Tutorial 3 - EMLT

MSE vs SSE → normalized so easier to compare w # of data points difference

MSE → mean of $\frac{1}{n}\sum(y_i - \hat{y}_i)^2$ } can be decomposed into bias variance

SSE → sum of sq error

RSS vs TSS

→ how much predicted vals likely to vary } quantified variance of error

$y_i = \hat{y} + \varepsilon$

$\sum \varepsilon = RSS = y_i - \hat{y} = (y_i - \beta_0 - \beta_1 x_1)$ ← predicted value

model { RSE = mean of RSS = $\sqrt{\frac{RSS}{n-p-1}}$

↓ obs ↓ predictors incl intercept

} lower better [comparing models w diff # of predictors i.e. takes degrees of freedom into acc]

TSS = Total variability of response $(y_i - \bar{y})^2$ } dataset

($\bar{y}$: mean)

$R^2 = 1 - \frac{RSS}{TSS}$ } $> R^2 \Rightarrow$ larger proportion of variability explained by model

if $\frac{RSS}{TSS} <$

$TSS > RSS$

TSS ↓ original data variability

RSS ↓ model variability

Simple + multiple LR output

… Output exceeds the size limit. Open the full output data in a text editor

OLS Regression Results

response variable

Dep. Variable:	mpg	R-squared:	0.821 — R^2
Model: → type of	OLS	Adj. R-squared:	0.818 } R^2 independ of DOF [# of p]
Method:	Least Squares	F-statistic:	252.4
Date:	Fri, 24 Nov 2023	Prob (F-statistic):	2.04e-139
Time:	18:49:52	Log-Likelihood:	-1023.5
No. Observations:	392	AIC:	2063.
Df Residuals: n - p	384	BIC:	2095.
Df Model:	7		
Covariance Type:	nonrobust		

t value → | p value → | 95% confidence interval

	coef	std err	t	P>|t|	[0.025	0.975]
inter → const	-17.2184	4.644	-3.707	0.000	-26.350	-8.087
predictors: cylinders	-0.4934	0.323	-1.526	0.128	-1.129	0.142
displacement	0.0199	0.008	2.647	0.008	0.005	0.035
horsepower	-0.0170	0.014	-1.230	0.220	-0.044	0.010
weight	-0.0065	0.001	-9.929	0.000	-0.008	-0.005
acceleration	0.0806	0.099	0.815	0.415	-0.114	0.275
year	0.7508	0.051	14.729	0.000	0.651	0.851
origin	1.4261	0.278	5.127	0.000	0.879	1.973

Omnibus:	31.906	Durban-Watson:	1.309
Prob(Omnibus):	0.000	Jarque-Bera (JB):	53.100

…

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

[2] The condition number is large, 8.59e+04. This might indicate that there are strong multicollinearity or other numerical problems.

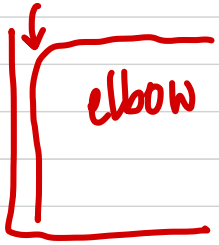
$$mpg = -17.2184 - 0.4934(cylinders) + 0.0199(displacement) + \dots + \varepsilon$$

[Estd err]

Prediction - quantitative response variable

name	idea	model	Model fit assessment	Assumption(s)	EXAMPLE
Simple	Least sq: get β_1 & β_0 by minimizing ε [std error] one predictor X	$Y = \beta_0 + \beta_1 X + \varepsilon$	hypothesis test + R^2	relation b/w Y & X_i linear + additive	# hrs student X studies vs grade in exam y
multiple	p predictors $x_1, x_2, \ldots x_p$ similar LSq approach minimize ε in this case RSS $= \sum (y_i - \hat{y}_i)^2$ $\beta_1 \beta_0 \ldots \beta_p$	$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \varepsilon + \vdots + \beta_p x_p$	Hypothesis test (F-stat) R^2 & RSE		# hrs student studied + x_1 # hrs sleep x_2 before exam + ... vs grade y

Classification — qualitative response variable

name	idea	model	Model fit assessment	Assumption(s)	EXAMPLE
Logistic regression [can be extended to multiple]	Fit β_0 & β_1 using maximum method [estimate probability]	$p(E) = \frac{e^{\beta_0+\beta_1 E}}{1+e^{\beta_0+\beta_1 E}}$ $p(E) = P(Y=1 \mid E)$	ROC curve 	• linearity (b/w odds & predictors) • independence • absence of multicollinearity	student passes exam or not (1 or 0) ← pass (1) or fail (0) depending on # of hours studied

