SVM → Classification

$y_i = \begin{cases} 1, & \text{if data is } \\ -1, & \text{if data is } \times \end{cases}$

$d = \frac{1}{2}$ distance between two // lines $= \frac{2}{\sqrt{\Sigma_j (a_j)^2}}$

maximise d so minimise $\Sigma_j (a_j)^2$ ← margin denominator over $a_0,\dots,a_n$

subject to $(a_1 x_{i1} + a_2 x_{i2} + \dots + a_n x_{in} + a_0) y_i \geq 1$

$a_1x_1 + a_2x_2 + a_0 = -1$

$a_1x_1 + a_2x_2 + a_0 = 0$

$a_1x_1 + a_2x_2 + a_0 = 1$

Correct side of the line: $\left(\sum_{j=1}^{n} a_j x_{ij} + a_0\right) y_i - 1 \geq 0 \Rightarrow 0 \geq 1 - (z) y_i$

Error for data point i: $\max\left\{0, 1 - \left(\sum_{j=1}^{n} a_j x_{ij} + a_0\right) y_i\right\}$

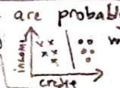
Wrong side of the line: $\left(\sum_{j=1}^{n} a_j x_{ij} + a_0\right) y_i - 1 \leq 0 \Rightarrow 0 \leq 1 - (z) y_i$

Soft Classification (trade off)

$\min_{a_0,\dots,a_n} \sum_{i=1}^{m} \max\left\{0, 1 - \left(\sum_{j=1}^{n} a_j x_{ij} + a_0\right) y_i\right\} + \lambda \sum_{j=1}^{n} (a_j)^2$

error in classifying data points — margin denominator error

near-zero coefficients are probably not relevant for classification eg. we see that income is not helpful for classifying



Scaling between 0 to 1: $x_{ij}^{scaled} = \frac{x_{ij} - x_{min j}}{x_{max j} - x_{min j}}$

Scaling between a and b: $x_{ij}^{scaled[a,b]} = \left(x_{ij}^{scaled[0,1]}\right)(b-a) + a$

Standardisation → scale to Normal Distribution: factor j has mean $\mu_j = \frac{\sum_{i=1}^{n} x_{ij}}{n}$ and sd σ_j

$x_{ij}^{standardised} = \frac{x_{ij} - \mu_j}{\sigma_j}$

Distance Metrics

Distance between two points $\tilde{x}$ and $\tilde{y}$: $\sqrt{\sum_{i=1}^{n} |x_i - y_i|^2}$ = straight line distance

Attributes can be weighted by importance: $\sqrt{\sum_{i=1}^{n} w_i |x_i - y_i|^2}$ → $w_i = 0$ for unimportant attributes

KNN Classification: New point's class is the most common among the k neighbours. Choose good value of k using validation

Model Validation: Real effects same in all data sets. Random effects different in all data sets.

Observed performance = real quality + random effects

Build Model using Training Set (Fit) → Choose Best Model using Validation Set → Estimate quality of chosen model using Test Set

Use 50-70% of data for training

k-Fold Cross Validation

Train	Validation
①②③	④
①②④	③
①③④	②
②③④	①

Take average validation score. Once we select the model in the validation step, we train the model again using all the data. Do not average the coefficients.

Classification → supervised. We have observed the response for each data point. Use that info in addition to attributes to figure out how to deal w new data points.

Clustering → unsupervised. We have not observed the response for each data point & only using the attribute information.

Outlier types → point outliers, contextual outlier, collective outlier

Bad data → sensor fails, contaminated experiment, wrong input

Bad data → omit or impute. | Real/correct Data → outliers are expected in large data sets eg normal distribution

Clustering

Euclidean	p-norm distance Minkowski
$\sqrt{(x_1-y_1)^2 + (x_2-y_2)^2}$	$\sqrt[p]{\lvert x_1-y_1\rvert^p + \lvert x_2-y_2\rvert^p}$
Manhattan $\lvert x_1-y_1\rvert + \lvert x_2-y_2\rvert$	or $\sqrt[p]{\sum_{i=1}^{n} \lvert x_i-y_i\rvert^p}$

∞-norm: $\sqrt[\infty]{\sum_i |x_i - y_i|^\infty} = \sqrt[\infty]{\max_i |x_i-y_i|^\infty} = \max_i |x_i - y_i|$

K-means (unsupervised): $y_{ik} = \begin{cases} 1, & \text{if data } i \text{ is in cluster } k \\ 0, & \text{if not} \end{cases}$

x_{ij} = jth coordinate of data i; z_{jk} = jth coordinate of cluster center k

Minimize $\sum_i \sum_k y_{ik} \sqrt{\sum_j (x_{ij} - z_{jk})^2}$ over y, z subject to $\Sigma_k y_{ik} = 1, \forall i$ — each point only in 1 cluster

⓪ Pick k clusters within data range

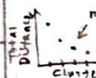
① Assign each data point to nearest cluster center

② Recalculate cluster center (centroids) ← repeat until no change

→ Expectation Maximisation → cluster centers are averages/exp; maximisation is assigning points to clusters

→ Heuristic → fast but does not guarantee best answer.

