Feature Selection, Preparation, and Reduction

Feature Selection, Preparation, and Reduction

- Learning accuracy depends on the data!
  - *Is the data representative of future novel cases* - critical
  - Relevance
  - Amount
  - Quality
    - Noise
    - Missing Data
    - Skew
  - Proper Representation
  - How much of the data is labeled (output target) vs. unlabeled
  - Is the number of features/dimensions reasonable?
    - Reduction

Transformed/Derived Variables (Meta-Features)

- Transform initial data features into better ones
  - Quadric machine was one example
- Transforms of individual variables
  - Use area code rather than full phone number
  - Determine the vehicle make from a VIN (vehicle id no.)
- Combining/deriving variables
  - Height/weight ratio
  - Difference of two dates
  - *Do some derived variables in your group project – especially if features mostly given!*
- Features based on other instances in the set
  - e.g. This instance is in the top quartile of price/quality tradeoff
- Transforming/Deriving features requires creativity and some knowledge of the task domain but can be *very* effective in improving accuracy

Feature Selection and Feature Reduction

- Given *n* original features, it is often advantageous to reduce this to a smaller set of features for actual training
  - Can improve/maintain accuracy if we can preserve the most relevant information while discarding the most irrelevant information
  - And/or can make the learning process more computationally and algorithmically manageable by working with less features
  - Curse of dimensionality requires an exponential increase in data set size in relation to the number of features to learn without overfit – thus decreasing features can be critical
- *Feature Selection* seeks a *subset* of the *n* original features which retains most of the relevant information
  - Filters, Wrappers
- *Feature Reduction* <u>*combines/fuses*</u> the *n* original features into a smaller set of newly created features which hopefully retains most of the relevant information from *all* the original features

FEATURE SELECTION

Feature Selection - Filters

- Given n original features, how do you select size of subset
  - User can preselect a size p ($< n$) – not usually as effective
  - Usually try to find the smallest size where adding more features does not yield improvement
- Filters work independent of any particular learning algorithm
- Filters seek a subset of features which maximize some type of between class separability – or other merit score
- Can score each feature independently and keep best subset
  - e.g. 1st order correlation with output, fast, less optimal
- Can score subsets of features together
  - Exponential number of subsets requires a more efficient, sub-optimal search approach
  - How to score features is independent of the ML model to be trained on and is an important research area
  - Decision Tree or other ML model pre-process

Feature Selection - Wrappers

- The feature subset selection algorithm is a "wrapper" around the learning algorithm
  1. Pick a feature subset and pass it to learning algorithm
  2. Create training/test set based on the feature subset
  3. Train the learning algorithm with the training set
  4. Find accuracy (objective) with validation set
  5. Repeat for all feature subsets and pick the feature subset which gives the highest predictive accuracy (or other objective)
- Basic approach is simple
- Variations are based on how to select the feature subsets, since there are an exponential number of subsets

Feature Selection - Wrappers

- Exhaustive Search - Exhausting
- Forward Search – O(n^2 · learning/testing time) - Greedy
  1. Score each feature by itself and add the best feature to the initially empty set *FS* (*FS* will be our final Feature Set)
  2. Try each subset consisting of the current *FS* plus one remaining feature and add the best feature to *FS*
  3. Continue until stop getting significant improvement (over a window)
- Backward Search – O(n^2 · learning/testing time) - Greedy
  1. Score the initial complete *FS*
  2. Try each subset consisting of the current *FS* minus one feature in *FS* and drop the feature from *FS* causing least decrease in accuracy
  3. Continue until dropping any feature causes a significant decreases in accuracy
- Branch and Bound and other heuristic approaches available

FEATURE REDUCTION

PCA – Principal Components Analysis

- PCA is one of the most common feature reduction techniques
- A linear method for dimensionality reduction
- Allows us to combine much of the information contained in n features into p features where $p < n$
- PCA is *unsupervised* in that it does not consider the output class/value of an instance – There are other algorithms which do (e.g. Linear Discriminant Analysis)
- PCA works well in many cases where data features have mostly linear correlations
- Non-linear dimensionality reduction is also a successful area and can give better results for data with significant non-linear correlations between the data features

