPages(I) + NTuples(R)) * product of RF's of matching selects
- Sequential scan: NPages(R)

Physical DB design

- Attributes mentioned in WHERE clause are candidate for index scan (only 1 index can cluster)

Conceptual schema defines logical structure.
Students(sid....)
Physical schema describes files and indexes used
- relations stored as unordered

Entity-Relation model (ER)



key + participation = one to one

Weak entity, thick border line, identified uniquely by considering p. key of another entity.

binary relation better

can have more than 1 column as primary key

steps
skip superkeys and keys not contained by relation

Decomposition

Lossless, not new rows after join
if $\rho_x(r) \bowtie \rho_y(r) = r$

Dependency Preserving.
if $(F_x \cup F_y)^+ = F^+$

Function Dependencies

$X \to Y$ means X determines Y
X and Y are sets of attributes
Superkey: set of col. in table determine all cols.
Candidate key: minimal set of col. that determine all cols
Have $R \to W$, decompose to two table one with everything - W and RW

F^+ = closure of F is set of all FDs that are implied by F

Armstrong Axioms
Reflexivity: if $X \supseteq Y$, then $X \to Y$
Augmentation: if $X \to Y$, then $XZ \to YZ$ for all Z
Transitivity: If $X \to Y$ and $Y \to Z$, then $X \to Z$

Sound and complete
$X \to Y$ and $X \to Z$ $X \to YZ$.
$X \to YZ$ then $X \to Y$, $X \to Z$

Normal Forms

1st normal: all attributes atomic

if R in BCNF, rows can't be inferred → Boyce-Codd Normal Form (BCNF)

R is in BCNF if and the only non-trivial FDs over R are key constraints

ER to relational

Entity sets to tables, simple relationship sets to tables
1) keys for each participating entity set (as p. key and foreign key) and attributes (many-to-many)
2) Have key constraint, only that's p key will be p. key in this relation

Participation Constraint use NOT NULL

Weak Entity: have p. key and foreign key to owner and use ON DELETE CASCADE when reference foreign key
Relational table also need p. key of entities

Scanned with CamScanner

NP-hard problems

Hamiltonian/Rudrata Path/Cycle

- A path/cycle that visits each v once

Vertex cover

Set of vertices such that each edge is incident to at least one v in set.

Independent set

A set of v that no 2 v's are adjacent

3-SAT

v = or ∧ = and

$(x_1 \vee x_2 \vee \bar{x}_3) \wedge (\dots)$

Circuit-SAT

Uses circuit of NAND gate, I/O constants, and "?" to make output 1

NAND

A	B	O
0	0	1
0	1	1
1	0	1
1	1	0

Integer Programming

LP that restricts some/all constraints to be integer

Clique

Given: G and b, find complete subgraph that has b vertices

3D-Matching

has list of (b, g, p).

3-appearance 3SAT

3SAT but x_i appears at most 3 times

ZOE (0-1 equality)

Given $A \in 2^{m \times n}$, solve $Ax = \vec{1}$

Subset Sum

Given set of ints, is there non-empty subset sum = 0

Traveling Salesperson (TSP)

Shortest possible route to visit each v and return to s.

ZOE to Subset Sum

Represent x in a higher radix so no carry over.

ZOE → Rudrata Cycle

ZOE → Rudrata Cycle w/ Paired Edges.

For each x we make gadget of two vertices joined by 3 e's, corresponding to 0 and 1.

Reduction

A → B, B can use to solve A, B is harder.

Prove A has sol iff B has sol

Rudrata path → Rudrata cycle

add vertex x and edges $\{(s, x), (x, t)\}$

SAT → 3SAT

$f: (a_1 \vee a_2 \vee \dots a_k)$

$\to (a_1 \vee a_2 \vee x_1)(\bar{x}_1 \vee a_3 \vee x_2) \dots (\bar{x}_{k-3} \vee a_{k-2} \vee a_{k-1} \vee a_k)$

3SAT → 3-appearance 3SAT

make variables $x_1, \dots x_{30}$ to replace each x and add clauses $(x_1 \vee \bar{x}_2)(x_2 \vee \bar{x}_3) \dots (x_{30} \vee \bar{x}_1)$

3SAT → Independent Set

Graph contains all $x, \bar{x}$ as v. For each clause, form a {1,2,3}-gon connected ORed literals. Add edges b/w all x and $\bar{x}$.

Independent Set → Vertex Cover

$f = 1$, $h: s' \mapsto (V - s')$

The complement of IS is VC

Independent set → Clique

$f: (V, E) \mapsto (V, V \times V - E)$, $h = 1$

3-appearance 3SAT to 3D matching

Make a graph (gadget), b, g, p forms a graph. Each gadget is a rectangle that forces you to choose b/w two possible ways to assign 2 triples.

To represent $(x \vee \bar{y} \vee z)$ we need 3 gadgets, 6 b, 6 g, and 12 p's. If have leftover p, introduce b & g that takes any p.

{NP} → Circuit SAT

Compiling to circuit is polynomial

3D matching → ZOE

pick ZOE variables $x_1, \dots x_m$ representing feasible triple. Matrix A - constraints that each b, g, p used once.

Longest Path

Find longest directed path in G.

Each row of A will correspond to 2-v graph. Join all 2 vertex gadget end to end.

RCPE → RC

start w/ |C| then repeat for larger C.

P: problems solved in poly time.

NP: problems verified in poly time

NP-Complete: intersection of NP and NP-hard.

NP-hard: problems that are at least as hard as hardest in NP

Approximation Algorithm

approx algor A(I), then ratio is

maximization prob: $\min_I \frac{A(I)}{OPT(I)} \le 1$

minimization prob: $\max_I \frac{A(I)}{OPT(I)} \ge 1$

For minimization problem:

1) Solve another problem that gives lower-bound on optimal problem
2) Find a way to turn this non-feasible solution into a feasible solution

infeasible < OPT ≤ k* infeasible ≤ k* OPT

= lower-bound < OPT ≤ Approx ≤ k* lower bound

MST is 2-approx for TSP.

c-TSP is NP-complete

Knapsack is easy approx

Primality Testing

Fermit's little theorem

if p is prime number and a is a number s.t. (p, a) = 1, then $a^{p-1} = 1 \bmod p$. or $a^p = a \bmod p$

Proof: $\prod_{\text{distinct } x \in \mathbb{Z}/p\mathbb{Z}} (a \cdot x) = \prod x$

Correctness

Test never miss a prime, but thinks a composite is prime. (no false negative)

1) m is prime
2a) m is composite and there is a fermat witness.

Probability that we say it's prime after running k samples → $(\frac{1}{2})^k$.

2b) m is composite and no fermat witness

Carmichael number.

Do Miller-Rabin Primality Test.

- if there exists a s.t. $a^2 = 1 \bmod n$ and $a \neq \pm 1 \bmod n$, then n is not a prime.

One-Way Function

$x \to f(x) \to y$ easy

For every PPT machine A

$Pr[A(f(x)) = x' \text{ and } f(x') = f(x)] \le 1/\text{poly}(n)$

OWF are bijective.

$Pr[A(f(x)) = x] \le 1/\text{poly}(n)$.

Negligible function $g \to g(n) < 1/\text{poly}(n)$

Sum, prod of NF is NF

Scanned with CamScanner

Rabin's Function

For a size n, choose N with n bits. Mapping is $x^2 \bmod N$.

Factoring → Rabin's

give $y = z^2 \bmod N$ to Rabin's solver, output $\pm z$ or $\pm z'$ w.p. $\pm$

N factored as $(z+z')(z-z')$

Encryption function verifies Decryption functions

NP-hard is not subset of NP-search problem.

Tips: Derive algorithm to see if assignment possible: reduce to max flow.

add $\{s, t\}$,

let $s \to v$ be the limited constraints.

can use ∞ weights

and let $v \to t$ be what we need in order to satisfy

tips: split each v into two v_i

Rudrata Cycle → Directed Rudrata Cycle

turn (u,v) to (u,v) and (v,u)

3SAT → 4SAT

create a new variable y and make

$(x_1 \vee x_2 \vee \bar{x}_4)$ to $(x_1 \vee x_2 \vee \bar{x}_4 \vee y) \wedge (x_1 \vee x_2 \vee \bar{x}_4 \vee \bar{y})$

Prove NP-completeness

1) in NP
2) reduced from NP-hard problem

If a and N are co-prime, a has multiplicative inverse mod N

The node w/ highest post order in DFS of undirected connected graph is in source component

Find x for $8x = 7 \bmod 99$

$8^{-1} = 62 \bmod 99$ so $x = 7 \times 62 \bmod 99$

What value of ω to use FFT to mult 2 degree 5 poly?

result = degree 10, has 11 coeff, next higher of 2 is 16.

thus $\omega = e^{2\pi i/16}$

If solution/structure of problem looks like binary tree, try Huffman encoding.

Shortest path w/ K edges

$L(s,k)$ = length of shortest path ~~from~~ s to v using $\le k$ edges

$L(v,k) = \min \{L(u,k-1) + w_{uv} \mid u \text{ is a neighbor of } v\}$

$x + y = 1$

$\max(x,y) \ge 1/2$

To check if p is a factor of n, we can check if $n/p * p == n$.

# of SCC in n vertex DAG is n.

directed graph can be decomposed into DAG of SCC

Cross edge only if directed graph

if all variables same prop. then Huffman is balanced binary tree.

Minimum vertex Cover is poly-time solvable on tree

↖ not an NP problem.

Maximum Flow is a poly solvable.

Every problem in NP can be written as an integer linear program.

MST is an NP problem

Scanned with CamScanner

All-pairs shortest paths

Floyd-Warshall algorithms

$d[i,j,k]$ = the shortest distance between i and j using vertices $\{1,\dots k\}$

$$d[i,j,k] = \min \begin{cases} d[i,j,k-1], & k \text{ is not on shortest path.} \\ d[i,k,k-1]+d[k,j,k-1], & k \text{ is on shortest path.} \end{cases}$$

base

$$d[i,j,0] = \begin{cases} \ell(i,j) & (i,j) \in E \\ \infty, & \text{otherwise} \end{cases}$$

```
for k=1 to n:
    for i=1 to n:
        for j=1 to n:
            d[i,j,k] = min{d[i,j,k-1], d[i,k,k-1]+d[k,j,k-1]}
```

Independent set in a tree

$I(v)$ = |largest indep. set of subtree rooted at v|

$$I(v) = \max \begin{cases} 1 + \sum_{u \xrightarrow{\text{grandchild}} v} I(u), & v \text{ is included.} \\ \sum_{u \xrightarrow{\text{child}} v} I(u), & v \text{ is not included.} \end{cases}$$

Linear Programming

Constraints and opt. func

Simplex algorithm (run in exponential time)

start at corner keep going to better neighbor.

m constraints in n variables, there are $\binom{m}{n}$ corners

Parameterizing LPs

$$\sum a_i x_i \le b_i \sim \begin{pmatrix} \sum a_i x_i + s = b_i \\ s \ge 0 \end{pmatrix}$$

Convert to min problem with equality in non-negative variables.

Flows in Graph (max flow) Ford Fulkerson

```
while ∃ path s⇝t do
    P ← s⇝t path in G.
    f = min_{e∈P} c_e
    for (u,v) ∈ P do:
        c_{u,v} ← c_{u,v} - f
        c_{v,u} ← c_{v,u} + f
```

min cut obtained by reachable nodes in max flow resider graph.

$O(FE)$

res. graph includes back edges

Bipartite matching

b_i likes g_i, pairing possible.

- connect s to all b and t to all g, use max flow algorithm.

If all edges have integral capacities, then flow on each edge is integral

Reductions

Problem P reduces to Q if P can be solved using Q as a subroutine. Pre and post processing.

Zero sum Games

	C_1	C_2	y_1, y_2
x_1	r_1	3	-1
x_2	r_2	-2	1

Column player's payoff

$$\min_{y_1,y_2} \max(3y_1 - y_2, -2y_1 + y_2)$$

Row player's payoff

$$\max_{x_1,x_2} \min(3x_1 - 2x_2, -x_1 + x_2)$$

write as a linear program. ← P.

Hard Problems

SAT: search problem, validate TSP can be search problem. Given a problem I, find solution S or say no.

Duality

mult each constraint by y_i and make new constraints base on coefficient of x_i's and nonnegative

if change primal constraint to equality, we will change dual LP so there is no nonnegativity for y_i

if we have k neg edges, need to do Dijstra's 2^k times

To find min-st cut that has x on one side and y on another, add an ∞ weight edge (x,y)

Scanned with CamScanner

Greedy

Huffman-Encoding

- Prefix-free encoding
- In optimal tree, least frequent letter at greatest depth.
- WLOG, the two least frequent letters are siblings
- Combine 2 lightest trees until only 1 tree

Horn SAT satisfiability

Greedy

- make variables false unless necessary

1. all f except unary + constraints
2. greedily satisfy positive clauses

Set Cover, DP

Greedy

Recusively include set that covers biggest # of not-yet-covered elements.

1. while $UF \neq U$ do
2. Cover as many more in one set $\rightarrow F$

If optimal picks k sets, greedy picks $O(k \log n)$ sets where n = # of elements

DP

Longest path is DAG

$L[i] = 1 + \max_{\exists \text{edge } i_b \to i} L[i_b]$

Longest increasing sequence

Construct DAG, form connections $a[i] \to a[j]$ if $i<j$ and $a[i] < a[j]$

Edit distance

$E[i,j]$ = E.d. between $X[1 \ldots i]$ and $Y[1 \ldots j]$

$$E[i,j] = \min \begin{cases} 1 + E[i,j-1] & \text{we did an insertion.} \\ 1 + E[i-1,j] & \\ E[i-1,j-1] + \text{diff}(x[i], y[j]) & \text{keep or sub} \end{cases}$$

base: $E[i,0] \leftarrow i$

$E[0,j] \leftarrow j$.

Gambling Strategy

game 1: \$1 $\rightarrow$ $\begin{cases} 2 & \text{w.p } 1/2 \\ 0 & \text{w.p } 1/2 \end{cases}$

2: \$3 $\rightarrow$ $\begin{cases} 6 & \text{w.p } 2/3 \\ 1 & \text{w.p } 1/3 \end{cases}$

Prime number as goal, start \$100

$P[i,m]$ = Pr {optimal strategy reaches goal starting with \$m at end of game i

$$P[i,m] = \max \begin{cases} \frac{1}{2} P[i+1, m+1] + \frac{1}{2} P[i+1, m-1], \\ \frac{2}{3} P[i+1, m+3] + \frac{1}{3} P[i+1, m-3]. \end{cases}$$

base cases

$$P[n,m] = \begin{cases} 1, & m \text{ prime.} \\ 0, & \text{otherwise.} \end{cases}$$

Knapsack (w/ repetition)

$K[w]$ = value of best bundle w/ weights $\leq w$.

$K[w] = \max_{i : w_i \leq w} (v_i + K[w - w_i])$

$K[0] = 0$

Knapsack (w/o repetition)

add item in particular order.

$K[w,i]$ = max value of bundle w/ weights $\leq w$ and items $\leq i$

$$K[w,i] = \max_i \begin{cases} K[w - w_i, i-1] + v_i. & \text{(picked)} \\ K[w, i-1], & \text{(discarded)} \end{cases}$$

$K[0, \cdot] = 0$

runtime is $\Theta(nW)$, input #, expo. in input size.

Chocolate bar ($n \times n$ grid of 1 and 0's)

$T[(x_1,y_1),(x_2,y_2)]$ = min # breaks req. to separate just this rectangle.

$$T[(x_1,y_1),(x_2,y_2)] = \min_{\substack{x_1 < x < x_2 \\ y_1 < y < y_2}} \begin{cases} 1 + T[(x_1,y_1),(x,y_2)] + T[(x,y_1),(x_2,y_2)] \\ 1 + T[(x_1,y_1),(x_2,y)] + T[(x_1,y),(x_2,y_2)] \end{cases}$$

base case 0 cuts if homogeneous

Traveling Salesman

Goal: find cheapest visit every node once and come back home.