Tuning K Means → Run multiple times w different centers until best solution is found. → try different K-values



marginal benefit of adding another cluster or clusters starts to be small. Additional clusters do not (Elbow Diagram) lead to significant improvement.

Change Detection - Cumulative Sum (CUSUM)

$S_t = \max\{0, S_{t-1} + (x_t - \mu - C)\}$ check is $S_t \geq T$ } Increase

C and T are model parameters that you choose

$S_t = \max\{0, S_{t-1} + (\mu - x_t - C)\}$ check is $S_t \geq T$ } Decrease

Higher T → Detect change slower + less likely to falsely detect change

Lower T → Detect change faster + highly likely to falsely detect change

Descriptive: Understand how the values of attributes relate to the value of a response

Predictive: Predicting the value of a response, given the values of attributes

Prescriptive: Determining the best course of action

Remove outliers → Impute Missing Data Values → Scale Data → Fit Lasso Model on all Variables → Fit Linear Regression using variables chosen → Validation → test

Analytics Models

Basic Machine Learning → Classification: SVM, KNN

Clustering: K-means

Time Series → CUSUM, Exponential Smoothing, ARIMA, GARCH

Regression → Linear, Logistic, Advanced

Tree Based → CART, Random Forest

Design of Experiments → A/B Testing, Factorial Design, Multi Armed Bandit Models

Probability Based → Distribution Fitting, Queuing, Markov Chains, Simulation

Optimization Model → prescriptive analytics, statistical modelling

Advanced → non-parametric models, graph analysis, competitive models, deep learning

Time Series — x_t: the observed response

Two ways to answer:
$S_t = x_t$ ⇒ the observed blood pressure is a real indicator of the baseline
$S_t = S_{t-1}$ ⇒ today's baseline is the same as yesterday's

Exponential Smoothing $S_t = \alpha x_t + (1-\alpha) S_{t-1}$, $0 < \alpha < 1$

- $\alpha \to 0$: a lot of randomness in the system. Lots of fluctuations mostly due to randomness, thus yesterday's baseline a good indicator of today's baseline
- $\alpha \to 1$: not much randomness in the system. If we observe a fluctuation it probably means today's baseline close to observed

α large = trust x_t (less randomness), α small = trust S_{t-1} (more randomness)

Single Exponential Initial Condition $S_1 = x_1$

Trends (Double Exponential Smoothing) T_t: trend at time period t

$S_t = \alpha x_t + (1-\alpha)(S_{t-1} + T_{t-1})$

$T_t = \beta(S_t - S_{t-1}) + (1-\beta) T_{t-1}$

Initial Condition: $S_1 = x_1$, $T_1 = 0$

Cyclic Patterns / Seasonalities (Triple Exponential Smoothing)

$S_t = \alpha x_t / C_{t-L} + (1-\alpha)(S_{t-1} + T_{t-1})$

L: length of a cycle. C_t: multiplicative seasonality factor for time t

$C_t = \gamma(x_t / S_t) + (1-\gamma) C_{t-L}$, Initial Conditions: $C_1, \dots, C_L = 1$ — First L values of C set to 1

Trend: $S_t - S_{t-1}$ denotes difference between t & $t-1$ baseline

Seasonality / Cyclic Pattern: x_t / S_t denotes ratio between observed and expected baseline

$S_t = \alpha x_t + (1-\alpha) S_{t-1} = \alpha x_t + (1-\alpha)[\alpha x_{t-1} + (1-\alpha) S_{t-2}]$
$= \alpha x_t + (1-\alpha)\alpha x_{t-1} + (1-\alpha)^2 \alpha x_{t-2} + (1-\alpha)^3 \alpha x_{t-3} + \dots$

Every past observation contributes to current baseline estimate, newer observations weighted more.

