Your Project Proposals

- See description in Learning Suite
    - Remember your example instance!
- Examples – Look at Irvine Data Set 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 can be gathered in a relatively short time
    - Want you to have to battle with the data/features a bit

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

Gathering Data

- Consider the task – What kinds of features could help
- Data availability
  - Significant diversity in cost of gathering different features
  - More the better (in terms of number of instances, not necessarily in terms of number of dimensions/features)
    - The more features you have the more data you need
  - Data augmentation, Jitter – Increased data can help with overfit – handle with care!
- Labeled data is best
- If not labeled
  - Could set up studies/experts to obtain labeled data
  - Use unsupervised and semi-supervised techniques
    - Clustering
    - Active Learning, Bootstrapping, Oracle Learning, etc.

When to Gather More Data

- When trying to improve performance, you may need
  - More Data
  - Better Input Features
  - Different Machine learning models or hyperparameters
  - Etc.
- One way to decide if you need more/better data
  - Compare your accuracy on training and test set
  - If bad training set accuracy then you probably need more and better data, features, noise, etc., or you might need a different learning model/hyperparameters
  - If test set accuracy much worse than training set accuracy then gathering more data is usually a good direction, though regularization or learning model/hyperparameters could still be a major issue

Feature Selection - Examples

- Invariant Data
  - For character recognition: Size, Rotation, Translation Invariance
    - Especially important for visual tasks
  - Chess board features
    - Is vector of board state invariant?
- Character Recognition Class Assignment Example
  - Assume we want to draw a character with an electronic pen and have the system output which character it is
  - Assume an MLP approach with backpropagation learning
  - What features should we use and how would we train/test the system?

Data Representation

- Data Types
  - Continuous
  - Categorical/Symbolic
    - Nominal – No natural ordering
    - Ordered/Ordinal
    - Special cases: Time/Date, Addresses, Names, IDs, etc.
    - Already discussed how to transform categorical to continuous data for models (e.g. perceptrons) which want continuous inputs
- Normalization for continuous values (0-1 common)
  - What if data has skew, outliers, etc.
  - Standardization (z-score) – Transform the data by subtracting the average and then dividing by the standard deviation – allows more information on spread/outliers
  - Look at the data to make these and other decisions!

Transforming Continuous to Ordered Data

- Some models are better equipped to handle nominal/ordered data

- Basic approach is to discretize/bin the continuous data
  – How many bins – what are tradeoffs? – seek balance
  – Equal-Width Binning
    • Bins of fixed ranges
    • Does not handle skew/outliers well
  – Equal-Height Binning
    • Bins with equal number of instances
    • Uniform distribution, can help for skew and outliers
    • More likely to have breaks in high data concentrations
  – Clustering
    • More accurate, though more complex
  – Bin borders are always an issue

Supervised Binning

- The previous binning approaches do not consider the classification of each instance and thus they are unsupervised (Class-aware vs. Class-blind)

- Could use a supervised approach which attempts to bin such that learning algorithms may more easily classify

- Supervised approaches can find bins while also maximizing correlation between output classes and values in each bin
  - Often rely on information theoretic techniques
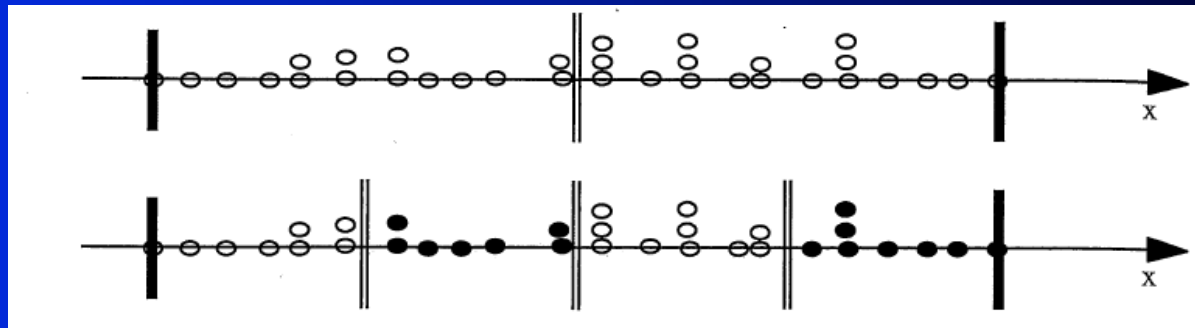
Output Class Skew