Goal': find least cost path that visits each node once.

$C[S,j]$ = smallest cost to go 1 to j using vertices in S

$$C[S,j] = \min_{i \in S} (C[S-j, i] + \ell[i,j])$$

base: $|S| = 1, i = 1 \Rightarrow C[S,i] = 0$

$|S| = 1, i \neq 1 \Rightarrow C[S,i] = \infty$, look at all j's

Scanned with CamScanner

Graph

DFS, directed edges $(e(u,v))$ $O(V+E)$ if use adjacency list

Tree/forward edges: $u\ v\ v\ u$ (( ))

$pre(u) < pre(v) < post(v) < post(u)$

Back edge: $v\ u\ u\ v$ (( ))

$pre(v) < pre(u) < post(u) < post(v)$

$\exists$ (back edges) $\Leftrightarrow \exists$ (directed cycles)

Topological sort

$u \rightsquigarrow v$ means $post(u) > post(v)$.

Sort the vertices in decreasing post values. Return reverse postorder.

SCC

u, v is in SCC if $\exists\, u \rightsquigarrow v, \exists\, v \rightsquigarrow u$,

every graph is DAG of SCC

Find a list of SCCs, linearlized shortest path costs $O(|V|+|E|)$

1. DFS on $(V, E_{reversed})$ to id vertex in sink SCC (largest post value)
2. DFS from sink-vertex to id the sink SCC
3. Delete sink SCC, repeat

Shortest Path

Dijstra: $O((|V|+|E|)\log V)$

Operation	binary heap	d-ary heap
insert	$\log n$	$\log_d n$
delete min	$\log n$	$d \log_d n$
decrease key	$\log n$	$\log_d n$.

Bellman-Ford (find shortest paths w/ neg edges)

```
for all u ∈ V                    O(|V||E|)
    dist(u) = ∞
    prev(u) = nil
dist(s) = 0
repeat |V|-1 times:
    for all e ∈ E:
        update(e)
```

Run one more time to detect if there is neg cycles.

	1	2	3
S			
A			
B			
C			
T			

MST shortest spanning tree

- lowest weight tree that reach all v.

Kruskal's algorithm

```
Sort edges by weight.
for all u ∈ V:
    makeset(u)
X = {}
for all edges {u,v} ∈ E
    if find(u) ≠ find(v):
        add edge {u,v} to X
        union(u,v)
```

Union-find data structure, $O(|E|\log V)$

Prim's algorithm (use cut property)

```
for all u ∈ V
    cost(u) = ∞
    prev(u) = nil
Pick any initial node u_0
cost(u_0) = 0
H = makequeue(V) (PQ, using cost-value)
while H is not empty:
    v = deletemin(H)
    for each {v,z} ∈ E:
        if cost(z) > w(v,z):
            cost(z) = w(v,z)
            prev(z) = v
            decreasekey(H,z)
```

$O(\log(V)(|E|+|V|))$

Cut Property

Partially built X that has S list of edges, for a lightest edge e from S to $V-S$, $X \cup \{e\}$ will be part of some MST.

Modular

$x \equiv y \bmod N$ N divides $(x-y)$

$x \equiv x' \bmod N$ and $y \equiv y' \bmod N \Rightarrow x+y \equiv x'+y' \bmod N$ and $xy \equiv x'y' \bmod N$.

if $x \geq y$ then $\gcd(x,y) = \gcd(x \bmod y, y)$

$ax \equiv 1 \bmod N$, a is inverse of x

P is prime if $a^{p-1} \equiv 1 \pmod p$ for every $1 \leq a \leq p$.

RSA: $N = pq$, both primes and pick e relative prime to $(p-1)(q-1)$, d be inverse of e modulo $(p-1)(q-1)$

then $(x^e)^d \equiv x \bmod N$

Scanned with CamScanner

Master Theorem

mult: $\Theta(N^{\log_2 3})$, mult matrix: $\Theta(n^{\log_2 7})$

$T(n) = aT(n/b) + f(n)$

$c^* = \log_b a$

if $f(n) \in O(n^c)$, $c < c^*$, $T(n) \in \Theta(n^{c^*})$

if $f(n) \in \Theta(n^{c^*}(\log n)^k)$, $k \geq 0$, $T(n) \in \Theta(n^{c^*}(\log n)^{k+1})$

if $f(n) \in \Omega(n^c)$, $c > c^*$ and $af(\frac{n}{b}) \leq kf(n)$, $T(n) \in \Theta(f(n))$

$T(n) = T(\sqrt{n}) + O(n)$ terminates in $\log \log n$ steps.

$1 + 2 + \dots n \Rightarrow O(n^2)$

$1 + 2 + \dots n^2 \Rightarrow O(n^3)$

Complete graph has $\frac{V(V-1)}{2}$ edges

Divide and Conquer

Matrix multiplication: decrease a

FFT

$w = e^{\frac{2\pi i}{n}}$, $w^j = e^{\frac{2\pi i}{n} j}$

$$M_n = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & w^1 & & w^{n-1} \\ \vdots & & & \vdots \\ 1 & w^{2n} & \cdots & w^{(n-1)(n-1)} \end{bmatrix} \quad M_{ij} = w^{ij}$$

$M_n(w)^{-1} = \frac{1}{n} M_n(w^{-1})$

$x^3 + 2x + 3x + 5 \Rightarrow \begin{bmatrix} 5 \\ 3 \\ 2 \\ 1 \end{bmatrix}$

$f(x) = f_e(x^2) + x f_o(x^2)$

- Squaring n^{th} root of unity results in $\frac{n}{2}^{th}$ roots of unity

$p(w) = 0 + 1w + \dots + 7w^7$

$p(-w) = (0 + 2w^2 + 4w^4 + 6w^6) - (1w + 3w^3 + 5w^5 + 7w^7)$

→ even terms minus odd terms

$P(x) = A_e(x^2) + xA_o(x^2)$

$$\begin{pmatrix} P(w^0) \\ \vdots \\ P(w^3) \\ P(w^4) \\ \vdots \\ P(w^8) \end{pmatrix} = \begin{pmatrix} P_e(w^0) + P_o(w^0) \\ \vdots \\ P_e(w^3) + P_o(w^3) \\ P_e(w^0) - P_o(w^0) \\ \vdots \\ P_e(w^3) - P_o(w^3) \end{pmatrix} = \begin{pmatrix} A_e((w^0)^2) + w^0 A_o((w^0)^2) \\ \vdots \\ A_e((w^3)^2) + w^3 A_o((w^3)^2) \\ A_e((w^0)^2) - w^0 A_o((w^0)^2) \\ \vdots \\ A_e((w^3)^2) - w^3 A_o((w^3)^2) \end{pmatrix}$$

$\Theta(n \log n)$

$\langle values \rangle = FFT(\langle coef. \rangle, w)$, $\langle coef \rangle = \frac{1}{n} FFT(\langle values, w^{-1} \rangle)$

IFFT ↑ Inverse FFT

$\begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$

read in reverse: $\begin{bmatrix} c_3 x^3 \\ c_2 x^2 \\ c_1 x^1 \\ c_0 x^0 \end{bmatrix}$

$e^{i\theta} = \cos\theta + i\sin\theta$

if $f(a) = \Omega(g(a))$

then $\log f(a) = \log(g(a))$

Union-Find

The runtime of m find operations is $O((m+n)\log^* n)$

$\log^*$ is the # of iterated log until you hit 1.

Scanned with CamScanner

Search aco.

States, actions, successor, goal-test.

Search-Tree

- Node represent entire paths to state
- Doesn't use closed set so can revisit

```
Graph-search(prob, fringe) return solution
X closed ← empty set (tree-search does
   fringe ← insert initial states
   while fringe is not empty
       node ← Remove-front(fringe
       if (Goal-test(node) return node
X      if (state[node] not in closed):
           add node to closed.
           for child-node in expand(node, prob):
               fringe ← insert(child_node)
X tree search doesn't have those lines
```

DFS

- stack, LIFO
- Don't care edge cost
- Graph search: not optimal, complete
- Tree search: not optimal, not complete
- Run in $O(b^d)$ time, $O(bd)$ space
  (b = branching factor, d = max depth)

BFS

- Queue, FIFO, don't check edge weight
- Graph/tree: complete, optimal
- Runs in $O(b^s)$ time, $O(b^s)$ space
  (s is shallowest solution)

UCS dijstra's

- use pq, complete/optimal

$$V^*(G-1) = \frac{\gamma(p)R(G)}{1-(1-p)\gamma}$$

Greedy

- only heuristics,
- use pq, going to lowest heuristic
- Graph search: not optimal, complete
- Tree search: not optimal, not complete

A* search

- pq, going to lowest of edge costs + heuristic
- Tree search: Complete, Optimal if admissible
- Graph search: Complete, optimal if consistent

Admissible: For all nodes, $0 \le h(x) \le$ cost(x, nearest(goal))

Consistent: For all edges x→y, $h(x) - h(y) \le$ cost(x,y)

heuristics are lower bound on path costs (optimistic)

M booleans has 2^M combo (for state-space) For h: good idea to use Manhattan distance

Constant h(x) = 0 makes A* behave like UCS, thus admissible and consistent

A* using h(x) = c will be optimial

CSP

of state spaces = (# of values)^(# of variables to assign)

```
Backtracking-search(csp):
  return Recursive-back({}, csp);
recusive-back(assignment, csp):
  if assignment is complete: return assignment.
  var ← select-unassigned-variable
  for each value in order-domain-value:
     if value is consistent with assignment
        add value to assignment
        result ← recursive-back(assignment, csp);
        if result ≠ failure return result;
        else remove value from assignment
  return failure
```

IDDFS = iterative deepening DFS

$O(b^s)$ time, $O(s)$ space.

Relationship b/w 2 variables create 2 arcs (arcs are directed)

arc A→B takes values from domain A base on relation A−B, plunk tail, will add 0→A to queue

higher degree more risk-taking

Forward checking: cross out values visible constraint when added to assignment

Arc Consistency with AC3

- Runs in time $O(n^2d^3)$ where n is # of nodes and d is size of domain
- Enforcing 1 arc is $O(d^2)$. Enforing an arc may remove 1 of the $O(nd)$ total variable, means we must enforce another $O(n)$ arcs. $O(n \cdot nd \cdot d^2) = O(n^2d^3)$

```
AC-3(csp): return csp, possibly w/ less domain
  queue ← queue of all arcs, initially all arcs in csp
  while queue not empty:
     (Xi, Xj) ← Remove-First(queue)
     if Remove-Inconsistent-Value(Xi, Xj):
        for each Xk in Neighbors(Xi) do
           add (Xk, Xi) to queue
Remove-Inconsistent-Values(Xi, Xj)
  remove ← false
  for each x in Domain(Xi):
     if no value y in Domain(Xj) allows
     (x,y) to satisfy constraint Xi ↔ Xj
        then delete x from Domain(Xi); removed ← true
  return removed
```

To learn R, need visit s,a,s' pair once for all.

To learn T, need visit s,a,s' pair ∞ times for all.

Tree Structure CSPs

- choose a root and turn undirected edges to directed.
- Linearize tree. Perform arc consistency in reverse linear order, then assign in forward.
- $O(n)$ edges, so $O(nd^2)$ time

Ordering

- MRV (Minimum Remaining Value)
  → Choose node w/ fewest remaing move to assign next.
- LCV (Least Constraint Value)
  → Assign value which rule out fewest values

Scanned with CamScanner

Game Trees

$1+\lambda+\lambda^2+\dots = \frac{1}{1-\lambda}$

MiniMax

- Opponents playing suboptimally will NEVER lower your score

Expectimax

- chance node.

Non-Zero

- Values for both player.

Alpha-Beta Pruning

α : MAX's best option
β : MIN's best option.

```
def max-value(state, α, β):
1. V = -∞
2. for each successor of state:
3.     V = max(V, value(successor, α, β))
4.     if V ≥ β return V
5.     α = max(α, V)
   return V.
```

for min-value
- 1. V = ∞ ...
- 2. V = min( )
- 4. V ≤ α return V
- 5. β = min(β, V)

Utilities of Lotteries

(increasing func can be both risk-averse and seeking)

The expected monetry value (EMV) of lottery is weighted sum outcome of dollar values

- Take utility of all outcome first, than apply weighted sum.

Risk-averse: perfer expected outcome (concave u. function) $\frac{d}{dx}f < 0$ ⌒
Risk-neutral: indifferent / or —
Risk-seeking: prefer lottery ⌣

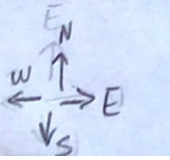
direct evalution

calculate $V^\pi(s)$ from reward from s in each episode and take average

$L = [p, A; (1-p), B]$
$U = \sum_i p_i U(S_i)$
$U_{exp} = U(\sum_i p_i S_i)$
need to pay then use $U(m-i)$

Value-Iteration

- When s get values of exit reward, k = steps, $V_k(s) = r$ = # of steps to get reward.



MDP

$U([s_0, a_1, s_1, a_2, s_2]) = R(s_0, a_0, s_1) + \gamma R(s_1, a_1, s_2) + \gamma^2 R(s_2, a_2, s_3) + \dots = \frac{R_{max}}{1-\gamma}$

$T(s,a,s')$: prob. of $s \to s'$ taking a.
$R(s,a,s')$: reward of $s \to s'$ taking a.
discount $\gamma \in [0,1]$
If know R and T use value/policy iteration

$$T(s,a,s') = P(s'|s,a)$$

Bellman Equations

$$V^*(s) = \max_a \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^*(s')]$$

$$Q^*(s,a) = \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma (\max_{a'} (s',a'))]$$

$$V^*(s) = \max_a Q^*(s,a)$$

$V^*(s)$: value of s if act optimally
$Q^*(s)$: value of taking a at s if act optimally at s' onward.

Equivalent formulas

$$V^*(s) = \max_{a \in A} \sum_{s' \in S} T(s,a,s')[R(s,a,s') + \gamma (\max_{a' \in A} Q(s',a'))]$$

$$Q^*(s,a) = \sum_{s' \in S} T(s,a,s')[R(s,a,s') + \gamma (V^*(s'))]$$

Value iteration

: For $i = 1 \dots Iter$:

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V_k(s')]$$

*convergence at $V_{k+1}(s) = V_k(s) = V^*(s)$

$O(|S|^2|A|)$ per iteration

Policies

: replace a with $\pi(s)$

Optimal policies

: $V^*(s) == V^{\pi^*}(s)$.

where $\pi^*(s) = \arg\max_a \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^*(s')]$

Policy iteration

: May need to solve system of equations

For $i = 1 \dots IterPolicy$
  For $k = 1 \dots IterValue$

$$V_{k+1}^{\pi_i}(s) \leftarrow \sum_{s'} T(s, \pi_i(s), s')[R(s, \pi_i(s), s') + \gamma V_k^{\pi_i}(s)]$$

$$\pi_{i+1}(s) = \arg\max_a \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^{\pi_i}(s')]$$

Each update of $V_{k+1}^{\pi_i}$ takes $O(|S|^2)$ time, each update of π_i is $O(|S|^2|A|)$

Model base RL

Count (s,a,s') to find $T(s,a,s')$ and we know $R(s,a,s')$

Model Free RL

↳ policy matters, suboptimally policy may not lead to convergence

Temporal Difference (TD) learning

: passive (fixed policy)

- On arrival of sample $(s, \pi(s), s', r)$ do
  $V := r + \gamma V^\pi(s')$
  $V^\pi(s) := \alpha V + (1-\alpha) V^\pi(s)$

Good for learning Q-value, not policies
Don't know anything about unvisited states
Give exponentially less weight to older samples

Q-learning

: active RL

- On arrival of sample (s,a,s',r) do
  $q := r + \gamma \max_{a'} Q(s',a')$
  $Q(s,a) := \alpha q + (1-\alpha) Q(s,a)$

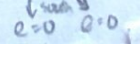
* For deterministic succor, finite k to converge, else ∞ for V to converge

* For Q(s,a) to converge, we must visit all state action pairs infinitely.

