An Overview of Statistical Machine Learning Part I

Babak Shahbaba, Ph.D.

Associate Professor, Department of Statistics
University of California, Irvine

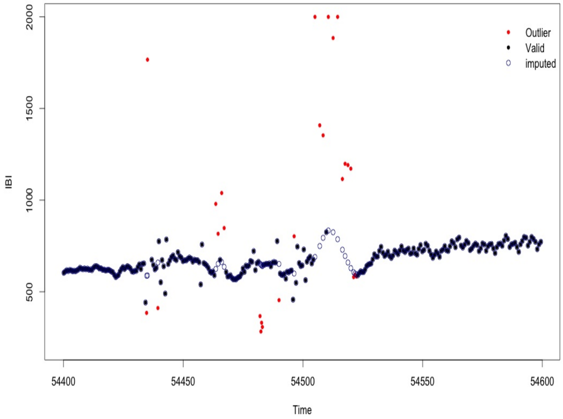
Irvine, CA
July 11, 2017

Statistical methods in machine learning

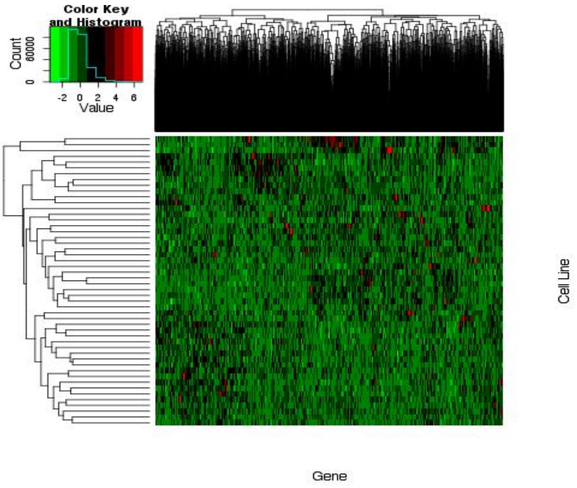
- We focus on complex and/or high dimensional problems
- We discuss statistical methods designed for such problems
- Our overall objective is to use these statistical methods to make decisions under uncertainty
- Typically, our decisions are in the form of predictions
- To achieve this objective, we need to learn from the data:
    - detect patterns
    - discover relationships
    - reduce dimensionality and complexity

Motivating Examples

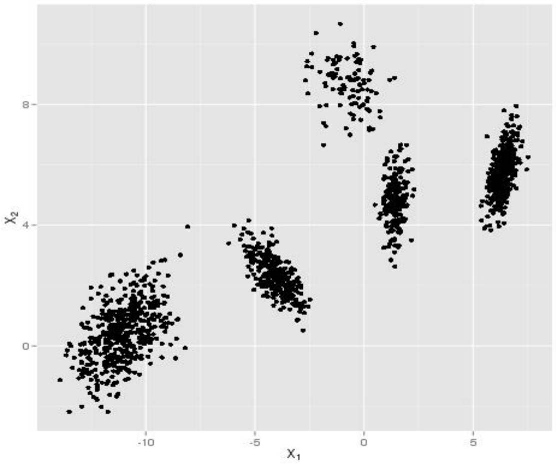
Automatic data cleaning and processing



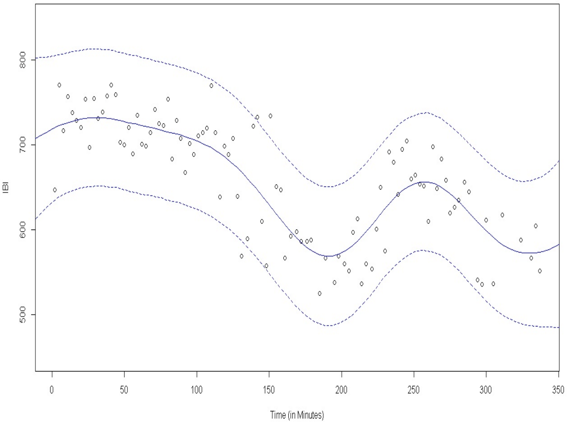
High throughput studies



Clustering objects



Regression



Classification

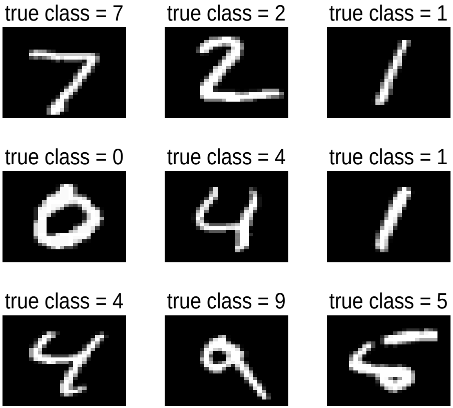


Figure: Fig1.5a in Murphy (2012)

Classification

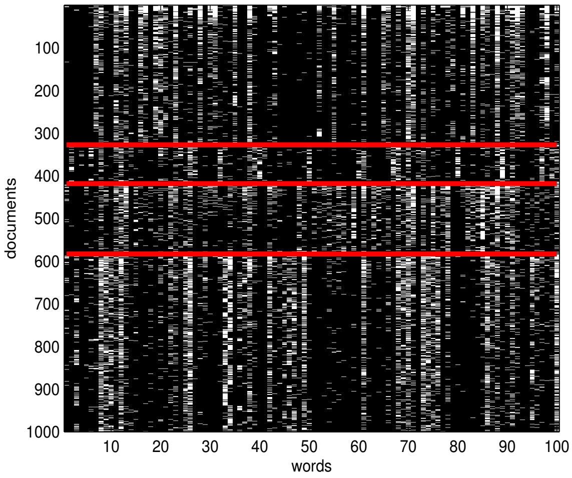


Figure: Fig1.2 in Murphy (2012)

Discovering networks

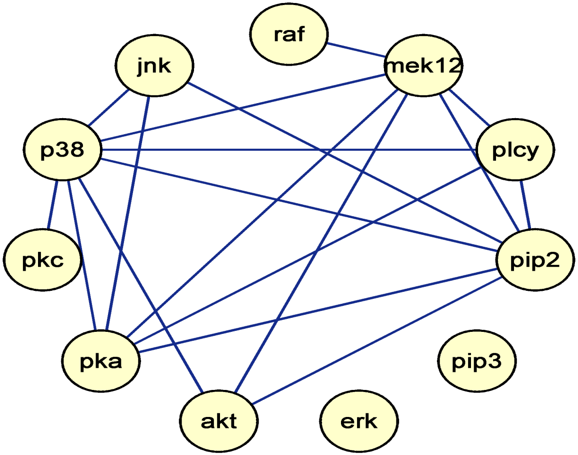


Figure: Fig1.11 in Murphy (2012)

Some Preliminary Concepts

Supervised vs. unsupervised learning

- Learning problems discussed in machine learning are divided into two main categories
  - Supervised learning: our objective is to find (learn) a mapping from a set of inputs (predictors, attributes, covariates, features), x, to outputs (response, target, outcome), y.
  - Unsupervised learning: there are no clear, well-defined outputs; our objective is to discover interesting patterns, relationships, and structures in a set of inputs, x.
  - Semi-supervised learning
  - Reinforcement learning

Supervised learning

- Regression
  - Continuous response variables: $y \in \mathbb{R}$
  - Forecasting
  - Longitudinal analysis
  - Time series analysis
  - Spatio-temporal analysis
- Classification
  - Categorical response variable: $y \in \{1, \dots, C\}$
  - Document classification
  - Image classification
  - Face detection and recognition

Discriminative vs. generative models

- Discriminative classification models
  - We model $P(y|x)$, $y \in \{1, \dots, C\}$, and use it to predict the class given inputs.
  - Tends to be less sensitive to outliers
  - Possible to use arbitrary preprocessing of inputs
- Generative classification models
  - We model $P(x|y)$, $y \in \{1, \dots, C\}$, and use Bayes' rule to find $P(y|x)$ in order to make predictions given inputs.
  - Easy to fit
  - Can handle missing features and unlabeled data
  - If the assumed distribution of inputs is correct, they tend to perform better since they use more information to estimate parameters.