- Nuclear reactor data – Meltdowns vs. non-meltdowns
- If occurrence of certain output classes are rare
  - Machine learner might just learn to output the majority class output value
- Most accurate southern California weather forecast
- Approaches to deal with Skew
  - Undersampling: if 100,000 instances and only 1,000 of the minority class, keep all 1,000 of the minority class, and drop majority class examples until you reach your desired distribution (50/50?) – but lose data
  - Oversampling: Make duplicates of every minority instance and add it to the data set until you reach your desired distribution (Overfit possibilities)
    - Could add new copies with jitter (be careful!)
  - Have learning algorithm weight the minority class higher, or class with higher misclassification cost (even if balanced), learning rate, etc.
  - Adjust the classification threshold (e.g. MLP must only exceed .3 to be high, less than majority vote for K-nearest neighbor, etc.)
  - Use Precision/Recall or ROC rather than just accuracy

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
  - This instance is in the top quartile of price/quality tradeoff
- This approach requires creativity and some knowledge of the task domain and can be *very* effective in improving accuracy

Relevant Data

- Typically do not use features where
  - Almost all instance have the same value (no information)
    - If there is a significant, though small, percentage of other values, then might still be useful
  - Almost all instances have unique values (SSN, phone-numbers)
    - Might be able to use a variation of the feature (such as area code)
  - The feature is highly correlated with another feature
    - In this case the feature may be redundant and only one is needed
  - Careful if feature is too highly correlated with the target
    - Check this case as the feature may just be a synonym with the target and will thus lead to overfitting (e.g. the output target was bundled with another product so they always occur together)

Missing Data

BP	HR	Out
.5	67	0
.8	112	1
?	?	1
1	?	0

- Need to consider approach for learning and execution (could differ)
- Throw out data with missing attributes
  - Could lose a significant amount of training set
  - Missing attribute may contain important information, (didn't vote can mean something about congressperson, extreme measurements aren't captured, etc.).
  - Doesn't work during execution
- Set (impute/imputation) attribute to its mode/mean (based on rest of data set)
  - Too big of an assumption?
- Set attribute to its mode/mean given the output class (only works for training)
- Use a learning scheme (NN, DT, etc) to impute missing values
  - Train imputing models with a training set which has the missing attribute as the target and the rest of the attributes (including the original target) as input features. Better accuracy, though more time consuming - multiple missing values?
- Impute based on the most similar complete instance(s) in the data set
- Train multiple reduced input models to handle common cases of missing data
- Let unknown be just another attribute value – Can work well in many cases
  - Natural for nominal data
  - With continuous data, can use an indicator node, or a value which does not occur in the normal data (-1, outside range, etc.), however, in the latter case, the model will treat this as an extreme ordered feature value and may cause difficulties

Dirty Data and Data Cleaning

- Dealing with bad data, inconsistencies, and outliers
- Many ways errors are introduced
  - Measurement Noise/Outliers
  - Poor Data Entry
  - User lack of interest
    - Most common birthday when B-day mandatory: November 11, 1911
    - Data collectors don't want blanks in data warehousing so they may fill in (impute) arbitrary values
- Data Cleaning
  - Data analysis to discover inconsistencies
  - Noise/Outlier removal – Requires care to know when it is noise and how to deal with this during execution – Our experiments show outlier removal during training increases subsequent accuracy
  - Clustering/Binning can sometimes help

