

Lecture 14: The End

Dimensionality Reduction, Wrapping up

Prof. Alexandra Chouldechova
95-791: Data Mining

March 3, 2017

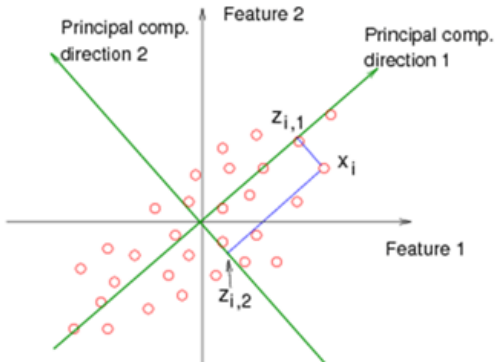
Agenda

- **Dimensionality reduction**
 - Principal Components Analysis [incl limitations]
 - Correspondence analysis
 - Multidimensional scaling

- **Cross-validation pitfalls**

- **Classification with imbalanced data**

A picture

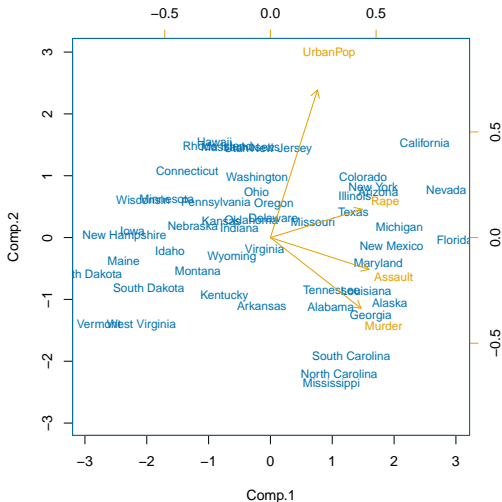


[source: <https://onlinecourses.science.psu.edu/stat857/>]

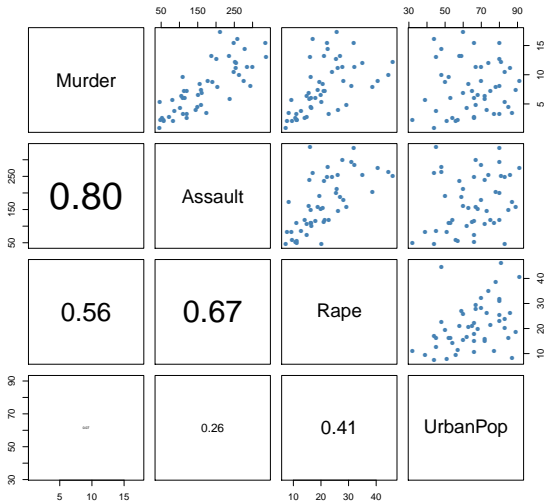
- This Figure shows an observation $x_i = (x_{i1}, x_{i2})$ along with z_{i1} , its projection onto the first principal component direction, and z_{i2} , its projection onto the second principal component direction

Biplot of US Arrests data

```
arrests.pca$loadings = -arrests.pca$loadings # Flip loadings (phi's)
arrests.pca$scores = -arrests.pca$scores # Flip scores (z's)
biplot(arrests.pca, scale = 0) # Construct biplot
```



Why does PCA work well on this data?



In this pairs plot we can clearly see that the crime rate variables---**Murder**, **Assault**, and **Rape**---are **highly correlated** with one another. They provide **redundant information**. The first loading vector winds up forming a combination of these three features, essentially compressing 3 features into 1.

Proportion Variance Explained

- Dimensionality reduction techniques such as PCA work well when the data is **essentially low dimensional**
 - i.e., when there are many groups of highly correlated features
- i.e., We may have p features, but we might be able to describe the data with just $k \ll p$ linear combinations of them
 - For instance, if we have data on children, **height**, **weight** and **age** will all be highly correlated
 - PCA will be able to identify a linear combination of these features that we we'll roughly be able to interpret as the child's **size**
- In general, to understand how well PCA is doing, we look at the **proportion of variance explained** (PVE) of each principal component.