Forecasting Single: $S_{t+1} = \alpha x_{t+1} + (1-\alpha) S_t$, x_{t+1} unknown

Best guess: $x_{t+1} = S_t$, $F_{t+1} = \alpha S_t + (1-\alpha) S_t = S_t$

$F_{t+k} = S_t$, $k = 1, 2, \dots$

Double: $S_{t+1} = \alpha x_{t+1} + (1-\alpha)(S_t + T_t)$

Best estimate of next baseline = most current baseline estimate

Best estimate of trend = most current trend estimate

Forecast $F_{t+1} = S_t + T_t$, $F_{t+k} = S_t + kT_t$, $k = 1, 2, \dots$

Triple: $S_t = \alpha x_t / C_{t-L} + (1-\alpha)(S_{t-1} + T_{t-1})$

Best estimate for next time period seasonal factor $C_{t+1} = C_{(t+1)-L}$ Forecast: $F_{t+1} = (S_t + T_t) C_{(t+1)-L}$

$F_{t+k} = (S_t + kT_t) C_{(t+1)-L+(k-1)}$, $k = 1, 2, \dots$

Expo smoothing bases its forecast primarily on most-recent data points

ARIMA (p, d, q) → eg. $D_{(2)t} = (x_t - x_{t-1}) - (x_{t-1} - x_{t-2})$

d = order of differences eg. $D_{(3)t} = [(x_t - x_{t-1}) - (x_{t-1} - x_{t-2})] - [(x_{t-1} - x_{t-2}) - (x_{t-2} - x_{t-3})]$

p = time period for autoregression

Order-p means S_t is a function of $\{x_t, x_{t-1}, \dots, x_{t-(p-1)}\}$

ARIMA uses autoregression on differences, using p time periods of previous observations to predict dth order differences

q = order of moving average. errors $\varepsilon_t = (\hat{x}_t - x_t)$

order-q MA goes back q time periods ie $\varepsilon_{t-1}, \varepsilon_{t-2}, \dots, \varepsilon_{t-q}$

ARIMA: $D_{(d)t} = \mu + \sum_{i=1}^{p} \alpha_i D_{(d)t-i} - \sum_{i=1}^{q} \theta_i (\hat{x}_{t-i} - x_{t-i})$

ARIMA forecasts values | GARCH → Estimate or forecast variance

Variance is a proxy for volatility or risk

GARCH tracks variance/squared errors not linear errors & tracks raw variances not difference of variance

GARCH is used to analyse time series data

Autoregressive means using previous response data to predict current data

Module 8: Basic Regression (linear relationship between predictors and response)

Simple Regression: $y = a_0 + a_1 x_1$

m predictors: $y = a_0 + a_1 x_1 + a_2 x_2 + \dots + a_m x_m$

y_i = data point i, $\hat{y}_i$ = model prediction for data point i

data point i prediction error $y_i - \hat{y}_i = y_i - (a_0 + a_1 x_{i1})$

Sum of Squared Errors $\sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} (y_i - (a_0 + a_1 x_{i1}))^2$

8.2 Maximum Likelihood — Error $\sim N(0, \sigma^2)$, i.i.d.

Observations: $z_1, \dots, z_n$. Model Estimates: $y_1, \dots, y_n$

PDF for observing z_i if true value is y_i: $\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(z_i - y_i)^2}{2\sigma^2}}$; Joint Density over n terms:

$\prod_{i=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(z_i - y_i)^2}{2\sigma^2}} = \left(\frac{1}{\sigma\sqrt{2\pi}}\right)^n e^{-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(z_i - y_i)^2}$

Maximise PDF ⇒ minimize $\sum_{i=1}^{n} (z_i - y_i)^2$

AIC — L^*: max likelihood value, k: no. of parameters estimated

AIC = $2k - 2\ln(L^*)$. Penalty term 2k helps avoid overfitting

prefer models with lower AIC: Model 1 AIC = 75, Model 2 AIC = 80

Relative likelihood: $e^{\frac{(AIC_1 - AIC_2)}{2}} = e^{\left(\frac{75-80}{2}\right)} \approx 8.2\%$

Model 2 is 8.2% as likely as Model 1 to be better, so Model 1 is probably better

BIC = $k\ln(n) - 2\ln(L^*)$. BIC penalty term > AIC Penalty Term

BIC penalizes overfitting hence encourages models w fewer parameters than AIC does. Use BIC when data points > parameters

$\lvert BIC_1 - BIC_2 \rvert > 10$	smaller model very likely better	BIC & AIC favour simpler models. Simple models less likely to be overfit to random effects
$6 < \lvert BIC_1 - BIC_2 \rvert < 10$	" " likely "	
$2 < \lvert BIC_1 - BIC_2 \rvert < 6$	" " somewhat likely "	
$0 < \lvert BIC_1 - BIC_2 \rvert < 2$	" " slightly likely "	

P-values estimate probability that the coefficient might be zero through hypothesis testing. If p-value > 0.05, remove corresponding attribute

R^2 value = coefficient of determination. The estimate of how much variability your model accounts for.