Unsupervised learning

- Density estimation
  - Finite mixture models
  - Infinite mixture models
- Clustering
  - K-means clustering
  - Hierarchical clustering
  - Finite mixture models
- Dimensionality reduction
  - Principal component analysis (PCA)
  - Factor analysis (FA)
  - Independent component analysis (ICA)

Parametric vs. nonparametric

- In general, supervised learning involves finding $P(y|x)$, whereas, unsupervised learning involves finding $P(x)$.
- More specifically, we use $P(y|x, \theta)$ and $P(x|\theta)$, where θ represent all the model parameters.
- Parametric models: Number of parameters is fixed (finite).
- Nonparametric models: Number of parameters grows (possibly to infinity) with the amount of data.

Some Modeling Challenges

Curse of dimensionality

- Curse of dimensionality (Bellman, 1961) refers to challenges imposed by high dimensional data.
- These challenges are mainly due to sparsity.
- Hastie et. al. (2009) have demonstrated this using a simple example.

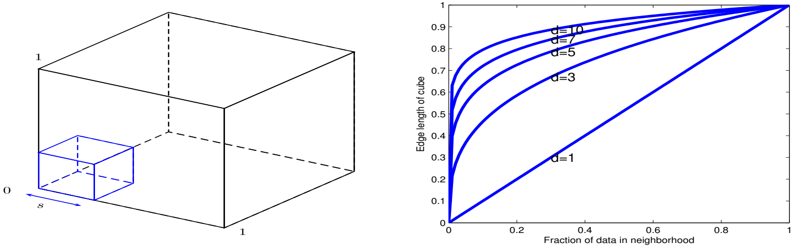
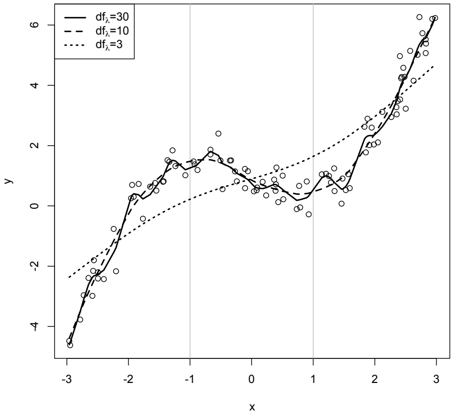


Figure: Fig1.6 in Murphy (2012)

Overfitting

- Overfitting is a common challenge in applying machine learning methods.
- It refers to situations when a model performance well on the observed data, but performs poorly on future observations.
- This is mainly due to the fact that many machine learning methods can lead to arbitrarily complex models.
- As a result, these models identify patterns peculiar to the observed data but not generalizable to the whole population.

Overfitting



Occam's razor

- We will talk about techniques for controlling complexity throughout this course.
- In general, it is recommended to use more complex models only when they result in substantial (i.e., statistically significant) improvement in performance (i.e, substantial decrease in deviance).
- The above principle is widely known as Occam's razor stating that "entities should not be multiplied beyond necessity", or in simple words: "everything equal, we should use the simplest solution".
- Ideally, we prefer to use complex models that include simpler models as special cases and have an intrinsic complexity-controlling mechanism.

Model comparison and model selection

- For predictive models, we can define our goal as finding the model with the lowest expected loss, in this case the lowest expected prediction error $EPE = E[L(y, \hat{y})]$.

$$\begin{aligned} EPE &= E(y^2) + E(\hat{y}^2) - 2E(y\hat{y}) \\ &= \mathrm{Var}(y) + E(y)^2 + \mathrm{Var}(\hat{y}) + E(\hat{y})^2 - 2E(y)E(\hat{y}) \\ &= \mathrm{Var}(y) + (E(y) - E(\hat{y}))^2 + \mathrm{Var}(\hat{y}) \\ &= \mathrm{Var}(y) + \mathrm{Bias}^2(\hat{y}) + \mathrm{Var}(\hat{y}) \end{aligned}$$

- Note that future observations y are independent of $\hat{y}$.

Bias-variance tradeoff

- In the above derivation of EPE, the first term, $\mathrm{Var}(y)$, reflects the random variation of the response variable regardless of what model we use.
- Therefore, only the last two terms depend on our model for $\hat{y}$; so we should try to minimize these two terms.
- In general, there is a tradeoff between bias and variance: complex models tend to have lower bias and higher variance, whereas simple models tend to have higher bias and lower variance.

Prediction error

- We typically evaluate predictive models based on their prediction error presented as the expectation of an assumed loss function, L,

$$\mathrm{Err} = E[L(y, \hat{y})]$$

  where y is the observed value of the response variable, and $\hat{y}$
- For regression models, we usually set $L(y, \hat{y}) = (y - \hat{y})^2$
- For classification models, we usually set $L(y, \hat{y}) = 1$ when $y \neq \hat{y}$, and zero otherwise; this is known the 0-1 loss function

Prediction error

- We use the observed data to estimate error
- Building a predictive model based on the observed data and evaluating it based on the same data will provide optimistic estimates of prediction error
- The optimistic prediction error on the the training data set itself is called the “apparent error”
- To avoid this issue, we usually use an independent test set to estimate prediction error
- If the sample size is large enough, we can divide the observed data into two independent training and test sets

Cross-validation

- When the sample size is relatively small, we recycle the data using K-fold cross-validation (CV)
  - Split the data into K roughly equal parts
  - For $k = 1, \ldots, K$, treat the kth part as the test set to evaluate the model trained on the remaining $K - 1$ parts
  - Obtain the CV estimate of prediction error as

$$\widehat{\mathrm{Err}}_{CV} = \frac{1}{n} \sum_{i=1}^{n} L(y_i, \hat{y}_i)$$

    where $\hat{y}_i$ is the prediction using the parts that do not include the ith observation
- Setting $K = n$ is called "leave-one-out"

Training, validation, and test

- Many model building strategies involve fine tuning a set of parameters, whose values affect model performance.
- For example, the tuning parameter could be a vector of p binary indicators $r_1, \dots, r_p$ such that if $r_j = 1$, we include the j^{th} predictor in our model.
- We usually choose an appropriate values for such parameters based on the performance of the model on a third dataset (usually a subset of the training set) called the *validation* set.
- After we decided the values of these tuning parameters, and fix the model, we evaluate its performance on the test set.