Min. state space for CSP

- Each element in s. space is partial assignment along w/ domains of all variables

Game Tree Ex:



Scanned with CamScanner

Training set: to train model
parameters: weights of perceptron
Validation (held-out) set: select model
hyperparameters: # of hidden layers
Testing set to evaluate model

Naive Bayes

Maximum Likelihood Estimate

$P(y) = \text{count}(y)/\text{count}()$

$P(a|y) = \text{count}(a,y)/\text{count}(y)$

$P(\text{class}|\text{sample}) \propto P(\text{sample}, \text{class})$

Perceptron

decision boundary defined by $w^T f(x) = 0$

Train perceptron

- initialize weight vector.
- model classify incorrectly.

False positive (should be −): $w \to w - f(x^*)$

False negative (should be +): $w \to w + f(x^*)$

bias b: $w^T x + b$

Multi-class variant

Update.

predicted (wrong) y: $w_y = w_y - f(x^*)$

True y^*: $w_{y^*} = w_{y^*} + f(x^*)$

Weakness

- Data must be linearly separable.

Modification

Feature lifing: $(x,y) \to (x, y, (x^2+y^2))$

− rear, + far

Loss function (i.e. softMax, zero-one loss)

Goal: min loss(pred(x^i), y^i; w)

Non-Linearity functions (i.e. ReLU)

Gradient Descent

$w \leftarrow w - \alpha \nabla f(w) = w - \alpha \frac{\partial f}{\partial w}$

$0.01 < \alpha < 0.1$

Batch Gradient Descent

$w \leftarrow w - \alpha \cdot \nabla_w \text{Loss}(w; x_{(1:N)}, y_{(1:N)})$

use entire training dataset; performs update after computing gradient samples.

Mini-batch Gradient Descent ← pick randomly

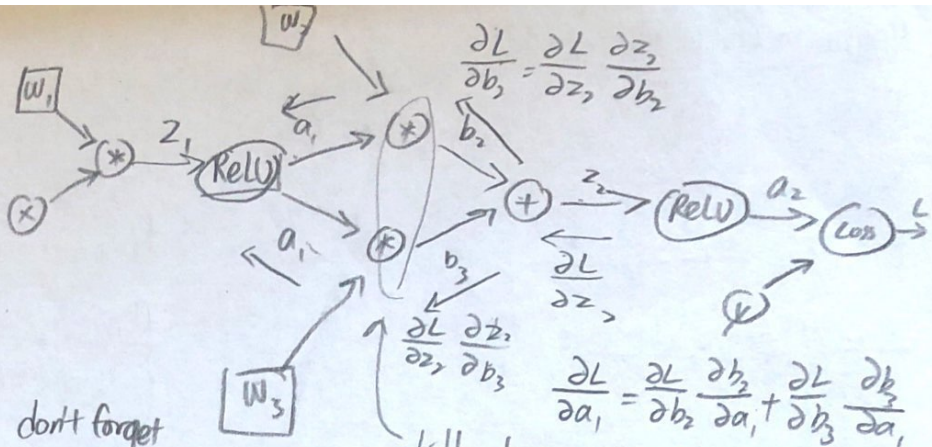
$w \leftarrow w - \alpha \cdot \nabla_w \text{Loss}(w; x_{(i:i+m)}, y_{(i:i+m)})$

performs an update for every mini-batch of n training examples

Stochastic Gradient Descent

$w \leftarrow w - \alpha \cdot \nabla_w \text{Loss}(w; x_i, y_i)$

Perform an update for a single training x_i, y_i



don't forget chain rule.

hidden layer size = 2.

Zero-One Loss

$l^{acc}(w) = \frac{1}{m}\sum_{i=1}^{m} (\text{sign}(w^T f(x^i)) == y^i)$

Softmax

$p(y=1|f(x);w) = \frac{e^{w^T f(x^i)}}{e^{w^T f(x^i)} + e^{-w^T f(x)}}$

$p(y=-1|f(x);w) = \frac{e^{-w^T f(x^i)}}{e^{w^T f(x^i)} + e^{-w^T f(x^i)}}$

Objective: $l(w) = \prod_{i=1}^{m} p(y=y^i|f(x^i);w)$

Log: $ll(w) = \sum_{i=1}^{m} \log(p(y=y^i|f(x^i);w))$

find good w: $\max_w ll(w)$ or $\min_w -ll(w)$

Finite Difference Approximation

$\nabla g(w) \approx = \left[\frac{g(w_1+h, w_2) - g(w_1, w_2)}{h}, \frac{g(w_1, w_2+h) - g(w_1, w_2)}{h}\right]$

$O(NM)$

N = number of parameters, M = # of data points

Back propagation

Exact answer, $O(NM)$

Multi-class softmax

$p(y=A|f(x);w) = \frac{e^{w_A^T f(x)}}{e^{w_A^T f(x)} + e^{w_B^T f(x)} + e^{w_C^T f(x)}}$

Activation functions

Sigmoid: $\sigma(x) = \frac{1}{1+e^{-x}}$

ReLU: $\max(0,x)$

tanh: $\tanh(x)$

Leaky ReLU: $\max(0.1x, x)$

Dropout "randomly set some neurons to 0 in forward pass"

Forces network to have redundant representation.

Regularization to prevent overfitting.

Scanned with CamScanner

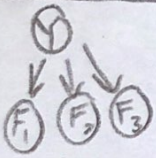
Minimum Correcting Update

$w_y = w_y - \tau f(x)$

$w_{y^*} = w_{y^*} + \tau f(x)$.

$\tau = \min\left(\frac{(w_y - w_{y^*}) \cdot f + 1}{2 f \cdot f}, C\right)$

General Naive Bayes



$|Y|$ parameters

$P(Y, F_1 \dots F_n) = P(Y) \prod_i P(F_i, Y)$

$|Y| F^n$ values $n \times |F| \times |Y|$ parameters

Inference for NB

1) Get join prop. of label and evidence for each label

$P(Y, f_1, \dots f_n) = \begin{bmatrix} P(y_1, f_1 \dots f_n) \\ \vdots \\ P(y_k, f_1 \dots f_n) \end{bmatrix} = \begin{bmatrix} P(f_1) \prod_i P(f_i | y_1) \\ \vdots \\ P(f_k) \prod_i P(f_i | y_k) \end{bmatrix}$

2) sum to get prob of evidence.

3) normalize by dividing 1) by 2)

$P(f_1 \dots f_n)$

$P(1 | f_1 \dots f_n)$

↑ compute and compare.

need $P(Y)$ and $P(F_i | Y)$ table by counting

Confidences from Classifier

Calibration

- weak calibration: higher confidences mean higher accuracy
- strong calibration: confidence predicts accuracy rate

Approx Q-learning

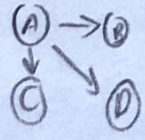
feature * weights sum to Q.

$V(s) = \vec{w} \cdot \vec{f}(s)$

$Q(s,a) = \vec{w} \cdot \vec{f}(s,a)$

difference $= [R(s,a,s') + \gamma \max_{a'} Q(s',a')] - Q(s,a)$

$w_i \leftarrow w_i + \alpha \cdot \text{difference} \cdot f_i(s,a)$



rows in BN is smaller than # rows of join table

minimax cares about relative, expectimax cares about exact.

Exploration functions

$Q(s,a) \leftarrow (1-\alpha) Q(s,a) + \alpha [R(s,a,s') + \max_{a'} f(s',a')]$ where

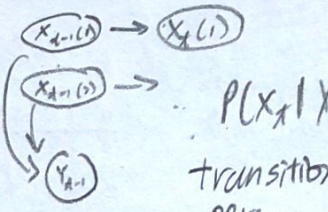
$f(s,a) = Q(s,a) + \frac{k}{N(s,a)}$ ← favors unvisited more.

Baynes Net

Number of parameters for $P(X_t | X_{t-1})$ is number of arrows minus one from a column

i.e. $N(N)$ arrows → $N(N-1)$

(# states)(# states - 1)



$P(X_t | X_{t-1}) = \prod_{n=1}^{N} P(X_t(n) | X_{t-1}(n))$

transition matrix for each person requires 2 parameters

CPT: $\sum_{X_{t+1}} P(X_{t+1} | X_t) = 1$.

Any variable which has no descendant that are query variables or evidence variables can be deleted before running elimination, w/o affecting result.

D) separator

applying linear twice = same as once

output of softMax is a probability.

$r_1 = \max(h_1, 0)$, ReLU.

when h_1 is positive, derivate = 1, else 0.

$y_1 = \frac{e^{r_1}}{e^{r_1} + e^{r_2}}$

$\frac{dy_1}{dr_1} = y_1(1 - y_1)$

If constraint graph is a tree, don't need to backtrack

Scanned with CamScanner

Likelihood weighting only takes upstream evidence into account when sampling.
if in a table X does not change with Y then $X \perp\!\!\!\perp Y$, if in A B C table $p(B)$ stays same
when A changes, than $A \perp\!\!\!\perp B \mid C$

$x_0 \to x \to \cdots x_t$ ($x_t \to v$)

$P(x_i) = \sum_{x_{i-1}} P(x_i \mid x_{i-1}) \cdots \sum_{x_0} (x_1 \mid x_0) P(x_0)$

$= \sum_{x_t} P(v, x_t)$

$P(v) = \sum_{x_t} P(v \mid x_t) P(x_t)$, can introduce x_t

$P(x_{t+1}, v) = \sum_{x_t} P(x_{t+1}, x_t \mid v)$

Probability

$P(x_1, x_2, x_3) = P(x_1) P(x_2 \mid x_1) P(x_3 \mid x_1, x_2)$

$P(x, y) = P(x \mid y) P(y) = P(y \mid x) P(x)$

Bayes $P(x \mid y) = \frac{P(x,y)}{P(y)} = \frac{P(y \mid x)}{P(y)} P(x)$

Independent if $P(x \mid y) = P(x)$

and $P(x, y) = P(x) P(y)$

X is conditionally independence of Y given Z $(X \perp\!\!\!\perp Y \mid Z)$ iff

$\forall x, y, z : P(x \mid z, y) = P(x \mid z)$

$\forall x, y, z : P(x, y \mid z) = \frac{P(x, y, z)}{P(z)}$

$= \frac{P(x \mid y, z) P(y, z)}{P(z)} = P(x \mid z) P(y \mid z)$

$P(A \mid B, C) P(B \mid C) P(C) = P(A, B, C)$

assumes nothing is independent

draw out bayes Net, have active means not guarantee independent

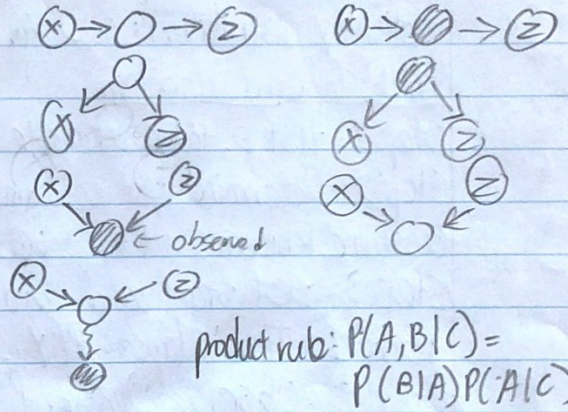
Bayes' Net

Conditional distribution for each node.

$P(X \mid a_1, a_2, \ldots)$ (one for each parent's value)

D-Separation (dependence separation)

Active / Inactive



product rule: $P(A, B \mid C) = P(B \mid A) P(A \mid C)$

Check if 2 variables are independent

1) shade all observed nodes
2) enumerate all candidate path
3) For each path:
   1) decompose into triples
   2) if all triples ACTIVE, is d-connected
4) if no path active, d-separated, guaratee conditionally independent

$EU(y) = \sum_x P(x) U(x, y)$

Decision Network

Maximum Expected Utility (MEU)

$MEU(e) = \max_a \sum_s P(s \mid e) U(s \mid a)$

$MEU(e, E') = \sum_{e'} P(e' \mid e) MEU(e, e')$

Value of Perfect Information (VPI)

$VPI(E' \mid e) = MEU(e, E') - MEU(e)$

if our observation, E', is independent of state given other evidence VPI = 0

$VPI(X) \geq 0$

$VPI(X, Y) \neq VPI(X) + VIP(Y)$

$VPI(X, Y) = VPI(X \mid Y) + VPI(E(Y))$

$= VPI(Y \mid X) + VPI(E(X))$

Sampling

Prior Sampling

sample from upstream to downstream

1. Ignore evidence and simply sample from joint probability
2. Do inference by counting right sample.

Rejection Sampling

We want $P(C \mid +s)$ → tally C outcomes, reject samples that don't have $S = +s$

Likelihood Weighting

evidence in downstream use Gibbs!

Fix evidence variables and weight samples

Weight samples by $P(\text{evidence variable} \mid \text{parents})$

Use weights of sample to compute prob.

Gibbs Sampling

Step 1: Fix evidence

Step 2: Initialize other variables

step 3: Repeat

- Choose non-evidence variable X
- Resample X from $P(X \mid \text{all other variables})$

evidence variable → $P(C \mid +s)$

product.

Scanned with CamScanner

Bayes Net: absence of edge implies independence. For a bayes net to represent a join distribution, it can only make a subset of conditional independence assumptions given by a joint.
fully independent bayes net can be represented by any
Marginalize renders neighbors of marginalized variable dependent
Conditioning fixes the observed variables and render their ancestors dependent according to d-separation

Markov Models
Treat as linear Bayes Net
A conditional probability table
for $P(X_t|X_{t-1})$
Mini-forward alg.: $P(X_t) = \sum P(X_t|X_{t-1})P(X_{t-1})$
Stationary distribution: Solve for $P(X_t) = P(X_{t-1})$

HMM forward algorithm
Step 1: 1rst p. table available to you
Step 2: Determine what we want,
assume knowing $t-1^{th}$ version
Step 3: set start and end goals
- Goal 1: Time elapse: $P(X_{t-1}|E_{1...t-1}) \to P(X_t|E_{1...t-1})$
- Goal 2: Observe Update: $P(X_t|E_{1...t-1}) \to P(X_t|E_{1...t})$

$\to P(X_t|E_{1...t-1}) = \sum_{x_{t-1}} P(X_{t-1}|E_{1...t-1})P(X_t|X_{t-1})$

$\to P(X_t|E_{1...t}) = \frac{P(X_t|e_{1...t-1})P(e_t|X_t)}{\sum_{X_t} P(X_t|e_{1...t-1})P(e_t|X_t)}$ $= P(E_t|E_{1...t-1})$

Forward algorithm
$P(X_t|e_{1:t}) \propto P(e_t|X_t)(\sum_{x_{t-1}} P(X_t|X_{t-1})P(X_{t-1}, e_{1:t-1}))$

Particle Filtering
Time update: move each sample by sampling from transition state distribution
Observation update: adjust particle weights base on likelihood.
Then: resample distribution (normalize) redrawing particles all of equal weight.