Bayes Classifier — qualitative response with K distinct ordered classes [classification]

LDA ↓ QDA	only 1 predictor uses estimate for prior ($\hat{\pi}_k$), mean ($\hat{\mu}_k$) & variance ($\hat{\sigma}^2$) to assign $E=x$ to the class with largest $\hat{p}_k$ decision function	$\hat{\mu}_k$ = mean $= \frac{1}{n_k}\sum x_i$ $\hat{\sigma}^2$ → variance $= \frac{1}{n-K}\sum_{k}\sum(x_i-\mu_k)^2$ $\hat{\pi}_k$ → prior $= n_k/n$ $\hat{p}_k$ = delta	• confusion matrix • ROC	PDF = normal variance same across k classes	subscription service dataset with past customer behavior to predict subscription renewal

Eg #1 : Relation b/w # of hrs student studies & their score

• any relation? • how strong of a relation

1) Decide linear model — 1 predictor # of hrs + quantitative response

$Y = \beta_0 + \beta_1 X + \varepsilon$

- Y: response variable [exam score]
- β_0: intercept
- β_1: slope
- X: predictor [# of hrs studied]
- ε: error term

2) 'Fit model' - get β_0 + β_1 :

minimize sum of squared diff ε /Least sq method

3) Hypothesis test — is there a relation? } if β_1

use t-tests → formula for t value

$\beta_0 = 0$ & $\beta_1 = 0$ } H_0 = no relationship

H_1 = relationship

larger ⇒ stronger evidence against H_0

4) Strength of model — how much relationship? } R^2 [closer to 1 ⇒ better fit]

5) assumptions + testing

Linearity — test using residuals plot (obs vs predicted values)
if no pattern ✓✓

additive - not relevant

need see relation b/w salary (Y) based on → years of experience (X_1)
level of education (X_2)
location? (X_3)

(i) decide multiple regression [2/3 predictors + quantitative response]

salary ← $Y = \beta_0 + X_1\beta_1 + X_2\beta_2 + \varepsilon$ → error term

β_0 → intercept
X_1 → YOE
X_2 → level of education
β_1, β_2 → coefficients — imp : β_1 represents expected change in Y for a one-unit change in X_1 holding X_2 constant

(ii) $\beta_0, \beta_1, \beta_2$ estimated using least squares (similar to OLS)

(iii) Hypothesis testing:

$H_0 : \beta_1, \beta_2 = 0$ } no effect
$H_1 : \beta_1, \beta_2 \neq 1$ } effect

} use test statistic (same as OLS)

(iv) strength of model — using R^2 & RSE — covered in lecture

(v) assumptions

linearity : residual plot to test [individual plot of response + predictor shows linear]

additive : effect of each predictors — yrs of experience
+
education
is independent of the other
[no interaction b/w the two]

Logistic Regression

Scenario – likelihood of student passing based on # of hrs of study (X_1) + whether or not attended prep course (X_2)

model: $\log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2$

↳ multiple predictors

p = probability of passing the exam

interpretation: β_0 = log-odds of passing when X_1 & X_2 are zero

β_1 : change in log-odds for one unit increase in X_1, while holding X_2 constant

if we convert this to probability of student scoring low, medium, or high marks, it becomes multinomial logistic regression

LDA example

- binary target : 1 = subscription renewed } have in data
  0 = not

- have data — interactions, usage patterns & other features

} if a customer renews a service

Train LDA model - using features + response

& use to make prediction

becomes QDA when spread / variability of data within each class is not same

[can't decide contextually — have to do data exploration + model tweaking]

+ becomes Naive Bayes when stronger assumptions

[text classification — spam filtering]

knn example — predictors mix of qualitative + quantitative

- iris flower species based on characteristics (sepal, petal width + length)
- customer subscription renewal
- fraud credit card transaction

in pre data - compare k # with all characteristics. Assign to where highest # lies

adjust k according to results

maybe $k = \sqrt{n}$
↳ # of data pts

[algorithm includes distance calc]