Labeled and Unlabeled Data

- Accurately labeled data is always best
- Often there is lots of cheaply available unlabeled data which is expensive/difficult to label – internet data, etc.
- Semi-Supervised Learning – Can sometimes augment a small set of labeled data with lots of unlabeled data to gain improvements
- Active Learning – Out of a collection of unlabeled data, ML model queries for the next most informative instance to label
- Bootstrapping: Iteratively use current labeled data to train model, use the trained model to label the unlabeled data, then train again including most confident newly labeled data, and re-label, etc., until some convergence
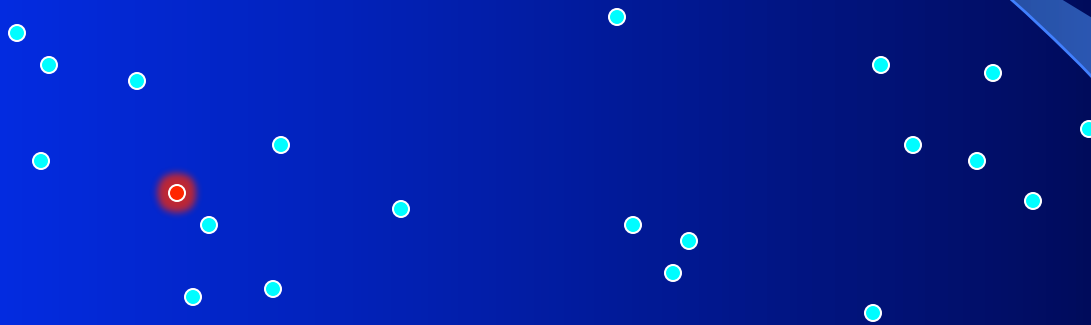- Combinations of above and other techniques being proposed

Active Learning
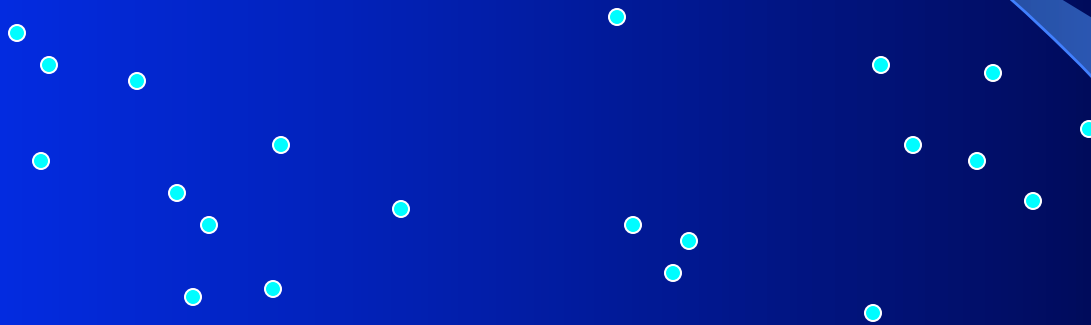
Often query:
1) A low confidence instance (i.e. near a decision boundary)
2) An instance which is in a relatively dense neighborhood

Active Learning

Often query:
1) A low confidence instance (i.e. near a decision boundary)
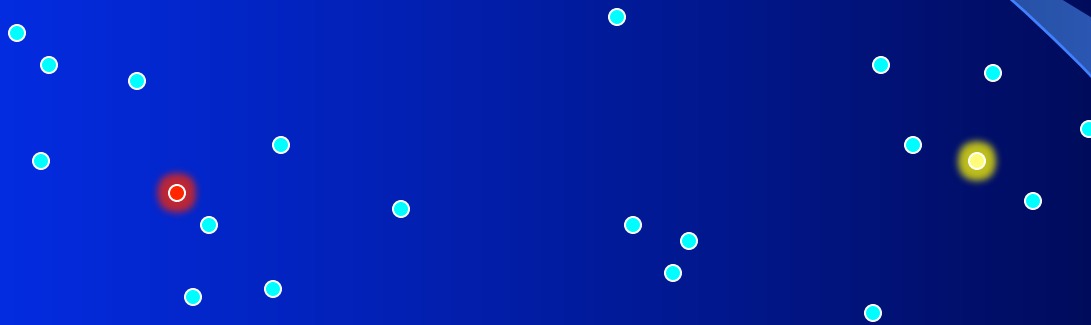2) An instance which is in a relatively dense neighborhood

