Statistical learning You-tube notes:

- Introduction

Make some graph before start analyze the data. See the correlation of variables first.

Some examples in statistical learning problems

E.g Spam emails features are more and more complicated these days. (solution?)

Learning tasks in hand-writing figures recognition problems

Classify a sample based on its features, use heat map single figure can capture the whole data set information and classify easily

- Supervised learning and unsupervised learning:

For unsupervised learning:

no outcome variables, just a set of predictors/features measured on samples

Objective is to find groups of samples for similarity and find linear combinations of features with the most variation. Difficult to know how well you are doing. It is a good way as a pre-processing step for supervised learning

E.g Netflix prize -- 98% missing training data: to decrease the MSE

Comparison of Machine Learning and Statistical learning:

Machine learning: large scale, concentrate on prediction accuracy

Statistical learning: precision and uncertainty

- Statistical learning and regression

1. Statistical learning and regression

find joint prediction of several predictors, y=f(x1,x2,x3)+EPMS, then make predictions of Y for new points X=x. The ideal f(x)=E(Y|X=x) is the regression function, expected value~average

Find whether function g is good enough: E[(Y-g(X))^2|X=x] minimize the error

EPMS = Y-f(X), irreducible error, still make errors

$E\left[(Y-\hat{f}(X)^2 \mid X=x\right]=\left[f(x)-\hat{f}(x)\right]^2+Var(\varepsilon)$, here the first one is reducible and the second term variance is irreducible, can improve the first one

One way to estimate function f or compute the expectation of Y: we can calculate the average of X=x neighbors if possible

Nearest Neighbors suitable for: small number of predictors p<=4 and large number of N to average. Not suitable if p is large. Reason: the curse of dimensions since nearest neighbors may be far away in high dimensions. For example, in two dimensions, the points are not local if in a circle. In order to get 10% of data, we may need to go to quite far.

2. Parametric and Non-parametric models

Linear model is important parametric model:

$f_L(X)=beta_0+beta_1X_1+...+beta_pX_p$, it does not rely on any simply one dimension so no problem for curse of dimension.

Quadratic model may fit better:

$f_Q(X)=beta_0+beta_1X+beta_2X^2$

Hint: thin-plate spline is a good way to show 2-dimensional function surface

We can even control the roughness of the surface

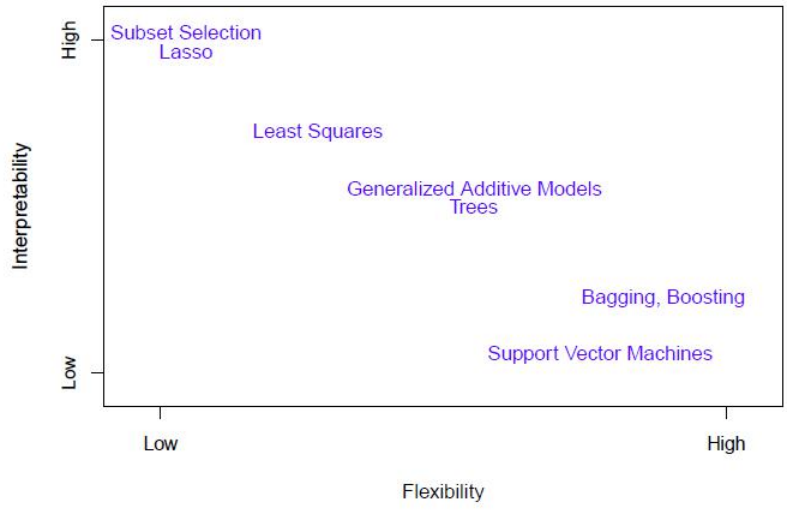
Over-fitting problem: get almost no training errors under this model

Some trade-offs:

Prediction accuracy and interpret-ability: linear models good to explain but thin-plat splines are not

Good fit versus over-fit or under-fit

Parsimony versus black-box: prefer a simpler model involving fewer variables over a black-box predictor involving them all.



3. Model accuracy

Assessing model accuracy:

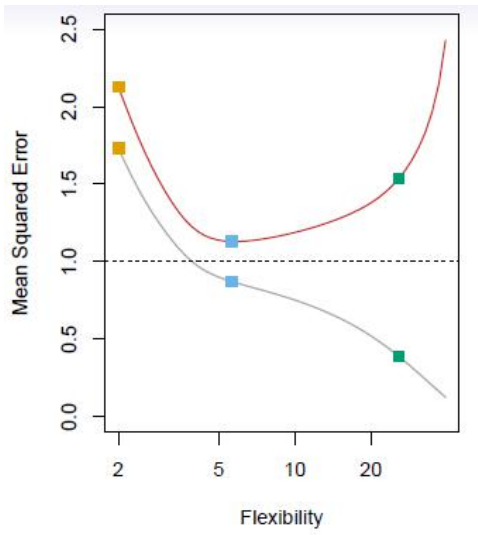
Average squared prediction error: $MSE_{Tr} = Average_{i \in Tr}\left[y_i - \hat{f}(x_i)\right]^2$

Introduce test data error to avoid over-fitting:

$$MSE_{Te} = Average_{i \in Te}\left[y_i - \hat{f}(x_i)\right]^2$$

Comparison of Test error line and training error line:

Training continue to decrease but test error goes above later



Should have the most good prediction error!

Bias-Variance Trade-off:

The reducible part above can be divided into variance and bias:

$$E\left(y_0 - \hat{f}(x_0)\right)^2 = Var\left(\hat{f}(x_0)\right) + \left[Bias\left(\hat{f}(x_0)\right)\right]^2 + Var(\varepsilon)$$

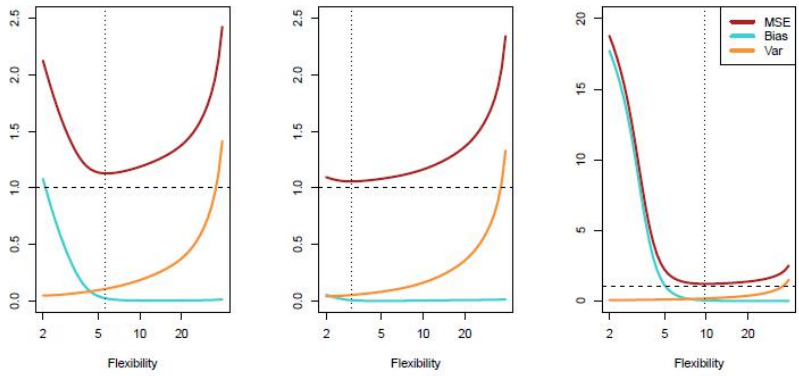
Here, $Bias\left(\hat{f}(x_0)\right) = E\left[\hat{f}(x_0)\right] - f(x_0)$

The expectation averages over the variability of y0 as well as variability in Tr.

So as the flexibility of f increase, it variance increases and its bias decreases.

Choose the flexibility based on average test error on bias-variance trade off

Plots for this bias-variance trade-off plot:



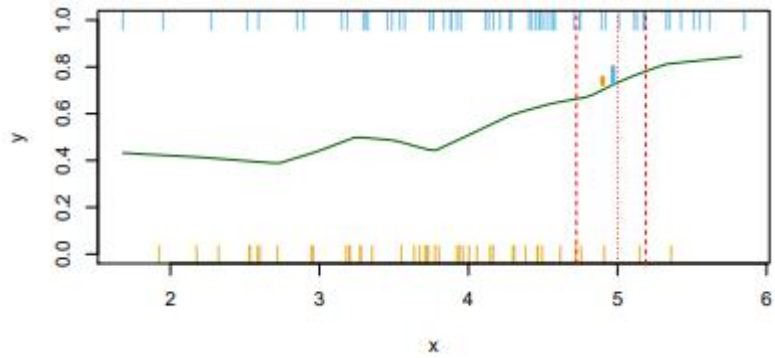
4. K-nearest neighbors (1/3 problems belong to K-nearest neighbors problems) Very important

Classification problem

$p_k(x) = \Pr(Y = k \mid X = x), k = 1,2,...K$, conditional class probability at x

Then Bayes optimal classifier at x: $C(x) = j$, if

$p_j(x) = \max\{p_1(x), p_2(x), ...p_k(x)\}$, nearest-neighbor averaging can be used here for the classification problems. However, breaks down as dimension grows, the impact of C(x) is less than on pk(x).



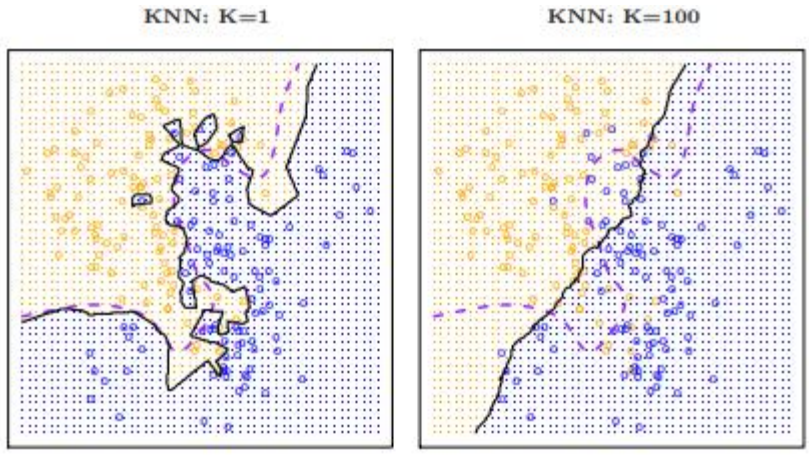
For multiple dimensions, we can still use the classification problem, assign to the majority of neighbors as well.

Some details for classification: measure the performance of classification

$Err_{Te} = Ave_{i \in Te} I\left[y_i \neq \hat{C}(x_i)\right]$, find the classification error

The Bayes classifier using the true pk(x) has the smallest error

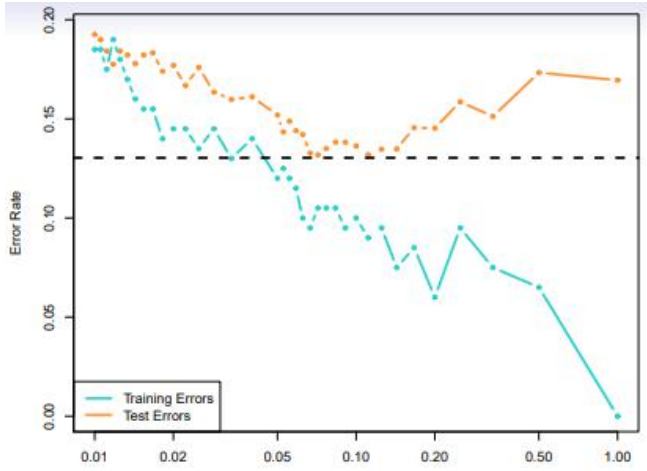
Support-vector machine would build models for C(x) and also build structured models for representing pk(x) -- logistic models



K=1 is a popular choice: closest neighbor method, a little bit noisy

K=100, consider really large range of points

Choice of K is tuning parameters:



In R studio, by using attach(data.frame), we can get variables by simply calling their names

- Linear Regression

1. Simple Linear regression

Simple is good sometimes

It is a simple approach to supervised learning.

Assumption: the dependence of Y on X1, X2...XP is linear

Under a single predictor X:

$Y = beta_0 + beta_1 X + \varepsilon$, intercept and slope

Estimation of the parameters by least squares:

Residual and the residual sum of squares (RSS):

$e_i = y_i - \hat{y}_i$, $RSS = e_1^2 + ...e_n^2 = \left(y_1 - \hat{\beta}_0 - \hat{\beta}_1 x_1\right)^2 + ...\left(y_n - \hat{\beta}_0 - \hat{\beta}_1 x_n\right)^2$

The least squares choose beta to minimize the RSS, then formula:

$\hat{\beta}_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$, $\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$, using average of y and x as well

We want to know how precise the coefficients are, assess the accuracy of the coefficient estimates:

The standard error of estimator reflects how it varies under repeated sampling:

$$SE(\hat{\beta}_1)^2 = \frac{\sigma^2}{\sum_{i=1}^{n}(x_i-\bar{x})^2}, SE(\hat{\beta}_0)^2 = \sigma^2\left[\frac{1}{n}+\frac{\bar{x}^2}{\sum_{i=1}^{n}(x_i-\bar{x})^2}\right], \sigma^2 = Var(\varepsilon)$$

Standard error for the slope and intercept.

These standard errors can be used to compute confidence intervals!

A 95% confidence interval is defined as a range of values such that with 95% probability, the range would contain the true unknown value of the parameter. It has the form: $\hat{\beta}_1 \pm 2SE(\hat{\beta}_1)$.

Explain: if we get repeated samples, each time we sample a new dataset and get estimation, with 95% of all estimations' true values would be included in this range.

With 100 sampled dataset and corresponding estimations, 95% of the true estimations values would be in that confidence interval.

2. Hypothesis Testing

Standard errors can also be used to perform hypothesis tests on the coefficients. The most common hypothesis test involves testing the null hypothesis of:

H0: there is no relationship between X and Y versus the alternative hypothesis

HA: there is some relationship between X and Y

$H_0 : \beta_1 = 0, H_A : \beta_1 \neq 0$

T-test: to test the null hypothesis, compute a t-statistic, $t = \frac{\hat{\beta}_1 - 0}{SE(\hat{\beta}_1)}$, this will have a t-distribution with n-2 degrees of freedom, assuming $\beta_1 = 0$. Then use statistical software to compute the probability of observing any value equal to $|t|$ or larger. We call this probability the P-value!!! If the P-value is very small, the conclusion is to refuse H0 and indicates that there is relationship.

The confidence interval would contain zero if you can not reject H0.

Residual Standard Error:

$$RSE = \sqrt{\frac{1}{n-2}RSS} = \sqrt{\frac{1}{n-2}\sum_{i=1}^{n}(y_i-\hat{y}_i)^2}$$

R-Squared or fraction of variance explained:

$R^2 = \frac{TSS-RSS}{TSS} = 1-\frac{RSS}{TSS}, TSS = \sum_{i=1}^{n}(y_i-\bar{y})^2$ is the total sum of squares

It explains how much variance we could have explained from this model.

It can be shown that in this simple linear regression setting that $R^2 = r^2$, here r is the correlation between X and Y.

$$r = \frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i-\bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i-\bar{y})^2}}$$

R-Squared can be hard to be found quite large. Above 0.5 can be really strong.

3. Multiple Linear regression

More than one predictor:

$$Y = beta_0 + beta_1 X_1 + beta_2 X_2 + ... + beta_p X_p + \varepsilon$$

We interpret beta as the average effect on Y of a one unit increase in certain X, holding all other predictors fixed.

We can use a hyper-plane to visualize the data.

Assumptions: The ideal should be uncorrelated variables.

Otherwise, problems: the variance of all coefficients tend to increase a lot and it is hard to interpret since everything else would be changed.

In fact, predictors usually change together!

Estimation and prediction for Multiple Regression

$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + ... + \hat{\beta}_p x_p$, then we need to

minimize the sum of squared residuals by taking derivatives:

$$RSS = \sum_{i=1}^{n} \left(y_i - \hat{y}_i\right)^2 = \sum_{i=1}^{n} \left(y_i - \hat{\beta}_0 - \hat{\beta}_1 x_{i1} - ... - \hat{\beta}_p x_{ip}\right)^2$$

Estimates are the Multiple least squares regression coefficient estimates.

Observe the t-statistic table and correlation table of variables and compare whether there are some self-correlation.

4. Model selection

Is at least one predictor useful? Use F-statistic:

TSS: no predictor model measure

$$F = \frac{(TSS - RSS)/p}{RSS/(n-p-1)} \sim F_{p,n-p-1}$$

An F statistic is a value you get when you run an ANOVA test or a regression analysis to find out if the means between two populations are significantly different.

It's similar to a T statistic from a T-Test;

A-T test will tell you if a *single* variable is statistically significant and an F test will tell you if a *group* of variables are jointly significant.

Decide the most important variables:

The most direct approach is called all subsets or best subsets regression. Compute the least squares fit for all possible subsets and then choose based on some criterion. (But can not explore all of them if p is too large, will be 2^p models)

Forward selection:

Begins with the null model with an intercept no predictors. Fit p simple linear regressions and add to the null model the variable that results in the lowest RSS one by one. Add combinations of two with lowest RSS and continue until some criterion/stopping rules.

Backward selection:

Begins with all variables in the model. Remove the variable with the largest p-value, least statistically significant. Fit again and continue the steps.

Some criterion: Mallow's Cp, AIC (information criterion), Bayesian IC, adjusted R-square and cross validation CV.

Other considerations in the regression model:

Some predictors are not quantitative but qualitative. These are factor variables or categorical predictors like gender, status or ethnicity.

Qualitative Predictors:

Baseline model!

Xi would be either 1 or 0 or levels depending on their groups.

Resulting model:

$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i = \beta_0 + \beta_1 + \varepsilon_i, else, \beta_0 + \varepsilon_i$

May introduce additional dummy variables if there are more than levels.

Thus K levels needs K-1 dummy variables to represent each of the categories

$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \varepsilon_i = \beta_0 + \beta_1 + \varepsilon_i, else, \beta_0 + \beta_2 + \varepsilon_i, else, \beta_0 + \varepsilon_i$

The level with no dummy variables would be called the baseline model.

It doesn't matter which baseline to choose but contrast is important.

5. Interactions and Non-linear models

Extensions of the linear models first:

Interactions: we previously assume that predictors are independent.

In marketing it is called synergy effect, and here interaction effect. Predictors change together in the model.

Models take the form:

$Y = beta_0 + beta_1 X_1 + beta_2 X_2 + beta_3 X_1 * X_2 + \varepsilon$, there would be interactions

The interactions p-value may be really small and thus significant

Pay attention to explain the unit change effect in interactions model

Hierarchy:

Sometimes it is the case that an interaction term has a very small p-value but the associated main effects do not.

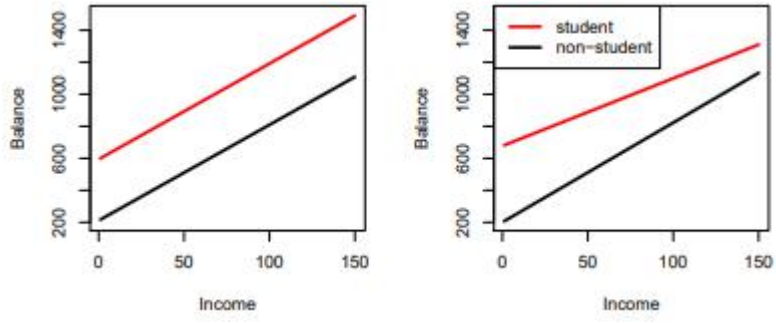
Principle: once we include an interaction in a model, we should also include the main effects even if the p-value with their coefficients are not significant.

Otherwise, that would be hard to interpret.

Interactions between qualitative and quantitative variables:

E.g: $balance_i = \beta_0 + \beta_1 income_i + \beta_2 + \beta_3 income_i, for, student, else, 0$

They may have different intercept and slope



Non-linear effects of predictors:
polynomial regression

Compare different degree for polynomials.

$$Y = beta_0 + beta_1 X_1 + beta_2 X_1^2 + \varepsilon$$

That is still linear model since it is linear in coefficients

Generalizations of the Linear Model:

1. Classification: Logistic regression, SVM
2. Non-linearity: kernel smoothing, splines nearest neighbors
3. Interactions: tree-based, bagging, random forests and boosting
4. Regularized fitting: ridge regression and Lasso

A Lab trick:

```
Reg-plot = function(x,y,...){
    Fit = lm(y~x)
    Plot(x,y,...)
}
```

You can add any plot command in to ... to pass it into function

Some detailed analysis here:

First some problems:

1. Non-linearity of the response-predictor relationship

   The linear regression model assumes that there is a straight-line relationship between the predictors and the response. Not the case.

   Here plot the residual plot. The presence of some patterns should indicate non-linearity features in linear models.

   A simple approach is to use non-linear transformations of the predictors, such as log form of X, root X, and X-square, in the regression model.

2. Correlation of error terms

   Assumption of error terms should be unrelated. Otherwise, it would under-estimate the true errors and confidence interval would be narrower. P-values would be smaller than it should be thus mislead the significance of variables. This is caused since many observations are at adjacent time points and they are positively correlated.

   By observing trend of residuals, common trends contain heteroscedasticity. If we take the log form of predictors, this trend may be reduced a little bit.

