Skabelon Bog

Mirza Hasanbasic

May 18, 2018

1. Udgave

This page is intentionally left blank.

Contents

This page is intentionally left blank.

Acronyms

Abstract

Short summary of the contents in English... a great guide by Kent Beck how to write good abstracts can be found here:

https://plg.uwaterloo.ca/~migod/research/beckOOPSLA.html

Part I

02612

Chapter 1

Lecture 1

This lecture is about `Introduction to Computational Data Analysis [OLS, Ridge, Fischer LDA, KNN]` where chapter `ESL Chapters 1, 2, 3.1, 3.2, 3.4.1, 4.1 and 13.3` should be looked upon.
From [1, p. 15]

- Supervised learning:
  - The computer is presented with example inputs and their desired outputs.
  - Relies on valid label assignments and a useful response.
- Unsupervised learning:
  - No labels are given to the learning algorithm.
  - The computer learns a structure from the input data.
  - Unsupervised learning can be a goal in itself (discovering hidden patterns in data) or a means towards an end.
    * Feature generation
    * Outlier detection

1.1 Chapter 2

In this section we will look into supervised learning. Where we look into Least Squares and Nearest Neighbors.

1.2 Chapter 3

We are looking into Linear Methods for Regressions. The Linear Regression models and Least Squares and the Ridge Regression.

1.3 Chapter 4

Here we are looking at Linear Methods for Classification

1.4 Chapter 13

Finally we are looking at the k-Nearest Neighbor classifiers

1.4.1 Terminology

The Data Matrix $\boldsymbol{X}$.

- Rows correspond to samples
- Columns correspond to variables
- Size of $\boldsymbol{X}$ is $(n \times p)$, n is samples and p variables

Where the response variable $\boldsymbol{y}$, that is usually a $(n \times 1)$ vector and known as the output.

The model coefficients β a $(p \times 1)$ vector

The model error e a $(n \times 1)$ vector

The prediction error ε a $(n \times 1)$ vector and also known as the residual

A regression model that is linear in β can be written as

$$\boldsymbol{y} = \boldsymbol{X}\beta + e$$

from lecture [1, p. 28]

- Ordinary Least Squares
- Bias-Variance
- Expected Prediction Error
- Ridge Regression

1.5 Linear

Given a continuous response measurement, find the relation of this variable to a set of input variables.

$$\boldsymbol{y} = \boldsymbol{X}\beta + e$$

and example to this can be, predict house prices based on interest rate, unemployment rate, region, size and building year.

1.5.1 Least Squares

The linear model we have vector input $X^T = (X_1, X_2, ..., X_p)$, we predict the output Y via the model

$$\hat{Y} = \hat{\beta}_0 + \sum_{j=1}^{P} X_j \hat{\beta}_j$$

where $\hat{\beta}_0$ is intercept, also known as *bias*. Write up with dummy, so we have, the first variable 1 in X

$$\hat{\boldsymbol{Y}} = \boldsymbol{X}^T \hat{\beta}$$

And X is a column vector.
The most popular model is *Least Squares*, we wish to minimize the residual sum of squares as seen in [13, p. 12]

$$\text{RSS}(\beta) = \sum_{i=1}^{N} (y_i - x_i^T \beta)^2$$

and in matrix notation this is

$$\text{RSS}(\beta) = (\boldsymbol{y} - \boldsymbol{X}\beta)^T (\boldsymbol{y} - \boldsymbol{X}\beta)$$

where $\boldsymbol{X}$ is an $N \times p$ matrix with each row an input vector, and $\boldsymbol{y}$ is an N-vector of the outputs in the training set. Differentiating w.r.t. β we get the normal equations

$$\boldsymbol{X}^T (\boldsymbol{y} - \boldsymbol{X}\beta) = 0$$

and the unique solution is given by

$$\hat{\beta} = (\boldsymbol{X}^T \boldsymbol{X})^{-1} \boldsymbol{X}^T \boldsymbol{y}$$

From lecture [1, p. 40] then we know that **correlated** predictors are a problem for two reasons

- The Variance of the estimates tends to increase

- Interpretation becomes hazardous. When x_j changes everything else changes as well.

Where we **ideally** want to have predictors that are uncorrelated

- Often we need a designed experiment to obtain this.

- That is when coefficients can be estimated and tested separately.

- Interpretation is easy. A unit change in x_j causes a change of β_j in Y, holding all other variables fixed.

1.5.2 Bias - Variance

So Bias is the difference between an expected value and the true value.

The variance is, that with high variance we might end up far from the true value and with low variance we get almost the same result every time, and how far we are from the true, depends on the bias.

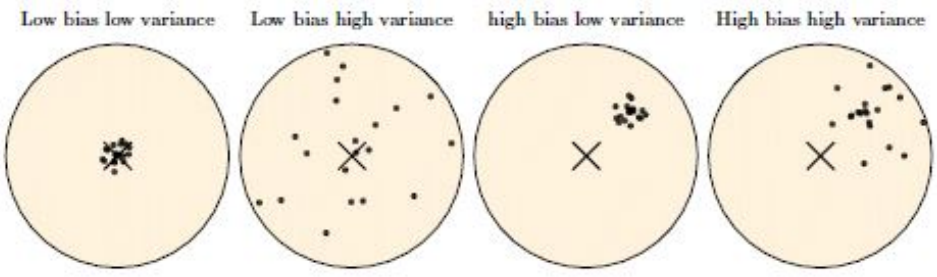


Figure 1.1: Bias and variance

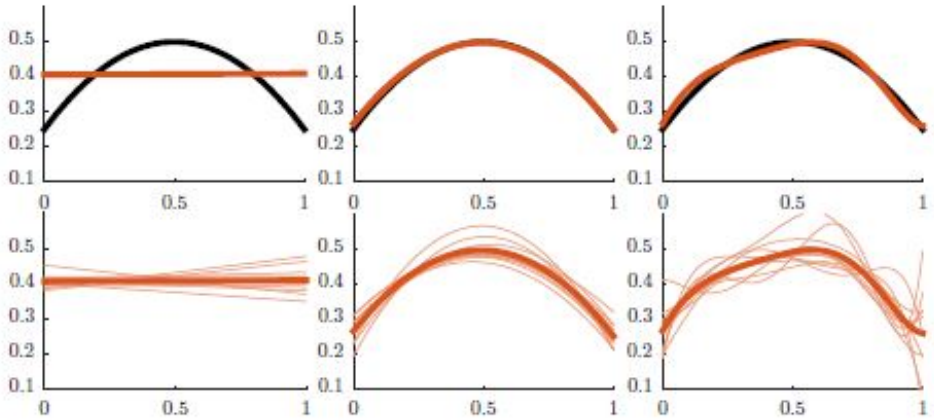


Figure 1.2: Bias-variance decomposition for the three linear regression models. In the top row is shown the bias term. Namely, how much the average values of the models trained on different random data sets (illustrated with the thick red line) differ from the true mean values of the data illustrated by the black line. In the bottom row is shown the variance term. Namely, how much each model wiggles around the mean of all models.

From lecture [1, p. 48] then There are three sources of errors contribute to the EPE (Expected Prediction Error)

$$\begin{aligned} Err(x_0) &= E[(Y-\hat{f}(x_0))^2 | X = x_0] \\ &= \sigma_\varepsilon^2 + [E\hat{f}(x_0) - f(x_0)]^2 + E[\hat{f}(x_0) - E\hat{f}(x_0)]^2 \\ &= \sigma_\varepsilon^2 + \text{Bias}^2(\hat{f}(x_0)) + \text{Var}(\hat{f}(x_0)) \\ &= \text{Irreducible Error} + \text{Bias}^2 + \text{Variance} \end{aligned} \tag{1.1}$$

Bias Variance trade off and EPE
If a model is not complex enough it produce **underfit** models, that can't learn the signal from the data.

But if a model is too comples, it produce **overfit** models that memorize the noise instead of the signal

1.5.3 Ridge Regression

Ridge Regression [13, p. 61] and from [1, p. 53] then

- We wish to lower the variance $\hat{y} = \boldsymbol{X}\beta$
- Lowering the size of β will lower the variance of $\hat{y}$

- Lower the size of β by shrinkage
  - $\hat{\beta}_{\text{ridge}} = \arg\min_{\beta} ||y - X\beta||^2 + \lambda||\beta||^2$
  - OLS (Ordinary Linear Squares) criterion plus extra term
  - λ controls the amount of shrinkage

and it is Important: Data are centered (mean of each variable is zero) and normalized (variables scaled such that standard deviation equals 1). Centering data removes the mean, causes the shrinkage to shrink towards zero and Normalizing data puts equal importance to all variables due to the penalty term $||\beta||^2$
Impose a penalty parameter from [13, p. 63]

$$\hat{\beta}_{\text{ridge}} = \arg\min_{\beta} \left\{ \sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{P} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{P} \beta_j^2 \right\}$$

Where $\lambda \geq 0$ is a complexity parameter that controls that amount of shrinkage.
Writing this to matrix form, we have

$$\hat{\beta}_{\text{ridge}} = (\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I})^{-1}\boldsymbol{X}^T\boldsymbol{y}$$

where $\boldsymbol{I}$ is $p \times p$ identity matrix.
This

- Stabilizes the inverse numerically
- Ridge regression solutions are available even when $p > n$

1.6 Linear Classifiers

- Fischer Linear Discriminant Analysis
- K-Nearest-Neighbor classification

1.6.1 Fischer Linear

Find a linear combination $Z = a^T X$ such that the between-class variance is maximized relative to the within-class variance

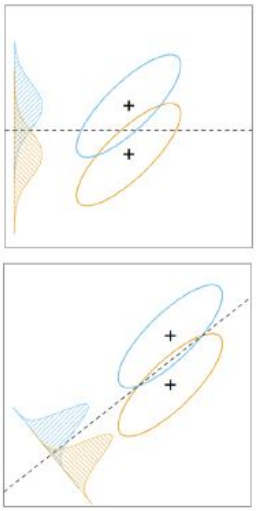


Figure 1.3: Example of the Linear Discriminant Analysis

Maximize between-class variance $a^T \Sigma_B a$ where

$$\Sigma_B = \sum_{j=1}^{K} (\mu_j - \mu)^T (\mu_j - \mu)$$

Minimize the within-class variance $a^T \Sigma_W a$ where

$$\Sigma_W = \sum_{j=1}^{K} \sum_{i=1}^{n_j} (X_{ij} - \mu_j)^T (X_{ij} - \mu_j)$$

where the group mean is u_j and total mean is μ. Then the separating line is given by

$$\max_a \frac{a^T \Sigma_B a}{a^T \Sigma_W a}$$

from lecture [1, p. 59]

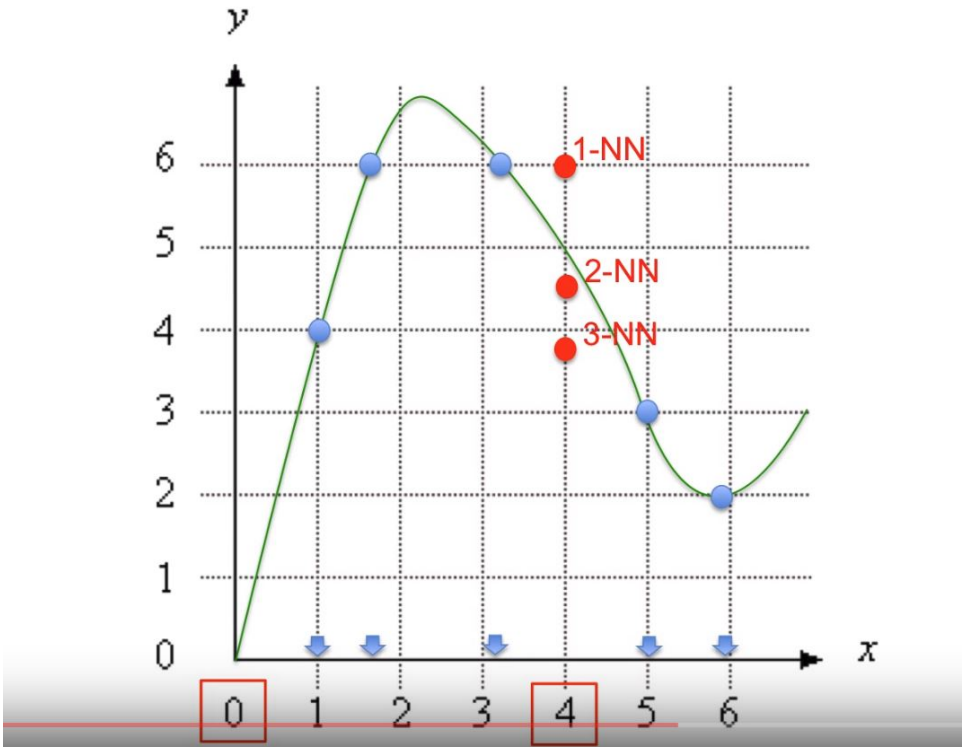
1.7 k-Nearest-Neighbor Classifiers

Remember to standardize data. If one measurement is 1000 kilometers and a second is 0.01 gram, then the first feature will dominate.
Classify observations according to the majority class of the K nearest neighbors
Small values of K gives low bias, large values will give low variance.
In k-NN classification, the output is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most

common among its k nearest neighbors (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of that single nearest neighbor.

1.8 K-Nearest-Neighbors regression

Estimate the response of an observation as the average response of the K nearest neighbors.
Define distance measure of proximity, eg Euclidean distance
It is general practice to standardize each variable to mean zero and variance 1
In k-NN regression, the output is the property value for the object. This value is the average of the values of its k nearest neighbors.



Chapter 2

Lecture 2

This lecture is about `Model Selection [CV, Bootstrap, Cp, AIC, BIC, ROC]` where chapter `ESL Chapter 7 and 9.2.5. You may safely skip sections 7.8 and 7.9` should be looked upon.

- Model Complexity
  - Bias, variance, overfitting and underfitting
- Model Selection
  - Methods for selecting an appropriate model from an ensemble of candidates.
    - Training, test and validation set
    - Cross-validation
    - Methods based on information criteria
- Model Assessment
  - Bootstrap
  - Sensitivity, specificity and ROC curves

2.1 Chapter 7

Bias Variance trade-off. The AIC and BIC and Cross-validation. Different Bootstrap methods

2.2 Chapter 9

Tree-Based method Spam Example.

2.3 Model Complexity

- Overfitting and underfitting
- Regularization
- Bias and variance

2.4 Overfitting and underfitting

From lecture [2, p. 7] then we know that we prefer a simple model and a model that work e.g. have low EPE.
So if a model is too simple, then Our data set will not be accurately described. Model assumptions are wrong.

And if it is too complex then The model becomes too flexible. We fit noise in data. We need lots of data to support such a model.
So we might wish to do regularization and there are two approaches

- Bayesian prior to β
- Penalty to models with large β

From lecture [2, p. 18] then we can use Ridge Regression. We can use a flexible model and avoid overfitting via regularization.
With the simple solution

$$\beta_{\text{ridge}} = (\boldsymbol{X}^T\boldsymbol{X} + \lambda \boldsymbol{I})^{-1}\boldsymbol{X}^T\boldsymbol{y}$$

where regularization parameter λ.

- $\lambda = 0$ gives $\beta_{\text{ridge}} = \beta_{\text{OLS}}$
  - No bias
  - High Variance
- $\lambda \to 0$ gives $\beta_{\text{ridge}} \to 0$
  - High bias
  - No Variance

2.5 Regularization

[2, p. 17]
From wiki:[1]
Early stopping can be viewed as regularization in time. Intuitively, a training procedure like gradient descent will tend to learn more and more complex functions as the number of iterations increases. By regularizing on time, the complexity of the model can be controlled, improving generalization.
In practice, early stopping is implemented by training on a training set and measuring accuracy on a statistically independent validation set. The model is trained until performance on the validation set no longer improves. The model is then tested on a testing set.

2.6 Bias Variance trade-off

Model selection: estimating the performance of different models in order to choose the best one.
Model assessment: having chosen a final model, estimating its prediction error (generalization error) on new data.
If we are in a data-rich situation, the best approach for both problems is to randomly divide the dataset into three parts: a training set, a validation set, and a test set. The training set is used to fit the models; the validation set is used to estimate prediction error for model selection; the test set is used for assessment of the generalization error of the final chosen model. Ideally, the test set should be kept in a "vault," and be brought out only at the end of the data analysis. Suppose instead that we use the test-set repeatedly, choosing the model with smallest test-set error. Then the test set error of the final chosen model will underestimate the true test error, sometimes substantially [13, p. 222]

If a model is not complex enough it produce **underfit** models, that can't learn the signal from the data.

But if a model is too comples, it produce **overfit** models that memorize the noise instead of the signal

2.6.1 Bias Variance Decomposition

Assume that $Y = f(X) + \varepsilon$ and $E(\varepsilon) = 0$ and $Var(\varepsilon) = \sigma_\varepsilon^2$
Then we have

[1] regularization mathematics

$$\begin{aligned} Err(x_0) &= E[(Y - \hat{f}(x_0))^2 | X = x_0] \\ &= \sigma_\varepsilon^2 + [E\hat{f}(x_0) - f(x_0)]^2 + E[\hat{f}(x_0) - E\hat{f}(x_0)]^2 \\ &= \sigma_\varepsilon^2 + \text{Bias}^2(\hat{f}(x_0)) + \text{Var}(\hat{f}(x_0)) \\ &= \text{Irreducible Error} + \text{Bias}^2 + \text{Variance} \end{aligned} \tag{2.1}$$

It does not depend at all upon our choice of model but simply represents the intrinsic difficulty of the problem. We cannot make this term any larger or smaller by selecting one model over another, unless $\sigma_\varepsilon^2 = 0$

The second term $[E\hat{f}(x_0) - f(x_0)]^2$, squared bias, tells us how much the average values of models trained on different training datasets differ compared to the true mean of the data $\hat{f}(x_0)$

The last term $E[\hat{f}(x_0) - E\hat{f}(x_0)]^2$ is the variance term. It tells us how much the model wiggles when trained on different sets of training data. That is when you train models on N different sets of training data and the models are nearly the same, this term is small.