→ Step 1: Sample from initial distribution
Step 2: Sample transition probability
Step 3: Observation update; $E_1 = 1$
Step 3.5: Normalize weights, get $P(X_1)$
Throw away current Particles
Step 4: Resample from new $P(x)$

Chain rule: $Pr(X_1, X_2, \ldots X_n) = \prod_{i=1}^{n} Pr(X_i | X_1, \ldots X_{i-1})$

Naive Bayes
Assume features are conditionally independent
$\arg\max_y P(Y=(y)|\text{Observed Features}) =$
$\arg\max_y \frac{P(Y=(y); \text{Observed Features})}{P(\text{Observed Features})}$
$\propto \arg\max_y P(Y=(y); \text{Observed Features})$

Maximum likelihood estimate - Training
What distribution would have been most likely to produce this data?
i.e. count

Inference
get a new data point,
Calculate $P(Y, A=a, B=b)$
return the more likely one

Laplace Smoothing
1. add K to numerator of every entry
2. add [K*|D|] to all denominators

A node in Bayes Net is conditionally independent from rest given it's parents, children, and all children's parents

No ind. assumption:
$\sum_a P(A=a|B) = \sum_a \sum_b P(A=a, B=b)$
$= \sum_a \sum_b P(A=a)P(B=b) = 1$
$P(A,B,C) = P(A|BC)P(B|C)P(C) =$
$= P(A,B|C)P(C) = P(C|A,B)P(A,B)$
$P(A|B,C) = \frac{P(A,B,C)}{\sum_a P(A=a,B,C)} =$
$\frac{P(B,C|A)P(A)}{P(B,C)} = \frac{P(B|A,C)P(A,C)}{P(B,C)}$

Given $A \perp\!\!\!\perp B | C$
$P(A|B) = \frac{\sum_c P(A|C=c)P(B|C=c)P(C=c)}{\sum_{c'} P(B|C=c')P(C=c')} = \frac{\sum_c P(B|A,C=c)P(A,C=c)}{P(B)}$

Given $A \perp\!\!\!\perp B | C$ and $A \perp\!\!\!\perp C$
$P(A,B,C) = P(A)P(B,C) = P(A|B)P(B|C)P(C)$
$= P(A|C)P(B|C)P(C)$

Scanned with CamScanner

Transactions & Concurrency control

Atomicity: either none or all instructions executed
Consistency: database remains in consistent state afterwards
Isolation: runs as if it is the only transaction
Durability: committed changes are never lost.

Conflict serializability
Conflict serializable if acyclic (topological sort)

Two-Phase Locking (2-PL)
- Transaction obtain S before R and X before W.
- Can't get new locks after releasing any locks

Strict 2PL: Only release locks when tranx finished

Lock Manager: Granted set, Mode, Wait Queue
- Wait-Die
- Wound-Wait

Wait-for graph: an edge from T1 to T2 if T1 is waiting for a lock T2 has, has deadlock if cyclic

Recovery

Undo to preserve Atomicity
Redo to preserve Durability

Steal - system can evict/flush pages with uncommitted updates (break Atom)
No force - allow commits without forcing dirty pages to disk (break Durability)

Aries
- Assume strict 2PL

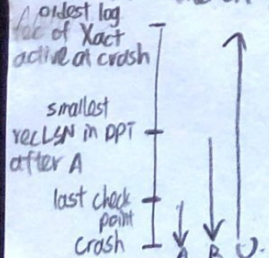
Write-ahead-logging
- Force log record for an update before corresponding data page gets to disk
- Force all log records for a transaction on commit.

ATT: Trans, lastLSN, status
DPT: Page, recLSN.

Types of Records - Transaction
When transaction commits:
- Write commit log record
- Flush log to disk
- Release transaction's locks
- Write END and remove from ATT

When it aborts:
- Write abort log record
- Write CLR for each update in reverse
- Write END and remove from ATT
- Release locks

Checkpoint
- Writes Begin_checkpoint record to log
- Writes ATT and DPT into an End_checkpoint record
- Reflect the state b/f begin_checkpoint



Analysis

Start at last begin_cp and scan.
For each record:
if Update/CLR:
  if not in ATT, add it.
  if in ATT, update lastLSN
  if page not in DPT, add it to DPT.
else if Abort/Commit
  update ATT
else if End
  remove transaction from ATT.

Redo

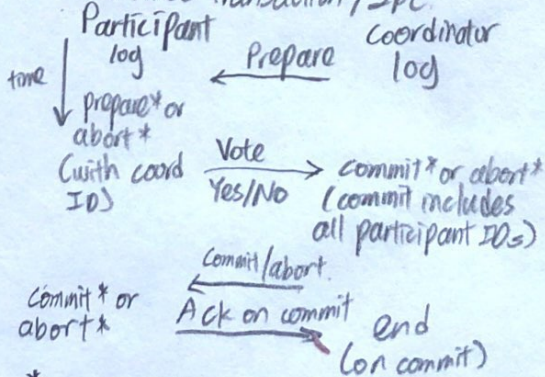
For each Update/CLR begin from smallest recLSN:
- Redo the operation UNLESS any is true:
  - affected page not in DPT
  - affected page in DPT, but recLSN > LSN
  - pageLSN (in DB) ≥ LSN

transactions that have committed should be durable (so no redo)

Undo

abort all transactions in ATT.
ToUndo = {lastLSN's of all Xacts in TT}
while ToUndo not empty:
  Remove largest LSN from toUndo:
  if CLR and undoNextLSN == null:
    Write END record for Xact
  elif CLR and undoNextLSN != null:
    Add undoNextLSN to ToUndo
  else:
    if Update w/ prevLSN:
      undo and log CLR
      Add prevLSN to ToUndo
    if Update w/ no prevLSN:
      Write END record for Xact

Distributed Transaction / 2PC



*: wait for log flush b/f sending next message

flushedLSN: max LSN flushed so far
CLRs are never undone, but may be redone.

Replication

Single-master replication:
have a "hot-standby" and sends updates every time real DB is updated
requires 2PC at every site.

Single-master log-shipping:
Send ARIES logs to single "warm" standby.

Multi-master replication:
Writes happen anywhere, updates are shipped to replicas)
- More load balancing → low latency
- Use 2PC to maintain consistency → high latency

Multi-master log-shipping:
Writes happen anywhere, logs are shipped
Replicate coordinator, use Paxos protocol

Log shipping requires less bandwith than data shipping

All schedules ∈ serializable
∈ view serial ∈ conflict serial
∈ serial

Cascadeless schedules are across all

When are updates in pool...
persist to dist? On eviction from B. pool
logged in memory? Immediate on update
logged on disk? On eviction or commit

Scanned with CamScanner

Two PC requires WAL, but not all log records must be flushed to disk b/f their message is sent

If one of the joining relations fits in RAM, Naive Hash Join can outperform Grace Hash Join

Union is pipeline breaker
Little memory, parallel hash Join is also pipeline breaker.

Selinger does not consider cross product.

$F^+ \cup G^+ \neq (F \cup G)^+$

Superkey are okay for BCNF

Sort-merge I/O,
do it by passes,
0 pass need both read (all) and write (after apply RT)

interesting orders for downstream use

The slotted page format allows tuples to be moved to a different location on the same page without changing record ID

Grace hash join:
(# of partition phase + 1)([R] + [S])

Scanned with CamScanner

SQL

SELECT [DISTINCT] <columns>
FROM <single table>
[WHERE <predicate>]
[GROUP BY <columns>
  [HAVING <predicate>]] ASC, DESC
[ORDER BY <column list>]; [LIMIT] <number>

SQL Joins (multi-table queries)

SELECT [DISTINCT] <columns>
FROM <table> ← preserve left
  [INNER | LEFT | RIGHT | FULL] [OUTER] JOIN <table>
  ON <predicate>
[WHERE <predicate>]

SQL keywords

- UNION ∪ EXCEPT
- INTERSECT ∩
- IN ← nested query, can have > ANY
- EXIST
- COUNT, ALL, ANY, NOT, NULL

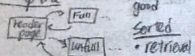
LIKE 'B%' matches strings start w/ B

Create View ___ or With Name(...,..) AS (query)
AS (query)

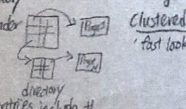
Files and Buffer Management

Files of pages, pages of record

Unordered heap file

as list: 

Page directory:



Directory entries include # free bytes or reference of page

Page Layout

Fixed length record, packed
- record number in page
- delete need re-arrange

Fixed length record unpacked
- page header has "bitmap" denotes "slots" of record

Variable length record, slotted page
- slot directory, footer.
- Each slot has pointer to beginning of record and length of record
- slot grow from end
- need reorganization to prevent fragmentation

Heap

- full scan good

Sorted

- retrieval in order, range

Clustered files and indexes

- fast lookup and modification

Buffer Management

- FrameId, PageId, Dirty?, Pin Count
Requestor of page must:
1) indicate wether page written (dirty)
2) unpin page

LRU = Least recently used
- Pages at the end of reading = last B pages
- Costly (need to find last used) and bad for sequential scan

MRU = Most recently used
- Good for sequential scan
- $\approx (B-1)/N$ hit rate

Clock Policy
- Replace, set pin, set ref bit, advance clock

Sorting

Double buffering
- Main thread runs $f(x)$
- 2nd thread drains/fills unused I/O buffs in parallel
- Same # of I/O but faster

External merge sort
- Pass 0 (conquer a batch):
  - quicksort each page and make $\lceil N/B \rceil$ sorted runs
  - Pass 1, ..., Sort B-1 runs are at a time
  - Num pass $= 1 + \lceil \log_{B-1} \lceil N/B \rceil \rceil$
  - Cost $= 2N * (\#\text{ of passes})$
  - How big a table sorted in k passes? $B(B-1)^{k-1}$

Hashing
- arranged on disk so that no 2 records that are incomparable are separated by a greater or smaller record

Record Layout

Fixed length: use arithmetic to find field

Variable length: delimiters (CSV), use footer

Record header, has pointer to end of variable length fields

Cost of Operations

	Heap File	Sorted File	Clustered ⅔ fill, Index $f=2i+1$
Scan all records	$B*D$	$B*D$	$1.5B*D$
Equality search	$0.5B*D$	$(\log_2 B)*D$	$(\log_F 1.5B+1)*D$
Range search	$B*D$	$((\log_2 B+\text{pages}))D$	$((\log_F 1.5B)+\text{pages})D$
Insert	$2D$	$((\log_2 B)+B)*D$	$((\log_F 1.5B)+2)*D$
Delete	$(0.5*B+1)*D$	$((\log_2 B)+B)D$	$((\log_F 1.5B)+2)*D$

B: Number of data blocks
R: Number of records per block
D: Average time to read/write block

Big O

Scan all records	$O(B)$	$O(B)$	$O(B)$
Equality search	$O(B)$	$O(\log_2 B)$	$O(\log_F B)$
Range search	$O(B)$	$O(\log_2 B+\text{pages})$	$O(\log_F B+\text{pages})$
Insert	$O(1)$	$O(B)$	$O(\log_F B)$
Delete	$O(B)$	$O(B)$	$O(\log_F B)$

Next block concept

- Sequential blocks on same track
- blocks on same cylinder
- block on adjacent cylinder
→ minimize seek and rotational delay
pre-fetch
- seeker time, ~2-3 ms on average
- rotational delay, ~0-4 ms
- Transfer time ~0.25 ms per 64 kB page

Indexes

ISAM, insertion and overflow cause linked list

B+ Tree

$d \in$ #entries $> d$
$f = 2d+1$
fill-factor ≥ 1/2

Insert
Overflow: leaf node split, copy mid to inner
inner node split, mid to upper inner

k = height
max elements for k: $2d(2d+1)^{k-1}$

Bulk Loading
- Fill leaf pages to some fill factor
- update parent until full, split parent to fill factor

Composite key
- Lexicographic order, all = except last one can be <, >.
- Alternative 1 Index (B+ Tree) (hybrid cluster)
  - Record contents are stored in index file (super cluster)
- Alternative 2: By reference, <K, rid of matching data record>
- Alternative 3: By list of reference, <K, list of rids of matching data records>

Cluster
- Sort heap file
→ Efficient range searches, locality, compression
→ Expensive to maintain, only ⅔ full

Prefix key compression

- On split, determine minimum splitting prefix to copy up
- compress so keys are diff w/ shortest letters

Suffix key compression

Move common prefix to header

T: Single-pass streaming w/ sequential input and output disks is nearly all sequential IOs
Single pass required for amount of RAM
External merge sort make use of one-pass streaming in merges

Scanned with CamScanner

Relational algebra (return set)

π : projection, retain columns (▯)
σ : selection, retain row (▭)
ρ : renaming $\wedge$ = and
$\cup$: union, tuples in r1 or in r2
$-$: set diff., tuples in r1 but not in r2
$\times$: Cross product, size: $|R1| \times |R2|$
$\cap$: intersection, tuples in r1 and r2
$\bowtie$, $\bowtie_\theta$: joins
ρ(relation_name (1→sid1, 4→sid2), R1×S1)
 ↖ new table name   inner
$\cap \equiv S1-(S1-S2)$.
theta join ($\bowtie_\theta$): $R \bowtie_\theta S = \sigma_\theta(R \times S)$
Natural join ($\bowtie$):
$R \bowtie S$: $\pi_{\text{unique field}}\, \sigma_{\text{eq. matching field}}(R \times S)$
Division (/).
$A/B = \pi_x(A) - \pi_x((\pi_x(A) \times B) - A)$
Group by / Aggregation
$\gamma_{\text{age, AVG(rating), COUNT(*)>2}}$ (Sailors)
 Optional HAVING

Join Costs

Notations: [R] = # of pages store R,
p_R = # of records per page for R
|R| = cardinality (# of record) = $[R]p_R$

Simple Nested Loop
for record R
 for record S:
  join r_i, s_j
Cost: $(p_R[R]) * [S] + [R]$

Page Oriented Nested Loop Join
for each page in R
 for each page in S
  for each record r in $page_R$
   for each record s in $page_S$
    join r_i, s_j
Cost: $[R] * [S] + [R]$

Block/batch Nested loop Join
for each block b_R of B-2 pages in R
 for each page in S
  for each record r in b_R
   join r_i, s_j
Cost: $\lceil \frac{[R]}{B-2} \rceil * [S] + [R]$

Nested Loop Join
Cost F = [Access cost C] + [Cardinality of C] * [Access cost F]

> Can use index on composite key for 1st column only

> n machine, B buffer each, N pages sort and hash
> $1 + \lceil 1 + \log_{B-1}(N/nB) \rceil$
> Lossless test
> ABDE ∩ BCDF = BD not contain
> BD → BDE ⊄ either decompose
> so not lossless

speedup /
scaleup −

Sort-Merge Join

```
while not done {   ← merging scan phase
  while (r<s) { advance r }
  while (s<r) { advance s }
  mark s  // save start of "block"
  while (r==s) {
    while (r==s) {
      yield <r,s>
      advance s
    }
    reset s to mark
    advance r
  }
}
```

Cost: Sort R + Sort S + ([R]+[S])   (2 passes: 4*([R]+[S]))

Grace Hash Join   Cost: [R] < ~B²

$R \bowtie_\theta S = \sigma_\theta(R \times S)$
Partition tuples from R and S by join key
Build & Probe a separate hash table for each partition

PsuedoCode

```
For Cur in {R,S}
  For page in Cur
    read page into input buffer
    for tup on page
      place tup in output buf hash_p(tup.joinkey)
      if output buf full then flush to disk partition
  Flush output bufs to disk partitions
For i in [0...(B-1)]
  For page in R_i
    For tup on page
      build tup in memory hash_r(tup.joinkey)
  For page in S_i
    For tup on page
      Read page into input buf
      Probe memory hash_r(tuple.joinkey) to match
      Send all matches to output buf
      Flush output buf if full
```

B-1 partitions, need to fit B-2 buffer page for hash table

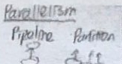
Index Nested Loop Join (index on S)

for each tuple r in R
 for each tuple s in S where $r_i == s_j$ do
  add <r, s> to result
Cost $= [R] + ([R] * p_R) *$ cost to find S tuple matching r
Alt 1: cost traverse tree from root to leaf (2-4 IOs)
Alt 2 or 3:
 1. cost to look up RID: 2-4 IO's + (alt 2 adds one more)
 2. Cost to retrieve record: → Clustered: 1 I/O per page of matching S tuple
  Unclustered: 1 I/O per matching S tuple

Parallelism

> A is superkey if A is not on RHS of every FD

Pipeline Partition



shared memory: mem + disk
shared disk: disk, share nothing (cluster)

Kinds of Query Parallism
Intra query
1. inter-operator (pipeline) vs (bushy tree parallism).
2. intra-operator (partition).
inter-query
- each query runs on separate processor
- requires concurrency control

Intra-operator parallism
Data partition
 Range: good for equijoins, range queries, group-by
 Hash: good for equijoins, group-by
 Round-robin: spreading load
cost: (2 * # of partition phase + 1)([R] + [S])

Hybrid hash join
k = (smallest page size − (B−2)) / (B−3)
# page for hash join = B − 2 − k
Cost = hash join − ...