PCA Overview

- Seek new set of bases which correspond to the highest variance in the data
- Transform *n*-dimensional *normalized* data to a new *n*-dimensional basis
  - The new dimension with the most variance is the first principal component
  - The next is the second principal component, etc.
  - Note z_1 combines/fuses significant information from both x_1 and x_2
- Can drop dimensions for which there is little variance

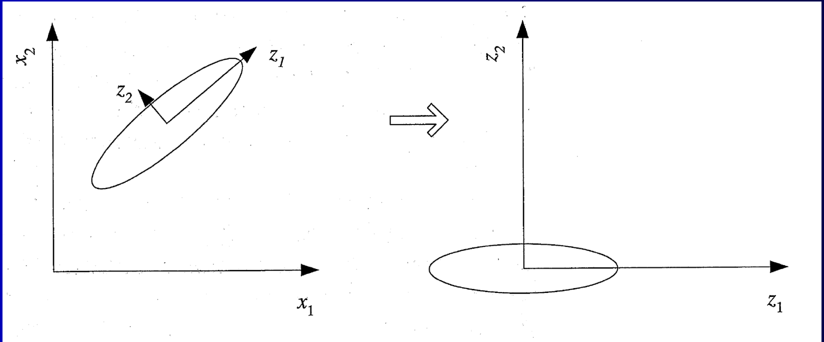


Figure 6.1 Principal components analysis centers the sample and then rotates the axes to line up with the directions of highest variance. If the variance on z_2 is too small, it can be ignored and we have dimensionality reduction from two to one.

Variance and Covariance

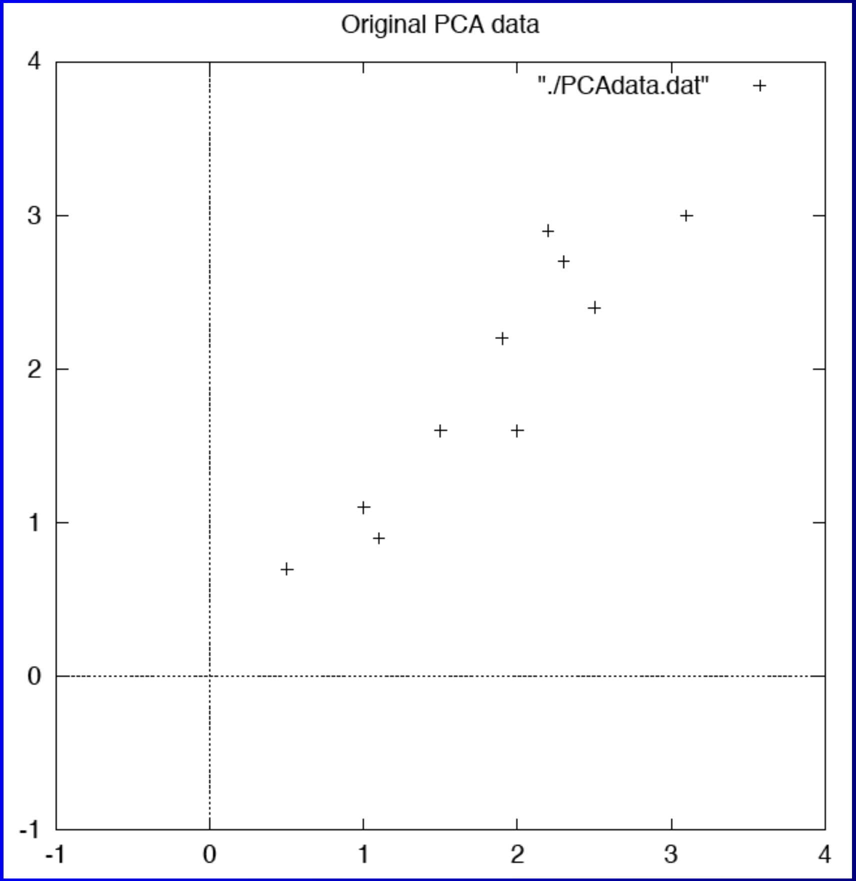
- Variance is a measure of data spread in one feature/dimension
  - *n* features, *m* instances in data set
  - Note *n* in variance/covariance equations is number of instances in the data set, apologies
- Covariance measures how two dimensions (features) vary with respect to each other
- Normalize data features so they have similar magnitudes else covariance may not be as informative