Resampling

name	idea	model	Practical example	Drawbacks	Python relevant function(s)
Validation set approach	• use train + valid sets • find error rate for predictions on validation set [validation is an intermediate step]	:	will share a document detailing steps	• validation set is randomly created so error rate can vary • splitting in 3 means train set is smaller so possible that error(validation) >> error(test) [model trained on fewer data pts]	• train_test_split() og data / train step1 \ temp / valid step2 \ test • predict()
LOOCV	• same as above with only 1 observation comprising of validation set • but do it n times and take avg of error	• can be anything OLS / logistic / etc. assessment: • MSE for regression • # of missclassified obs for classification		• fixes bias & test error issues seen above BUT, computationally VERY expensive!! [only worth it (possibly) for linear reg]	cross_validate() ↳ with cv = n-1 [can do data.shape[0]]
k-fold	• divide data into k groups • fit model using all groups excluding 1 • predict on k^{th} group • compute error + repeat k times and average the error			• fixes bias & test error issues seen in ① • fixes (most) computation issues seen in ② • ideal!! [in most cases]	cross_validate() ↳ with cv = kFold() function to partition data (randomly) into k groups)
Bootstrapping [StatQuest has an amazing video on Youtube] (someone called it the general purpose tool for quantifying uncertainty in a stat model [getting SE]	• randomly sample [with replacement] n observations • compute the statistic • repeat the steps (resample with replacement) • compute SE [tells us how far estimate is from actual val; SD of distb.] • infer • best part – requires no assumptions abt distribution of data	can be used to estimate parameters like: mean, median, CIs, SDs also can be used in context of regression to estimated S.E of coefficients ++ many others	mostly relevant in cases where population data is not accessible • for eg performance of drug on few people [avg perf] • mean height of students in school using 1 section from each class	• fails if sample is not representative of the population :c [no bias correction]	covered in this course: • custom written in ISLP To explore in your own time: scipy.stats.bootstrap()

Subset selection

why? → increase interpretibility [less confusing with fewer predictors]
→ decrease (chances of) overfitting [a.k.a. curse of dimensionality]
→ increase computational efficiency
→ combat multicollinearity issues

Best subset selection {fit OLS model for every comb of predictors} → !!! computationally expensive
1) $\forall\ k = 1, 2, \ldots p$
2) All comb of models with k predictors are fitted & h_k is picked (done p times)
3) pick best out of $h_1, \ldots, h_p$ [using criteria]
↓ best out of those [smallest RSS or largest R^2]

Subset selection
[uses a criteria — (why diff from RSS/R^2) ↳ because depends on train error (which is lowest with all p)
C_p, AIC, BIC, Adjusted R^2]

stepwise model selection

Forward stepwise
1) h_0 = no predictors (inital model)
2) $\forall\ k = 0, 1, \ldots, p-1$
- Fit all models with all predictors in h_k and 1 additional predictor ★explain
- h_{k+1} = the best [smallest RSS of highest R^2]
3) pick best of $h_1, h_2, \ldots, h_p$ using criteria

Backward stepwise
1) h_p = all predictors
} same steps by dropping 1 instead of adding this time

step 1 h_0 + 1 predictor ∀ predictors + pick best picked = h_1
step 2 now h_1 + 1 of all predictors; best = h_2

why? • reduce multicollinearity
• if $p > n \Rightarrow$ overfitting

Shrinkage methods

↳ Ridge regression
↳ formula → our coefficients
minimize RSS + $\lambda \sum_{j=1}^{p} \beta_j^2$ } shrinkage penalty
Tuning parameter { $\lambda = 0 \Rightarrow$ least squares }

$\lambda \sum \beta_j^2$ = penalty
penalizes large coefficients?

λ has to be picked / tuned — maybe using cross validation techniques?

↓
- overfitting
- multicolinearity (unstable)
- interpretability
- generalization
- outlier robustness

↳ Lasso [very similar; just uses a different formula]