Proportion Variance Explained

- Assume as usual that all the variables have been centered to have mean 0.
- The **total variance** present in the data is defined as

$$\sum_{j=1}^p \text{var}(X_j) = \sum_{j=1}^p \frac{1}{n} \sum_{i=1}^n x_{ij}^2$$

and the variance explained by the m th principal component is

$$\text{var}(Z_m) = \frac{1}{n} \sum_{i=1}^n z_{im}^2$$

- The PVE of the m th component is the ratio of these quantities:

$$\text{PVE}(Z_m) = \frac{\text{var}(Z_m)}{\text{total variance}}$$

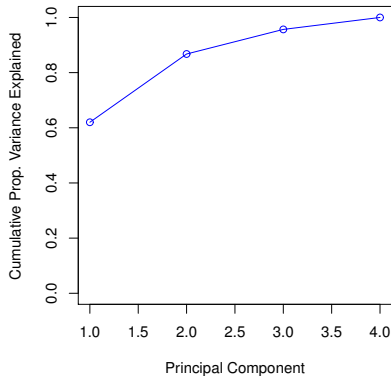
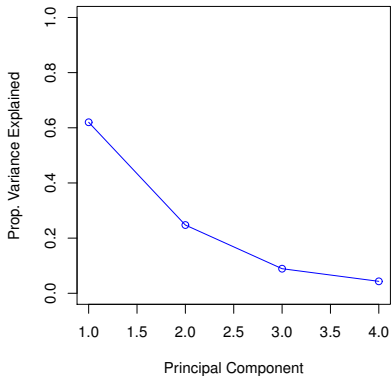


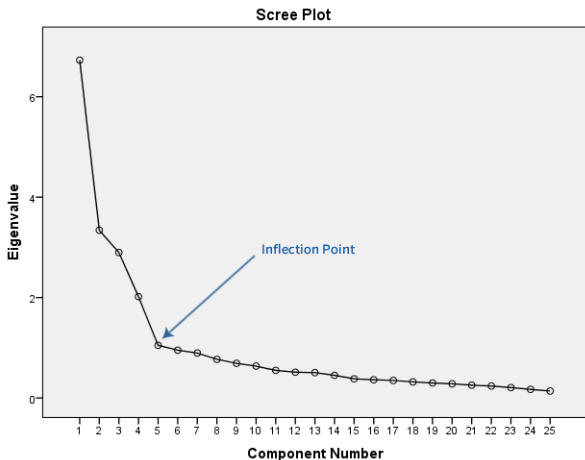
Figure 10.4 from ISL

- Left panel shows $\text{PVE}(Z_m)$ for all 4 principal components in the **USArrests** data
- Right panel shows cumulative PVE: i.e., values of $\sum_{m \leq k} \text{PVE}(Z_m)$ for $k = 1, 2, 3, 4$

How many principal components should we use?

- There's no simple answer to this question
- Cross-validation is not available for this problem
 - CV allows us to estimate test error... but we're doing unsupervised learning here and we don't really have a notion of *test error* to work with
 - If we treated our principal components as *derived features* in a regression or classification task, we could certainly run Cross-validation in that setting
- Often, people like to look at so-called **scree plots**, which is what we showed on the previous slide

Finding elbows in scree plots



[source: <https://gugginotes.wordpress.com/>]

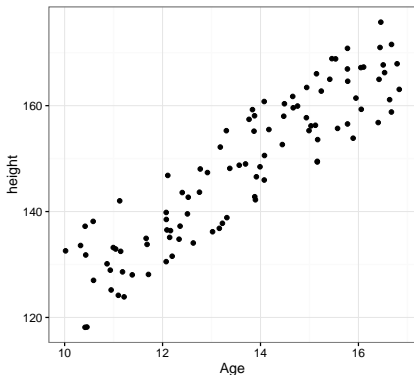
- *Eigenvalue* y -axis label should be interpreted as PVE
- Rule-of-thumb: Stop at the *elbow* in the Scree plot. ($k = 5$ here)

Principal Components Regression

- Suppose we're back in the supervised learning setting where we have observations (x_i, y_i)
- We have a lot of feature (p is large), and we suspect that many of them may be redundant/highly correlated
- We can perform PCA on the data matrix \mathbf{X} , treating this as a feature engineering step
- This will give us $k < p$ new features z_1, \dots, z_k , corresponding to the top k principal components
- Then we can model y on z_1, \dots, z_k instead of on the x_j
- This method is called **Principal Components Regression (PCR)**
 - Of course, there's a classification version of PCR as well

