|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Curse of dimensionality  Regression fn  Theoretical MSE  Var(X) = E(X2) - [E(X)]2 | | | | Higher dimension -> need more data  Y = f(X) + . OR f(x) = E(Y|X = x). f = systematic info that X provides about Y. captures errors and other discrepancies  Assume E() = 0. is the irreducible error  MSE() = E = E  is the reducible error. (Can model to be close to f by minimizing MSE). is irreducible | | | | | | | |
| Parametric mtd | | | Assume f is dependent on a finite num of unknown predictors, i.e. f(X) = . Estimate f = Estimate .  Pros: simple and interpretable. Cons: chosen parametric form may not match true unkown f (e.g. quadratic f)  f(X) = for polynomial of def k. Can better approx true regression fn, but tend to overfit | | | | | | | | |
| Non-parametric mtd | | | | Does not make explicit assumptions about f. Impossible to represent f by finite num of params (e.g. only assume f is twice differentiable), E.g. kernel smoothing, spline smoothing. Prone to overfitting | | | | | | | |
| Higher flexibility = More complex model = Lower bias/Higher accuracy but Larger Variance   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | flexibility | Subset Selection, Lasso | Least Squares | Generalized Additive Models, Trees | SVM | Bagging, Boosting | | interpretability | Subset Selection, Lasso | Least Squares | Generalized Additive Models, Trees | Bagging, Boosting | SVM | | | | | | | | | | | | |
| Train MSE  Test MSE  Bias-Variance Trade off  Bias-Variance decomposition  Var(aX + bY) + a2Var(X) + b2Var(Y)  is constant (an obs) | | | | | MSE = , where is estimated from training data (x1, y1), …, (xn, yn)  test MSE = , where (, ) are test data  For an estimator : bias = Bias = . Var = Var  Decompose expected test MSE: | | | | | | |
| Classification  Response Y is qualitative (spam or not; recognise digit, 0-9, in CV) | | | | Y , where is a set of finite elements. Elements of are class labels  A classifier C(X) assigns a class label to a predictor X  Training error rate = , where is the predicted class label, I is the indicator function (1, if not equal)  Test error rate = , where (, ) are test data. Bias-variance trade off also apply here | | | | | | | |
| Inference in High Dimensions | | H0: µ = 0, : µ ≠ 0, where X1, …, Xn , with EX1 = µ = (µ1, …, µp) and Var(X1) = ∑ (p x p matrix)  Assumption: X1, …, Xn iid from a dist. By CLT, ~ N(0, I) (Identity matrix), where  This works only for p n, but not when p n. Alternatively, can test each H0: µk = 0, for each k = 1, …, p  - But this require Bonferroni correction, and with large p, level of significance = is very small, always cannot reject H0 | | | | | | | | | |
| LA Basics | | (XY)T = YTXT | | | | | An n x n matrix is invertible, if an n x n matrix Y, s.t. XY = In. Y is denoted X-1 | | | | |
| An n x n matrix is symmetric if X = XT | | | | | If X is invertible, then XX-1 = X-1X = In | | Let X and Y be n x n invertible matrix. (XY)-1 = Y-1X-1 | | |
| Linear Regressions (LR) | | Yi =  ,  i = 1, …, n | | | | | OR **Y** = **X +** , where **X** = design matrix. Estimator for is | | | | |
| Vector Norms | For **v** = (v1, …, vp) | | | | | norm = length of v | | norm  norm num of non-zero coord in v (e.g = 2) | | | |
| Trace | | For a sq matrix X = ((xij)) , trace = tr(**X**) = | | | | | Generally tr(**ABC**) ≠ tr(**BAC**), except  Cyclic property: tr(**ABC**) = tr(**BCA**) = tr(**CAB**) | | | tr(**A** + **B**) = tr(**A**) + tr(**B**)  tr(**A**) = tr(**A**T) | |
| Matrix differentiation | | Let g() = **A**, where **A** is a n x p matrix not depending on , is a p x 1 vector. Then | | | | | | | | | If g() = **A**, then |
| Let f() = **M**, where **M** is a p x p Symmetric matrix not depending on , is a p x 1 vector. Then | | | | | | | | | |
| Finding OLS (ordinary least squares) soln | | | Q()== (**Y**T - )= **Y**T**Y** - 2(**X**T**Y**) +  = –2(**X**T**Y**) + 2(**X**T**X**)= 0. To get (provided **X**T**X** is invertible) | | | | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Simple LR | | | Y = . is the irreducible error. Assume E() = 0. Thus, E(Y|X) =  Prediction given X = x:  Note that minimize the theoretical prediction MSE. E(Y - a - bX)2 = , among all (a,b)  Given data (x1, y1), …, (xn­, yn), E(Y - a - bX)2 ≈ Q(a,b) (sample version) =  Least squares mtd: estimate by minimizing Q(a,b) over all (a,b)  i.e. solve = 0 and = 0  to get and where , . and aka least squares coefficients  Least squares fit . Residual: | | |
| Estimation accuracy and inference | | | Diff data can yield diff estimates for . To measure how good the estimates are...  bias() = E - and bias() = E - . Estimator is unbiased if expectation = true value. Both and are unbiased  SE of = = . SE of = = , where = var() | | |
| RSS | | | is unknown and can be estimated from data.  If we had sample of the errors, , then could have estimated by sample variance  However, = is unknown  So, estimate by (use n-2 as 2 less degree of freedom as we estimate )  = 0 (assumption). Residual Sum of Squares = RSS = | | |
| Association btw X and Y | | | For LR, 95% CI for is , for is . Test H0: = 0 vs H1: ≠ 0 by checking whether 0 is in 95% CI.  If inside = don't reject H0 = no r/s btw X and Y. If not inside = reject H0 = there is r/s btw X and Y  OR use T = ~ t-dist w n-2 degrees of freedom if is 0.  If p-value < significance level, e.g. 0.05, reject H0, i.e. there is r/s btw X and Y | | |
| Goodness of fit | | | Residual standard error, RSE = . Good fit = smaller RSE. But RSE is in units of Y, so how small is subjective  Standardization: Compare RSS w total sum of squares, TSS = = RSS of (Y = ). TSS/(n-1) = var of y1, ..., yn.  R2 = . If R2 is close to 1, then Y is strongly associated w X, good fit. Measures how well linear model is to baseline model | | |
| Multiple LR | | | Y = . Estimate by minimizing sample MSE Q(a0, …, ap) = , where a = (a0, …, ap)T and **y** = (y1, …, yn)T. **X** is the design matrix =  Estimator for is . (XTX)-1 DNE when p ≥ n  Given a new observation (X1, …, Xp) of predictors, the least squares prediction is  (X1, …, Xp) = (xi1, …, xip) leads to least squares fit . Residual: ei = | | |
| Under some conditions, E= , i.e. is unbiased. The var-covariance matrix of is Var() =  var() is estimated by , where RSS =  = 0 implies Xj has no influence on Y. T statistics, Tj = , where vj is the jth diagonal elem of . If = 0, then Tj ~ tn-p-1 | | |
| No r/s at all = H0 : – (1). More generally, H0 : for q indices j1, ..., jq – (2)  (1) is a special case of (2) w q = p and j1 = 1, ..., jq = p  To test (2), use F-statistic, F = ~ Fq, n-p-1, where RSS0 is the RSS for the reduced LR model that does not contain  RSE = . R2 = , where RSS = , TSS = | | |
| Qualitative predictors / Factors | | | Levels of a factor = possible values of factor (e.g. gender: male/female, weather: Sunny/Cloudy/Rainy)  For factor w 2 levels, create a dummy variable, xi1 =  If factor has ≥ 2 levels, then - 1 dummy variables are needed, e.g. xi1 = , xi2 =  Baseline level = (where all xij = 0) = Rainy in above e.g.  Linear model w weather factor is yi =  = diff in average response/Y btw the baseline (Rainy) and category corresponding to xi1 = 1 (Sunny) | | |
| Interactions & Nonlinear predictors | | | Y = , where  Model is still linear (in regression coeffiicients), interaction terms just treat as additional predictors, (hypo testing still applies)  - Hierarchical principle: if we include interaction in model, should also include main effects, even if p-values not significant  Interpret as the increase in effect of X1 with 1 unit incr in X2 (or vice versa)  If scatter plot is non-linear, can use polynomial regression, yi =  This is also a linear model, just with nonlinearly transformed predictors. *Full* polynomial regression contains interaction term | | |
| Subset / Variable selection | | | Full model containing all predictors has low bias, high var. When p > n, there is no unique least squares coefficient estimate  Removing irrelevant features improves interpretability  Mtds of feature selection: 1) Subset selection 2) Shrinkage 3) Dimension reduction | | |
| Best subset selection | | | Among p variables, find k ≤ p variables that provide best fit to data. Best fit = smallest RSS = largest R2  Naive algo (Best subset selection): 1. Let M- denote null model, containing no predictors. Model simply predicts sample mean for all obs  2. For k = 1,2,...,p: (a) Fit all models that contain exactly k predictors. (b) Pick best among these models and call it Mk.  3. Select single best model from among M0, …, Mp using cross-validated prediction error, Cp(AIC), BIC, or adjusted R2  However, total num of diff models is 2p, which can get very large. | | |
| Forward stepwise selection | | | 1. Let M0 denote null model  2. For k = 0, …, p-1: (a) Consider all p-k models that augment the predictors in Mk w one additional predictor  (b) Choose best among these p-k models and call it Mk+1. Best = smallest RSS = highest R2  3. Select single best model from among M0, …, Mp using cross-validated prediction error, Cp(AIC), BIC, or adjusted R2  For p variables, only fit a total of 1 + p(p+1)/2 which is 2p for large p. But algo is greedy in nature | | |
| Backward stepwise selection | | | 1. Let Mp denote full model, containing all predictors  2. For k = p, p-1, …, 1: (a) Consider all k models that contain all but one of the predictors in Mk, for a total of k-1 predictors  (b) Choose best among these p-k models and call it Mk-1. Best = smallest RSS = highest R2  3. Select single best model from among M0, …, Mp using cross-validated prediction error, Cp(AIC), BIC, or adjusted R2  For p variables, only fit a total of 1 + p(p+1)/2 which is 2p for large p. But algo is greedy in nature | | |
| Standardiza-tion | | | Let **x**j = (x1j, …, xnj)T, for each j = 1, …, n  Centering: Remove mean by **x**j **x**j - , where  After normalization by Mtd 2, = 1  Mtd 2 often used for variable selection algos like Lasso, LARS, ... | Normalization (after centering):  - Mtd 1: , where  - Mtd 2: , where | |
| Forward stepwise selection (alternative way) | | | Start w standardized predictors and model w only intercept, i.e. null model. Denote **y** = (y1, …, yn)T and ,  At intial step, = for all i = 1, …, n  At each step, compute current correlation btw each predictor and the residuals, for all j = 1,…,p  and update by and , where **x**j = (x1j, …, xnj)T  - this add X w highest correlation to residuals, leading to lower RSS.  - Also correlation of this w future residuals will be 0 = this predictor is fully added into the model | | |
| Forward stagewise regression | | | To make forward stepwise selection less greedy, use this  Start w standardized predictors and model w only intercept, i.e. null model. Denote **y** = (y1, …, yn)T and  At each step, compute current correlation btw each predictor and the residuals, for all j = 1,…,p  and update by and , where **x**j = (x1j, …, xnj)T  - Choosing for each step = same as forward stepwise selection  - For small , more steps (more than p) are needed to reach full model  - Variable may be selected in diff steps. If variable is re-picked, then no new variable is added in the step | | |
| Model selection | | R2 = , where RSS = , TSS = . For a fixed k, use R2 to select the best subset of k predictors in best subset selection  RSS or R2 not suitable when models have diff num of predictors. Adding a variable always decr RSS/ incr R2  Adjusted R2 = . Adjusted R2 may decr if reduction in RSS is relatively small after adding a variable. Choose largest adjusted R2 | | | |
| Mallow's Cp = , where k = num of variables in model, = , where RSSfull is RSS of full model w all p predictors  Lack-of-fit term: decr w k. Model complexity term: incr w k. Complexity term penalizes large model. Choose model w smallest Cp | | | |
| Akaike's information criterion for linear models, AIC = , same as Mallow's Cp. Choose smallest AIC  Bayesian information criterion for linear models, BIC = , log(n) places penalty on large values of k. Choose smallest BIC  BIC places heavier penalty than AIC when n ≥ 7 (as log n > 2 when n ≥ 7). Hence BIC often selects a smaller model than AIC | | | |
| Training data: data to estimate model params. Validation data: hold-out data for selecting models. Test data: data to assess model prediction accuracy. training MSE = . validation MSE = . test MSE =  Estimate test MSE by validation MSE on validation dataset. (less data for training, so overestimate test MSE)  - K-fold Cross-validation (CV): Choose int K ≤ n. divide K-fold... calculate validation MSE, MSEr for each rth fold  K-fold CV estimate of test MSE = . Select model w smallest CV MSE  - Leave-one-out-cross-validation (LOOCV): special case of K-fold CV w K = n  LOOCV had approx unbiased estimate of test MSE but relatively large var.  Alternatively, smaller K = larger estimation bias but smaller var. Usually just use K = 5 or K = 10 | | | |
| CV for model selection | | E.g. for Best Subset Selection: For each k = 1, …, p, for each r = 1, … K:  - Hold out rth fold, merge other K-1 folds to from training dataset D-r  - Use D-r to find best model Mk,r that contains exactly k variables  - Calculate validation MSE, MSEr, k of the model Mk,r on the rth fold of data  Calculate CV error of each k = 1, …, p: CV(k) = and select k that minimizes CV(k). Suppose this k is  Use all training data to fit best model containing exactly variables. (Similar procedure for forward/backward stepwise selection) | | | |
| Supple-mentary | | Mathematically, a statistical model is a collection of joint dist/density about data  E.g. LR w iid Gaussian irreducible errors, Yi = . Assume x1, …, xn are fixed and given. Assume ~ N(0, )  Joint dist of data is given by joint density (y1, …, yn) = , where denots density of normal dist w mean and var . A distinct value of (, ) leads to a distinct joint density | | | |
| M1 = {} = {}  Less formal notation: M1 = { Yi = : N(0, )}  Linear model w only intercept term: M0 = {}  M0 is nested inside M1 since M0 M1. If model M is nested inside model N, then RSS of N is ≤ RSS of M (so RSS and R2 cannot be used to select model from nested models) | | | |