3. Outliers

   Some wrong observations or extremely different points. Residual plots can be used to identify outliers. In practice, it would be hard to define how large residuals should be. Thus we can also plot studentized residuals, compute by dividing each residual by its estimated standard error. If the result is larger than 3, it is possibly outliers.

   Once decide, simply remove the outliers but outliers may instead indicate a deficiency with the model, like missing predictors.

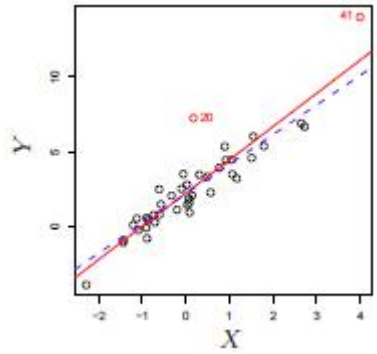
4. Non-constant variance of error terms

One assumption of the model is $Var(\varepsilon_i) = \sigma^2$, heteroscedasticity may violate. Thus one way is to take log(Y) or square-root of Y and it would shrink the large values a lot and reduce the trend/variance of errors.

We can also use Weighted Least Squares! If each of raw observations is uncorrelated with variance σ^2, then their average has variance $\sigma_i^2 = \sigma^2 / n_i$, thus by using weighted least squares, with weights proportional to the inverse variance, $w_i = n_i$, it would work better.

5. High leverage points

observations with high leverage have an unusual value for xi, like Point 41:



Removing high leverage observations has a much more substantial impact on the least squares line. Thus need to identify high leverage points.

Define:

Simple linear regression: look for observations that outside of normal range.

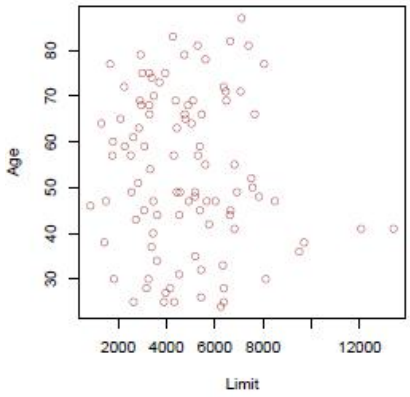
In multiple linear would be hard: quantify an observation's leverage. Compute the statistic:

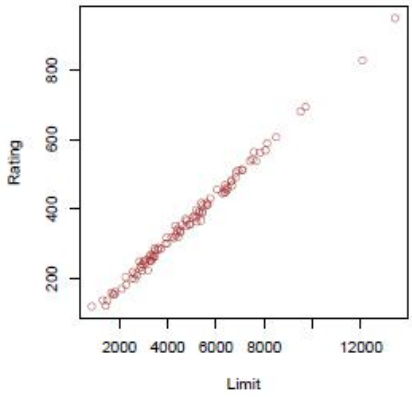
$h_i = \frac{1}{n} + \frac{(x_i - \bar{x})^2}{\sum_{i'=1}^{n} (x_{i'} - \bar{x})^2}$, we can extend to distance of xi from average of x for higher dimensions. The average of hi is equal to $(p+1)/n$, thus if hi is larger than this bound. Then it can be a high leverage point.

If a point is both outliers and high-leverage, it can be really dangerous!!!

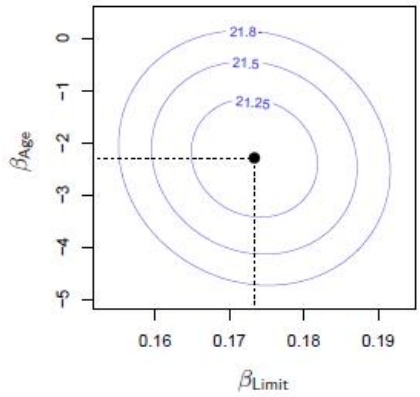
6. Col-linearity

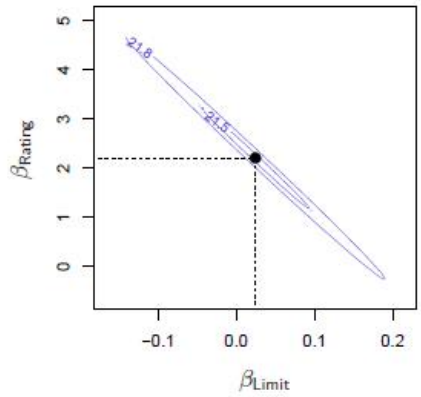
Col-linearity refers to the situation in which two or more predictor variables are closely related to one another.





Look at the contour plot:





The right one shows col-linearity since many pairs of beta coefficients can lead to quite similar RSS values. (Abnormal!)

Analysis: col-linearity reduces the accuracy of the estimates of the regression coefficient. It cause the standard error for beta to grow, then reduce the t-statistics. Thus p-value becomes larger and we may fail to reject H0 thus beta may be defined to be zero.

The power of hypothesis test: the probability of correctly detecting a non-zero coefficient -- is reduced by col-linearity

Identify the col-linearity:

1. Look at the correlation matrix of the predictors. Otherwise may be multiple col-linearity problem.
2. Compute variance inflation factor(VIF):

$VIF(\hat{\beta}_j) = \frac{1}{1 - R^2_{X_j|X_{-j}}}$, R^2 from a regression of X_j onto all of the other predictors. If R-square is close to one, then col-linearity appears and VIF would be larger. A VIF that exceeds 5 or 10 indicates a col-linearity.

- Classification

1. Introduction

Given a feature vector X and a qualitative response Y in the set C.

Predict its classification label. Often we estimate the probabilities that X to C.

Use box-plot to give the intuitive description of classes first

For two classes 0 and 1, use linear regression to predict y and check larger than 0.5 or not for the class prediction. *But linear regression may produce negative values. Thus Logistic regression should be better!!*

For multiple classes, linear regression not suitable.

2. Logistic regression

Write $p(X) = \Pr(Y = 1 \mid X)$, then $p(X) = \frac{e^{\beta_0 + \beta_1 X}}{1 + e^{\beta_0 + \beta_1 X}}$, then p(x) would always be in zero and one. One transformation:

$\log\left(\frac{p(X)}{1 - p(X)}\right) = \beta_0 + \beta_1 X$, log odds or logit transform of p(X)

Use Maximum Likelihood method, find suitable beta to maximize:

$l(\beta_0, \beta) = \prod_{i:yi=1} p(x_i) \prod_{i:yi=0} (1 - p(x_i))$

Logistic regression ensures that our estimation lies in 0 and 1.

Once you get the estimated coefficient, calculate the probability and compare.

3. Multivariate Logistic regression

$$\log\left(\frac{p(X)}{1 - p(X)}\right) = \beta_0 + \beta_1 X_1 + \ldots + \beta_P X_P$$

$$p(X) = \frac{e^{\beta_0 + \beta_1 X_1 + \ldots + \beta_P X_P}}{1 + e^{\beta_0 + \beta_1 X_1 + \ldots + \beta_P X_P}}$$

4. Multiple class logistic regression

Case-control sampling and logistic regression

For example, we get the logistic model for the whole dataset, then for certain subset/region, the only incorrect term is the intercept.

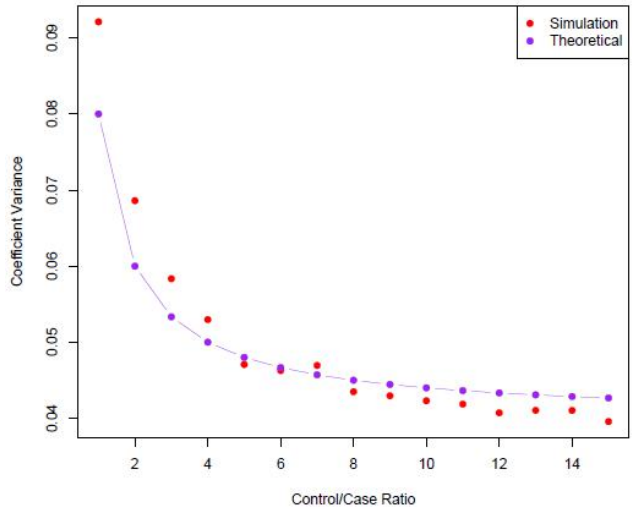
Pick a nice balanced sample and then correct the intercept to take into account the fact that you've selected on the dependent variable to learn more about rarer classes than a random sample would be able to tell you:

E.g: there are 160 cases, 302 controls -- $\tilde{\pi}$ is equal to 0.35, while in MI, the prevalence π is only 0.05, then no need to change other beta except for intercept to correct the logistic model for the MI region.

Write $\tilde{\pi}$ as the probability of conditional Y=1 in the whole set, and π as the probability of conditional Y=1 in subset, then:

$$\hat{\beta}_0^* = \hat{\beta}_0 + \log\frac{\pi}{1 - \pi} - \log\frac{\tilde{\pi}}{1 - \tilde{\pi}}$$

Also, often cases are rare and we take them all; up to five times that number of controls is sufficient. Sampling more controls than cases would reduce the variance of the parameter estimates but after a ratio of about 5 to 10, the variance reduction would be smaller and smaller:



Logistic regression with more than two classes:

$$\Pr(Y = k \mid X) = \frac{e^{\beta_{0k} + \beta_{1k} X_1 + \ldots + \beta_{Pk} X_P}}{\sum_{l=1}^{K} e^{\beta_{0l} + \beta_{1l} X_1 + \ldots + \beta_{Pl} X_P}}$$

Thus there is a linear function for each class

this is also called multinomial regression

Extremely unbalanced binary data issue: Case-control study design!!!!

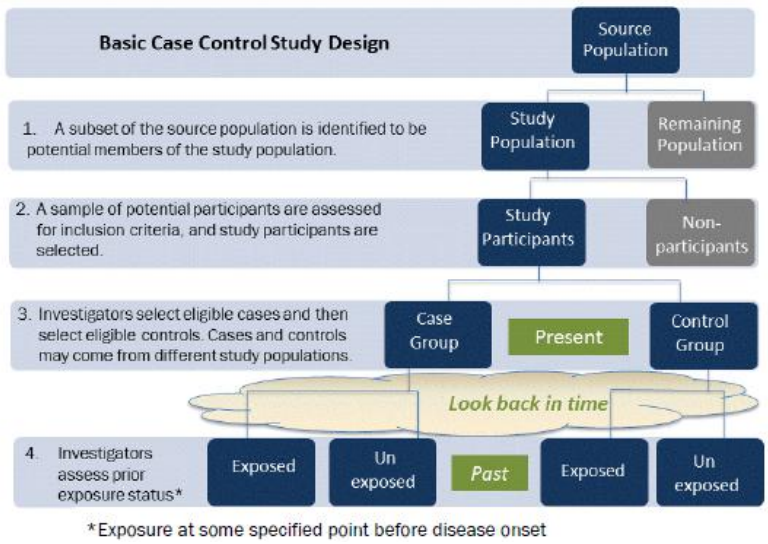
Do we need to use all the data? No, take the sample!

Case-control studies are used to determine if there is an association between an exposure and a specific health outcome. These studies proceed from effect (e.g. health outcome, condition, disease) to cause (exposure). Case-control studies assess whether exposure is disproportionately distributed between the cases and controls, which may indicate that the exposure is a risk factor for the health outcome under study. Case-control studies are frequently used for studying rare health outcomes or diseases.

At baseline:

Selection of cases and controls based on health outcome or disease status

Exposure status is unknown



	Diseased	Non-diseased
Exposed	a	b
Non-exposed	c	d

The exposure odds ratio (OR).*:
Case-Control Study
OR = (a/c)/(b/d) = (a/b)/(c/d)

Interpret the odd ratio:

The odds ratio is interpreted the same way as other ratio measures (risk ratio, rate ratio, etc.).

For example, investigators conducted a case-control study to determine if there is an association between colon cancer and a high fat diet. Cases were all confirmed colon cancer cases in North Carolina in 2010. Controls were a sample of North Carolina residents without colon cancer. The odds ratio was 4.0. This odds ratio tells us that individuals who consumed a high fat diet have four times the odds of colon cancer than do individuals who do not consume a high-fat diet.

Popular ways to select controls:

1. Base or case base sampling

   Using controls selected from the source population such that each person has the same chance of being selected. This type of sampling only works with a previously defined cohort. The odds here provides a valid estimate of the risk ratio without assuming that the disease is rare in the source population.

2. Cumulative density or survivor sampling

   When controls are sampled from those people who remained free of the health outcome at the end of follow-up then we call the sampling cumulative density sampling or survivor sampling. Controls can not ever become the cases when using this type of sampling. The odd ratio estimates the rare ratio only if the health outcome is rare.

3. Incidence density sampling or risk set sampling

   When cases are incident cases and when controls are selected from the at-risk source population at the same time as cases occur (controls must be eligible to become a case if the health outcome develops in the control at a later time during the period of observation) then we call this type of sampling incidence density sampling or risk set sampling. The control series provides an estimate of the proportion of the total person-time for exposed and unexposed cohorts in the source population. In these case-control studies, the odds ratio estimates the rate ratio of cohort studies, without assuming that the disease is rare in the source population

Note that it is possible, albeit rare, that a control selected at a later time point could become a case during the remaining time that the study is running.

This differs from case-control studies that use cumulative density sampling or survivor sampling, which select their controls after the conclusion of the study from among those individuals remaining at risk.

Selecting controls in a risk set sampling or incidence density sampling manner provides two advantages:

1. A direct estimate of the rate ratio is possible.

2. The estimates are not biased by differential loss to follow up among the exposed vs. unexposed controls.

For example, if a large number of smokers left the source population after a certain time point, they would not be available for selection at the end of the study – when controls would be selected in a study that uses cumulative density sampling or survivor sampling. This would give the investigators biased information regarding the level of exposure among the controls over the course of the study.

Source of Controls:

1. Population controls are non-cases sampled from the source population giving rise to cases.
2. Neighborhood or friend controls are appropriate for selection as controls if these individuals would be included as cases if they developed the health outcome of interest. It is not appropriate to select neighbors or friends as controls if they share the exposure of interest.

3. Hospital controls - There are certain problems with hospital controls in that they may not be from the same source population from which the cases arose.

Advantages of case-control studies:

Case-control studies are the most efficient design for rare diseases and require a much smaller study sample than cohort studies. Additionally, investigators can avoid the logistical challenges of following a large sample over time. Thus, case-control studies also allow more intensive evaluation of exposures of cases and controls. Case-control studies that use incidence density sampling or risk set sampling yield a valid estimate of the rate ratio derived from a cohort study if incident cases are studied and controls are sampled from the risk set of the source population. If properly performed (i.e. appropriate sampling), case-control studies provide information that mirrors what could be learned from a cohort study, usually at considerably less cost and time.

Disadvantages of case-control studies:

Case-control studies do not yield an estimate of rate or risk, as the denominator of these measures is not defined. Case-control studies may be subject to recall bias if exposure is measured by interviews and if recall of exposure differs between cases and controls. However, investigators may be able to avoid this problem if historical records are available to assess exposure. Choosing an appropriate source population is also difficult and may contribute to selection bias.

Terminology:

Cohort studies:

An observational study in which subjects are sampled based on the presence (exposed) or absence(unexposed) of a risk factor of interest. These subjects are followed over time for the development of a health outcome of interest.

Cross-sectional studies:

An observational study in which subjects are sampled at one point in time, and then the associations between the concurrent risk factors and health outcomes are investigated.

Exposure odds ratio (OR):

the odds of a particular exposure among persons with a specific health outcome divided by the corresponding odds of exposure among persons without the health outcome of interest. Yields a valid estimate of the incidence rate ratio or risk ratio derived from a cohort study, depending on control sampling.

Incident case:

a person who is newly diagnosed as a case.

Prevalent case:

a person who has a health outcome of interest that was diagnosed in the past.

Risk ratio (RR):

the likelihood of a particular health outcome occurrence among persons exposed to a given risk factor divided by the corresponding likelihood among unexposed persons.

Source population:

the population out of which the cases arose.

Example: we have the total set:

	Diseased	Non-diseased	Total
Exposed	7	1000	1007
Non-exposed	6	5634	5640

Therefore, the incidence in the exposed individuals would be 7/1,007 = 0.70%, and the incidence in the non-exposed individuals would be 6/5,640 = 0.11%. Consequently, the risk ratio would be 0.70/0.11=6.52, suggesting that those who had the risk factor (exposure) had 6.5 times the risk of getting the disease compared to those without the risk factor. This is a strong association. Now the problem is: we may not have the source of all the data (whole population).And if I take a sample, maybe none of the cases would be included.

OR = 1 Odds of disease is the same for exposed and unexposed

OR > 1 Exposure increases odds of disease

OR < 1 Exposure reduces odds of disease

One way to solve: find these rare cases in the source population and determine the exposure status. Then data: (estimate the risk factor)

	Diseased	Non-diseased	Total
Exposed	7	10	unknown
Non-exposed	6	56	unknown

With this sampling approach I can no longer compute the probability of disease in each exposure group, because I no longer have the denominators in the last column. In other words, I don't know the exposure distribution for the entire source population. However, the small control sample of non-diseased subjects gives me a way to estimate the exposure distribution in the source population. So, I can't compute the probability of disease in each exposure group, but I can compute the odds of disease in each group.

The Odds Ratio (number of cases/number of not cases)

The odds of disease in the exposed group are 7/10, and the odds of disease in the non-exposed group are 6/56. If I compute the odds ratio, I get (7/10) / (5/56) = 6.56, very close to the risk ratio that I computed from data for the entire population. We will consider odds ratios and case-control studies in much greater depth in a later module. However, for the time being the key things to remember are that:

1. The sampling strategy for a case-control study is very different from that of cohort studies, despite the fact that both have the goal of estimating the magnitude of association between the exposure and the outcome.
2. In a case-control study there is no "follow-up" period. One starts by identifying diseased subjects and determines their exposure distribution; one then takes a sample of the source population that produced those cases in order to estimate the exposure distribution in the overall source population that produced the cases. [In cohort studies none of the subjects have the outcome at the beginning of the follow-up period.]
3. In a case-control study, you cannot measure incidence, because you start with diseased people and non-diseased people, so you cannot calculate relative risk.
4. The case-control design is very efficient. In the example above the case-control study of only 79 subjects produced an odds ratio (6.56) that was a very close approximation to the risk ratio (6.52) that was obtained from the data in the entire population.
5. Case-control studies are particularly useful when the outcome is rare is uncommon in both exposed and non-exposed people.

Sampling methods in scientific study:

Cross-sectional data: represent a point in time, intend to look at prevalence rates and risk factors.

Cohort: subjects selected according to exposure, a group of people would share something in common. The cohort may be chosen according to exposure patterns but must be identified before disease status has been determined. Determination of disease status may be prospective/retrospective. It allows calculation of relative risk

Case-control: subjects selected according to outcome: cases and controls. Select according to disease outcome (cases and controls). The investigator looks back to determine exposure or risk factors. Necessarily retrospective so no need to wait for disease outcome.

5. Linear Discriminant Analysis

Discriminant Analysis:

model the distribution of X in each of the class separately and then use the Bayes Theorem to flip things around and obtain Pr(Y|X)

Under Normal distribution for each class, it leads to linear and quadratic discriminant analysis.

Bayes Theorem: Flip things around

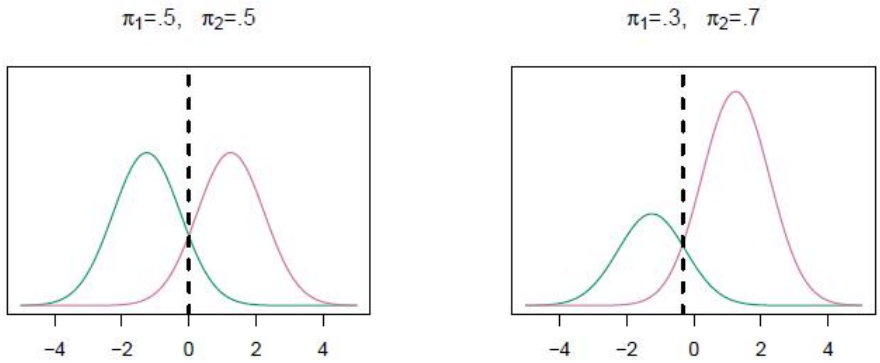
$$\Pr(Y=k \mid X=x)=\frac{\Pr(X=x \mid Y=k)\Pr(Y=k)}{\Pr(X=x)}$$

Transform for discriminant analysis:

$\Pr(Y=k \mid X=x)=\frac{\pi_k f_k(x)}{\sum_{l=1}^{K}\pi_l f_l(x)}$, $f_k(x)=\Pr(X=x \mid Y=k)$, is the density for X in class k. $\pi_k=\Pr(Y=k)$ is the marginal or prior probability for class k.