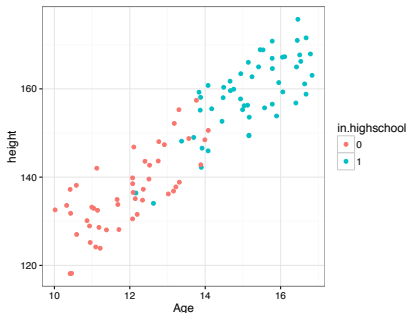
Simple PCR example

- Suppose that we have two features: **age** and **height**
- Everyone in our sample is between 10 and 17 years old, so these features are **highly correlated**
- Instead of using both features in our models, we could just use the 1st principal component.



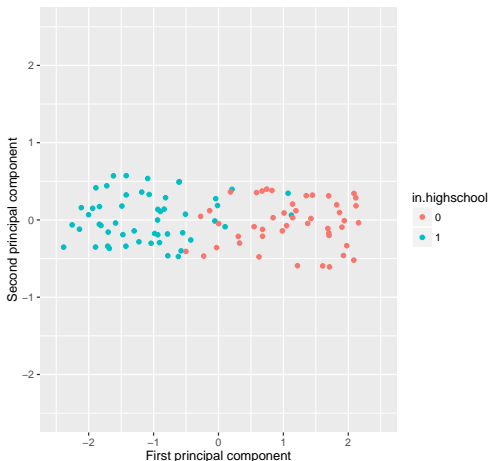
PCR Success case

- Will PCR work? This depends on the outcome, y
- Let's look at a classification setting where $y = 1$ if the individual is in high school, and 0 otherwise.



- Using just the first principal component for classification will work really well here!

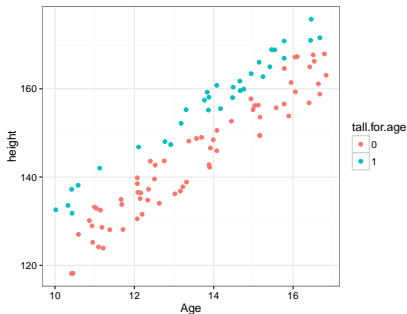
PCR Success case



- This is what the data looks like when plotted in terms of the two principal components
- We can clearly see that a logistic model with $y \sim z_1$ will classify really well

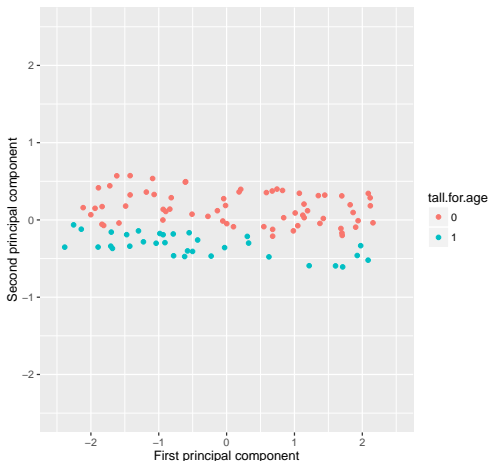
PCR Failure case

- Now what if our outcome was instead $y = 1$ if the person is *tall for their age*, and 0 otherwise.
- Here's a scatterplot of the data, colour-coded by outcome.



- The first principal component is going to be entirely orthogonal to the interesting direction for classification!

PCR Failure case



- A logistic model with $y \sim z_1$ will fail completely!
- **Take-away:** PCR will work well when the leading principal components define directions that capture variation in the outcome y

PCA captures linear directions of variation

- PCA works well when the relationships between the features are **linear** (e.g., when features are linearly correlated)
- Shown below is an example where the two axes are clearly strongly associated, but the association is **non-linear**
- The PCA directions are reasonable... but don't *really* capture the key trend in the data
- **Principal curves** can be applied in such settings: This method generalizes PCA by fitting 1-dimensional *curves* instead of *lines*