| Revisit Forward Stepwise Selection: Suppose we are at the beginning of a step, w current fit  If we were to add variable Xj, then the fit becomes  Residual becomes **e** = **y** - . RSS = = **e**T**e** =  Adding Xj that minimizes RSS is equivalent to adding X­j­ that maximizes | | | |
| Compared to validation dataset approach, K-fold cross validation utilizes data better, but is computationally slower.  Compared to LOOCV, K-fold cross validation has larger bias but smaller variance.  Compared to adjusted R2, AIC or BIC, K-fold cross validation does not need to know the model complexity. | | | |
| Tut | Meaning of Intercept H0 = mean of Y is 0 when all predictors are 0. If p-value < 0, reject H0  Meaning of H0 = for all other fixed Xj, where j ≠ i, Xi has no impact on Y. If p-value < 0, reject H0 | | | | |
| Suppose true r/s btw X and Y is linear.  - Training RSS of cubic regression is lower than training RSS of LR, since more flexible, higher var, lower bias, thus fit training data better  - Test RSS of LR to be lower, as it is the correct model, and has lower var than cubic model  Suppose true r/s btw X and Y is not linear, but we don't know exact r/s  - Training RSS of cubic is lower than training RSS of LR, since more flexible, better fit training data  - Test RSS of cubic model might be lower, depending on how far the true r/s is from linear | | | | |
| pairs(df)  library(fastDummies); df.d <- dummy\_cols(df, select\_columns = factor,  remove\_selected\_columns = TRUE)  cor(df.d)  df$factor <- as.factor(df$factor)  fit <- lm(y~. , data=df); summary(fit)  p-value of F-statistic  p-value of individual coefficient t-test  plot(fit) # In residual plot: Any pattern/trend, Variance of residuals change,  fit <- lm(y~.^2, data = df); help(formula) (a\*b is interpreted as a+b+a:b)  fit <- lm(sqrt(y) ~ . + I(x1^2), data = df) | | | | Scatterplot matrix of all variables  Create dummy variables for factor variables  Create correlation matrix  Convert column to factor  Fit LR  If small = r/s exist btw all X and Y  If small = X have statistically significant r/s to Y  Outliers? Any obs w larger leverage in leverage plot  Fit LR w interaction effects. .^2 = include all 2-way interactions  I() to treat x1^2 as 1 var, instead of x1 + x1:x1 |

|  |  |  |
| --- | --- | --- |
| Model Bias | | MLinear = {}. Nonparametric model, MNP = {f is a differentiable fn}  Let f0 denote true r/s btw mean response and predictors, i.e. Y = f0(X1, …, Xp) +  We consider a model M, and use the data to find a best fn from M as our estimate of f0  If f0 M, then model bias = 0. If f0 M, then there is nonzero model bias.  Let be dist btw any 2 functions f and g. The magnitude of model bias of M = min{}  Note bias = model bias + bias from estimating function ("from model space")  M2 is more flexible than model M1 if M1 M2. Model bias of M2 is no larger than model bias of M1  Estimating has randomness (diff datasets yield diff estimates). Roughly, estimation variance quantifies the spread of  More flexible model has larger spread. (higher variance, lower bias) |
| Ridge Regression | | Ordinary Least Squares (OLS): estimate by minimizing RSS =  MOLS = {}. When p ≈ n, or p is much larger than n, model MOLS has high estimation var  To reduce model flexibility: Subset selection require ≤ k < p where  Subset selection model: MSS = {}  MSS MOLS (for k < p). Subset selection model is less flexible than OLS. Constraint shrinks some , j = 1,…,p to exact 0. |
| Ridge regression use constraint , for some > 0  Ridge regression model: MRidge = {}  is a hyperparameter controlling model flexibility. Larger leads to more flexible model. But MRidge MOLS, i.e. MRidge is less flexible.  Estimate by minimizing penalized RSS = , for some ≥ 0, where  Roughly, one-to-one correspondance btw and . Larger leads to smaller and more flexible models. does not need to be penalized.  To estimate coefficient: 1) Center predictors by , where . 2) Estimate intercept, .  3) Design matrix, **X** = . 4) Stack all centred response values, **y** =  Ridge estimate of , . is always invertible when > 0. When > 0, estimate is unique  Compare w OLS estimate, . When p > n, **X**T**X** is not invertible |
| Ridge Regression Shrinkage Effect | | Conside simple case when n = p, design matrix **X** is the p x p identity matrix, xij = 1 if i = j and xij = 0 otherwise  minimizes penalized RSS = . Solution is . OLS estimate  So when > 0 and yj ≠ 0. i.e. RR coefficient estimates are shrunken proportionally towards 0, relatively to OLS estimates |
| Conside a slightly more general case when n = p, design matrix **X** is a diagonal matrix the p x p .  minimizes penalized RSS = . Solution is . OLS estimate . So  Shrinkage: < 1 when > 0. Suppose = 1, when xj is large (e.g. 100), then shrinkage ≈ 1 , so effect of is small.  When xj is small (e.g. 0.1), shrinkage ≈ 0.01, effect of is big. Scale of predictor has impact on shrinkage effect |
| So, we standardize all predictors to have the same var:  for each i, for each variable Xj: , where and  All now have roughly the same amt of shrinkage after standardization |
| RR Bias-Variance Trade off | | RR model: = {}. Larger leads to more flexible RR modelsA comparison of a graph  Description automatically generated with medium confidence   |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | |  | |  |  |  |  | Estimation var | Model bias | | Model Flexibility | Incr |  |  |  |  |  |  | | Decr |  |  |  |  |  |  |   And 0 ≤ ≤ OR 0 ≤ ≤ 1.  For a given dataset, higher means more flexible model  Solution path: for each , obtain a curve as a fn of  For LHS graph, draw vertical line at = some value, intersection would be RR estimates at that value of |
| CV for hyper-parameter selection  ( for both Ridge and Lasso) | | A graph with numbers and lines  Description automatically generatedSelect by K-fold CV. 1) Determine a range of candidate values for :  2) Randomly partition data into K-fold.  3) For each fold k = 1, … K, leave out the kth fold and use remaining K-1 fold as training data  - For each , = 1, …, m, find or using training data and get validation error on  the kth fold of data  4) Calculate mean and SD of the validation error, ,  Suppose minimize , can choose = OR  1-SE rule: Find smallest /least flexible model within 1 SE, of ,  i.e. select .  On RHS, = smallest CV error. Within one SE of , is least flexible. Choose |
| Lasso Regression | | Constraint: for some real > 0. Larger leads to more flexible model  = {, and }. When > 0, MOLS, i.e. is less flexible  Estimate by minimizing penalized RSS = , for some ≥ 0, where  Roughly, one-to-one correspondance btw and . Larger leads to smaller and more flexible models. does not need to be penalized. |
| Lasso Regression Shrinkage Effect | | Conside simple case when n = p, design matrix **X** is the p x p identity matrix, xij = 1 if i = j and xij = 0 otherwise  minimizes penalized RSS = . Solution is . OLS estimate  So when > 0 and yj ≠ 0. |
| Lasso vs Ridge | | A comparison of a graph  Description automatically generated with medium confidenceRidge estimate,  Ridge shrinks OLS estimate towards 0 proportionally by a factor 1/(1+)  Soft-thresholding: Lasso shrinks OLS estimates towards 0 by an absolute amt /2, and entirely to 0 when  Lasso does variable selection, while ridge does not (since some variables are not selected, i.e. coefficient = 0) |
| Geometric Intuition | | A diagram of a circle and a circle  Description automatically generatedConside case p = 2  Lasso: Minimize RSS , subject to constraint  Ridge: Minimize RSS , subject to constraint  Lasso: defines a diamond (LHS). Ridge: defines a disk (RHS)  Red contours = line where RSS is the same. Smallest RSS is at black dot  For Lasso, closest contour intersect w diamond is at = 0, but for Ridge, intersection would not reach either or = 0  Still possible for Ridge to have beta at 0, but only when OLS estimate at one of the axis (very rare) |
| Scale Equi-variance | | Scale equivariance = multiplying Xj by a constant c ≠ 0 leads to scaling the estimated coefficient by 1/c  OLS: Suppose each Xj is replaced with wjXj. Then the design matrix **XW** with **W** =  . After scaling each variable Xj by wj, the new OLS of is  , i.e. OLS estimates are scale equivariant  Ridge estimate is NOT scale equivariant: , as not scaled by **W**  Lasso estimate also NOT scale equivariant  controls the amt of penalty on each . If 2 variables are on diff scales, the same has diff amt of penalty on  In most cases, same amt of penalty on each is desirable. So, we can standardize all predictors s.t. they have the same variance/scale |
| Supple-mentary | Conside simple case when n = p, design matrix **X** is the p x p identity matrix, xij = 1 if i = j and xij = 0 otherwise. Ridge estimate, . Estimation Bias = E. Estimation var = Var | |
| Same linear model, 1 minimize RSS, while another minimize MAE, since both model same, estimates same = same model bias. Only estimation bias might be diff.  If p = 2 and true r/s is Y = 1 + X1 + 2X2 + . Model bias of ridge when = 10 since 12 + 22 = 5 < 10 (for Lasso, |1| + |2| = 3 < 10). So true model Mridge. Model bias = 0 but still have estimation bias, as long as gamma ≠ 0  Even if estimates for ridge/lasso = estimate for OLS, still DON'T have same estimation var. Estimation var is about how an estimate is sensitive to change of data, and not determined by a particular dataset. Since Ridge/lasso strictly less flexible than ordinary linear model when > 0, estimation var for ridge < estimation var for OLS | |
| always true. 2 cases, if , then . If , then  But not always true that . See RHS plot in 'RR Bias-Variance Trade off'. At x = 1.0 is OLS estimate. For Rating, there is a point where it is greater than OLS estimate for Rating. | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Intuition for LARS | | | In forward stepwise regression: Start w standardized predictors and null model. Denote **y** = (y1, …, yn)T and  A diagram of a graph  Description automatically generatedAt each step, compute current correlation btw each predictor & residuals, j = 1,…,p and update by and , where **x**j = (x1j, …, xnj)T and = small +ve constant  - If is larger in this step, then correlation btw and the residual in the next step is smaller  - If a appropriate value of is used, then there is some other predictor, **x**k, s.t. and **x**k have the same absolute correlation w the residual in the next step  For graph, if choose delta < 1, then next step would still chose to add x3, as x3 still have higher correlation than x2. So appropriate delta to use would be 1 | | | | | |
| Least Angle Regression (LARS) | | | Active set: set of predictors currently in model. Forward stepwise regression: fully include predictor into active set at each step  Forward stagewise regression: Partially include a predictor into the active set  LARS: Only add an appropriate amt of a predictor into the active set | | | | | |
| Start w standardized predictors and null model. Initially, i = 1,…,n. Denote **y** = (y1, …, yn)T and  Set = 0, for j = 1, …, p  1) Find predictor **x**j w largest correlation w residual. Set , where is the largest value s.t. some other predictor **x**k has the same correlation w the current residual as **x**j  2) Move and in the dirn, , defined by their joint least squares coefficient of the current residual on (**x**j, **x**k), until some other predictor **x**l has the same correlation w the current residual.  3) Continue in this way until all p predictors have been added | | | | | |
|  | | | A diagram of a triangle  Description automatically generatedSuppose p = 2. Project **y** onto span{**x**1, **x**2}. Denote projection still by **y**. Set  1) **x**1 is more correlated w the residual **y** - . at .  - Select X1 and augment in dirn of **x**1 to .  s.t. at  So our is always appropriate, compared to forward-stagewise selection ( is a fixed constant) and forward-stepwise selection () | | | | | |
|  | | | A graph of different colored lines  Description automatically generatedSuppose Xj is the predictor to be added in Step k  Let Ak be the active set at the beginning of the kth step. (Includes Xj)  Let be the coefficient vector for the predictors in Ak (k-1 nonzero values, and 0 for Xj)  Let be the design matrix containing only predictors in Ak  E.g. If A2 = {X1, X3}, then original design matrix, **X** = , and =  The dirn for step k is  is updated by  In graph, correlations of diff predictors would become the same when we move in the dirn . Labels at the top indicate which variables enter the active set at each step  Note corr does not always decr for all predictors. | | | | | |
| LARS for Lasso | | | minimizes penalized RSS = , for some ≥ 0 and  A comparison of a graph  Description automatically generated with medium confidenceSoln path: for each , obtain a curve as a fn of or or  A graph of different colored lines  Description automatically generatedAs incr, predictors converge to 0. Unlike for Ridge, predictors would not reach 0 exactly.  Dotted line = predictor added to active set  LARS and Lasso only differ when a solution path reaches 0 | | | | | |
| Start with standardized predictors and null model. Initially, i = 1,…,n. Set and = 0, for j = 1, …, p  1) Find predictor **x**j w largest correlation w residual. Set , where is the largest value s.t. some other predictor **x**k has the same correlation w the current residual as **x**j  2) Move and in the dirn, , defined by their joint least squares coefficient of the current residual on (**x**j, **x**k), until some other predictor **x**l has the same correlation w the current residual.  In 2), if a non-zero coefficient hits 0, then drop its variable from the active set of vars and recompute the current joint least squares dirn.  3) Continue in this way until all p predictors have been added | | | | | |
| Lasso Bias Var Trade-off | | | Lasso model: = {}. Larger leads to more flexible Lasso models  And 0 ≤ ≤ OR 0 ≤ ≤ 1. For a given dataset, higher means more flexible model   |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | |  | |  |  |  |  | Estimation var | Model bias | | Model Flexibility | Incr |  |  |  |  |  |  | | Decr |  |  |  |  |  |  | | | | | | |