Convex Quadratic: minimize $\Sigma_i c_i x_i^2$ subject to $\Sigma_i a_{ij} x_i \geq b_j, \forall j$ and all $x_i \geq 0$

Linear Program: minimize $\Sigma_i (\log c_i) x_i$ subject to $\Sigma_i a_{ij} x_i \geq b_j$ for all j and all $x_i \geq 0$

Integer Program: maximize $\Sigma_i c_i x_i$ subject to $\Sigma_i a_{ij} x_i \geq b_j$ for all j and all $x_i \in \{0, 1\}$

Convex Program: minimize $\Sigma_i c_i |x_i - 6|$ subj to $\Sigma_i a_{ij} x_i \geq b_j\ \forall j$ and all $x_i \geq 0$

Linear Program: minimize $\Sigma_i c_i x_i$ subj to $\Sigma_i a_{ij} x_i \geq b_j\ \forall j$, all $x_i \geq 0$

General non-convex program: Maximize $\Sigma_i c_i x_i$ subject to $\Sigma_i \Sigma_k a_{ikj} x_i x_k \leq b_j\ \forall j$, all $x_i \geq 0$

Module 9 · Advanced Data Preparation. Box-Cox Transformation

Heteroscedasticity = unequal variance. A Box-Cox transformation is a logarithmic transformation. Stretches out smaller range to enlarge variability & shrinks larger range to reduce its variability. $t(y) = (y^{\lambda}-1)/\lambda$. $t(y)$ can be close to normal

Detrending: remove trend effects from time series data

Detrend data before using time series data in regression model

9.3 Principal Component Analysis (PCA). PCA transforms data ① Removes correlations within the data ② ranks coordinates by importance. | Concentrate on first n principal components: reduces effect of randomness & likely to have higher signal to noise ratio

9.4 PCA Eigenvectors & Eigenvalues

A is square matrix. λ is eigenvalue of A s.t. $\det(A-\lambda I)=0$

Given λ, solve $A\tilde{v} = \lambda\tilde{v}$ to find corresponding eigenvector $\tilde{v}$

X = initial matrix of data; x_{ij} is the jth factor of data point i

Scale s.t. $\forall j$, $\frac{1}{m}\sum_i x_{ij} = \mu_j = 0$. Then find all eigenvalues of X^TX (a square matrix) s.t. $\det(X^TX - \lambda I) = 0$

Given each λ_j value, $\tilde{v}_j$ is the corresponding eigenvector found when we solve $X^TX\tilde{v}_j = \lambda_j\tilde{v}_j$, $\forall \tilde{v}_j \in \mathbb{R}^n$, n = no. of factors/columns in X

V = matrix of eigenvectors (sorted by eigenvalue)

$V = [\tilde{v}_1\ \tilde{v}_2 \ldots]$ where $\tilde{v}_j$ is jth eigenvector of X^TX

Principal Components: note each $\tilde{v}_j$ has order $n\times 1$ where n = no. of factor columns in X.

$X\tilde{v}_1$ = 1st component. $X\tilde{v}_1$ order is $m\times 1$

$X\tilde{v}_2$ = 2nd component. $X\tilde{v}_2$ order is $m\times 1$

kth new factor value for ith data point: $t_{ik} = \sum_{j=1}^{m} x_{ij} v_{jk}$

PCA finds new L factors $\{t_{ik}\}$ where $L < n$

regression finds coefficients $b_0, b_1, \ldots, b_L$

$y_i = b_0 + \sum_{k=1}^{L} b_k t_{ik} = b_0 + \sum_{k=1}^{L} b_k \left[\sum_{j=1}^{m} x_{ij} v_{jk}\right]$

$= b_0 + \sum_{j=1}^{m} x_{ij}\left[\sum_{k=1}^{L} b_k v_{jk}\right] = b_0 + \sum_{j=1}^{m} x_{ij}\,[a_j]$

Implied regression coefficient for factor x_j: $a_j = \sum_{k=1}^{L} b_k v_{jk}$

v_{jk} is the jth entry of eigenvector k, $\tilde{v}_k \in \mathbb{R}^n$, n = no. of columns. There are L principal components.