Parallel Hash Join Grace
- Pass 1 is like hashing
- Pass 2 is local GHS per node (hash both R and S)

Parallel Sorting
- Range partition first then sort
Parallel Sort Merge
- Pass 0 → n-1 are like parallel sort
- Pass n merge partitions by node

Symmetric shuffle; same last page
Asymmetric shuffle: If R already partitioned, just partition S
Broadcast Join:
 If R is small, send it to all nodes and join at each node w/ S partitioned

Relational Query Optimization

Query Parser → Query Rewriter → Query Opt. → Query Executor
Selection:
$\sigma_{c1 \wedge \dots \wedge cn}(R) \equiv \sigma_{c1}(\dots(\sigma_{cn}(R)))$ cascade
$\sigma_{c1}(\sigma_{c2}(R)) \equiv \sigma_{c2}(\sigma_{c1}(R))$ commute
Projection:
$\pi_{a1}(R) \equiv \pi_{a1}(\dots(\pi_{a1,\dots,an-1}(R))\dots)$ cascade
Cartesian
$R \times (S \times T) \equiv (R \times S) \times T$ associative
$R \times S \equiv S \times R$
- Selection cascade and pushdown
 - Apply selections as soon you have relevant columns
- Projection cascade and pushdown
- Avoid Cartesian Product
Materialize inner loops
System R (Selinger) optimizer
1. Plan Space
 Only consider left-deep plans
2. Cost estimation
 - consider CPU & I/O costs
3. Search Alg: DP.

Translating SQL to "Relational algebra"
1. Join (FROM, WHERE, AND) use $\bowtie$
2. Selection (AND), σ_θ
3. Group By (use GROUP BY)
4. HAVING
5. Projection (SELECT), π.
$\sigma_{R.a=S.b}(R \times S) = R \bowtie_{R.a=S.b} S$
Physical properties: sort and hashed
Cost estimate
cost = # I/O + CPU-factor * # tuples
Catalogs keep stat of relations
Selectivity (sel) = |output| / |input|
Also called reduction factor.

Result size estimation

Result Cardinality = Max # tuples * product of RF's
- Term col = value : RF = 1/NKeys(I)
- Term col1 = col2 : RF = 1/(MAX(NKeys(I1), NKeys(I2)))
- Term col > value : RF = (High(I) − value) / (High(I) − Low(I) + 1)
- If missing stat, assume 1/10

Histogram
Assume uniform distribution in bin
and (∧), multiply, or (∨) add, − product of disjuncts
sel for $R \bowtie_p \sigma_q(S) = S_p S_q |R||S|$

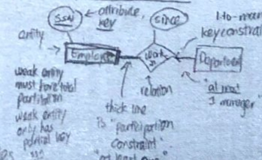
Enumeration of plans
- Single-table queries include select, projects, groupBy/agg

Cost estimate for single plan
- Index I on primary key match selection
 - Height(I) + 1 for B+ tree
- Clustered index I: I = index
 - (NPages(I) + NPages(R)) * product of RF's of matching selects
- Non-clustered index I:
 - (NPages(I) + NTuples(R)) * product of RF's of matching selects
- Sequential scan: NPages(R)

Physical DB design
- Attributes mentioned in WHERE clause are candidate for index scan (only 1 index can cluster)

Conceptual schema defines logical structure
Students(sid...)
Physical schema describes files and indexes used
- relations stored as unordered

Entity-Relation model (ER)



key + participation = one to one
Weak entity, thick border line, identified uniquely by considering p. key of another entity
binary relation better

can have more than 1 column as primary key

steps
skip superkeys and keys not connected by relation

Decomposition
Lossless, not new rows after join
if $\rho_x(r) \bowtie \rho_y(r) = r$
Dependency Preserving
if $(F_x \cup F_y)^+ = F^+$

Function Dependencies

X → Y means X determines Y
X and Y are sets of attributes
Superkey: set of col. in table determine all cols.
Candidate key: minimal set of col. that determine all cols
Have R → W, decompose to two table one with everything − W and RW
F^+ = closure of F is set of all FDs that are implied by F

Armstrong Axioms
Reflexivity: if X ⊇ Y, then X → Y
Augmentation: if X → Y, then XZ → YZ for all Z
Transitivity: If X → Y and Y → Z, then X → Z
Sound and complete
X → Y and X → Z X → YZ.
X → YZ then X → Y, X → Z

Normal Forms
1st normal: all attributes atomic
if R in BCNF, rows can't be inferred → Boyce-Codd Normal Form (BCNF)
R is in BCNF if and the only non-trivial FDs over R are key constraints

ER to relational

Entity sets to tables, simple relationship sets to tables
1) Keys for each participating entity set (as p. key and foreign key) and attributes (many-to-many)
2) Have key constraint, only that's p. key will be p. key in this relation
Participation Constraint use NOT NULL
Weak Entity: have p. key and foreign key to owner and use ON DELETE CASCADE when reference foreign key
Relational table also need p. key of entities

Scanned with CamScanner

I/O

Working set (WS) = min. # of pages needed for process to perform well

$D = \Sigma |WS_i| >$ memory ⇒ thrashing

Clustering = on page-fault, bring in pages around faulting page

Working set tracking, above 2 help w/ compulsory misses.

Reverse Page Mapping/Coremap

- For every page descriptor, keep a list of pte that points to it

Virtual memory map

32-bit	64 bit
Kernel	Kernel
User	empty (canonical Hole)
	User

Block-devices: open(), read(), write(), seek()

Character-devices: get(), put()

Blocking Interface: "wait"

Non-blocking: "don't wait"

Asynchronous: "tell me later"

I/O instructions: in/out

Memory Mapped I/O:
- load/store
- HW maps control registers and display mem into physical address space

Programmed I/O
- Each byte transferred via processor
- Pro: easy to program
- Con: consumes processor cycles

DMA (direct mem access)
- Give controller access to mem. bus
- Ask to transfer data block to/from mem. directly

I/O interrupts
- Pro: handles interrupts well
- Con: interrupts high overhead

Polling
- Pro: low overhead
- Con: waste cycles
- interrupt first then polling

Device Driver

Top half:
- accessed in call path from kernel's syscall
- implement open(), close(), read(), write(), put + to sleep

Bottom half: run as interrupt routine, may wait up & Get input

Queuing

Stable: avg. arrival = avg depart.

N = throughput (B) × res. time (L)

λ = arrival rate

T_{ser} = time to service

C = square of coef — Poisson / determistic C = 0, Disk ≈ C = 1.5

Memoryless (C = 1): arrival independent

μ = service rate = $1/T_{ser}$

u = utilization = $\lambda/\mu = \lambda T_{ser}$

T_q = Time in queue $= T_{ser} \times \frac{u}{1-u} \times \frac{1+c}{2}$

$L_q = \lambda T_q$ = length of queue (Little law)

Response time = $T_q + T_{ser}$

Modeling Queue (M/M/1)
- 1st letter = distribution of arrival
- 2nd letter = service distribution
- 3rd letter: # of servers

Mem-less: $f(x) = \lambda e^{-\lambda x}$

Disk Scheduling

FIFO
- Fair, long seek

SSTF: Shortest seek time first
- reduce seek but lead to starvation

SCAN: Elevator algorithm
- Take closest request in dir. of travel
- No starvation

C-SCAN: circular scan
- only goes in one direction
- Fairer than SCAN, no bias for mid.

One queue w/ N jobs/s better service time and utilization.

u > 1 = unbound = unstable

Files / directory

- Named permanent data storage

Logical block addressing (LBA)
- every sector has an int. address from 0 up to max # of sectors
- Open b/f read/write

One block = multiple sectors.

Open performs Name Resolution
- translate pathname to "file #"
- Hard link makes directory a dag.

File contains data and metadata
- Owner, size, last open, access right

FAT (File Allocation Table)

- most commonly used
- File # is index of file's block list root.
- Append to list to grow.
- Unused tracked via free list

FAT32 bits used in window, USB drive.

FAT stored on disk.

Format = zero blocks, link up FAT free list

Quick format = link up free list

Good sequential access

Bad random access.

No external fragmentation

Too many small f = internal frag.

Bad locality for files and metadata

Reliability

- Availability: prob that sys. can accept request, in %.
- Durability: ability for sys to recover data from fault.
- Reliability: ability of a sys to perform in a period of time

RAID 1
- Full duplicate
- 100% cap overhead.
- Bandwidth bad for write
- Write to 2 phy.
- Can optimize read.
- Hot spare: idle disk

RAID5: High I/O Rate Parity
- Data stripped across mult. disks

$P0 = D0 \oplus D1$

Parity ⊕ xor.

Increase BW for single disk

Reliability solution
- Careful Ordering
- Copy on Write File layout
- Never update in place, reuse unused, parallel, ZFS
- Transaction for atomic updates
- Begin: get trans Id.
- Updates
- Commit.

Log structured filesys, data stays in log form

Journaling filesys, log used for recovery
- All changes treat as transaction

Journaling for reliability
- Update to sys metadata using transaction, update to non-directory files done in place, used in NTFS, ext4.

Directory

- A file contains <file_name: file_number> mapping

In FAT, file attribute are kept in directory.

linked list of entry

Read root

Read first block of root

search, repeat

FAT has no access rights

FAT has no header in file blocks

Index Filesys: FFS inodes

Good seq, random, no ext frag.

Good locality.

inode_disk store pointers

inode appear in BSD 4.1

File metadata in inode.

4MB @ level 2, 4GB, 4TB

BSD 4.2 reserve 10% disk space.

- Problem: missing block due to rotational delay
- Solution: skip sector positioning, read-ahead, block group: set of track

Full logging Filesys

- All updates to disk done in transaction
- Great for random writes

Redo
- Prepare
- Commit
- Redo
- Garbage Collection
- Recovery

Writes are written sequentially in log so it's faster to write 100 random write in log fs.

Writing sequential sector is faster in normal because no cost of logging

Distributed Sys

Goal: transparency; ability of sys to mask complexity through simple interface

Networking

MAC address
- 48-bit unique id

IP address
- IP4 ⇒ 32-bit
- IP6 ⇒ 128-bit
- Assign by network admin or dynamically when computer connects to network.

Port #: 16-bit id assign to process by OS.

Globally, endpoint id by (IPaddress, Port #)

Flow control: avoid overflowing the rec. buffer

Congestion control: avoid overflowing buffer of a router along way

Layering (add header to each layer, depends app. only to IP)
- Application (skype, etc)
- Transport (Port #, flow control, TCP, UDP)
- Network (IP address)
- Datalink (MAC, addressing)
- Physical (bit, code scheme)

Fate-sharing: where state is stored

End-to-End: what func does each layer implement

Datalink

- Enable end hosts to exchange frame on same link

LAN
- group of hosts are in same proximity or phy. tech connect by switches

Network
- deliver package to new address.

Wide Area Network
- read IP dest of package, lookup forward table, send to output port

MAC -> SSN, IP -> address

all IP at Berkeley start w/ 0xA9E5

UDP -> best effort

Transport
- Provide End-to-End

UDP
- Datagram service

TCP
- Reliable, in-order
- Flow control, congestion control

lower 3 layers implemented everywhere,

Top 2 layers implemented by host

IP is the waist, interoperability,

Hard link

- set dir entry to contain f # for file

Soft Link
- dir entry contains name and path
- Map one name to another

B+ tree

NTFS (New technology File Sys)
- Variable length extents
- Sequence of <attribute: value> pair
- B+ tree
- Master File Table
- Journaling

Mem Map Files
- "map" file directly into an empty region of address space

mmap() allocate memory, use by exec

mmap better for seq or random read than normal read. (we use space)

Bitmap allocation is fixed while free list decrease as ...

ACID Properties

- Atomicity: all trans happen or none
- Consistency: transaction maintain integrity
  - Ex: no neg money
- Isolation: concurrency, exec. of trans. isolated from another
- Durability: if tran commits, effect persist.

mmap need dirty bit.

All RAID have ability to improve throughput

PA corr. to location in RAM and various devices

mmap maps user VA to a PA corr. to page cache of a particular file

Performance

Response time or latency: time to perform an operation (s)

Bandwidth or throughput: Rate at which op are performed (op/s)

Latency (n) = overhead + n/transfer cap.

Bandwidth = n/latency = $n/(S + n/B) = B/(B \cdot S/n + 1)$

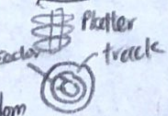
where latency = S + n/B

half-power point n = S*B, Bandwidth = B/2

Magnetic disk
- Good seq., bad random

SSD
- Good read, bad write.

HDD



Cylinder: all track under head at given point

Latency = Queuing + Controller + Seek (position head to track) + ½ Rotation + Transfer

SSD

no seek or rotational, erasing is expensive.

Latency: Queuing + Controller (find free block if for write) + transfer

Eff Bandwidth = $n/(S + n/B) = B/(1 + SB/n)$

Highest bandwidth: read

full kindle heavier than empty

Drawback of Layering
- Layer N may duplicate layer N-1 functions
- Layer may need same info
- Performance, big header

End-to-End Principle
- If hosts can implement functionality correctly, implement in network / lower layer only as a performance enhancement
- But do so only if it does not impose burden on app that do not require that function.

Phy: send bits
datalink: connect 2 hosts on same phy. media
Network: connect 2 host/machine in WAN
Transport: Connect 2 processes on (remote) hosts
App: Enable app to run.
TCP: SOCK_STREAM
UDP: SOCK_DGRAM

Flow Control
- Producer-consumer w/ bounded buffer

Circular buffer
- Byte k stored in pos. (k mod N) + 1.

AdvertisedWindow = MaxRcvBuffer − (LastByteRcvd − LastByteRead)
SenderWindow = MaxRcvBuffer − (LastByteSent − LastByteAck)

General Paradox
- 2 general can't coordinate simult. attack using unreliable network

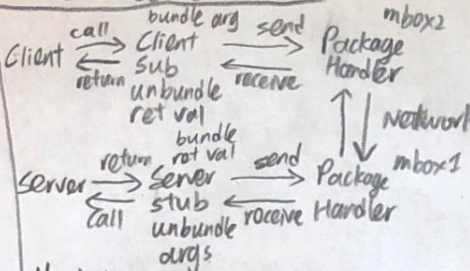
2PC algorithm
Coordinator: Send Vote-Req → Worker: Wait Vote-Req, Send Vote-Commit or Vote-Abort (P-phase)
If all Vote-Commit, send Global commit, else send Global-Abort → Obey Global commit or abort (C-phase)

If coord fails, worker must block waiting for coord to recover and send global
2PC ensures durability
Upon recovery, it can restore state and resume:
- Coord aborts in INIT, WAIT, or ABORT
- Coord commits in COMMIT
- Worker aborts in INIT, ABORT
- Worker commits in COMMIT
- Worker ask Coord in READY
- Worker waiting for global decision can ask fellow worker.

PAXOS: doesn't have block problem.