| Lasso vs Ridge | | | When all predictors are related to the response, Ridge would be better as likely to give smaller test MSE.  When only some predictors (small num) are related to response, Lasso is better, as it can perform variable selection. | | | | | |
| Variable selection - subset selection | | | *df <- na.omit(df); dim(df)*  *library(leaps)*  *regfit.best <- regsubsets(y ~ ., df, nvmax = k)*  *regfit.fwd <- regsubsets(y ~ ., df, method = "forward")*  *regfit.bwd <- regsubsets(y ~ ., df, method = "backward")*  *summary(regfit.xxx)*  *plot(regfit.xxx, scale = 'r2')*  *coef(regfit.xxx, k)* | | remove na values; see dimension of df  nvmax = k: compute until max of k predictors  Best subset selection (model w smallest RSS). nvmax default to 8 if omitted.  Forward stepwise selection  Backward stepwise selection  1 row = 1 model. \* = predictor in model. (Best: predictor in k-1 predictor in k)  Scale = ranking, also have 'adjr2', 'aic', 'bic'  Get coefficient of fitted model w k predictors | | | |
| CV | # Set K = num of folds. partition data into K folds  *K = 10; folds = sample(1:K, nrow(df), replace=TRUE)*  # compute the CV error, p = num of predictors  *cv.errors = matrix(NA, K, p , dimnames = list(NULL, paste(1:p)))*  *for (j in 1:K) {*  *best.fit = regsubsets(y ~ ., data = df[folds != j, ], nvmax = p)*  *for (i in 1:p) {*  *pred = predict(best.fit, df[folds == j, ], id = i)*  *cv.errors[j, i] = mean( (df$y[folds == j] - pred)^2 )*  *}}* | | | | | # overall CV errors  *mean.cv.errors = apply(cv.errors , 2, mean)*  *mean.cv.errors*  # use all training data to fit the selected model  *regfit.final = regsubsets(y ~ ., data = df, nvmax = optimal\_num\_of\_predictors)*  *coef(regfit.final, optimal\_num\_of\_predictors)* | | |
| Variable selection - shrinkage mtd | | | *library(glmnet)*  *x = model.matrix(y ~ ., df)[ , -1]; y = df$y*  *grid <- 10^seq(5 , -1, length = 100)*  *lasso.mod = glmnet(x, y, alpha = 1, lambda = grid)*  *plot(lasso.mod)* | LASSO regression  # Set up design matrix and response vector. -1 as we don't want intercept  Different lambdas to test for  alpha = 1 for Lasso. Fit Lasso for diff lambda values  Lasso solution path. Numbers on top horizontal axis = num of predictors in model | | | | |
| CV for lambda | | | *cv.out = cv.glmnet(train\_x, train\_y, alpha = 1)*  *plot(cv.out)*  *bestlam = cv.out$lambda.min*  *lasso.mod = glmnet(x, y, alpha = 1)*  # Get coefficient of Lasso model w lambda = bestlam  *predict(lasso.mod, type = "coefficients", s = bestlam)* OR  *lasso.pred=predict(cv.out, test\_x, s="lambda.min")*  *mse = mean(lasso.pred - test\_ y)^2* | | | | A graph with a line and numbers  Description automatically generatedLeft vertical line = min CV error  Right vertical line = CV error < min CV error + 1 SE  *predict(cv.out, test\_x, s="lambda.min", type = 'coefficient')* | |
| LARS | | # design matrix  *x = model.matrix(y ~ ., df)[,-1]; y = df$y*  *library(lars)*  *A diagram of steps with numbers  Description automatically generatedlars(x, y, type = ’lar’)*  #output is plot below  *cv.lars(x, y, type = "lar")*  CV error min at num of steps = 10 (i.e. only add 10 predictors to model) | | | | | | *plot(lars(x, y, type = ’lar’))* # LARS solution path  *plot(lars(x, y, type = ’lasso’))* #LASSO solution pathA graph of different colored lines  Description automatically generated  5th predictor in df is 1st one added |
| Supple-mentary | | | Can generalize Ridge and Lasso to Lq norm:  Estimate by minimizing penalized RSS = , i.e. subjecting to constraint  When q = 1: Lasso, q = 2: Ridge. So q ≤ 1 can do variable selection. | | | | | |
| Elastic Net (in R). For some , estimate by minimizing penalized RSS =  = 0: Ridge. = 1: Lasso. So ≤ 1 can do variable selection  *ridge.fit = glmnet(X, y, alpha = 0); lasso.fit = glmnet(X, y, alpha = 1); elastic.net.fit = glmnet(X, y, alpha = 0.8)* | | | | | |
| A graph of different colored lines  Description automatically generatedPurple line = test MSE, green = estimation var, black = squared bias  All lines become flat at the end as is too large s.t. all estimated coefficients = 0. Then estimation var = 0 and squared bias is max | | | | | |
| Tut | | | |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | |  |  | training RSS | test RSS | Var | Squared Bias | Irreducible error | |  |  | Steadily incr | Decrease first then incr, leads to U shape | Steadily decr | Steadily incr | Remain constant | | Value of controls model flexibility. As , model flexibility . For training RSS, always when model flexibility  When = 0, = 0 for all j. Model fit is inadequate & test RSS is high. When = ∞, model OLS. If p n, model might overfit data & hence exhibits large varability test RSS is high. btw 0 and ∞ provide trade-off btw bias & var of model. Hence test RSS show U shape | | | | | | | | | | | | |
| *library(leaps); regfit.full <- regsubsets(Y~., df, nvmax=10)*  *reg.summary <- summary(regfit.full)*  *plot(1:10, reg.summary$cp, type='b', xlab='k', ylab='Cp')*  *OR plot(regfit.full, scale='Cp')*  *min.k.cp <- which.min(reg.summary$cp)*  *coef(regfit.full, min.k.cp)*  *library(glmnet); grid=10^seq(10, -2, length-100)*  *cv.out=cv.glmnet(df, Y, alpha=1); plot(cv.out)*  *bestlam=cv.out$lambda.min*  *out=glmnet(df, Y, alpha=1, lambda=grid)*  *lasso.coef=predict(out, type='coefficients', s=bestlam)* | | | | # best subset selection, add *method='forward' or 'backward'* for stepwise  # Make plot of Cp # can be $bic, adjr2  #can be scale = 'bic', 'adjr2'  # Find best model by Cp, should use *which.max* for adjr2  # Fit lasso model w CV to find optimal  # Best via CV  # Coefficients w best lambda | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Degrees of Freedom | AIC = . BIC = . k = degrees of freedom (df)/model complexity  In OLS, k = num of coefficients in the model. E.g. if p predictors, k = p + 1 (intercept)  Subset selection, Ridge, Lasso, LARS are all less flexible than OLS.  E.g., Since Ridge don't perform variable selection, all predictors included, i.e. kRidge = p + 1. Will always select OLS  E.g. Assume intercept known. For best subset selection, we try all models w k predictors, then dfsubset > k = df of OLS w k predictors | | | | |
| More general df: Effective df = . Assume predictors are non-random  = fitted response from model. = irreducible error, estimated by . RSSfull = RSS of OLS w all p predictors.  Intuition: Larger model fits training data better = larger covariance btw y1,…,yn and = larger  E.g. large when , for all i, i.e. overfit | | | | |
| E.g. Consider model . Estimate by . Fitted response is  Then | | | | |
| Assume intercept is known; +1 if estimating intercept. OLS w p predictors: = p  Ridge regression: < p if > 0  Best-subset selection w exactly k < p predictors: No closed-form solution, but df > k  LARS: = k for the kth step. Lasso: ≈ num of selected predictors in the model  AIC = . BIC = . But now k = effective df | | | | |
| Workflow | 1. Use training dataset to train a few candidate models 2. Use validation dataset to evaluate/assess the candidate models  3. Choose one model from the candidates 4. Train the chosen model with the training (+ validation) data  5. Evaluate/assess the trained chosen model with the test dataset  Step 1 and 2 may be merged when using CV for model assessment (performance on unseen data) and selection (training/validation data to select 1 model from several candidate models) | | | | |
| CV for model assessment | | 1) Choose int K ≤ n. (e.g. K = 5). 2) Divide into K disjoint folds of roughly equal size.  3) For each r = 1, … K: - Hold out rth fold, merge other K-1 folds to from training dataset D-r  - Use D-r to estimate model parameters  - Calculate validation MSE, MSEr of the model on the rth fold of data  4) K-fold CV estimate of the MSE for model on unseen data = CV MSE =  Assess multiple models simultaneously by performing the cross-validation process for each model  Model selection: Choose the model with the minimal CV MSE | | | |
| CV for hyperparameter selection | | 1) Determine a range of candidate values for : 2) Randomly partition data into K-fold.  3) For each , = 1, …, m:  - For each fold r = 1, … K: - leave out the rth fold and use remaining K-1 fold as training data  - Find or using training data and get validation error on the rth fold of data  4) Calculate CV prediction error,  CV for model assessment equivalent to CV for hyperparameter selection | | | |
| CV for model & hyperparameter | | For each mtd M = {best subset selection, lasso, ridge}: For each candidate value of hyperparameter (or k for subset selection):  for each fold r = 1, ..., K ... then do the same thing. Select (M, ) w minimal CV prediction error | | | |
| Uncertainty | Estimated quantities (e.g. ) have uncertainty. Quantify uncertainty by var of the estimated quantities (e.g. var()) or CI  Low var mean unlikely to change even w substantial change in data  Purpose: 1) Statistical inference: e.g. test H0: = 0. 2) Decision making: e.g. lower bound of CI for average returns > 0 = invest in stock | | | | |
| E.g. | Invest in X and Y. Invest a fraction, into X, (1-) into Y. Choose to min total risk/variance, i.e. min Var(X + (1-)Y)  Soln: , where = var(X), = var(Y), = cov(X, Y)  Since , , unknown, need estimate from data. , ,  Estimate by . IF we have the joint dist of X, Y, then. 1) Can calculate var() mathematically.  2) Approximate var() numerically, i.e. draw samples from joint dist of X, Y, calculate . Repeat B times to get multiple . Then approx var() with the sample var of the multiple = | | | | |
| Bootstrap | From above e.g., use bootstrap to mimic process of drawing samples from join dist of X, Y  Dataset, D := {(X1, Y1), …, (Xn, Yn)} is our sample data from the population (an estimated/empirical version of the population)  Bootstrap: Randomly sample from D with replacement to get (, ), …, (, ) iid. (i.e. uniform dist on D)  D\* := {(, ), …, (, )}. Note \* to indicate bootstrap. Size of D\* = Size of D  Thus, we can obtain B bootstrap datasets, , …, . For each , calculate . Let  Each is a bootstrap version of , and a bootstrap estimate of . Estimate SD of by | | | | |
| Bootstrap in LR | Y = . OLS estimate for is  If predictors are nonrandom/fixed (since we are doing E(Y|X)), then Var() =  If predictors **X** are random, then need use bootstrap, as we dk the joint dist of **X** | | | | |
| **X** = n × (p + 1) design matrix of original data. **y** = response vector Combine **y** and **X** into a data frame D (1st col = **y**)  Bootstrap:  - Draw n rows from D randomly w replacement := bootstrap data frame D\*  - Find the OLS estimate using D\*  For b = 1, …, B, repeat the bootstrap process to obtain the bootstrap OLS estimate vectors , …,  Estimate Var() by the (p + 1) × (p + 1) sample covariance matrix  ,  where | | | *p = 6; n = 50* # num of predictors; sample size  *q = 3; B = 1000* # num of True predictors; num of bootstrap datasets  # True value of regression coefficients  *beta = c(rep(1,q),rep(0,p-q))*  # original data  *X = matrix(rnorm(n\*p),n,p); Y = X %\*% beta + rnorm(n)*  *D = data.frame(Y,X)*  # bootstrap OLS estimate  *beta.boot = matrix(NA,B,p+1)*  *for (b in 1:B) { D.boot = D[sample(1:n,n,replace=T),]*  *beta.boot[b,] = coef(lm(Y~X,data=D.boot)) }*  # calculate bootstrap standard deviation of OLS estimate  *var.hat.beta = var(beta.boot); sqrt(diag(var.hat.beta))* | |
| Bootstrap in Lasso | Lasso estimate of , is obtained by minimizing penalized RSS = for some ≥ 0  Estimate by bootstraping for a fixed value of | | | | |
| **X** = n × (p + 1) design matrix of original data.  Combine **y** and **X** into a data frame D (1st col = **y**)  Bootstrap:  - Draw n rows from D randomly w replacement := bootstrap data frame D\*  - Find the lasso estimate using D\*  For b = 1, …, B, repeat the bootstrap process to obtain the bootstrap OLS estimate vectors , …,  Estimate by the (p + 1) × (p + 1) matrix  ,  where  For e.g. on RHS, only = 1. Other = 0  So SE for other when as more certain that = 0 | | *library(glmnet)*  *#’ Function to bootstrap lasso estimate*  *#’ @param lam lasso hyperparameter*  *#’ @return bootstrap variance (matrix) of lasso estimate*  *boot.lasso = function(X,Y,lam,B=1000) {*  *n = nrow(X); p = ncol(X); D = data.frame(Y,X)*  *beta.boot = matrix(NA,B,p+1)*  *for (b in 1:B) { idx = sample(1:n,n,replace=T)*  *beta.boot[b,] = t(as.matrix(coef(glmnet(D[idx,-1],D[idx,1]),s=lam)))*  *}*  *return(var(beta.boot)) }*  Using same p, n, q, B, beta, X, Y, as above  # bootstrap lasso with lambda=0.2;  *sqrt( diag( boot.lasso(X,Y,lam = 0.2) ) )*  *sqrt( diag( boot.lasso(X,Y,lam = 0.5) ) )* | | |
| Bootstrap Percentile | Bootstrap: Primarily used to obtain SE of an estimate. Also can provide approx CI for a population parameter  Suppose the parameter of interest is , and we obtain B = 1000, , …,  Let a be the 5% quantile of the B values , …, . Let b be the 95% quantile of the B values , …,  [a,b] = 90% bootstrap percentile of θ | | | | |
| Supple-mentary | Bootstrap cannot be used to estimate prediction error, as each bootstrap sample has significant overlap w the original dataset. This will cause the bootstrap to underestimate the true prediction error.  CV is much better, as each of the K validation folds is distinct from the other K-1 folds used for training, i.e. there is no overlap. | | | | |
| If model always yields for all i = 1,…,n (i.e. overfit, training error is 0) | | | | |
| Bootstrap datasets are conditionally indep given original data, but not marginally indep. The (unconditional) joint density of B bootstrap datasets , …, , if existing is where p(D) = joint density of D, and = conditional density | | | | |
| If B is too small, sample var of bootstrap estimates is not stable, and thus less accurate  In theory, B should be as large as possible. However if B is too large, then need more computation time | | | | |
| Bootstrap Pros: Universal, no need to know/estimate dist of data; powerful | | | | Cons: Computationally slow (can be parallelize) |
| To select for lasso using CV (and also w bootstrap): After getting , use CV on to select value of for lasso. Find lasso estimate ... | | | | |