Check out slide 23 from 6.5 for detailed differences

- less computationally exp than subset selection

essentially — introduce "some" bias to get "major" drop in variance

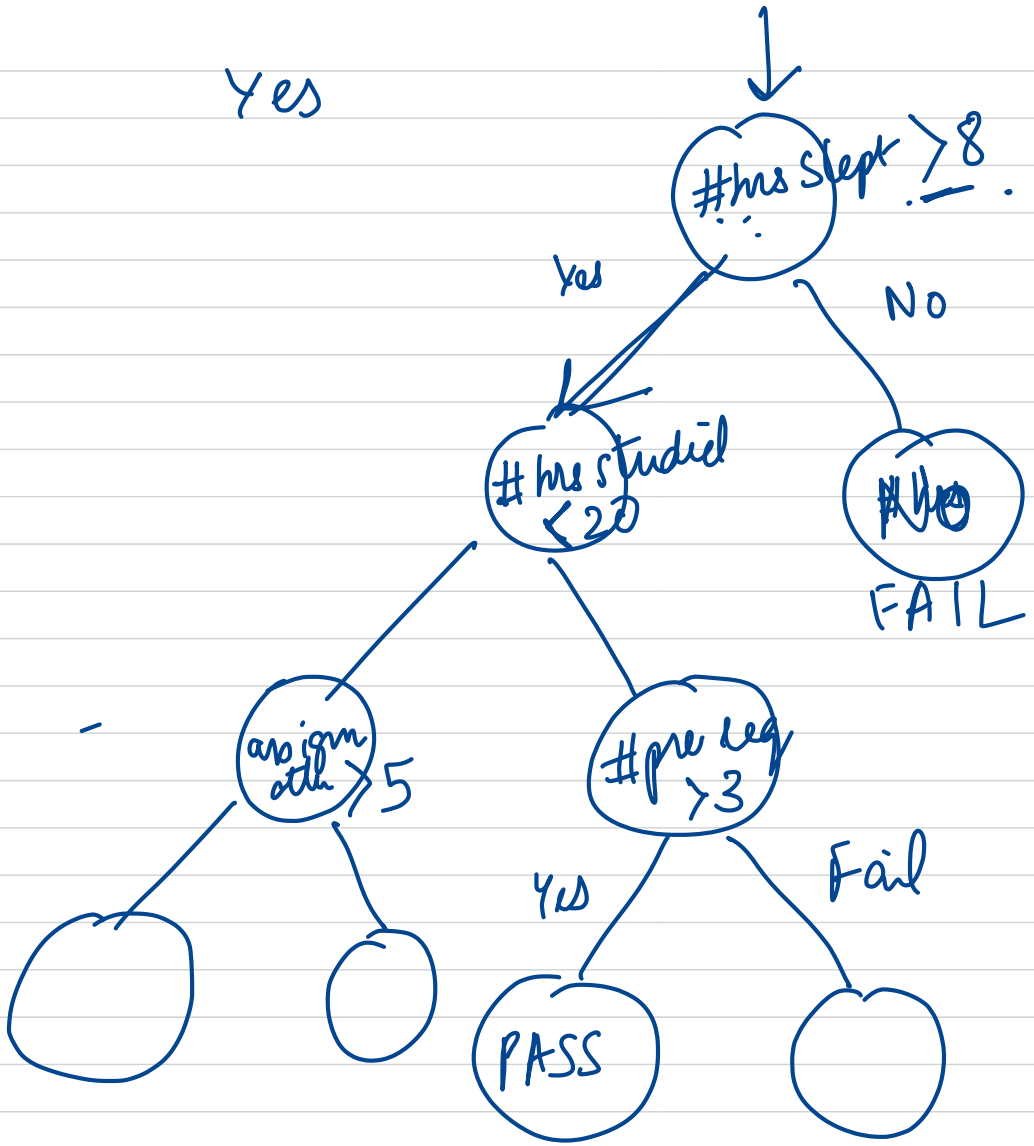
Tree methods

Why?
- interpretable
- can handle qualitative data
- can be displayed graphically
- don't need scaling / normalization

Why not?
- predictive accuracy lower than other models
- small change in data can lead to a large change in the tree

2 Decision Trees :

Tree	Predicts	Steps / Algorithm [similar for both (exception & cost function)]	Cost function
Classification	qualitative variable	1) Tree built using recursive binary splitting (on train data) → greedy method (locally optimal choice) (i) All values lined up, different split points tested [using cost fn chs] (ii) select X_i with least Gini index / entropy (iii) Repeat (i) & (ii) till you reach "stopping criteria" ↳ eg 5 obs per region 2) Cost complexity pruning (explained below) + k fold validation 3) return "best" subtree	Gini index / entropy
Regression	quantitative variable	1) Tree is built by recursive binary splitting (on train data) ↳ $X_1, X_2 \ldots, X_p$ (i) All values lined up, different split points tested [using cost fn chs] ↳ RSS (ii) select X_i with least RSS (iii) Repeat (i) & (ii) till you reach "stopping criteria" ↳ eg 5 obs per region [min count on # of train instances @ leaf] 2) Use cost complexity pruning to simplify large / complex trees ↳ prone to overfitting, high variance [K-fold validation to pick optimal α] Calculate a "tree score" for each tree with α (multiple values) easy formula = $SSR + \alpha(T)$ ↳ (removed nodes) ↳ leaves To pick α — start with all data & fit a tree. Then perform cross valid with diff-α trees • $\alpha = 0$ first, then increase α until pruning leaves gives lower tree score ($SSR + \alpha(T)$) 3) Return the "best subtree" from step 2 with corresponding α	RSS [in case of pruning for classification, replace SSR with gini index.