Classify to the highest density: compare the $\pi_k f_k(x)$

Now plug in the Gaussian Density:



Why discriminant analysis:

1. If classes are well separated, the parameters estimates for logistic regression model are very surprisingly unstable. LDA can over come this problem.

2. If n is small and distribution of predictors X is approximately normal in each of the classes, LDA is more stable than logistic regression.

3. LDA is popular for multiple classes classification since it provides low-dimensional views of the data.

6. Uni-variate Linear Discriminant Analysis

Linear discriminant analysis when p=1:

The Gaussian Density has the form:

$f_k(x)=\frac{1}{\sqrt{2\pi}\sigma_k}e^{-\frac{1}{2}\left(\frac{x-u_k}{\sigma_k}\right)^2}$, using the mean and variance of x in class k

Here we assume $\sigma_k=\sigma$ for all the classes

Plug into the Bayes Formula:

$$\Pr(Y=k \mid X=x)=p_k(x)=\frac{\pi_k \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{1}{2}\left(\frac{x-u_k}{\sigma}\right)^2}}{\sum_{l=1}^{K} \pi_l \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{1}{2}\left(\frac{x-u_l}{\sigma}\right)^2}}$$

To find the largest pk(x), simplify the form and get the discriminant score:

Taking logs and discard terms not dependent on k:

Score: $\delta_k(x)=x\frac{u_k}{\sigma^2}-\frac{u_k^2}{2}$

Compare the score and find the largest:

Estimate the mean, variance and proportion for the real dataset with histogram and other tools. How to do in details:

$$\begin{aligned}
\hat{\pi}_k &= \frac{n_k}{n} \\
\hat{\mu}_k &= \frac{1}{n_k}\sum_{i:\, y_i=k} x_i \\
\hat{\sigma}^2 &= \frac{1}{n-K}\sum_{k=1}^{K}\sum_{i:\, y_i=k}(x_i-\hat{\mu}_k)^2 \\
&= \sum_{k=1}^{K}\frac{n_k-1}{n-K}\cdot\hat{\sigma}_k^2
\end{aligned}$$

where $\hat{\sigma}_k^2=\frac{1}{n_k-1}\sum_{i:\, y_i=k}(x_i-\hat{\mu}_k)^2$ is the usual formula for the estimated variance in the kth class.

Idea: calculate the variance within each class first and try to average them to fit the uniform variance assumption.

7. Multivariate Linear Discriminant Analysis

Linear Discriminant Analysis when P>1:

Introduce the multiple normal density and its discriminant score:

$$\text{Density: } f(x)=\frac{1}{(2\pi)^{p/2}|\mathbf{\Sigma}|^{1/2}}e^{-\frac{1}{2}(x-\mu)^T\mathbf{\Sigma}^{-1}(x-\mu)}$$

$$\text{Discriminant function: } \delta_k(x)=x^T\mathbf{\Sigma}^{-1}\mu_k-\frac{1}{2}\mu_k^T\mathbf{\Sigma}^{-1}\mu_k+\log\pi_k$$

This discriminant function is still linear in variable X thus form is simple

When there are K classes, linear discriminant analysis can be viewed exactly in a K-1 dimensional plot because it classifies to the closest center and span a K-1 dimensional plane.

From discriminant values to probabilities:

$$\widehat{\Pr}(Y=k|X=x)=\frac{e^{\hat{\delta}_k(x)}}{\sum_{l=1}^{K}e^{\hat{\delta}_l(x)}}.$$

Thus classify to the largest discriminant value amounts to classifying the largest probabilities shown above.

Calculate the misclassification error for the training data --- may be over-fitting:

We may also use the Naive Bayes Classifier and classify to the larger class, then for 1000+ ones and several zeros, the error would be still small.

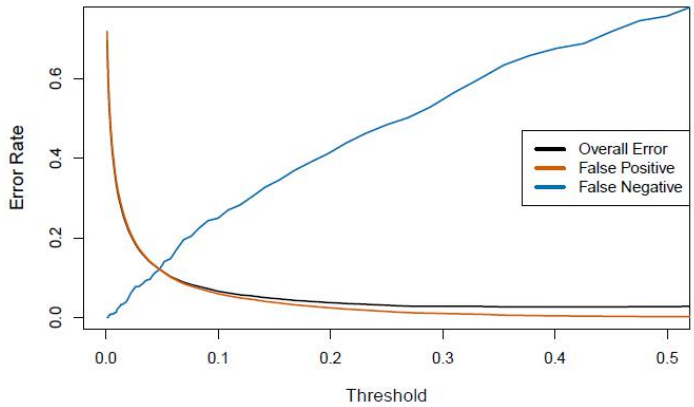
Types of errors:

False Positive rates: fraction of negative examples that are classified as positive

False negative rates: fraction of positive examples that are classified as negative

We should separate these two kinds of errors to give accurate inference.

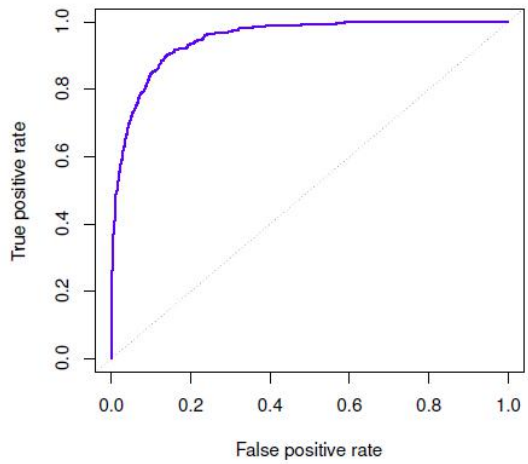
For these two classes, if we change the value threshold from 0.5 to others, the Ratio of two errors would change! And we can decrease and balance:



Also, we can check the ROC plot:

45 percent line is no information line, want higher true positive rate and lower false positive rate:

Sometimes we use the AUC or area under the curve to summarize the overall performance, higher AUC is better.



8. Quadratic Discriminant Analysis

With Gaussian but different $\sum_K$ in each class, we get quadratic discriminant analysis.

$$\Pr(Y=k \mid X=x)=\frac{\pi_k f_k(x)}{\sum_{l=1}^{K}\pi_l f_l(x)},\ f_k(x)=\Pr(X=x \mid Y=k)$$

$$f_k(x)=\prod_{j=1}^{p} f_{jk}(x_j)$$

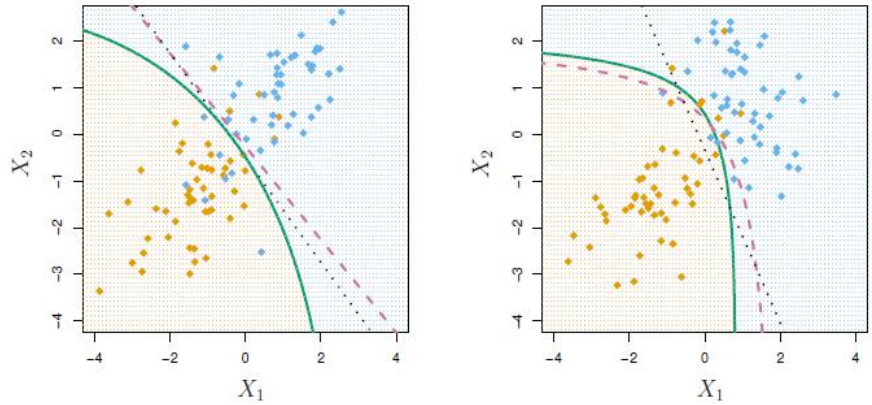
Conditional independence model in each class, we get Naive Bayes.

For Gaussian, this means that the co-variance matrix would be different/diagonal.

Now the discriminant score:

$$\delta_k(x) = -\frac{1}{2}(x-u_k)^T \sum_k^{-1}(x-u_k) + \log \pi_k - \frac{1}{2}\log|\Sigma_k|$$

Since the co-variance now is different, the quadratic terms matter.



The left one is with true boundary linear, right one is with quadratic one.

Plug in x1 and x2, calculate the discriminant values of points in the two dimensional space and find the estimated boundary shown above.

Quadratic DA is significantly better when the real one is quadratic as well.

Naive Bayes:

Assume features are independent in each class, useful when p is very large since multivariate methods make QDA and LDA break down.

Gaussian naive Bayes assumes each $\sum_k$ is diagonal:

$$\delta_k(x) \propto \log\left[\pi_k \prod_{j=1}^{p} f_{kj}(x_j)\right] = -\frac{1}{2}\sum_{j=1}^{p}\frac{(x_j-\mu_{kj})^2}{\sigma_{kj}^2} + \log \pi_k$$

It can be used for mixed feature vectors (qualitative and quantitative). If features are qualitative, replace the density with probability mass function over discrete categories.

Probability mass function is to give the probability that a discrete random variable is exactly equal to some values.

Now Comparison of Logistic Regression and LDA:

For LDA, if we take the logit form of the probability, the logit still has linear relationship with features, thus it has the same form.

$$\log\left(\frac{p_1(x)}{1-p1(x)}\right) = c0 + c1x1 + \ldots + cpxp$$

The difference is in how the parameters are estimated. Logistic use conditional likelihood method but LDA use the full likelihood on Pr(X, Y).

Some questions/answers for KNN:

1. KNN spends more time on computation test rather than train. Since training is just storing things and labels. Test would be used to calculate
2. Find the value of K with smallest test error
3. All distance (Manhattan, MINKOWSKI, TANIMOTO, JACCARD, MAHALANOBIS) can be used for KNN algorithm, Manhattan is used for continuous distance. Hamming distance can be used for KNN for categorical variables.
4. It can be used for both regression and classification, mean or the median to estimate the result of prediction
5. KNN algorithm works better if all of the data have same scale, with smaller number of variables p and no assumptions about the functional form of problem being solved.
6. KNN algorithm can be used for imputing missing value of both categorical and continuous variables.
7. **Large K means simple model, simple model always consider as higher bias but lower variance!!**
8. We can try increase the value of k to decrease noise in the data. In KNN, we can face the over-fitting problem due to the curse of dimensions. Dimension reduction and feature selections can reduce the problem!
9. KNN is memory-based approach is that the classifier immediately adapts as we collect new training data.
10. The computational complexity for classifying new samples grows linearly with the number of samples in the training dataset in the worst-case scenario.
11. In case of very large value of k, we may include points from other classes into the neighborhood. In case of too small value of k the algorithm is sensitive to noise.
12. The decision boundary is not linear. KNN does not require an explicit training step. The decision boundary would become smoother by increasing the value of K. We can choose optimal value of k with the help of cross validation. Euclidean distance treats each feature as equally important.
13. Calculating the distance between 2 observation will take D time, N*D time for N observations in test data. The training time for any value of k in KNN algorithm is the same.

- Cross Validation

1. Prediction error and validation set

Cross-validation and bootstrap re-sampling methods:

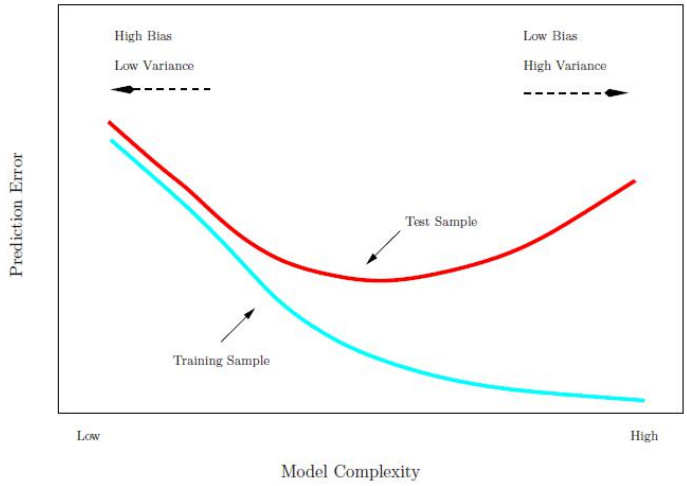
These methods refit a model of interest to samples formed from the training set, in order to obtain additional information about the fitted model.

Recall the distinction between the test error and the training error:

Test error: average error from the prediction

Training error: error from the observations used in its training

Generally speaking, the training error can be dramatically underestimate the test error.



Face the over-fitting problem.

The lower model complexity indicates higher bias and lower variance. Since as the model complexity goes up, more parameters would be estimated thus the variance goes up. More factors have been considered thus lower bias.

More on prediction-error estimates:

Best solution: a large designated test set. Often not available.

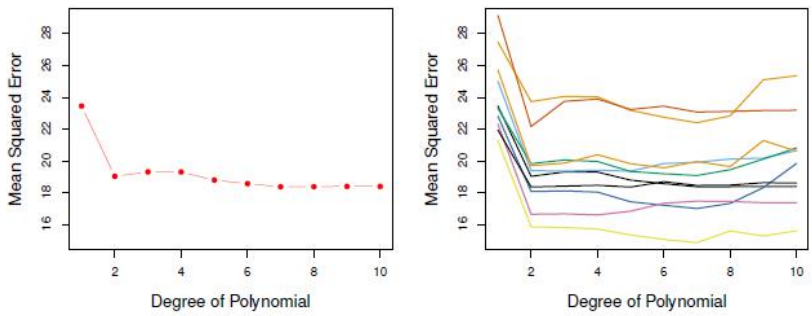
But we can make a mathematical adjustment to the training error rate to estimate the test error rate. Cp statistic, AIC and BIC.

Also, we can hold out a subset of training observations from the fitting process, then apply statistical learning methods to those held out observations.

Two part validation:

Random splitting into two parts, blue is training set, right is validation/hold out set.

It is wasteful if the set is small. We try and also randomly split into two parts for many more times on a polynomial regression. Here the variance varies a lot because of the random splitting.



Drawback of validation set approach:

Test error would vary a lot. Only a subset is used to train the model. The validation set error may tend to overestimate the test error for the model fitted on the whole data set. Larger training set should have lower errors!

2. K-Fold Cross-validation

Widely used approach

Idea is to randomly divide the data into K equal-sized parts. We leave out part k, fit the model to the other K-1 parts and then obtain predictions for the left-out part. This is done in turn for each part k=1,2,...,K and then the results are combined.

1	2	3	4	5
Validation	Train	Train	Train	Train

Details: let the K parts be C1, C2,...,CK where Ck denotes the indices of the observations in part k. If N is a multiple of K, then $n_k = n / K$

Compute: $CV_{(K)} = \sum_{k=1}^{K} \frac{n_k}{n} MSE_k, MSE_k = \sum_{i \in C_k} (y_i - \hat{y}_i) / n_k$,

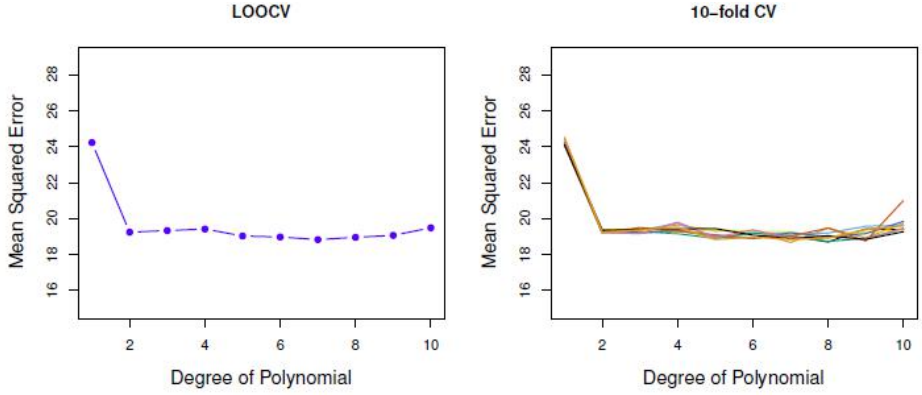
that is the weighted average of all mean standard errors.

If K=n, then this is the leave-one out cross validation! LOOCV

LOOCV sometimes useful but doesn't shake up the data enough. Since almost all of them are quite correlated thus their average can have high variance/low bias

A better choice for K may be 5 or 10.

More constant for 10-cross validation:



Some issues with the cross-validation:

Since training set is only (K-1)/K as big as the original training set, the estimation of prediction error would be biased upwards since less data means less complexity and higher bias compared with the whole dataset.

The bias is minimized for LOOCV but it has higher variance.

Cross validation can also be used for classification problems:

$$CV_K = \sum_{k=1}^{K} \frac{n_k}{n} Err_k, Err_k = \sum_{i \in C_k} I(y_i \neq \hat{y}_i) / n_k$$

And the estimated standard deviation of it is:

$$SE(CV_K) = \sqrt{\sum_{k=1}^{K} \left(Err_k - \overline{Err_k} \right)^2 / (K-1)}$$

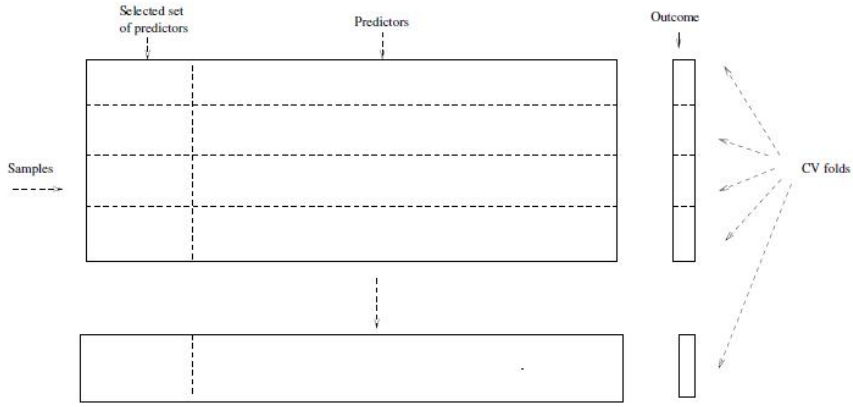
3. Cross-Validation Do's and Do not

Right and wrong notice:

E.g: 5000 predictors and 50 samples, find the most correlated 100 predictors Cross validation assumes that your classifier has been perfect.

Correct way to do it:

We have to kick out the validation set when we choose the predictors and then calculate the cross validation errors for the whole. Otherwise, validation set information would also be included and errors are underestimated.



4. Bootstrap1

Powerful and flexible tool to quantify the uncertainty associated with a given estimator or statistical learning method.

Example: invest on two assets with returns of X and Y. invest alpha in X and 1-alpha in Y and minimize Var(alpha*X+(1-alpha)*Y)

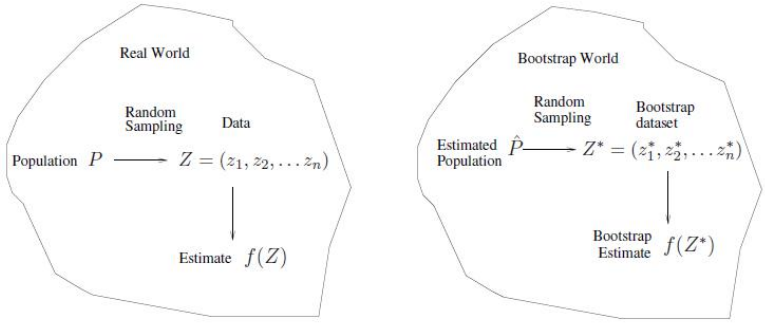
Then the formula is:

$$a = \frac{\sigma_Y^2 - \sigma_{XY}}{\sigma_X^2 + \sigma_Y^2 - 2\sigma_{XY}}, \sigma_X^2 = Var(X), \sigma_Y^2 = Var(Y), \sigma_{XY} = Cov(X, Y)$$

But we need to estimate these variance quantities. We can simulate 100 pairs of X and Y and estimate alpha and repeat for many many times like 1000. And we can get the distribution of alpha as well. And calculate to get the mean and standard error by these 1000 results.

Back to the real problem: we can not generate the data by ourselves but we can sample the data from the whole set with replacement. **But all the sample size has exactly the same size of original set. It means there may be repeated samples!!**

5. Bootstrap2



In bootstrap process, we estimate our model on estimated population not the real population.

For more complicated cases: time series cases

If the data is a time series, we ca not simply sample the observations with replacement since the data are not independent. We assume that sampling is i.i.d from the whole set. Then for time series, they use block bootstrap. Across these blocks, they are independent and put them together to form a new time series.

Other uses of the bootstrap:

Confidence interval:

From that previous 1000 results, we can provide a confidence interval from the Histogram shown.The confidence interval would contain the true value 90% of time.