Covariance and the Covariance Matrix

- Considering the sign (rather than exact value) of covariance:
  - Positive value means that as one feature increases or decreases the other does also (positively correlated)
  - Negative value means that as one feature increases the other decreases and vice versa (negatively correlated)
  - A value close to zero means the features are independent
  - If highly covariant, are both features necessary?
- Covariance matrix is an $n \times n$ matrix containing the covariance values for all pairs of features in a data set with n features (dimensions)
- The diagonal contains the covariance of a feature with itself which is the variance (i.e. the square of the standard deviation)
- The matrix is symmetric

PCA Example

- First step is to center the original data around 0 by subtracting the mean in each dimension – (normalize first if needed)



Data	x	y	x'	y'
	2.5	2.4	0.68	0.49
	0.5	0.7	-1.32	-1.21
	2.2	2.9	0.38	0.99
	1.9	2.2	0.08	0.29
	3.1	3.0	1.28	1.09
	2.3	2.7	0.48	0.79
	2.0	1.6	0.18	-0.31
	1.0	1.1	-0.82	-0.81
	1.5	1.6	-0.32	-0.31
	1.2	0.9	-0.62	-1.01
mean	1.82	1.91	0	0

PCA Example

- Second: Calculate the covariance matrix using the centered data
- Only 2 × 2 for this case

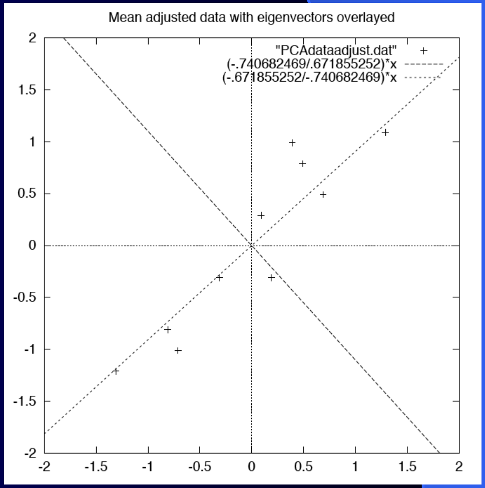
Data	x	y	x'	y'
	2.5	2.4	0.68	0.49
	0.5	0.7	-1.32	-1.21
	2.2	2.9	0.38	0.99
	1.9	2.2	0.08	0.29
	3.1	3.0	1.28	1.09
	2.3	2.7	0.48	0.79
	2.0	1.6	0.18	-0.31
	1.0	1.1	-0.82	-0.81
	1.5	1.6	-0.32	-0.31
	1.2	0.9	-0.62	-1.01
Mean	1.82	1.91	0	0

Covariance	**Matrix**
0.60177778	0.60422222
0.60422222	0.71655556

PCA Example

- Third: Calculate the unit eigenvectors and eigenvalues of the covariance matrix (remember your linear algebra)
  - Covariance matrix is always square $n \times n$ and positive semi-definite, thus n non-negative eigenvalues will exist
  - All eigenvectors (principal components) are orthogonal to each other and form the new set of bases/dimensions for the data (columns)
  - The magnitude of each eigenvalue corresponds to the variance along each new dimension – Just what we wanted!
  - We can sort the principal components according to their eigenvalues
  - Just keep those dimensions with the largest eigenvalues

