Sample Selection

Probabilistic: uniform (equal chance to be selected/hard to practice: no resources): Random/Stratified (30%+70%--pick sample)/Systematic (select skip 2 ppl)/Cluster (random N clusters—pick 1 cluster)
Non-Probabilistic: non-uniformly (bias problem): Convenience/purposive/snowball (tree)/quota (cluster)
Bias: voluntary/under coverage/non-response/convenience/response/over coverage
Bias in data and AI: culture/unbalanced train data/feature fail capture/feedback loop (based on what happened in the past)
NAs: drop/replace (duplicate, r.v./stats/closest)/

Regression:

KNN:

Take average of k-closest observations: Distance: numerical Majority: categorical
Loss Function: MSE (square: avoid cancel out, absolute: not differential, root: keep same unit)

Linear Regression

Loss Function: MSE (minimize) = 1/n ||yi-yi_hat||^2, beta_hat=(X'X)^-1X'Y
Regression line: y_hat=beta1_hatX+beta0_hat (beta1_hat=sum(xi-xbar)(yi-ybar)/sum(xi-xbar)^2
Qualitative Predictors: create dummy variable (1 or 0)--yi=beta0+ei (if xi=0) [1,0,0][0,1,0][0,0,1]
Interpretation: beta0: average outcome among (xi=0), beta0+beta1=average outcome among (xi=1), beta1 average difference in outcome between 0 and 1 有 interaction 时要看清楚

Poly Regression

和 linear 一样就是 beta_M*x 变 beta_M*x^M

Model Assessment

Validation

Train: (x,y) Validate(x,y_hat) | Test (y)
Overfit: predictor 太多(high dimension feature, poly degree high, cross terms 多)/coefficients extreme
选好的 predictor 组合：stepwise(forward)/validation
Stepwise: N predictors=2^N models O(J^2)<<2^J
CV: score=averaged Loss Function across all validation sets
K=10=>10 uniformly sized chunks=10 iteration=10 parallel models
Leave-one-out: 不分 chunk 极端版本 CV max computation
Bootstrap: sample uniformly data from observed by estimating stats property (with replacement) until same observations as original data=new parallel universe
Standard Error: n increase, se(beta0) decrease; larger converge=spread well=sum(xi-xbar)increase; better data=smaller sigma
CI: beta0_hat+-2SE(beta0) If repeat create Cis from sampled data (by bootstrap), there's 95% probability that target parameter will be in [,]

Regularization

Modify Loss Function: add penalty; Minimize L: find smaller betas and avoid too much betas
lambda: regularization parameter 用 cv 选>>close to zero: recover MSE; too large: beta_hat close to 0. Lambda + coefficient -
Ridge(l2): square 算起来方便一点因为 mse 也带 2; penalty 是圆形 找圆形和 L 边圆的交点; lambada + penalty + 减少 beta 数量 make coefficient small
LASSO(l1): absolute; 0 时没有 derivative; penalty 是正方形 L 圆心为 beta 找正方形顶点和圆形的交点 lambda+ beta – beta 趋近于 0 zero out most coefficient=variable selection

Confusion Matrix

FPR: FP/(FP+TN) FNR:FN/(FN+TP)

Decision Tree

Take average of the output value Forward (start empty)
Argmin{j(feature), t_j(分割点的 value)} {N1/N MSE(R1)+N2/N MSE(R2)}

Classification:

KNN

Take Majority of K nearest neighbors: Distance (Euclidean(normalize)/Hamming/Manhattan)
K- grained decision boundary + variance; K+ smooth boundary + bias
Range 差别太大时：normalize/scale; 一个 group 太多时：sampling more

Logistic

Use logistic function to model pi=P(Y=1|X)=1/(1+exp(-beta0-beta1x))
Interpretation: beta1<0, higher predictor, lower outcome; For every 1 unit increase in outcome, log odds change be beta1 so odds change by exp(beta1); the observed outcome for class 1 is xx% while it is 1-xx% for class 2
Odds: P(Y=1|X)/(1- P(Y=1|X)) >> In(Odds)=beta0+beta1X; P(Y=1|X=0)/P(Y=0|X=0) >> In(Odds)= beta0+beta1X=beta0
Cross Entropy: l(p|Y)=-sum[yilogpi+(1-yi)log(1-pi)]
Beta0 – shift right / ; beta1+ <0 \ shift right; beta1+ >0 / shift right