The above interval is called a Bootstrap percentile confidence interval. Simplest.

For cross validation, no overlapping parts. Clean separation between sets. But each bootstrap sample has significant overlap with the original set. About 2/3 of the original data points appear in each bootstrap sample. Some of the observations are not novel for the model.

This will cause the bootstrap underestimate the true prediction error.

- Model selection
  1. Linear model subset selection

     Linear model selection and regularization

     Some ways to extend the linear model framework.

     Why consider alternatives to least squares?

     1. Prediction accuracy: when p>n, control the variance
     2. Model interpretation: remove irrelevant features -- feature selection

     Three classes of methods:

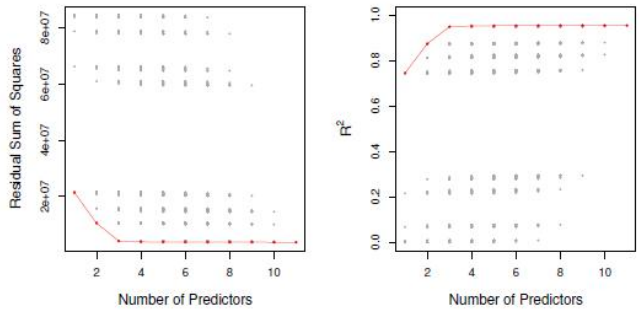
     1. Subset selection

        Find p predictors that we believe to be related to the response

        Consider null model first and try all possible subsets of predictors. The best subset selection means the one with smallest RSS or largest R-Square

        Number of sets: $\binom{p}{m} = \frac{p!}{m!(p-m)!}$

        Choose the one based on cross-validation prediction error, Cp, AIC, BIC and R-square. Based on the Test Error!!!



Picture shown above indicates the relationship of number of predictors and RSS/R^2, points are different combinations of predictors.

More predictors would make things better but it can be flat.

2. Shrinkage

Fit a model involving all p predictors then shrink some coefficient to zero relative to the least squares estimates. Shrinkage/regularization has the effect of reducing variance and can perform variable selection.

3. Dimension reduction

Project p predictors into a M-dimensional subspace, where M<p.

2. Forward stepwise selection

Subset selection can not be applied with very large p.
too many combinations.

And enormous search space can lead to over-fitting and high variance of the coefficient estimates.

Start with a model with no predictors and add predictors to the model one at a time until all of the predictors are in the model. At each step, the variable that gives the best improvement would be added.

Algorithm: consider all single predictors one by one based on previous model during each loop.

Forward Stepwise Selection

1. Let $\mathcal{M}_0$ denote the *null* model, which contains no predictors.
2. For $k = 0, \ldots, p-1$:
   - 2.1 Consider all $p-k$ models that augment the predictors in $\mathcal{M}_k$ with one additional predictor.
   - 2.2 Choose the *best* among these $p-k$ models, and call it $\mathcal{M}_{k+1}$. Here *best* is defined as having smallest RSS or highest R^2.
3. Select a single best model from among $\mathcal{M}_0, \ldots, \mathcal{M}_p$ using cross-validated prediction error, C_p (AIC), BIC, or adjusted R^2.

It has computational advantage over best subset selection:
1+p+p-1+p-2... = p^2 v.s 2^p models

But it does not guarantee the best possible model for p predictors since it does not search all the models compared with best subset selection.

3. Backward stepwise selection

An alternative to forward stepwise selection

It begins with the all predictors. Delete the most useless predictor and keep that K-1 model to continue:

Backward Stepwise Selection

1. Let $\mathcal{M}_p$ denote the *full* model, which contains all p predictors.
2. For $k = p, p-1, \ldots, 1$:
   - 2.1 Consider all k models that contain all but one of the predictors in $\mathcal{M}_k$, for a total of $k-1$ predictors.
   - 2.2 Choose the *best* among these k models, and call it $\mathcal{M}_{k-1}$. Here *best* is defined as having smallest RSS or highest R^2.
3. Select a single best model from among $\mathcal{M}_0, \ldots, \mathcal{M}_p$ using cross-validated prediction error, C_p (AIC), BIC, or adjusted R^2.

There are about 1+p(p+1)/2 models to consider in total so it can be applied when p is too large to apply best subset selection.

It can not guarantee to have the best possible model for p predictors.

And it requires the number of samples n is larger than p.

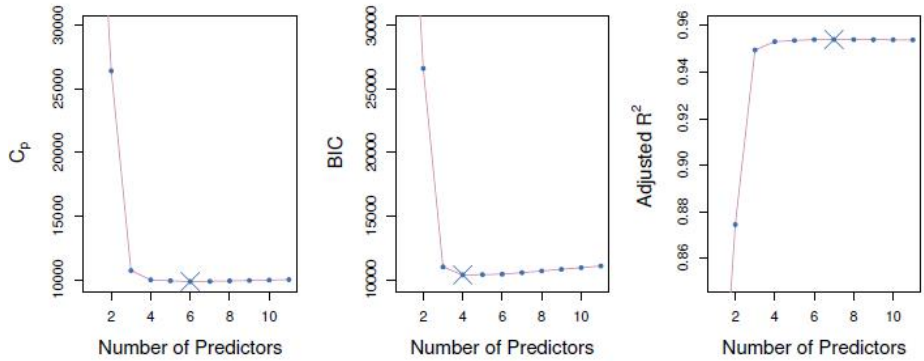
The model contains all predictors would always have the smallest RSS and the largest R-square. We want to find a model with low test error not training error. Thus RSS and R-square are not suitable with different No. Of predictors.

4. Estimating test error -- Mallow Cp, AIC, BIC and Adjusted R-squared

Two approaches:

adjustment to the training error to account for bias or directly estimate the test error in CV or validation set.

All these methods are to adjust the training error.



We want smaller Cp and BIC but larger R-square. These figures become flat.

Simpler is better!!

Mallow's Cp:

$$C_p = \frac{1}{n}\left(\mathrm{RSS} + 2d\hat{\sigma}^2\right),$$

where d is the total # of parameters used and $\hat{\sigma}^2$ is an estimate of the variance of the error ϵ associated with each response measurement.

For the M3 for example, there are 4 parameters in total (intercept).

Thus we want the smallest Cp for the model.

AIC:

defined for a large class of models fit by maximum likelihood:

AIC = -2logL + 2d, L is the maximized value of the likelihood function for the estimated model and d is the total number of predictors.

For linear models, AIC and Cp are proportional so they are the same for linear models but not for other models.

BIC:

$BIC = \frac{1}{n}\left(RSS + \log(n) d \hat{\sigma}^2\right)$, since log(n)>2 when n>7, so

BIC has heavier penalty on models' variables. Thus it leads to smaller models compared with Cp and AIC.

Adjusted R-square:

$R^2 = \frac{TSS - RSS}{TSS} = 1 - \frac{RSS}{TSS}, TSS = \sum_{i=1}^{n}\left(y_i - \bar{y}\right)^2$ is the total sum of squares

$RSS = \sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2 = \sum_{i=1}^{n}\left(y_i - \hat{\beta}_0 - \hat{\beta}_1 x_{i1} - \ldots - \hat{\beta}_p x_{ip}\right)^2$

Adjusted R-square = $1 - \frac{RSS/(n-d-1)}{TSS/(n-1)}$, add penalty to number of predictors to the R-square.

Smaller value of RSS/(n-d-1) does not mean for smaller RSS!!! Trade-off

Thus it pays a price for the value.

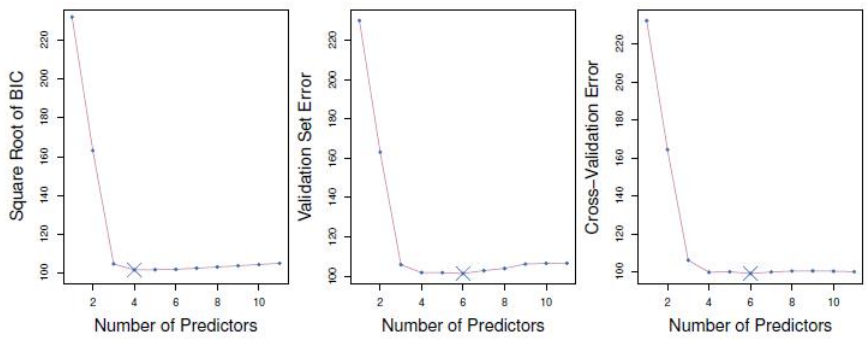
5. Estimating test error -- cross validation

Validation and cross-validation

Each of the procedures returns a sequence of models Mk indexed by model size k=0,1,2,.. Our job here is to select k. Once selected, we return model Mk.

We compute the validation set error or the cross-validation error for each model Mk under consideration, and then select the k for which the resulting estimated test error is the smallest.

This procedure has an advantage relative to AIC, BIC, Cp and adjusted R-square, it provides a direct estimate of the test error. No needs for the error variance estimation.



6. Ridge regression

Shrinkage methods: ridge regression and lasso

Least square:

$$RSS = \sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2 = \sum_{i=1}^{n}\left(y_i - \hat{\beta}_0 - \hat{\beta}_1 x_{i1} - \ldots - \hat{\beta}_p x_{ip}\right)^2$$

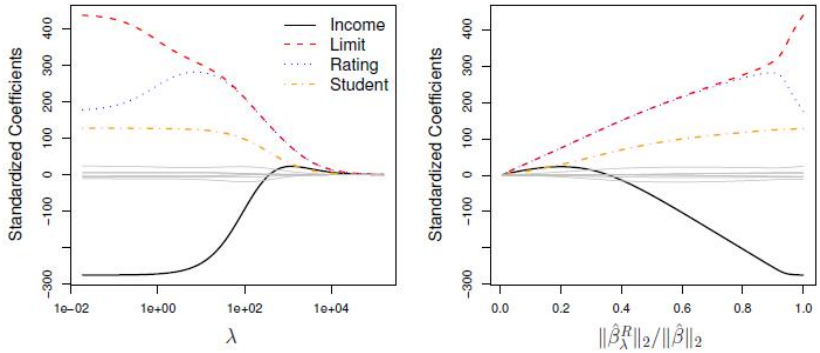
Ridge: minimize

$$\sum_{i=1}^{n}\left(y_i - \hat{\beta}_0 - \hat{\beta}_1 x_{i1} - \ldots - \hat{\beta}_p x_{ip}\right)^2 + \lambda\sum_{j=1}^{p}\beta_j^2 = RSS + \lambda\sum_{j=1}^{p}\beta_j^2$$

Here lambda is non-negative and a tuning parameter to be determined separately. It would avoid coefficient being too large

The second term is a shrinkage penalty term. If lambda is really large, the coefficients have to very large. Thus lambda would shrink beta towards zero. It control the relative impact of these two terms. Selecting lambda seriously!

But it doesn't select parameters by setting things to zero, just very small.



Larger lambda pushes all the coefficients on the left to zero finally. Cut at a certain lambda, we can decide the parameters eventually. The right one's x-axis is a normalized L-2 norm, lambda goes to smaller then. It is normalized by comparing with the least square coefficients estimation. Thus two plots flip.

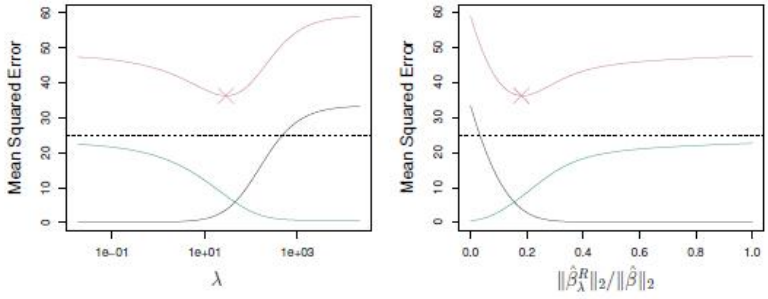
$$\|\beta\|_2 = \sqrt{\sum_{j=1}^{p}\beta_j^2}$$

Scaling in ridge: multiplying all predictors by constant would not influence the model. But multiplying a given predictor would influence.

Thus it is better to do the standardizing the predictors first here:

$$\bar{x}_{ij} = \frac{x_{ij}}{\sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(x_{ij} - \bar{x}_j\right)^2}}$$

Ridge regression improve over least squares: as lambda increases, it controls the coefficients thus lower the variance. Check below:



Squared bias black, variance green and MSE test purple

7. Lasso

Ridge will contain all p predictors in the final model eventually. Even small.

Lasso would overcome the disadvantage:

$$\sum_{i=1}^{n}\left(y_i-\hat{\beta}_0-\hat{\beta}_1x_{i1}-\ldots-\hat{\beta}_px_{ip}\right)^2+\lambda\sum_{j=1}^{p}\left|\beta_j\right|=RSS+\lambda\sum_{j=1}^{p}\left|\beta_j\right|$$

The lasso here uses L-1 penalty not L-2 penalty.

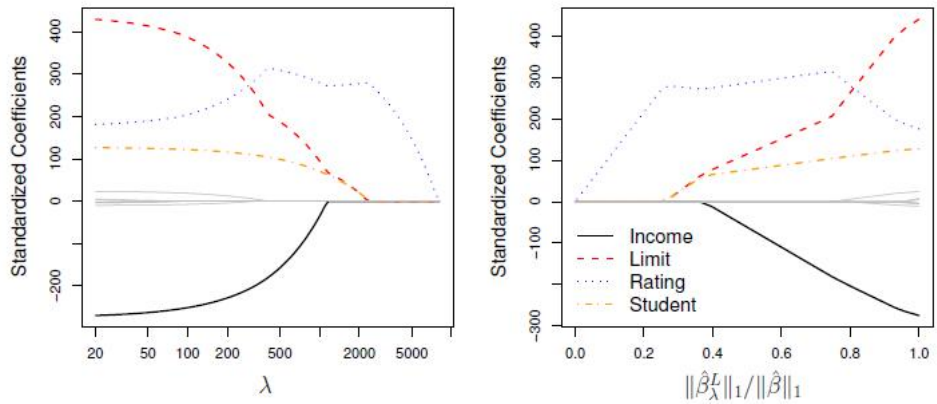
L-1 norm:

$$\left\|\beta\right\|_1=\sum\left|\beta_j\right|$$

When the lambda is large enough, it forces some coefficients to be zero!

Thus it performs variable selection. Convex optimization

Some coefficients are zero:



Why those coefficients are zero?

Compare with Ridge and lasso:

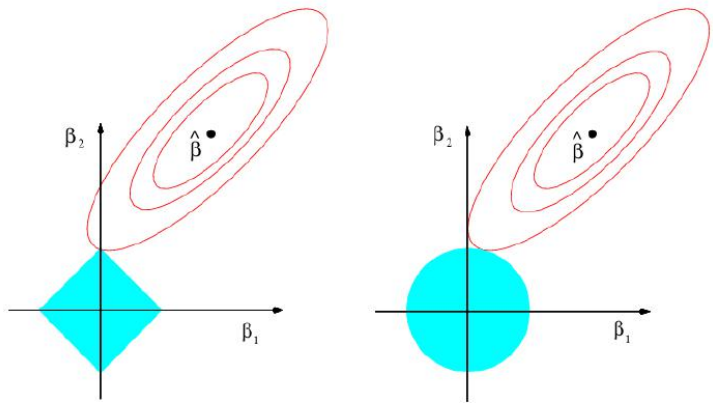
They are minimizing different things! Lasso gives the budget for things. For ridge, the budget is the sum of squares.

$$\underset{\beta}{\text{minimize}}\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_jx_{ij}\right)^2 \quad \text{subject to} \quad \sum_{j=1}^{p}|\beta_j|\le s$$

and

$$\underset{\beta}{\text{minimize}}\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_jx_{ij}\right)^2 \quad \text{subject to} \quad \sum_{j=1}^{p}\beta_j^2\le s,$$

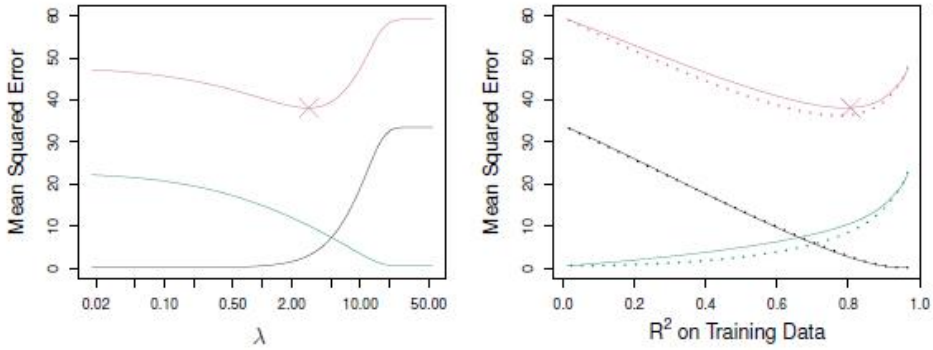
respectively.



The left one is the lasso and right Ridge. Two coefficients in the plot, The sum of squares would form a contour.

For Ridge, that would be circle, find the first point that the contour hits the circle. That the no coefficients can reach the zero exactly. But for Lasso, that region has corners and zeros can be reached.

One comparison example for Lasso and Ridge:



Left: Plots of squared bias (black), variance (green), and test MSE (purple) for the lasso.

Right: Comparison of squared bias, variance and test MSE between lasso (solid) and ridge (dashed). Both are plotted against their R2 on the training data, as a common form of indexing. The crosses in both plots indicate the lasso model for which the MSE is smallest.

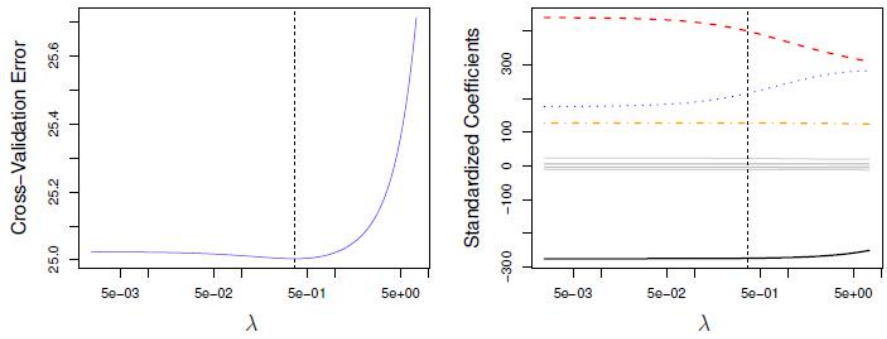
Conclusion: no correct answers to choose certain models. If the data is sparse, fewer predictors would be simpler then Lasso is good.

8. Tuning parameters

Selecting tuning parameter for ridge regression and Lasso

Cross validation method: choose a grid of lambda values and compute the cross validation error rate for each value of lambda. Choose the lambda with smallest cross-validation errors. Finally, the model needs to re-fit all the observations and the selected value of the tuning parameters.

Ridge example for tuning parameter:



Find suitable lambda above to make error small and decide coefficients

9. Dimension reduction

Dimension reduction methods: transform the predictors and then fit a least squares model using the transformed variables. Fewer parameters!!!

Detailed algorithm:

- Let $Z_1, Z_2, \ldots, Z_M$ represent $M < p$ *linear combinations* of our original p predictors. That is,

$$Z_m = \sum_{j=1}^{p} \phi_{mj} X_j \tag{1}$$

for some constants $\phi_{m1}, \ldots, \phi_{mp}$.

- We can then fit the linear regression model,

$$y_i = \theta_0 + \sum_{m=1}^{M} \theta_m z_{im} + \epsilon_i, \quad i = 1, \ldots, n, \tag{2}$$

using ordinary least squares.

Then we need to estimate new theta parameters, fewer things to estimate. By choosing proper parameters, dimension reduction can outperform OLS.

Transformation:

$$\sum_{m=1}^{M} \theta_m z_{im} = \sum_{m=1}^{M} \theta_m \sum_{j=1}^{p} \phi_{mj} x_{ij} = \sum_{j=1}^{p} \sum_{m=1}^{M} \theta_m \phi_{mj} x_{ij} = \sum_{j=1}^{p} \beta_j x_{ij},$$

where

$$\beta_j = \sum_{m=1}^{M} \theta_m \phi_{mj}. \tag{3}$$

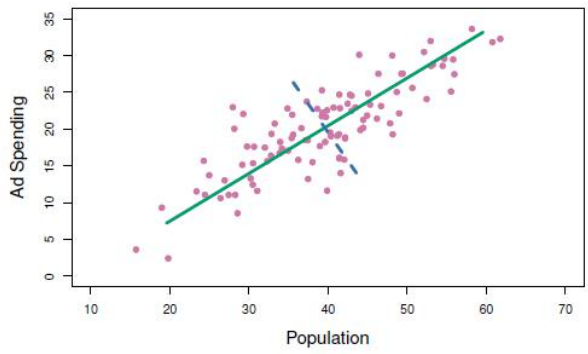
Thus the dimension reduction result is still the linear regression model based on original predictors but the coefficients have to take specific form.