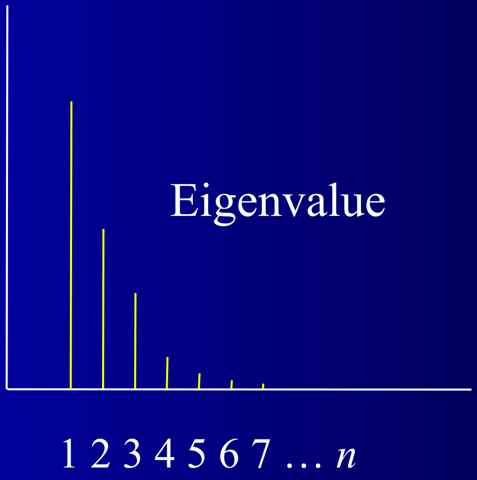
Principal Component	Eigenvalues	Eigenvectors
1	1.26610816	-0.67284685, -0.7397818
2	0.05222517	-0.7397818, 0.67284685



PCA Example

- Below are the two eigenvectors overlaying the centered data
- Which eigenvector has the largest eigenvalue?
- Fourth Step: Just keep the *p* eigenvectors with the largest eigenvalues
  - Do lose some information, but if we just drop dimensions with small eigenvalues then we lose only a little information, hopefully noise
  - We can then have *p* input features rather than *n*
  - The *p* features contain the most pertinent *combined* information from all *n* original features
  - How many dimensions *p* should we keep?

PC	Eigenvalues	Eigenvectors
1	1.26610816	-0.67284685, -0.7397818
2	0.05222517	-0.7397818, 0.67284685



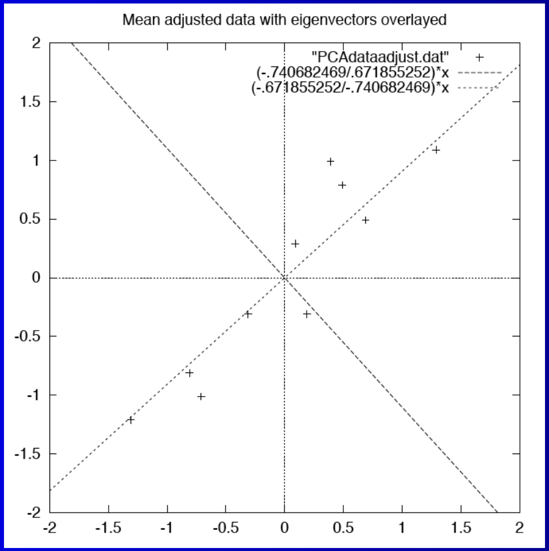
Proportion of Variance

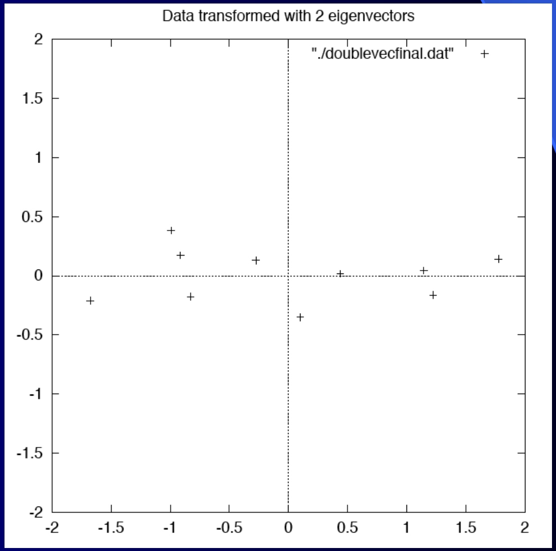
$$\frac{\sum_{i=1}^{p} \lambda_i}{\sum_{i=1}^{n} \lambda_i} = \frac{\lambda_1 + \lambda_2 + \ldots + \lambda_p}{\lambda_1 + \lambda_2 + \ldots + \lambda_p + \ldots + \lambda_n}$$

1.226 / (1.226 + .052) = .96

PCA Example

- Last Step: Transform the n features to the p ($< n$) chosen bases (Eigenvectors)
- Transform data (m instances) with a matrix multiply $T = A \times B$
  - A is a $p \times n$ matrix with the p principal components in the rows, component one on top
  - B is a $n \times m$ matrix containing the transposed centered original data set
  - T^T is a $m \times p$ matrix containing the transformed data set
- Now we have the new transformed data set with p features
- Keep matrix A to transform future centered data instances
- Below is the transform of both dimensions. Would if we just kept the 1st component for this case?