| Tutorial | Var(aX + (1-a)Y) = Cov(aX + (1-a)Y, aX + (1-a)Y) = Cov(aX, aX) + 2Cov(aX, (1-a)Y) + Cov((1-a)Y, (1-a)Y) =  Var is minimized when | | | | |
| Pr(1st bootstrap obs ≠ jth obs from original sample) = (n-1)/n = 1 - 1/n  Pr(jth obs is not in bootstrap sample) = Pr() = Pr() Pr() = (1 - 1/n)n  When n = 5, Pr(jth obs in bootstrap sample) = 1 - (1 - 1/5)5. When n , 1 - (1 - 1/n)n 1 - 1/e | | | | |

|  |  |  |
| --- | --- | --- |
| Principle Components Analysis (PCA) | = . = = Projection of on the green line  A line with a point and a dotted line  Description automatically generated with medium confidenceWant to minimize the sum of squared "residuals"/approximation errors, i.e.  min , subject to are on the same straight line  Green line pass through avg, and is determined by dirn w = 1  , where denotes inner product. is shared by projections  Find optimal by minimizing sum of squared "residuals" = , subject to = 1  Minimizer, = first principal component. = first principal component score. Then | |
| First Principal Component | minimizes , where = 1  Hence, equivalent to maximizing , where = 1  . So for any  = Variance of , which is maximise by setting  Residual variance = Total variance – Explained variance | |
| Note . Note Eigenvector **v** and eigenvalue :  So , where = sample covariance of Xi  Find to maximize  where = 1  Fact: has p eigenvalues, ordered by , w associated eigenvectors **v**1, …, **v**p and for j = 1, …, p  Fact: Maximum value of  where = 1, is and maximizer is  So first principal component: | |
| Algo: Find-Leading-Eigenvector(). So = Find-Leading-Eigenvector()  1) Find sample covariance, . 2) Find largest eigenvalue and its eigenvector . 3) Output | |
| More Principal Components | Approximate by . The residual is  Pretend are observed "data". Apply PCA to by finding to maximize , where = 1  Then = Find-Leading-Eigenvector() = second principal component. Scores = 2nd principal component score  Residual of "residuals" : , and can keep repeating process to find more principal components.  Note 2 approximations, and  Fact: . So second approximation is improved by adding | |
| Algo: PCA-SEQ(k, ), where k is num of desired principal components. Output: and for j = 1, …, k  1) Set for i = 1, …, n.  2) For step j = 1, …, k: { = Find-Leading-Eigenvector() ; ; } | |
| , with p eigenvalues and their associated eigenvectors.  Theorem: Eigenvector is exactly the jth principal component, , for j = 1, …, p  Algo: PCA-Eigen(). Output , , and for j = 1, …, k  1) Compute . 2) Obtain eigen-pair (, ), …, (, ) of and set for j = 1, …, p  3) If needed, compute principal component scores: | |
| PCA properties | Principal components are orthonormal: for j ≠ k and j = 1, …, p  Principal components scores are centered: j = 1, …, p  Principal component scores are uncorrelated: for j ≠ k  Eigenvalues are variance: var() = j = 1, …, p  Total Variance of data:  Represent by its principal components: | |
| Fraction of Variance Explained | If , for k ≤ p, then variance of is  is the variance explained by the first k principal components  Fraction of variance explained by first k principal components =  0 < FVEk ≤ 1 for 1 ≤ k ≤ p. FVEk increases with k. Trade-off btw approximation quality and dimension  FVEk measures how well we appox data/fraction of info retained by using first k principal components | |
| Optimality | Can we find other orthonormal s.t , gives better approximation?  No. Because ≤ . i.e. first k principal components are optimal and give the smallest approximation error | |
| Principal Components Regression (PCR) | Dimension reduction: for k < p, . Each approximation is determined by ()  PCR: - Build linear regression on (), …, ()  - Predictors are . Num of predictors = k. k is a hyperparameter controlling model flexibility of PCR  PCR works well when first few principal components are sufficient to capture the most variation in predictors and the r/s w the response. If many predictors are needed, then normal LASSO might perform better, w lower test MSE than PCR | |
| R code  PCA | PCR | *library(ISLR)*  *df <- na.omit(df)*  *X <- model.matrix(Y ~ . , data = df)*  *pca <- prcomp(X[,-1])*  *pca* | *library(pls)*  *pcr.fit = pcr(Y ~ . , data = df, scale = TRUE, validation = "CV")*  # scale predictors to same scale  *summary(pcr.fit)*  *validationplot(pcr.fit, val.type = "MSEP")* |
| Supplemen-tary | Eigen/spectral-decomposition. .  , are eignevalues w associated orthonormal eigenvectors **v**1, …, **v**p , i.e.  is maximized by and the maximal value is .  Proof: Let , then (since orthogonal) ≤ | |
| Assume **X** has been centered, and is the covariance matrix of **X**  Eigen-decomposition: , where = diag() and is a unitary matrix, i.e.  Eigen-decomposition is unique if we require . The jth principal component is the jth col of **Q**  Singular value Decomposition (SVD): , where , are orthogonal matrices, is a rectangular diagonal matrix  Eigen-decomposition of via SVD,  Comparing with above eigen-decomposition, ,  We can avoid computing the covariance matrix of centered **X**  Principal-components score: is the (i, j)-entry of  SVD gives rise to both principal components and scores. There are at most min{n, p} nonzero eigenvalues | |
| is in the form of for some coefficients a1, …, ap, i.e. is a LC of the p original predictors  Since , with for j = 1, …, p | |
| First principle component is not unique. can also be regarded as the first principle component | |
| Both PCR and Lasso can reduce the dimension of predictors thus model flexibility. Below are the differences between them.  PCR: - Create k < p predictors, each new predictor being a linear combination of the original p predictors;  - Perform dimension reduction in an unsupervised way, i.e., without using the response values.  Lasso: - Select k < p predictors, each being one of the original predictors;  - Dimension reduction is done in a supervised way, i.e., by using response values. | |
| Tutorial | *pcr.pred = predict(pcr.fit, test, ncomp = k)*  *mean((pcr.pred - test$Y)^2)* | |
|  | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Problem 1 (20 points)** | Suppose we estimate the regression coefficients in a linear regression model by minimizing    subject to  for a tuning parameter s.  For parts (a) through (d), indicate whether the statement is correct. Justify your answer. | (a) As we s from 0, the training RSS will initially, and then start to to exhibit a U shape.  **Answer.** False. As s , model flexibility , and training RSS always as the model flexibility . | | |
| (b) As we s from 0, the test RSS will first and then , exhibiting a U shape.  **Answer.** True. As s from 0, the test RSS will first and then , and thus exhibits a U shape. | | |
| (c) As we s from 0, the estimation variance will decrease. **Answer.** True. As s from 0, the model flexibility , so that the estimation variance . | | |
| (d) a constant c > 0 s.t, the training RSS remains unchanged for all s satisfying s ≤ c.  **Answer.** True. The most flexible model in the above is the ordinary linear regression model, which corresponds to s = c :=. When s ≤ c, the OLS estimate satisfies the constraint so, soln is the OLS in this case, and the training RSS remains unchanged for s ≤ c. | | |
| **Problem 2 (10 points)** | Suppose there are p predictors. We consider two models: The ordinary linear regression model with all predictors, and the principal components regression with k < p principal components. | (a) Which regression model has less model flexibility? Justify your answer.  **Answer.** PCR has less model flexibility. Suppose there are p predictors and k principal components are used in the PCR. Because PCR with k = p is equivalent to OLS, while in general, k < p, so that PCR has less flexibility. | | |
| (b) Does principal components regression perform variable selection? Justify your answer.  **Answer.** No, because the PC scores are linear combination of all predictors, so that the PCR generally includes all predictors. | | |
| **Problem 3 (15 points)** | For a dataset with p = 5 predictors X1, X2, X3, X4, X5 and n = 100 observations. Plot gives the LASSO solution paths, where the x-axis is the ratio and the y-axis is the standardized regression coefficients; = lasso estimate of the coefficient vector, = OLS estimate, and A line graph with different colored lines  Description automatically generated | | | (a) List the predictors according to the order in which they are added into the full model for lasso regression.  **Answer.** X5, X4, X2, X1, X3 |
| (b) Lasso estimate (, . . . , ) minimize the penalized RSS  Let be the smallest value of such that = 0. Suppose is the lasso estimate of with = , and suppose is the ordinary least squares estimate of . Is it true that || < || for this dataset?  **Answer.** No. When = , the lasso estimate for is the intersection of vertical line marked by 4 and the solution path for . The OLS estimate of is the intersection of the vertical line marked by 7 and the solution path for . In this case, the OLS estimate of is in magnitude smaller than the lasso estimate with = . |
| (c) How does lasso regression perform variable selection? **Answer.** Lasso regression performs variable selection by shrinking some estimated coefficients to zero. |
| **Problem 4 (25 points)** | dat <- Hitters[,**c**('Salary','Hits', 'HmRun', 'Runs', 'RBI', 'Walks', 'Years')]  **names**(dat); dat <- **na.omit**(dat); **set.seed**(4211)  *# perform best subset selection*  regfit <- **regsubsets**(Salary**~**., dat) reg.summary <- **summary**(regfit)  **plot**(regfit, scale='bic')  **coef**(regfit, **which.min**(reg.summary**$**bic))  *# perform lasso regression*  grid=10**ˆseq**(2,**-**3,length=20) cv.lasso <- **cv.glmnet**(**as.matrix**(dat[,**-**1]), dat[,1], alpha=1, lambda=grid)  lasso.fit <- **glmnet**(**as.matrix**(dat[,**-**1]), dat[,1], alpha=1, lambda=grid)  **predict**(lasso.fit,type="coefficients", s=cv.lasso**$**lambda.min) | | ## [1] "Salary" "Hits" "HmRun" "Runs" "RBI" "Walks" "Years"  A graph with black and white squares  Description automatically generated  ## (Intercept) -228.959  Hits 3.054 Walks 4.468 Years 34.442  ## 1 x 7 sparse Matrix of class "dgCMatrix"   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | (Intercept) | Hits | HmRun | Runs | RBI | Walks | Years | | -204.293 | 2.218 |  |  | 1.884 | 4.043 | 32.517 | | |
| (a) What is the model selected by best subset selection with BIC. What are the estimated values for the coefficients of this selected model?  **Answer.** Model: Salary = + × Hits + Walks + × Years + ε. Estimated coefficients for (, , , ) are (−228.96, 3.05, 4.47, 34.44). | | | |
| (b) What is the model selected by lasso with CV for selecting the tuning parameter. Estimated values for the coefficients of this model?  **Answer.** The model is Salary = + × Hits + Walks + × Years + × RBI + ε.  Estimated coefficients for (, , , , ) are (−204.29,2.22,4.04,32.52,1.88). | | | |
| (c) Briefly explain why ridge regression does not perform variable selection.  **Answer.** Ridge regression uses l2 norm penalty which does not shrink some coefficients to exactly zero. | | | |
| (d) Identify a major problem of the following code for assessing the performance of ridge regression on unseen data via cross validation.  *cv.ridge <-* ***cv.glmnet****(****as.matrix****(dat[,****-****1]), dat[,1], alpha=0, lambda=grid)*  *lambda <- cv.ridge****$****lambda.min; K <- 5; cv.err <- 0; folds <-* ***sample****(1****:****K,****nrow****(dat),replace=T)*  ***for****(r* ***in*** *1****:****K) {*  *ridge.fit <-* ***glmnet****(****as.matrix****(dat[folds****!=****r,****-****1]), dat[folds****!=****r,1], alpha=0, lambda=lamb*  *pred <-* ***predict****(ridge.fit, newx=****as.matrix****(dat[folds****==****r,****-****1]), s=lambda)  cv.err <- cv.err* ***+ sum****((pred****-****dat[folds****==****r,1])****ˆ****2)*  *}*  **Answer.** The selection of hyperparameter, λ, is part of the training, and should be nested inside the cross-validation process. | | | |
| (e) If the variable Hits were not selected in lasso regression, can we still use bootstrap to estimate the variance of its associated lasso coefficient? If not, justify your answer; if yes, briefly describe how to perform bootstrap properly.  **Answer.** Yes, with the following bootstrap procedure: For b = 1, . . . , B,  • we generate a bootstrap dataset by resampling with replacement,  • use cross validation on to select , • produce the bootstrap lasso estimate by using and .  Finally, estimate the variance of the lasso coefficient by the sample variance of the bootstrap estimates , . . . , . | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| PCA | Assume EX = 0 below. The first principal component maximizes among all s.t.  is covariance matrix of predictors. All principal components are eigenvectors of  , where X = (X1, …, Xp) is the column vector of predictors.  Let = . Then maximizes among all s.t. | | |
| Partial Least Squares | Response in not involved in PCA. Can use Y to guide constructing linear combinations of predictors for regression  Intuition: Choose a dirn to maximize , but also want to account for X to some extent  Partial least square: Find that maximizes s.t.  = first PLS dirn. kth PLS dirn maximizes s.t. and for j = 1, …, k-1  Partial least squares seeks dirn s that have high variance and have high correlation w response  PLS regression: Build linear regression model on scores | | |
| *library(pls); library(ISLR)* | *pcr.fit = pcr(Y~., data = df, scale=T); summary(pcr.fit)* | *pls.fit = plsr(Y~., data=df, scale=T); summary(pls.fit)* |
| Sliced Inverse Regression (SIR) (not tested) | Main intuition: identifying relevant dirns (slices) of the predictor variables that have strong r/s with the response. It aims to find linear combinations of predictors (slices) that maximize the ratio of the between-group variance to within-group variance in the response.  Let for j = 1, …, q and i = 1, …, n  SIR tries to find all coefficients from the model , where f is an unknown fn of q arguments  Dimension reduction when q p. q is often small so that f can be well estimated  Let for j = 1, …, q. Let . Let . Then  All lies in the q-dimensional linear subspace spanned by . This subspace = effective dimension reduction space (EDR-space) | | |