Each original attribute's implied regression coefficient is equal to a linear combination of the principal components' regression coefficients

Module 10 · Advanced Regression. CART: Regression in a tree

$y = a_0$ where $a_0 = \frac{\sum_{i \text{ in node}} y_i}{\#\text{ data points in node}}$ = avg. response in node

Quick derivation of best fit: $\min_{a_0} \sum_i (y_i - a_0)^2$

Set derivative = 0, i.e. $2\sum_i(y_i - a_0) = 0 \Rightarrow \sum_i y_i = k a_0$

$a_0 = \frac{\sum_i y_i}{k}$ | Each leaf's individual model is tailored to the subset of data points that follow all of the branches leading to the leaf | How to reject a potential branch ① Low improvement benefit ② Too few data points (each leaf contains > 5% of original data)

Fitting to small subsets of data will cause overfitting

Random Forests ① Bootstrap n points from n original data points ② Branching: Randomly choose a small number of factors & choose best factor within that set to branch on ③ Don't need to prune | Regression Trees → take average prediction

Classification Trees → use mode/most common prediction

Averages between trees neutralizes overfitting. Random forests are black boxes | Models that are less explainable might give us better predictions

Logistic Regression. Standard linear regression $y = a_0 + a_1x_1 + \ldots + a_jx_j$

log regression: p = probability of new event you want to observe

$\log\frac{p}{1-p} = a_0 + a_1x_1 + a_2x_2 + \ldots + a_jx_j$ | if $\alpha = -\infty \Rightarrow p = 0$, if $\alpha = +\infty \Rightarrow p = 1$

$p = \frac{1}{1+e^{-(a_0 + a_1x_1 + a_2x_2 + \ldots + a_jx_j)}}$ (α)

Thresholding: Answer "yes" if probability p is at least some number, otherwise "no". Eg if $p \geq 0.3$, then give loan

AUC Curve: Plots sensitivity against 1 − specificity

AUC = probability that the model estimates a random "yes" point higher than a random "no" point

If AUC = 0.5 means just guessing

True Classification \ Model Classification	Yes	No
Yes	TP	FN
No	FP	TN

T or F refer to whether the model is correct or not, and P or N refer to whether the model says the point is in the category

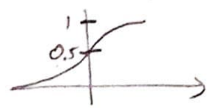
Sometimes the cost of a false positive is so high that it's worth accepting more false negatives or vice versa

Logistic Regression Variables $a_0, a_1, \ldots, a_m$

Constraints: None. Objective Function

· Maximize $\prod_{i: y_i = 1} p(x_i) \prod_{i: y_i = 0} (1 - p(x_i))$

where $p(x_i) = \frac{1}{1+e^{-(a_0 + \sum_{j=1}^{m} a_j x_{ij})}}$



Time Series Models

ARIMA Variables μ, φ_i, θ_i

Constraints: none

Objective Function: minimize $\sum_{t=1}^{n}(x_t - \hat{x}_t)^2$ where $D_{(d)t} = \mu + \sum_{i=1}^{p} \varphi_i D_{(d)t-i} - \sum_{i=1}^{q} \theta_i(\hat{x}_{t-i} - x_{t-i})$

Exponential Smoothing Variables α, β, γ

Constraints:

$0 \leq \alpha \leq 1$

$0 \leq \beta \leq 1$

$0 \leq \gamma \leq 1$

Objective: Minimize $\sum_{t=1}^{n}(x_t - \hat{x}_t)^2$

GARCH Variables $\omega, \beta_i, \gamma_i$

Constraints: None. Objective: minimize $\sum_{t=1}^{n}(\sigma_t^2 - \hat{\sigma}_t^2)^2$ where $\hat{\sigma}_t^2 = \omega + \sum_{i=1}^{p}\beta_i\sigma_{t-i}^2 - \sum_{i=1}^{q}\gamma_i\epsilon_{t-i}^2$

Module 11 · Variable Selection. Overfitting: When # of factors is close to or larger than # of data points, the model might fit too closely to random effects. Simplicity: Less data is required. Less chance of insignificant factors. Easier to interpret

Optimization and Models

Linear Regression Model Variables $a_0, a_1, \ldots, a_m$

Constraints: None. Objective Function:

Minimize $\sum_{i=1}^{n}(y_i - (a_0 + \sum_{j=1}^{m} a_j x_{ij}))^2$

Find best fit for $a_0, a_1, \ldots, a_m$

Statistical model for regression

$y_i = a_0 + \sum_{j=1}^{m} a_j x_{ij}$

a_j are constant coefficients; x_{ij} are variables

Optimization model for regression

minimize $\sum_{i=1}^{n}(y_i - (a_0 + \sum_{j=1}^{m} a_j x_{ij}))^2$

a_j are variables; x_{ij} are constant

Linear Regression: no constraints

Lasso Regression Constraint: $\sum_{j=1}^{m}|a_j| \leq T$

Ridge Regression Constraint: $\sum_{j=1}^{m}(a_j)^2 \leq T$

Elastic Net Constraint:

$\lambda\sum_{j=1}^{m}|a_j| + (1-\lambda)\sum_{j=1}^{m}(a_j)^2 \leq T$

SVM

Hard Classification Variables: $a_0, a_1, \ldots, a_m$

Constraints $(a_0 + \sum_{j=1}^{m} a_j x_{ij})y_i \geq 1$, $\forall i$

Objective Function: Maximise $\sum_{j=1}^{m}(a_j)^2$

Soft Classification Variables $a_0, a_1, \ldots, a_m$

Constraints: none. Objective function:

minimize $\sum_{i=1}^{n} \max\{0, 1 - (a_0 + \sum_{j=1}^{m} a_j x_{ij})y_i\} + \lambda\sum_{j=1}^{m}(a_j)^2$

K-means Clustering Variables z_{jk} → coordinate j of cluster center k

y_{ik} (1 if point i in cluster k, 0 if not)

Constraints: $\sum_k y_{ik} = 1$, $\forall$ data point i → each point assigned to a cluster.

Objective function: minimize $\sum_i \sum_k y_{ik} \sqrt[p]{\sum_j (x_{ij} - z_{jk})^p}$

ie minimize total distance from data points to their cluster centers.

Linear Program: $f(x)$ is a linear function

minimize or Maximize $C + \sum_{i=1}^{n} c_i x_i$

Constraint set X is defined by linear equations & inequalities $\sum_{i=1}^{n} a_{ij} x_i \leq b_j$ or $\geq$ or $=$ $\forall j$

Easy to solve even for large instances

Convex Quadratic Program

· $f(x)$ is a convex quadratic function: minimize $f(x)$ or maximize $-f(x)$

· Constraint set X is defined by linear equations & inequalities

Easy / fast to solve, but not as quickly as linear Programs

Convex Optimization Program

Obj function $f(x)$ is · Concave (if maximizing) · Convex (if minimizing)

· Constraint set X is a convex set

· Easy to solve, but solve can take longer

Integer Program · Linear Program PLUS some (or all) variables restricted to take only integer values · Variables could be binary · either 0 or 1. More difficult to solve even with good software packages

General Non-Convex Program

· Optimization problem is not convex ↳ hard to find optimal solutions

Optimization Algorithms: Find a good direction to move from the current solution, and determine how far to go in that direction

Survival Model: Cox Proportional Hazard Model

Given (for each data point): Predictor Variables x_1, x_n, Event time y

$h(t) = h_0(t)e^{(\beta_1x_1 + \ldots + \beta_nx_n)} = h_0(t)e^{\beta_1x_1}e^{\beta_2x_2}\ldots e^{\beta_nx_n}$

β_j = coefficient for each predictor variable

$e^{\beta_jx_j}$ = prob. multiplier due to factor j

Baseline (If all predictor variables are zero), probability of event happening at time t

Prob of event happening at time t, given specific values of predictors (& coefficients)

$\beta_j = 0$: x_j has no effect. $\beta_j > 0$: +ve x_j, higher prob. $\beta_j < 0$: positive x_j, lower probability

$\hat{y} = w_0 x_0 + w_1 x_1 + \dots + w_n x_n + b$

Variable Selection

$$\sum_{i=1}^{M} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{M} \left(y_i - \sum_{j=0}^{P} w_j x_{ij}\right)^2$$

Observed, Estimate — Cost function

Ridge (global)

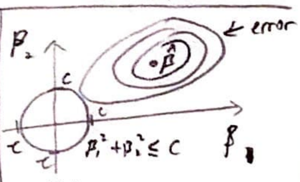
introduce small amount of bias to reduce variability.

$$\sum_{i=1}^{M} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{M} \left(y_i - \sum_{j=0}^{P} w_j x_{ij}\right)^2 + \lambda \sum_{j=0}^{P} w_j^2$$

penalty

$\sum_{j=0}^{P} w_j^2 \le T_{Ridge}$

$\lambda \uparrow \Rightarrow$ slope gets close to zero (ie horizontal)



Ridge

Minimize

When sample sizes are relatively small, then ridge regression can improve predictions made from new data (ie reduce variance) by making the predictions less sensitive to training data.