PCA Algorithm Summary

- Center the *n* training set features (subtract the *n* means)
- Calculate the covariance matrix with centered features
- Calculate the unit eigenvectors and eigenvalues of the covariance matrix
- Keep the *p* (< *n*) eigenvectors with the largest eigenvalues
- Matrix multiply the *p* eigenvectors with the centered features to get a new set of *p* features
- Given a novel instance during execution
  - Center the normalized instance (subtract the *n* means)
  - Do the matrix multiply (step 5 above) to change the new instance from *n* to *p* features

PCA Homework

Original Data				
ID	*x*	*y*	*x'*	*y'*
1	.2	-.3	-	-
2	-1.1	2	-	-
3	1	-2.2	-	-
4	.5	-1	-	-
5	-.6	1	-	-
mean			0	0

Terms		
m	5	Number of instances in data set
n	2	Number of input features
p	1	Final number of principal components chosen

- Use PCA on the given data set to get a transformed data set with just one feature (the first principal component). Show your work along the way.
- Show what % of the total information is contained in the 1st principal component.
- Do not use a PCA package to do it. You need to go through the steps yourself, or **program it yourself**. You may use a spreadsheet, Python, etc. to do the arithmetic for you.
- You may use Python or any web tool to calculate the eigenvectors/eigenvalues for the covariance matrix.
- Optional: After, use any PCA solver (e.g. sklearn) and use it to solve the problem and check your answers.

PCA Summary

- PCA is a linear transformation, so if the features have highly non-linear correlations, the transformed data will be less useful
  - Non linear dimensionality reduction techniques can sometimes handle these situations better (e.g. LLE, Isomap, Manifold-Sculpting)
  - PCA is good at removing redundant linearly correlated features
- With high dimensional data the eigenvector is a hyper-plane
- Interesting note: The 1st principal component is the multiple regression plane that delta rule will always discover
- Caution: Not a "cure all" and can lose important info in some cases
  - How would you know if it is effective?
  - Just compare accuracies of original vs transformed data set

Practical Feature Reduction

- Assume you have a data set with 50 features
- You might like to reduce if possible (you might hope for 10 or so, but let the results decide)
- Could try PCA – Compare with non-PCA results to see how effective PCA is for the particular data set
- Could also try a wrapper (e.g. backward greedy) and compare its results and then go with what gives the best accuracy
- PCA
  - Pro: Potentially fuses most information from all features into new smaller set of features
  - Con: Will fail if features have lots of non-linear correlations
- Wrappers
  - Pro: Can handle data features with arbitrary non-linear correlations
  - Con: Does not fuse info, those features which are dropped are completely gone

Group Project Teams

Group 1	Group 2	Group 3	Group 4	Group 5	Group 6	Group 7	Group 8
Talin K.	Zach Y.	Joseph N.	Maddy L.	Nelson S.	Kurt H.	Beckett R.	Grant A.
Hunter G.	Stephen M.	Max P.	Brayden L.	Sam B.	Nicholas C.	Julio F.	Brett A.
Harley C.	David M.	Issac N.	Trace W.	Jordan J.	Jake M.	Adam G.	Jacob B.
		Hyrum D.			Logan S.		

Your Project Proposals

- See description in Learning Suite
  - Remember your example instance!
- Examples – Look at Irvine Data Set, Kaggle, or OpenML to get a feel of what data sets look like
- Stick with supervised classification problems for the most part for the project proposals
- Tasks which interest you
- Too hard vs Too Easy
  - Data should be able to be gathered in a relatively short time
  - And, want you to have to battle with the data/features a bit. You can’t just use a pretty much ready to go data set that you find!