| Comparison | Similarity: All try to reduce dimensions via constructing derived predictors that are linear combinations of original predictors  Differences: –  PCA does not account for the response  –  PLS maximizes both (1) the correlation btw the response and the derived predictors and (2) the variances of the derived predictors  –  SIR finds the EDR-space | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Classification | | Identify which set of categories/class labels an observation belongs to. Class label = response, object to be classified = predictor  Classifier: A set of rules/function that assigns an observation to a category  LR not suitable. E.g. if Y = 1 for drug overdose, 2 for stroke, 3 for seizure.  This falsely imply an ordering on outcomes. Give wrong impression that diff btw stroke and drug overdose = diff btw stroke and seizure  E.g. Binary classification also cannot use LR. Y = 0 for stroke, 1 for seizure.  Although is an estimate of Pr(seizure|X1, …, Xp). some predicted probabilities would be negative (i.e. impossible) | | |
| Logistic Regression | | E.g. Pr(default = Yes|balance). Can write as p(balance). Y = 1 if default = Yes; 0 if default = No.  Classifier: Predict default = Yes if p(balance) > 0.5 (threshold can be changed)  Logistic function/model: p(X) = . Increasing X by 1 unit changes log odds by  Odds = . Log odds/logit = . | | |
| Multiple logistic regression | | Predictors X = (X1, …, Xp). Logistic model: p(X) = . And  Classify X = x to Y = 1 if p(X) > 0.5 and to Y = 0 otherwise. p(X) > 0.5 is equivalent to > 0 | | |
| Maximum Likelihood Estimator (MLE) | | Data (X1, Y1), …, (Xn, Yn), where X1, …, Xn are fixed covariates  Each Yi is independently sampled from a pdf/pmf f(y; Xi, ). is a parameter/vector of parameters  Likelihood fn of is . Negative log-likelihood fn is  MLE of is  Yi ~ Bernoulli(p(Xi)) and has pmf f(y; Xi, ) =  1 predictor: Negative log-likelihood fn of is  Multiple predictors: ,  Numeric optimization mtd need to find minimizer of and | | |
| Derivation | f(y; Xi, ) = | | | |
| Penalized Maximum Likelihood | | When p is large, MLE estimates have large var as model too flexible. Can reduce flexibility using constraint  i.e. Minimize penalized negative log-likelihood w ridge-type constraint:  OR Minimize penalized negative log-likelihood w lasso-type constraint:  OR Elastic net penalty: Minimize  is a hyperparameter that controls model flexibility. Larger = smaller model flexibility  controls relative weights of ridge type and lasso type penalties | | |
| *logistic.fit = glm(Y~., data.frame(Y, X), family=binomial(link="logit"))*  *summary(logistic.fit)*  *cv.glmnet.fit = cv.glmnet(X, as.factor(Y), alpha=1, family='binomial')*  *predict(cv.glmnet.fit, type='coefficients', s=cv.glmnet.fit$lambda.min)* | Fit ordinary logistic regression (w/o penalty)  Logistic regression w penalty. Need cv to choose lambda  Get coefficients | |
| Logistic Classifier | | Logistic regression w p predictors **X**i = (Xi1, …, Xip) is  Suppose is an estimate of . Estimated probability  For a threshold (e.g. = 0.5), logistic classifier | | |
| Bayes Classifier | | Consider K classes w class labels 1, …, K. Note Bayes' theorem  Prior probability: . Posterior probability: . Write  Pr(X = x) =  Bayes classifier: Equivalently . Classify X = x to the kth class if k maximizes  For a fixed x, denominator is same for all k. So just need maximize or maximize log | | |
| Linear Discriminant Analysis (LDA) when p = 1 | | Assume each to be Gaussian, i.e. . and for k = 1, …, K are unknown parameters  Assumption: , and denote this common variance by  The k maximizing also maximizes . = discriminant fn and is linear in terms of x  LDA classifier: Classify X = x to the kth class if k maximizes | | |
| Data: (Xi, Yi) for i = 1, …, n. Estimate by (k = class label, = num of obs in kth class)  Estimate common by . Estimate = Pr(Y = k) by the proportion  Then . | | |
| LDA w p > 1 | | p predictors: X = (X1, …, Xp). Assume a multivariate Gaussian dist for , i.e. , where **x** = (x1, …, xp) is a col vector, is a col vector and is a p x p covariance matrix  is determinant of and is inverse of  f1, …, fK have same covariance but diff mean vectors  Linear discriminant fn . Classify X = **x** to the kth class if k maximizes | | |
| Data: (**X**i, Yi) for i = 1, …, n. Estimate by (n = n1 + … + nk, = num of obs in kth class)  Estimate common by . Estimate = Pr(Y = k) by the proportion  Then . | | |
| Decision Boundary | | Bayes classifier: Classify X = x to the class for which is largest; has the lowest possible error rate out of all classifiers  E.g. if K = 2 and , then makes  Bayes decision boundary is the point x s.t.  A graph of a function  Description automatically generated with medium confidence  When p = 2, K = 3, and  Dashed lines = Bayes decision boundaries; formed by points **x** s.t.  Solid lines = decision boundaries of LDA classifier w estimated model parameters  LDA generally does not perform well in high dimensions | | |
| Error Rate | | Suppose C is a K-classes classifier trained on data (**X**i, Yi), i.e. C(**X**i) {1, …, K} for i = 1, …, n  Let I{C(**X**i) ≠ Yi} = 1 if C(**X**i) ≠ Yi and 0 otherwise. Misclassification: C(**X**i) ≠ Yi  Training error rate is proportion of misclassified observations =  Test error rate on test data w m observations (), …, () is . Test data should not be used for building C  Null classifier Cnull(**X**i) = 0 for all **X**i. Can compare error rate of classifier with null classifier  A graph of a function  Description automatically generated with medium confidenceSensitivity: Percentage of true 'class 1' that are correctly identified  Specificity: Percentage of true 'class 0' that are correctly identified  In binary classification, to improve sensitivity, we can use lower threshold = (so more predicted as 'class 1')  To improve specificity, use higher threshold  Value of controls trade off btw sensitivity and specificity  Black solid line = overall error rate  Blue dashed line = 1 - sensitivity = % of true 'class 1' incorrectly identified  Orange dotted line = 1 - specificity = % of true 'class 0' incorrectly identified | | |
|  | | *measure = function(predY , testY) { m = rep(NA,4)  m[1] = mean(predY==testY); m[2] = mean(predY!=testY)  m[3] = mean(predY[testY==1] == testY[testY==1])*  *m[4] = mean(predY[testY==0] == testY[testY==0])*  *m = matrix(m,1,4)  colnames(m) = c(’accuracy’,’error␣rate’, ’sensitivity’, ’specificity’)*  *return(m) }*  *logistic = function(X,beta) {*  *pX = apply( X,1,function(x) { s = beta[1] + sum(beta[-1]\*x) } )*  *return(pX) }*  *1) logistic.fit = glm(Y~.,data.frame(Y,X), family=binomial(link="logit"))*  *2) library(glmnet); cv.glmnet.fit = cv.glmnet(X, as.factor(Y), alpha = 0, family='binomial')*  *predY = round(predict(MODEL.fit,newdata=data.frame(testX),type=’response’))*  *3) library(MASS); lda.fit <- lda(Y~., data.frame(Y,X))*  *predY <- predict(lda.fit, data.frame(testX))$class measure(predY ,testY)* | | Calculate classification measures  predY = predicted Y  testY = Y in test data  Compute probability P(Y = 1|X = x)  X = design matrix. beta = coefficients.  return array of probabilities  Logistic classifier  Logistic classifier w ridge-type penalty. Use alpha = 1 for lasso-type penalty  Predictions for 1) and 2)  LDA classifier  Predictions for 3)  Calculate error metrics |
| Supple-mentary | | LDA classifier: Classify X = x to the kth class if k maximizes OR maximize OR maximize log  log  . First {} dependent on k, 2nd {} not | | |
| If we use Linear Regression, the estimated probability may be negative or greater than 1 | | |
| Interpretation of coefficient : When other covariates are fixed, increasing by 1 changes log odds by | | |
| Logistic: For X = x, we classify Y = 1 when . Since . And  We can equivalently do classification based on whether | | |
| Overall error rate does not always decrease as threshold increase | | |
| LDA generally DONT perform well in high dimensions: partially because it's diff to accurately estimate the covariance matrix Σ when . Also, in high dimensions, approximating by the multivariate Gaussian distribution may not be adequate. | | |

|  |  |  |
| --- | --- | --- |
| Decision Tree | A graph with colored dots  Description automatically generatedPartition of predictor space w straight line decision boundaries  For trees, start from root node, split by some criterion, go to internal nodes, then at the end are leaves = terminal nodes  Classification trees: for classification.  Regression tree: Y = . Although Y is cts, predicted Y is discrete  Trees are simple and have better interpretability. Meaning prediction accuracy not so good  But combining large num of tress can result in huge improvements in prediction accuracy, at expense of some loss in interpretation | |
| Regression Tree | Suppose there are p predictors, X1, …, Xp. Predictor space = set of possible values for X1, …, Xp  Decision tree divides the predictor space into q non-overlapping regions R1, …, Rq. The regions are hyper-rectangles/boxes in the  Given a fixed value of q, find boxes R1, …, Rq that minimizes the RSS , where is the mean response for training observations within the jth box and nj = num of observations in Rj | |
| Greedy Approach: Computationally exp to consider every possible partition. So use top-down greedy approach of recursive binary splitting. 1) Select Xj and cutpoint s, s.t. splitting the predictor space into {X|Xj < s} and {X|Xj ≥ s} leads to greatest possible reduction in RSS  2) Repeat above 1) recursively on each of {X|Xj < s} and {X|Xj ≥ s}  3) No further split for a region if a stopping criterion is reached, e.g. when region contains ≤ = 5 observations  Greedy: because at each step, best split is made, rather than looking ahead and picking split that will lead to better tree in future step | |
| If some region in predictor space is not a rectangle = partition not from recursive binary splitting | |
| Classification Tree | | Regression tree, predicted response for is mean response of observations in Rj. Classification tree: prediction is via majority vote  RSS or MSE not suitable for classification tree |
| Use classification error rate = fraction of training obs in a region not belonging to the mode class in that region = E = , where K = num of classes, = proportion of training obs in jth region that are from the kth class |
| However, classification error rate not sufficiently sensitive for tree growing. Gini index & Deviance are preferred  1) Gini index for the jth terminal node = . Total Gini index of a tree w q terminal nodes =  - measure of total variance across the K classes in a terminal node. - is small if all of the are close to 0 or 1  - measure of purity: small value = a node contains mostly obs from a single class  2) Deviance/cross-entropy (of jth terminal node) = , where we assume if  Total deviance of a tree w q terminal nodes = |
| 1) Select Xj and cutpoint s, s.t. splitting the predictor space into {X|Xj < s} and {X|Xj ≥ s} leads to greatest possible reduction in deviance or Gini index. Step 2) and 3) same as Regression tree |
| Tree vs linear classifiers | A graph of a graph of a graph  Description automatically generated with medium confidenceIf there is clear evidence of non-linear r/s, then tree more suitable. If linear r/s, linear classifier (logistic classifier) better  A graph of a line and a line  Description automatically generated with medium confidenceLHS: linear classifier decision boundary  RHS: tree decision boundary | |
| Prune Trees | Greedy tree-building approach likely to overfit data, especially when is small  Consider tree size = num of terminal nodes. Larger tree = larger model flexibility = larger var = smaller bias  Reduce model flexibility by limiting num of splits based on below 2 strategies  1) Start w empty tree. Grow tree only if decr in RSS/deviance/Gini index due to each split > some (high) threshold  - But, this strategy is too short-sighted: a seemingly worthless split early on in the tree might be followed by a split that leads to a large reduction in RSS/deviance/Gini index later on  2) Grow a full or very large tree T0, and then prune it back to obtain a smaller subtree | |
| Let |T| = tree size. Given and any subtree , consider the penalized RSS =  For classification tree, consider the penalized Gini index or deviance = or  When , trees of large size get penalized. is a hyperparameter making trade off btw subtree's complexity and its fit to training data | |
| GROW(): Use recursive binary splitting to grow a large tree T0 on data , stopping only when each terminal node ≤ obs  PREDICT(**x**, T): Locate terminal node Rj s.t. **x** . Output - mean response of obs in Rj OR - majority class label in Rj as prediction for **x**  PRUNE(T0, ): Find subtree T that minimizes the penalized RSS or penalized Gini index/deviance | |
| For each candidate value of , for each k = 1,...,K:  1) Hold the kth fold of data, and merge the other folds into a single training data  2) Construct T0 = Grow( , ) . 3) Prune the tree to obtain T = Prune(T0, )  4) For each observation (**x**i, Yi) in the kth fold of data, use the tree T from Step 3 to predict response = T(**x**i) = Predict(**x**i, T)  5) Calculate the CV MSE = OR the CV classification error =  6) Finally, the CV error of is | |
| Tree algo | 1) Determine threshold . 2) Use CV to select . Suppose selected value is  3) Use to construct final decision tree by, - growing tree T0 = GROW(all training data, ), - pruning T0 to obtain final tree = PRUNE(T0, ) | |
| Bagging | 1) Oracle Aggregation. Let T be a decision tree built based on a dataset of n obs. For a fixed obs **x**, T(**x**) is the predicted response  Suppose E(Y|**x**) = f(**x**), where f is true r/s btw predictor and response. Then expected regression MSE = MSE(T(**x**)) = E{T(**x**) - f(**x**)}2 =  [ET(**x**) - f(**x**)]2 + Var(T(**x**)) = squared bias + estimation var  If we have B indep datasets, for each of which we build a tree Tb, b = 1, …, B, then we can get a new prediction  Note that T1, …, TB are iid, like T. ETb(**x**) = ET(**x**) for all b = 1, …, B. So EToracle(**x**) = E = ET(**x**)  Bias of Toracle(**x**) = Toracle(**x**) - f(**x**) = ET(**x**) - f(**x**), which is the same bias of a single tree  Var(Tb(**x**)) = Var(T(**x**)) for b = 1, …, B. So estimation var of Toracle(**x**) = Var(Toracle(**x**)) = Var = Var(T(**x**))  So oracle aggression keeps same bias but reduces estimation variation and MSE compared to a single tree | |