No · Student - pass / fail

hrs studied ✓

hrs slept ✓

assignments attempted

pre-req courses taken ✓

hrs slept 21 assign
8, studied 7

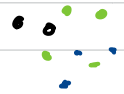
Ensembling techniques

1) Bagging (To reduce variance) ; concat: less interpretable

(→ with replacement)

i) sample B bootstrapped training sets from the data set

ii) construct B regression trees (no pruning)
classification



iii) For a test observation, average predictions from B sets
- most commonly occuring class

2) Random forests (similar to bagging)

i) B bootstrapped training sets

ii) B trees from bootstrapped sets but at each split, random subsets of $m = \sqrt{p}$ predictors considered.

iii) same as above

↓
to get uncorrelated trees & lower variance as this ensures trees don't always choose the most powerful predictors

Bagging / Forests vs boosting
↓ ↓
additive sequential

BART – combination of both

3) Boosting — also additive (more sequential)

(i) Fit a regression tree
(ii) compute the residuals ↓ the new tree's train set will be weighted more towards errors
(iii) Fit a new tree with d nodes using residuals as response values
(iv) this will be the base tree
(v) update residuals and fit a new tree, add a shrunken version of this to the base tree
(vi) repeat B times

Algorithm in lecture slides

4) BART (Bayesian Additive Regression Trees) [Additive] (Stanford online video on youtube)

idea: K trees & B iterations, at every iteration we still maintain k trees but make some sort of random changes to each tree (more mathematical)

Final prediction – avg prediction across all k trees at all steps

$\hat{f}^b(x) = \sum \hat{f}_k^b(x)$ ← b^{th} tree

$\hat{f}_k^b(x)$ = prediction at x for the k^{th} regression tree in the b^{th} iteration

at the end of each iteration, K trees from that iteration are summed

Steps: (i) At the first iteration — all k trees have a single root node

notation (mean of response / # of trees) $\hat{f}_k^1(x) = \frac{1}{nK}\sum y_i$; $\hat{f}^1(x) = \frac{1}{n}\sum y_i$

(ii) At each iteration after, predictions from all but the k^{th} tree are subtracted (b) for each tree k. from response at previous iteration (partial residual)

$$r_i = y_i - \sum_{\forall k' < k} \hat{f}_{k'}^b(x_i) - \sum_{k' > k} \hat{f}_{k'}^{b-1}(x_i) \qquad \forall i = 1, \ldots, n.$$

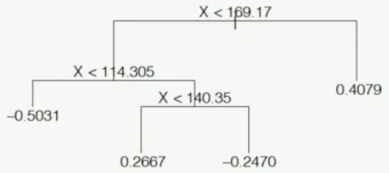
(iii) Fit a tree to the partial residual by performing a random perturbation (still best fit from options) to the tree from $b-1^{th}$ iteration

[eg: add/prune branches
change prediction in each terminal node of tree]

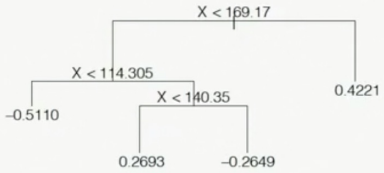
SVC : (perturbations)

Examples of possible perturbations to a tree

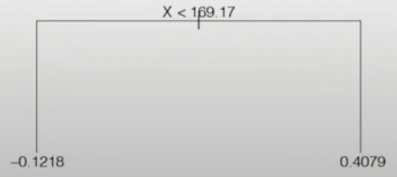
(a): $\hat{f}_k^{b-1}(X)$



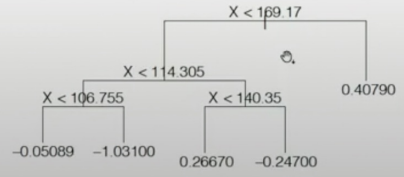
(b): Possibility #1 for $\hat{f}_k^b(X)$



(c): Possibility #2 for $\hat{f}_k^b(X)$



(d): Possibility #3 for $\hat{f}_k^b(X)$



Support vector machines

Basic idea:

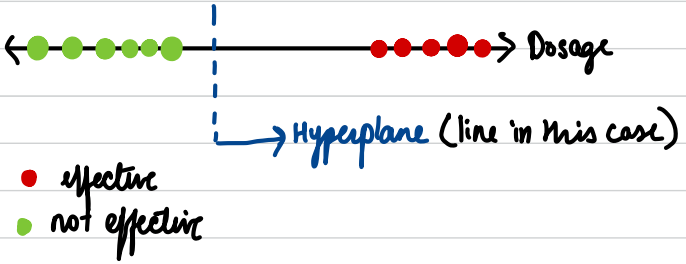
use Hyperplanes to categorize into classes

2-D - line

3-D - plane

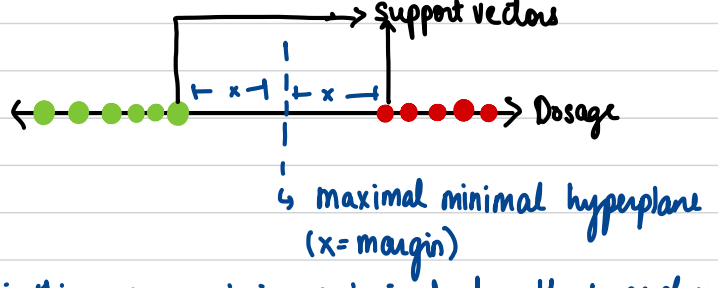
4+ D - hyperplane with p-1 dimensions

Hyperplane - divides p-dimensional subspace into two



Maximal minimal classifier

↳ hyperplane: farthest from training obs



in this case, margin is maximized when the hyperplane is in center

caveat: sensitive to outliers, overfitting

Support vector classifiers

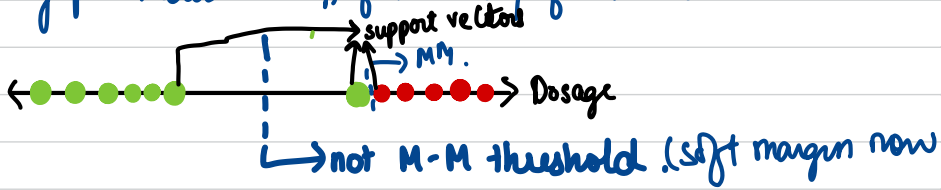
[similar to M-M but uses a "soft margin"]

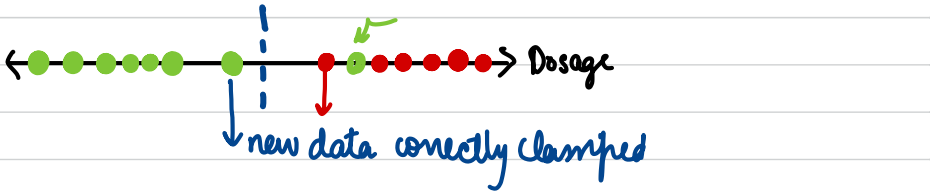
↳ margin that does not perfectly separate the 2 classes

back to bias-variance tradeoff — in this case, misclassifying a few training obs, lead to better test results

How?

Tuning parameter C → # & severity of violations





- C is chosen using cross validation

rule: no more than C observations on the wrong side of hyperplane

C = 0 ⇒ Maximal margin hyperplane

C >> ⇒ less variance, high bias

C << ⇒ high variance, less bias

SVMs:
start with data in low dimension, move the data into a higher dimension,
& find a classifier that seperates the data into 2 groups.

How? → using kernel functions (polynomial)

one-v-one classification
- compare each comb of 2 classes
- classify test to one of these two
- assigned to most frequently classified class

one-v-all
- K SVMs — compare each class to K-1 classes
- assign test obs to class with observation farthest away from hyperplane

Start at 9:05 / 9:07 depending on attendance
9:05 - 9:20 6.9
9:20 - 9:30 6.9 exercises
9:30 - 9:45 6.10
9:45 - 10 6.10 exercises

6.9 Survival analysis & censored data } example: wait time for a call centre

outcome – "time until event occurs"

T: true survival time } T is independent of C

C: censoring time → when patient drops out of study / dies during [censored individuals]

$Y = \min(T, C)$; $\delta = \begin{cases} 1 \\ 0 \end{cases}$ if $T \leq C$ ← status indicator

Summary:
K-M (only time)
Log-odds (time + feature (like drug / gender))
Cox (time + feature + age (numeric))

important concepts:

- independent censoring (T is indp of C)
  ↳ example: sicker sick patients drop out early, overestimates survival time

Kaplan-Meier survival curve $S(t) = \Pr(T > t)$
↳ quantifies probability of surviving past time t.