Clustering

Clustering

- Building statistical models to identify the underlying structure of data without focusing on of a specific outcome variable is known as **unsupervised learning**.
- An important class of unsupervised learning is **clustering**, which is commonly used to identify sub-groups within a population.
- In general, cluster analysis refers to methods that attempt to divide the data into sub-groups such that the observations within the same group are more similar compared to the observations in different groups.
- In this lecture, we discuss clustering methods that are not based on any probabilistic model; clustering methods based some underlying probabilistic model are discussed in the next lecture

Clustering

- The core concept in any cluster analysis is the notion of similarity and dissimilarity. It is common to quantify the degree of dissimilarity based on a **distance** measure defined for a pair of observations.
- The most commonly used distance measure is the **squared Euclidean distance**:

$$d_{ij} = (x_i - x_j)^2$$

  where d_{ij} refers to the distance between observations i and j, x_i is the value of random variable X for observation i and x_j is the value for observation j.
- In general, if we measure p random variables, $X_1, \ldots, X_p$, the squared Euclidean distance between two observations i and j in our sample is

$$d_{ij} = (x_{i1} - x_{j1})^2 + \ldots + (x_{ip} - x_{jp})^2$$

K-means Clustering

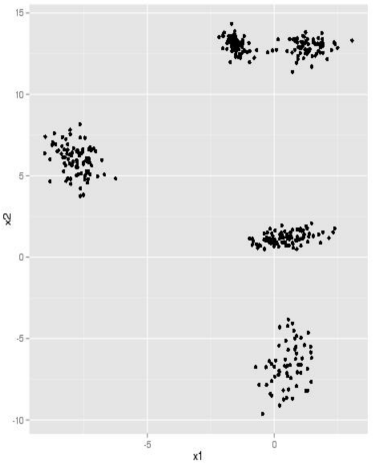
- *K-means clustering* is a simple algorithm that uses the squared Euclidean distance as its measure of dissimilarity.
- We start by specifying the number of clusters (groups) K. This is the number of groups that we believe exist in the population.
- Our goal is then to group the n observations into K clusters, such that the overall measure of dissimilarity is small within groups and large between groups.
- Initially, we divide the observations into K groups randomly.
- Then the algorithm iteratively improves the clusters.

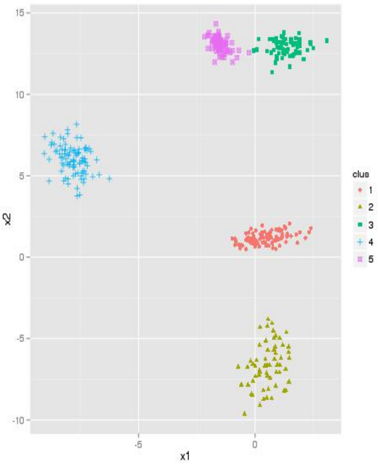
K-means Clustering

- For each cluster, we define the **center** or **centroid** as an imaginary observation, whose measurements are the sample average of all observations in that cluster.
- After randomly partitioning the observations into K groups and finding the center of each cluster, the K-means algorithm finds the best clusters by iteratively repeating these steps:
  1. For each observation, find its squared Euclidean distance to all K centers, and assign it to the cluster with the smallest distance.
  2. After regrouping all the sampling units into K clusters, re-calculate the K centers.

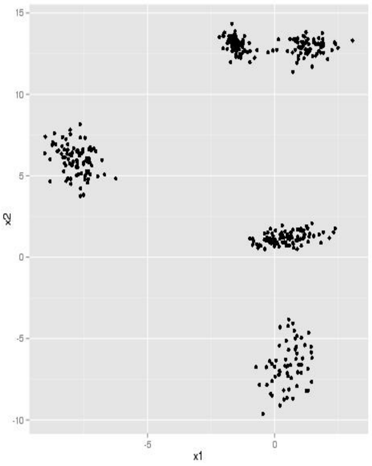
  We repeat the above steps until the clusters do not change (i.e., the centers remain the same after each iteration).

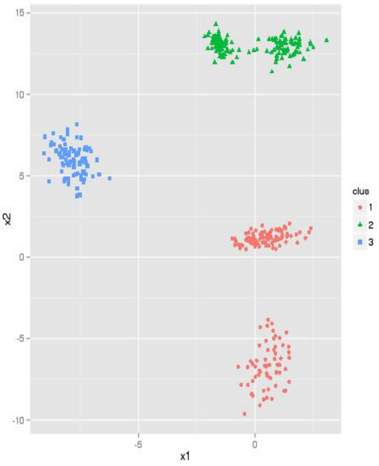
Example: $K = 5$





Example: $K = 3$



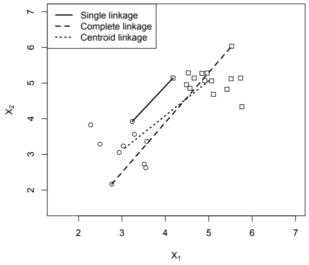


Hierarchical clustering

- There are two general algorithms for hierarchical clustering Hastie et. al.:
  - **Agglomerative** (bottom-up): We start at the bottom of the tree, where every observation is a cluster (i.e., there are n clusters). Then we merge two of the clusters with the smallest degree of dissimilarity (i.e., the two most similar clusters). Now we have $n-1$ clusters. We continue merging clusters until we have only one cluster (the root) that includes all observations.
  - **Divisive** (top-down): We start at the top of the tree, where all observations are grouped in a single cluster. Then we divide the cluster into two new clusters that are most dissimiliar. Now we have two clusters. We continue splitting existing clusters until every observation is its own cluster.

Hierarchical clustering

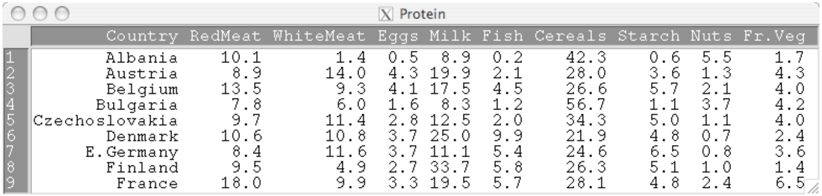
- Of the above two strategies, agglomerative algorithm is the most common.
- Both algorithms, however, require a measure of dissimilarity between two clusters.
- We need to specify a distance measure for two clusters analogous to the distance measure we defined for two observations.



Hierarchical clustering

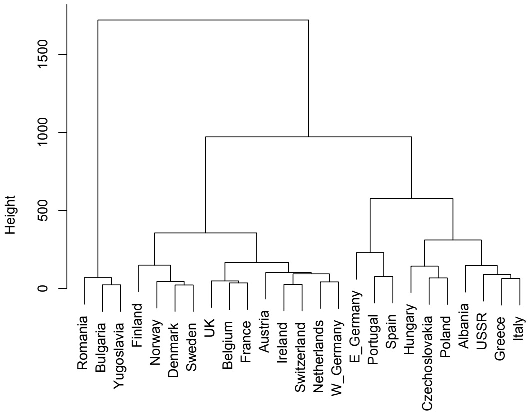
- These are some common methods to calculate the overall distance between two clusters:
  - *Single linkage* clustering use the minimum d_{ij} among all possible pairs as the distance between the two clusters. This is the distance between two observations, one from each cluster, that are closest to each other.
  - *Complete linkage* clustering uses the maximum d_{ij} as the distance between the two clusters. This is the distance between two observations, one from each cluster, that are furthest apart.
  - *Average linkage* clustering uses the average d_{ij} over all possible pairs as the distance between the two clusters.
  - *Centroid linkage* clustering finds the centroids of the two clusters and uses the distance between the centroids as the distance between the two clusters.