BFT (Byzantine Fault Tolerance)
- Allow machine to make coordinated decision even if (< n/3) malicious.

Byzantine General Problem
- 1 coord, n-1 works
- with f faults need n > 3f
to solve, impossible if n = 3.

Remote Procedure Call (RPC)


Marshalling / bundling means converting values to canonical form, serializing obj
Equivalent to regular procedure call
Compiler generate stubs:
stub generator, IDL
Binding: process of converting user-visible name into Network endpoint
Static: fixed at compile time
dynamic: performed at runtime, DNS
- access control, fail-over
- if server fail, use another one

Problem: Non-atomic failure
- use distributed trans / Byzantine commit
- Problem performance − cache

send B, 2 B's received ⇒ endianness reversed

Key-Value storage
put(key, value), get(key)
- Distributed Hash Table (DHT)
- Fault tolerance, scalability, consistency

Directory-based architecture
directory maintains mapping btw key and machines (node) that store value
W + R ≥ N + 1 ⇒ Quorum Consensus
- At least 1 node contains update.

Security
Authentication
- Ensures user is claimed
Data integrity
- Ensure data not change from src to dest.
Confidentiality
- Ensure data read by authorized users
Non-repudiation
- Can't later say didn't do

Symmetric key
- Same key for encryption and decryption.
- Vulnerable to tampering and replay attack
- Can use XOR, or XOR w/ one time pad

Block Ciphers
- Can encrypt block separately
→ Same plaintext ⇒ same ciphertext
→ add counter to previous block
→ Data Encryption Standard (DES) → 56-bit key, bank use triple
→ Advance Encryption Standard (AES) − 128, 192 or 256 bits.

Authentication via Secret key
- A can ask B to decrypt a random value x, vulnerable to man-in-middle attack.

Secure Hash Function
- Hash func: short summary, digest, of data, fixed length
H is secure if
- infeasible to find M_2 with $h_1 = H(M_2)$, message w/ same digest
- A small change in message change many bits of digest
- Hard to find $H(m_3) = H(m_2)$

Cryptographic Hashing
- Basic building block for integrity
- H(x) is publicly known hash func
- Digest d = HMAC(K, m) = H(K | H(K | m))
- HMAC(K, m) is a hash-based message authentication function
  - send d and m to receiver
  - Reciever use K, secret key, to recompute HMAC(K, m) to see if agree w/ d.

Standard Cryptographic Hash Func
MD5
SHA-1 (SSL/TLS, SSH)
SHA-2

Key distribution
3rd party, Auth. Server, Kerberos

Usage
- A ask S for key to talk to B
  - Not encrypted, everyone knows
- $S \to A$: [Use K_{ab} (This is A! Use K_{ab}) K_{sb}] K_{sA}
- $A \to B$: Ticket [This is A! Use K_{ab}] K_{sb}

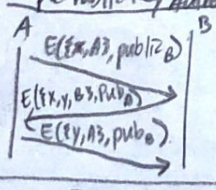
B knows K_{AB} sanctioned by S.
can include timestamp or checksum

Asymmetric encryption
public key (e) for encryption, private key (d) for decryption.

Authentication
- combo of public and private

Alice → Bob: [(I'm Alice) Private Rest of Mess] B_{public}
RSA use in ssh, SSL/TLS for https
Elliptic Curve Cryptography (ECC) shorter key & sign than RSA.

Properties of RSA
- Generate large, random prime.
- much smaller
→ use public key crypto to key. Exchange a (short) symmetric session

Simple Public Key authentication


Non-Repudiation
Alice publish E(x, KD) where KD is her private key to prove who she is, a signature

Digital Certificates
Trusted authority (Verisign) signs binding btw A and p. key K_V private
C = E(§A, K_E§, K_V private)
C = digital certificate
To get A's public key, use D(C, K_V public) = §A, k_E§

Key Value store Challenge and solution
- Fault Tolerance → replication
- Scalability → cache, parallel get,
- Consistency → Quorum consensus

Recursive query system.
Adv: Faster, easier maintain consistency
DBadv: scalability bottleneck

Iterative query system.
Adv: scalable
dBadv: slower

Consistent hashing
- Treat possible hash space as a circle.

TCP seqno field to achieve logically ordered packets.
Client open TCP w/ server
3-way handshake
- client send random sequence number (x) to server, server send random seqno (y) and ack (x+1), client ack and send by y+1

Subnet mask tells you if host is local or remote
TCP servers: socket, bind, listen, accept.
TCP client: socket, connect.

Pipes are unidirectional, Socket are bidirectional
Use poll, select to wait on multiple TCP socket in single thread.

Scanned with CamScanner

Computers can share IP address using NAT
2^{24} IP address in /8 block.
TCP provides reliable delivery and congestion avoidance.
Window size: add. increase, multi. decrease
AdvWin is for new data, resend okay

Even if advWin is 0 and all packets have been acknowledged, sender will still attempt to send 1 byte of new data.

RPC can suffer from network/server failure, not local calls
With DMA, CPU does not involve in transfer of each byte, only set up
Modern controller:
RAID, Track buffer, DMA, disk scheduling (elevator)
Sector interleaving
IP connects 2 machines, need TCP/IP to connect 2 processes in 2 machines
Page fault trap are unaffected by disabling interrupts
2PC guarantee all player commit or all abort, not simultaneous
Pointers are 4 bytes
Pinto do not have sbrk()
service = Queuing + controller + seek + 1/2 rotational + transfer
rot = 1/x RPM
BW = data transfer / time to service
max IO per s (IPOS)
= # disk * IPOS(1 disk)
= # disk * (1 / service time).
FFS, need 2 read to get first byte of data
(1 for inode, 1 for first block)
cylinder group for reliability and performance.
Access right are check on file open/create.
Transaction D committed once written to log
Change in log then change in disk
Uint32_t is 4 byte long
When check block #, check it is less than current length / Blocksize
UNIX 4.2 BSD attempts to allocate data block contiguously
K→M→G→T
1B = 8 bit
DNS is recursive, distributed service
TCP slowstart algorithm, start w/ small transmission window, increase for each ACK and divide by half when start dropping.

Belady's anomaly
Non Volatile Ram
Can improve durability to a fs. for write-behind
TCP window size = BW × latency.
How large of a disk can fs support
$2^{\#\text{bits}} \times$ Blocksize
Hard links points to same inode, soft links simply list a dir. entry.
Redundant Array of Inexpense Disks
BSD 4.2, placing data and inode in same cylindar group and data in seq block
A "broadcast network" is one which multi. receivers can receiver a message at same time from a single sender
Max BW ⇒ look for bottle neck in Network
data BW = BW × $\left(\frac{\text{MTU-header}}{\text{MTU}}\right)$
Optimal window size = roundtrip latency × effective BW
BSD has 10 direct pt
Index allocation need to read in index block
need link count to support hard links
To support soft link, add a new type of file.
int Symlink(char *src, char *dest)
For FAT at most 2 I/O needed to find random page,
basic linked allocation need N-1
2PC Guarantee A and D
Stub handles marshalling.
Log structed fs write just append and read need to through log
FFS use bitmap to track allocation of inode (same for Pintos)
Ack doesn't count toward throughput (bytes/s)

unlink operation
update inode
freemap and block freemap
Hard link count persist on disk, so struct inode_disk
nonce used to prevent replay attack.
HMAC = Hash-base Message Authentication Code
-create secure message digest, input, n, key
AdvWin = MaxRcvBuf − (LastByteRecv − LastByteRead)
SendWin = MaxRcvBuf − (lastByteSent − lastByteAck)
After Not Getting Ack, timeout
AdvWin only for new data, If we see Ack lower then can resend
Easy solution receiver send another Ack
FAT need to perform a disk access for every block to find pt to that block
if process waits and thread waits, then deterministic output
RAID 5 strip unit
Look at routing table to use Ethernet or Wifi
TCP servers: socket, bind, listen, accept.
TCP Clients: socket, connect
} socket
If Master does not see commit, must abort
Master block entire sys when crush during commit b/f send global commit.
DMA move data b/w mem and I/O devices.

iterative query system:
- Client ⇌ master
  ↓↑ worker
- more scalable
- dis: slower, harder to enforce data consistency
- Ask master to get right worker

Recursive query system:
- Client ⇌ master
  ↓↑ worker
- Faster
- Easier to maintain consistency
- dis: scalability bottleneck, especially for large "values"

large value use iterative approach.

Scanned with CamScanner

In journaling, log on the side for important stuff like metadata updates. data still exist as data on disk.

In log-structured fileys, the log is the file system. The primary record of the data is the log. The file is tracked not as a "solid file", but as the result of several "logged actions". Make small write nice but need to rebuild the file from actions to read.

Raw # of cylinders and arm speed in arms/s then use $\frac{1}{3}$ cylinders/arm speed for seek
rot still 1/2

The C in calculate T_{io} is base on distribution of service (i.e. 2nd letter)

Scanned with CamScanner

- allocation of page frame
  - Global vs local
  - Equal allocation
  - Proportional allocation
    - s_i = size of processor p_i
    - $S = \Sigma s_i$
    - m = total # of frames
    - $a_i = p_i = \frac{s_i}{S} \times m$

Priority allocation
- use priority instead of size

Page-fault frequency allocation
- if actual rate too low, processes lose frame

Thrashing = a process busy swaping in and out.

Δ = Working-set window = fixed # of page references

Δ small ⇒ not encompass entire locality

$D = \Sigma |WS_i|$ = total demand frame

if $D > m$ ⇒ thrashing.

- Clustering
  - On page fault - bring in pages "around" faulting page.
  - Good for sequential read.

Operational Parameter for I/O
- Byte (keyboard) vs block (network)
- Sequential vs random

Block devices (read(), write().)
character devices (get())
Network: socket

Timing
Blocking interface: "wait" (sleep til ready), put proces to
Non-blocking interface: "Don't wait"
- return quickly

Asynchronous interface: "Tell me later"
- notify later to user

Cache
"memoryless" - independent source of event

offset bit = size of cache blocks in bytes
index bit = size of each bank
= cache size / (associativity, size of cache block)

vast are tag bits.

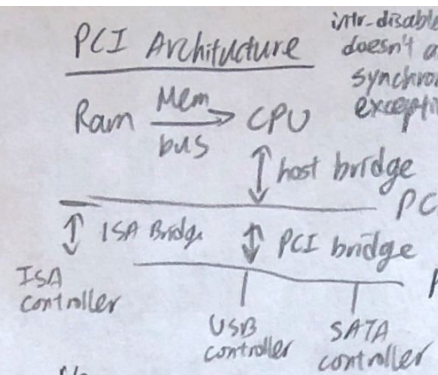
Clocks needs Dirty, Use, writable and Valid bits in PTE.

MIN amount of TLB entries = # of pages for WS for 1 process.

Belady's anomaly

demand paging: fully-assoc, wb, block size: 1 pg.

PCI Architecture (intr-disable() doesn't affect synchronous exception (trap))



Memory-mapped I/O: load/store instructions
I/O instructions: in/out instructions

- Programmed I/O
  - Each byte transferred via process in/out or load/store
  - DMA (direct mem access)
  - Give controller access to mem bus
  - Ask it to transfer data blocks.
- I/O interrupts
  - Generate interrupts when need service
  - Pro: handles unpredicted events well
  - Con: high overhead.
- Polling
  - Periodically check register
  - Pro low overhead
  - Con: may waste cycles.

Device Drivers: Device-specific code in kernel interact directly w/ device hardware.
Top-half: accessed in call path from syscall, interface, put thread to sleep til done
Bottom-half: run as interrupt routine, wake thread up when I/O complete

Latency(n) = Overhead + n/transferCapacity
= S + n/B

Bandwith = $n/(S+n/B) = B*n/(B*S+n)$
$= B/(B*S/n+1)$

disk latency = queueing time + controller time + seek time + rotation Time + Xfer time.

pred burst time: $\alpha(Burst) + (1-\alpha)(Estimate_{x-1})$ = Estimate

imposing strict order prevents deadlock.

process need to flush TLB on context switch.

cache with same index goes into same slot, use tag to check if correct.

don't forget
0x0 ⇒ 0000 in binary

Swapping will occur as phy memory ≤ ΣRss
thrashing we should add memory

B and b and segmentation
changes offset
demand paging is fully associative, thus conflict miss = 0.
All syscall share interrupt 0x30.
synchronization primitive vs semaphore
1 byte = 8 bits, K, M, G, T, P.

Totally Fair Scheduler
Thread will run: $\max\left(\frac{T_i.priority}{\Sigma T_j.priority} \cdot min\_latency, min\_quanta\right)$

IPT
size is ∝ to physical mem, global,
entry contains task id and virtual address

double phy. mem ⇒ double pages ⇒ PTE bit ↑ 1
entries for page map is determined by size of virtual address and size of page
bits in PTE determine by control bit and # phy page.
page size double ⇒ PTE bit ↓ 1
Stack grows down, Heap grows up

IPT
add up proces ID, VPN, and access info bit and that's IPT entry, times # physical pages that's space needed.

Banker's algorith is slow, re-evaluate @ every request.
Hard to know MAX

No dual-mode ⇒ no virtual mem because unprotected and can change address translation info

size of pointer is 4

copy-on-write
to efficiently implement virtual copies
location is copied into page table and both set to read-only, when one write it will actually copy and change to read-write after allocate new page

Address = frame * frame_size + offset

External fragmentation may occur in B and B and Segmentation

with DMA, CPU doesn't have to transfer of each byte.

when pg fault, OS invalidate TLB entry and pte corresponding to evicted pg

pg fault ⇒ update PT/IPT.
thread_current()

Scanned with CamScanner

Scheduling
Lottery for MFQS prevents starvation.
Earliest deadline first is schedulable if
$\sum_{i=1}^{n} \left(\frac{C_i}{D_i}\right) \le 1$
Quanta for CPU-bound job:
max(min quanta, $\frac{w_i}{\sum w}$ (target latency))
SRTF is optimal
MLFS
higher priority queues are "foreground" tasks
Each queue has own scheduling algorithm.

Real time schulding
hard → meet all deadline
soft → meet all w/ high prob.

Deadlock
• mutual exclusion
- only 1 thread at a time can use a resource
• Hold and wait
- Thread holding resource is waiting for other thread.
• Non-Preemption.
- Not voluntarily give up, only completion
• Circular wait.

Resource Allocation Graph
- Request edge - direct edge from $T_1 \to R_1$
- Assigment - $R_1 \to T_1$
Deadlock ⇒ starvation.

Handling deadlock
- ensure never go deadlock
- ignore and pretend deadlock doesn't exit (UNIX)

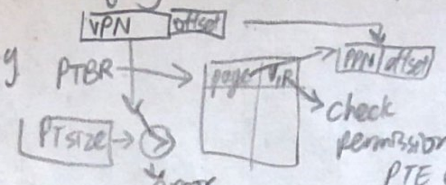
Deadlock detection algorithm
[FreeResources]: currently free resources of each type
[Request$_X$]: current request from thread X
[Alloc X]: current resouce hold by X.

```
[Avail] = [FreeResources]
Add all nodes to UNFINISH.
do {
    done = true;
    Foreach node in UNFINISH {
      if ([Request_node] <= [Avail]) {   // if ([Max_node]-[Alloc_node] <= [Avail])
                                         //   for banker's algo.
        remove node from UNFINISH
        [Avail] = [Avail] + [Alloc_node]
        done = false
      }
    }
} until (done)
```
- Nodes left in UNFINISH ⇒ deadlock

Prevent deadlock
• Inf. resource
• No sharing
• Don't allow wait.
• Request everything @ beginning
• Order request order
Unsafe ≠ deadlock

Translation
virtual → physical

PTBR = page table base register.