10. Principal Components and Partial Least Squares

Principal components regression:

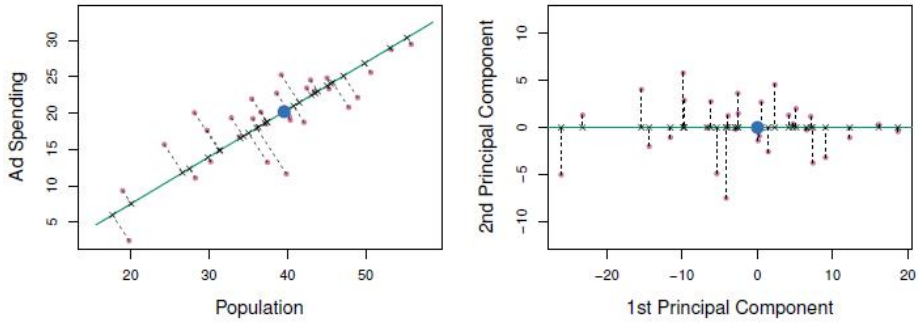
find principal components and operate linear regression on these components. Use new components to capture the most variance in the dataset.

Find the direction along the data that varies the most! And find the next one that is uncorrelated with the first one, the second one and so on.

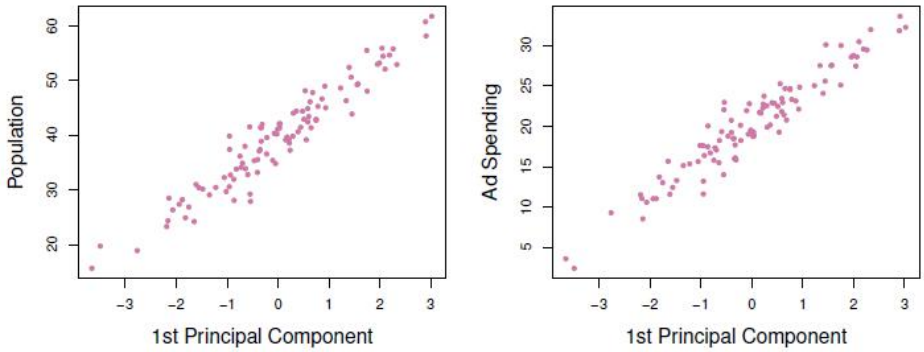


The first principal component would be chosen to minimize the sum of the squared perpendicular distances to each point. Data would vary along the line:

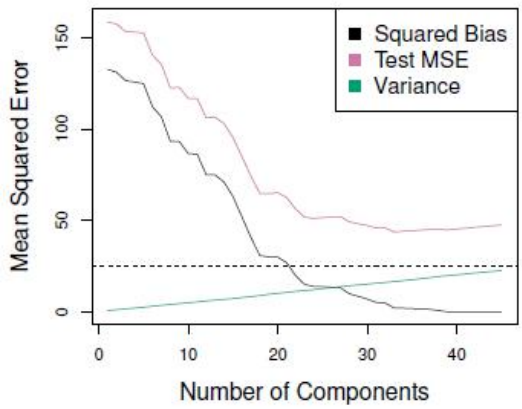
The perpendicular distances have been shown above:



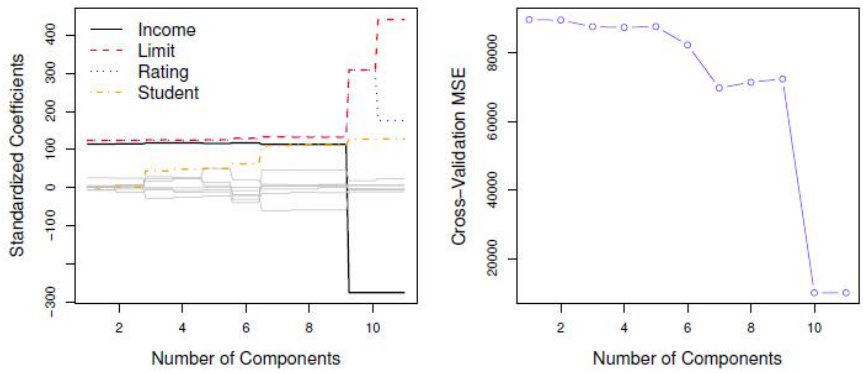
We can also plot the correlation of the first component and the predictor: They are highly correlated!



More components lead to more variance but less bias:



Now estimate the components coefficients:



We can get the coefficients based on different number of components. And the cross validation error would be smallest when contains the full predictors. Thus principal components may be not suitable for some datasets.

Partial least squares:

PCR identifies the linear combinations or directions. These directions are identified in an unsupervised way since no response would be helpful in deciding and constraining things! Thus no guarantee that the directions would be the best directions to use for prediction.

PLS identifies these new features in a supervised way. It makes use of the response Y in order to identify new features that not only approximate the old features well, but also that are related to the response. That is, find directions that help explain both the response and the predictors.

Conclusion: sparsity is very popular nowadays. Thus methods like Lasso are hot.

- Non-linear models

1. Polynomial Regression

Moving beyond linearity

The truth is never linear! But often linearity assumption is good enough

Methods: polynomial, step functions, splines, local regression and generalized addictive models all offer a lot of flexibility without losing interpretation ability.

Degree-d polynomial regression:

$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + ... + \beta_d x_i^d + \varepsilon_i$, linear in coefficients and not linear in X.

Details: create new variables, X1=X, X2=X^2... and treat as multiple linear regression. Not interested in the coefficients but the estimation values

Since f(x) is a linear function of all betas, we can get a simple expression for point-wise variance Var(f(x)) at any value x. Then compute the fit.

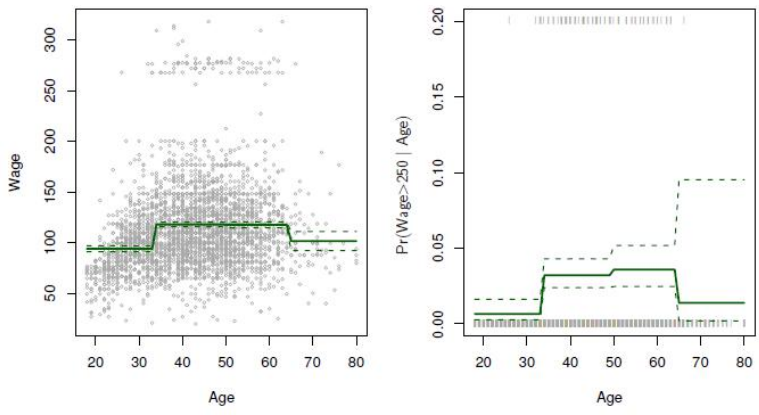
Also, we can fix the degree d or choose d by cross validation.

We can also put polynomial regression into logistic regression to fit. We can do separately on several variables or even stack the variables into a bigger matrix.

Step functions:

Cut the continuous variables into distinct regions. Then put these separated functions together to form a non-linear function, piece-wise function:

C1(X)=I(X<35), C2(X)=I(35<=X<50).....



That works well if there are some **natural cuts** in variable!!!

Implement: create a series of dummy variables representing each group or create interactions that are each to interpret, satisfying some conditions:

I(year<2005)*Age, I(year>=2005)*Age, this allows for different linear functions in each group!

In R command: I(year<2005) or cut(age, c(18,25,40,65,90))

Smoother alternatives: splines

2. Piece-wise Regression and splines

Use different polynomials in regions

$$y_i = \begin{cases} \beta_{01} + \beta_{11} x_i + \beta_{21} x_i^2 + \beta_{31} x_i^3 + \epsilon_i & \text{if } x_i < c; \\ \beta_{02} + \beta_{12} x_i + \beta_{22} x_i^2 + \beta_{32} x_i^3 + \epsilon_i & \text{if } x_i \geq c. \end{cases}$$

Splines have the maximum amount of continuity

We can also enforce the splines to be continuous at boundary

Linear Splines:

A linear spline with knots ξ_k from 1...K is piece-wise linear polynomial continuous at each knot:

$$y_i = \beta_0 + \beta_1 b_1(x_i) + \beta_2 b_2(x_i) + \cdots + \beta_{K+3} b_{K+3}(x_i) + \epsilon_i,$$

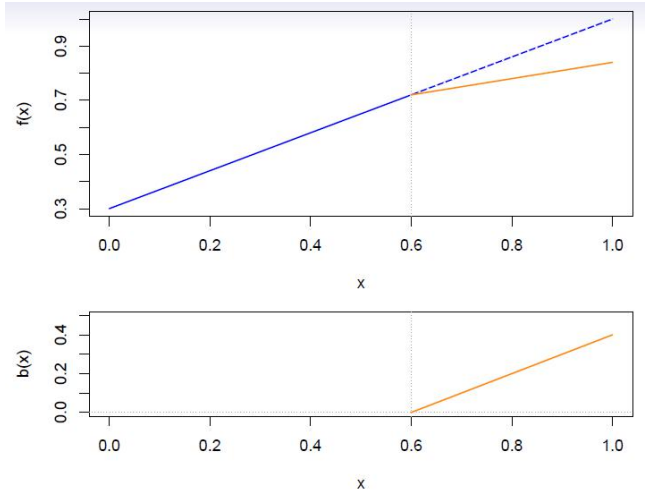
where the b_k are *basis functions*.

$$\begin{aligned} b_1(x_i) &= x_i \\ b_{k+1}(x_i) &= (x_i - \xi_k)_+, \quad k = 1, \ldots, K \end{aligned}$$

Here the $()_+$ means *positive part*; i.e.

$$(x_i - \xi_k)_+ = \begin{cases} x_i - \xi_k & \text{if } x_i > \xi_k \\ 0 & \text{otherwise} \end{cases}$$

Under the Linear splines, each part is a linear function but they connect well.



It has only one knot above and the slope has been changed after that knot.

Cubic Splines:

A cubic spline with knots ξ_k from 1...K is piece-wise cubic polynomial with continuous derivatives up to order 2 at each knot.

Model:

$$y_i = \beta_0 + \beta_1 b_1(x_i) + \beta_2 b_2(x_i) + \cdots + \beta_{K+3} b_{K+3}(x_i) + \epsilon_i,$$

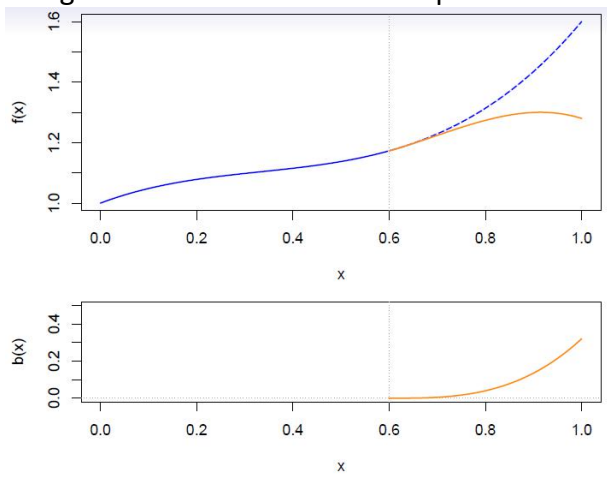
$$\begin{aligned} b_1(x_i) &= x_i \\ b_2(x_i) &= x_i^2 \\ b_3(x_i) &= x_i^3 \\ b_{k+3}(x_i) &= (x_i - \xi_k)_+^3, \quad k = 1, \ldots, K \end{aligned}$$

where

$$(x_i - \xi_k)_+^3 = \begin{cases} (x_i - \xi_k)^3 & \text{if } x_i > \xi_k \\ 0 & \text{otherwise} \end{cases}$$

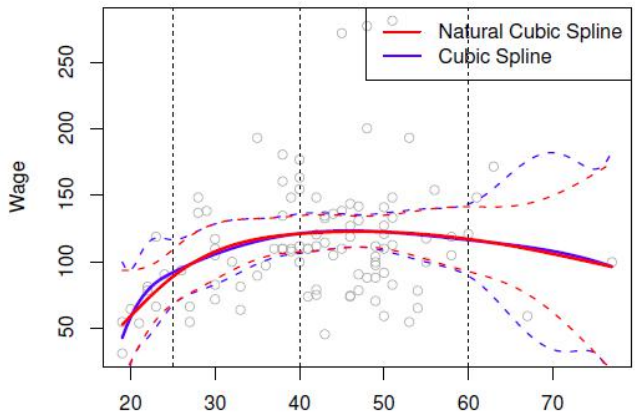
The model would change the whole line to be a little cubic after knots.

But it doesn't change the derivative at the knot point.



Natural Cubic Splines:

The cubic spline may go beyond the data finally. Thus natural cubic splines add boundary constraints. It adds 4 constraints here.



Splines can have high variance at the outer range of the predictors, when X takes either small or large values. Natural spline is a regression with additional boundary constraints. That the function is required to be linear at the boundary. That natural spline can make more stable estimates at boundary. And the

confidence interval can be narrower.

Knot placement:

One strategy is to decide K, the number of knots and place them at appropriate quantiles of observed X. We should place more knots in places where there may vary more rapidly and place fewer knots if it is stable.

A cubic spline with K knots has K+4 parameters or degree of freedom. And a natural spline with K knots has K degrees of freedom.

3. Smoothing splines

Mathematical:

Consider the criterion to fitting a smooth function g(x) to some data:

$$\underset{g\in S}{\text{minimize}} \sum_{i=1}^{n} (y_i - g(x_i))^2 + \lambda \int g''(t)^2 dt$$

The first term is the RSS to make g(x) match the data for each point. The second is to make the function smooth. It adds up all non-linearity cases in the function. It is a roughness penalty and controls how wiggly g(x) is. It can be defined by the tuning parameter and lambda is positive.

Smaller lambda means more wiggly the function is, finally interpolating when lambda is equal to zero. As lambda goes to infinity, the function g(x) becomes linear.

The solution of this is a natural cubic spline with a knot at every unique value of x, the roughness penalty still controls the roughness via lambda.

Some details for smoothing splines:

1. Smoothing splines avoid the knot-selection issue, leaving a single lambda to be chosen.
2. The vector of N fitted values can be written as $\hat{g}_\lambda = S_\lambda y$, where S_λ is a n*n matrix determined by the position of x and the particular value of lambda. This representation is the linear smoother, thus it can also be considered as polynomial regression.
3. We can calculate the effective degrees of freedom:

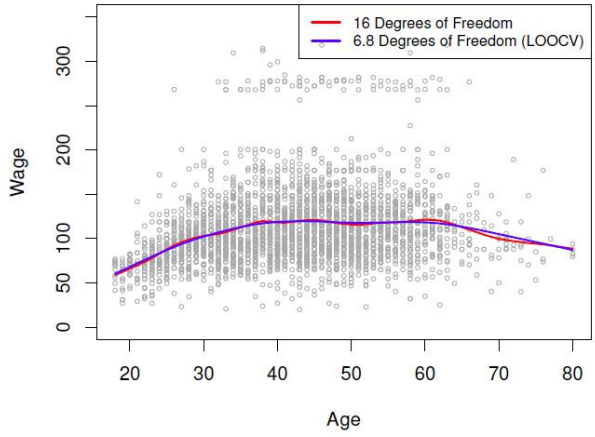
$$df_\lambda = \sum_{i=1}^{n} \{S_\lambda\}_{ii}$$

Thus we can specify degrees of freedom rather than lambda.

We can calculate the leave-one-out cross validation error is given by:

You just kick out the point xi, then use that estimated function g to to calculate the error of that point. We pick the value of lambda that minimize the error:

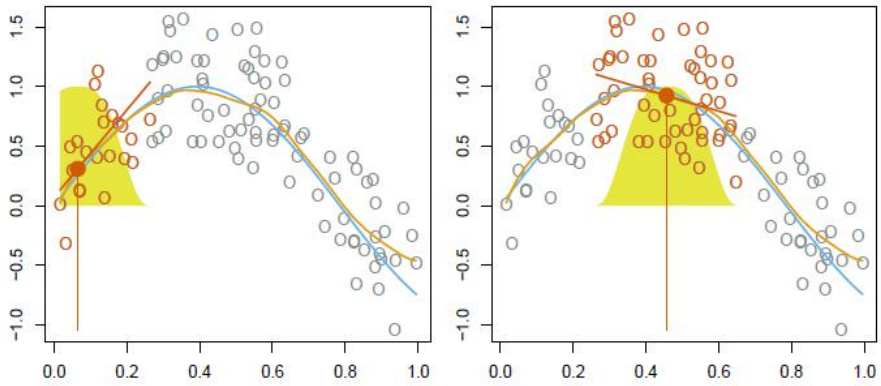
$$RSS_{cv}(\lambda) = \sum_{i=1}^{n} \left(y_i - \hat{g}_\lambda^{(-i)}(x_i)\right)^2 = \sum_{i=1}^{n} \left[\frac{y_i - \hat{g}_\lambda(x_i)}{1 - \{S_\lambda\}_{ii}}\right]^2$$



4. Local regression and generalized additive models

Local regression:

A different approach for fitting flexible non-linear functions, which involves computing the fit at a target point X0 using only nearby training observations.



Local regression illustrated on some simulated data, where the blue curve represents f(x) from which the data were generated, and the light orange curve corresponds to the local regression estimate f(x).

The orange colored points are local to the target point x0, represented by the orange vertical line. The yellow bell-shape superimposed on the plot indicates weights assigned to each point, decreasing to zero with distance from the target point.

The fit f(x0) at x0 is obtained by fitting a weighted linear regression (orange line segment), and using the fitted value at x0 (orange solid dot) as the estimate f(x0).

Thus it focuses locally on the data and it is a rich topic!

Some things to decide: define the weighting function K, whether to fit a linear, constant or quadratic regression in step 3 below, how many points to choose (span). The span plays a role like tuning parameter lambda in smoothing splines since it controls the flexibility of the non-linear fit. Smaller span means more local and wiggly, otherwise a global fit to the the data. We can use cross validation to decide the tuning parameter.

Algorithm 7.1 *Local Regression At $X = x_0$*

1. Gather the fraction $s = k/n$ of training points whose x_i are closest to x_0.
2. Assign a weight $K_{i0} = K(x_i, x_0)$ to each point in this neighborhood, so that the point furthest from x_0 has weight zero, and the closest has the highest weight. All but these k nearest neighbors get weight zero.
3. Fit a *weighted least squares regression* of the y_i on the x_i using the aforementioned weights, by finding $\hat{\beta}_0$ and $\hat{\beta}_1$ that minimize

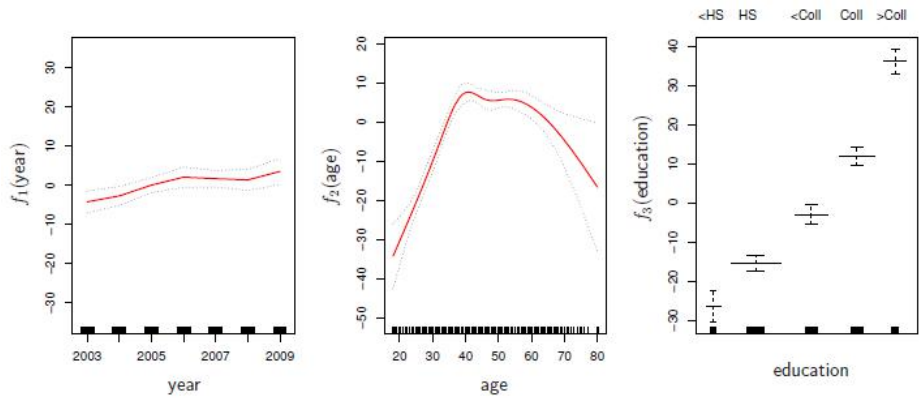
$$\sum_{i=1}^{n} K_{i0}(y_i - \beta_0 - \beta_1 x_i)^2. \tag{7.14}$$

4. The fitted value at x_0 is given by $\hat{f}(x_0) = \hat{\beta}_0 + \hat{\beta}_1 x_0$.

Generalized Additive models:

It allows for flexible non-linearity in several variables but retains the additive structure of linear models. Interpret model:

$$y_i = \beta_0 + f_1(x_{i1}) + f_2(x_{i2}) + \ldots + f_p(x_{ip}) + \varepsilon_i$$



We can plot functions for different variables. The last one is piece-wise. Thus we can judge the contribution of different predictors.