Code: 1/C=lambda

Multiclass

One vs Rest(All): class 2 vs class 1+3>>train 3 binary classifiers
All vs All: 1 vs 2, 2 vs 3, 3 vs 1>>train C(K,2) binary classifiers>>test pts: most winner pt
KNN: K=3; Logistic: In(P(Y=j|X)/(P(Y=K|X)))=beta0,j+beta1,jX >> largest prob

SVM and kernels

Decision Boundary: w^T x+b=0 >> choose best w
X = decision surface(x_a)+direction of w (r*w/||w||) >> w^T x+b=r ||w|| >> r=y(x)/||x|| distance of point x to the decision boundary D (+/-)
Unsigned distance: tn*y(xn)/||w|| {tn=+1/-1}
Optimize: argmax(w,b) 1/||w|| min(n in 1,N) tn(w^T xn+b) Distance
Xn*=closest pt to D: tn(w^T Xn* +b)=C>0 >>tn(W*^T Xn+B*)>=1 (solve for b) >> all pt satisfy tn y(xn)>=1
Specify w/b: w^T x+b=-1 ⇔ tn=-1; pt distance to the D: 1/||w||; margin: 2/||w||
W=sum(an*tn*xn) {an:certain coefficient, not equal 0: active pt}{tn:=1/-1}{xn:each pt}{W: scale sum of active pt }带入 D 则表示 only care about what's Support Vector
Min ½ ||w||^2+lambda error(w,b) s.t. tn(w^T xn+b)>=1
Penalize 2 type error a/||w||: margin violation {a in (0,1)} misclassification {a>=1}
Lambda + margin – hard; lambda – margin + soft
Adding features>>higher dimension>>classes linearly separable>>kernel 降维升维>算更快
Kernel: Polynomial (1+x^T z)^d {d: # of feature} Gaussian(RBF) boundary 会变/不一定是 linear
K(xn,xm) compute without mapping: data is linearly separable without specify that mapping

Decision Trees

K features 2^K combinations>>Find Q give most Info>>build tree greedily
Information Gain: max entropy reduction
H(parent)-sum (prob_i * H(child)) H(node)=-sum(prob_i*log2(prob_i))
Gini Index: min impurity
G(parent)-sum(prob_i*G(node)) G(node)=1-sum(prob_i^2)
Purity, majority, thresholding Pruning: cross validation>>avoid overfit

Random Forest

Each tree 只用自己的那组 bootstrapped data; each tree randomly chose subset of features; final decision is majority vote

Unsupervised

Clustering: K-means

1 Assign label of closest prototype 2 refit move each prototype to center of gravity [gradient descent]
Indicator: r_nk{1: x_n assigned to cluster K, 0: otherwise}
Distortion: J=sum[sum[r_nk ||Xn-Mu_k||^2]] >> {0,1} * Distance (Mu_k is prototype of class K)
Minizine Distortion: J 对 Mu_k 求导>>Mu_k=sum(r_nk*Xn)/sum(r_nk)⇔average value in cluster k/# of pts in cluster k 随着循环，J 会减小(converge)
Prototype: sample mean of pts associated with this cluster >> K-means

Hierarchical Clustering

Agglomerative: Backward, start with N clusters, merge, 直到 J 减少
  Single linkage: merge closest(no boundary), Complete: merge farthest members, average: merge average dissimilarity smallest
Divisive: forward, start with 1 cluster, split, as long as J is being reduced
  Accurate, complex

Dimension Reduction: PCA

Scale: make unit same Dimension Projection
{standardization: (Z=Xnew=Xold-Xmean)/sd}: data~Gaussian-like distribution 不影响 outliers
{Normalization: Xnew=(Xold-Xmin)/(Xmax-Xmin) range in (0,1)}: data do not assume distribution 受 outlier 影响
Goal: find subspace W >> look projection with highest sample variance because it's most informative choice
Orthonormal:\ /=# 其他为 0，每行 norm 相加为 1
Max Mu1^T S Mu1 subject to ||Mu1||^2=1: 找最大的 eigenvalue 的 eigenvector
Dimension Projection: u1 = 1st PC; Projection S=X^T*X >> SVD on X = single value decomposition: M=a S V^T