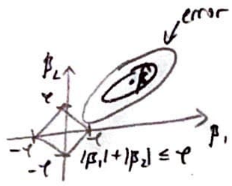
Coefficients shrink toward zero to reduce variance in estimate ie regularization

Lasso (global)

$$\sum_{i=1}^{M} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{M} \left(y_i - \sum_{j=0}^{P} w_j x_{ij}\right)^2 + \lambda \sum_{j=0}^{P} |w_j|$$

Cost function, Penalty

$\sum_{j=0}^{P} |w_j| \le T_{lasso}$



Remove

Lasso can exclude variables. (variable selection)

Data needs to be scaled for Lasso ↳ coefficients can go to zero.

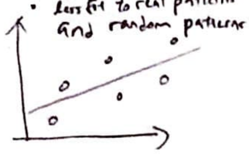
Choice of T tradeoff between no. of variables & quality of model

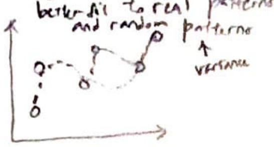
Elastic Net (global)

$$\sum_{i=1}^{M} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{M} \left(y_i - \sum_{j=0}^{P} w_j x_{ij}\right)^2 + \lambda \sum_{j=0}^{P} |w_j| + (1-\lambda) \sum_{j=0}^{P} w_j^2$$

. Scale data

If two factors are highly correlated, Lasso will pick just one & leave the other out. Ridge will include both, & because both coefficients shrink, they get closer to each other.





Underfit (allow misclassifications)	Overfit
Average job fitting training set	Good job fitting training set
Ok job fitting testing set	Bad job fitting testing set
Low variability/variance	High variability/variance
SSE similar for varying datasets	SSE very different for varying datasets

to find sweet spot between bias & variance use regularization, boosting & bagging

Forward Selection, Backward Elimination & Stepwise Regression (greedy)

At every step, the stepwise regression fits a different model

Design of Experiments.

Multi Armed Bandits → As we get more sure of the best answer, we're more likely to use it

Exploration - Exploitation Trade off.

A/B Testing.

Ad	Clicks	Users	%
A	46	1003	5%
B	97	997	10%

Factorial Design (combination) → ANOVA to determine factor importance

Fractional Factorial Design

→ Test each choice the same # of times

→ Test each pair of choices the same # of times

Missing Data

Categorical → add category "missing"

Numerical → create new column w binary yes/no (ie missing)

Create interaction terms w binary variable

Split 2 models → one w no missing & one w missing. like a tree

Imputation

Mean/median (numeric)

mode (categorical)

regression imputation

+ perturbation → imputed value less accurate but will give better estimate of the overall spread of the values

→ no more than 5% per factor

→ data used twice in regression will lead to overfitting

Optimization

Variables → decisions to be made

Constraints → restrictions on variables

Objective Functions → Solution quality measure

Solution → values for each variable

Diet Problem

Variables → x_i = amt of food i in daily diet

Constraints

$\sum_i a_{ij} x_i \ge m_j$ for each nutrient j

$\sum_i a_{ij} x_i \le M_j$ for each nutrient j

$x_i \ge 0$ for each food i

Objective function

minimize $\sum_i c_i x_i$

where a_{ij} = amt of nutrient j per unit of food i

m_j = min daily intake of nutrient j

M_j = max daily intake of nutrient j

c_i = per-unit cost of food i

Optimization Binary Variables

Variables → $y_i = 1$ if invest in stock, 0 if not

Constraint → $y_{AMZN} + y_{GOOG} + y_{AAPL} \ge 1$

Objective → include constraint w binary

New Soln = old soln + θt

step size, improving direction

Bayesian Model

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

Graph to Communities → Louvain Algo

Neural Network

recognize images, speech, writing, language

Non Parametric Test

McNemar (yes/no)

Wilcoxon Signed Rank Test (numeric)

Mann Whitney (unpaired)

Paired vs Unpaired

Exact Values → Parametric (mean)

Ranks of Values → Non Parametric (median)

Counts of Binary Outcomes → Binomial (which better) Ex comparing number of strikes thru A or B better

Competition → Game Theory

↳ perfect info (know everyone's strategy)

↳ imperfect info (no info about others)

↳ zero sum (I win, lose)

↳ non zero sum (total benefit higher or lower)

→ player & opponents' decisions must be accounted for.