PCA

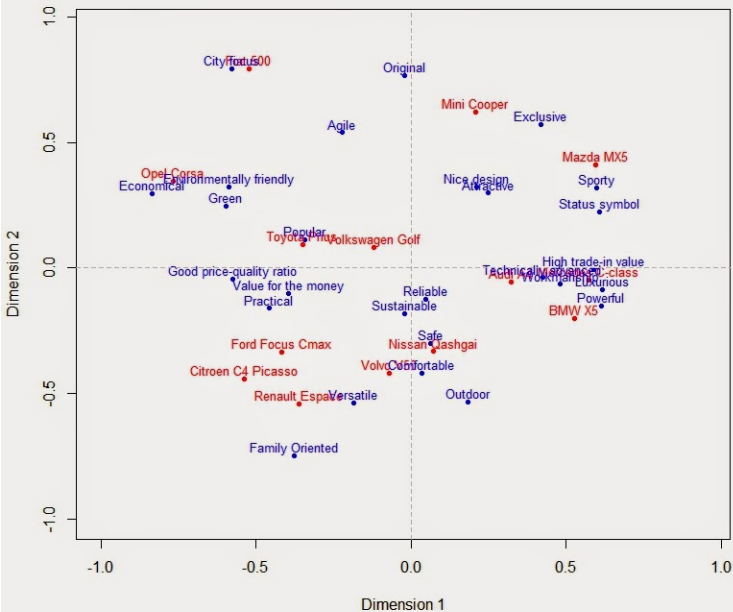


Principal curve

Correspondence Analysis

- PCA is great and widely used
- But it's limited to numeric or *ordinal* data
- We may want to perform dimensionality reduction with **categorical features**
- **Correspondence analysis** is an extension of PCA that gracefully handles categorical features
- The figure on the next page shows a biplot obtained by running Correspondence analysis on a dataset where consumers were asked to check whether or not they associated particular attributes (**blue points**) with various car models (**red points**)

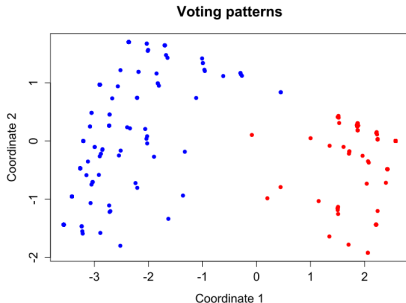
Joint plot



[source: <http://joelcadwell.blogspot.com/>]

Multidimensional scaling

- **Multidimensional scaling** (MDS) is a dimensionality reduction methods for visualizing the level of similarity among individuals in a dataset
- Instead of feeding in the data set \mathbf{X} , we feed in a distance matrix specifying the pairwise distances between all observations
 - Recall: For hierarchical clustering, we also don't need \mathbf{X} . We just operate on the distance matrix
- Here's MDS output from an analysis of voting similarity between **Republicans** and **Democrats** (blue) in the house of representatives



Multidimensional scaling: US cities

- Here's an example of what MDS would do if applied to a matrix giving pairwise distances between all of the “major” cities in the US



- **Awesome!** It doesn't get the right rotation, but we can't expect it to. The reconstruction is otherwise excellent.

REVIEW

Wrapping up

- To wrap up, we'll look at a couple of challenges that commonly arise in real world data
- Topics:
 - Cross-validation: The right way and the wrong way.
 - Classification with imbalanced data

Cross-validation: Right and Wrong

- Suppose we're in a classification setting with $Y \in \{0, 1\}$
- Getting data points is really *expensive*, so we wind up with only $n = 120$ observations
- However, we have *a lot* of features collected on each observation:
 $p = 5000$
- Fitting a model with $n = 120$ and $p = 5000$ is hopeless, so we take the following approach:
 - ① For each of the $p = 5000$ features, calculate the feature's **correlation with y** . Identify the $q = 100$ predictors that have the **highest correlation** with the outcome y .
 - ② Apply a classifier using just these 100 predictors
- **Question:** How do we estimate the **test error** of our classifier?
- Can we apply Cross-validation to just Step 2?

Cross-validation: The wrong way

- 1 For each of the $p = 5000$ features, calculate the feature's correlation with y . Identify the $q = 100$ predictors that have the highest correlation with the outcome y .
- 2 Apply a classifier using just these 100 predictors
 - It is **very wrong** to estimate test error by applying Cross-validation just on Step 2
 - Doing so would ignore the fact that we used the y values of the training data in Step 1
 - To get valid error estimates, each step of Cross-validation **must** go through **every part** of the model-fitting process that in any way **makes use of the outcome variable y**
 - If we just Cross-validate Step 2, we can **easily** wind up with a CV error estimate of 0% when the true test error is 50%.