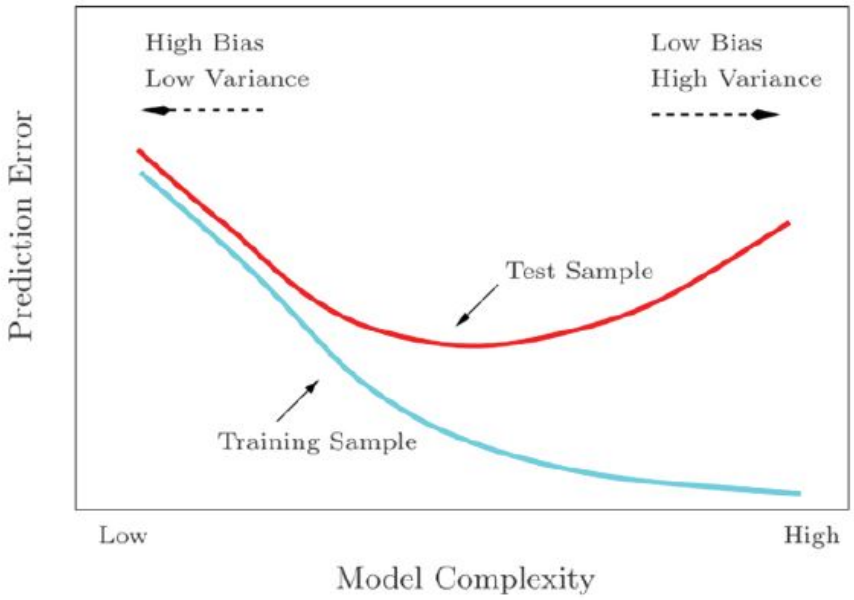


Figure 2.1: Bias and variance the model complexity

2.7 Cross-Validation

The more complex the model, the better we will fit the training data. Often we overfit to the training data.

Overfit models can perform poorly on test data - high variance.

Underfit models can perform poorly on test data - high bias.

As discussed, we perform **Model Selection** and **Model assessment**. For both of these purposes, the best approach is to evaluate the procedure on an independent test set. [2, p.]
Often there is insufficient data to create a separate validation and test set. In this instance use cross validation instead.[2, p. 30]

- The primary method for estimating a tuning parameter, e.g. λ
- No underlying assumptions
- Is simple, intuitive and easy to implement

Split randomized data in K equal parts

For each $k = 1, 2, ..., K$ fit the model for parameter λ to the other $K-1$ parts, give $\hat{\beta}^{\neg k}(\lambda)$ and compute its error in predicting the kth part:

$$E_k(\lambda) = \sum_{i \in k\text{th part}} (y_i - X_i \hat{\beta}^{\neg k}(\lambda))^2$$

This gives the cross-validation error

$$CV(\lambda) = \frac{1}{K} \sum_{k=1}^{K} E_k(\lambda)$$

Do this for many values for λ and choose the value of λ that makes $CV(\lambda)$ smallest.

From [13, p. 243], then to summarize, if the learning curve has a considerable slope at the given training set size, five- or tenfold cross-validation will overestimate the true prediction error. Whether this bias is a drawback in practice depends on the objective. On the other hand, leave-one-out cross-validation has low bias but can have high variance. Overall, five- or tenfold cross-validation are recommended as a good compromise:

From lecture [2, p. 32] we use the **one standard error rule**, where we choose the smallest model whose error is no more than one standard error above the error of the best model. This compensates that the CV procedure chooses somewhat too complex models.
Consider a classification problem with a large number of predictors, as may arise, for example, in genomic or proteomic applications. A typical strategy for analysis might be as follows:

- Screen the predictors: find a subset of "good" predictors that show fairly strong (univariate) correlation with the class labels
- Using just this subset of predictors, build a multivariate classifier.
- Use cross-validation to estimate the unknown tuning parameters and to estimate the prediction error of the final model.

2.7.1 Considerations

Observations are supposed to be independent. Otherwise information will leak from validation set to training set and we will overfit.

- Permute data before splitting in folds - the data set might be sorted in some way.
- Normalize training data separately for each fold.
- Impute missing values separately for each training set.
- If observations are sampled in groups, let each group go into the same fold.
- Be extremely careful with data sampled from dynamic systems (time-dependent data).
- Be careful to perform ALL pre-processing within the CV folds.

[2, p. 34][13, p. 241-249]

2.7.2 Leave one out cross-validation

Be careful using leave-one-out CV (K = N), the folds are too similar. Use K = 5 or 10 as a good compromise between bias and variance. [2, p. 34]

2.8 Information criteria

- Optimism and training error
- Information Criteria

2.8.1 Optimism of the training error

From lecture [2, p. 37] and [13, p. 228] then the training error for OLS regression is

$$\frac{1}{N}\sum_{i=1}^{N}(y_i - x_i\hat{\beta})^2$$

Think that we for each x_i can get a new measurement y_i^0. What would the error for these new y_i^0 be? The in-sample-error

$$\frac{1}{N}\sum_{i=1}^{N}(y_i^0 - x_i\hat{\beta})^2$$

Selecting the model with the smallest in-sample-error would be an effective model selection tool.
The difference between the in-sample-error and the training-error is called optimism. It is possible to show that quite generally

$$\frac{2}{N}\sum_{i=1}^{N}Cov(\hat{y}_i, y_i)$$

The more we overfit data the greater $Cov(\hat{y}_i, y_i)$ will be and thereby increasing the optimism.

In the linear case with d variables and squared error loss we have expected optimism

$$\text{expected optimism} = d\sigma_e^2$$

Therefore we have that

$$\text{expected in-sample-error} = \text{expected training-error} + 2\frac{d}{N}\sigma_e^2$$

the number d of inputs or basis functions we use, but decreases as the training sample size increases from [13, p. 230].
An obvious way to estimate prediction error is to estimate the optimism and then add it to the training error

2.8.2 CP-statistic

Given a squared error loss and a model with d variables we can calculate the so-called C_p-statistic

$$C_p = \overline{\text{err}} + 2\frac{d}{N}\sigma_e^2$$

where

$$\overline{\text{err}} = \frac{1}{N}\sum_{i=1}^{N} L(y_i, \hat{f}(x_i))$$

Using this metric we can select the best model by choosing the model that minimizes C_p.

Expected training error, use the actual training error

$$\frac{1}{N}\sum_{i=1}^{N}(y_i - x_i\hat{\beta})^2$$

Noise variance, use the mean squared error (MSE) of low bias model (OLS or KNN) as the estimate $\hat{\sigma}_e^2$ from lecture [2, p. 39]

2.8.3 AIC

To use AIC for model selection, we simply choose the model giving smallest AIC over the set of models considered. [13, p. 231] AIC uses a log-likelihood loss function (logL) to estimate the error term (maximized log-likelihood)[2, p. 40]

$$AIC = -\frac{2}{N}\log \text{L} + 2\frac{d}{N}$$

For the Gaussian case with a tuning parameter λ we define

$$AIC(\lambda) = \overline{err}(\lambda) + 2\frac{d(\lambda)}{N}\hat{\sigma}_e^2$$

For the Gaussian model C_p and AIC are identical. Where $d(\lambda)$ is the effective number of parameters in the model tuned with λ

I Often leave-one-out cross-validation and AIC tend to choose models which are too complex. [2, p. 45]

2.8.4 Effective number of parameters

From [13, p. 232], then the concept of "number of parameters" can be generalized, especially to models where regularization is used in the fitting.
For a linear fitting method then from lecture [2, p. 41]

$$\hat{Y} = SY$$

then we can calculate the effective number of parameters as

$$df(S) = trace(S)$$

i.e. the sum of the diagonal elements of S.
Both OLS and ridge regression are linear fitting methods.

2.8.5 BIC

The Bayes Information Criterion, BIC, is like AIC based on a log likelihood loss function. It is motivated from a Bayesian approach selecting the model with the largest posterior probability. [2, p. 42]

$$BIC = -2\log \mathrm{L} + \log(N)d$$

Under the Gaussian model and squared error loss

$$BIC(\lambda) = \frac{N}{\hat{\sigma}_e^2}\left(\overline{err}(\lambda) + \log(N)\frac{d(\lambda)}{N}\hat{\sigma}_e^2\right)$$

Again, select the model that has the lowest value.
BIC tends to penalize complex models more heavily, giving preference to simpler models in selection. As with AIC, σ_ε^2 is typically estimated by the mean square error of a low bias model. [13, p. 233]
For AIC and BIC, then for a given set of models:

- As $n \to \infty$ the probability that BIC selects the correct model approaches one, whereas AIC tends to select a model which is too complex
- For small sample sizes BIC tends to select a model which is too simple

So, BIC is good for large models, while AIC is good for small models.

2.9 Bootstrap Methods

For the bootstrap method, look at lecture [2, p. 46] and book [13, p. 249]
Bootstrap is a general method for assessing statistical accuracy, furthermore the statistical bootstrap is also an absurdly simple and effective way of solving apparently hard problems.
Note:
If you don't have enough data to train your algorithm you can increase the size of your training set by (uniformly) randomly selecting items and duplicating them (with replacement).
Take a sample of the time of day that you wake up on Saturdays. Some Friday nights you have a few too many drinks, so you wake up early (but go back to bed). Other days you wake up at a normal time. Other days you sleep in.

Here are the results: [3.1, 4.8, 6.3, 6.4, 6.6, 7.3, 7.5, 7.7, 7.9, 10.1]

What is the mean time that you wake up? Well it's 6.8 (o'clock, or 6:48). A touch early for me.

How good a prediction is this of when you'll wake up next Saturday? Can you quantify how wrong you are likely to be?
It's a pretty small sample, and we're not sure of the distribution of the underlying process, so it might not be a good idea to use standard parametric statistical techniques.
Why don't we take a random sample of our sample, and calculate the mean and repeat this? This will give us an estimate of how bad our estimate is.
I did this several times, and the mean was between 5.98 and 7.8

Given a training set $Z = (z_1, z_2, ..., z_N)$ where $z_i = (x_i, y_i)$.
In figure 2.2, $S(\boldsymbol{Z})$ is any quantity computed from the data $\boldsymbol{Z}$, for example, the prediction at some input point. From the bootstrap sampling we can estimate any aspect of the distribution of $S(\boldsymbol{Z})$, for example, its variance,

$$\hat{\mathrm{Var}}[S(\boldsymbol{Z})] = \frac{1}{B-1}\sum_{b=1}^{B}(S(\boldsymbol{Z}^{*b}) - \overline{S}^{*})^2$$

where

$$\overline{S}^{*} = \sum_{b}\frac{S(\boldsymbol{Z}^{*b})}{B}$$

The variance of a parameter, S, is estimated by taking the variance over B bootstrap replications (the value of the parameter for the specific bootstrap samples).

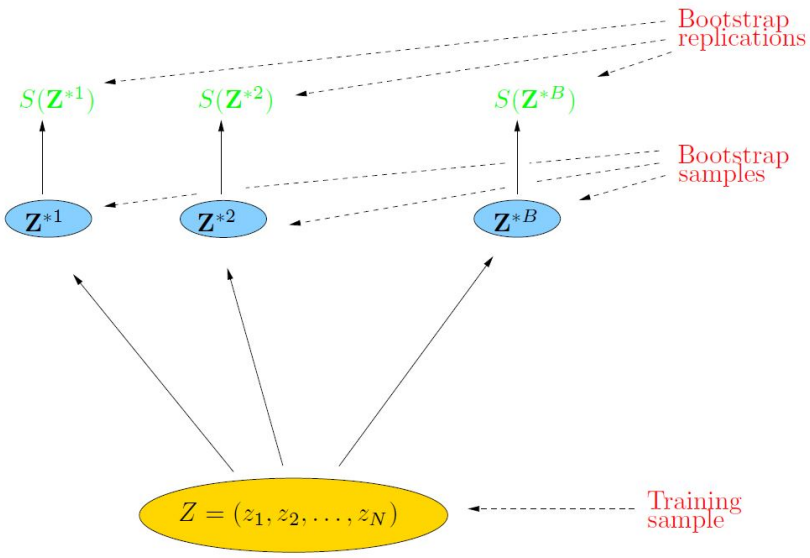


Figure 2.2: Schematic of the bootstrap process. We wish to assess the statistical accuracy of a quantity $S(\boldsymbol{Z})$ computed from our dataset. B training sets $Z^{*b}, b = 1, ..., B$ each of size N are drawn with replacement from the original dataset. The quantity of interest $S(\boldsymbol{Z})$ is computed from each bootstrap training set, and the values $S(\boldsymbol{Z}^{*1}), ..., S(\boldsymbol{Z}^{*B})$ are used to assess the statistical accuracy of $S(\boldsymbol{Z})$

2.10 Classifier performance

2.10.1 Classifiers

2.10.2 LDA

Find a linear combination $Z = a^T X$ such that the between-class variance is maximized relative to the within-class variance

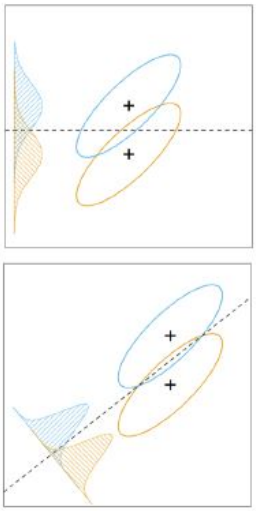


Figure 2.3: Example of the Linear Discriminant Analysis

Maximize between-class variance $a^T \Sigma_B a$ where

$$\Sigma_B = \sum_{j=1}^{K} (\mu_j - \mu)^T (\mu_j - \mu)$$

Minimize the within-class variance $a^T \Sigma_W a$ where

$$\Sigma_W = \sum_{j=1}^{K} \sum_{i=1}^{n_j} (X_{ij} - \mu_j)^T (X_{ij} - \mu_j)$$

where the group mean is u_j and total mean is μ. Then the separating line is given by

$$\max_a \frac{a^T \Sigma_B a}{a^T \Sigma_W a}$$

from lecture [1, p. 59]
[13, p. 106-]

2.10.3 Confusion Matrix

The confusion matrix have TP, FP, FN and TN, where

- TP = True Positive which means hits
- FP = False Positive which means false alarm
- FN = False Negative which means misses

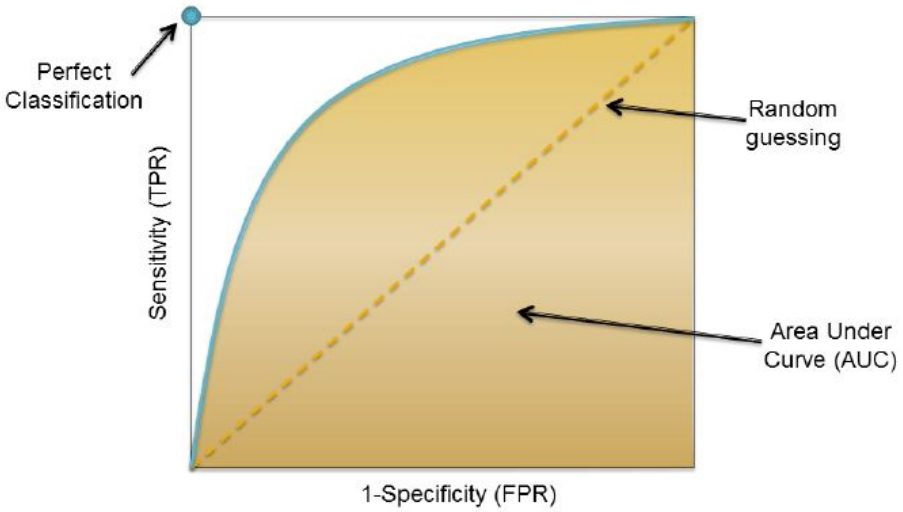
- TN = True Negative which means correct rejection

Then we have the

- Sensitivity (TPR) which is Your method's ability to identify positives
- Specificity (SPC) Your method's ability to identify negatives
- Positive Predictive Value (PPV) Proportion of positives which are correctly classified
- Negative Predictive Value (NPV) Proportion of negatives which are correctly classified
- False Discovery Rate (FDR) Proportion of positives which are incorrectly classified

2.10.4 ROC Curve

Sensitivity and specificity changes as the cut-off value changes.



2.10.5 AUC

Can we explicitly maximize AUC when creating models?
AUC is discrete, hard to optimize model parameters directly w.r.t. AUC.
Most continuous methods such as maximum likelihood corresponds well to AUC.

Regularization Select regularization parameter based on AUC.
But use AUC with caution.

Assume you are segmentting a small part, 1%, of an image (e.g. a tumor in a mammography)
You have a large number of true negatives (TN) and the penalty for a segmentation twice as large as it should be?

Specificity = TN/(TN+FP) = 1mio/(1mio+10000) which is Specificity: 100% => 99%.