| 2) Use bootstrap aggregation / bagging to mimic oracle aggregation  Generate B bootstrapped datasets via indep resampling, for each we build a bootstrap decision tree  Given an obs **x**, aggregate all trees to predict response: Tbag(**x**) = OR Tbag(**x**) = majority vote among , …,  Let T\* be any of the bootstrapped trees,  Bias: E(**x**) = ET\*(**x**). So ETbag(**x**) = ET\*(**x**). Variance: Var(Tbag(**x**)) ≤ Var(T\*(**x**)).  So bagging generally has smaller variance and hence smaller MSE than a single tree T\* while still having same bias | |
| Out-of-Bag error estimation. - On average, each bagged tree only use around 2/3 of the original obs  For a given bootstrap tree T\*, the obs not used by T\* aka out-of-bag (OOB) obs  For the ith obs (**x**i, Yi), there are roughly B/3 bootstrap trees that don't make use of (**x**i, Yi)  Let = { that don't make use of (**x**i, Yi)}  OOB estimate for **x**i is the bagging of trees in . Regression: use mean of predictions. Classification: majority vote of predictions  OOB error = for regression and for classification  OOB error is an estimate of the test error of a bagged model | |
| Random Forest (RF) | Bootstrapped datasets have substantial overlap, so bootstrap trees are highly correlated, and so reduction in estimation var is limited  RF: When building decision trees that are bagged later, each time a split in a tree is considered, a random selection of m predictors is chosen as split candidates from the full set of p predictors. The split is allowed to use only 1 of these m predictors  - An indep selection of m predictors is taken at each split. Typically m ≈ | |
| - RF decorrelate the trees. Since split for each tree select from random m predictors, top split likely diff for each trees  - Suppose there is 1 very strong predictor. In collection of bagged trees, most or all of the trees will use this very strong predictor in the top split. Hence, almost all of the bagged trees look quite similar to each other, i.e are highly correlated  - Averaging many highly correlated quantities does not lead to as large of a reduction in variance as averaging many uncorrelated quantities  - Bagging will not lead to a substantial reduction in variance over a single tree in this setting | |
| Regression: sum total amt of RSS decreased due to splits for a given predictor, averaged over all B trees. Large value = impt predictor.  Classification: sum total amt that the Gini index or deviance is decreased by splits for a given predictor, averaged over all B trees. E.g. If B = 3, and X1 is used to split an internal node in two trees. Suppose  – for one tree, before split involving X1, the total training RSS of the tree is 10, and after split, the total training RSS is 8,  – for the other tree, before split involving X1, the total training RSS of the tree is 7, and after split, the total training RSS is 6.  Then the RSS decreased due to splits over X1, averaged over all B = 3 trees, is [(10 − 8) + (7 − 6)]/3 = 1  If the RSS decreased due to splits over X2 averaged over all trees = 3, then X2 is more important than X1 | |
| Boosting | Fit a seq of small trees, each built upon the residuals of previous trees.  By fitting small trees to the residuals, boosting approach slowly improves the estimation/prediction in areas where it don't perform well | |
| Fix hyperparameters d and . Set T(**x**) = 0 and ri = Yi for all i in training set  For b = 1, …, B, repeat: { - Fit a tree Tb w d splits to the training data (**x**1, r1), …, (**x**n, rn)  - Update T by adding in a shrunken version of the new tree, i.e. T(**x**) T(**x**) + Tb(**x**)  - Update residuals, ri ri - Tb(**xi**)}  Output boosted model, T(**x**) =  B = num of trees. If B is too large = overfit. Use CV to select B  = shrinkage param to control learning rate. Typically 0.01 or 0.001. Very small require large B to get good performance  d = num of splits in each tree. Control complexity of boosted model. d = 1 or 2 ofter works well. d is also the interaction depth, controlling the the interaction order of the boosted model. d = 1 (no interaction). d = 2 (interaction btw 1st and 2nd selection var) | |
| Trees are built sequentially in boosting, but independently/parallelly in bagging and RF | |
| Supple-mentary | Tbag(**x**) = . Let D = training data. Suppress **x** from T(**x**) for ease of notation. Conditional on D, T1, …, TB are iid so  E(Tbag|D) = E(T\*|D). And Var(Tbag|D) = Var = Var(T\*|D). Var(X) = E(Var(X|Y)) + Var(E(X|Y))  Var(Tbag) = E{Var(Tbag|D)} + Var{E[Tbag|D]} = E{Var(T\*|D)} + Var{E[T\*|D]} = E{Var(T\*|D)} + { E{Var(T\*|D)} + Var{E[T\*|D]} } =  E{Var(T\*|D)} + Var(T\*) { Var(T\*) - E{Var(T\*|D)} as B ∞ } ≤ Var(T\*) | |
| In decision tree, num of regions, q, determines the model flexibility. As q incr, decision tree becomes more flexible, bias decr, var incr, and MSE decr first and then incr to form a U shape | |
| Gini index/deviance is minimized when the obs in each terminal node are from a single class. In this case, the Gini index/deviance is 0. | |
| When - 1, each terminal node contains only 1 obs, so training RSS = 0 | |
| Comparison btw pruning and best subset selection  The tree size is analogous to the number of predictors to be included in the best subset selection  Pruning finds the best subtree w the given tree size, and best subset selection finds the best linear model w the given num of predictors | |
| Pruning reduces estimation variance but increases bias. In bagging, the estimation variance has been reduced by aggregation of multiple trees. In general, pruning is unnecessary in bagging; we want each bootstrap tree to have a relatively small bias. | |
| Why averaging many highly correlated quantities don't lead to as large of a reduction in var as averaging many uncorrelated quantities?  Suppose . WLOG, asuume E(Xb) = 0 and Var(Xb) =  Then Var() =  In practice, is often positive | |
| Why is boosted model not T(**x**) = ?  In boosting, tree is built upon previous tree by using residuals, and model is updated by T(**x**) T(**x**) + Tb(**x**) | |
| Why use small trees (d=1 or d=2) in boosting? In boosting, each tree just needs to have a small improvement in ares where previous trees don't do well, so not necessary to use a big tree to fit the data too "hard" | |
| Why should B be chosen carefully in boosting, while be as large as possible in bagging and RF?  Large B in boosting may cause overfitting, while in bagging and RF, large B actually contributes to larger reduction in estimation variance. | |
| Tut | *library(tree); rtree=tree(Y~., df[train,]); plot(rtree); text(rtree) # get conditions for split in text form;*  *mean((predict(rtree, df[test,]-df$Y[test]^2) #MSE*  *cv = cv.tree(rtree); plot(cv$size, cv$dev); k=cv$size[which.min(cv$dev)]; pruned = prune.tree(rtree, best=k); plot(pruned); text(pruned)* | |
| *library(randomForest); p=ncol(df) - 1 #num of predictors*  *bagging=randomForest(Y~., df, subset=train, mtry=p, importance=T) #mtry = num of predictors to consider when choosing split*  *bagging.mse=mean((predict(bagging, df[test,] - df$Y[test])^2); importance(bagging)* | |
| *rf = randomForest(Y~., df, subset=train, mtry=sqrt(p), importance=T)*  *rf.mse = mean((predict(rf, df[test,] - df$Y[test])^2); importance(rf)* | |
| *library(gbm); lambda=seq(0.01, 0.2, length.out=10);* Suppose we fixed num of trees = 1000  *train.MSE=sapply(lambda, function(lam){boosting=gbm(Y~., data=df[train,], n.trees=1000, shrinkage=lam, distribution='gaussian')*  *yhat=predict(boosting, df[train,]; mean((yhat-df$Y[train])^2)})*  *plot(lambda,train.MSE); boost=gbm(Y~., data=df[train,], n.trees=1000, shrinkage=0.1, distribution='gaussian'); summary(boost)* | |

|  |  |  |  |
| --- | --- | --- | --- |
| Single Layer Neural Network (input + 1 hidden + output layer) | | | - p input variables X = (X1, …, Xp). - nonlinear fn f(X) to predict Y. - feed-forward neural network.  are activations. g is a nonlinear activation fn. Input: X; Hidden: A; Output: f(X) -> Y  , where and for j = 0, 1, …, p and k = 1, …, K are parameters, estimated by minimizing RSS |
| Activation Functions | Sigmoid activation fn . ReLU (rectified linear unit)  is activated when nonzero (for ReLU) or close to 1 (for sigmoid) | | |
| Multilayer Neural Network (> 1 hidden layer) | MNIST Digits: 28 x 28 grayscale image. So p = 784 features, X1, …, Xp. Each Xj (0, 255). Response: 0,1,…,9. n = 60000  K1 = 256 units at first layer. K2 = 128 units at second layer. One-hot encoding for response: Y = (Y0, …, Y9). q = 10 units at output layer  At first hidden layer: For k = 1, …, K1: . Second hidden layer: For = 1, …, K2, .  **W**1 = all parameters btw input layer and first hidden layer. **W**2 = all params btw first and second layers  Let , for m = 0, …, 9. **B** = set of parameters for m = 0, …, 9  Softmax activation fn for modeling fm(X) = Pr(Ym = 1|X) =  Fit model by minimizing cross-entropy = , where (yi0, …, yi9) is the one-hot encoding for response of ith obs (use lower case variables to denote observed data)  weight: for j ≥ 1, for k ≥ 1, for ≥ 1. bias/intercept terms (not the bias in bias-variance): , , | | |
| Convolutional NN (CNN) | | CIFAR100 Database: 60000 labeled images. 20 superclasses, w 5 classes per superclass. Each image has resolution of 32 x 32 pixels.  RGB color vector (R,G,B) for each pixel. Each image is a 3-way array/feature map: 32 x 32 x 3 array of nums  CNN builds up image in hierachical manner. First identify low-level features, then combine them to form higher-level features  Uses convolution layers and pooling layers | |
| - dot-product: . If subimage of input image is similar to filter, then score is high, otherwise low  Convolve an image w a filter: .  Filters learned during training. Convolution w filter find common patterns that occur in diff parts of image. Results in new feature map | |
| Max pool: . Each non-overlapping (e.g. 2 x 2) block is replaced by its max  Pooling sharpens feature identification and allows for location invariance. Reduce dimension (e.g. by factor of 4) | |
| Filters are typically small, e..g each color channel 3 x 3  Feature maprs are reduced in size after each pool layer. So usually increase num of filters in next convolution layers to compensate  At the end, feature maps are flattened and fed into 1 or more fully connected layers  resnet50 trained on imagenet 1000-class image database has 50 layers | |
| Recurrent NN (RNN) | A diagram of a diagram  Description automatically generatedRNN to deal w sequential data.  Feature for each obs is a seq of vectors X = (X1, …, XL) (e.g. X is a document of L words, each Xj is a 1-hot vector that encodes a word)  Output Y can be a seq (e.g. in language translation), or a scalar (e.g. sentiment label of a movie review)  Each is a hidden layer, receiving and as input, and producing an output  The same weights **W**, **U**, **B** are used at each step in seq (hence the term recurrent) | | |
| Suppose , i.e. p-dimensional predictors. and , i.e. K activations/components  Then . And  Data pairs (**x**i, yi) with **x**i = (**x**i1, …, **x**iL) and each **x**il = (xil1, …, xilp)  Training: Minimize , where ai,L-1,s is the value of AL-1,s when the input is seq **x**i | | |
| A graph of a number of squares  Description automatically generated with medium confidenceFeature engineering. Truncate/pad seq of words W1, …, WL to be same length (e.g. L = 500)  1) Represent word Wl by a one-hot encoded binary vector Xl of dimension D, w all 0s and a single 1 in the position for that word in the dict  - Extremely sparse feature representation, dont work well in general  2) Word embedding: Embed words by using a lower-dimensional m x D matrix **E**  - **E** can be pretrained on very large corpora of documents; word2vec and GloVe are popular  - **E** can also be learned from data at hand | | |
| Fitting NN | Find and to minimize RSS, where  Solving this is hard as it is a non-convex optimization | | |
| Gradient Descent: Let w being a vector of all parameters  1) Start w an initial value of of , and set t = 0  2) Iterate until the objective fn R() fails to decr: {  - Find a vector that reflects a small change in s.t. reduces R(), i.e. R() < R(). - Set t t + 1}  Find dir by using the gradient vector,  This gradient points upwards, so take and , where is the learning rate (e.g. )  Note suppose Q(u) = f(g(u)). Then | | |
| Backpropagation. Let .  Then and . Let  Backpropagation: For k ≥ 1 and j ≥ 1: {  }  Forward pass: Given , calculate and and and  Backward pass: Update parameters by and | | |
| Starting values: – exact 0 starting value leads to 0 derivatives, and the algorithm never moves – large starting value often leads to poor solutions – usually starting value is chosen to be a random value near 0  Multiple minima: partially addressed by trying multiple starting values | | |
| Regulari-zation | Previous e.g. MNIST dataset, w one-hot encoding: yi = (yi0, …, yi9) w yim = 1 if ith obs is digit m and 0 otherwise  Let be vector of d parameters  Ridge-type penalty: ,  Lasso-type penalty: | | |
| Stochastic Gradient Descent (SGD): For each t, in the backward pass, instead of over all n obs, only sum over a random minibatch.  E.g. suppose n = 10 and minibatch size = 3. At step t+1, sample 3 obs (e.g. 1, 4, 9). Then SGD updates:  and | | |
| Early stopping. Suppose minibatch size = 128 obs. Epochs = num of times an equivalent of the full training set has been processed.  E.g. if n = 48000, then each epoch contains 48000/128 ≈ 375 minibatch gradient updates  Can perform Early stopping as form of regularization | | |
| Dropout. At each SGD update, randomly remove a fraction of (both input and hidden) units  The weights of the surviving units are scaled up by a factor of 1/(1-)  In practice, dropout is achieved by setting activations for the "dropped out" units to 0, while keeping architecture intact | | |