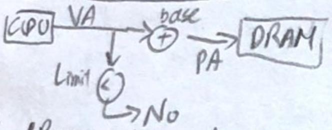
Paging (one per process)


PTE contains: valid, R, W
use bit for clock
Multi-level page table to fix sparse address space
- on context switch: save single PageTablePtr Register
Sharing point to same PPT.
Space used: $(1 + 2^{\#entries} + \ldots + 2^{\text{level}-1 \cdot \#entries})$ · size of page

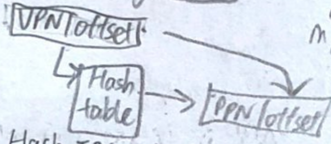
PTE example
PPN | Free | R, W, D, V..

Pintos doesn't have sbrk().
memcpy(des, src, s)

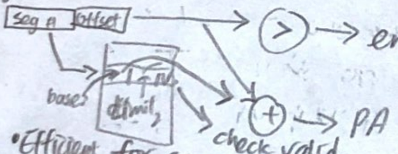
Uniprogramming
• run @ same place in physical memory
• can access any physical address.

Base-and-bound


• Program get continous region of memory
- fragmentation
- Bad for sharing, sparse address

Inverted Page Table


Hash IPV with Anchor Table
- Avoid collision shifting problem.
- Hash VPN → lookup in anchor table → lookup in page frame table

Segmentation


• Efficient for sparse address space
• allow stack and heap to grow.
• need protection in seg table
• stored on context switch
- segment table store in CPU
- might store all process mem onto disk
• Must fit variable-size chuck
• External fragmentation: gap b/w allocated chunks
• internal: don't need all mem within allocated chunks
• Can have mem protection if each process has b and b

Coremap (Reverse page mapping)
Physical page → virtual page
for clock algorithm
• for every page descriptor, keep linked list of PT entries
• Object-base reverse mapping

Second-chance list algo
split mem in Active list and SC list.
• Access pages in Active @ full speed
• Otherwise page fault
• Overflow from Active to SC
• Access more SC to active
• 0 on second = FIFO
• all on second = LRU.
• Pro: few disk access
• Con: overhead trapping to OS

in syscall handler use f→eax = ... to "return"
int is 4 byte.
char 1 byte

Kernel → User
- init. address-space control block
- read program off disk & store in mem
- initialize PT.

User → Kernel ("trap")
- Syscall.
• Synchronous exception
- seg fault, page fault
• interrupts are asynchronous exception, can be disabled

Caching
• Temporal locality
- recently accessed data close to processor
• Spatial locality
- contiguous blocks are close to each other.

Directed-Maped Cache
Cache Tag | Cache Index | Byte select
match cache tag and index
N-set associative cache
match N of them
Fully associative doesn't have cache index bits
TLB caches translation
Write through / back

Context switch
• Invalidate TLB: simple but expensive.
• include Process Id in TLB

compulsory misses: fix by prefetch
conflict misses: increase cache size or associativity
capacity misses: increase cache size
coherence misses: cause by external processes

Precise exception
• Precise ⇒ state of machine is preserved as if program executed up to offending instruction.

Effective Access Time (EAT)
EAT = Hit Rate × Hit Time + Miss Rate × Miss Time
= Hit Time + Miss Rate × Miss Penalty
MIN: replace page not used for longest time
Belady's anomaly
- FIFO miss rate doesn't drop when ↑ frames
FIFO not stack algorithm.
LRU approx. MIN.
Clocks approx LRU.

othello program printf for MLQS

N-chance version of clock
large N approx LRU well
small N for efficiency.
clean page use N=1, dirty N=2.
PTE-entry for Clock: Use, modified, valid, read-only.
if don't have hardware "use" bit, trap to OS, mark page read-only

Scanned with CamScanner

Prove K_n can be edge colored w/ n colors.
Number vertice $\in 0,\dots,n-1\rangle$, and colors same
For edge(i,j), color it by $(i+j) \bmod n$
For any edges adjacent to vertex i, (i,j) and (i,j') we have $i+j \neq i+j' \bmod(n)$, thus legal

Prove a graph w/ d max. degree can be edge colored by $\geq d-1$ color.
Remove an edge and colors the rest, the edge is incident to two vertices and each has at least $d-1$ other edges for the removed each is next to at most $2d-2$ edge, use the extra color.

$K_{3,3}$ with one crossing

prove can't/can have only one crossing.
Remove an edge and use euler ($e \leq 3v-6$)
if true.

$a \mid x \Leftrightarrow x = ca$
(a divides x; c constant)

Prove $\sum_{i=1}^{\infty} \frac{1}{3^i} < \frac{1}{2}$ using $\sum_{i=1}^{\infty} \frac{1}{3^i} < \frac{1}{2} - \frac{1}{2 \cdot 3^n}$

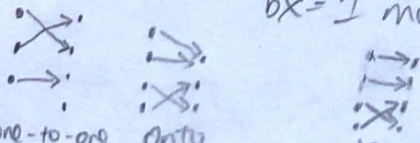
What theorem allows us to know the existence of multiplicative inverses?
Let n, x be positive integers. Then x has a multiplicative inverse modulo n if and only if $\gcd(n,x)=1$.

Prove $\gcd(a,m)=1 \Leftrightarrow ax \equiv 1 \bmod m$ existance
x has a mult. inverse mod n, then $\gcd(n,x)=1$
prove by contradiction, $d>1$

$$xa \equiv 1 \bmod m$$
$$xa = bn+1$$
$$\frac{xa}{d} = \frac{bn}{d} + \frac{1}{d}$$

Contradiction $\frac{1}{d}$ is not in which other two is.
If $\gcd(n,x)=1$, then x has a mult inverse mod n.

$$an + bx = 1$$
$$bx \equiv 1 \bmod n$$



one-to-one Not onto | Onto Not one-to-one | bijection, have inverse.

No two distinct professors have neat handwriting unless they both have the last name "Sahai."

$$\forall p_1, p_2 \in \text{Profs}, p_1 \neq p_2,$$
$$[\neg(\text{Sahai}(p_1) \wedge \text{Sahai}(p_2)) \Rightarrow \neg(\text{Neat}(p_1) \wedge \text{Neat}(p_2))]$$

Surely connected for n vertices
try to make biggest disconnected graph, all nodes connect to everything except one node
$k > (n-2)(n-1)/2$

if $x \equiv 1 \bmod 6$ then
$x \equiv 1 \bmod 2$ and $x \equiv 1 \bmod 3$

$$\forall x(P(x) \wedge Q(x)) \Rightarrow \forall x P(x) \wedge \forall x Q(x)$$
$$\exists x(P(x) \vee Q(x)) \Rightarrow \exists x P(x) \vee \exists x Q(x)$$
$$\forall x P(x) \vee \forall x Q(x) \Rightarrow \forall x(P(x) \vee Q(x))$$

$$(a \wedge b) \Rightarrow c \equiv (a \Rightarrow c) \wedge (b \Rightarrow c)$$
$$(a \vee b) \Rightarrow c \equiv (a \Rightarrow c) \vee (b \Rightarrow c)$$
$$P \wedge (Q \vee R) \equiv (P \wedge Q) \vee (P \wedge R)$$
$$P \vee (Q \wedge R) \equiv (P \vee R) \wedge (P \vee Q)$$
$$(A \vee B) \wedge (C \vee D) \equiv (A \wedge C) \vee (A \wedge D) \vee (B \wedge C) \vee (B \wedge D)$$
$$(A \wedge B) \vee (C \wedge D) \equiv (A \vee C) \wedge (A \vee D) \wedge (B \vee C) \wedge (B \vee D)$$

Chinese Remainder Theorem
$x \equiv a_1 \bmod m_1$, $\gcd(m_1, m_2) = 1$
$x \equiv a_2 \bmod m_2$ egcd(m_1, m_2) gives $1 = b_1 m_1 + b_2 m_2$
$x \equiv a_2 b_1 m_1 + a_1 b_2 m_2 \pmod{m_1, m_2}$

Bijections (injective)
f is one-to-one if all $a \in A$; $f(a) = f(b)$ implies $a = b$
f is onto, subjective, if for all $b \in B$, there exist some $a \in A$ such that $f(a) = b$

Scanned with CamScanner

intersection: $A \cap B$
disjoint $A \cap B = \emptyset$
union: $A \cup B$
$B \setminus A = \{x \in B \mid x \notin A\}$
natural #s: $\mathbb{N}$
integers: $\mathbb{Z}$
rational #s: $\mathbb{Q}$
real #s: $\mathbb{R}$
complex: $\mathbb{C}$
and: $\wedge$
or: $\vee$
not: $\neg$

$P \Rightarrow Q \equiv \neg P \vee Q \equiv \neg Q \Rightarrow \neg P$

Exactly one X, Y, Z true.

$(X \wedge \neg Y \wedge \neg Z) \vee (\neg X \wedge Y \wedge \neg Z) \vee (\neg X \wedge \neg Y \wedge Z)$

$\neg(P \wedge Q) \equiv \neg P \vee \neg Q$

$\neg(\forall x) P(x) \equiv \exists x (\neg P(x))$

$a \mid b$ a divides b

Use strong induction of $\sum_{i=1}^{n} \frac{1}{i^3} \leq 2 - \frac{1}{n^2}$

if ask to prove $\sum_{i=1}^{n} \frac{1}{i^3} \leq 2$.

Sum of degrees $= 2|E|$

If $\gcd(x,y) = 1$, then $mx + ny = 1$

egcd(x,y) returns $(d, a, b) := d = d = ax + by$

```
egcd(x,y)
if y=0 then return (x,1,0)
  else
      (d,a,b) := egcd(y, mod(x,y))
      return (d, b, a - floor(x/y) * b)
```

```
gcd(x,y)
if y=0 return x;
else return gcd(y, x mod y)
```

$e \leq 3v - 6$ ⇐ true for planar graph.

Euler's formula: $v + f = e + 2$ ⇐ connected graph

complete graph: $|E| = n(n-1)/2$ $|V| = n$

hypercube: $|E| = n2^{n-1}$ $|V| = 2^n$

In planar graph, edge is adjacent to at most 2 faces

average degree $= \frac{2|E|}{|V|}$

Improvement Lemma: If M proposes to W on kth day, then on every day later W has someone on string at least as much as M. (she likes)

Well-Ordering Principle

First-day...

If $S \subseteq \mathbb{N}$ and $S \neq \emptyset$, then S has a smallest element

The stable marriage terminates with a pairing.

Strengthening Induction Hypothesis

Assume $0 \leq i \leq k$ is true for i., $\sum_{i=1}^{n} \frac{1}{i^2} \leq 2 - \frac{1}{n}$.

Bipartite graph: $2e \geq 4f$

Does a stable marriage that is neither male optimal or female optimal. Find the optimals first and try all comb.

Hamiltonian Path - visit each vertex exactly once

Path: $(v_1, v_2)(v_2, v_3) \ldots (v_{k-1}, v_k)$, no repeat vertex.

Cycle: Path w/ $v_1 = v_k$

walk: sequences of edges w/ possible repeated vertex or edge

tour: a walk that starts and end at same node.

either or means xor.

$P \Rightarrow Q \equiv$ if P, then Q; Q if P; P only if Q; P is sufficient for Q; Q is necessary for P; Q unless not P;

$P \wedge \neg P = \text{False}$ P is 123

$P \vee \neg P = \text{True}$

True $\vee Q$ = True

False $\vee Q = Q$

True $\wedge Q = Q$

False $\wedge Q$ = False

Scanned with CamScanner

Timing

stable at: # reg in path + clk-to-q.

Input goes to output on rising edge of clk after clk-to-q time.

$f_{max} = 1/t_{critical\text{-}path}$

setup: (max. over all path)

clk-to-q + CL + setup_time < period

Hold: (min. over all path)

clk-to-q + CL > hold_time

Critical path: longest delay

1 GHz = 1 ns = 1000 ps

CPU

* stall can happen at fetch or decode; at any given time can both only do one by 0

1. Instruction Fetch (IF)
   - Fetch Inst. and increment PC
2. Instruction Decode (ID)
   - Gather data from field *
   - read opcode then read data.
3. Execute (ALU)
   - Arithmetic-Logic Unit (*, +, -, ...)
4. Memory Access
   - Only load and store do anything here
5. Register Write
   - Write data back to register *

Register Transfer Language (RTL)

Ex: Add rd rs rt

R[rd] = R[rs] + R[rt]; PC += 4

Mem[R[rs]] ← if use memory

Controls

RegDst, RegWr, nPC_sel, ExtOp (is "x" if ALUSrc = 0), ALUSrc, ALUCtr, MemWr, MemtoReg

Pipeline Hazards

1. Structural Hazards (sol: read/write same cycle)
   - Required resource is busy
2. Data Hazards (sol: forwarding)
   - data dependency
   - Wait for previous write to finish
3. Control Hazard (stall)
   - branches (add 2 bubbles)

- Stalling instruction for a cycle - bubble

* Read data at ID stage, write data at WB stage.
* sometimes read/write on same cycle, other don't

data hazard: stall until ID and WB same block (assume read/write same clock)

branch hazard: stall until IF one clock after branch know (i.e. MEM)

Cache

Spatial locality (put entire block to cache) * false sharing: 2 proc. share same cache.

Temporal locality (put accessed data into cache)

Cache size → amt. of memory cache can hold.

blocksize → size of cache block.

N-way → associativity
= how many data blocks can be in the cache for same set/index
= how many data blocks need to be checked for a cache access

Cache write hit policy:
- Write-back (dirty-bit): write to cache
- Write-through: write to mem directly.

Cache write miss policy: No write allocate: store count as miss
Write allocate: write in mem and put in cache.

Address → cache location
- Index - which blocks in cache do we need to check for our data (which set)
- Offset - which specific byte we are referring to in a block.
- Tag - use to check if data in a block in matching set corresponds to our address.