Result: AUC is very high, even though the segmentation is poor. Interpret with caution when prevalence is off.
Summary

- Over- and underfitting
- Model selection
  - Traning/validation/test set
  - Cross-validation
    - One standard error rule
  - Information criterion
    - C_p-statistics
    - AIC
    - BIC
- Model assessment
  - Bootstrap
  - Sensitivity vs specificity and ROC curves
  - Confusion matrices

Chapter 3

Lecture 3

This lecture is about `Sparse Regression [Lasso, Elastic Net] Case 1 presentation` where chapter `ESL Chapter 3.3, 3.4 and 18` should be looked upon.
From [3]

- Curse of dimensionality
- Regularization
- Multiple hypothesis testing

3.1 Chapter 3.3

Subset selection, where we look at the best-subset selection. We are also talking about Forward and backward stepwise selection and the forward stagewise regression. In the end it will be the Prostate Cancer Data Example

3.2 Chapter 3.4

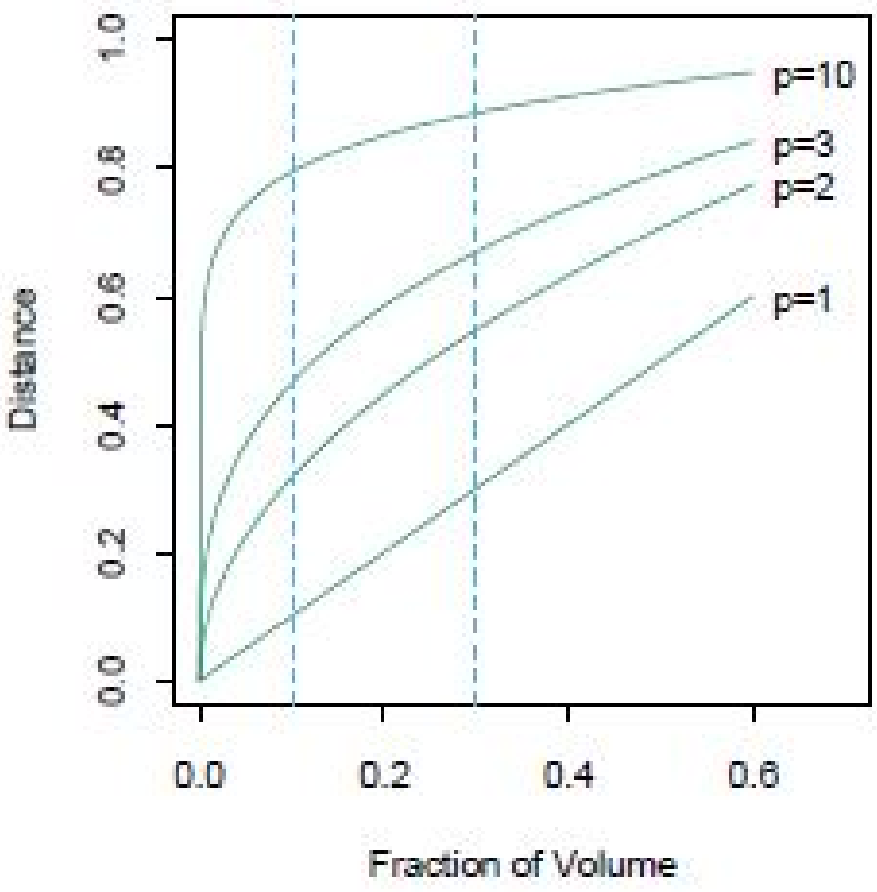
This part will be about shrinkage methods, with the Ridge Regression and The Lasso. Discussion: Subset Selection, Ridge Regression and the Lasso and in the end of the section looking at Least Angle Regression

3.3 Chapter 18

This chapter is about high dimensional problems, where $p \gg N$

3.4 Curse of dimensionality

Dog/cat example. If we have 10 dataset, then the space between these dataset will be bigger and bigger, with more dimenions.



As seen in the picture, then with 10 features the side length has to be 80% to cover 10% of data.
For data fitted to a unit sphere the median distance from the center of the sphere to the closest point is

$$d(p,n) = \left(1 - \left(\frac{1}{2}\right)^{1/n}\right)^{1/p}$$

as seen in lecture [3, p. 11]

But its not all bad. There are some blessings of dimensionality

- Several features will be correlated and we can average over them
- Underlying distribution will be finite, informative data will lay on a low-dimensional manifold
- Underlying structure in data (samples from continuous processes, images etc) will give an approximate finite dimensionality.

3.5 Dimension reduction

How to decrease the dimension and identify the most important variables, and get rid of the redundant or irrelevant variables.

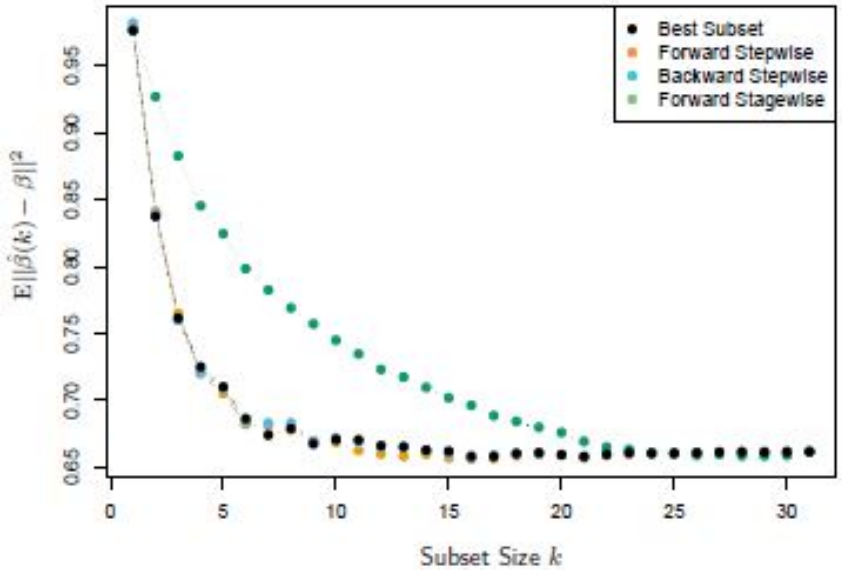
- Combinatoric search, forward and backward selection
- Regularization of parameters
- Projection to lower dimensions - latent variables
- Clustering of features
- Structuring parameter estimates

3.5.1 Forward and backward selection

If we do combinatoric search then: Try all possible combinations of features and select the optimal one.
The **Pros:** You will find the best combination., but the **Cons:** Number of combinations to test may be extremely large.
Forward-stepwise selection is a greedy algorithm, producing a nested sequence of models. In this sense it might seem sub-optimal compared to best-subset selection. However, there are several reasons why it might be preferred:

- Computational; for large p we cannot compute the best subset sequence, but we can always compute the forward stepwise sequence (even when $p \gg N$).
- Statistical; a price is paid in variance for selecting the best subset of each size; forward stepwise is a more constrained search, and will have lower variance, but perhaps more bias.

from [13, p. 58]
Backward-stepwise selection starts with the full model, and sequentially deletes the predictor that has the least impact on the fit. The candidate for dropping is the variable with the smallest Z-score. Backward selection can only be used when N > p, while forward stepwise can always be used.



3.6 Regularization

Instead of controlling model complexity by setting a subset of coefficients to zero we can shrink all the coefficients some way towards zero.
Three established standard techniques

- Ridge regression uses quadratic shrinkage, L_2-norm
- Lasso regression uses absolute-value shrinkage, L_1-norm
- Elastic net which is a hybrid method

3.6.1 Ridge Regression

Ridge regression shrinks the regression coefficients by imposing a penalty on their size. The ridge coefficients minimize a penalized residual sum of squares, it solves[13, p. 61]

$$\min_{\beta}(Y - X\beta)^T(Y - X\beta) + \lambda\beta^T\beta$$

Increased λ will make the estimated β's smaller but not exactly zero and We typically do not penalize the intercept β_0 [3, p. 23]
Optimization of a weighted sum

$$\arg\min_{\beta} ||Y - X\beta||_2^2 + \lambda||\beta||_2^2$$

3.6.2 The Lasso

The lasso is a shrinkage method like ridge, with subtle but important differences. [13, p. 68]
The Lasso regression solves

$$\min_{\beta}(Y - X\beta)^T(Y - X\beta) + \lambda|\beta|$$

Notice that the L_2-penalty is replaced by a L_1-penalty.

This makes the solution nonlinear in Y and a quadratic programming algorithm is used to compute it.
For large enough λ some of the β will be set to exactly zero.
The effective numbers of parameters, equals the number of coefficients different from zero.
From lecture [3, p. 27] then Lasso regularization will gear parameters towards zero.

$$\arg\min_{\beta} ||Y - X\beta||_2^2 + \lambda||\beta||_1$$

The only difference from Ridge regression is that the regularization term is in absolute value. But this difference has a huge impact on the trade-off we've discussed before. Lasso method overcomes the disadvantage of Ridge regression by not only punishing high values of the coefficients β but actually setting them to zero if they are not relevant. Therefore, you might end up with fewer features included in the model than you started with, which is a huge advantage

3.6.3 Elastic net

By combining the L_1 and the L_2-norm we obtain sparsity and shrinkage

$$\min_{\beta} \frac{1}{2n}||Y - X\beta||_2^2 + \lambda\left(\frac{1}{2}(1-\alpha)||\beta||_2^2 + \alpha||\beta||_2\right)$$

From lecture [3, p.39] the Advantage: Combines the shrinkage of ridge and parameter selection of the lasso to obtain a robust sparse estimate.
The reasons to use elastic net is seen on lecture [3, p. 49]

- Get rid of irrelevant variables/select important variables (lasso)
- When p > n, the number of non-zero coefficients can exceed n unlike the lasso.
- Works well when covariates are highly correlated; allows us to "average" highly correlated features and obtain more robust estimates (grouping features).

The drawback is the issue of tuning two parameters. Use a grid search, a fine grid in λ and fewer values for α

3.7 Multiple testing

If we test one hypothesis at an α-level of significance there is a chance α of falsely rejecting the hypothesis. [3, p.52-53]
This is no longer the case if we do many tests!
The family-wise error rate (FWER) is the probability of at least one false rejection, and is a commonly used overall measure of error. [13, p. 686]
For M independent test at significance level α

$$FWER = 1 - (1 - \alpha)^M$$

- 20 experiments conducted at a 5% significance level
- Assume that the effect of different colors are independent, then $FWER = 1 - (1 - 0.05)^{20} \approx 0.64$
- There is 64% probability of at least one false rejection

From lecture [3, p.60] we should use the Bonferroni correction we rescale α with a number of tests.
So we reject a hypothesis if its p-value is below α/M
From lecture [3, p. 61] and [13, p. 687] we can look at the False Discovery Rate (FDR). We can have more significant findings if we allow for a few mistakes. The false discovery rate is a technique to control the number of falsely detected significant features, where the FDR is

$$FDR = E\left(\frac{FP}{FP + TP}\right)$$

where

- FP = False positives (false discoveries)
- TP = True positives (true discoveries)

If we accept hypotheses where $FDR < q$ then we will expect that among our findings there will be q mistakes.

- Gain: We control false positives - added power.
- Cost: Increased number of false negatives.

We prefer to get a few false discoveries (percentage-wise) but gain more information, than ensuring no false discoveries and loosing some information.
Summary

- Introduction
  - The Curse of dimensionality
  - The blessings of dimensionality
  - Dimension reduction
- Regularization
  - Ridge, Lasso and Elastic Net
  - Shrinkage and sparsity
  - Best practices
- Multiple hypothesis testing
  - Why is it a problem
  - Bonferroni correction
  - False discover rate and Benjamini-Hochberg algorithm

This page is intentionally left blank.

Chapter 4

Lecture 4

This lecture is about `Linear Classifiers and Basis Expansion [LDA, QDA, Logistic regression, Splines]` where chapter `ESL Chapter 4.3, 4.4, 5.1 and 5.2` should be looked upon.
[4]

4.1 Chapter 4.3

Here we will talk about the LDA

4.2 Chapter 4.4

This chapter will be about Logistic Regression

4.3 Chapter 5.2

And finally this chapter will be about the piecewise polynomials and splines.

4.4 Linear Discirminant Analysis

$f_k(x)$ is the class-conditional density of X in class $G = k$ and let π_k be the prior probability of class k with $\sum_{k=1}^{K} \pi_k = 1$. Then we have the probability of class k, given observation x as

$$P(G = k|X = x) = \frac{f_k(x)\pi_k}{\sum_{\ell=1}^{K} f_\ell(x)\pi_\ell}$$

where f_ℓ is distribution for class ℓ and π_ℓ is a priori probability for class ℓ
We need a stochastic model for data to calculate probabilities, and we assume that the data come from a different Gaussian distributions.
A straight line will be our decision boundary.
We look at the log-odds-ratio for the two classes k and ℓ

$$\log \frac{P(G=k|X=x)}{P(G=\ell|X=x)} = \log \frac{f_k(x)}{f_\ell(x)} + \log \frac{\pi_k}{\pi_\ell}$$

where $f(x) = \frac{1}{(2\pi)^{p/2}|\Sigma_k|^{1/2}} \exp(-\frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k))$ which is the multivariate Gaussian with a common covariance matrix $\Sigma_k = \Sigma \forall k$
This linear log-odds function implies that the decision boundary between classes k and ℓ is linear in x in p dimensions a hyperplane $a + x^T b = 0$
The decision boundary becomes

$$\log \frac{\pi_k}{\pi_\ell} - \frac{1}{2}(\mu_k + \mu_l)^T\Sigma^{-1}(\mu_k - \mu_l) + x^T\Sigma^{-1}(\mu_k - \mu_l)$$

From this equation we see that the linear discriminant functions

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

are an equivalent description of the decision rule with $G(x) = \arg\max_k \delta_k(x)$
We do not know the parameters of the Gaussian distributions, and will need to estimate them using our training data.

$$\hat{\pi}_k = N_k/N$$

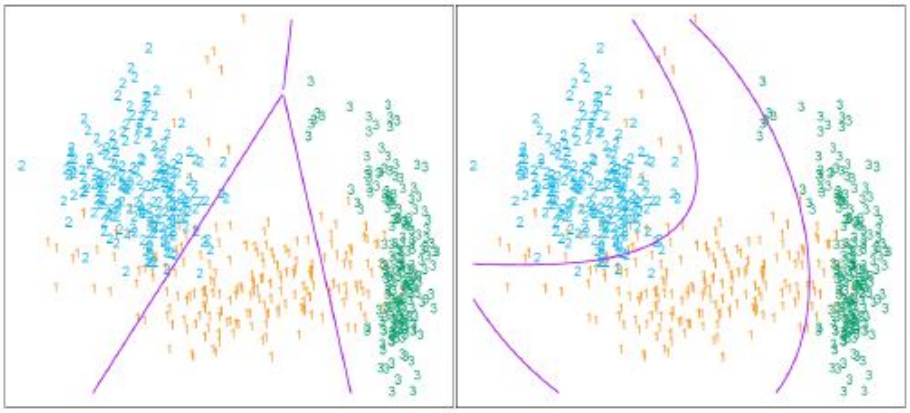
where N_k is the number of class-k observations

$$\hat{\mu_k} = \sum_{g_i=k} x_i/N_k$$

and

$$\hat{\Sigma} = \sum_{k=1}^{K} \sum_{g_i=k} (x_i - \hat{\mu_k})(x_i - \hat{\mu_k})^T/(N-K)$$

As seen in [13, p. 110]
If we have more than two classes, then one decision line for each of classes and one discriminant function for each class.
So Linear discriminant analysis assumes that the covariance structures are equal.
When we drop this restriction we get quadratic discriminant analysis, QDA, and the decision boundaries becomes non-linear.



4.5 Regularized discriminant analysis

It takes a lot of observations to estimate a large covariance matrix with precision. Three increasingly harsh regularizations are available
From lecture [4, p. 15] We make a compromise between LDA and QDA which allows to shrink the seperate covariance of QDA toward a common covariance as in LDA.

$$\hat{\Sigma}_k(\alpha) = \alpha\hat{\Sigma}_k + (1-\alpha)\hat{\Sigma}$$

where $\hat{\sigma}$ is the pooled covariance matrix as used in LDA and $\alpha \in [0,1]$
Then shrink the covariance toward its diagonal

$$\hat{\Sigma}_k(\gamma) = \gamma\hat{\Sigma} + (1-\gamma)\text{diag}(\hat{\Sigma})$$

Similar modifications allows $\hat{\sigma}$ itself to be shrunk toward the scalar covariance structure

$$\hat{\Sigma}_k(\gamma) = \gamma\hat{\Sigma} + (1-\gamma)\hat{\Sigma}^2\boldsymbol{I}$$

In these situations the features are high-dimensional and correlated, and the LDA coefficients can be regularized to be smooth or sparse in the original domain of the signal.

4.6 Reduced rank discriminant analysis

So far we have discussed LDA as a restricted Gaussian classifier. Part of its popularity is due to an additional restriction that allows us to view informative low-dimensional projections of the data.

Hence, this technique is very useful for illustrating class separation.
If K = 3, for instance, this could allow us to view the data in a two-dimensional plot, color-coding the classes. In doing so we would not have relinquished any of the information needed for LDA classification. More on [13, p. 114]

4.7 Logistic Regression

The logistic regression model arises from the desire to model the posterior probabilities of the K classes via linear functions in x, while at the same time ensuring that they sum to one and remain in [0, 1].
From lecture [4, p. 21] we want to derive expressions for the two class problem, i.e. $P(G = red|X = x)$ and $P(G = blue|X = x)$ when $\log \frac{P_r}{P_b} = \beta_0 + x\beta$
Logistic regression models are usually fit by maximum likelihood, using the conditional likelihood of G given X. So the likelihood for N observations is

$$L(\beta_0, \beta) = \prod_{i=1}^{N} P(G = g_{x_i}|X = x_i)$$

as seen in lecture [4, p. 22]. Where the log-likelihood is

$$\ell(\theta) = \sum_{i=1}^{N} \log P(G = k|X = x_i; \theta)$$

We wish to maximize the log-likelihood, wrt to β where $\beta = \{\beta_0, \beta\}$

$$\ell(\beta_0, \beta) = \sum_{i=1}^{N} (\beta_0 + x_i\beta) - \log(1 + \exp(\beta_0 + x_i\beta))$$

This is known as maximum likelihood and the result is called logistic regression.
To maximize the log-likelihood, we set its derivatives to zero

$$\frac{\partial \ell(\beta)}{\partial \beta} = 0$$

and to solve this equation we use the Newton-Raphson algorithm, which requires the second derivative or Hessian Matrix [13, p. 121]
From lecture [4, p. 29] then the reason to use logistic regression is

- Statistics
  - Identify variables important for separating the classes
    - Biostatistics and epidemiology

- Classification
  - Predict class belonging of new observations
    - For example spam/email or diseased/healthy
- Risk prediction
  - Estimate probability (risk) for each class
    - Fraud detection in insurance claims

4.7.1 Interpreting the coefficients

The example is we have a model of lung cancer (yes/no) as a function of smoking (number of cigarettes per day).
If $\beta = 0.02$, then a unit increase in smoking (one extra cigarette) means an increase in lung cancer risk of $\exp(0.02) \approx 1.02 = 2\%$ from lecture [4, p. 30]
More can be seen on [13, p. 124]

4.8 L1 Regularized Logistic Regression

If we have few observations, low n, and high dimension, high p data is a problem also for logistic regression.
One solution is an elastic net regularization of the likelihood.

$$\max_{\beta_0,\beta} \left\{ \sum_{i=1}^{N} \left[y_i(\beta_0 + \beta^T x_i) - \log(1 + \exp(\beta_0 + \beta^T x_i)) \right] - \lambda \sum_{j=1}^{P} |\beta_j| \right\}$$

As with the lasso, we typically do not penalize the intercept term, and standardize the predictors for the penalty to be meaningful.
The equation is concave and a solution can be found using nonlinear programming methods.