| Data augmentation: Make many copies of each (**x**i, yi) and add a small amt of noise to **x**i, or do some random transformation of **x**i  This makes model more robust to small perturbations in **x**i  E.g. for image classification, can perform natural transformations to each training image when it is sampled by SGD  Typical distortions are zoom, horizontal and vertical shift, shear, small rotations, horizontal flips | | |
| Hyperpa-rameters | Num of hidden layers, num of units per layer. If using data augmentation: details of data augmentation  If using regularization: dropout rate , strength for lasso and ridge penalty. If using SGD: learning rate, minibatch size, num of epochs. | | |
| Supple-mentary | A diagram of a network  Description automatically generated- When using NN for classification, if output layer need M classes = output layer should have M units (due to one-hot encoding)  - For 2 layer fully connected NN. If input is 28 x 28 grayscale image, then p = 784 features. Suppose K1 = 256 and K2 = 128. And response is 0, 1, …, 9. Then total num of parameters = (p+1)K1 + (K1 + 1)K2 + (K2 + 1)10. (rmb need add bias term)  - For RNN, if output is scalar, input seq is p-dimensional vector and each hidden layer has K units. Show each basic building block by a single-layer neural network  - Backpropagation formula for and      - Check = 0 leads to 0 derivatives  When = 0, . Then and = 0 (for ReLU activation fn)  Then . But not for | | |
| In SGD, for diff t, minibatch is diff. In practice, a minibatch is drawn w/o replacement | | |
| If minibatch size = k and sample size is n, then num of minibatch gradient updates per epoch = n/k | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Sure Indepen-dence Screening (SIS) | Sparsity assumption: when p is large, maybe only a few predictors are related to response.  When p n, Lasso, LARS, subset selection find it diff to identify those relevant predictors.  E.g. lasso, when p is large, need to be large to shrink coefficients to 0 large bias and make it diff to achieve good bias-var tradeoff | | |
| SIS first screen out most irrelevant variables. Then perform variable selection via Lasso/LARS/etc w the reduced num of variables  1) Standardization: - Center response by redefining , where  - Standardize predictors , where and  2) Let = correlation btw jth predictor and response. If Xj is irrelevant to response, then rj should be small in magnitude | | |
| Let q = num of predictors to be kept after screening. SIS: Keep predictors whose |rj| is among the 1st q largest of |r1|, …, |rp|  q often chosen to be less than n (q = ), and q = p if p < n otherwise. | | |
| SIS-Lasso: 1) Perform SIS to screen out irrelevant predictors. 2) Perform Lasso on surviving predictors.  (Can swap Lasso to other var selection mtds, e.g. LARS)  However, SIS tends to miss some impt predictors. Soln: Iterative SIS (ISIS) | | |
| ISIS: 1) Center response and standardize predictors. 2) Apply SIS on data to screen out most predictors. Let A = set of surviving predictors  Iterate: { 1. Apply variable selection to select some predictors from A. Update A to be selected predictors.  2. Fit linear model w yi and predictors in A, and compute residuals Ri  3. Apply SIS on data Ri as response values and w predictors NOT in A. This results in some surviving predictors not in A. Add these  predictors into A, if num of predictors in A < q} | | |
| *library(SIS)*  *model10 = SIS(x, y, family='gaussian', penalty='lasso', iter=FALSE)*  *model10$sis.ix0* # SIS surviving predictors | | *model11 = SIS(x,y,family='gaussian', penalty='lasso', iter=TRUE)*  *model11$ix* # ISIS surviving predictors  # Screening = A after SIS. Selection = A after variable selection. Conditional-Screening = predictors not in A but to be added into new A |
| Markov Chains | Let S be a finite set of elements, e.g. S = {hot, cold, warm}. Elements of S are called states, S is called a state space  Consider a seq of r.v. Q1, Q2, … S.  Markov chain = seq of r.v. w Markov property. Markov assumption/property: Conditional on Qi, Qi+1 is indep of Q1, …, Qi-1  So Q1, Q2, … form a Markov chain if Pr(Qi+1 = ti+1|Q1 = t1,…, Qi = ti) = Pr(Qi+1 = ti+1|Qi = ti) = transition probability whenever Pr(Q1 = t1,…,Qi = ti) > 0, for t1, …, ti, ti+1 S  Dist of a Markov chain is determined by dist of Q1 and the transition probabilities | | |
| A diagram of a cold  Description automatically generatedTime-homogeneous Markov chain: Pr(Qi+1 = t|Qi = s) = Pr(Qi = t|Qi-1 = s) for all i ≥ 2 and all s,t S (i,=.e. prob same throughout)  Transition probabilities P(s t) = Pr(Qi+1 = t|Qi = s). Let elements of S be s1, …, sp  Define Markov chain using 1) Transition probabilities for j, k.  2) Initial probability dist for j = 1,…,p  . E.g. directed weighted graph. A = | | |
| Expectation-Maximization Algo | | Gaussian Mixture Models (GMM). Let X1, …, Xn be iid observable data. Let Z1, …, Zn Bernoulli(r) be iid hidden variables  E.g. Zi indicate whether patient has a (not directly observable) disease and Xi is the corresponding blood pressure  Simple GMM =  The marginal dist of Xi is a mixture of 2 Gaussian dist, i.e. its prob density = , where denotes prob density of the Gaussian dist w mean and var  Goal: - Estimate parameters from the observed data x1, …, xn. - Infer Zi given Xi = xi  GMM vs classification: In GMM, Z1, …, Zn are not observable. In classification Z1, …, Zn are observed and part of the data  Complete data: (X1, Z1), …, (Xn, Zn). Incomplete data: X1, …, Xn. Missing data: Z1, …, Zn | |
| Likelihood w Incomplete Data: find parameters to maximize likelihood  Incomplete log-likelihood based on incomplete data . Directly maximizing L is diff  Suppose Zi are observable, complete log-likelihood based on complete data .  Maximizing complete log-likelihood is easier | |
| EM algo: 0) Provide initial values for parameters  1) Estimate Z1, …, Zn based on give values of parameters. Estimate Zi by . Step 1 = E step  2) Based on estimated Z1, …, Zn , find parameters to maximize complete log-likelihood. Step 2 = M step  3) Repeat 1) and 2) until convergence is reached | |
| General EM algo: Let **X** = (X1, …, Xn) be observed data. Let **Z** = (Z1, …, Zn) be hidden variables. Let  Initialization: Set up initial value for  For k = 1,2,…, { E step: Compute based on . M step: Find to max complete log-likelihood }  E.g. of stop criterion: or for a small value of (e.g. = 10-8) | |
| Let denote the Gaussian prob dist. Let be values of parameters in kth iteration  E Step: Compute  M Step: Compute likelihood based on and **X**.  Find derivatives and set to 0. . .  .  Solving, . . . | |
| Hidden Markov Models (HMM) | E.g. Part-of-speech (POS) tagging in NLP. "The sailor close the hatch". POS tags: article noun verb article noun  Sentence is observed, while POS tags are not. Goal is to build a model for X and Q, to infer Q from X  Let X = (X1, …, Xn) be a seq of obs w symbol space T. In POS, T = vocabulary.  Let Q = (Q1, …, Qn) be a seq of hidden random variables w state space S. In POS, S = POS tags | | |
| A diagram of a complex equation  Description automatically generated with medium confidenceFirst-order HMM: - Markov assumption = Q1, …, Qn form a time-homogeneous Markov chain  - Output independence: dist of Xi depends only on the hidden variables Qi, i.e. Pr(Xi|Q1,…,Qi,…,Qn,X1,…,Xi-1,Xi+1,…,Xn) = Pr(Xi|Qi)  Define first-order HMM by (π, A, B): - Initial dist π of Q1  - Transition probability matrix A of markov chain Q  - Emission probability matrix B of probabilities Pr(Xi = t|Qi = s) for s S and t T | | |
| Likelihood. Let **x** = (x1, …, xm) and **q** = (q1, …, qm). Let (π,A,B) be a given first-order HMM  Goal: Compute likelihood P(**x**), i.e. prob of observing seq **x**. Let P(**x**, **q**) = joint prob of **x** and **q**. Let P(**x**|**q**) = conditional prob of **x** given **q**  Marginal Prob of **x** = P(**x**) =  Using eqn P(a,b|**c**) = P(a|**c**, b)P(b|**c**). Then P(**x**|**q**) = P(xm|**q**)P(x1,…,xm-1|**q**) = P(xm|**q**, x1, …, xm-1)P(x1,…,xm-1|**q**) = P(xm|qm)P(x1,…,xm-1|**q**) (using output independence assumption). Then P(x1,…,xm-1|**q**) = P(xm-1|qm-1)P(x1,…,xm-2|**q**) w the same logic...  So P(**x**|**q**) = P(xm|qm)P(xm-1|qm-1) ... P(x1|q1).  Using earlier e.g. for **x** = (3,1,3) and **q** = (hot,hot,cold). P(313|HHC) = P(3|H)P(1|H)P(3|C) = .4 \* .2 \* .1 = 0.008  Also P(**q**) = P(qm, q1,…, qm-1) = P(qm|q1,…, qm-1)P(q1,…, qm-1) = P(qm|qm-1)P(q1, …, qm-1) (using Markov assumption) = P(qm|qm-1)…P(q2|q1)P(q1)  Using earlier e.g. for **q** = (hot,hot,cold). P(HHC) = P(C|H)P(H|H)P(H) = .4 \* .6 \* .8 = .192  So P(**x**) = = P(xm|qm)P(xm-1|qm-1) ... P(x1|q1) \* P(qm|qm-1) … P(q2|q1)P(q1) | | |
| Training. In above, only have 1 seq **x** and 1 hidden seq **q**. In practice have multiple sequences  Observed data: n sequences **x**1, …, **x**n, each seq **x**i = (xi1, …, ), where mi = length of **xi**. Hidden seq: **q**1, …, **q**n , each seq **q**i = (qi1, …, )  As in the EM algo, treat hidden sequences **q**1, …, **q**n as missing data. Complete data: (**x**1, **q**1), …, (**x**n, **q**n)  IF we had the complete data, then - is estimated by , (frac of starting state = sj)  - is estimaed by where mi = length of **x**i. (num of state sj sk/num of LHS state = sj)  - Emission probability of B is estimated by (frac of state = sj giving obs = tk)  E.g. **x**1 = (3,3,2). **q**1 = (hot, hot, cold). **x**2 = (1,1,2). **q**2 = (cold,cold,cold). **x**3 = (1,2,3). **q**3 = (cold,hot,hot)  Then = 1/3. = 2/3. Estimate A by P(hot hot) = 2/3, P(hot cold) = 1/3. P(cold cold) = 2/3. P(cold hot) = 1/3  Estimate B by P(hot 1) = 0/4 = 0. P(cold 1) = 3/5. P(hot 2) = 1/4. P(cold 2) = 2/5. P(hot 3) = 3/4. P(cold 3) = 0/5 = 0 | | |
| In reality, hidden seq **q**1, …, **q**n are not observable.  Observe that  In spirit of the EM algo, estimate probabilities in numerator and denominator by Pr(ql = sj, ql+1 = sk|**x**1, …, **x**n) and Pr(ql = sj|**x**1, …, **x**n)  If (π,A,B) are known, then conditional probabilities can be computed easily. So can use EM-type iterative procedure. | | |
| *library(HMM)* # Initialise HMM w known state space, symbol space, initial dist, A, B  *hmm <- initHMM(c("H", 'C'), c('1', '2', '3'), c(.3, .7), matrix(c(.6,.4,.5,.5), nrow=2, byrow=T), matrix(c(.2,.4,.4,.5,.4,.1), nrow=2, byrow=T) )*  *obs <- simHMM(hmm, 100)* # Simulate 100 obs. *vt = viterbiTraining(hmm, obs$observation)*  *OR source('HMM.R'); BaumWelch(obs$observation, hmm$transProbs, hmm#emissionProbs, hmm$startProbs, n.iter=100)* | | |
| Supp | Show that the initial dist π, and transition prob matrix A, completely determine the dist of a time-homogeneous Markov chain  Let Q1, …, Qm be a time-homogeneous Markov chain, w state Space S = {s1, …, sp}  For any q1, …, qm S, write P(q1, …, qm) for P(Q1 = q1, …, Qm = qm). Then P(q1, …, qm) = P(qm|q1, …, qm-1)P(q1, …, qm-1) (Bayes theorem) = P(qm|qm-1)P(q1, …, qm) = … = P(qm|qm-1)P(qm-1|qm-2)…P(q2|q1)P(q1)  In the above, each P(qj|qj-1) is given by a transition prob in A and P(q1) is given by initial dist π | | |
| Show that a first=order HMM is determined by the initial dist π of Q1, transition prob matrix A of the Markov chain Q and the emission prob matrix B of probabilities Pr(Xi = t|Qi = s) for s S and t T  Equivalent to showing that the joint prob P(**x**,**q**) can be calculated from π,A and B  P(**x**, **q**) = P(**x**|**q**)P(**q**) = P(xm|qm)P(xm-1|qm-1) ... P(x1|q1) \* P(qm|qm-1) … P(q2|q1)P(q1)  In above, P(xj|qj) is computed from B. P(qj|qj-1) is computed from A. P(q1) is computed from π | | |
| Tut | NN w 2 hidden layers: p = 4 input units, 2 units in first hidden layer, 3 units in second hidden layer, and a single output. ReLU activation  . for  for  Total, there are (p+1)K1 + (K1 + 1)K2 + (K2 + 1) = 23 parameters | | |
| CNN takes in 32 x 32 grayscale image and has a single convolution layer w three 5 x 5 convolution filters (w/o boundary padding)  Output is three 28 x 28 image. Total 5 x 5 x 3 = 75 parameters  How can model be thought of as an ordinary feedforward NN w the individual pixels as inputs, and w constraints on the weights in the hidden units. What are the constraints?  Think of 3 blocks (not layers) of 28 x 28 hidden units, each w a diff set of parameters. Take 1 block as example. Suppose hidden units in this blocks are indexed by Ajk, for j,k = 1, …, 28  Then , where g = activation fn and Xuv = pixel at uth row and vth col. To make this equivalent to CNN, need to impose constraints: for all j, k; for u, v > 5;  for u = j, j+1, …, j+4 and v = k, k+1, …, k+4; otherwise set  Without the constraints, there are (32 x 32 + 1) x 28 x 28 x 3 = 2.4108 x 106 parameters | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| K-means clustering | | A clustering of K clusters is formed by sets C1, …, Ck containing the indices of the obs and satisfying: = {1, 2, …, n} AND for j ≠ k. | | |
| Good clustering = small within-cluster variation (WCV). , where = num of elements in  Given K, find a good clustering of K clusters, C1, …, CK to minimize  1) Initialization. Randomly assign a num, from 1 to K, to each of the obs  2) Iterate until the cluster assignments stop changing: - For each of the K clusters, compute the cluster centroid =  - Assign each obs to the cluster whose centroid is closest, i.e. place each index i into Ck s.t. , where is the centroid computed  Run algo multiple times to have multiple initialization to find min | | |
| Hierarchical Clustering | | | A diagram of a bar graph  Description automatically generatedA diagram of a circle with different colored circles  Description automatically generatedDon't require us to pre-specify the num of clusters. Basic idea: build a hierachy in a bottom-up fashion  1) Start w each point as its own cluster.  2) Identify the closest 2 clusters and merge them  3) Repeat until all points are in a single cluster  Y axis on dendrogram = dissimilarity measure  To determine K, from dendrogram, draw a horizontal cut. Obs below the cut can be interpreted as clusters | |