Details: thus we can fit a model based on natural splines like:

Lm(wage~nature spline(year)+nature spline(age)+education) And use plot.GAM command to visualize the estimation in R.

We can also use some linear or non-linear and use ANOVA command in R to compare the models.

We can also contain low-order interactions in the GAM model.

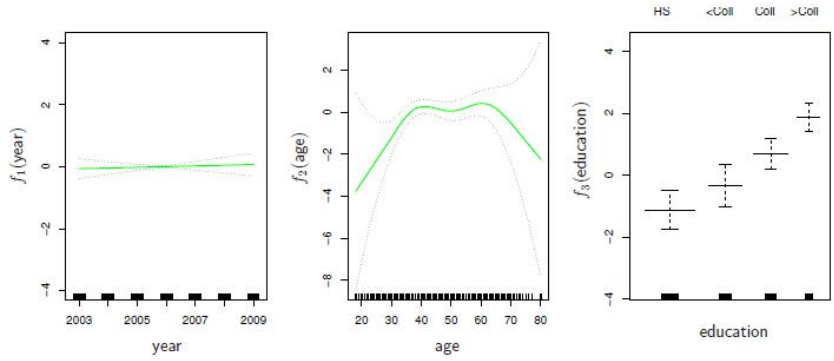
ns(age, df=5):ns(year, df=5) bi-variate smoother

GAM can also be used in classification:

together with the logistic regression model

$$\log\left(\frac{p(x)}{1-p(x)}\right) = \beta_0 + f_1(x_1) + f_2(x_2) + \ldots + f_p(x_p)$$

It can also visualize the function for each predictor:



- Decision Trees

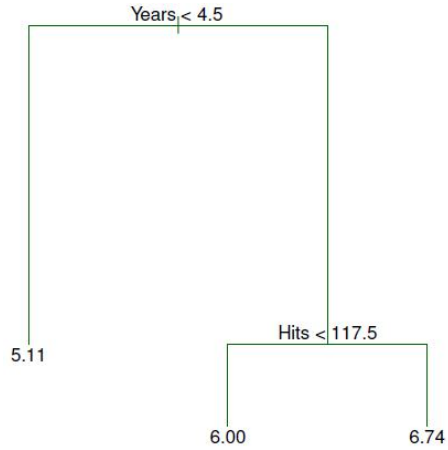
1. Decision trees

Tree based methods for regression and classification

Stratifying or segmenting the predictor space into a number of simple regions. The set of splitting rules used to segment the space can be summarized in a tree, the decision tree methods.

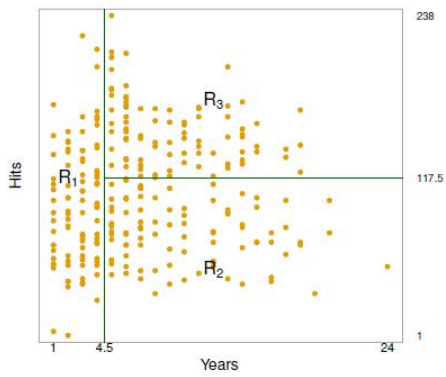
For accuracy, improvements contain bagging, random forest and boosting. They would grow multiple trees.

Stratify -- divide into levels



All data first then give the first split at year<4.5.

Two dimensional partition of the tree above:



Some terminology: tree, the region R1, R2 and R3 are terminal nodes. Decision trees are typically drawn upside down. Leaves are at the bottom of the tree.

This kind of explanation is too simplified but it is very intuitive and easy to display. And it can handle the missing data problem a little bit! No equations here!

Regression model with the tree:

We divide the predictor space into J distinct and non-overlapping regions. For every observation that falls into the same region, we would make the same prediction, which is simply the mean of the response value for the training observations in that region!

More details for it: Since the region could have any shape. We choose to divide the predictor space into high-dimensional rectangles or boxes.

And the goal is to find boxes R1, R2,...RJ that minimize the RSS, given by

$\sum_{j=1}^{J}\sum_{i\in R_j}\left(y_i-\hat{y}_{R_j}\right)^2$, that is the sum of squared error based on each region.

Problem with this principle: it is computationally infeasible to test all the combinations of the regions.

Thus we use a top-down, greedy approach that is known as recursive binary splitting method! It begins at the top of the tree and then successively splits the predictor space by diving into two new branches. It is greedy since at each step of the process, the best split is made at that particular step, based on the greatest reduction in RSS in each region.

Finally, we would have a better tree after some steps!! But we needs to specify the stopping rule, like no more than five observations in each region.

2. Pruning trees

Predict the response based on the training points in the region
It may over-fit the data, leading to poor test set performance
Smaller tree would lead to lower variance and better interpretation

Pruning a tree:
grow a large tree To and then prune it back in order to obtain a sub-tree.
Cost complexity pruning: weakest link pruning
Want the tree with small variance and also control the size of the tree:
Alpha as tuning parameter can be decided by cross validation

$\sum_{m=1}^{|T|}\sum_{i:x_i\in R_m}\left(y_i-\hat{y}_{R_m}\right)^2+a|T|$, here Rm is the rectangle corresponding to the terminal node. This kind of objective function can also be written as

$C(T)=\sum_m n_m Q_m(T)+a|T|$ for classification problem -- reduce the impurity

Tree algorithm:

1. Build a large tree based on some stopping rules.
2. Apply cost complexity pruning to obtain a sequence of best sub-trees as a function of alpha.
3. Use K-fold cross validation to choose alpha:

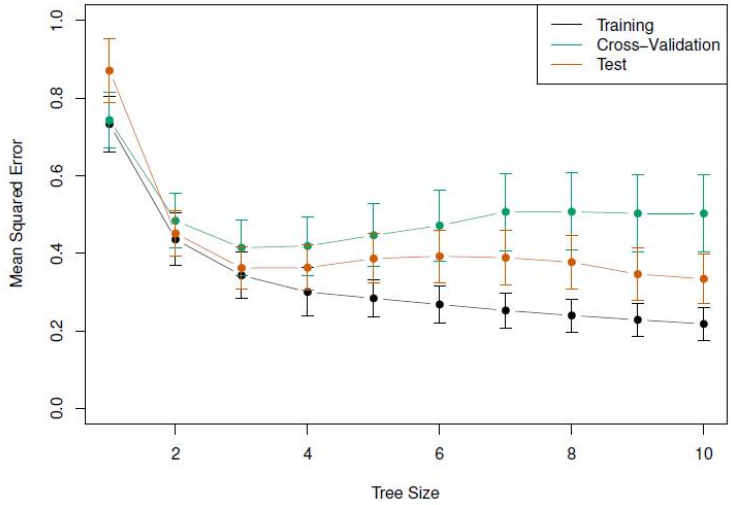
For each k=1,...K, repeat steps 1 and 2 on the K-1/K fraction of the training data, excluding the k-th fold.

Evaluate the mean squared prediction error on the data in the left-out k-th fold as a function of alpha.

Average the results and pick alpha to minimize the average error.

4. Return step 2 that corresponds to the chosen value of alpha

We need to check the test error to find the most suitable tree size:
The following plot indicates 3 or 4 terminal nodes would be good.



3. Classification trees

Predict a qualitative response not a quantitative one

Predict the each observation belongs to the most commonly occurring class

Use the classification error rate:

Classification noise:

the fraction of the training observations in that region that do not belong to the most common class:

$E = 1 - \max_k(\hat{p}_{mk})$, here $\hat{p}_{mk}$ represents the proportion of training observations in the m-th region that are from the k-th class

But this is not a very smooth way.

Good alternative way to measure node purity: Gini Index!!!

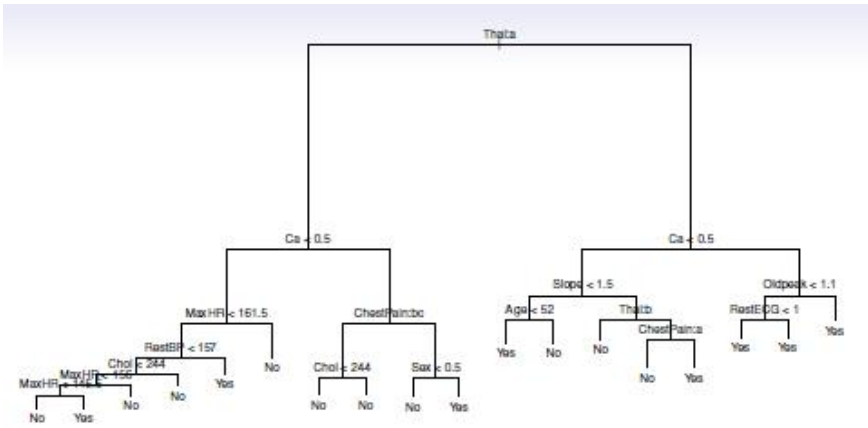
Gini index: $G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$, a measure of total variance across the K classes. IF the region is pure, then G would be zero, else if evenly distributed, the G should be maximized. Thus we want to a smaller G.

Cross-entropy:

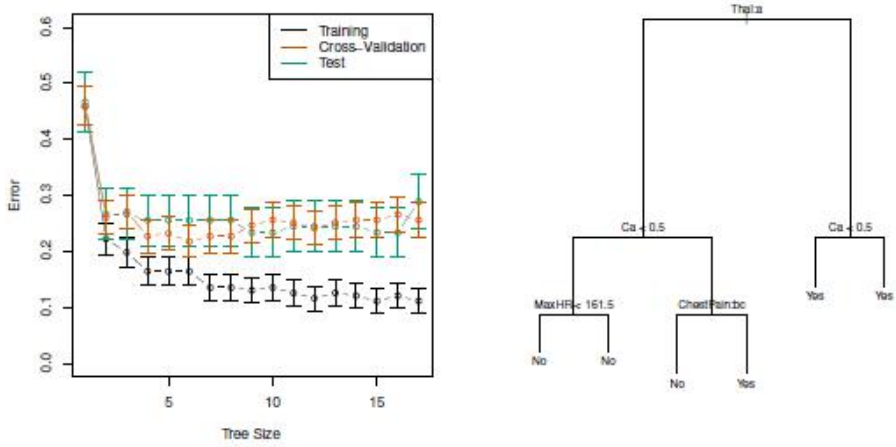
It turns out that the Gini index and the cross-entropy are similar:

$$D = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}$$

The following example shows:



After pruning:



Tree versus linear models:

The accuracy should depend on the shape of boundary. Linear models should work better on linear boundary. Tree works on partition.

Some judgement on trees:

1. Easy to explain and display graphically
2. Can handle qualitative predictors without dummy variables
3. Problem would be accuracy improvement

4. Bootstrap aggregation/bagging and random forests

Bootstrap, aggregation and bagging are for reducing the variance.

First, averaging a set of observations reduces the variance.

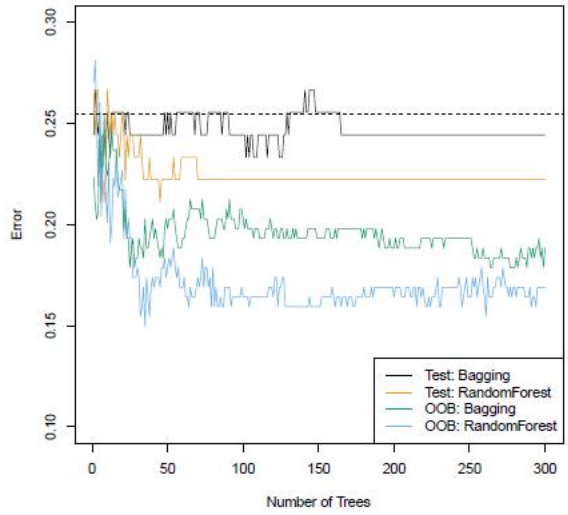
If we have multiple training sets, we can reduce the variance

Bagging:

We can take repeated samples from the single training data set. Generate B different bootstrapped training data sets. Then train our method on the b-th bootstrapped training set to get the prediction of a point x and average.

$$\ddot{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^{*b}(x)$$

For classification problem, we can take the majority vote as well



Out of Bag Error Estimation:

Estimate the test error of a bagged model:

By taking repeated samples, we can prove that about 2/3 samples would be used to use, remaining 1/3 samples are called out-of-bag (OOB) observations. Thus we calculate the average error of this part as OOB error.

Else we can even use LOO cross-validation error for bagging if B is large.

Random Forests:

This random selection process is more likely to be an unsupervised step here

Improve over bagged trees by de-correlating the trees. This reduces the variance when we average the trees.

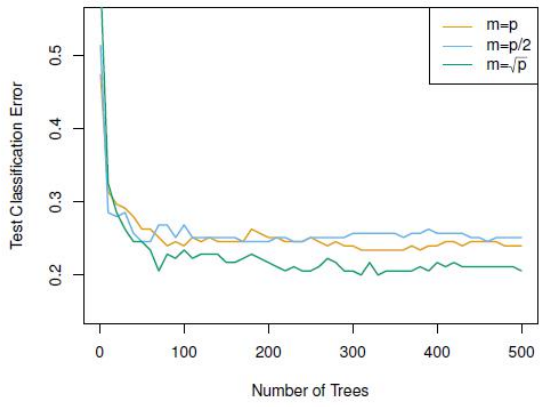
It forces trees to use different predictors to split the dataset different times

When we build decision trees, 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 one of those m predictors.

Approximately, $m \approx \sqrt{p}$. Thus this generates almost different trees.

Number of trees plot:

We can consider all possible trees.



Some details to discuss for this random forest:

At each split, the algorithm is not allowed to consider a majority of the available predictors. Since if there are very strong predictors, all trees would be quite similar to each other.

Within each trees, both predictors selection and training data selection would be done. Thus if m=p above, that would be exactly bagging process.

5. Boosting

Boosting is a general approach for regression and classification.

Boosting: each tree is grown using information from previously grown trees

Each tree is built based on the previous residual of the trees collected.

Boosting algorithm:

1. Set $\hat{f}(x) = 0$ and $r_i = y_i$ for all i in the training set.
2. For $b = 1, 2, \ldots, B$, repeat:
   - 2.1 Fit a tree $\hat{f}^b$ with d splits ($d+1$ terminal nodes) to the training data (X, r).
   - 2.2 Update $\hat{f}$ by adding in a shrunken version of the new tree:

$$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x).$$

   - 2.3 Update the residuals,

$$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i).$$

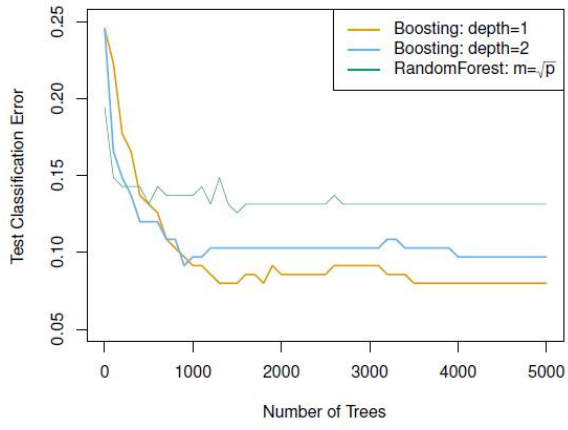
3. Output the boosted model,

$$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x).$$

Details: learners are learned sequentially with early learners fitting simple models to the data and analyze data for errors. We fit consecutive trees/random sample and at every step, the goal is to solve for net error from the prior tree.

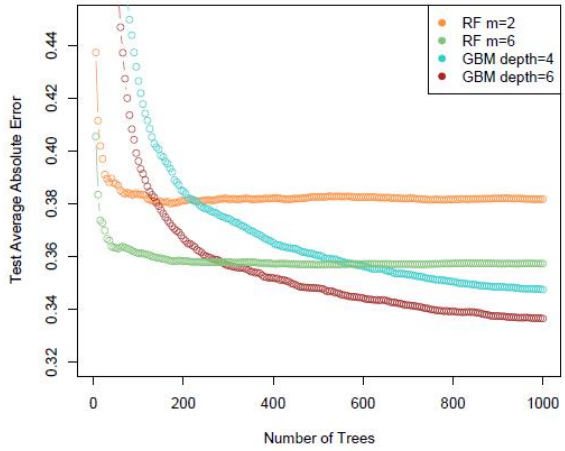
If an input is misclassified by a hypothesis, its weight is increased so that next hypothesis is more likely to classify it correctly. By combining the whole set at the end converts weak learners into better performing model.

The boosting approach learns slowly. We fit a decision tree to the residuals from the model and add this new decision tree into the fitted function in order to update the residuals.



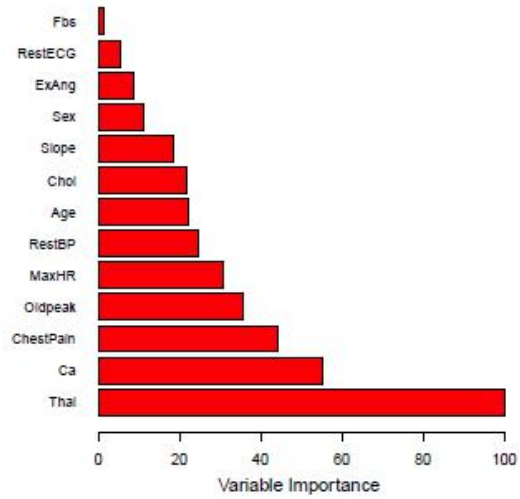
Some parameters to decide for the boosting process:

1. Number of trees: B. if B is too large, over-fitting. Or use cross validation to find B
2. Shrinkage parameter lambda: small positive number, control the rate at which boosting learns. Generally around 0.01 or 0.001. If lambda is too small, we would need large B to achieve good performance.
3. Number of splits d in each tree: if d=1, then each tree is a stump and each contains only one predictor. If d>1, then d controls the interaction depth.



Variable importance measure:

For bagged/RF regression trees, the total amount of the RSS is decreased over a given predictor averaged over all B trees. ***We add up the total amount that the Gini index is decreased by splits over a given predictor over all B trees!!***



All these ensemble methods are good at prediction but challenging in interpreting things than single tree.

Some more details from the "Elements of Statistical Learning" for Boosting can be check in the book.

Tree boosting: combine many weak learners (trees) into a strong classifier

Each tree is created iterative

The tree's output (h(x)) is given a weight w relative to its accuracy

The ensemble output is the weight sum: $\hat{y}(x) = \sum_t w_t h_t(x)$

After each iteration, each data sample is given a weight based on

its misclassification. The more often a data sample is misclassified, the more importance it becomes.

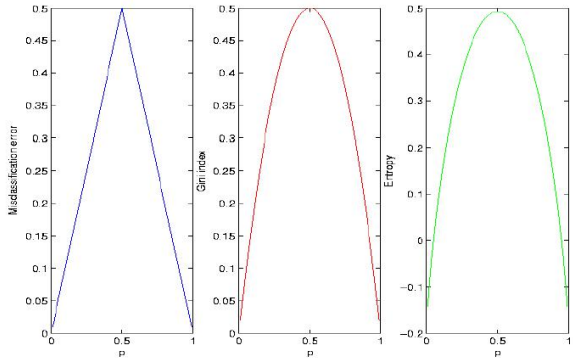
The goal is to minimize an objective function: $O(x)=\sum_i l(\hat{y}_i, y_i)+\sum_t \Omega(f_t)$, here the first term is the distance between the truth and the prediction of that sample. And the second one is regularization function to penalize the complexity.

Something about Splitting:

Most implementations consider only binary splits: $p_k(m)=n_m^{-1}\sum_{x_i\in R_m} I(y_i=k)$, which is the proportion of observations from class k in region Rm.

Then observations in node m are classified to the majority class

$$k(m)=\arg\max_k p_k(m)$$



How to operate:

When considering which parent node P with n_p observations to split into two children, nodes L and R with n_L, n_R observations respectively, we aim to maximize:

$$M(split)=Q(P)-\left(\frac{n_L}{n_P}Q(L)+\frac{n_R}{n_P}Q(R)\right),$$

- Misclassification error

$$Q_m = n_m^{-1}\sum_{i\in R_m}\mathbf{1}(y_i\neq k(m)) = 1-\max_k p_k(m)$$

- Gini index

$$Q_m=\sum_{k=1}^{K} p_k(m)(1-p_k(m)).$$

(the variance of a Bernoulli r.v. is $p(1-p)$)

- Entropy or deviance (another measure of variance)

$$Q_m=-\sum_{k=1}^{K} p_k(m)\log p_k(m).$$

here Q is the Misclassification error/Gini Index/Information entropy

That is the impurity decrease from parent to children (Greedy, no global optimize)

This would help in giving a more reasonable split to decrease impurity!