- Logistic regression is more robust than LDA
  - It relies on fewer assumption
  - When is this a bad thing when compared to LDA?
- Logistic regression handles categorical variables better than LDA
- Observations far away from the boundary are down-weighted
  - You will have a look at how this works during the exercises

- Breaks down when classes are perfectly separable
- Easy to interpret and explain
- Surprisingly often hard to beat
- Can be combined with regularization of parameters (n < p)
- Can be generalized to multi-class problems

4.9 Basic Expansions and Regularization

- We are not limited to use our data as they are
- Linear models
  - Easy to interpret
  - First order Taylor expansion of non-linearities
  - Might be ok even for non-linear data if we have a few observations
- Non-linear problem - transform data and use linear model
  - $h_m(X) = X_j^2$ and $h_m(X) = X_j X_k$
  - $h_m(X) = \log(X_j)$ or $h_m(X) = \sqrt{X_j}$
  - $h_m(X) = \frac{X - m_X}{s_X}$ (always used when using regularization
  - $h_m(X_{(i)} = i$ sort data $X_{(1)} \leq X_{(2)} \leq ...$ and use the rank
  - Either replacing X with $h_m(X)$ or expanding $\{X, h_m(X)\}$

In this chapter and the next we discuss popular methods for moving beyond linearity. The core idea in this chapter is to augment/replace the vector of inputs X with additional variables, which are transformations of X, and then use linear models in this new space of derived input features.
We denote by $h_m(X) : \mathbb{R}^p \to \mathbb{R}$ the mth transformation of $X, m = 1, ..., M$, we then have the model

$$f(X) = y = X\beta = \sum_{m=1}^{p} \beta_i x_i \to \sum_{m=1}^{M} \beta_i' h_i(X)$$

Sometimes the problem at hand will call for particular basis functions $h_m m$, such as logarithms or power functions.More often, however, we use the basis expansions as a device to achieve more flexible representations for $f(X)$.

From the lecture [4, p. 47] then we wish to have cubic splines.
Then we wish to have $h_0(x) = 1$, $h_1(x) = x$, $h_2(x) = x^2$, $h_3(x) = x^3$, $h_4(x) = (x-5)^3_+$ and so on.
We often prefer smoother functions, and these can be achieved by increasing the order of the local polynomial. Cubic splines have continuous first and second derivatives at the knots, which means smooth functions. We should be able to obtain non-linear functions with a linear model.

This page is intentionally left blank.

Chapter 5

Lecture 5

This lecture is about `Support Vector Machines and Convex Optimization` where chapter `ESL Chapter 4.5, 12.1, 12.2 and 12.3.1` should be looked upon.

- Convex optimization using Lagrange multipliers
- Optimal Separating Hyperplanes
- Support Vector Machines
- The Kernel trick

5.1 Constrained Optimization

We learn to solve

$$max_x f(x) \tag{5.1}$$
$$g(x) = 0 \tag{5.2}$$
$$h(x) \geq 0 \tag{5.3}$$

using Lagrange multipliers

5.1.1 Unconstrained optimization

Assume that f is nice, i.e. continuously differentiable
Then a local maxima, $x*$ fulfills, that gradient is zero $\nabla_x f(x*) = 0$ and the Hessian is negative definite, $v^T \nabla^2_{xx} f(x*)v < 0, \forall v \in \mathbb{R}^n$
where

$$\nabla_x f(x) = \begin{pmatrix} \frac{\partial f(x)}{\partial x_1} \\ \vdots \\ \frac{\partial f(x)}{\partial x_n} \end{pmatrix}$$

and

$$\nabla^2_{xx} f(x) = \begin{pmatrix} \frac{\partial^2 f(x)}{\partial x_1^2} & \cdots & \frac{\partial^2 f(x)}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f(x)}{\partial x^n \partial x_1} & \cdots & \frac{\partial^2 f(x)}{\partial x_n^2} \end{pmatrix}$$

A negative second derivative guarantees a local maximum (otherwise a saddle point or local minimum).

5.1.2 Constrained optimization

We introduce a constraint that x must fulfill $max_x f(x)$ with $g(x) = 0$. The stationary points are defined by $\nabla f = -\lambda \nabla g$ for some constant λ
We define the Lagrange primal function

$$L(x, \lambda) = f(x) + \lambda g(x)$$

and the Lagrange multiplier λ
We wish to find a solution $(x*, \lambda*)$ to $\max_x \min_x L(x, \lambda)$ by solving

$$\frac{\partial L}{\partial x} = 0$$

and

$$\frac{\partial L}{\partial \lambda} = 0$$

which is the same as $\nabla L(x, \lambda) = 0$
The stationary points, $x*$, might be local maxima, local minima or saddle points. Verify that the Hessian is negative semi-definite.
One example from lecture [5, p. 8], where we $\max_x f(x_1, x_2) = 1 - x_1^2 - x_2^2$ with constraint $g(x_1, x_2) = x_1 + x_2 - 1 = 0$
The Lagrangian is

$$L(\boldsymbol{x}, \lambda) = f(x) + \lambda g(x) = 1 - x_1^2 - x_2^2 + \lambda (x_1 + x_2 - 1)$$

were we $\nabla L(\boldsymbol{x}, \lambda) = 0$ and then we have 3 functions with 3 unknown and the solution is $(x_1^*, x_2^*, \lambda) = (1/2, 1/2, 1)$

5.1.3 Inequality constraints

Constrained optimization with inequality constraints with $max_x f(x)$ with $h(x) \geq 0$. The optimum is either within the feasible region $h(x) \geq 0$ or along the edge $h(x) = 0$, see the lecture [5, p. 9] for pictures
Notice that ∇h is always pointing inwards since $h > 0$ in the feasible region and $h = 0$ along the edge.
The Lagranges problem for inequality constraints, a local maximum to the constrained optimization problem with $max_x f(x)$ with $h(x) \geq 0$ then the Lagrange function is

$$L(x, \mu) = f(x) = \mu h(x)$$

is given by $(x*, \mu*)$ when KKT conditions are

- $\nabla_x L(x*, \mu*) = 0$
- $\mu* \geq 0$
- $\mu * h(x*) = 0$ complementarity conditions
- $h(x*) \geq 0$
- Negative definite constraints on Hessian

For a minimization problem we change sigh $L(x, \mu) = f(x) - \mu h(x)$

5.1.4 Multiple constraints

Constrained optimization with inequality constraints and equality with $max_x f(x)$ with $h(x) \geq 0$ and $g(x) = 0$ then the Lagrange multipliers $L(x, \lambda, \mu) = f(x) + \sum_j \lambda_j g_j(x) + \sum_k \mu_k h_k(x)$

5.1.5 Lagrange dual problem

The Lagrange primal problem is

$$\max_x \min_\lambda L(x, \lambda, \mu)$$

If we swap the order of min and max we get the Lagrange dual problem,

$$\min_\lambda \max_x L(x, \lambda, \mu)$$

Often these two problems have the same solution, where we define Lagrange dual function

$$L_D(\lambda, \mu) = \max_x L_p(x, \lambda, \mu)$$

From lecture [5, p. 14] then Slater's condition says

> The primal and dual optimization problems are equivalent when f is concave and constraints are convex.

- There must be some x fulfilling all constraints
- Linear constraints are OK
- A local optimum will also be the global optimum
- Not necessary to check conditions on the Hessian

5.2 Optimal Separating Hyperplane

From the lecture slides [5, p. 16] then

- Binary classification
- Sometimes data are perfectly separated by a straight line
- No overlap, one class on one side and the other class on the other side
- Not very useful in practice but it can be modified into the powerful Support Vector Machine

From the book [13, p. 130]
We have seen that linear discriminant analysis and logistic regression both estimate linear decision boundaries in similar but slightly different ways.
These procedures construct linear decision boundaries that explicitly try to separate the data into different classes as well as possible. They provide the basis for support vector classifiers.
With a linear decision functions $y_{\text{new}} = \text{sign}(x_{\text{new}}\beta + \beta_0)$
For practical reason we label the two classes 1 and -1, where fitting the model involves choosing values for β and β_0.
Binary classification, extensions can be made as "One vs. the rest" and "One vs one" both approaches uses several models.
So we need a **decision function**, Many hyperplanes can separate the two classes, what would be the optimal?

5.3 Support Vector Machine

From lecture [5, p. 31] we have the SVM. Most classification problems have overlapping classes, where we modify the (Optimal separating hyperplanes) OSH such that we allow for some overlap. This is where SVM comes in, where we use this together with the kernel trick SVM is on of our most flexible classifiers.

From OSH we got

$$\arg\min_{\beta,\beta_0} \frac{1}{2}||\beta||^2$$

such that

$$y_i(x_i\beta - \beta_0) \geq 1 \quad \forall i$$

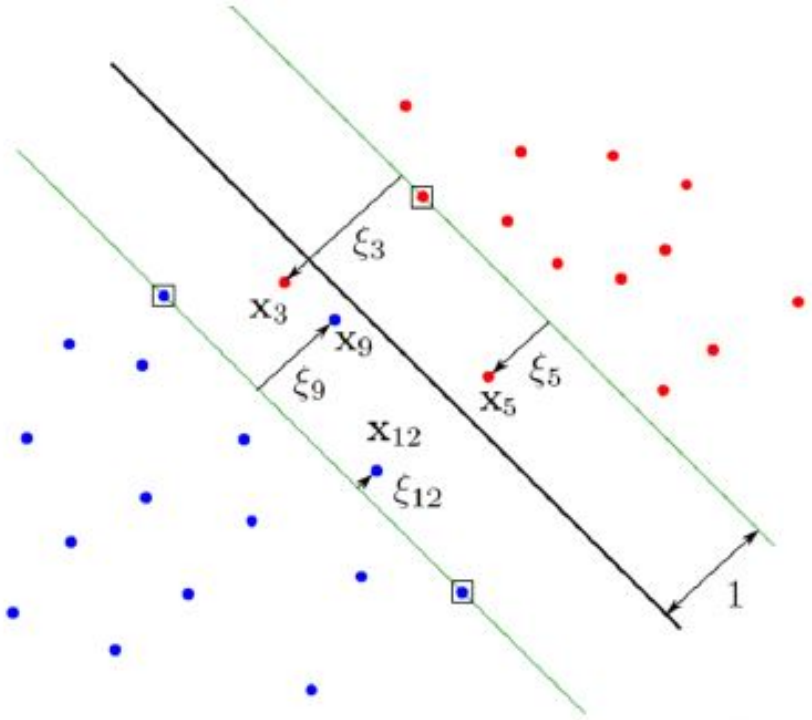
Now we do allow some overlap

$$\arg\min_{\beta,\beta_0} \frac{1}{2}||\beta||^2 + \lambda \sum_{i=1}^{n} \xi_i$$

such that

$$\begin{aligned} y_i(x_i\beta - \beta_0) &\geq 1 - \xi_i \quad \forall i \\ \xi_i &\geq 0 \quad \forall i \end{aligned} \tag{5.4}$$

We give our self a budget for overlap. Smaller budget - larger λ - noisier solution



So our SVM is defined as

$$\arg\min_{\alpha} \alpha\mathbf{1} - \frac{1}{2}\alpha^T Y X X^T Y \alpha$$

such that

$$\begin{aligned} 0 \leq \alpha_i \leq \lambda \forall i \\ \sum \alpha_i y_i = 0 \end{aligned} \tag{5.5}$$

Both are quadratic programming problems with linear constraints. More can be seen on lecture [5, p. 34-35]

5.4 Basis expansion and kernels

- We can do SVM (and OSH) on a transformed feature space

- Transformed features gives non-linear decision boundaries.
- With the Kernel trick we can use an infinite dimensional feature expansion

We use $h(X)$ instead of X

$$\arg\min_{\alpha} \alpha\mathbf{1} - \frac{1}{2}\alpha^T Y h(X) h(X)^T Y \alpha$$

such that

$$\begin{aligned} 0 \leq \alpha_i \leq \lambda \forall i \\ \sum \alpha_i y_i = 0 \end{aligned} \tag{5.6}$$

So the term $h(X)h(X)^T$ does not depend on M, the number of basis functions.
We only need to specify $K(X)$ such that $h(X)h(X)^T = K(X)$ we call K a kernel. Then h is implicitly defined by K.
The common kernels can be seen on lecture [5, p. 38]

$$\arg\min_{\alpha} \alpha\mathbf{1} - \frac{1}{2}\alpha^T Y K(X) Y \alpha$$

such that

$$\begin{aligned} 0 \leq \alpha_i \leq \lambda \forall i \\ \sum \alpha_i y_i = 0 \end{aligned} \tag{5.7}$$

So to classify a new observation

$$\begin{aligned} \hat{y}_{\text{new}} &= sign(\beta h(x_{\text{new}} + \beta_0) \\ &= sign\left(\sum_{i=1}^{n} \alpha_i y_i K(x_{\text{new}}, x_i) + b_0\right) \end{aligned}$$

calculate b_0 using one of the points, i, on the margin

$$b_0 = y_i - \sum_{j=1}^{n} \alpha_j y_j K(x_i, x_j)$$

We have found an efficient way of maximizing the margin between classes and select parameters using cross validation. [5, p. 40,43]
Problem is, that Kernel methods do not scale well. Limited to around 10000-20000 observations. Kernel methods do not do variable selection in any reasonable or

automatic way, e.g. With more features than observations there is always a separating hyperplane.

Other problem is, Potential problem with large number of features if many of them are garbage
Furthermore look at lecture [6, p. 3-6]

Chapter 6

Lecture 6

This lecture is about `Sub-Space Methods [PCA, CCA, PCR, PLS]` where chapter `ESL Chapter 14.5.1, 14.5.5 and 3.5` should be looked upon.

- PCA - Principal Component Analysis
  - The review to rule them all
  - The dos and dont's
  - Applications
- PCR - Principal Component Regression
- PLS - Partial Least Squares
- CCA - Canonical Correlation Analysis

6.1 Chapter 14.5.1

Chapter 14 is about unsupervised learning. This section is about the PCA

6.2 Chapter 14.5.5

This section is about the SPC, which is the Sparce Principal Components

6.3 Chapter 3.5

This chapter is about Linear Methods for Regression, while this section is about methods using derived input directions. Here we will talk about the PCR and PLS

6.4 PCA

So we know that regression and classification are confirmatory. What this means is:

- Answers to particular questions
  - Does wine-drinking influence heart disease? (regression)
  - How well can we separate between normal and abnormal ECG? (classification)
- Supervised - solutions are governed by the outcome variable

PCA is exploratory from lecture [6, p. 8]
This mean, that

- Explore examples of typical (common) observations based on data set
- Unsupervised, no outcome variable - let data speak for itself
  - Structure in data
  - Outlier detection
  - Dimensionality reduction (data compression)

Principal components are discussed in Sections 3.4.1 [13], where they shed light on the shrinkage mechanism of ridge regression. Principal components are a sequence of projections of the data, mutually uncorrelated and ordered in variance.
The principal components of a set data in $\mathbb{R}^p$ provide a sequence of best linear approximations to that data of all ranks $q \leq p$
If we have 100 observations that occupy 75% of 1-D space. Then we would need 100^{10} observations to get the same coverage in 10 dimensions.

6.4.1 PCA - idea

The idea behind PCA, is that we have a linear transformation of data: $S = XL$

We preserve relations between variables $S = XL$ s.t. $L^T L = I$

With the PCA we have $\boldsymbol{Z}$ observations, then we center the data by removing the mean $\boldsymbol{X} = \boldsymbol{Z} - \mu_Z$, then we rotate the coordinate system, first axes in direction of maximal variance and then the observations are given as coordinates in a new coordinate system. $s_i = p_i L$ and $s_i = (s_{i1}, s_{i2})$

PCa idea is a linear transformation of data $S = XL$ and to preserve relations between variables $S = XL$ subject to $L^T L = I$

From lecture [6, p. 15-20], then the PCA derivation is

We have the transformation

$$S = XL, \quad L^T L = I$$

We maximize variance of projected data, which means the maximize variance of each PC/columns of S

$$\text{cov}(S) = \frac{1}{n} S^T S = \frac{1}{n} L^T X^T X L = L^T \Sigma L, \quad \Sigma = \text{cov}(X)$$

The first PC

$$\begin{aligned} \arg\max_I I^T \Sigma I \text{ s.t.} I^T - I = 1 \\ L_P = \lambda(I^T I - 1) \\ \frac{\partial L_p}{\partial I} = 2\Sigma I - 2\lambda I = 0 \& \Leftrightarrow \Sigma I = \lambda I \end{aligned} \tag{6.1}$$

This is the Eigenvalue problem: Covariance is maximized for l equal to the eigenvector of Σ corresponding to the largest eigenvalue λ .
Remaining PCs: Orthogonalize data wrt previous components and repeat. But! L is orthogonal, so no need to orthogonalize. Eigenvectors and -values are the solutions.
S is the score and the size is $n \times p$ and PC's are uncorrelated - $S^T S$ is diagonal.
Where L is the loadings, size is $n \times p$ and columns are known as the principal axes.
It is a rotation matrix $L^T L = I$
With Gaussian data we have $x_i \in N(\mu, \Sigma)$ and transformed data is $s_i \in N(0, D)$ with

$$D = \begin{bmatrix} \sigma_1^2 & \cdot & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_k^2 \end{bmatrix}$$

where $\sigma_1^2 \geq ... \geq \sigma_k^2$
So the singular value decomposition of X

$$X = UDV^T$$

Here U is and $N \times p$ orthogonal matrix $(U^T U = I_p)$ whose columns u_j are called left singular vectors.