Example: Protein consumption in 25 European countries

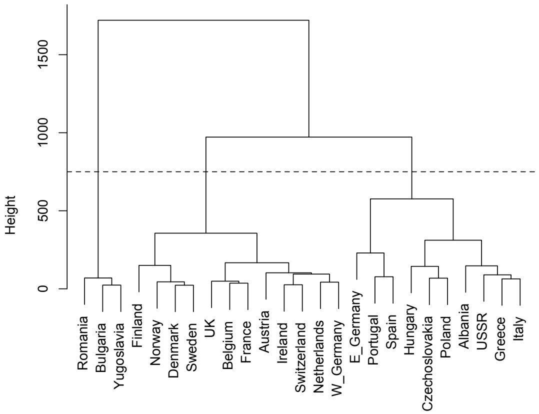

Protein

	Country	RedMeat	WhiteMeat	Eggs	Milk	Fish	Cereals	Starch	Nuts	Fr.Veg
1	Albania	10.1	1.4	0.5	8.9	0.2	42.3	0.6	5.5	1.7
2	Austria	8.9	14.0	4.3	19.9	2.1	28.0	3.6	1.3	4.3
3	Belgium	13.5	9.3	4.1	17.5	4.5	26.6	5.7	2.1	4.0
4	Bulgaria	7.8	6.0	1.6	8.3	1.2	56.7	1.1	3.7	4.2
5	Czechoslovakia	9.7	11.4	2.8	12.5	2.0	34.3	5.0	1.1	4.0
6	Denmark	10.6	10.8	3.7	25.0	9.9	21.9	4.8	0.7	2.4
7	E.Germany	8.4	11.6	3.7	11.1	5.4	24.6	6.5	0.8	3.6
8	Finland	9.5	4.9	2.7	33.7	5.8	26.3	5.1	1.0	1.4
9	France	18.0	9.9	3.3	19.5	5.7	28.1	4.8	2.4	6.5

Example: Protein consumption



Example: Protein consumption



Dimensionality Reduction

Dimensionality Reduction

- We now discuss several dimensionality reduction methods such as principal component analysis (PCA), factor analysis (FA), and independent component analysis (ICA)
- We are mainly interested in unsupervised learning techniques for presenting high-dimensional data in low dimensional spaces, hoping to make the underlying structure in the data and patterns easier to see
- Very often, such unsupervised learning methods for dimensionality reduction are used a preprocessing step for supervised learning methods (discussed later) without taking the outcome variable into account

Principal component analysis

- For a set of variables, we denote the centered matrix of of observed data as x
- The principal components are a set of orthonormal basis, $v_1, v_2, \ldots, v_p$, in the column space of x such that
  - v_1 is the basis with the largest sample variance
  - v_2 is the basis with the second largest sample variance that it is orthogonal to v_1
  - v_3 is the basis with the j^{th} largest sample variance that it is orthogonal to v_1, v_2
  - and so forth

Principal component analysis

- To find these principal components, we first find the eigenvectors of the covariance matrix $s = x^\top x / n$, and then order them based on the descending order of their corresponding eigenvalues, λ_j,

$$sv_j = \lambda_j v_j, \qquad j = 1, \ldots, p, \qquad \lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_p$$

- Recall that given a matrix s, eigenvectors are special vectors, v_j, such that we have $sv_j = \lambda v_j$ so v_j remains on the same line, but it either stretches, shrinks, reverses directions, or stays unchanged
- For each eigenvectors v, λ is the corresponding eigenvalue

Eigenvalues and eigenvectors (review)

- Suppose $s_{p \times p}$ has p independent eigenvectors, $v_1, \ldots, v_p$, which are the columns of an eigenvector matrix v
- The corresponding eigenvalues form a diagonal matrix $\Lambda = diag(\lambda_1, \ldots, \lambda_p)$
- Then, we can write them in a matrix form $sv = v\Lambda$, from which we can get

$$\begin{aligned} v^{-1}sv &= \Lambda \\ s &= v\Lambda v^{-1} \end{aligned}$$

- For symmetric matrices (such as covariance matrices), we have real eigenvalues and orthogonal eigenvector matrix. Therefore,

$$s = v\Lambda v^{\top}$$

Alternative view based on SVD

- Instead of explaining PCA based on finding eigenvectors of the square matrix $x^\top x$, we can explain it in terms of a singular value decomposition of x itself

$$x = u \Sigma v^\top$$

  where u and v, called left and right singular vectors, are orthonormal and Σ, called singular values, is a diagonal matrix

- Then,

$$x^\top x = v \Sigma^\top u^\top u \Sigma v^\top = v \Sigma^\top \Sigma v^\top = v \Lambda v^\top$$

  where Λ is a diagonal matrix that contains the eigenvalues of $x^\top x$

- Therefore, v contains the orthogonal eigenvectors of $x^\top x$, and the diagonal elements σ_i^2 are the positive eigenvalues of $x^\top x$

Singular value decomposition (SVD)

- Ordering $\sigma_1 \geq \ldots \geq \sigma_r > 0$, where r is the rank, we can write

$$x = u_1 \sigma_1 v_1^\top + \ldots + u_r \sigma_r v_r^\top = \sum_{i=1}^{r} \sigma_i u_i v_i^\top$$

- Left and right singular vectors are also known as Karhunen-Loève bases
- PCA is sometimes referred to as the Karhunen-Loève transformation

Principal component analysis

- After we find the eigenvectors v and order them according to their corresponding eigenvalues in decreasing order, we obtain a new set of derived variables, $z = (z_1, \dots, z_p)$, as follows:

$$z = xv$$

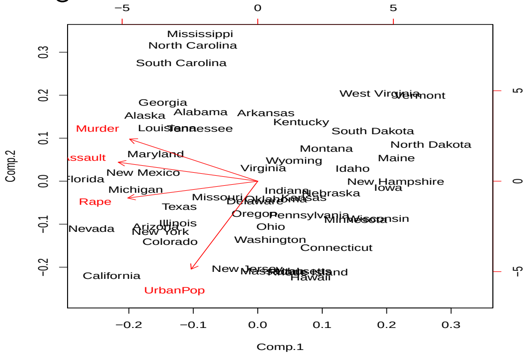
- The columns of z are called *scores*, and the columns of v are called *loadings*
- Note that the sample variance of z_j, i.e., the j^{th} column of z, is

$$\mathrm{Var}(z_j) = z_j^T z_j / n = v_j^T x^T x v_j / n = v_j^T s v_j = \lambda_j$$

Therefore, the first column of z has the highest variance, the second column has the second highest variance, and so forth.

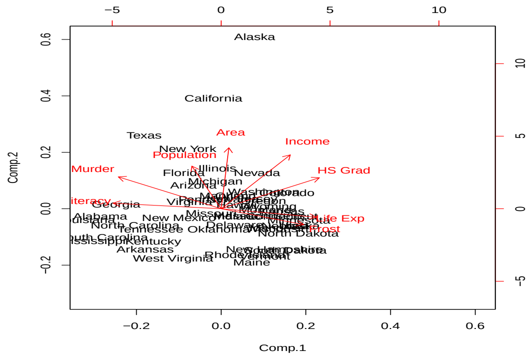
Example: US Arrests

- It is common to present the PCA results in a biplot using the first two principal components
- The following biplot shows the results based on the `USArrests` dataset, which is available in R, and according to its help file, it contains statistics in arrests per 100,000 residents for assault, murder, and rape in each of the 50 US states during 1973.