		Class1	Class2	p_1	p_2	Gini	Error
	Parent	20	20	0.5	0.5	0.5	0.5
Split 1	Left	10	20	1/3	2/3	4/9	1/3
	Right	10	0	1	0	0	0
Split 2	Left	15	5	0.75	0.25	0.375	0.25
	Right	5	15	0.25	0.75	0.375	0.25

Gini: $M(\text{split 1}) = 0.167$, $M(\text{split 2}) = 0.125$.
Classification error: $M(\text{split 1}) = M(\text{split 2}) = 0.25$

Gini tends to be more sensitive to node purity

The split1 would be better than split2 according to the Gini Index
The problem of trees is, small changes in the data lead to completely different trees. And it is difficult in modeling additive structures.

- Support Vector machines
  1. Maximal Margin Classifier

     Approach the two-class classification problem:

     Try and find a plane that separates the classes in feature space or soften separation/enrich and enlarge feature space to separate.

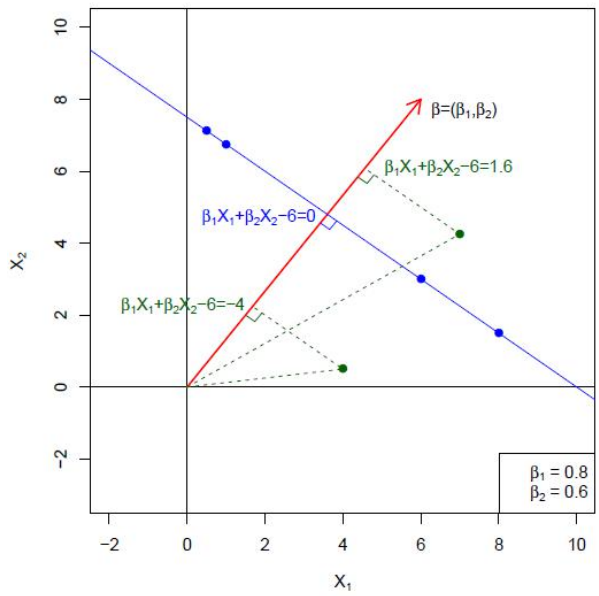
     Hyper-plane definition:

     Hyper-plane in p dimensions is a flat subspace of dimension p-1

     Form: $\beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p = 0$

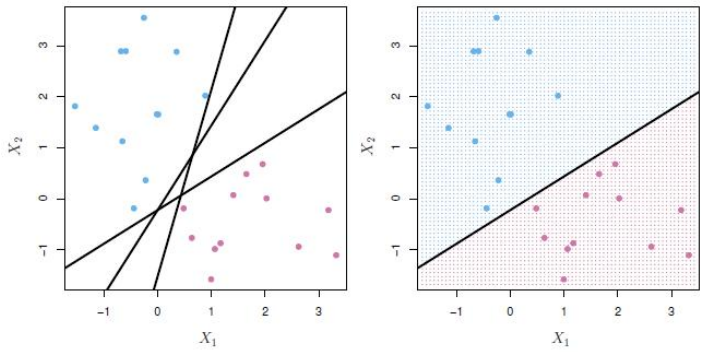
     In p=2, that is a line. The vector $\beta = (\beta_1, \beta_2, \ldots \beta_p)$ is called the normal vector that points in a direction orthogonal to the hyper-plane surface.

     Hyper-plane in two dimensions:



The blue line is the hyper-plane. The red one is the normal vector of the blue one. For any given point, we can project it on the normal. We can get the distance from point to normal vector. All the points on the hyper-plane project onto the normal vector would get zero.

We can get the distance from points to hyper-plane and the sign matters.



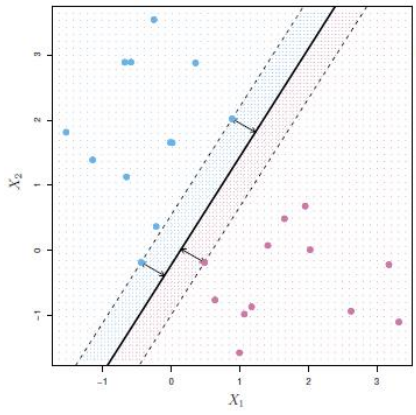
Each line in the left can separate the two kinds of points.

$f(X) = \beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p$, if f(x)>0, one side, else on the other side.

Thus this defines a separating hyper-plane.

Then how to choose a best hyper-plane:

Maximal Margin Classifier: find the one that makes the biggest gap or Margin between the two classes.



Constrained optimization problem

$$\underset{\beta_0,\beta_1,\ldots,\beta_p}{\text{maximize}}\, M$$

$$\text{subject to } \sum_{j=1}^{p} \beta_j^2 = 1,$$

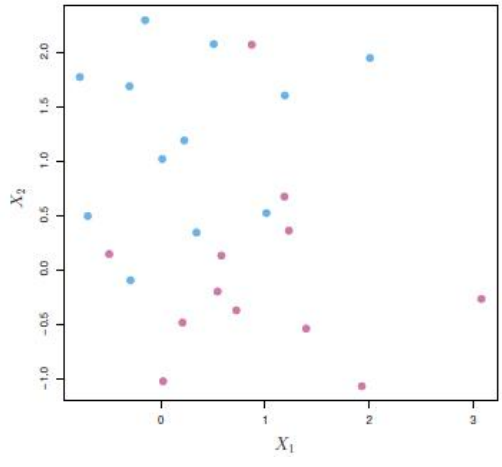
$$y_i(\beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip}) \geq M$$

$$\text{for all } i = 1, \ldots, N.$$

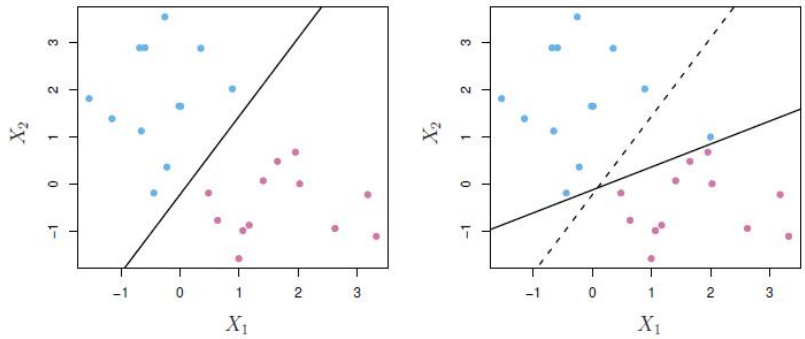
If you make a big gap on training data, it is likely that there will also be a big gap on the test data. This may be over-fitting but always work well. The constraints are measured in maximizing the distance. It is a convex quadratic problem and can be solved easily by software.

2. Support vector machine

Non-separable data: sometimes can not separate unless N<p



Also, there maybe noisy data! Sometimes a single point may influence the separating hyper-plane dramatically, which is really noisy! Not robust -- outliers



The support vector classifier maximize a soft margin!

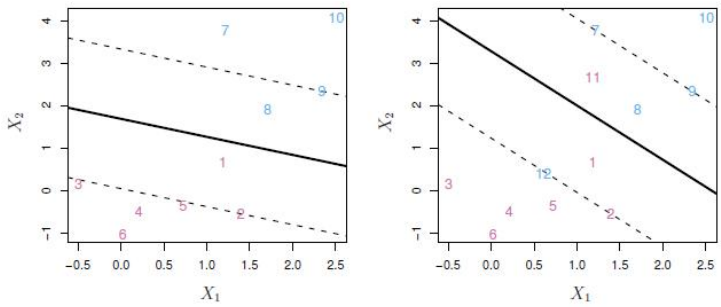
We can make the margin a little bit wider or smaller -- regularization

The margin is decided not only on the closest point

Allows some slacks or discount and add some budget for all the discount

And C is a regularization parameter here. It would change the margin result.

C gets bigger, the margin would be more stable.



$$\underset{\beta_0,\beta_1,\ldots,\beta_p,\epsilon_1,\ldots,\epsilon_n}{\text{maximize}} \; M \quad \text{subject to} \quad \sum_{j=1}^{p} \beta_j^2 = 1,$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) \geq M(1 - \epsilon_i),$$

$$\epsilon_i \geq 0, \; \sum_{i=1}^{n} \epsilon_i \leq C,$$

Some understanding about C tuning parameter:

C decides the number and severity of the violations to the margin and hyper-plane that we would tolerate. As C increases, we have more tolerance. C as a tuning parameter can be decide by cross-validation.

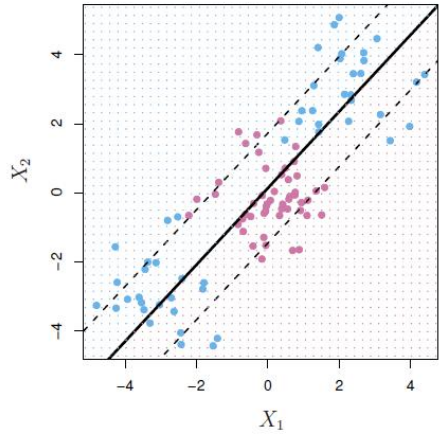
C controls the bias-variance trade-off of the model. When C is small, margin is narrowly violated and highly fits the data thus low bias and high variance.

Also, an observation that lies strictly on the correct side of the margin would not affect the hyper-plane.

Observations that lies on the correct/wrong side of the margin are support vectors. When C is large, many observations would influence the hyper-plane.

The decision relies on a subset of observations thus it is robust to the behavior of observations that are far away from the hyper-plane. This is quite different from other regression/classification methods

Sometimes linear boundary would fail!



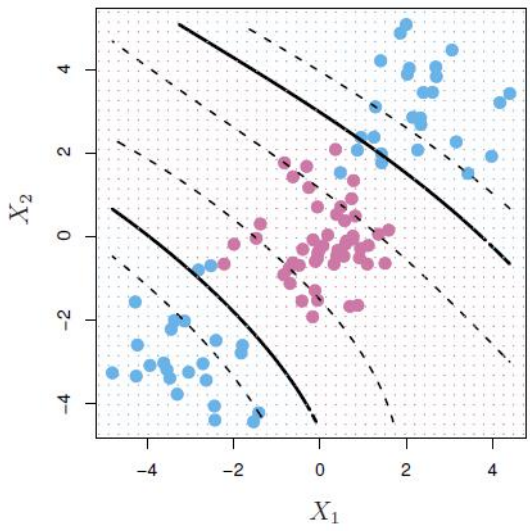
3. Kernels and support vector machines

Feature expansion:

Enlarge the space of features by including transformations: X1^2, X1^3, X1X2, X1X2^2... thus go from a p-dimensional space to a M>p dimensional one. Thus fit a support-vector classifier in the enlarged space. And it is a non-linear boundary in the original space.

For example, the boundary may be:

$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1^2 + \beta_4 X_2^2 + \beta_5 X_1 X_2 = 0$, can even be cubic



There may be pairs of boundaries in higher dimensional space.

Non-linearity and kernels:

There is a more elegant and controlled way to introduce non-linearity in support-vector classifiers -- through the use of kernels/inner products

Inner products and support vectors:

Inner products between vectors: $\langle x_i, x_{i'} \rangle = \sum_{j=1}^{P} x_{ij} x_{i'j}$

Thus the linear support vector classifier can be represented as:

$$f(x) = \beta_0 + \sum_{i=1}^{n} a_i \langle x, x_i \rangle$$

Thus to estimate the parameters in the support vector classifier, we need to calculate $\binom{n}{2}$ pairs of inner products between all pairs of training points.

It can be estimated that most of the alpha parameters can be zero! Then: $f(x) = \beta_0 + \sum_{i \in S} \hat{a}_i \langle x, x_i \rangle$, these alpha are positive and S is the support set.

Kernels and support vector machine:

We can expand the space by d-dimension polynomials with kernel function:

It computes the inner products needed for d dimensional polynomials:

Kernel function:

$$K(x_i, x_{i'}) = \left(1 + \sum_{j=1}^{p} x_{ij} x_{i'j}\right)^d$$

After the kernel comes to the support vector machine:

$$f(x) = \beta_0 + \sum_{i \in S} \hat{a}_i K \langle x, x_i \rangle$$

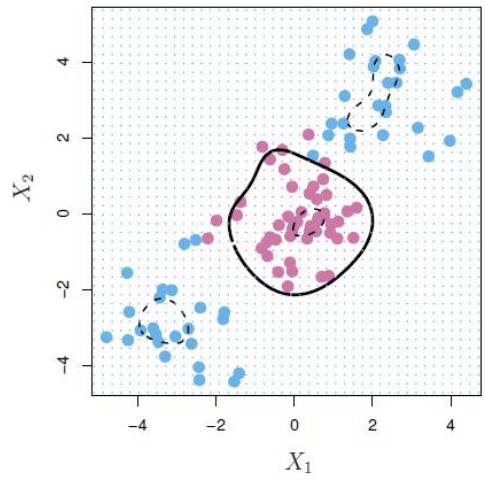
One example for the kernel:

Radial Kernel: often use in non-linear SVM

$$K \langle x_i, x_{i'} \rangle = \exp\left(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2\right)$$

Gamma is the tuning parameter, if gamma is larger, then small standard deviation, we would get much wiggly boundaries.

With large gamma, the boundary is smooth.



But how to explain that this kind of infinity dimensional kernels would not over-fit the data? It controls the variance by squashing down most dimensions severely. Most wiggly dimensions would be finally set to zero.

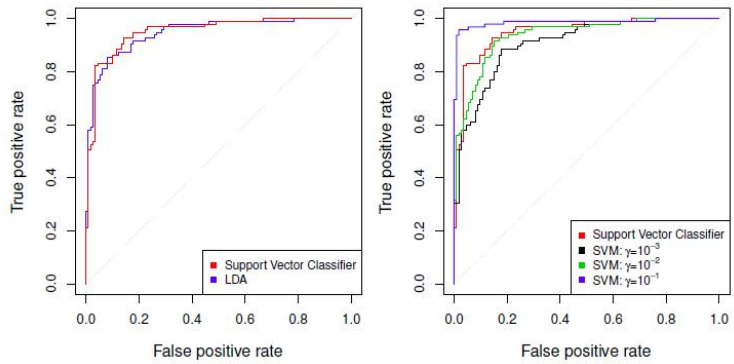
4. Comparison with logistic regression

Here the ROC curve is obtained by changing the threshold 0 to some t.

And record false positive and true positive rate as t varies:

AUC: area under the curve, sum of the area

Here shows the training error and smaller gamma indicates more fit to the training data, thus may be over-fitting -- should check the test error in real.



All above about SVM are two classes -- then for more than two classes:

Some alternatives:

OVO:

one versus one: fit all pairs of binary classifiers then classify x to the class that wins the most pairwise competitions.

Anyway, if the multiple class problem has n classes, the OVO ensemble will be composed by n(n-1)/2. The label assigning stage is then performed by majority voting. Say you have three classes, A, B and C. The OVO ensemble will be composed of 3 (= 3 * (3 - 1) / 2) binary classifiers. The first will tell A from B, the second A from C, and the third B from C. Now, if x is to be classified, x is presented to each binary classifier of the ensemble to create a vector of individual classifications, e.g. (A, B, B). The final step to assign a label to x is the majority voting. In this example, x would belong to class B.

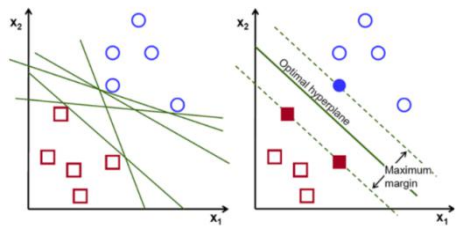
OVA:

one versus all: fit K different 2-class SVM classifiers, each class versus the rest. Then classify the x to the larger f(x)

OVA One versus All. Fit K different 2-class SVM classifiers $\hat{f}_k(x)$, $k = 1, \ldots, K$; each class versus the rest. Classify x^* to the class for which $\hat{f}_k(x^*)$ is largest.

Thus each classifier can provide a margin for the point needs to classify. And we need to find the class with largest margin for that point.

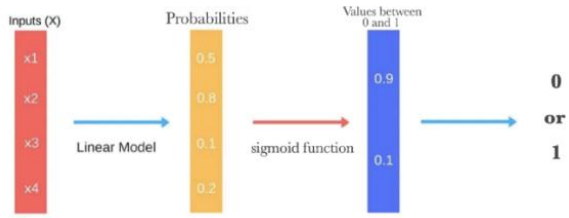
Now compare support vector machine and logistic regression:



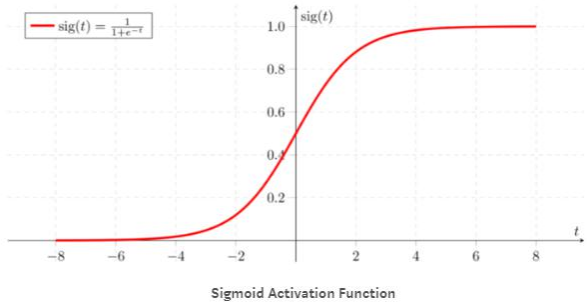
In logistic regression, we take the output of the linear function and squash the value within the range of 0 and 1 using the logistic function/SIGMOID function. It map it into a value between 0 and 1 but never exactly at those limits.

Logistic functions would give you discrete outcome which is the prediction but linear regression gives a continuous outcome!

Steps for logistic regression:



SIGMOID function plot:



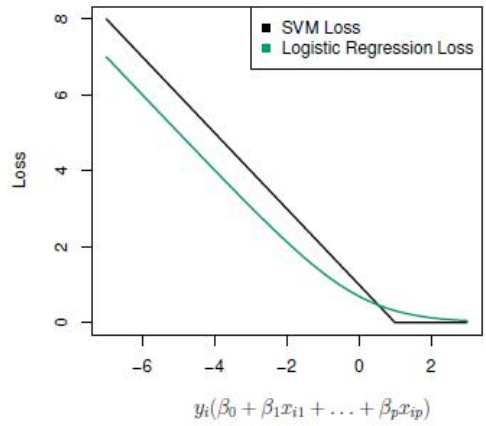
Sigmoid Activation Function

We want to maximize the likelihood that a random data point get classified correctly (maximum likelihood estimation).

Loss function of linear SVM and regularized LR:

$$\min_w \lambda\|w\|^2 + \sum_i \max\{0, 1 - y_i w^T x_i\}$$
$$\min_w \lambda\|w\|^2 + \sum_i \log\left(1 + \exp\left(1 - y_i w^T x_i\right)\right)$$

The only difference in the loss function -- SVM minimizes hinge loss while logistic regression minimizes logistic loss.

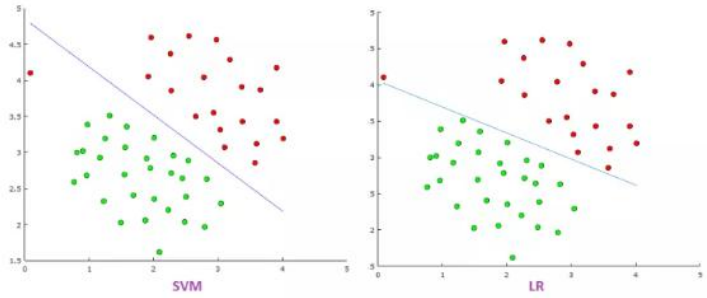


There are some differences to note:

1. Logistic loss diverges faster than hinge loss, thus more sensitive to outliers. Logistic loss does not go to zero and may lead to minor

degradation in accuracy so SVM may be marginally better than logistic.
Sometimes LR is too sensitive:

We may not want classifier to be so sensitive.



Detailed explanation:
https://towardsdatascience.com/support-vector-machine-vs-logistic-regression-94cc2975433f

When to use SVM or logistic regression:

1. If classes are nearly separable then SVM. Better than LR or LDA
2. If not, then LR and SVM similar
3. If estimate probability, then LR
4. For nonlinear boundaries, can use kernels with LR and LDA but computation expensive.