- $d_1 < d_2 < \cdots < d_n$ denote the K unique survival times among the non-censored individuals.
- q_k denotes the number of events that took place at time d_k. (i.e. the number of patients that died at time d_k)
- τ_k denotes the number of individuals alive and in the study just before d_k (the **at risk** patients).
- The set of patients that are at risk are the **risk set**.

Probability of surviving past time d_k

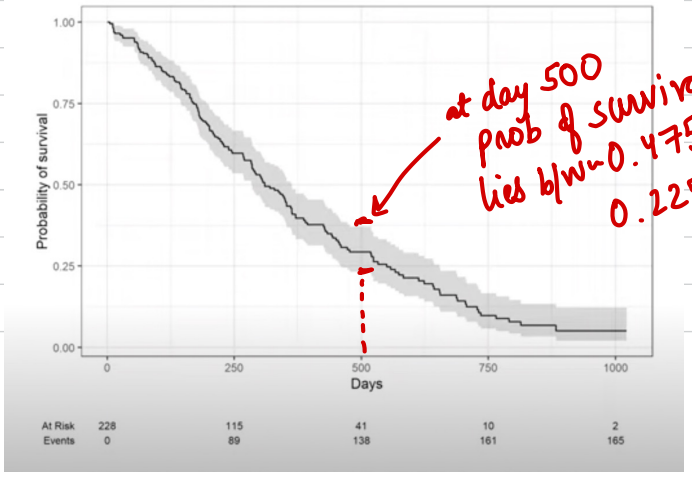
$$\hat{S}(d_k) = \prod_{j=1}^{k} \left(1 - \frac{q_j}{\tau_j}\right)$$

Day 1: at least one patient passes away

Probability of survival $\left(\frac{\#\text{ of surviving patients}}{\text{patients at risk}}\right)$

Then:
on each day, multiply prev possibility of survival by
↓
when at least another patient dies

$\frac{\#\text{ of surviving patients}}{\text{patients at risk}}$



at day 500 prob of survival lies b/w 0.475, 0.225

Log-Rank test

tests if there is a difference b/w two survival curves

$$W = \frac{E - E(E)}{\sqrt{\mathrm{Var}(E)}}$$

expectation

expectation to cheat sheet!!

↳ variance under assumption of H_0 (no diff)

W ≈ normal

get p value to test

$$W = \frac{\sum_{k=1}^{n} (q_{1k} - E(q_{1k}))}{\sqrt{\sum_{k=1}^{n} \mathrm{Var}(q_{1k})}}$$

# of patients that died at d_k in group 1

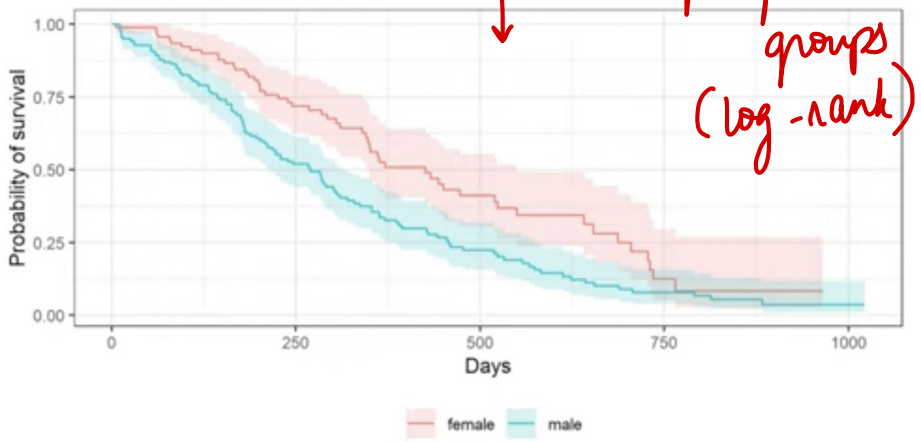
log-rank test stat

$$\mathrm{Var}(q_{1k}) = \frac{q_k (z_{1k}/z_k)(1 - z_{1k}/z_k)(z_k - q_k)}{z_k - 1}$$

# of ind. @ risk at time d_k in group 1

incidents

to compare for 2 groups (log-rank)



female					
At Risk	90	53	21	3	0
Events	0	24	42	52	53
male					
At Risk	138	62	20	7	2
Events	0	65	96	109	112