| For 2 obs and , use their Euclidean distance = D(, ) = , where and  For 2 clusters, dissimilarity = linkage  Complete-Linkage(Ck, Cq) = . Single-Linkage(Ck, Cq) =  Average-Linkage(Ck, Cq) = , where nk, nq = size of Ck, Cq.  Centroid-Linkage(Ck, Cq) = , where (centroid of clusters Ck). defined similarly  Average and complete prefered over single as they tend to yield more balanced dendrograms | |
| Centroid Linkage often use in bioinformatics but has problem of inversion  A diagram of a diagram of a line  Description automatically generated with medium confidenceA graph of a function  Description automatically generatedInversion: When merging 2 clusters, the linkage btw the 2 merged clusters, where at least 1 of these 2 clusters are merged from another 2 clusters, is lower than the linkage of the previous merging. | |
| Hypo Testing | | 1) Define null and alternative hypotheses, H0 and H1. H0 = default state of belief about the world (e.g. coeff = 0, no diff in expected values)  2) Construct test statistic. E.g. let / denote means, = num. To test H0: , use 2-sample t-statistics.   |  |  |  | | --- | --- | --- | |  | Truth | | | Decision | H0 | H1 | | Reject H0 | Type I error | Correct | | Do not reject H0 | Correct | Type II error |   3) Compute p-value (prob of observing a test statistic at least as extreme as the observed statistic, assuming H0 is true). Small p-value = H0 likely false  4) Decide whether to reject H0. Small p-value = reject H0 since unlikely for test statistic to be this large.  Want to ensure small Type I error rate. (since more serious)  If we only reject H0 when p-value < , then Type I error rate (prob of making Type I error) will be at most | | |
| Multiple Testing | | Want to test m null hypotheses, H01, …, H0m. Suppose we reject any null hypothesis w p-value < 0.01  Then we expect to falsely reject appox 0.01 \* m null hypotheses. If m = 10000, then we expect to falsely reject 100 H0 by chance | | |
| |  |  |  |  | | --- | --- | --- | --- | | Decision | H0 true | H1 true | Total | | Reject H0 | V | S | R | | Do not reject H0 | U | W | m - R | | Total | m0 | m - m0 | m |   Family-wise error rate (FWER) = prob of making at least 1 type I error when conducting m hypothesis tests. FWER = Pr(V ≥ 1)  FWER = 1 - Pr(don't falsely reject any H0) = 1 - Pr()  If tests are indep and all H0j are true, then FWER = | | |
| Bonferroni Correction: control FWER at level , reject any H0 w p-value <  FWER = Pr(falsely reject at least 1 H0) = Pr() ≤ , where = event we falsely reject jth H0  If we only reject H0 when p-value < , then FWER ≤ (since Pr(Aj) ≤ ) | | |
| A graph of a graph showing a line  Description automatically generated with medium confidenceWhen m is large, Bonferroni Correction is super conservative (i.e. rarely reject H0)  Instead, can control the false discovery rate: FDR = E(V/R) = E(num of false rejections/total num of rejections)  FWER controls Pr(at least 1 false rejection). FDR controls fraction of candidates in smaller set that are really false rejections. Use Benjamini-Hochberg Procedure:  1) Specify q, level at which to control the FDR  2) Compute p-values p1, …, pm for the null hypotheses H01, …, H0m  3) Order the p-values s.t. p(1) ≤ p(2) ≤ … ≤ p(m).  4) Let L = max{j: p(j) ≤ qj/m}  5) Reject all H0j for which pj ≤ p(L). Then FDR ≤ q  On graph: Bonferroni = reject H0 below green line. FDR = reject H0 in blue | | |
| Supp | | If we reject H0: coin is fair when the coin come up all tails. Suppose, among 10 flips, 7 are tails. What is the p-value? The events that are at least as extreme as the observed one (7 tails out of 10 flips) are:  - 7 tails out of 10 flips, w prob (10 Choose 7) \* P(T7) \* P(H3) = 120/1024. P(8 tails) = 45/1024. P(9 tails) = 10/1024. P(10 tails) = 1/1024.  So p-value = (120+45+10+1)/1024 = 176/1024 | | |
| Tut | Suppose we test m null hypotheses, all of which are true. We control the Type I error for each null hypothesis at level α.  - In total, how many Type I errors do we expect to make? **Answer.** Expected number of Type I errors is mα, regardless whether the tests are indep.  - Suppose that the m tests that we perform are independent. What is the family-wise error rate associated with these m tests? Hint: If two events A and B are independent, then Pr(A ∩ B) = Pr(A)Pr(B).  **Answer.** Let V = num of Type I errors. Let Aj = event jth hypothesis is not rejected  FWER = Pr(V ≥ 1) = 1 - Pr(V = 0) = 1 - Pr(A1 ∩ … ∩ Am) = 1 - Pr(A1)…Pr(Am) = 1 - (1 - α)m  - Suppose that m = 2, and that the p-values for the two tests are positively correlated, so that if one is small then the other will tend to be small as well, and if one is large then the other will tend to be large. How does the family-wise error rate associated with these m = 2 tests qualitatively compare to the answer in (b) with m = 2? Hint: First, suppose that the two p-values are perfectly correlated.  **Answer.** Suppose situation is A1 = A2. Then FWER = Pr(V ≥ 1) = 1 - Pr(A1 ∩ A2) = 1 - Pr(A1) =  When A1 ≈ A2, then FWER is approximately α, which is relatively smaller than the FWER in (b). In other words, when the tests are highly positively correlated, testing m hypotheses is like testing one hypothesis, when FWER is concerned.  - Suppose again that m = 2, but that now the p-values for the two tests are negatively correlated, so that if one is large then the other will tend to be small. How does the family-wise error rate associated with these m = 2 tests qualitatively compare to the answer in (b) with m = 2? Hint: First, suppose that whenever one p-value is less than α, then the other will be greater than α. In other words, we can never reject both null hypotheses.  **Answer.** Imagine A1 ∩ A2 = ø. Then FWER = 1 - Pr(A1 ∩ A2) = 1 - Pr(ø) = 1.  i.e. if tests a negatively correlated, FWER will be substantially inflated if not controlled properly. | | | |
| Suppose that we test m hypotheses, and control the Type I error for each hypothesis at level α. Assume that all m p-values are independent, and that all null hypotheses are true.  (a) Let the random variable Aj = 1 if the jth null hypothesis is rejected, and 0 otherwise. What is the distribution of Aj? **Answer.** Aj ∼ Bernoulli(α). (b) What is the distribution of ? **Answer.** Under the independence assumption, ∼ binomial(m, α). (c) What is the standard deviation of the number of Type I errors that we will make?  **Answer.** The number of Type I errors is V = ∼ binomial(m, α), so that var(V) = mα(1 − α) and the standard deviation is . | | | |
| Suppose we test m null hypotheses, and control the Type I error for the jth null hypothesis at level αj , for j = 1, . . . , m. Argue that the family-wise error rate is no greater than . Let Bj = event we reject the jth null hypothesis.  **Answer.** WLOG, assume the first m0 null hypotheses are true. Then FWER = Pr(V ≥ 1) = Pr(B1 ∪ … ∪ ) ≤ = ≤ | | | |
| |  |  | | --- | --- | | Null Hypothesis | p-value | | H01 | 0.0011 | | H02 | 0.031 | | H03 | 0.017 | | H04 | 0.32 | | H05 | 0.11 | | H06 | 0.90 | | H07 | 0.07 | | H08 | 0.006 | | H09 | 0.004 | | H10 | 0.0009 | | | | Control the Type I error for each null hypothesis at level α = 0.05. Reject which null hypotheses?  **Answer.** Reject H01, H02, H03, H08, H09, H10 since p-value < 0.05  Now suppose that we wish to control the FWER at level α = 0.05. Reject which null hypotheses? **Answer.** If use Bonferroni correction to control FWER, reject null hypotheses if p-value < α/m = 0.005. Reject H09, H10  Now suppose that we wish to control the FDR at level q = 0.05. Reject which null hypotheses? **Answer.**  First create table:   |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  | H10 | H01 | H09 | H08 | H03 | H02 | H07 | H05 | H04 | H06 | | j | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | | p-value | 0.009 | 0.0011 | 0.004 | 0.006 | 0.017 | 0.031 | 0.07 | 0.11 | 0.32 | 0.9 | | qj/m | 0.005 | 0.01 | 0.015 | 0.02 | 0.025 | 0.03 | 0.035 | 0.04 | 0.045 | 0.05 |   So L = 5 and p(L) = 0.017. So reject H10, H01, H09, H08, H03  Of the null hypotheses rejected at FDR level q = 0.5, approximately how many are false positives (FP)?  **Ans.** When q = 0.5, we reject R = 5 null hypotheses. Since V/R ≈ FDR = 0.5, we approx have V ≈ 0.5R = 2.5 FP |
|  | Explain why the analysis is misleading (same num of H0 rejected) . Hint: Standard approaches for controlling FWER and FDR assume that all tested null hypotheses are adjusted for multiplicity, and that no “cherry-picking” of the smallest p-values has occurred.  **Answer.** For FWER with Bonferroni correction, we adjust the p-values by the number m of tests. By cherry-picking the k best managers, the tests with small p-values are adjusted by k = 10 ≪ m. This will then lead to higher chance of (falsely) rejecting some null hypotheses. Similar arguments apply to FDR. | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| High-Dimensional 1-sample Problem | | | Data: **X**i ~ **X** = (X1, …, Xd) for i = 1, …, n w d features. for j = 1, …, d. High dimension: d ≈ n or d n  Test d hypotheses, H0j: vs Haj: , where each is a given value. |
| CI and Hypo Test (d = 1) | | Suppose Y1, …, Yn ~ Y iid w E(Y) = . Hypo test: H0: vs Ha: . Suppose CIa is a 1 - a CI of , i.e. Pr( CIa) = 1 - a  So we reject H0 if CIa. The Type I error rate = Pr(Reject H0|H0 is true) = Pr( CIa|) = a | |
| Simulta-neous CI (SCI) | Intervals, are 1 - simultaneous CI for if Pr( for all j = 1, …, d) = 1 -  So reject H0j: if  FWER: WLOG, assume H01, …, H0r are true, where r ≤ d. Let Aj be event we do not reject H0j  FWER = 1 - Pr(A1 … Ar) = 1 - Pr( for j = 1, …, r) ≤ 1 - Pr( for j = 1, …, r and for j = r+1, …, d) = | | |
| Construct CI (d = 1) | Suppose Y1, …, Yn ~ Y iid w E(Y) = and Var(Y) = . Let being sample mean, define  q = quantile fn of T, i.e. Pr(T ≤ q()) = for . A 1 - CI = since  Pr() =  To find and : use CLT:  Let F be the cumulative dist fn (CDF) of T (depending on n) and be the CDF of the standard normal dist.  Then for any fixed t, as n ∞. Also and  Hence, when n is sufficiently large (e.g. n ≥ 30), and  Approximate CI is | | |
| Construct SCI | **X**i = (Xi1, …, Xid) ~ **X** = (X1, …, Xd) for i = 1, …, n with and . For each j = 1, …, d, define , where is the sample mean of the jth coordinate. Let and  Let and be the quantile fn of and respectively. An (oracle) 1 - SCI = dinvr  Pr() =  To find and | | |
| Write if as n ∞, where F(t) = CDF of T. Write for a dist F is as n ∞  E.g. with Z ~ N(0,1), also equivalently . If and is cts, then for  Gaussian appox: Multivariate CLT: when d is fixed and d n, , where **C** = correlation matrix of **X** = (X1, …, Xd)  Berry-Esseen Theorem (High-dimensional CLT): for **Z** ~ N(0, **C**), ,  i.e. even when d (order of magnitude eqiuivalent), still have the Gaussian approx | | |
| If Xj is sub-Gaussian in the sense that for all x > 0 for some constants C > 0 and a > 0 and for all j = 1, …, d, then  where = smallest eigenvalue of the correlation matrix of **X** and A > 0 is some constant.  This allows d for any fixed k when n is sufficiently large | | |
| Sofor **Z** ~ N(**0**, **C**). Continuous mapping theorem: If and g is a cts fn, then  **.** The fn g(t1, …, td) := max{t1, …, td} is cts  Thus, we have the Gaussian appox: with (Z1, …, Zd) ~ N (**0**, **C**). is the Gaussian counterpart of M  Similarly, | | |
| Approx SCI. implies for any . And implies for any  A 1 - approx (oracle) SCI is . For each j = 1, …, d: Reject H0j: if .  Then FWER is approx . To find and … | | |
| Bootstrap approx. Finding and requires **C** which is unknown. So estimate **C** by its sample version  Let ∑ be the covariance matrix of **X**. Its sample version is , where  Then for j, k = 1, …, d: , where is the elem in the jth row and kth col of and  Let (,…,) ~ N(**0**, ) and define and and for (,…,) ~ N(**0**, )  Bootstrap approx: With high prob, and  Approx chain: so that . Similarly so that | | |
| To estimate by : - Sort ,…,in increasing order, i.e. ,…,. Take .  To estimate by : - Sort ,…,in increasing order, i.e. ,…,. Take .  E.g. if B = 1000 and = 0.05, then and | | |
| Empirical (1 - ) SCI for : . For each j = 1, …, d: Reject H0j: if  FWER is appox | | |
| High-Dimensional Multiple-Sample Problem | | Data: **X**ki = (Xki1, …, Xkid) ~ **X**k for 1 ≤ k ≤ K, i = 1, …, nk . E.g. **X**ki is the ith patient in the kth group  and  For each and each j: Test hypothesis vs  E.g. suppose there are K diff gene therapies, and is the expected gene expression levels of d genes of patients under the kth gene therapy. means the kth and lth therapies have no diff on the jth gene expression.  Also called the MANOVA (multivariate ANOVA) problem. High-dimensional MANOVA: d n or d n | |
| SCI for constrasts. Suppose we construct SCI, for each constrast :  For each : Reject if  Let . FWER | |
| Construct SCIs. Define , where is the sample mean of the jth coordinate of the kth group  is the covariance matrix of the kth group.  is the jth diagonal element of . Define and  If , then w probability at least , for all and : ,  Oracle () SCI is | |
| Gaussian Approx: Define (Zk1, …, Zkd) ~ N(0, ∑k), . and  (, …, ) is the Gaussian counterpart of (Tkl,1, …, Tkl,d): They share the same mean **0** and the same covariance matrix  and are the Gaussian counterparts of L and M respectively.  Gaussian Approx: and . So and | |
| Bootstrap approx: Estimate by the sample covariance matrix , where  Define (,…,) ~ N(0, ) and , where and is the jth diag elem of  is the empirical/bootstrap counterpart of  Define and . and are empirical/bootstrap counterpart of and  Approx chain: so that . Similarly so that | |
| Estimate and . Bootstrap procedure: For b = 1, …, B, sample (,…,) ~ N(0, ) and compute and by  and  To estimate by : - Sort ,…,in increasing order, i.e. ,…,. Take .  To estimate by : - Sort ,…,in increasing order, i.e. ,…,. Take . | |
| Empirical (1 - ) SCI for : .  For each : Reject if . FWER is appox | |
| Partial Standardization. Take . Partially standardized empirical () SCI for :  . All SCIs have same width if  leads to fully standardized SCIs introduced above. Pratically, select to maximize power of tests, while maintaining good control on FWER. Computationally expensive, but parallelizable and scalable | |