Probability Models

Bernoulli $P(X=1)=p$ $P(X=0)=1-p$ (must be iid)

Binomial → must be independent trials
Probability of getting x successes out of n independent identically distributed Bernoulli (p) trials
$n \to \infty$ the Binomial Converge to Normal

pmf

$$P(X=x) = \binom{n}{x} p^x (1-p)^{n-x}$$

Geometric
Probability of having x bernoulli (p) trials/failures before first success

$$P(X=x) = (1-p)^x p$$

"number of tries between failures"

Poisson → good model for random arrivals

$$f_x(x) = \frac{\lambda^x e^{-\lambda}}{x!}$$

λ = average number of occurrences in a time period * memoryless

occurrences are iid

mean interarrival time is $1/\lambda$ so λ is the mean arrival rate per unit time

Exponential

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

If occurrences are poisson (λ) then time between successive occurrences are exponential (λ)

* memoryless

Weibull

$$f_x(x) = \frac{k}{\lambda}\left(\frac{x}{\lambda}\right)^{k-1} e^{-(x/\lambda)^k}$$

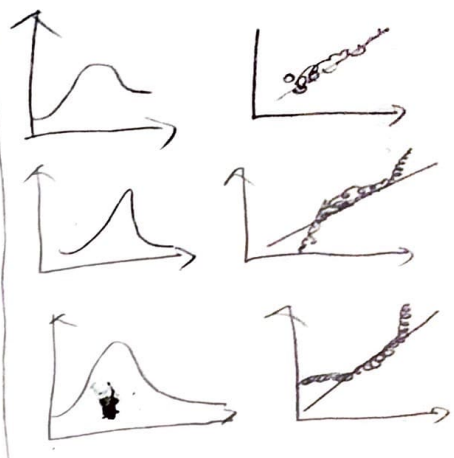
"time between failures"

$k<1$ → failure rate decreases with time
"worst things fail first"

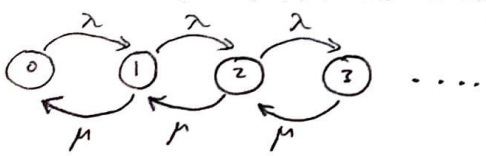
$k>1$ → failure rate increases with time
"things that wear out"

$k=1$ → failure rate constant w time } exponential

QQ Plot



Queuing



Arrival Rate = λ
Service Rate = μ

* memoryless.

Transition Equations (≥ 1 calls in the queue)

- $P(\text{next event is arrival}) = \frac{\lambda}{\lambda+\mu}$
- $P(\text{next event is finished call}) = \frac{\mu}{\lambda+\mu}$

Expected fraction of time employee is busy $= \frac{\lambda}{\mu}$ } utilization factor / traffic intensity

Expected waiting time before talking (per call in employee queue) $= \frac{\lambda}{\mu(\mu-\lambda)}$

Expected number of calls waiting in queue $= \frac{\lambda^2}{\mu(\mu-\lambda)}$

Avg no. of units in system (waiting & being served) $= \frac{\lambda}{\mu-\lambda}$

Avg time a unit spends in the system $= \frac{1}{\mu-\lambda}$

$P(\text{more than } k \text{ units in system}) = \left(\frac{\lambda}{\mu}\right)^{k+1}$

Simulation

Deterministic: same inputs give same outputs (no randomness)

Stochastic: system has randomness
↳ one random outcome might not be representative of system performance in the range of different situations that could arise

→ Validate a simulation by comparing it to real data as much as possible. If the simulation isn't a good reflection of reality, then any insights we gain from studying the simulation might not be applicable to reality

Markov Chain

memoryless i.e. don't care previous state only care present state

P_{ij} = transition prob from state i to j.

Steady State:

$(((\pi P)P)P)P \ldots = \pi P^{\infty}$ (hard to calculate)

alternative $\pi^* P = \pi^*$ when current state don't matter

and $\sum \pi^* = 1$

Memoryless: the next state of the process doesn't depend on any of its previous states, but does depend on current state

Game Theory

Cost = $1/gallon k = total demand

	BP $2.50	BP $2.00
Shell $2.50	Profit (both) $\frac{1}{2}(\$1.50) = \$0.75$ d	Profit (Shell) = 0
Shell $2.00	Profit (Shell) $= d(\$1) = \$1\,d$	Profit (Shell) $\frac{1}{2}(\$1) = \$0.5\,d$

Cox Proportional Hazard Model → Probability Model

Gradient Boost