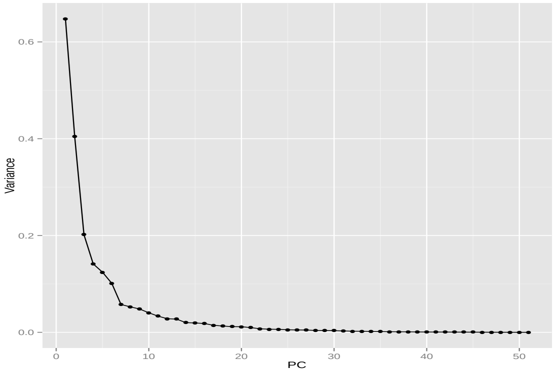
Example: States

- The following biplot is based on the `state.x77` dataset (also available in R)
- This dataset provides 8 different statistics (see the help file) on the 50 states of the United States of America



Example: Neurology data

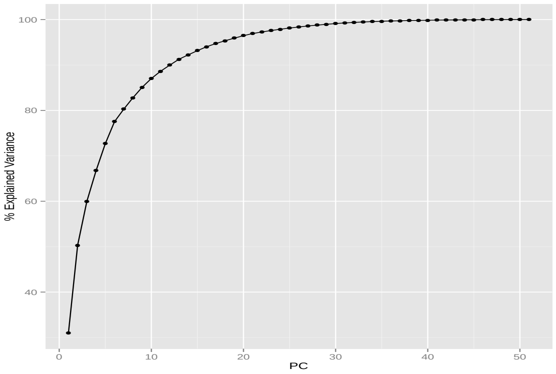
- To reduce dimensionality of x, we could choose the first q columns of z to represent the observed data
- For this, it is common to use the *scree* plot, which is the plot of all the eigenvalues (variances) in decreasing order
- The following is based on a neurology dataset (Burke et. al., 2014)



Example: Neurology data

- Alternatively, we can plot the cumulative percent of variance explained, which is calculated as follows:

$$\frac{\sum_{j=1}^{q} \lambda_j}{\sum_{j'=1}^{p} \lambda_{j'}}$$



Factor Analysis (FA)

- PCA involves a standardized linear projection which maximizes the variance in the projected space (Hotteling, 1933; Tipping and Bishop, 1999)
- $v_1, \ldots, v_q$ are the top q dominant eigenvector (associated with the largest eigenvalues) of the sample covariance
- Using the corresponding scores z, we can approximate x,

  $$\hat{x} = zv^\top$$

  which minimizes the reconstruction error (error is zero if $q = p$)

  $$\sum_{i=1}^{n} \|x_i - \hat{x}_i\|^2$$

- The main shortcoming of this approach is the lack of any underlying probabilistic model

Factor Analysis (FA)

- To address this issue, we use *Factor Analysis* (FA) models, which include *Probabilistic Principal Component Analysis* (PPCA) as a special case
- Instead of the above deterministic model, we use the following probabilistic model

$$\begin{aligned} x_i &= wz_i + \mu + \epsilon_i \\ z_i &\sim N(0, I) \\ \epsilon_i &\sim N(0, \Psi) \end{aligned}$$

- Here, z are q-dimensional latent variables
- w is a $p \times q$ matrix called the *factor loading matrix*

Factor Analysis (FA)

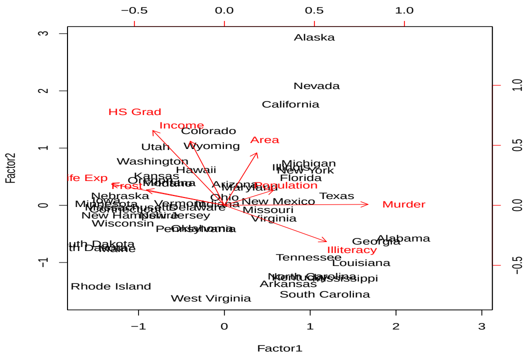
- Marginalizing over latent variables, we are in fact modeling the distribution of the observed data x as a multivariate normal

$$x_i \quad \sim \quad N(\mu, ww^\top + \Psi)$$

- That is, FA is a low rank representation of a multivariate normal distribution

Example: States

- The following biplot is based on factor analysis (two latent factors) of the `state.x77` dataset



Independent Component Analysis (ICA)

- The normality assumption for latent factors is very restrictive and could lead to unreasonable results
- We can relax this assumption and allow z to have any non-Gaussian distribution, but instead we assume that the components of z are independent so their joint distribution is separable,

$$P(z_i) = \prod_{j=1}^{q} P_j(z_{ij})$$

- Given z_i, we assume

$$x_i = wz_i + \epsilon_i$$

- The resulting model is known as Independent Component Analysis
- This approach is commonly used in signal processing so z are usually called source variables and w called mixing matrix

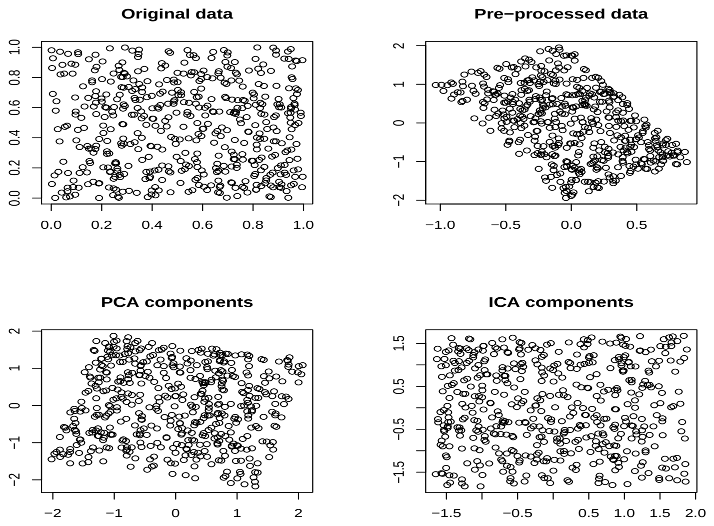
Independent Component Analysis (ICA)

- We denote the centered (by subtracting the mean) and whitened (e.g., by using PCA) data as x
- Because the data are whitened, we have $\mathrm{cov}(x) = I$; therefore, the mixing matrix is also orthogonal since $\mathrm{cov}(z)$ is also I
- Our goal is to find the mixing matrix, w
- After we estimate w, assuming a noise-free model, we can use it to find the sources

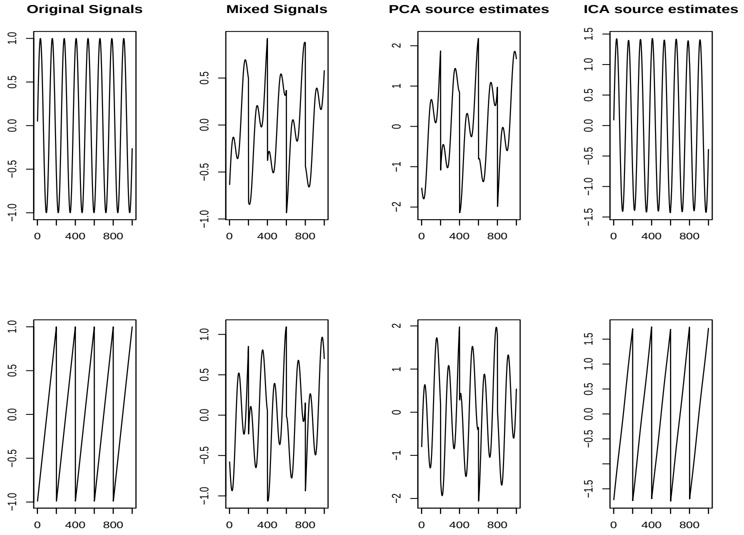
  $$z = vx$$

  where $v = w^\top$
- To estimate w, we can use maximum likelihood estimation or EM (see section 12.6 of Murphy, 2012)
- However, it is more common to use entropy-based methods

Example: Two independent uniforms (from fastICA)



Example: Two independent signals (from fastICA)



Controlling Complexity

Background

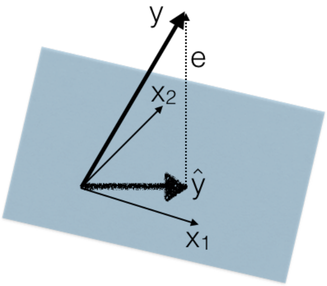
- Overly complex models tend to overfit the data.
- Here, we mainly focus on model complexity related to the number of variables included in the model.
- First, we discuss methods that use few derived variables instead of using a large number of original variable.
- Next, we discuss methods that control the number of variables and magnitude of their effects by penalizing against complexity.
- We assume the variables are standardized to have mean zero and variance 1.