V is a $p \times p$ orthogonal matrix ($V^T V = I_p$) with columns v_j called the right singular vectors, and D is a $p \times p$ diagonal matrix, with diagonal elements $d_1 \geq d_2 \geq ... \geq d_p \geq 0$ known as the singular values.

6.5 Sparce PCA

The methods discussed here, are deriving the principal components with sparse loading. They are all based on lasso L_1 penalties.
Since we use a L_1 penalty, then the constraint from [13, p. 550] encourages some of the loadings to be zero.
Further sparse principal components are found in the same way, by forcing the k'th component to be orthogonal to the first $k-1$ components. Unfortunately this problem is not convex and the computations are difficult.
Instead use a regression/reconstruction property of PCA.
From lecture [6, p. 51] express each PC as a regression problem

$$\arg\min_I ||s_i - XI||^2$$

Then optimize wrt. I using the score s_i from PCA. This will give the loadings from PCA.
The problem is that it cannot be solved when $p > n$ (samples > variables). The **solution** is **ridge regression**

$$\arg\min_I ||s_i - XI||^2 + \lambda ||I||^2$$

then Normalize solution to unit length

6.5.1 SPCA using the elastic net

Add a L_1 penalty to get sparse solutions

$$\arg\min_I ||s_i - XI||^2 + \lambda ||I||^2 + \gamma ||I||_1$$

the main Drawback, solution is guided by the original principal components. [6, p. 52]

6.6 Principal Component Regression, PCR

From lecture [6, p. 54] and [13, p. 79] then Principal component regression forms the derived input columns. WE have the scores S. Use $[s_1, s_2, ..., s_M]$ for some $M \leq p$ and we have a standard regression problem in a new variables

$$y = \beta_0 + [s_1, ..., s_M]\beta + e$$

PCR handles n < p by operating on a subset of PCs. PCR performs similar to ridge regression and Equivalent to OLS when M = p

6.7 Partial Least Squares

- Supervised method with latent variable structure
- Seeks directions which have high variance and have high correlation with the response
- Tune number of PLS components

From lecture [6, p. 55] and [13, p. 80]. This technique also constructs a set of linear combinations of the inputs for regression, but unlike principal components regression it uses y (in addition to X) for this construction.

The m'th PLS direction φ_m solves

$$\max_{\alpha} Corr^2(y, X\alpha)Var(X\alpha)$$

Subject to

$$||\alpha|| = 1, \alpha^T \Sigma \varphi_I = 0, I = 1, ..., m-1$$

Further analysis reveals that the variance aspect tends to dominate, and so partial least squares behaves much like ridge regression and principal components regression from [13, p. 82] and from lecture [6, p. 56]

- Behaves similar to ridge regression and principal component regression in shrinking coefficient estimates.
- Shrink low variance directions (like ridge)
- Can inflate high variance directions

6.8 Canonical Correlation Analysis

[6, p. 65]
From Wiki:
In statistics, canonical-correlation analysis (CCA) is a way of inferring information from cross-covariance matrices. If we have two vectors $X = (X_1, ..., X_n)$ and $Y = (Y_1, ..., Y_m)$ of random variables, and there are correlations among the variables, then canonical-correlation analysis will find linear combinations of the X_i and Y_j which have maximum correlation with each other

Chapter 7

Lecture 7

This lecture is about `Unsupervised Clustering [Hierarchical clustering, K-means, GMM, Gapstatistics]` where chapter `ESL Chapter 14.3` should be looked upon.

- Similarity and dissimilarity measures
- K-means clustering
- Hierarchical clustering
- Gaussian mixture
- Validation and model selection

7.1 Cluster analysis

Separating or clustering observations, and the intuitive but vague definition:
Given an underlying set of points, partition them into a collection of clusters so that points in the same cluster are close together, while points in different clusters are far apart.
The purpose for this is seeing structure in data, and gaining an understanding. We could also have a dimensionality reduction or outlier detection.

7.1.1 Proximity Matrices

In what sense are points close in one cluster and far from points in another cluster?
Similarity takes a large value when points are close.
Dissimilarity takes a large value when points are far apart. This reflects the distance between observations.

Any monotone-decreasing function can convert similarities to dissimilarities.
Both similarity and dissimilarity measures can be subjective. For example comparing the taste of three ice creams.

7.1.2 Dissimilarities Based on Attributes

The Euclidean distance

$$d(x_i, x_j) = \sqrt{\sum_{k=1}^{p} (x_{ik} - x_{jk})^2}$$

is useful for **quantitative variables**. Note that ordinal variables can be transformed to a quantitative scale. [7, p. 11]
Then we have the Manhatten distance

$$d(x_i, x_j) = \sum_{k=1}^{p} |x_{ik} - x_{jk}|$$

Quantitative variables and note Manhattan distance also called city block distance.
There is also the Mahalanobi distance

$$d(x_i, x_j) = \sqrt{(x_i - x_j)^T \Sigma^{-1} (x_i - x_j)}$$

The distance is based on data itself and the two points are assumed to be of the same distribution with equal dispersion Σ. **Quantitative variables**
and we also have the Tanimoto distance

$$d(x_i, x_j) = \frac{x_i^T x_j}{x_i^T x_i + x_j^t x_j - x_i^T x_j}$$

Let the sample x have $x_k = 1$ if it possesses the i'th attribute, and $x_k = 0$ otherwise.
The ratio of the number of shared attributes to the number possessed by x_i and x_j.
Often used in information retrieval and biological taxonomy - works well for **categorical variables.**
lastly we have weighted distances

$$d(x_i, x_j) = \sum_{k=1}^{p} w_k d(x_{ik}, x_{jk}), \quad \sum_{k=1}^{p} w_k = 1$$

Give different weight to the p attributes (variables).
Note that setting $w_k = 1/p$ does not necessarily give equal influence to the attributes.
We would have to normalize with the average distance for the k'th attribute.
From [13, p. 504], we learn that the **quantitative variables** are defined as:

Measurements of this type of variable or attribute are represented by continuous real-valued numbers. It is natural to define the "error" between them as a monotone-increasing function of their absolute difference

$$d(x_i, x_j) = l(|x_i - x_j|)$$

While **Ordinal variables** are defined as:

The values of this type of variable are often represented as contiguous integers, and the realizable values are considered to be an ordered set. Examples are academic grades (A, B, C, D, F), degree of preference (can't stand, dislike, OK, like, terrific). Rank data are a special kind of ordinal data. Error measures for ordinal variables are generally defined by replacing their M original values with

$$\frac{i-1/2}{M}, \quad i = 1, ..., M$$

in the prescribed order of their original values. They are then treated as quantitative variables on this scale.
and **Categorical variables** are defined as:

With unordered categorical (also called nominal) variables, the degree-of-difference between pairs of values must be delineated explicitly. If the variable assumes M distinct values, these can be arranged in a symmetric $M \times M$ matrix with elements $L_{rr'} = L_{r'r}, L_{rr} = 0, L_{rr'} \geq 0$. The most common choice is $L_{rr'} = 1$ for all $r \neq r'$, while unequal losses can be used to emphasize some errors more than others.

7.1.3 Object Dissimilarity

If the goal is to discover natural groupings in the data, some attributes may exhibit more of a grouping tendency than others. Variables that are more relevant in separating the groups should be assigned a higher influence in defining object dissimilarity. Giving all attributes equal influence in this case will tend to obscure the groups to the point where a clustering algorithm cannot uncover them.
Often observations have missing values in one or more of the attributes. The most common method of incorporating missing values in dissimilarity calculations

$$d(x_i, x_j) = \sum_{k=1}^{p} w_k d(x_{ik}, x_{jk}), \quad \sum_{k=1}^{p} w_k = 1$$

is to omit each observation pair x_{ik}, x_{jk} between observations x_i and x_j. This method can fail in the circumstance when both observations have no measured values in

common. In this case both observations could be deleted from the analysis. Alternatively, the missing values could be imputed using the mean or median of each attribute over the nonmissing data. For categorical variables, one could consider the value “missing” as just another categorical value, if it were reasonable to consider two objects as being similar if they both have missing values on the same variables

7.1.4 Clustering Algorithms

So unsupervised clustering is that we have data but no information about class belonging. Group data in clusters, and observations that are "near" each other should belong to the same class/cluster. This can help us unveil an unknown structure in data, it is like classification but without an answer.

7.1.5 Combinatorial Algorithms

7.1.6 K-means

We decide how many cluster there should be, this is our "K".
The algorithm alternates between two steps, until assignments do not change. We assign each point to the closest cluster center and compute new cluster centers according to the assignments.
From lecture [7, p. 20] and from [13, p. 509], that K-means is intended for situations in which all variables are of the quantitative type and the squared Euclidean distance is chosen as the dissimilarity measure.
K-mean minimizes within-cluster point scatter. One approach is to directly specify a mathematical loss function and attempt to minimize it through some combinatorial optimization algorithm. Since the goal is to assign close points to the same cluster, a natural loss function would be

$$W(C) = \frac{1}{2}\sum_{k=1}^{K}\sum_{C(i)=k}\sum_{C(j)=k}||x_i - x_j||^2 = \sum_{k=1}^{K} N_k \sum_{C(i)=k}||x_i - \tilde{x}_k||^2$$

where $\tilde{x}_k$ is the mean vector of the k’th cluster and N_k is the number of observations in k’th cluster

Thus, the criterion is minimized by assigning the N observations to the K clusters in such a way that within each cluster the average dissimilarity of the observations from the cluster mean, as defined by the points in that cluster, is minimized.

- For at given cluster assignment C of data points, compute the cluster means $\tilde{x_k}$

$$\tilde{x_k} = \frac{\sum_{i:C(i)=k} x_i}{N_k}, k = 1, ..., K$$

- For a current set of cluster means, assign each observation as

$$C(i) = \arg \min_{1 \leq k \leq K} ||x_i - \tilde{x}_k||^2, i = 1, ..., N$$

- Iterate above two steps until convergence

K is a user input; alternatively BIC (Bayesian information criterion) or MDL (minimum description length) can be used to estimate K. Outliers can considerable trouble to K-means

7.1.7 Gaussian Mixtures as Soft K-means Clustering

The E-step of the EM algorithm assigns "responsibilities" for each data point based in its relative density under each mixture component, while the M-step recomputes the component density parameters based on the current responsibilities

7.1.8 Example: Human Tumor Microarray Data

7.1.9 Vector Quantization

7.1.10 K-medoids

In [13, p. 515] and [7, p. 20] we have about K-medoids
As you know, k-medoid is based on centroids (or medoids) calculating by minimizing the absolute distance between the points and the selected centroid, rather than minimizing the square distance. As a result, it's more robust to noise and outliers than k-means.

7.1.11 Practical Issues

7.1.12 Hierarchical Clustering

In [13, p. 520] and [7, p. 22] we have about Hierarchical Clustering.

- Generates a tree of observations
- Each level of the tree reflects one number of clusters
- From all data in one cluster down to one observation in each cluster.

Furthermore

- Do not require input of number of clusters

- Uses dissimilarity between clusters
- Two approaches
  - Bottom-up: Agglomerative (commonly used)
  - Top-down: Divisive
- $n-1$ levels in the hierarchy
- At each level perform split or merge which gives largest between group dissimilarity

From [13, p. 523] we let G and H represent two group, where the dissimilarity $d(G,H)$ between G and H is computed from the set of pairwise observations dissimilarities. There is the *Single linkage* (SL) takes the intergroup dissimilarity to be that of the closest (least dissimilar) pair

$$d_{SL}(G,H) = \min_{\substack{i\in G \\ j\in H}} d_{ij}$$

Single Linkage tends to give unequal cluster sizes.
This is also often called the nearest-neighbor technique. Complete linkage (CL) agglomerative clustering (furthest-neighbor technique) takes the in- tergroup dissimilarity to be that of the furthest (most dissimilar) pair

$$d_{CL}(G,H) = \max_{\substack{i\in G \\ j\in H}} d_{ij}$$

Complete Linkage tends to give balanced trees independent of data
From lecture [7, p. 29] we have the ward-linkage hierarchical clustering
Cluster-cluster distance measured as the increment in within-cluster sum of squares

$$d_{\text{Ward}} = \sqrt{n_G n_H \frac{||\bar{x}_G - \bar{x}_H||_2^2}{n_G + n_H}}$$

where n_G and n_H are the respective number of observations in each group.
Tends to give a good compromise between balanced/unbalanced clusters
Note: Hierarchical clustering will always generate a dendrogram, even when data is completely random, so be careful with the interpretation.

7.2 The problem of validation

How the we select the number of clusters.
So the Gap-Statistic, compares the log criterion value with K clusters to the expected log criterion value for uniformly distribute data

$$G(K) = \log(U_k) - \log(W_k)$$

where U_k within cluster dissimilarity, simulated data - mean over 20 samples and W_k within cluster dissimilarity, actual data
Then choose

$$K* = \arg\min_k \{K | G(K) \geq G(K+1) - s'_{k+1}\}$$

where

$$s'_{k+1} = std(\log(U_k))\sqrt{1+1/20}$$

To obtain an ideal clustering, you should select k such that you maximize the gap statistic. Here's the exemple given by Tibshirani et al. (2001) in their paper, the plot formed by artificial data with 2 clusters. As you can see, 2 is clearly the ideal k, because the gap statistic is maximized at k=2
However, in many real-world datasets, the clusters are not as well-defined, and we want to be able to balance maximizing the gap statistic with parsimony of the model. Assuming that that plot is just going to continue to increase, of course, the results are less useful. So Tibshirani suggests the 1-standard-error method
Which informally is identifying the point at which the rate of increase of the gap statistic begins to "slow down"

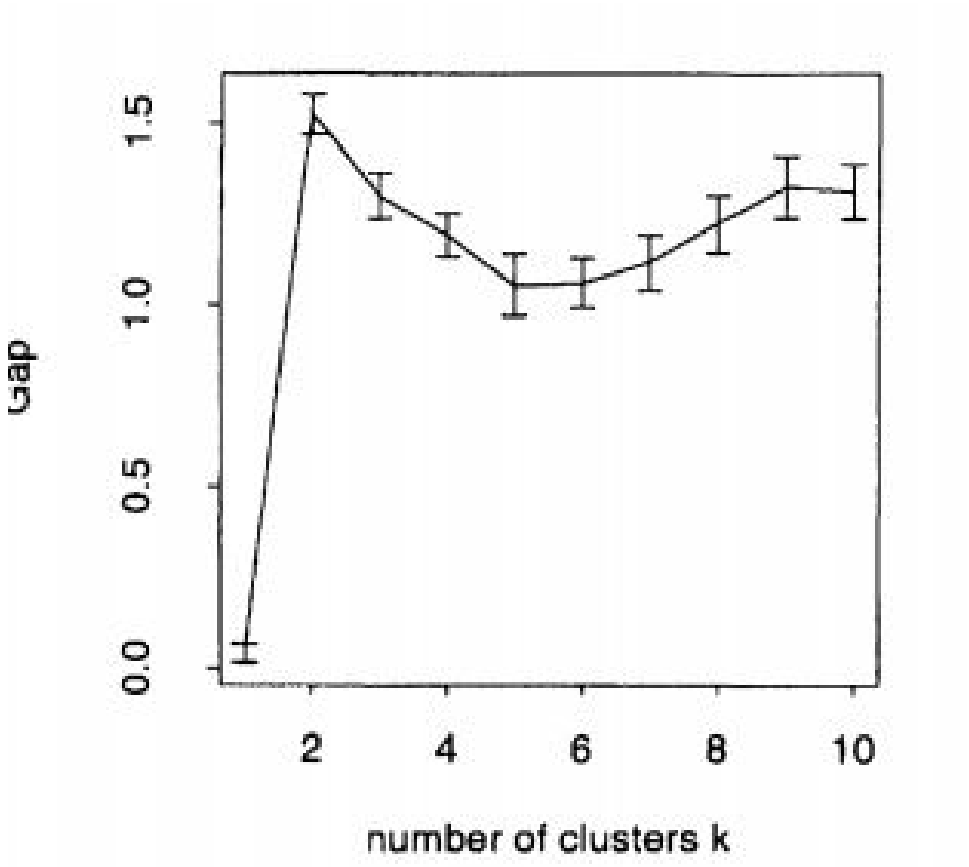
7.3 Gaussian Mixture Modeling and EM

Observation from three Gaussian distributions are $X_i \in N(\mu_1, \Sigma_1)$ if $Z_i = 1$, $X_i \in N(\mu_2, \Sigma_2)$ if $Z_i = 2$ and $X_i \in N(\mu_3, \Sigma_3)$ if $Z_i = 3$ and un-observed variables Z_i indicate cluster/distribution, that is $P(Z_i = 1) = \tau_1$, $P(Z_i = 2) = \tau_2$ and $P(Z_i = 3) = \tau_3$ where $\tau_1 + \tau_2 + \tau_3 = 1$
So the parameters in the Gaussian mixture model are

$$\Theta = (\tau_1, \tau_2, \tau_3, \mu_1, \mu_2, \mu_3, \Sigma_1, \Sigma_2, \Sigma_3)$$

and $\boldsymbol{Z} = (Z_1, ..., Z_n)$. Then the likelihood is

$$\ell(\theta; \boldsymbol{x}, \boldsymbol{Z}) = \prod_{i=1}^{n} \sum_{j=1}^{3} \mathbb{I}_{\{Z_i = j\}} \tau_j f(x_i; \mu_j, \Sigma_j)$$


Gap
0.0
0.5
1.0
1.5
2
4
6
8
10
number of clusters k

with ML estimate

$$\theta_{ML} = \arg\max_{\theta,\boldsymbol{Z}} \log \ell(\theta; \boldsymbol{x}, \boldsymbol{Z})$$

Finding a solution to $(\theta, \boldsymbol{Z})$ is much simplified by the EM-algorithm
The algorithm is a two-step iteration
Expectation step: Define the expectation value

$$Q(\theta|\theta^k) = E_{Z|x,\theta^k} L(\theta; x, Z)$$

Maximization step: Find parameter estimate

$$\theta^{k+1} = \arg\max_{\theta} Q(\theta|\theta^k)$$

First step defines the expectation value of the log likelihood given observed data, x, and current value of the parameter estimate θ^k
Second step chooses optimal θ given the expectation value whereupon the procedure is repeated until convergence.
So the GMM EM algorithm has 3 steps

- Initialize means μ, covariances Σ and mixing coefficients τ
- Expectation step: Calculate conditional probabilities T_{ij} for Z_i belonging to cluster j using Bayes formula
- Maximization step: Calculate weighted (using T_{ij}) mean and covariance estimates μ Σ. Calculate mixing coefficient τ based on mean of weights
- Iterate until convergence

This page is intentionally left blank.