WRONG approach

Step 1

Identify the 100 predictors with the highest value of $\text{corr}(X_j, y)$

Cross-validation

Split data into K folds

For $k = 1, 2, \dots, K$:

- Fit classifier using all the 100 selected predictors on all observations except those in fold k
- Use classifier to predict labels for data in fold k
- Calculate test error on fold k

Return: average test error across all of the folds

RIGHT approach

Cross-validation

Split data into K folds

For $k = 1, 2, \dots, K$:

- Using observations **not** in fold k , identify the 100 predictors with the highest value of $\text{corr}(X_j, y)$
- Fit classifier using these 100 selected predictors on all observations except those in fold k
- Use classifier to predict labels for data in fold k
- Calculate test error on fold k

Return: average test error across all of the folds

Classification with imbalanced data

- Suppose that we're trying to identify *fraudulent transactions*
- We build a random forest classifier, and find that it gets 97% accuracy
- Looking back at the data, we see that just 3% of our observations were from fraudulent transactions
- Our random forest got 97% accuracy just by classifying $\hat{y}_i = \text{not fraudulent}$ for every observation
- This is an example of **imbalanced data**
- Most classification problems you'll encounter in the real world are imbalanced in this way

Strategies for dealing with imbalanced data

- Stop using *classification accuracy* as your performance metric
 - Instead, assess how well your classifier is doing using more **task-relevant metrics**: Precision, Recall (Sensitivity), Specificity, PPV, **profit**
- Take steps to artificially **improve balance**:
 - Randomly remove instances of your over-represented class (**under-sample**)
 - Randomly add copies of your under-represented class (**over-sample**, sampling *with* replacement)
 - Cluster your over-represented observations into a large number of clusters, and select just one representative observation from each class.
 - Create synthetic samples from your under-represented class using a method like SMOTE (Synthetic Minority Over-sampling Technique)
- If using trees or forests: Try growing very **deep** trees and applying a cost-sensitive pruning technique

Where to go from here?

- We have covered a lot of Data mining methods and concepts in this class
- There is also a lot out there that we didn't get to talk about
- Here's a brief list of things you might be interested in learning more about in the future:
 - **Anomaly detection** (Outlier detection)
 - If we just get to observe *normally behaved* data, how can we figure out if future observations are **anomalous**?
 - Approaches: One-class classification, PCA-based and Clustering-based approaches
 - **Reinforcement learning**:
 - You need to make sequential decisions and learn a good model (a good **policy**) as you go along.
 - This forces a trade-off between **exploration** (seeing what happens if you try a *new idea*), and **exploitation** (going with the best decision according to your current model)
 - E.g., For each customer that browses your site, you need to decide which promos and products to advertise on the landing page. Promos and products change daily, so you need to **learn** and **update** your model *on the fly*

More topics

- **Text mining**
 - Suppose your data set consists of Yelp reviews, and you're trying to use reviews to identify restaurants that are starting to get much better or much worse
 - **Key challenge:** How can we turn text into **features**?
 - **Approaches:** Bag-of-words models, TF-IDF representations (term frequency-inverse document frequency), Topic models (Latent Dirichlet Allocation)
- **Network analysis**
- **Recommender systems** (e.g., Collaborative filtering)
- **Other cool methods**
 - Support vector machines (maximum-margin classifiers)
 - Graphical models, Bayes networks...
 - Neural networks (lately: Deep learning)

The End.

(Thanks!)

Acknowledgements

All of the lectures notes for this class feature content borrowed with or without modification from the following sources:

- 36-462/36-662 Lecture notes (Prof. Tibshirani, Prof. G'Sell, Prof. Shalizi)
- 95-791 Lecture notes (Prof. Dubrawski)
- *An Introduction to Statistical Learning, with applications in R* (Springer, 2013) with permission from the authors: G. James, D. Witten, T. Hastie and R. Tibshirani
- *Applied Predictive Modeling*, (Springer, 2013), Max Kuhn and Kjell Johnson