Review: Least squares estimates

- Recall the least square estimates for $X\beta = y$, where X is a $n \times p$ $(n > p)$ matrix:

$$X\hat{\beta} = \hat{y}; \qquad y = \hat{y} + e$$

  where we find the best solution $\hat{\beta}$ by making e small so y and $\hat{y}$ are "close" to each other
- To this end, we minimize $\|e\|^2 = \|y - X\hat{\beta}\|^2 = (y - X\hat{\beta})^\top(y - X\hat{\beta})$



Principal component regression

- To control the complexity of regression models, we can use PCA to reduce the dimensionality of the observed data, and hence the number of parameters
- Consider the centered matrix of predictors, x
- As discussed before, we can find principal components and the corresponding score, z
- We then define a new set of derived predictors using the first q columns of z
- We can choose q using the scree plot or cross validation

Principal component regression

- Principal component regression (PCR) is a linear regression model that uses $z_1, \ldots, z_q$ instead of the original predictors,

$$y = \gamma_0 + \gamma_1 z_1 + \ldots + \gamma_q z_q + \eta$$

  where η is the random noise.
- PCR works well when the variation of y mainly occurs along the directions of high variance in the space of predictors.

Partial least squares

- We could identify a set of new bases according to the relationship between predictors and the response variable.
- The partial least squares (PLS) method performs this task as follows:
  1. Find the univariate regression coefficient $\hat{\phi}_{1j}$ of y on each x_j.
  2. Obtain the first derived input $z_1 = \sum_{i=1}^{p} \hat{\phi}_{1j} x_j$.
  3. Orthogonalize the original inputs with respect to this direction by subtracting from each x_j its projection in the direction of z_1.
  4. We repeat the above procedure to obtain z_2 up to z_q, where $q < p$.
  5. We regress y on the new derived variables $z_1, \ldots, z_q$.

Bridge regression

- We now consider regularized regression models, which shrink the regression coefficients by imposing a penalty on their magnitude.
- In *bridge regression* models (Frank and Friedman, 1993), the coefficients are obtained by minimizing residual sum of squares subject to a norm constraint on the size of regression coefficients:

$$\text{minimize } RSS(\beta) = \sum_i (y_i - \beta_0 - x_i^T \beta)^2$$

$$\text{subject to } \sum_{j=1}^{p} |\beta_j|^\gamma \leq s$$

- We usually scale and center x, and center y so we don't have to deal with β_0.

Bridge regression

- Alternatively, we can find the estimate by solving the following optimization problem instead:

$$\text{minimize } RSS(\beta) + \lambda \sum_{j=1}^{p} |\beta_j|^\gamma$$

  where $\lambda \geq 0$ is the Lagrange multiplier.
- That is, we minimize a penalized residual sum of squares.
- In the matrix form,

$$\min_{\beta} (y - x\beta)^T (y - x\beta) + \lambda \sum_{j=1}^{p} |\beta_j|^\gamma$$

Ridge regression

- When $\gamma = 2$, we obtain a special case of the bridge regression known as the *ridge regression* (Hoerl and Kennard, 1970)

$$\text{minimize } RSS(\beta) + \lambda \sum_{j=1}^{p} \beta_j^2$$

- In ridge regression, the estimates are shrunk towards zero and each other.
- The ridge regression solutions are

$$\hat{\beta}^{\text{ridge}} \quad = \quad (x^T x + \lambda I_p)^{-1} x^T y$$

Ridge regression

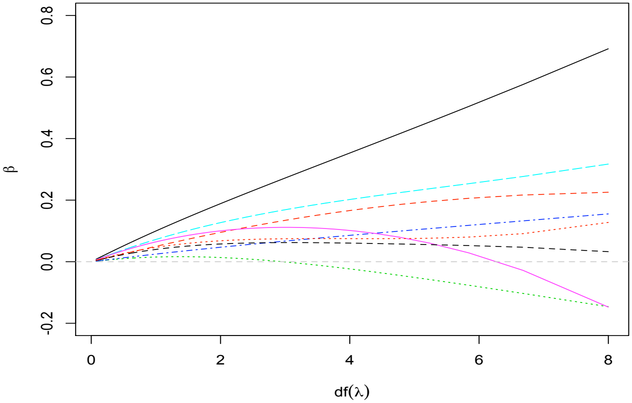
- Since $x^T x + \lambda I_p$ is non-singular as long as $\lambda > 0$, ridge regression provides a unique solution for a given λ even if $x^T x$ is not of full rank (e.g., $p > n$).
- The L_2 penalty applied to RSS shrinks the coefficients towards zero (and each other).
- The imposed penalty prevents the estimates of regression coefficients to become large.
- This is of course based on our belief that very large values of β are not very likely and should be discouraged.

Ridge regression

- The larger the value of λ, the greater the amount of shrinkage.
- However, since the effect of the penalty depends on the scale of predictors, it is common to standardize the predictors so they all have standard deviation 1.
- The estimates from ridge regression are biased but they have lower variance compared to least-squares estimates.
- The overall performance of course depends on how well we choose λ. To choose an appropriate λ, it is common to use cross validation.

Ridge regression for prostate cancer data

- The following plot shows the estimate of parameters for different values of λ based on the prostate cancer dataset (available in R).



Ridge regression for prostate cancer data

- The horizontal line shows the *effective degrees of freedom* defined as follows

$$
\begin{aligned}
df(\lambda) &= \mathrm{tr}(H_\lambda) \\
&= \mathrm{tr}[x(x^T x + \lambda I_p)^{-1} x^T] \\
&= \sum_{j=1}^{p} \frac{d_j^2}{d_j^2 + \lambda}
\end{aligned}
$$

where d_j is the jth diagonal element of D obtained from the Singular Value Decomposition (SVD): $X = UDV^\top$.

Lasso

- When $\gamma = 1$, the bridge regression becomes equivalent to the *lasso* (least absolute shrinkage and selection operator).
- Lasso is similar to ridge regression, but instead of L_2 penalty, we use the L_1 penalty $\sum_{j=1}^{p} |\beta_j|$

$$\text{minimize } RSS(\beta) + \lambda \sum_{j=1}^{p} |\beta_j|$$

- As before, the penalty results in the shrinkage of coefficients towards zero.
- However, by using the the L_1 penalty and a large enough λ, some of the coefficients could become exactly zero (i.e., become excluded from the model).

Lasso

- Figure 3.11 in Hastie et al. (2010) illustrates the difference between ridge regression and lasso.

