data mining Cheat Sheet

by mvyjayanti via cheatography.com/72036/cs/18263/

Naive Bayes and LogReg

P(A|C) = (P(C|A)P(A))/P(C)	predicts T/F, "S" shaped, from 0-1
posterior = (likelihood x prior)/normalizing constant	log(odds) = log(p/(1-p))
pros: easy/fast, assuming independence, categorical	z = estimated intercept/std error
cons: if not in set -> 0% -- can use Laplace estimation (add 1), bad estimator, independent predictor assumption -> unlikely	y = log(F)B1 + log(T/F)B2
LR: p = $e^{\log(odds)}/(1+e^{\log(odds)})$	likelihood = mul. all T x all (1-F)

log(L) =sum i to n(log(Tn) + sum(log(Fn))

R^2=(SS(mean) - SS(fit))/SS(mean)

ANNs

neuron = things that hold number from 0 to 1	boolean: T=1, F=0
ŷ = 1 if $w_1x_1+w_2x_2+...w_nx_n-t$(bias factor) >0	, -1 if <0

ANNs (cont)

$\hat{y}=sign(w_1x_1+w_2x_2+...w_nx_n-t=sign(\mathbf{w}\cdot\mathbf{x})$ λ=learning rate xij=val of jth attribute of training example xi	for weight update: $w_j^{(k+1)}$ = weight param associated w/ i^{th} input link after k^{th} iteration
$w_j^{(k+1)}=w_j^{(k)}+\lambda(y1-\hat{y}_i^{k}\hat{})x_{ij}$	error = y - ŷ
if error = 2, inc w of +ves	if error = -2, in w of -ves
Error E = ΣEk | k∈outputs	Ek = $1/2(tk-ok)^2$
output oi = $1/(1+e^{-net\ i})$	net i = Σwij*oi

Inductive Bias, No Free Lunch

IB: anything influencing hypothesis choice other than training set	part of language accessible, method of choosing
NFL: for any 2 algorithms A&B, there exists a dataset for which A outperforms B	assuming uniform P(x,y)→#of datasets for which A>B = # B>A

SVM

frontier that best segregates 2 classes by margins	polynomial kernel: k(x,y)= $(x*y+1)^d$
RBF kernel:k(x,y)= $e^{-\gamma(\|\|x-y\|\|^2}$	tune by k-fold cross-val (k=5)
adv: high dimension spaces, #of dimensions > #of samples	diff kernel functions for diff decisions
k1+k2 = even more complex	dis: if #features > #samples, CV
min($\|\|w\|\|^2$) for linear ξ: how far pt_i is from correct side	$wx_i+b>=1-\xi$ if $y_i=1$ $wx_i+b>=-1+\xi$ if $y_i=-1$
min(||w||+C(Σi=1→nξ_i))	max(($\Sigma\lambda_i$) - 1/2($\lambda_i\lambda_jy_iy_jx_ix_j$))
dist btw parallel planes = z/||w||	||w|| = sqrt(w_1+w_2...)
generalization error<= p(bar)(1-s^2)/s^2	p(bar) = avg correlation, s=strength

Errors

P(+) = 1/(1+ $e^{-(w0+w1x1...)}$)	error= misclassification
For new cases, predict:	1 if (w0+w1x1+...)->=1, 0 else
if w0 inc as x inc, p(+) inc	error = (FP+FN)/All
sensitivity = TP/(TP+FN)	specificity = TN/(TN+FP)
+ve predicted val = TP/(TP+FP)	-ve predicted val = TN/(TN+FN)
true error: error on true underlying distribution (unmeasurable)	apparent error: error on example used to train model (underestimates TE)
generalization: ability to predict unseen cases	Occam's Razor: should not be multiplied beyond necessity
Overfitting: memorizing training set	test error: error on ex. held out of training

KNN

select k: sqrt(n), if n is even, choose odd	Ri = {x:d(x,xi)<d(x,x2), i!=j}

euclidean distance =sqrt($(x-x1)^2+(y-y1)^2$)

Model Eval

Holdout	train on 2/3, test on 1/3, one is validation set (high variance on estimate)
Leave-one-out	train on N-1, test on 1 (good estimate)
K-folds Cross Val	divide set into k parts, LOO each, repeat N times, compute mean and std dev for each
Bootstrapping	randomly draw N points (can repeat), train, test on S - S1
Compare 2 methods: H0: meanLR = meanNB, H1: meanLR<meanNB	t=(meanNB-meanLR)/S (S:pooled variance), reject H0 if t>t alpha
OR H1: meanLR!=meanNB	2-tailed t. t alpha/2. ((meanVar)x(sqrt(n))) / S