Addition: n = number of features, m = number of training examples

1. If n is large relative to m:
   use logistic regression or SVM with no kernel (linear kernel)
2. If n is small and m is intermediate:
   Use SVM with Gaussian Kernel
3. If n is small and m is large
   Create/add more features then use logistic regression or SVM without kernel (linear kernel)

- Principal Components and clustering

1. Principal components analysis

Unsupervised and supervised learning: label supervise learning
Unsupervised learning no labels

Now working on unsupervised learning: only the features X1, X2... and we are not interested in prediction because we have no associated response Y.

But we want to find some interesting features of measurement:
Methods: Principal components analysis (data pre-processing); clustering
Advantage:

1. Easy to obtain unlabeled data than labeled data

Principal Components analysis:

It produces a low-dimensional representation of a dataset. It finds a sequence of linear combinations of the variables that have maximal

variance and are mutually uncorrelated.

The first component would have the highest variance across the data that is a linear combination of the features...

Details: The first principal component of a set of features is the Normalized linear combination of the features:

$Z_1 = \phi_{11} X_1 + \phi_{21} X_2 + ... \phi_{p1} X_p$ that has the largest variance.

Normalized means that $\sum_{j=1}^{p} \phi_{j1}^2 = 1$

We refer to the elements $\phi_{11}, ... \phi_{p1}$ are the loading of the first principal component. Notation: $\phi_1 = (\phi_{11}, ... \phi_{p1})^T$, by constraining the sum of squares, we can constrain the arbitrarily large variance.

The PCA plot has been shown before.

Computation details:

Find the linear combination with largest variance first:

- Suppose we have a $n \times p$ data set $\mathbf{X}$. Since we are only interested in variance, we assume that each of the variables in $\mathbf{X}$ has been centered to have mean zero (that is, the column means of $\mathbf{X}$ are zero).
- We then look for the linear combination of the sample feature values of the form

$$z_{i1} = \phi_{11} x_{i1} + \phi_{21} x_{i2} + \ldots + \phi_{p1} x_{ip} \qquad (1)$$

  for $i = 1, \ldots, n$ that has largest sample variance, subject to the constraint that $\sum_{j=1}^{p} \phi_{j1}^2 = 1$.
- Since each of the x_{ij} has mean zero, then so does z_{i1} (for any values of ϕ_{j1}). Hence the sample variance of the z_{i1} can be written as $\frac{1}{n} \sum_{i=1}^{n} z_{i1}^2$.

Give the optimization problem form and solve by linear algebra:

PCA was performed after standardizing each variable to have mean zero and standard deviation one.

It is necessary to normalize data before performing PCA: Scaling

The PCA calculates a new projection of your data set. And the new axis are based on the standard deviation of your variables. So a variable with a high standard deviation will have a higher weight for the calculation of axis than a variable with a low standard deviation. If you normalize your data, all variables have the same standard deviation, thus all variables have the same weight and your PCA calculates relevant axis.

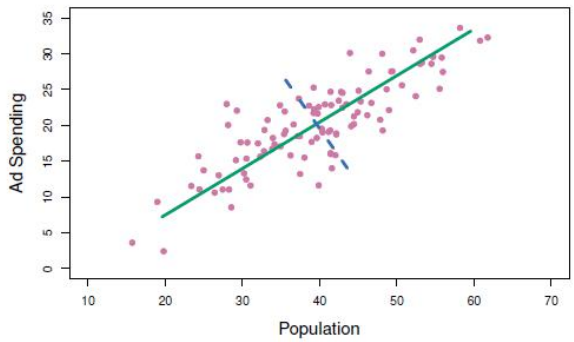
And get the Z1 vector as a linear combination of features.

- Plugging in (1) the first principal component loading vector solves the optimization problem

$$\underset{\phi_{11},\ldots,\phi_{p1}}{\text{maximize}} \frac{1}{n}\sum_{i=1}^{n}\left(\sum_{j=1}^{p}\phi_{j1}x_{ij}\right)^2 \text{ subject to } \sum_{j=1}^{p}\phi_{j1}^2 = 1.$$

- This problem can be solved via a singular-value decomposition of the matrix $\mathbf{X}$, a standard technique in linear algebra.
- We refer to Z_1 as the first principal component, with realized values $z_{11}, \ldots, z_{n1}$

The loading vector $\phi_1 = (\phi_{11},\ldots\phi_{p1})^T$ defines a direction in feature space along which the data vary the most. If we project the n data points onto this direction, the projected values are the principal component scores $z_{11},\ldots z_{n1}$ themselves.



Get the approximate (1, 1) direction. The projection reflects how far those points are from the direction.

2. Proportion of variance explained

Further principal components:

The second principal component is the linear combination of X1,...XP that has maximal variance among all linear combinations that are Uncorrelated with Z1.

Form:

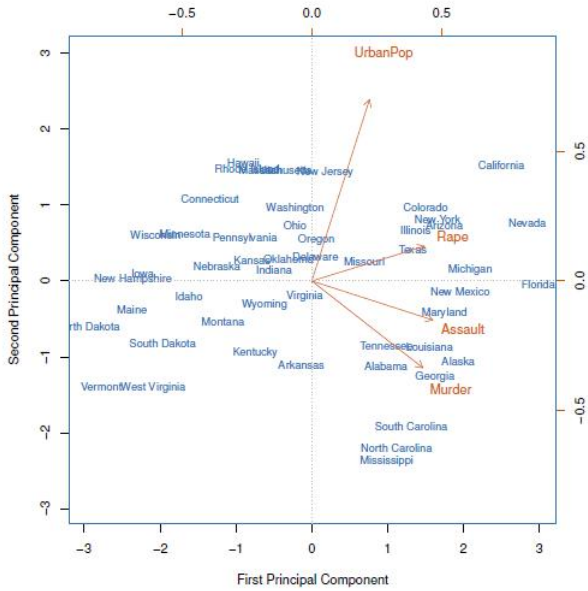
$$z_{i2} = \phi_{12}x_{i1} + \phi_{22}x_{i2} + \ldots + \phi_{p2}x_{ip},$$

where ϕ_2 is the second principal component loading vector, with elements $\phi_{12}, \phi_{22}, \ldots, \phi_{p2}$.

It turns out that constraining Z2 to be uncorrelated with Z1 is Equivalent to constrain the direction to be orthogonal/perpendicular to the first direction.

The principal component directions are the ordered sequence of right singular vectors of the data matrix X (n*p).

And the variances of the components are 1/n times the squares of the singular values. There are at most min(n-1, p) principal components.



Above displays the features and loading in a single plot. (Bi-plot)

All the points have been centered ahead of time. For each state point, there would be principal scores for them in this two principal components.

The first principal component got large positive loading on Rape, Assault and Murder. Thus a lot of states on the right are high crime regions.

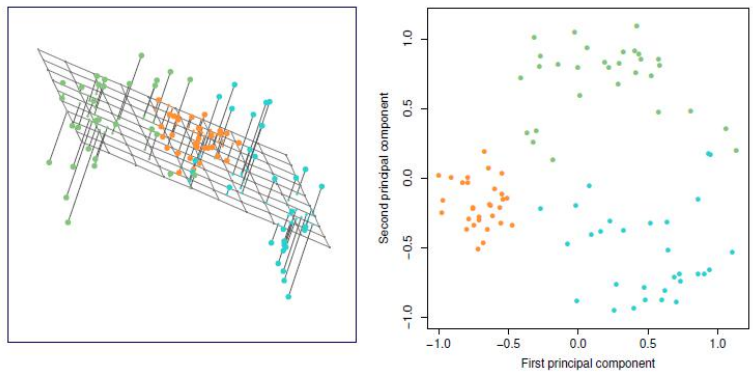
For the red line: it shows how strongly each characteristic influences a Principal component. ***The projection values on each each PC show how much weight they have on that PC.***

Another thing for loading plots: the angles between vectors tell us how characteristics correlate with one another. When two vectors are close, forming a small angle, they are represented positively correlated. If they meet each other at 90 degree, not correlated. If close to 180 degree, Then they are negatively correlated.

Loading show above:

	PC1	PC2
Murder	0.5358995	-0.4181809
Assault	0.5831836	-0.1879856
UrbanPop	0.2781909	0.8728062
Rape	0.5434321	0.1673186

Another interpretation:



The first principal component loading vector has a very special property:

it is the line in p-dimensional space that is closest to the n observations (using average squared Euclidean distance as a measure of closeness).

The appeal of this interpretation is clear: we seek a single dimension of the data that lies as close as possible to all of the data points, since such a line will likely provide a good summary of the data.

The notion of principal components as the dimensions that are closest to the n observations extends beyond just the first principal component. For instance, the first two principal components of a data set span the plane that is closest to the n observations, in terms of average squared Euclidean distance.

90 observations simulated in three dimensions.
Left: The first two principal component directions span the plane that best fits data. It minimizes the sum of squared distances from each point to the plane.
Right: The first two principal component score vectors give the coordinates of the projection of the 90 observations onto the plane. The variance in the plane is maximized. first three principal components of a data set span the three-dimensional hyper-plane that is closest to the n observations, and so forth.

Proportion Variance Explained:

We need to know the proportion of variance explained PVE by each one.

Calculate the total variance and separate variance explained by each one:

- The *total variance* present in a data set (assuming that the variables have been centered to have mean zero) is defined as

$$\sum_{j=1}^{p} \mathrm{Var}(X_j) = \sum_{j=1}^{p} \frac{1}{n} \sum_{i=1}^{n} x_{ij}^2,$$

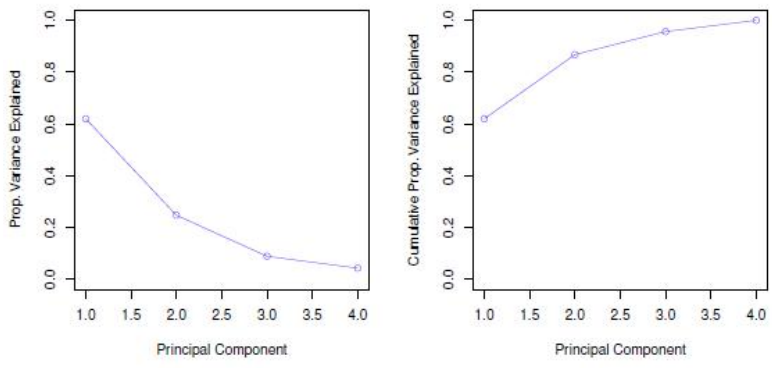
and the variance explained by the mth principal component is

$$\mathrm{Var}(Z_m) = \frac{1}{n} \sum_{i=1}^{n} z_{im}^2.$$

- It can be shown that $\sum_{j=1}^{p} \mathrm{Var}(X_j) = \sum_{m=1}^{M} \mathrm{Var}(Z_m)$, with $M = \min(n-1, p)$.

Thus the proportion is: $\dfrac{\sum_{i=1}^{n} z_{im}^2}{\sum_{j=1}^{p} \sum_{i=1}^{n} x_{ij}^2}$ and the PVE would sum to one.

Show the cumulative:



How many principal components should we use?

Cross validation not available: cross validation should have response to measure but we do not have labels/response here

We can use PCA to reduce dimensions first. The scree plot can be used as a guide to look for elbow! Shown above!

3. K-means clustering

It refers to a very broad set to find sub-groups, clusters in a data set.

Find similar groups of observations

Compare PCA and clustering:

PCA looks for a low-dimensional representation of the observations that explain a good fraction of the variance. Clustering looks for homogeneous sub-groups among the observations like market segmentation.

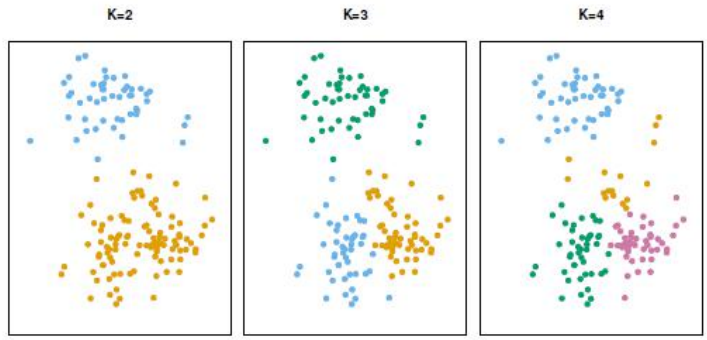
Methods: K-means clustering:

Partition the observations into a specified number of clusters

hierarchical clustering:

Do not know how many clusters in advance. We draw a dendrogram that allows us to view at once the clustering obtained.

K-means clustering: sample output of the clustering procedure



Details of K-means clustering:

Let $C_1, \ldots, C_K$ denote sets containing the indices of the observations in each cluster. These sets satisfy two properties:

1. $C_1 \cup C_2 \cup \ldots \cup C_K = \{1, \ldots, n\}$. In other words, each observation belongs to at least one of the K clusters.
2. $C_k \cap C_{k'} = \emptyset$ for all $k \neq k'$. In other words, the clusters are non-overlapping: no observation belongs to more than one cluster.

For instance, if the ith observation is in the kth cluster, then $i \in C_k$.

Good clustering: one for which the within-cluster variation is as small as possible:

The within-cluster variation for cluster C_k is a measure $WCV(C_k)$ of the amount by which the observations within a cluster differ from each other.

$\underset{C_1,\ldots C_K}{\text{minimize}}\left\{\sum_{k=1}^{K} WCV(C_k)\right\}$, thus we want to partition the observations into K clusters such that the total within-cluster variation summed over all K clusters is as small as possible.

Define within-cluster variation:

- Typically we use Euclidean distance

$$\text{WCV}(C_k) = \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2, \qquad (3)$$

where $|C_k|$ denotes the number of observations in the kth cluster.

- Combining (2) and (3) gives the optimization problem that defines K-means clustering,

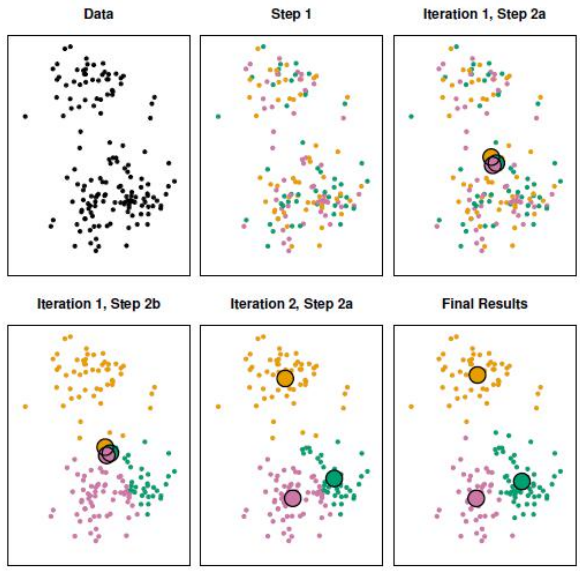
$$\underset{C_1,\ldots,C_K}{\text{minimize}} \left\{ \sum_{k=1}^{K} \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2 \right\}. \qquad (4)$$

K-means clustering algorithm:

1. random assign a number from 1 to K to each of the observations. Initial
2. Iterate until the cluster assignments stop changing:

2.1 For each of the K clusters, compute the cluster center. The k-th cluster center is the vector of the p feature means for the observations in the k-th cluster.

2.2 Assign each observation to the cluster whose center is closest (where closest is defined using Euclidean distance)



The algorithm reaches local minimum not global minimum.
Local means the derivative of function is zero but may not the lowest.
If we start from different place, the result may be different.

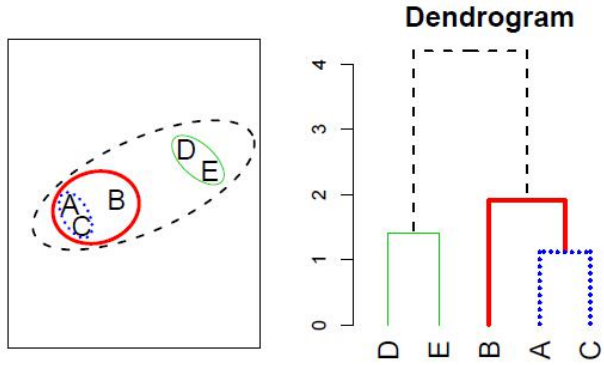
4. Hierarchical clustering

We need to specify the number of clusters first in K-means clustering.
Hierarchical clustering does not require to particular choice of K.
Methods: bottom-up or agglomerative clustering.

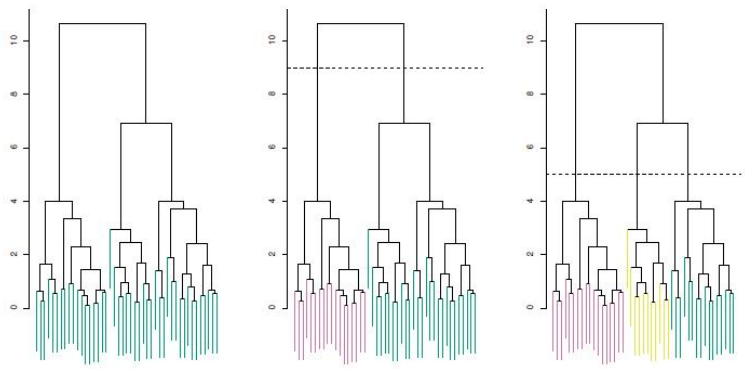
Hierarchical clustering idea:

Build a hierarchy in a bottom-up fashion:

1. Start with each point in its own cluster
2. Identify the closest two clusters and merge them
3. Repeat until a single cluster



The height of dendrogram shows the distance between clusters
Then make a cut in dendrogram to specify several clusters:



Now about the linkage of cluster: several methods to calculate

All these are based on Euclidean distance right now:

Complete:

Maximal inter-cluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the largest of these dissimilarities.

Single:

Minimal inter-cluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the smallest of these dissimilarities.

Average:

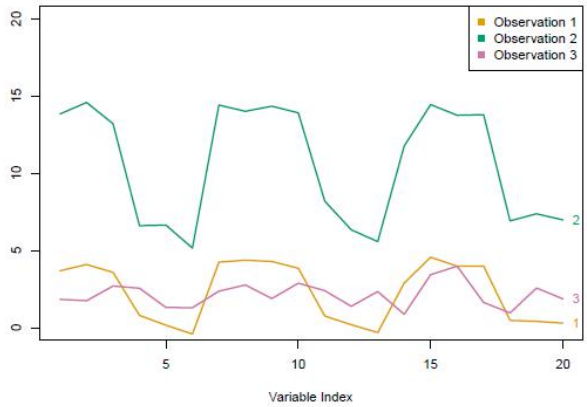
Mean inter-cluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the average of these dissimilarities.

Center:

Dissimilarity between the center for cluster A (a mean vector of length p) and the center for cluster B. Center linkage can result in undesirable inversions.

Choice of dissimilarity measure:

An alternative is to use correlation-based distance and it considers two observations to be similar if their features are highly correlated.



Three observations with measurements on 20 variables are shown. Observations 1 and 3 have similar values for each variable and so there is a small Euclidean distance between them. But they are very weakly correlated, so they have large correlation-based distance. On the other hand, observations 1 and 2 have quite different values for each variable, so there is a large Euclidean distance between them. But they are highly correlated, so there is small correlation-based distance between them.

Some practical issues:

Scaling of the variables matters! We calculate the distance thus they have to be on same scale!

How to choose proper number of clusters? We can find the longest drop-off in the dendrogram and cut there.

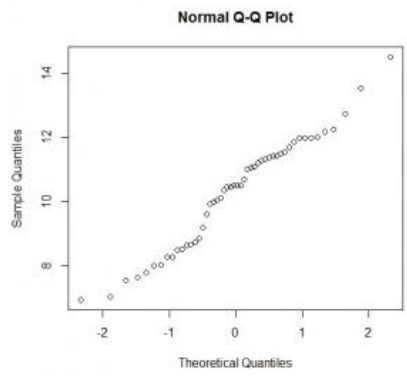
5. Example of Hierarchical clustering

   Skip

Diagnosis of linear regression assumption is common in interview:

1. Linearity -- residual plot vs X, random distributed around zero
2. Constant variance -- residual vs X. if shape appears, reject constant variance.
3. Independence -- residual vs time
4. Normality: error of distribution: check QQ plot distributed.

Theoretical quantiles and sample quantiles plot



What assumption do you need to check for logistic regression?

Prerequisite:

1. No request for linear relationship between dependent and independent variables. Error terms do not need to be normally distributed.
2. The dependent variables are not measured on an interval or ratio scale.

Assumptions:

1. Logistic regression requires observations to be independent of each other
2. Independent variables should not be highly correlated with each other since logistic regression is low tolerant of col-linearity among independent variables
3. Independent variables should be linearly related to log odds
4. It typically requires a large sample size