Regression with survival response

To predict time survival time (T)

n obs, $Y = \min(T, C)$

$\delta = \begin{cases} 1 & \text{if } Y = T \\ 0 & \text{otherwise} \end{cases}$

Hazard function: death rate in the instant time after t, given survival past that time

$$h(t) = \lim_{\Delta t \to 0} \frac{P_r(t < T \leq t + \Delta t \mid T > t)}{\Delta t}$$

↳ (very small)

Cox's proportional hazard model: response is hazard

estimate β (coefficients) w/out specifying form of $h_0(t)$

- maximize partial likelihood w.r.t β
- get p vals for $H_0: \beta_j = 0$
- get C.I.s for coefficients

Only assumption: hazard ratio is independent of time for 2 grps

6.10 Unsupervised learning – no response

goal: gain info abt features (x values)

PCA

problem: for large n, impractical to draw individual 2-feature plots to get relationships

solution: find small # of dimensions that observations vary along the most (PCA!) {converts correlations into a 2-D plot}
↳ highly correlated values cluster

each dimension found is a linear comb of p features: together form: $Z_1 = \phi_{11}X_1 + \phi_{21}X_2 + \dots + \phi_{p1}X_p$

- ϕ: loadings (maximize variance of Z_1) {and normalized}
- X: vector of n observations for the j^{th} feature (x_j)

Z_1 = vector with $z_{i1} = \phi_{11}x_{i1} + \phi_{21}x_{i2} + \dots \phi_{p1}x_{ip}$ scores

each Z_i is constructed sequentially (using same procedure) with Z_i being uncorrelated to Z_{i-1} [directions become orthogonal]

if $i > j$ Z_i more imp that Z_j

Things to look up/be aware of:
- proportion of variance
- uniqueness of principal components

Matrix values & matrix completion

problem: missing predictor values

solution: 1) remove rows with missing values
2) replace x_{ij} with mean of j^{th} predictor
3) matrix completion (using PCA)
- only possible when reason for missing data is random

Clustering — find subgroups in data set

k-means – partition into k distinct, non overlapping clusters {no observations ∈ >1 cluster and at least 1 cluster

$C_1, C_2, \dots C_K$ = sets containing observations

Steps: (i) Pick a K
(ii) Randomly assign each obs a cluster
(iii) Repeat until static:
→ for each cluster, compute the centroid [for k^{th} cluster, p dimensional vector of the feature means for the observations in the k^{th} cluster]
→ assign each observation a cluster using the closest centroid

[local optimum] → depends on initial assignments

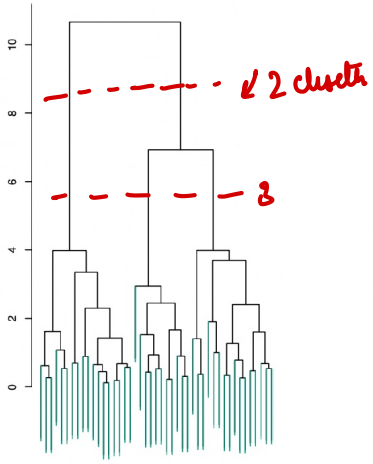
to make global:
- run the algo with different random initial assignments
- choose clustering that minimizes within-cluster variation

$$W(C_k) = \frac{1}{|C_k|}\sum\sum(x_{ij} - x_{i'j})^2$$

(euclidean dist b/w each obs)

Dendrogram (hierarchical clustering)

Dendrogram



Hierarchical clustering results in a **dendrogram** which is a tree-based representation of the observations.

- Each leaf (green stick) is an observation.
- As we move up the dendrogram, observations that are similar fuse into branches.
- Then branches fuse into other branches which indicates that the groups of observations are similar.
- The height at which two observations fuse indicates how different the two observations are.

horizontal splits to create clusters { using euclidean dist / correlation based dist

Linkage types (choice affects dendogram produced)

- complete (maximal intercluster dissimilarity)
  - all pairwise dissimilarities b/w clusters A & B
  - record the largest dissimilarity
- single (minimal intercluster dissimilarity)
  - step 1 - same as above (ii) record smallest
- avg • record avg (step #2)
- centroid

compute dissimilarity of centroids for clusters A & B (instead of pairwise)

Algorithm (hierarchical clustering)

- treat each of the n obs as its own cluster
- compute pairwise dissimilarity b/w each obs
- for i = n, n-1, .., 2:
  - get pair of clusters least similar among i clusters
  - fuse at height that indicates similarity (in the dendogram)
  - repeat for i = i-1