Active Learning

Often query:
1) A low confidence instance (i.e. near a decision boundary)
2) An instance which is in a relatively dense neighborhood

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 new smaller set of features which hopefully retains most of the relevant information from *all* features - Data fusion (e.g. LDA, PCA, etc.)

Feature Selection - Filters

- Given n original features, how do you select size of subset
  - User can preselect a size m ($< n$)
  - Can 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

- Optimizes for a specific learning algorithm
- The feature subset selection algorithm is a "wrapper" around the learning algorithm
  1. Pick a feature subset and pass it in 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 led to 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 either hit goal of m, or stop getting significant improvement
- Backward Search – O(n^2 · learning/testing time) - Greedy
  1. Score the initial complete set FS (FS will be our final Feature Set)
  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 either hit goal of m, or begin to get significant decreases in accuracy
- Branch and Bound and other heuristic approaches available

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 m features where $m < n$
- 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 has mostly linear correlations
- Non-linear dimensionality reduction is also a relatively new and successful area and can give much better results for data with significant non-linearities

PCA Overview

- Seek new set of bases which correspond to the highest variance in the data

- Transform n-dimensional 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

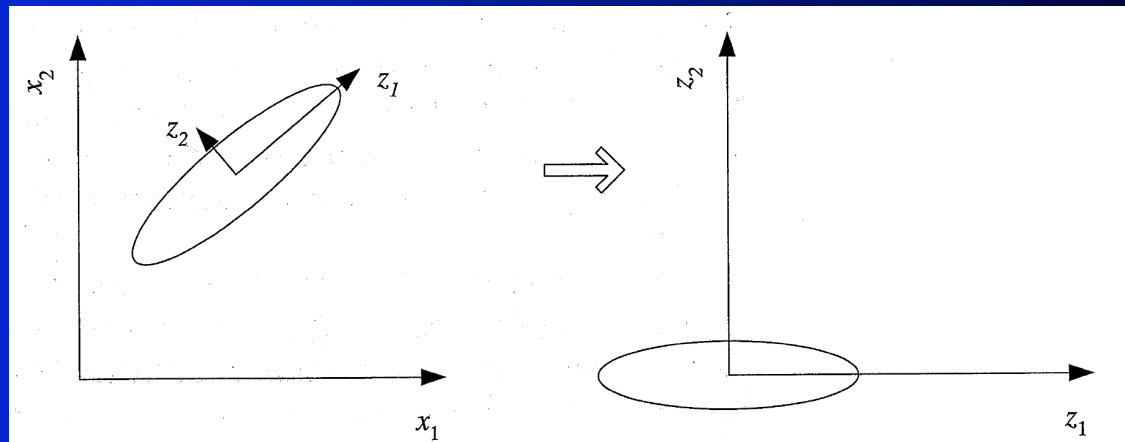- Drop those dimensions for which there is little variance

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 dimension (feature)