Chapter 8

Lecture 8

This lecture is about `Classification And Regression Trees Discussion of Case 1 and case competition` where chapter `ESL Chapter 9.2` should be looked upon.
Tree-based methods partition the feature space into a set of rectangles, and then fit a simple model.

8.1 Regression Trees

To evaluate the goodness of a new split, we can now simply compute the average sum-of-squares error between the observed y_i's and the mean value $y(v_2)$. This can be done by simply introducing a new impurity measure seen more below
Our data consists of p inputs and a response, for each of N observations: that is, (x_i, y_i) for $i = 1,2,...,N$ with $x_i = (x_{i1}, x_{i2}, ..., x_{ip}$. The algorithm needs to automatically decide on the splitting variables and split points, and also what topology (shape) the tree should have. Suppose first that we have a partition into M regions $R_1, R_2, ..., R_M$ and we model the response as a constant c_m in each region

$$f(x) = \sum_{m=1}^{M} c_m I(x \in R_m)$$

If we adopt as our criterion minimization of the sum of squares

$$\sum_{i \in v} (y_i - y(v))^2$$

then the best $\hat{c}_m$ is just the average of y_i in region R_m

$$\hat{c}_m = ave(y_i | x_i \in R_m)$$

More from [13, p. 307] How large should we grow the tree? Clearly a very large tree might overfit the data, while a small tree might not capture the important structure.

Tree size is a tuning parameter governing the model's complexity, and the optimal tree size should be adaptively chosen from the data. One approach would be to split tree nodes only if the decrease in sum-of-squares due to the split exceeds some threshold. This strategy is too short-sighted, however, since a seemingly worthless split might lead to a very good split below it.

From lecture [8, p. 31], then the way to find the right sized tree, is to grow the tree really large and then decide which splits were unnecessary and remove these.

This is called **pruning** the tree. Pruning a node amounts to removing its sub-tree, thereby making a terminal node

From [13, p. 308] then the preferred strategy is to grow a large tree T_0 stopping the splitting process only when some minimum node size is reached. Then this large tree is pruned using *cost-complexity pruning* , which we now describe.

We define a subtree $T \subset T_0$ to be any tree that can be obtained by pruning T_0, that is, collapsing any number of its internal nodes. We index terminal nodes by m with node m representing region R_m. Let $|T|$ denote the number of terminal nodes in T. Letting

$$N_m = \#\{x_i \in R_m\}$$

$$\hat{c}_m = \frac{1}{N_m} \sum_{x_i \in R_m} y_i$$

To evaluate the goodness of a new split, we can now simply compute the average sum-of-squares error between the observed y_i's and the mean value $\hat{c}_m$. This can be done by simply introducing a new impurity measure:

$$Q_m(T) = \frac{1}{N_m} \sum_{x_i \in R_m} (y_i - \hat{c}_m)^2$$

we define the cost complexity criterion

$$C_\alpha(T) = \sum_{m=1}^{T} N_m Q_m(T) + \alpha |T|$$

The idea is to find, for each α, the subtree $T_\alpha \subseteq T_0$ to minimize $C_\alpha(T)$. The tuning parameter $\alpha \geq 0$ governs the tradeoff between tree size and its goodness of the fit to the data. [13, p. 308]

8.2 Classification Trees

For regression we used sum of squares error node impurity measure, but this is not suitable for classification. In a node m, representing a region R_m, with N_m observations, let

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{x_i \in R_m} I(y_i = k)$$

the proportion of class k observations in node m.

We classify observations in the node to class

$$K = \arg\max_k \hat{p}_k$$

Then different measures of node impurity include the following

The misclassification error

$$Q_m(T) = \frac{1}{N_m} \sum_{i \in R_m} I(y_i \neq k(m)) = 1 - \hat{p}_{mk(m)}$$

The Gini index:

$$Q_m(T) = \sum_{k \neq k'} \hat{p}_{mk} \hat{p}_{mk'} = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$

The Cross Entropy or deviance:

$$Q_m(T) = -\sum_{k=1}^{K} \hat{p}_{mk} \log(\hat{p}_{mk})$$

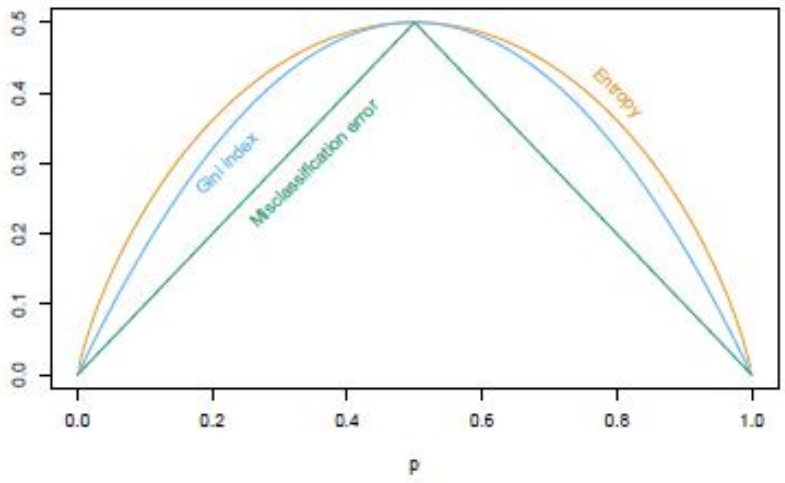


Figure 8.1: ode impurity measures for two-class classification, as a function of the proportion p in class 2. Cross-entropy has been scaled to pass through (0.5, 0.5).

The node impurity is weighted with the number of observations in each node. The split decision is based on the split that minimizes,

$$N_{\text{left}}Q_{\text{left}} + N_{\text{right}}Q_{\text{right}}$$

where, N is the number of observations in left or right node and Q is node impurity for left or right node. Lower is better.
Entropy and Gini index are better measures for growing tree because they are more sensitive to node probabilities. Use Gini index as split criterion when building the tree and Use missclassification rate as criterion when deciding which node to prune from lecture [8, p. 54-55]

8.2.1 Missing data

We can always, delete the observation or replace with mean or median.
For trees we can introduce an extra category "missing" - if it is a categorical variable or we can use a surrogate variable. In each branch, have a list of alternative variables and split points - as a backup.
From lecture [9, p. 3] we know that:

- Advantage:
  - Interpretability, tree defines a set of rules which are easy to follow.
  - Handles missing data.

  - Can take both continuous and categorical variables as input.

- Drawback:
  - Deep trees have high variance and low bias.
  - Small trees have low variance and high bias.
  - New data might completely change the shape of the tree.

This page is intentionally left blank.

Chapter 9

Lecture 9

This lecture is about `Ensemble Methods [Bagging, Boosting and Random Forest]` where chapter `ESL Chapter 8.7, 10.1 and 15` should be looked upon.

- Bootstrapping (recap)
- Theory behind and use of
  - Bagging
  - Boosting
  - Random forest
- Out-of-bag estimates (OOB)

9.1 Chapter 8.7

This chapter is about bagging with examples of Trees and simulated Data.

9.2 Chapter 10.1

This chapter is about boosting methods and an outline of the chapter

9.3 Chapter 15

This chapter is about Random Forests. We get a definition of Random Forests and details of Random Forests. We are introduced to Out of Bag samples, variable importance, proximity Plots and Random Forests and overfitting. Furthermore we are looking at analysis of Random Forests, the Variance and De-Correlation Effect, Bias and Adaptive Nearest Neighbors.

9.4 Bootstrapping Reviewed

9.5 Bagging

Many models built on bootstrap replicates of data. The output from all models aggregated into one output.

Bagging is using the bootstrap to improve predictions, where bagging averages predictions over a collection of bootstrap samples. Average many noisy but approximately unbiased models and thereby reduce the variance.
The Bagging algorithm

- Make B bootstrap samples of size N
- For $b = 1$ to B repeat step (a)

  a Fit a model using the b'th sample and make the prediction $\hat{y}_b$
- The bagging estimate is then given by the average of the ensemble of B predictors

$$\hat{y}_{\text{bagging}} = \frac{1}{B}\sum_{b=1}^{B}\hat{y}_b$$

from lecture [9, p. 6-8] and [13, p. 282].
The Bagging bias, from lecture [9, p. 14]. The bias of bagged trees is the same as that of the individual trees, as the trees are identically distributed

$$\begin{aligned} E(\hat{y}-y) =& E\left[\frac{1}{B}\sum_{b=1}^{B}(\hat{y}_b-y)\right] \\ =& \frac{1}{B}\sum_{b=1}^{B}E(\hat{y}_b-y) \\ & E(\hat{y}_b-y) \quad b=1,...,B \end{aligned} \tag{9.1}$$

From [13, p. 285], This suggests that bagging—drawing samples from the training data— will often decrease mean-squared error.
Then the bagging variance from lecture [9, p. 15]
The variance of the average of bagged estimates is (ρ is the correlation between pairs of trees)

$$\rho\sigma^2 + \frac{1-\rho}{B}\sigma^2$$

The second term goes to zero as B increases. We see that the size of the correlation between pairs of bagged trees, ρ, is what limits the variance reduction.

The Bagging method is Particularly good for high-variance, low-biased methods - such as trees.

- Regression: The regression trees are fitted to bootstrap samples of the training data. The result is the average over all of the trees.
- Classification: A committee of trees each cast a vote for the class - and the majority vote is used as the prediction.

9.6 Boosting

- Average many trees, each grown to reweighted versions of the training data.
- Weighting decorrelates the tree, by focusing on regions missed by past trees.

A set of weak classifiers (small trees) are grown in an adaptive manner to remove bias (hence they are not independent).

Weights are put on the observations where the weights are updated to emphasize misclassifications.

The final estimator is a weighted average of all the trees.

Boosting shrinks bias. Therefore, combine small trees (stubs) with high bias. Many stubs together form a (very) good model with Boosting

The AdaBoost.M1 algorithm, the predictions from all of them are then combined through a weighted majority vote to produce the final prediction:

$$G(x) = sign\left(\sum_{m=1}^{M} \alpha_m G_m(x)\right)$$

Here $\alpha_1, \alpha_2, ..., \alpha_M$ are computed by the boosting algorithm, and weight the contribution of each respective $G_m(x)$. Note that

$$G_m(x) \in \{-1, 1\}$$

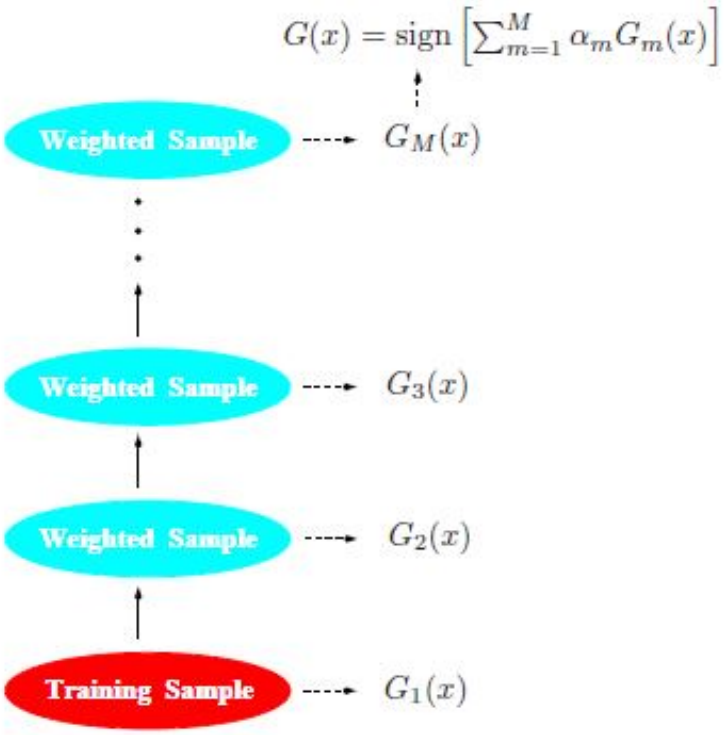


Figure 9.1: Schematic of AdaBoost. Classifiers are trained on weighted versions of the dataset, and then combined to produce a final prediction

Basically the algorithm is

- Initialize the observation weight $w_i = 1/N$, $i = 1, 2, ..., N$
- For $m = 1$ to M:
  - Fit a classifier $G_m(x)$ to the training data using the weights w_i
  - Compute the weighted error of the newest tree

$$err_m = \frac{\sum_{i=1}^{N} w_i \boldsymbol{I}(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}$$

  - Compute $\alpha_m = \log[(1 - err_m)/err_m]$
  - Update weights for $i = 1, ..., N$

$$w_i \leftarrow w_i \cdot \exp(\alpha_m \cdot \boldsymbol{I}(y_i \neq G_m(x)))$$

    and renormalize to w_i to sum to 1

- Output $G(x) = sign\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$

From lecture [9, p. 27] then boosting was intended as a committee method like Bagging.
Is a committee of weak learners where each cast a vote for the final prediction. However, these evolve over time and the members cast a weighted vote.
Boosting dominates Bagging on most problems, and therefore became the preferred choice.

9.6.1 Shrinkage in Boosting

Shrinkage, like ridge, can be added by controlling the learning rate, meaning the contribution of each tree is scaled by a factor $0 < \nu < 1$

9.7 Random Forests

Random forests is in lecture [9, p. 33] and book [13, p. 587]
It is

- Refinement of bagged trees
- Random forest tries to improve on bagging by decorrelating the tree, and reduce the variance

The algorithm for random forest algorithm

- Define number of trees, B. Typically a few hundred (overfitting is not a problem)
- For $b = 1$ to B repeat the steps below
  - Take a random sample of size N with replacement from the data (bootstrapping).
  - Repeat until minimum node size, $n_{\min}$ is reached (do not prune):
    - Take a random sample without replacement of the predictors (of size $m < p$)
    - Construct the first CART partition of the data (pick the best split among the m variables)
- Output: B tress

If the variables are identically distributed, but not necessarily independent, with positive pairwise correlation ρ, the variance of the average is

$$\rho\sigma^2 + \frac{1-\rho}{B}\sigma^2$$

As B increases, the second term disappears, but the first remains, and hence the size of the correlation of pairs of bagged trees limits the benefits of averaging from [13, p. 588]
From lecture [9, p. 37] then for, **classification** drop data down each tree in the forest and classify according to majority vote of B trees. For **regression** drop data down each tree in the forest and prediction is

$$\hat{y} = \frac{1}{B}\sum_{i=1}^{B} T_b(x)$$

where $T_b(x)$ is the estimate of x for each b'th random tree.

9.7.1 Out of bag samples

With lecture [9, p. 38] and [13, p. 592] then an important feature of random forests is its use of *out-of-bag* (OOB) samples.
The OOB estimates

- Samples not included in each bootstrap sample are called out-of-bag (OOB) samples.
- These can fairly be used for assessing individual tree performance!
- Using OOB samples for their corresponding trees, we obtain a missclassification rate.
- Results are similar to cross validation.
- OOB samples provide a way of constructing trees in a single sequence.
- As the forest is growing:
  - Assess prediction error using OOB samples.
  - Stop when error no longer decreases.

9.8 More on comparisons

- $n \ll p$
- Variable importance [13, p. 593]
- Proximity plots [13, p. 595]

So a note about **Bias and variance**.
Bagging and Random Forests have the same bias as the bias of an individual tree. This means that the gain obtained in prediction is due to variance reduction.
Boosting lowers bias as well as variance. Hence, we can use small trees.

9.8.1 Connection to ridge

In ridge regression we see a similar bias-variance trade-off which indicates that bagging and random forests are suitable for $n \ll p$ problems. [9, p. 38]
The ensamble averaging in RF reduces the contribution of any one variable much like the shrinkage in ridge regression and this is particular when m, the number of candidate variables in each split, is small.
But what when we have more variables than observations? e.g. $p > n$ [9, p. 47-49].
Then two measures, The Gini index and OOB estimate.

- Gini: The improvement in the split-criterion at each split is accumulated over all the trees for each variable.
- OOB: Measures prediction strength by first dropping the OOB sample down the tree, then permuting the values for the j'th variable and computing the prediction accuracy again. An average of the difference in accuracy for all the trees gives the measure of importance for variable j.

Variable importance for a random forest classification. The left plot is based on the Gini index whereas the right plot is based on OOB randomization. The OOB approach tends to spread the importance more uniformly. The rankings are usually similar.