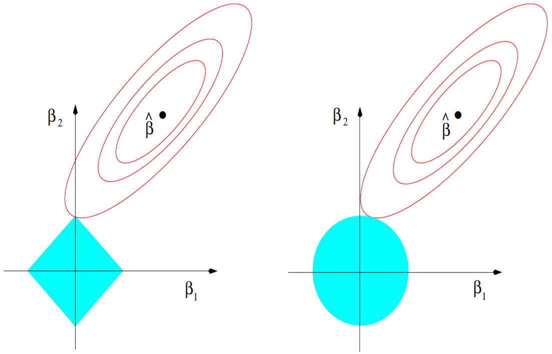
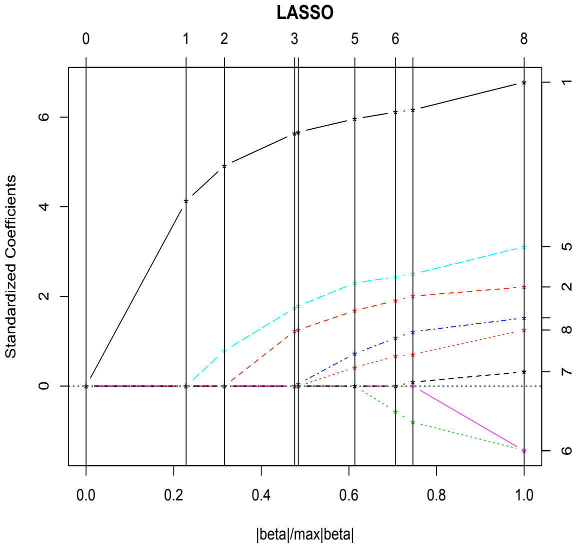


Figure: Figure 3.11 in Hastie et al. (2010). Left panel: Lasso; Right panel: Ridge regression.

Lasso

- It is clear that the L_1 penalty allows for some of the coefficients to be exactly zero.
- This is also clear from the fact that the derivative of the lasso penalty with respect to β remains constant for all $\beta > 0$, whereas in ridge regression the penalty is proportional to β.
- As the result, in ridge regression the effect of penalties reduces as β moves closer to zero, whereas in lasso, there is a continuing force until we reach zero.

Lasso for prostate cancer data



Spline

Background

- In this lecture, we discuss some strategies for relaxing the linearity assumption for the relationship between y and x in regression models.
- The main idea is to augment/replace the original input variables, x, with a set of functions, $h_m(x)$, for $m = 1, \ldots, M$, where $h_m(x) : \Re^p \rightarrow \Re$ is a transformation of x.
- This is referred to as *basis expansion* in x, and $h_m(x)$ is called a *basis function*.
- Then, we model y as a linear function of $h_1(x), \ldots, h_M(x)$,

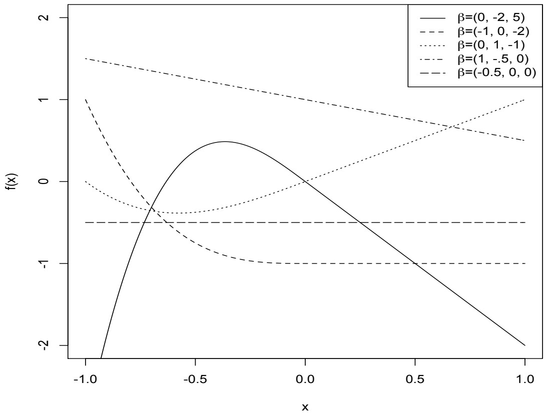
$$f(x) = \sum_{m=1}^{M} \beta_m h_m(x)$$

Background

- While the above model is in general nonlinear in x, it is linear in β.
- Therefore, we can still estimate model parameters using linear regression techniques after replacing the original variables, $x_1, \ldots, x_p$, with the new set of variables, $h_1(x), \ldots, h_M(x)$.
- As a results, we can comfortably use a wide range of basis functions in a computationally efficient way.

Background

- The following graph shows the plot of $f(x)$ with $h_1(x) = 1$, $h_2(x) = x$, and $h_3(x) = \min(0, x^3)$ and different β's.



Background

- We can create very flexible models using the above approach.
- In practice, however, we impose some constraints on the class of functions in order to control model complexity.
- For example, we can restrict the basis functions such that each basis function depends on one variable only.

$$\begin{aligned} f(x) &= \sum_{j=1}^{p} f_j(x_j) \\ &= \sum_{j=1}^{p} \sum_{m=1}^{M} \beta_{jm} h_{jm}(x_j) \end{aligned}$$

- The resulting model is called an *additive* model.

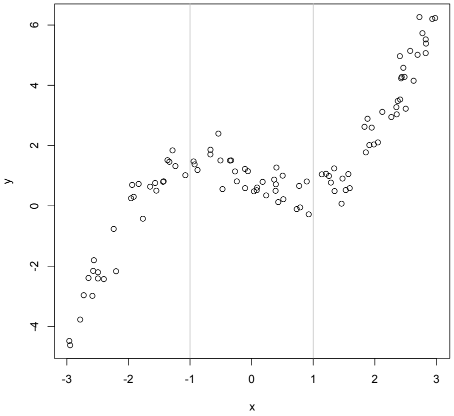
Background

- We can restrict $f(x)$ to be *piecewise constant*, where each function is constant in some region and zero outside the region.
- Or *piecewise linear*, where each function is linear in some region and zero outside the region.
- Or *piecewise polynomial*, where each function is a polynomial of degree q in some region and zero outside the region.
- We start by discussing these models for univariate cases.

Piecewise constant model

- We divide the domain of x into $K+1$ regions by specifying K *knots*, $\xi_1, \ldots, \xi_K$.
- For each region, we specify a function that is constant within the region and zero outside the region.
- For the following example, we set $\xi_1 = -1$ and $\xi_2 = 1$.

Piecewise constant model



Piecewise constant model

- We define the following three functions:

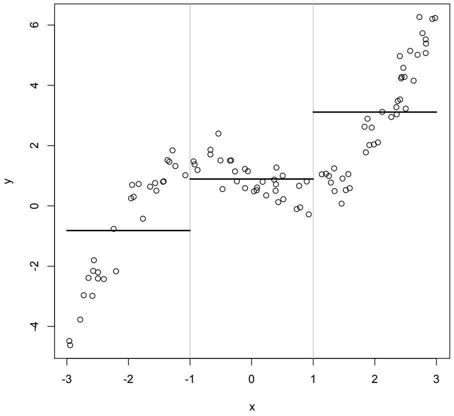
$$h_1(x) = I(x < \xi_1), \quad h_2(x) = I(\xi_1 \leq x < \xi_2), \quad h_3(x) = I(\xi_2 \leq x)$$

- The piecewise constant function model is then specified as follows:

$$f(x) = \sum_{m=1}^{3} \beta_m h_m(x)$$

- In this case, $\hat{\beta}_m = \hat{y}_m$, i.e., the mean of the observed response values in the m^{th} region