T:I:O Reserve Engineer: look at $hit_0 \cdots hit_n$ to look for associativity

Offset bits = $\log_2$(blocksize) $2^{10} = 1024$

Index bits = $\log_2$(# sets)

valid bit in cache = #blocks

tag bits in cache = valid bit * tag bit

sets = # blocks / N

blocks = cachesize / blocksize

sets = cachesize / (blocksize * N)

TAG >= $\log_2$(cachesize) - $\log_2$(N) - # offset bits

= $\log_2$((memory size / cachesize) * N)

= Address_length - # Offset_bits - # Index_bits

*Cache size = N * 2^(# offset bits + # index bits)

* if two addresses differ by multiple of 2^(# offset bits + # index bits) they map to same set in cache. (same index & offset, different tag)

↑ associativity (N) ⇒ Tag bits ↑; index bits ↓

↑ cache size ⇒ Index bits ↑, tag bits ↓

↑ block size ⇒ offset bits ↑, index bits ↓

Compulsory misses = first time accessing miss

Capacity misses = making fully associate won't make it hit

Conflict misses = making fully associate will make it hit

Average Memory Access Time (AMAT) = hit time + miss rate × miss penalty

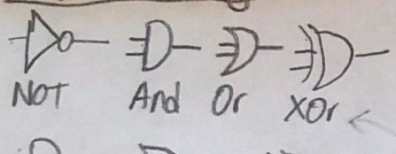
global miss at L2 = local miss at L2 × local miss at L2

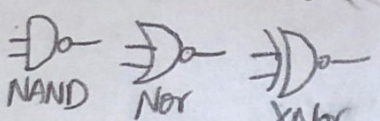
clock period low ns = high throughput

penalty = # of stalls × clock period

Scanned with CamScanner

Combinational Logic





"+" for or ⊕ for xor
"×" for and
"~" or "Hat" for NOT

$X\bar{X}=0$ $X+\bar{X}=1$
$X0=0$ $X+1=1$
$X1=1$ $X+0=0$
$XX=X$ $X+X=X$
$XY=YX$ $X+Y=Y+X$
$(XY)Z=X(YZ)$ $(X+Y)+Z=X+(Y+Z)$
$X(Y+Z)=XY+XZ$ $X+YZ=(X+Y)(X+Z)$
$XY+X=X$ $(X+Y)X=X$
$\bar{X}Y+X=X+Y$ $(\bar{X}+Y)X=XY$
$\overline{XY}=\bar{X}+\bar{Y}$ $\overline{X+Y}=\bar{X}\bar{Y}$

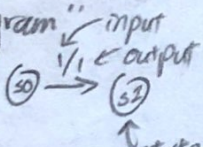
* draw Truth table.
* when see ~o ↑ means not

or = A+B
and = A*B
not = ~A
xor = (A×~B)+(~A×B)
nor = $\overline{A+B}=\bar{A}\bar{B}$
nand = $\overline{A\times B}=\bar{X}+\bar{Y}$
xnor = $\overline{A\oplus B}=(\sim A*\sim B)+(A*B)$

Ki ~ 1,000 ~ 2^{10}
Mi ~ 1,000,000 ~ 2^{20}
G ~ 1×10^{9} ~ 2^{30}
nano ~ 1×10^{-9}
pico ~ 1×10^{-12}

a[i] += a[0]
access sequence a[i], a[0], a[i]

Finite State Machine

"state transition diagram"


need: register, I/O
* Every state need a "out" arrow for every possible input

Floating Point

sign	Exponent	Significand / mantissa

Normalized floats

Value = $(-1)^{sign}\times 2^{(Exp-bias)}\times 1.Mant_2$

Denormalized floats

Value = $(-1)^{sign}\times 2^{(Exp-bias+1)}\times 0.Mant_2$

Exp	Significand	Meaning
0	anything	Denorm
1–(2^n-2)	anything	Normal
2^n-1	0	Infinity
2^n-1	Nonzero	NaN

n = bit of exp.
m = bit of mantissa.
← exp = 11...1
11...1 = 2^n-1 n: bits.

bias = $2^{n-1}-1$
How many zeros can be represented? 2
pos; real # can represent: $2^{n+m}-2^m-1$
largest finite pos. int. $(2-2^{-m})\times 2^{(exp-1)-bias}$
smallest pos? $2^{-m}\times 2^{-bias+1}$
smallest pos norm? $2^{-bias+1}$ smallest even can't rep: $2^{m+2}+2$.
smallest int can't be represented? limit by mantissa: $2^{m+1}+1$

IEE 754 single precision: n = 8
bias = 127
m = 32-9 = 23

Locality

- Sequencing instruction in a loop: both
- Subroutine prologue and epilogue: spatial
- string copy: both
- Nested if-then-else: spatial
- Two dimensional matrix multiply: both

* memory reference = data reference + instruction reference
instruction reference = 1 per line Mips
data reference = lb, sb
* conflict miss when there is another data store with + size of cache; i.e. 0 and 128 on 128 byte cache (solve by ↑ ass.)
* In MIPS, PC need to be word-aligned
** int is 4 bytes, so if stepsize = 64, every increment is 4×64 = 256 bits, will hit ½ times as in 512B as 256+256=512:
* LRU, least recently used vs random

* Counting # of cycles for pipeline
first line use 5 cycles,
each line adds 1 cycle,
when stall, add 1 cycle. (or nop.)
* clearly a hazard
lw $t0, 0($s1)
→ addiu $t0, $t0, 1

Scanned with CamScanner

Cache Coherency Ex.

After Operation	P1 Cache State	P2 Cache State	Mem. @0 Up to date?	Mem. @1 Up to date?
P1: read b1	Exclusive (1)	Invalid	Y	Y
P2: read b1	Owned (1)	Shared (1)	Y	Y
P1: write b1	Modified (1)	Invalid	Y	N
P2: write b1	Invalid	Modified (1)	Y	N
P1: read b0	Exclusive (0)	Modified (1)	Y	N
P2: read b0	Owned (0)	Shared (0)	Y	Y
P1: write b0	Modified (0)	Invalid	N	Y
P2: read b0	Owned (0)	Shared (0)	N	Y
P2: write b0	Invalid	Modified (0)	N	Y
P1: read b0	Shared (0)	Owner (0)	N	Y

Warehouse

PUE = Power Usage Effectiveness = $\frac{\text{Total Building Power}}{\text{IT Power}}$

- Always > 1
  - IT Power includes in Building Power
- IT equipment, then cooling, air flow, transformers, lightings dominant PUE.

Cut IT power in half: $\frac{(O+0.5I)}{0.5I}$ vs $\frac{O+I}{I}$

I/O

	Definition	Pro	Con
Polling	Forces hardware to wait on ready bit, check it regularly	• Easy to write • mouse / keyboard • low overhead. • deterministic	• Infeasible on fast transfer rate that rarely ready
Interrupts	Hardware fires "exception" when it becomes ready	• Necessary for fast devices that are rarely ready • Hard drive, network cards	• non deterministic • overhead, slow switch to polling

Disks

Disk access time = seek time + rotation time + transfer time + controller overhead

Seek time = time to move head to correct track = (# of tracks)/3

Rotation time = time for disk to rotate to correct sector, average is 1/2.

Transfer time, time to get data on/off disk

Ex

- 15000 cylinders, 1 ms to cross 1000 c.
- 7200 RPM
- Want 1 MB, transfer rate 200 MB/s
- 1 ms overhead.

seek = $\left(\frac{15000}{3}\right) \cdot \frac{1\text{ ms}}{1000\text{ c}} = 5\text{ ms}$

rotation = $\frac{1}{7200\text{ RPM} \cdot \frac{1\text{ min}}{60\text{ s}}} = 4.2\text{ ms}$

Transfer = size / transfer rate = $\frac{.1\text{ MB}}{200\text{ MB/s}} = 5\text{ ms}$

Average Page access time (APAT)

Same above, standard page size = 4 kiB

$APAT = T_M + T_T + P_T * (T_M + P_F * (T_D * (1 + P_D)))$

P_T = prob of TLB miss

P_F = prob of page fault when TLB miss

P_D = prob page is dirty when replaced

T_T = time to access TLB

T_M = time to access Mem

T_D = time to transfer a page to/from disk

size when calculate transfer is page size

Network

Each layer add head/tail info to package.

TCP/IP

- TCP guarantees delivery, IP makes best effort
- Application
- Transport
- Internet
- Link

1 byte = 8 bits

Effective bandwith = $\frac{(\text{chunk size} - \text{header/tail size})}{\text{chuck\_size}} \times$ original bandwidth (Mbit/s)

Dependability Measures

- Mean Time to Failure (MTTF)
- Mean Time to Repair (MTTR)
- Mean Time Between Failures (MTBF) = MTTF + MTTR
- Availability = $\frac{MTTF}{MTTF + MTTR}$
- 3 nines = 99.9%
- Annualized failure rate = Avg. # of failures per year

Hamming ECC

Bit	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Data	P1	P2	D1	P4	D2	D3	D4	P8	D5	D6	D7	D8	D9	D10	D11
P1	X		X		X		X		X		X		X		X
P2		X	X			X	X			X	X			X	X
P4				X	X	X	X					X	X	X	X
P8								X	X	X	X	X	X	X	X

* concurrently independent write: 5 not 4 (RAID)

* DMA (direct mem access) most useful when transfer large data block.

* save PC and user registers when interrupt

Config.		Pro	Con
RAID 0	Data disk w/o check info	• No overhead, fast read/write	• reliability
RAID 1	Mirror disk: extra copy	• Fast read/write, recovery	• 100% overhead.
RAID 2	Hamming ECC: one check disk per parity group	• smaller overhead.	• Redundant check disk
RAID 3	Single check disk for error correction. Transfer units are spread over all disks	• Smallest check info, overhead	• Read all data disks for small accesses to detect error.
RAID 4	Transfer unit = a sector within a single disk. Errors are detected w/in single transfer unit; can handle independent read/write per disk.	• Higher throughput of small reads	• still slow small writes (A single check disk is bottleneck)
RAID 5	Check info. is distributed across all disk in group	• Higher throughput in small writes	

Scanned with CamScanner

Virtual Memory

Ki = 2^{10} Mi = 2^{20} Gi = 2^{30}

- Page: block of mem.
- Maps Virtual Page # (VPN) to Physical Page # (PPN)
- Page fault: miss in page table (PT) means page not stored in memory.

VA: [VPN | P.O.]
PA: [PPN | P.O.]

PT

V	D	R/W	PPN

bits of:
P.O. (page offset) = $\log_2$(page size)
V.A. (Virtual Address) = $\log_2$(VA space)
P.A. (Physical Address) = $\log_2$(PA space)
VPN = VA bit − P.O. bit

Page Table entry bits:
1 Valid + 1 Read + 1 Write + 1 Dirty + PPN bits

Page table size:
2^{VPN}

How big is page table: PTE bits × PT size
How many PTE can be valid: 2^{PPN}

TLB (Cache for Page table)

Fully associative, LRU-Replacement.
Miss in TLB → go to PT.

VPN	V	R/W	D	PPN	LRU

TLB entry bits:
VPN + V + R/W + D + PPN + LRU
LRU bits = $\log_2$(# of TLB entry)
TLB reach = # of TLB entry × page size.
Page Table Base Register points to PT of processor currently running
TLB invalidated when swapping processes
Minimum # of bits for page table base register = $\log_2$(physical memory)

Cache Coherency: MOESI

1. Shared: Up-to-date data, other caches may have copy
2. Modified: up-to-date data, changed (dirty), no other cache has a copy, OK to write, memory out-of-date.
3. Exclusive: up-to-date data, no other cache has a copy, OK to write, memory up-to-date.
   - Avoids writing to memory if block replaced
   - Supplies data on read instead of going to mem.
4. Owner: up-to-date data, other caches may have a copy (they must be in Shared)
   - only cache that supplies data on read instead of going to memory
5. Invalid: not in cache

State	Cache Up to date?	Mem. Up to date	Other have copy?	Can resp. read?	Can write to other's without changing state?
Modified	Y	N	N	Yes, Required	Y
Owned	Y	Maybe	Maybe	Yes, Optional	N
Exclusive	Y	Y	N	Yes, Optional	N
Shared	Y	Maybe	Maybe	N	N
Invalid	N	Maybe	Maybe	N	N

MapReduce

Map (start_key, start_val) → (key, val) or list (key, val)
* slice data for workers.
* maps one key, value pair to 0 or more intermediate value pairs

Combine: list (key, val) → (key, list (val))
* combines values with same key

Reduce: (key, list (val)) → (final_key, final_val)
* Takes a key and a list of values that had key as input
* combines separate values to a final solution.

emit (..., ...)

SIMD

Tail case!, first go from i=0, i< k/4*4; i+=1, then from k/4*4 to k.
mm_add_epi32, for si, remember to cast (__m128i*) ...

Data Race

* 2 memory access same location, done with lock

Open MP

#pragma omp parallel: code inside block will run on each thread
- omp_get_num_threads(); - omp_get_thread_num()

#pragma omp parallel for
#pragma omp critical, private (var)

Amdal's Law

$$\frac{1}{(1-F)+\frac{F}{S}}$$

F: fraction sped up
S: amt speed up

** cpu does not need separate inst. for I/O
* except. in early pipeline override except in later stage.

* # of page fault = # array × $\frac{\text{size of array} \times \text{size (int or char)}}{\text{page size}}$

Scanned with CamScanner

MIPS: Calling Conventions

- Callee's responsibility: (put in prologue)
  - $s*, $sp
- Caller's responsibility: (save before call func.)
  - $v*, $ra, $a*, $t*

Number

unsigned: $[0, >^n-1]$

2's complement: $[-2^{(n-1)}, 2^{(n-1)}-1]$

negate: flip all, plus 1.

Hex: 1 to F

shift left = mult. by 2,

shift right = floor divided by 2.

C

sizeof(char) == 1

&i ⇒ get address of i

*p ⇒ dereference p

arr[i] ⇔ *(arr+i)

```
struct Student *s = malloc(sizeof(struct Student))
s -> name = "Oski" ⇔ (*s).name = "Oski",
s -> year = 1000 ⇔ (*s).year = 1000;
main() {
  char *c = "hi"; char d[] = "hi"}
```

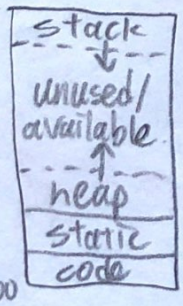
c is in stack, c[0] is in static, d[0] and d in stack

char[len+1], last one for '\0'

Big endian: most significant bit is stored in lower memory address

Little endian: most significant bit is stored in higher memory address

FFF



Code segment

- Instructions stored here
- Loaded when program starts
- Read only
- Doesn't grow.

Static segment

- "Global" variables
- String literals*
- Global-frame variables
- Loaded when program starts
- Two parts:
  - Read-only: string literals
  - Read-write: Variables and pointer containers.

Heap segment

- dynamically allocated.
- persists across fxn

Stack segment

- grow in "frames"
  - new frame per fxn call
- memory wiped when fxn return.

*don't have to save $ra if not jumping to another func.

TAL replaces psuedo-instructions w/ longhand form in MAL.

```
lw $t0 4($a0) # Load value at address in $a0+4 to $t0
              $t0 = $a0[1].
```

J-type

* opcode (6) | jump address (26)

opcode = 2 or 3 (j or jal)

* Jump to address {(PC+4)[31:28], jump address, 0b00}

To find jump addr:
- remove top 4 bits and last two bits

I-type

* opcode (6) | rs (5) | rt (5) | immediate (16)

opcode ≠ 2, 3, 0.

R-type

* opcode (6) | rs (5) | rt (5) | rd (5) | shamt (5) | funct (6)

opcode = 0.

shamt bit = $\log_2$(x-bit system)

if write typedef then don't have to write "struct" when calling.

struct node* ...

```
lw $t0, 4($s0)
  imm = 4
```

Scanned with CamScanner

addi reports overflow
addiu does not
both sign-extend immediate.

lh sign-extends, lhu does not.

sltiu performs an unsigned comparison.
slti performs a signed comparison
both sign-extend imm.

"unsign"
1. Do not report overflow.
2. zero extend value (lhu)
3. interpret data as unsigned # (sltiu)

pse-ind.

li $t0, 0xCAFEFOOD
⇒ lui $at, 0xCAFE
ori $t0, $at, 0xFOOD

beq .... stuff
.....
.....
Stuff:
beq hex
.....

PC+4
1.
2.

imm
0x2.

* string literal takes up one more byte for "\0"
* in MIPs, each instruction is 4 bytes

Compiler: C → MIPS
Assembler: MIPS → Bits
Linker: Multiple code images → single binary
Loader: Prep program for exec.
Calculates absolute address → linker.

Assemblers do 2 passes to resolve addresses, handing internal forward references.

object file format
- header: size & position of other pieces of o. file
- text segment: machine code
- data segment: binary representation of static data
- relocation info: identifys lines of codes that need fix up.
- symbol table: list of file's labels and data for reference

IJ-instr: srl $v0 $a0 26
jr $ra
v0 == 1 if $a0 is a pointer to I or J instruction

jal label to call a function
jr $ra to return a function

prologue: store (sw) $s*, $ra, $sp - # (addiu)
epilogue: restore (lw) $s*, $ra, $sp + #, output to $v*

slt $d $s $t # d == 1 if s < t

jal saves $ra to PC+4

sp ≡ stack pointer
fp ≡ frame pointer

and & 1 & n == n
or | 0 | n == n
xor ^
not ~
<< left shift
>> right shift

bytes = 8 bits
nibbles = 4 bits

--p, minus first
%d, print signed, %u prints unsign.

	Symbol T.	Relocation T.
What phrase is it written to?	Assembler	Assembler
What phase is it read from?	Linker	Linker
Why would you save a label into this table?	Labels that can be used by other files	Labels needed by this file.

Scanned with CamScanner