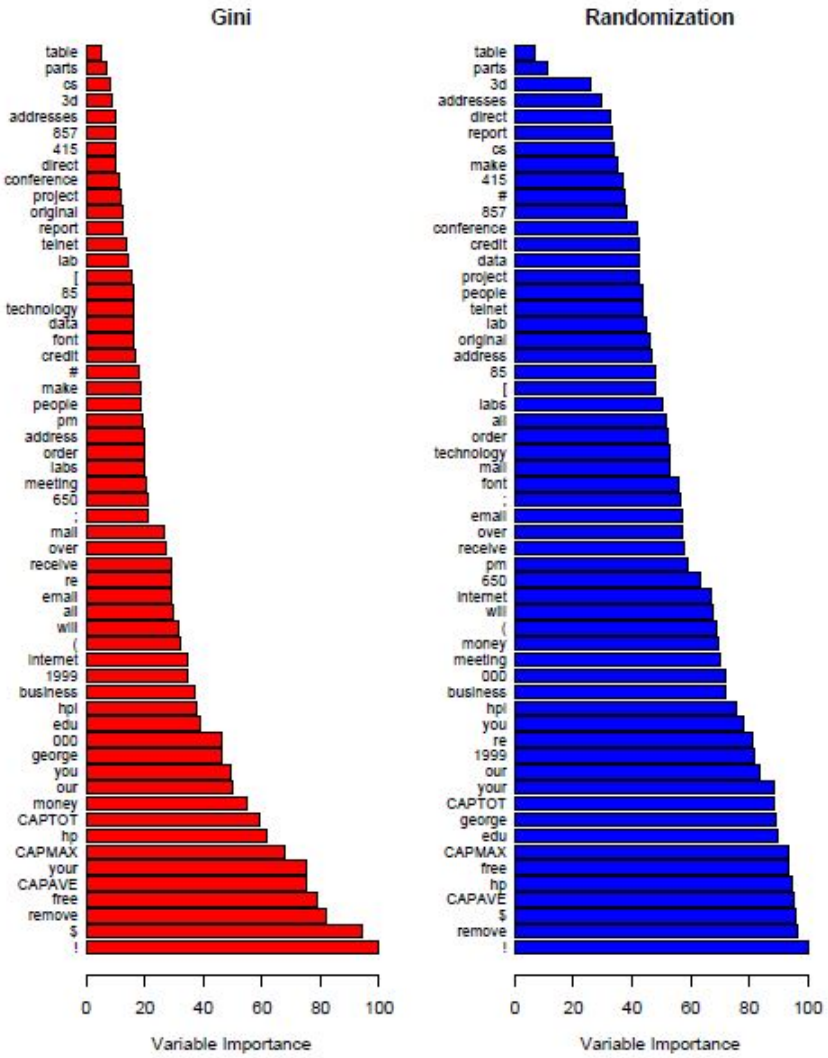


Figure 9.2: Variable importance plots for a classification random forest grown on the spam data. The left plot bases the importance on the Gini splitting index, as in gradient boosting. The right plot uses oobrandomization to compute variable importances, and tends to spread the importances more uniformly.

9.8.2 Proximity matrix and plots

- An $n\times$ proximity matrix is created by increasing the proximity value between two OOB samples by one when they end up in the same terminal node.

  - Hence, a large value in (i; j) reflects that observation i and j are similar according to the classifiers.

- The proximity matrix can be represented in two dimensions using multidimensional scaling. The idea behind this is to find a low-dimensional representation which preserves the pairwise distances.

- In general the proximity plots look like stars where points far from the decision boundary are the extremities, and the points close to the boundary are near the center.

9.8.3 Random forests and Overfitting

More on [13, p. 596]

This page is intentionally left blank.

Chapter 10

Lecture 10

This lecture is about Unsupervised Decomposition [SC, NMF, AA, ICA] where chapter ESL Chapter 14.6, 14.7. Article ‘‘Sparse Coding’’ Nature should be looked upon.

10.1 Chapter 14.6

This chapter is about Non-negative Matrix Factorization

10.2 14.7

This chapter is about Independent Component Analysis and Exploratory Projection Pursuit

- A short introduction to unsupervised learning and Factor analysis
- Non-negative Matrix Factorization
- Archetypal Analysis
- Independent Component Analysis
- Sparse Coding

Decomposition-the process of finding hidden internal representation of the data, i.e., to decompose the data into its internal representations.
Guiding Principle-simplicity of the representation.

10.3 Non-negative Matrix Factorization

It is a approach to principal components analysis, in which the data and components are assumed to be non-negative. It is useful for modeling non-negative data such as images.
The $N \times p$ data matrix $\boldsymbol{X}$ is approximated by

$$\boldsymbol{X} \approx \boldsymbol{W}\boldsymbol{H}, \quad w_{ij} \geq 0, h_{ij} \geq 0$$

The problem, however, cannot be solved analytically so it is generally approximated numerically.
We want to decompose a matrix into a product of two matrixes which makes it easier to work with than the original matrix:
User can specify the inner dimension of W (number of columns) as long as it is less then number of rows of W and number of columns of H .
However, there is a potentially serious problem with non-negative matrix factorization. Even in situations where $\boldsymbol{X} = \boldsymbol{W}\boldsymbol{H}$ holds exactly, the decomposition may not be unique.

10.4 Archetypal Analysis

Very similar to Previous NMF, but the data points are approximated by prototypes that are themselves linear combination of data points:
So, might work really good if we have a lot of information stored in similar clusters that are easily represented by a subset of samples.

10.5 Independent Component Analysis

While PCA is about correlation and maximising variance (good for reconstruction).
ICA - TRYING TO MAXIMIZE INDEPENDENCE.
It tries to find a linear transformation from sample feature space to a new feature space such that each of the individual new features are mutually independent = Mutual information of zero:

$$I(y_i, y_j) = 0$$

and mutual information between y and x is maximised
From lecture [10, p. 22], then ICA is a modern approach to the rotational ambiguity. The factorization is

$$\boldsymbol{X} = \boldsymbol{A}\boldsymbol{S}$$

We can assume $\boldsymbol{X}$ is pre-whitened such that $\boldsymbol{X}\boldsymbol{X}^T = \boldsymbol{I}$ if this is not the case perform SVD, such that

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$$

Where each of the columns of $\boldsymbol{X}$ is a linear combination of the columns of $\boldsymbol{S}$. Now since $\boldsymbol{U}$ is orthogonal, and assuming as before that the columns of $\boldsymbol{X}$ (and hence $\boldsymbol{U}$) each have mean zero, this implies that the columns of $\boldsymbol{S}$ have zero mean, are uncorrelated and have unit variance. In terms of random variables, we can interpret the SVD, or the corresponding principal component analysis (PCA) as an estimate of a latent variable model from [13, p. 558]
And perform ICA on $\boldsymbol{Y} = \boldsymbol{V}$, so

$$[\tilde{\boldsymbol{A}}, \boldsymbol{S}] = \text{ICA}(\boldsymbol{Y})$$

Then from lecture [10, p. 22] then $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\tilde{\boldsymbol{A}}\boldsymbol{S} = \boldsymbol{A}\boldsymbol{S}$ where $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\tilde{\boldsymbol{A}}$
Independent implies that $\boldsymbol{S}$ is uncorrelated (weaker condition):

$$\boldsymbol{S}\boldsymbol{S}^T = \boldsymbol{I}$$

But as

$$\boldsymbol{Y}\boldsymbol{Y}^T = \tilde{\boldsymbol{A}}\boldsymbol{S}\boldsymbol{S}^T\tilde{\boldsymbol{A}}^T = \tilde{\boldsymbol{A}}\tilde{\boldsymbol{A}}^T = \boldsymbol{I}$$

Thus ICA mounts to solving for an orthonormal matrix $\tilde{\boldsymbol{A}}$ such that

$$\boldsymbol{S} = \tilde{\boldsymbol{A}}^T\boldsymbol{Y}$$

are independent and non-Gaussian.
From both lecture [10, p. 22-24] and [13, p. 561] then Many of the popular approaches to ICA are based on entropy. The differential entropy H of a random variable Y with density $g(y)$ is given by

$$H(Y) = -\int g(y)\log g(y)dy$$

A well-known result in information theory says that among all random variables with equal variance, Gaussian variables have the maximum entropy. Finally the *mutual information* $I(Y)$ between components of the random vector Y is a natural measure of dependence

$$I(Y) = \sum_{j=1}^{P} H(Y_j) - H(Y)$$

Add more from [13, p. 561-562] and lecture [10, p. 24]

So; In summary, ICA applied to multivariate data looks for a sequence of orthogonal projections such that the projected data look as far from Gaussian as possible. With pre-whitened data, this amounts to looking for components that are as independent as possible.

10.6 Sparse Coding

Begins from [10, p. 33]
From web:
Sparse coding minimizes the objective

$$||AS - X||_2^2 + \lambda ||S||_1$$

In the equation A is a matrix of bases, S ia a matrix of codes and X is a matrix of data we wish to represent. λ implements a trade of between sparsity and reconstruction.

In the beginning, we do not have S however. Yet, many algorithms exist that can solve the objective above with respect to S. Actually, this is how we do inference: we need to solve an optimisation problem if we want to know the s belonging to an unseen x

Ultimate measure of sparsity given by ℓ_0 norm. However, this results in NP-hard optimization! Instead the ℓ_1 norm is commonly invoked. ℓ_1 is a convex proxy for ℓ_0

Chapter 11

Lecture 11

This lecture is about `Multi-Way Models` where chapter `WireOverview.pdf` should be looked upon.

- A brief history of multi-way analysis
- Tensor Nomenclature
- Tucker Decomposition
- CandeCom/PARAFAC (CP)
- Core Consistency Diagnostic
- Other tensor factorization models
- Missing values
- Software
- Applications

11.1 Tensors

Tensors, or multiway arrays, are generalizations of vectors (first-order tensors) and matrices (second-order tensors) to arrays of higher orders (N > 2). As such, a 3rd order tensor is an array with elements $x_{i,j,k}$

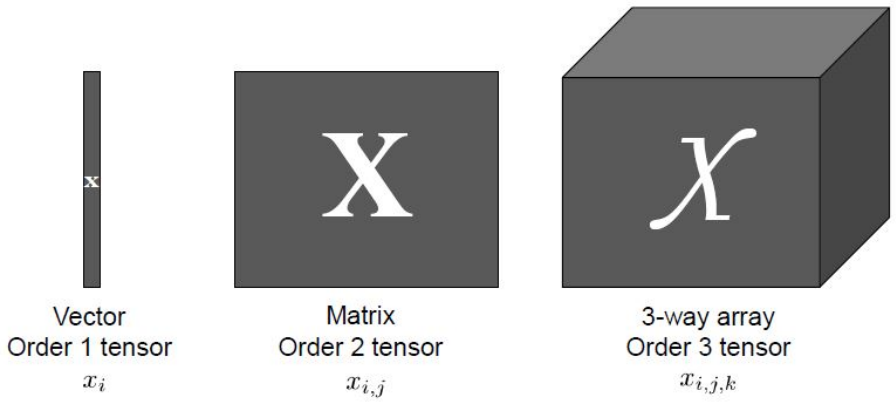


Figure 11.1: Tensor Example

The reason to use tensor decomposition

- Tensor decomposition admit uniqueness of the decomposition without additional constraints such as orthogonalityand independence
- Tensor decomposition methods can identify components even when facing very poor signal to noise ratios (SNR) and when only a relatively small fraction of all the data is observed.
- Tensor decomposition can explicitly take into account the multi-way structure of the data that would otherwise be lost when analyzing the data by collapsing some of the modes to form a matrix

The tensor vs matrix decomposition
Factorizing tensors have several advantages over two-way matrix

- Uniqueness
- Component identification even when only a relatively small fraction of all the data is observed
- Multi-way decomposition techniques can explicitly take into account the multi-way structure of the data that would otherwise be lost when analyzing the data by matrix factorization approaches by collapsing some of the modes

However, factorizing tensors has its challenges

- Its geometry is not yet fully understood
- The occurrence of so-called degenerate solutions
- Lack of guarantee of finding the optimal solution

11.2 Tensor Nomenclature

A general tensor of order N is written

$$\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_n}$$

we will use the following notation to more compactly denote the size of a tensor

$$\mathcal{X}^{I_1 \times I_2 \times \ldots \times I_n}$$

a given element of the tensor $\mathcal{X}$ is denoted by $x_{i_1,i_2,\ldots,i_N}$ from lecture [11, p. 10]. Consider the third order tensor $\mathcal{A}^{I \times J \times K}$ and $\mathcal{B}^{I \times J \times K}$. Scalar multiplication addition of two tensors and the inner product between two tensors are given by

$$\alpha\mathcal{B} = \mathcal{C}, \quad \text{where} \quad c_{i,j,k} = \alpha b_{i,j,k}$$

$$\mathcal{A} + \mathcal{B} = \mathcal{C}, \quad \text{where} \quad c_{i,j,k} = a_{i,j,k} + b_{i,j,k}$$

$$\langle \mathcal{A} + \mathcal{B} \rangle = \sum_{i,j,k} a_{i,j,k} b_{i,j,k}$$

As such, the Frobenius norm of a tensor is given by

$$||\mathcal{A}||_F = \sqrt{\langle \mathcal{A} + \mathcal{A} \rangle}$$

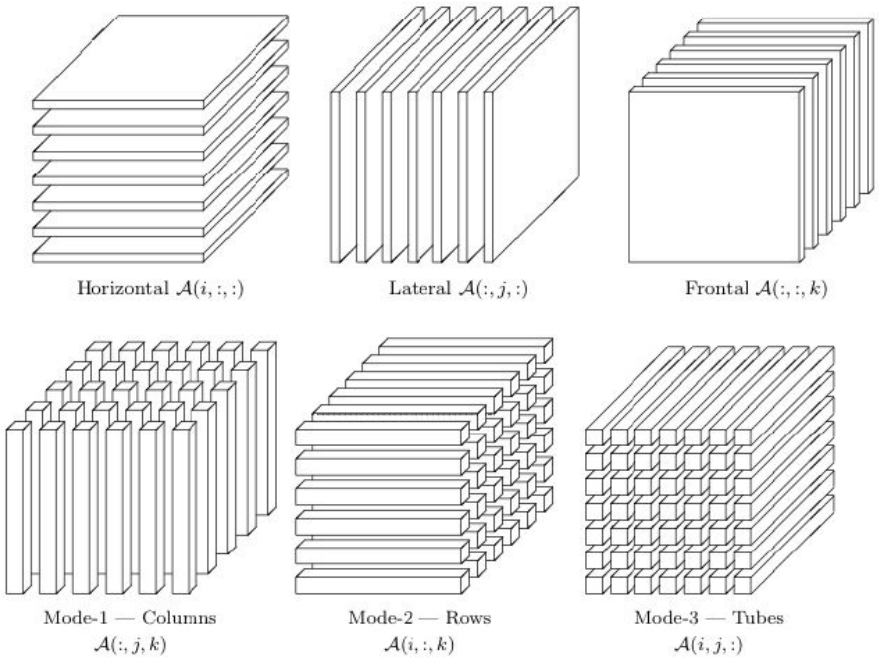
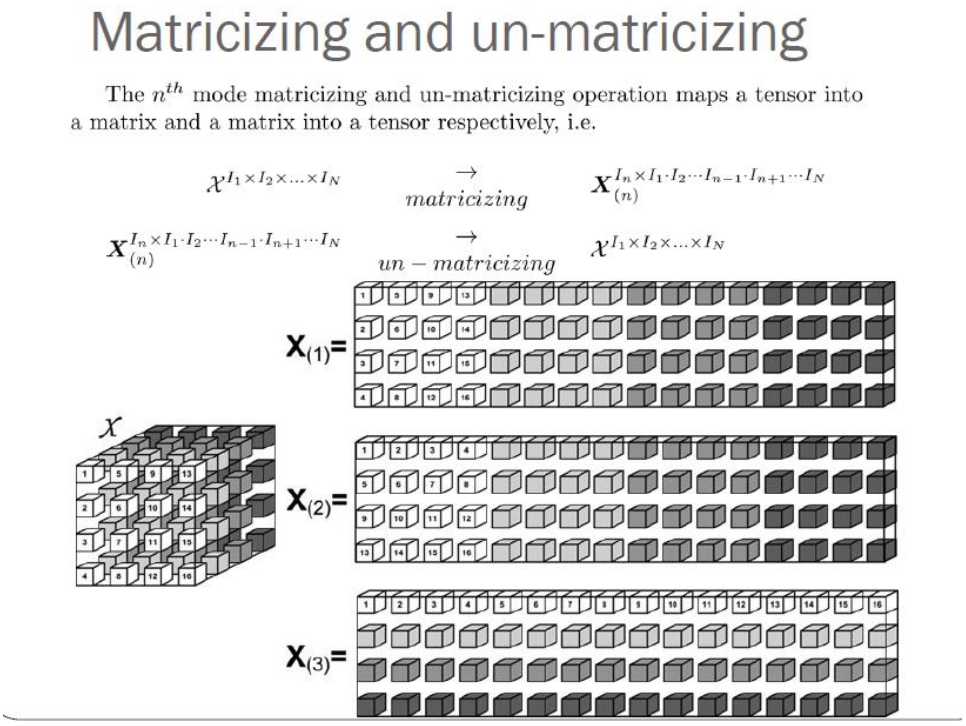


Figure 11.2: Slices and Fibers of tensors



The N-mode multiplication of an order N tensor $\mathcal{X}^{I_1 \times I_2 \times \ldots \times I_n}$ with a matrix $M^{J \times I_n}$ is given by

$$\mathcal{X} \times_n \boldsymbol{M} = \mathcal{X} \bullet_n \boldsymbol{M} = \mathcal{Z}^{I_1 \times I_{n-1} \times Jn \times I_n + 1 \times \ldots \times I_n}$$

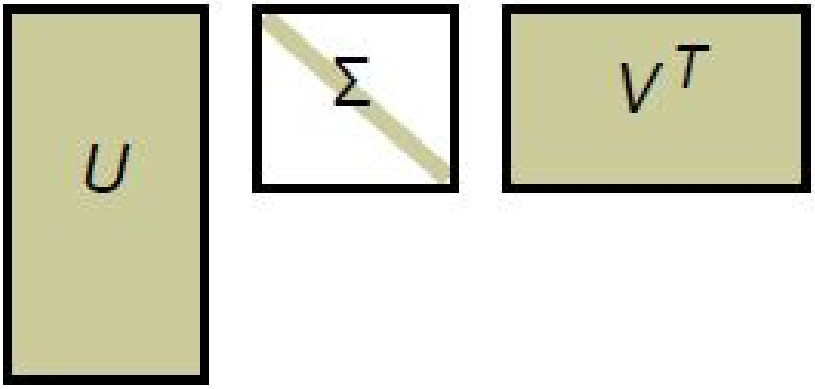
Where
So this operation is given by

$$[\mathcal{X} \times_n \mathbf{M}]_{(\mathbf{n})} = \mathbf{M}\mathbf{X}_{(\mathbf{n})}$$