Piecewise constant model

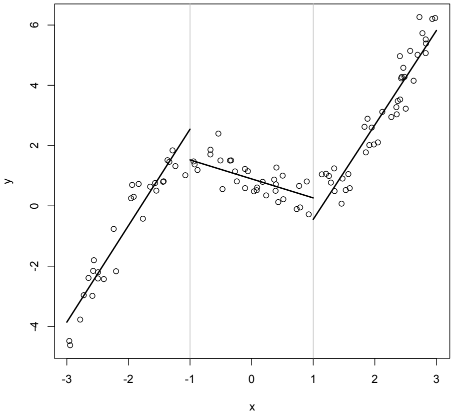


Piecewise linear model

- Next, we allow the functions to be linear in each region.
- For this, we include the following three functions along with the previous three functions, $h_1(x), h_2(x)$, and $h_3(x)$:

$$h_4(x) = h_1(x)x, \quad h_5(x) = h_2(x)x, \quad h_6(x) = h_3(x)x$$

Piecewise linear model



Piecewise linear model

- While the model seems to fit the data well, its discontinuity at the knots should be addressed.
- We usually prefer models that are continuous, since we don't believe there would be a drastic change in y as we move, for example, from $x = \xi_1^-$ to $x = \xi_1^+$.
- To make the above model continuous, we need to impose the following two constraints:

$$\beta_1 h_1(\xi_1) + \beta_4 h_4(\xi_1) = \beta_2 h_2(\xi_1) + \beta_5 h_5(\xi_1)$$
$$\beta_2 h_2(\xi_2) + \beta_5 h_5(\xi_2) = \beta_3 h_3(\xi_2) + \beta_6 h_6(\xi_2)$$

which means,

$$\beta_1 + \beta_4 \xi_1 = \beta_2 + \beta_5 \xi_1$$
$$\beta_2 + \beta_5 \xi_2 = \beta_3 + \beta_6 \xi_2$$

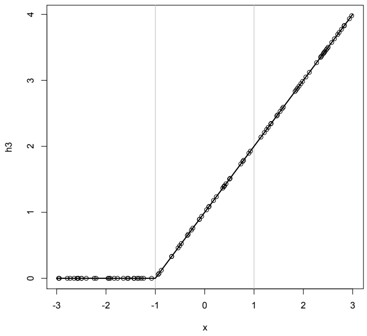
Piecewise linear model

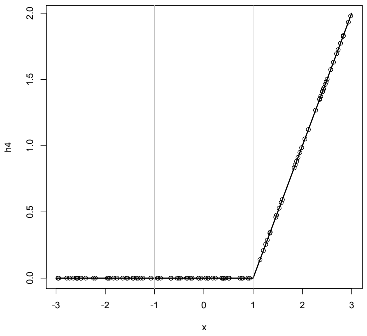
- Imposing the above two constraints reduces the dimensionality from 6 to 4.
- Alternatively, we can use the following four functions:

$$h_1(x) = 1, \quad h_2(x) = x, \quad h_3(x) = (x - \xi_1)_+, \quad h_4(x) = (x - \xi_2)_+$$

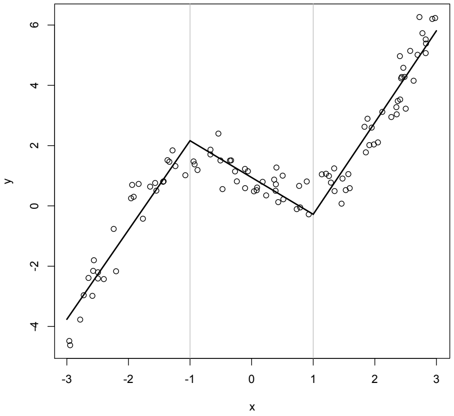
  where $(x - \xi)_+$ means $\max(0, x - \xi)$, i.e., the positive part of the function only.
- Next graph shows h_3 and h_4 functions with $\xi_1 = -1$ and $\xi_2 = 1$.

Piecewise linear model





Piecewise linear model



Piecewise polynomial model

- To make the model more flexible, we can include functions with higher degrees:

$$h_7(x) = h_1(x)x^2, \quad h_8(x) = h_2(x)x^2, \quad h_9(x) = h_3(x)x^2$$

- We impose constraints for continuity

$$\beta_1 h_1(\xi_1) + \beta_4 h_4(\xi_1) + \beta_7 h_7(\xi_1) = \beta_2 h_2(\xi_1) + \beta_5 h_5(\xi_1) + \beta_8 h_8(\xi_1)$$
$$\beta_2 h_2(\xi_2) + \beta_5 h_5(\xi_2) + \beta_8 h_8(\xi_2) = \beta_3 h_3(\xi_2) + \beta_6 h_6(\xi_2) + \beta_9 h_9(\xi_2)$$

- and *smoothness*

$$\beta_1 h_1^{'}(\xi_1) + \beta_4 h_4^{'}(\xi_1) + \beta_7 h_7^{'}(\xi_1) = \beta_2 h_2^{'}(\xi_1) + \beta_5 h_5^{'}(\xi_1) + \beta_8 h_8^{'}(\xi_1)$$
$$\beta_2 h_2^{'}(\xi_2) + \beta_5 h_5^{'}(\xi_2) + \beta_8 h_8^{'}(\xi_2) = \beta_3 h_3^{'}(\xi_2) + \beta_6 h_6^{'}(\xi_2) + \beta_9 h_9^{'}(\xi_2)$$

Splines

- In general, we can use a piecewise polynomial of order M (with degree $M-1$) that is continues and has continuous derivatives up to order $M-2$.
- We refer to these models as order-M *splines*, whose basis set would have the following form

$$\begin{aligned} h_j(X) &= X^{j-1}, \qquad j = 1, \ldots, M, \\ h_{M+\ell}(X) &= (X - \xi_\ell)_+^{M-1}, \qquad \ell = 1, \ldots, K \end{aligned}$$

where K is the number of knots.

Splines

- The continuous piecewise linear model discussed above is in fact an order-2 spline.
- In practice, we rarely need to go beyond order-4 splines, which are known as *cubic splines*.
- For cubic splines, the function, its first derivative, and its second derivative are all continuous at the knots.

Natural cubic splines

- As shown in Fig. 5.3 in Hastie et. al. (2010), the behavior of splines are erratic (high variance) near and beyond the boundaries.

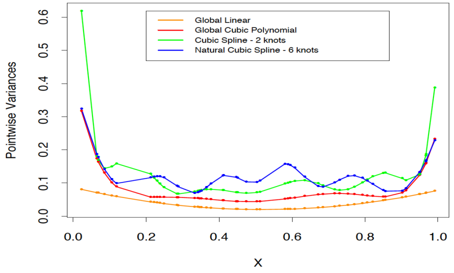


Figure: Figure 5.3 in Hastie el. al. (2010).

- To address this issue, we use *natural cubic spline*, which forces the function to be linear beyond the boundary knots.
- Therefore, the second and third derivatives are zero at the boundaries.

Smoothing splines

- Alternatively, we could use *smoothing splines*, where model complexity is controlled by regularization similar to ridge regression.
- For these models, we minimize the following penalized residual sum of squares:

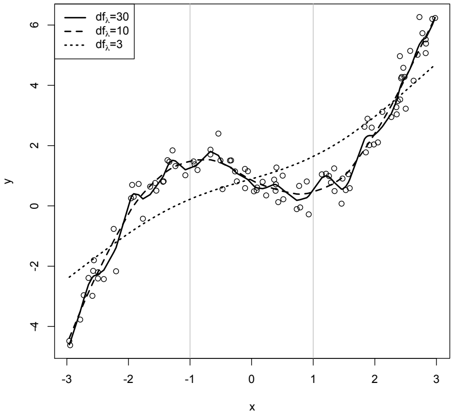
$$PRSS(f, \lambda) = \sum_{i=1}^{n} [y_i - f(x_i)]^2 + \lambda \int [f''(t)]^2 dt$$

where $f(x)$ is any function with second derivatives, and λ is a fixed smoothing parameter.

Smoothing splines

- Setting $\lambda = \infty$ results in a simple linear regression model with least squares estimates.
- Setting $\lambda = 0$ will force the model to pass through all the observed data points.
- We can choose the right λ using cross validation.

Smoothing splines


df_λ=30
df_λ=10
df_λ=3
y
x