Model Eval (cont)

OR H1: meanLR!=meanNB	2-tailed t. t alpha/2. ((meanVar)x(sqrt(n))) / S

Better: stratify each fold to contain same % of positives and negatives

Decision Trees

asks a question: classifies based on T/F	root, internal(arrows to and from), external(arrows to) (leaves)
break into categories	T/F and Y/N for each
P(Y|T), P(N|T), P(Y|F), P(N|F)	GI^2= 1-(Y|F/(Y|F + N|F))^2+(N|F/(Y|F + N|F))^2
GI^1 =	1-(Y|T/(Y|T + N|T))^2+(N|T/(Y|T + N|T))^2

GI^{all}=(T/(T+F) x GI^1)+(F/(T+F) x GI^2)

entropy: H(S)=P(y)log2(P(y))-P(n)log(P(n))	find H(S^{true}) and H(S^{false}), H(S)-w1H(S^{true})-w2H(S^{false})
w1 = T instances/all	w2 = F instances/all
w1 = T instances/all	w2 = F instances/all

largest info gain, least GI

ROC and Lift Curves

ROC: sensitivity vs. (1-specificity), higher val the better, flatter line the worse	sens: TP rate, 1-spec: FP rate
Lift curves: find % of each total response from sum of all	find % of each +ve responses from total +ve responses
y = +ve % / % of total	x = % of total

k-means clustering

user choose k, initialize k centers, loop: assign pts nearest those centers, move centroid of assigned pts	center in dense regions or random, optimizing (total distance)^2

returns local solution

Ensembles

Bagging: bootstrap aggregating	Boosting: changing weights on pts and building series of classifiers, start w=1

Ensembles (cont)

incorrect pts weighed by # that is inversely proportional to training error	w inc if misclassified, dec else classifiers combined by weighting-accuracy of training set
Arcing(Adaptive resample&combine): like boosting but change w by update method	eg. Arc x4: w(x) = 1+e(x)^4 e(x)=times x has been misclassified so far
depends on: strength(perf of individuals), diversity (uncorrelated errors)	bagging error: from reducing var boosting can reduce bias&var | bagging is > base classifier
boosting better or overfit noisy	Random forests: for tree,choose pts,for node, features subset w/ best IG,split, end,recurse, end

feature selection

removing irrelevant info for a better, faster model	drop missing values or encode them
drop: if all values are the same	if highly correlated, one of them
if low correlation with target| trees with least info gain	forward, backward, stepwise selection: best model with f1, then keep going until validation error stops dropping
beam or heuristic search	for computation interpretability | genetic algorithms
1) filters: all above + other correlation	2) wrappers: build a classifier with a subset+eval on validation data. but 2^d possible subsets

Bias and Var

PCA : dimensionality reduction	linear combo of OG features
max. variance: smallest # until 90% var explained	μ=E(y|x)=T(uk) ŷ=f(x,Θ)

Bias and Var (cont)

error: MSE = $(\hat{y}-\mu)^2$ var: $E(\hat{y}E(\hat{y}))^2$ bias:$(E(\hat{y}-\mu)^2$+noise	^best estimate of y given x and fixed params Θ

KNN,ANN,DT: low bias, high var

var: how much does my estimate var across datasets\| bias: systematic error prediction, inability to fit

EM Expectation Maximization clust.

hard clustering: each pt only belongs to one cluster	soft clustering: can belong to more than one cluster by %
EM: automatically discover all params for k "sources"→but we may not know source if we know μ,σ, can find likeliness	mixture models: probabilistic way of soft clustering each cluster Gaussian or multinominal
$1/\mathrm{sqrt}(2\pi\sigma^2)$*exp$(-(x_i-\mu_\beta)^2/2\sigma_\beta^2)$ $a_i=1-b_i=P(a_i)$	Bayesian posterior: $b_i = P(b\|x_i) = (P(x_i\|b)P(b)) / (P(x_i\|b)P(b) + P(x_i\|a)P(a))$

EM Expectation Maximization clust. (cont)

$\sigma_\beta^2=(b_1(x_1-\mu_\beta)^2+...)/(b_1+b_2+...)$	$\mu_\beta = (b_1x_1+b_2x_2+..)/(b_1+b_2+...)$
em: places randomly,for each pt $P(b\|x_i)$: does it look like it came from b	Working to adjust (μ_a, σ_a^2) and (μ_β, σ_β^2) to fit points assigned
Iterate until convergence P(a) = 1- P(b)	Could also estimate priors: P(b) = $(b_1+b_2+...)/n$

“What proportion of the data is each distribution describing”