For we have the SVD as n-mode multiplication
Notation generalizes matrix-matrix products:

$$\mathbf{A} \times_{\mathbf{1}} \mathbf{U} = \mathbf{U}\mathbf{A}$$
$$\mathbf{A} \times_{\mathbf{2}} \mathbf{V} = \mathbf{A}\mathbf{V}^{\mathbf{T}}$$
$$\mathbf{A} = \mathbf{U}\blacksquare\mathbf{V}^{\mathbf{T}} = \Sigma \times_1 \mathbf{U} \times_{\mathbf{2}} \mathbf{V}$$

Can express matrix SVD with n-mode products



It is good to know the Kronecker Product. If $\boldsymbol{A}$ is an $m \times n$ matrix and $\boldsymbol{B}$ is a $p \times q$ matrix then the Kronecker product $\boldsymbol{A} \otimes \boldsymbol{B}$ is the $mp \times nq$ block matrix

$$\boldsymbol{A} \otimes \boldsymbol{B} = \begin{bmatrix} a_{11}\boldsymbol{B} & \cdots & a_{1n}\boldsymbol{B} \\ \vdots & \ddots & \vdots \\ a_{m1}\boldsymbol{B} & \cdots & a_{mn}\boldsymbol{B} \end{bmatrix}$$

More explicity

$$\begin{bmatrix}
a_{11}b_{11} & a_{11}b_{12} & \cdots & a_{11}b_{1q} & \cdots & \cdots & a_{1n}b_{11} & a_{1n}b_{12} & \cdots & a_{1n}b_{1q} \\
a_{11}b_{21} & a_{11}b_{22} & \cdots & a_{11}b_{2q} & \cdots & \cdots & a_{1n}b_{21} & a_{1n}b_{22} & \cdots & a_{1n}b_{2q} \\
\vdots & \vdots & \ddots & \vdots & & & \vdots & \vdots & \ddots & \vdots \\
a_{11}b_{p1} & a_{11}b_{p2} & \cdots & a_{11}b_{pq} & \cdots & \cdots & a_{1n}b_{p1} & a_{1n}b_{p2} & \cdots & a_{1n}b_{pq} \\
\vdots & \vdots & & \vdots & \ddots & & \vdots & \vdots & & \vdots \\
\vdots & \vdots & & \vdots & & \ddots & \vdots & \vdots & & \vdots \\
a_{m1}b_{11} & a_{m1}b_{12} & \cdots & a_{m1}b_{1q} & \cdots & \cdots & a_{mn}b_{11} & a_{mn}b_{12} & \cdots & a_{mn}b_{1q} \\
a_{m1}b_{21} & a_{m1}b_{22} & \cdots & a_{m1}b_{2q} & \cdots & \cdots & a_{mn}b_{21} & a_{mn}b_{22} & \cdots & a_{mn}b_{2q} \\
\vdots & \vdots & \ddots & \vdots & & & \vdots & \vdots & \ddots & \vdots \\
a_{m1}b_{p1} & a_{m1}b_{p2} & \cdots & a_{m1}b_{pq} & \cdots & \cdots & a_{mn}b_{p1} & a_{mn}b_{p2} & \cdots & a_{mn}b_{pq}
\end{bmatrix}$$

and the Khatri-Rao product is

$$A = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix}$$

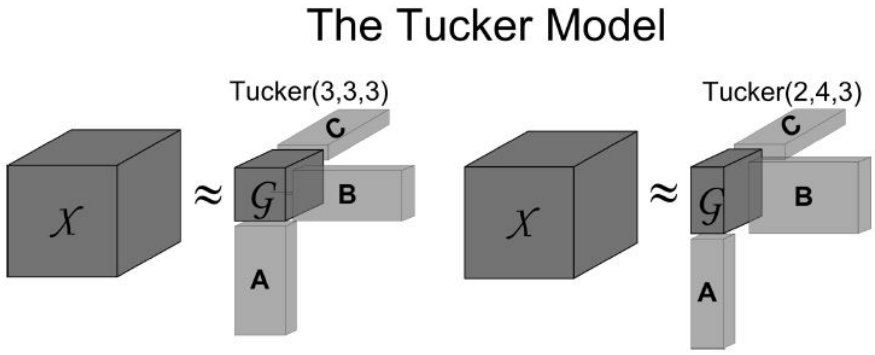
$$B = \begin{bmatrix} g & h & i \\ j & k & l \end{bmatrix}$$

Then $A \odot B$ is the matrix

$$A \odot B = \begin{bmatrix} ag & bh & ci \\ aj & bk & cl \\ dg & eh & fi \\ dj & ek & fl \end{bmatrix}$$

11.3 The Tucker Model

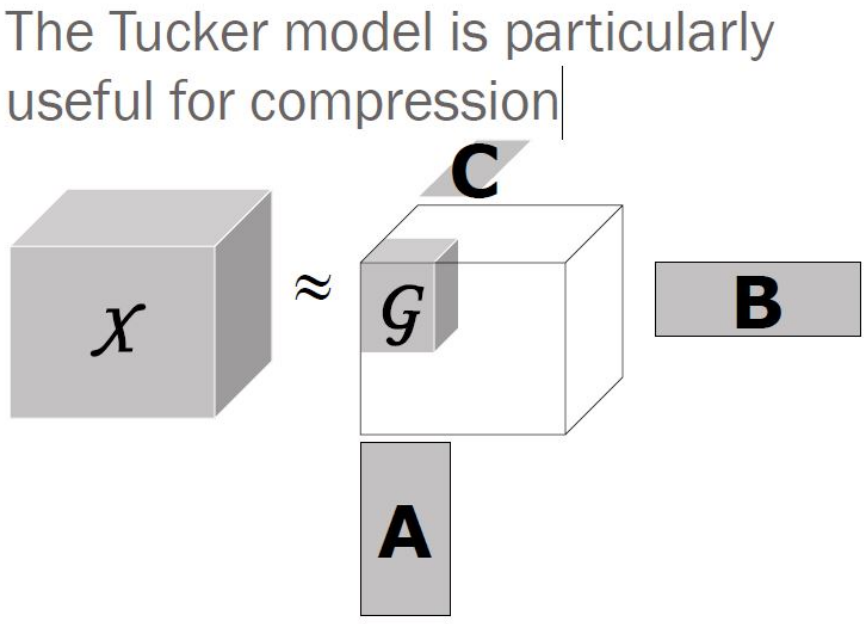
As seen in lecture [11, p. 17]
We have the Tucker model. It reads for a third-order tensor $\mathcal{X}^{I \times J \times K}$



where the so-called core array $\mathcal{G}^{L\times M\times N}$ accounts for all possible linear interactions between the components of each mode.

The Tucker model is not unique

As a result, the factors of the unconstrained Tucker model can be constrained orthogonal or orthonormal (which is useful for compression) without hampering the reconstruction error. However, imposing orthogonality/ orthonormalty does not resolve the lack of uniqueness as the solution is still ambiguous to multiplication by orthogonal/orthonormal matrices Q, R and S.



11.4 The CP model

The CP model can be considered a special case of the Tucker model where the size of each modality of the core array $\mathcal{G}$ is the same, i.e., $L = M = N$

11.5 Missing Values

Although methods based on imputation often are easier to implement (i.e., alternating least squares can be directly applied), they are useful only as long as the amount of missing data is relatively small as their performance tend to degrade for large amounts of missing data as the intermediate models used for imputation have increased risk of convergence to a wrong solution.
Factorizing tensors based on the CP model have been shown to recover the true underlying components from noisy data with up to 99% data missing for third-order tensors,40 whereas the corresponding two-way methods become rather unstable already with 25–40% of data missing

Chapter 12

Lecture 12

This lecture is about `Artificial Neural Networks and Self Organizing Maps` where chapter `ESL Chapter 11.1-11.5 and 14.4` should be looked upon.

- Artificial Neural Networks
- Autoencoders
- Self Organizing Maps

12.1 Chapter 11.1-11.5

This chapter is about the Neural Networks, where we get an introduction. How to fit neural networks then talk about issues of the neural networks, for example, the starting values, overfitting, scaling of the inputs, number of hidden units and layers and multiple minima.

12.2 Chapter 14.4

This section is about Self-Organizing Maps

12.3 Artificial Neural Networks

From lecture [12, p. 6] As a building block in artificial neural networks we introduce the artificial neuron. It takes a weighted sum of the inputs and gives a nonlinear transform as output. This activation function is usually chosen to be a sigmoid function,

$$\sigma(v) = \frac{1}{1+\exp(-v)}$$

So if we have a network of input layers of $\boldsymbol{X} = (x_1, x_2, ..., x_p)$ with hidden layers and output layer(s).
Then the mathematical formulation, for 2 layers, we have

$$a_j = \sum_{i=1}^{P} w_{ji}^{(1)} x_i + w_{j0}^{(1)}$$

where $j = 1, ..., M$ and the superscript (1) indicates that the corresponding parameters are in the first layer of the network.
Each output is then transformed using a differentiable nonlinear activation function to give

$$z_j = h(a_j)$$

In the context of neural networks, these are called hidden units.

$$a_k = \sum_{j=1}^{M} w_{kj}^{(2)} z_i + w_{k0}^{(2)}$$

where $k = 1, ..., K$ and K is total number of outputs. **For standard regression problems, the activation function is the identity so** $y_k = a_k$, but for binary classification problem, then we use the logistic sigmoid function

$$y_k = \sigma(a_k)$$

So so the mathematical expression is

$$f(X) = \sigma\left(\sum_{m=1}^{M} w_m^{(2)} h\left(\sum_{p=1}^{P} w_{mp}^{(1)} x_p + w_{p0}^{(1)}\right) + w_0^{(2)}\right)$$

where this can be written to

$$f(X) = \sigma\left(\sum_{m=0}^{M} w_m^{(2)} h\left(\sum_{p=0}^{P} w_{mp}^{(1)} x_p\right)\right)$$

This is for one hidden layer feed-forward neural network [12, p. 7]
For K-class classification ,there are K units at the top, with the k'th unit modelling the probability of class k.
So, we have from [13, p. 392] that, derived features Z_m are created from linear combinations of the inputs, and then the target Y_k is modeled as a function of linear combinations of the Z_m,

$$Z_m = \sigma(\alpha_{0m} + \alpha_m^T X), \quad m = 1, ..., M$$

$$T_k = \beta_{0k} + \beta_k^T Z, \quad k = 1, ..., K$$

$$f_k(X) = g_k(T), \quad k = 1, ..., K$$

where $Z = (Z_1, Z_2, ..., Z_M)$ and $T = (T_1, T_2, ..., T_K)$
So the softmax function in the output layer is

$$g_k(T) = \frac{\exp(T_k)}{\sum_{\ell=1}^{K} \exp(T_\ell)}$$

This is the same transform used in K-class logistic regression which produces positive estimates that sum to one.
The unknown parameters are called weights and they should be fitted to training data by minimizing a loss function $R(W)$.
For the regression we have

$$R(W) = \sum_{k=1}^{K} \sum_{i=1}^{N} (y_{ik} - f_k(x_i))^2$$

and for classification we have

$$R(W) = -\sum_{k=1}^{K} \sum_{i=1}^{N} y_{ik} \log f_k(x_i)$$

The generic approach to minimize $R(W)$ is by gradient descent, called *back-propagation* in this setting. Because of the compositional form of the model, the gradient can be easily derived using the chain rule for differentiation. This can be computed by a forward and backward sweep over the network, keeping track only of quantities local to each unit.
The advantages of back-propagation are its simple, local nature. In the back propagation algorithm, each hidden unit passes and receives infor- mation only to and from units that share a connection. Hence it can be implemented efficiently on a parallel architecture computer. [13, p. 395]
This is cross-entropy and the output will be a logistic regression in the hidden layer output.
Computing the derived features Z_m, are called hidden units because the values Z_m are not directly observed. In general there can be more than one hidden layer

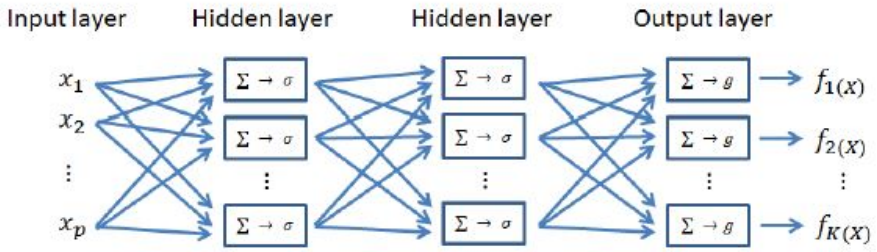


Figure 12.1: ANN with two hidden layers

Hidden layers, the number of hidden layers has to be decided. Each layer extracts features for the next layer allowing for hierarchical features.
Hidden units, more units mean more flexibility. It is usually better with too many than too few units. The extra weights can be shrunken towards zero with regularization

12.4 Autoencoders

Auto encoders are a family of unsupervised neural networks. There are quite a lot of them, e.g. deep auto encoders or those having different regularisation tricks attached–e.g. denoising, contractive, sparse. There even exist probabilistic ones, such as generative stochastic networks or the variational auto encoder.

If you want to solve a prediction problem, you will not need auto encoders unless you have only little labeled data and a lot of unlabeled data. Then you will generally be better of to train a deep auto encoder and put a linear SVM on top instead of training a deep neural net.

However, they are very powerful models for capturing characteristica of distributions. This is vague, but research turning this into hard statistical facts is currently conducted. Deep latent Gaussian models aka Variational Auto encoders or generative stochastic networks are pretty interesting ways of obtaining auto encoders which provably estimate the underlying data distribution

12.5 Some Issues in Training NN

12.5.1 Starting Values

Note that if the weights are near zero, then the operative part of the sigmoid is roughly linear, and hence the neural network collapses into an approximately linear model.

12.5.2 Overfitting

ANNs are very flexible models with many parameters to tune. Two techniques are used to avoid overfitting.

Cross Validation: Cross validation and evaluation on a separate validation set can be used to compare different model orders and to tune a regularization parameter. Fitting ANNs are time consuming and thus this might take a lot of time.

Early stopping: A simplified approach to regularization often applied to ANN. During the optimization, of the weights in the model, performance on a validation set is followed. Instead of having the optimizer finding a minimum we terminate the iteration when performance on validation data does not improve anymore. This approach saves a lot of time.

- We do not have to repeat the calculations for different CV folds and values on a regularization parameter.
- We do not even have to run the optimizer until convergence. We stop when the generalization error starts to increase.

The method does not give the best possible model but the technique is very common when fitting deep networks. [10, p. 24-25]

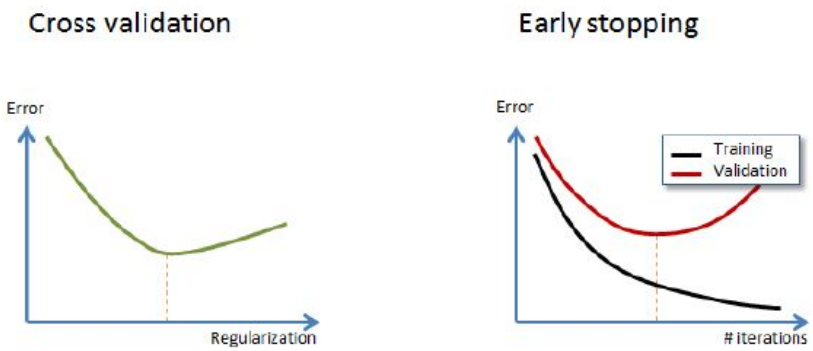


Figure 12.2: Cross-validation vs early stopping

12.6 Self Organizing Maps

- Unsupervised clustering
- Projection of data to a low dimensional space
- Observations are grouped into clusters much like K-means clustering
- The clusters have neighbors in a one or two dimensional grid.
- Neighbor clusters are enforced to lay close to each other also in feature space
- This creates a mapping of data down to one or two dimensions.
- Great for visualization
  - Exploratory data analysis
  - Overview of large amounts of text documents

The observation x_i are processed one at a time. We find the closest prototype m_j to x_i in Euclidean distance in $\mathbb{R}^p$.
The neighbors are defined to be all, such that the distance is small.
More about this on [13, p. 528]

Chapter 13

Case 1

Here we look more into Case 1

This page is intentionally left blank.

Chapter 14

Case 2

Here we look more into Case 2

This page is intentionally left blank.

Appendices

Bibliography

[1] Lars Arvastson and Line Clemmensen. *A - Computational Data Analysis Lecture 1*. May 2018.

[2] Lars Arvastson and Line Clemmensen. *B - Computational Data Analysis Lecture 2*. May 2018.

[3] Lars Arvastson and Line Clemmensen. *C - Computational Data Analysis Lecture 3*. May 2018.

[4] Lars Arvastson and Line Clemmensen. *D - Computational Data Analysis Lecture 4*. May 2018.

[5] Lars Arvastson and Line Clemmensen. *E - Computational Data Analysis Lecture 5*. May 2018.

[6] Lars Arvastson and Line Clemmensen. *F - Computational Data Analysis Lecture 6*. May 2018.

[7] Lars Arvastson and Line Clemmensen. *G - Computational Data Analysis Lecture 7*. May 2018.

[8] Lars Arvastson and Line Clemmensen. *H - Computational Data Analysis Lecture 8*. May 2018.

[9] Lars Arvastson and Line Clemmensen. *I - Computational Data Analysis Lecture 9*. May 2018.

[10] Lars Arvastson and Line Clemmensen. *J - Computational Data Analysis Lecture 10*. May 2018.

[11] Lars Arvastson and Line Clemmensen. *K - Computational Data Analysis Lecture 11*. May 2018.

[12] Lars Arvastson and Line Clemmensen. *L - Computational Data Analysis Lecture 12*. May 2018.

[13] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*. Vol. 2. Springer series in statistics New York, 2016.

This page is intentionally left blank.