- Covariance measures how two dimensions (features) vary with respect to each other
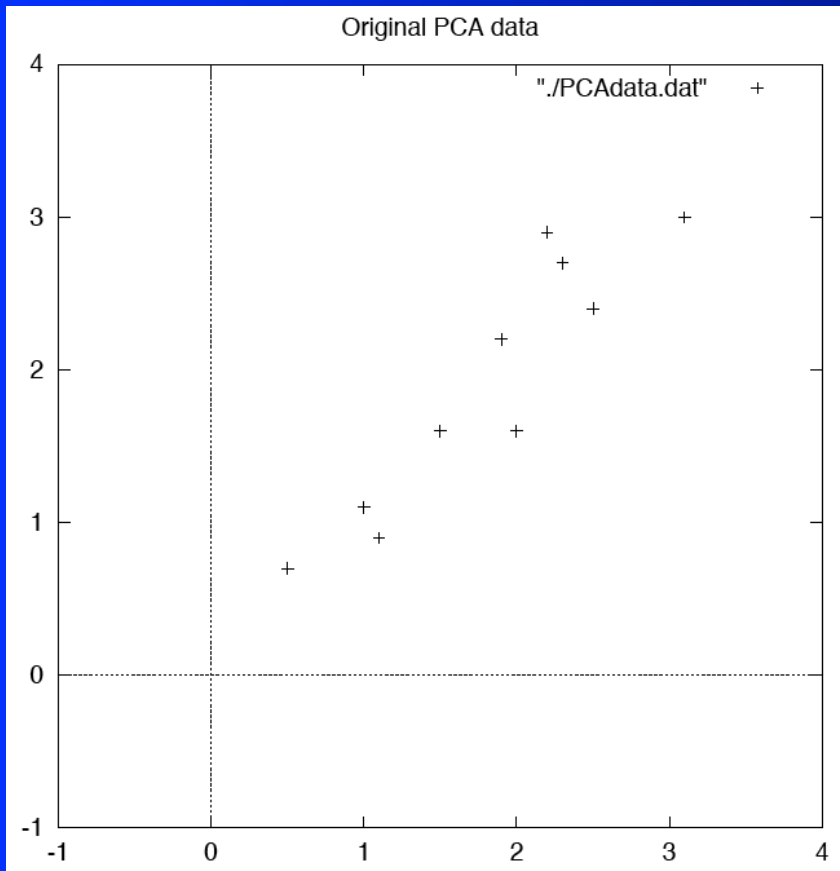
$$\text{var}(X) = \frac{\sum_{i=1}^{n} \left(X_i - \overline{X} \right)\left(X_i - \overline{X} \right)}{(n-1)}$$

$$\text{cov}(X,Y) = \frac{\sum_{i=1}^{n} \left(X_i - \overline{X} \right)\left(Y_i - \overline{Y} \right)}{(n-1)}$$

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 (which is 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

Original PCA data

X	Y			X'	Y'
2.5	2.4			0.69	0.49
0.5	0.7			−1.31	−1.21
2.2	2.9			0.39	0.99
1.9	2.2			0.09	0.29
3.1	3.0	$\Rightarrow$	$\overline{X} = 1.81$ $\Rightarrow$	1.29	1.09
2.3	2.7		$\overline{Y} = 1.91$	0.49	0.79
2.0	1.6			0.19	−0.31
1.0	1.1			−0.81	−0.81
1.5	1.6			−0.31	−0.31
1.2	0.9			−0.71	−1.01

PCA Example

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

X	Y			X'	Y'
2.5	2.4			0.69	0.49
0.5	0.7			−1.31	−1.21
2.2	2.9			0.39	0.99
1.9	2.2			0.09	0.29
3.1	3.0	$\Rightarrow$	$\overline{X}=1.81$ $\overline{Y}=1.91$ $\Rightarrow$	1.29	1.09
2.3	2.7			0.49	0.79
2.0	1.6			0.19	−0.31
1.0	1.1			−0.81	−0.81
1.5	1.6			−0.31	−0.31
1.2	0.9			−0.71	−1.01

$$\mathrm{cov}(X,Y) = \frac{\sum_{i=1}^{n}\left(X_i - \overline{X}\right)\left(Y_i - \overline{Y}\right)}{(n-1)}$$

$$\mathrm{cov} = \begin{bmatrix} 0.616555556 & 0.615444444 \\ 0.615444444 & 0.716555556 \end{bmatrix}$$
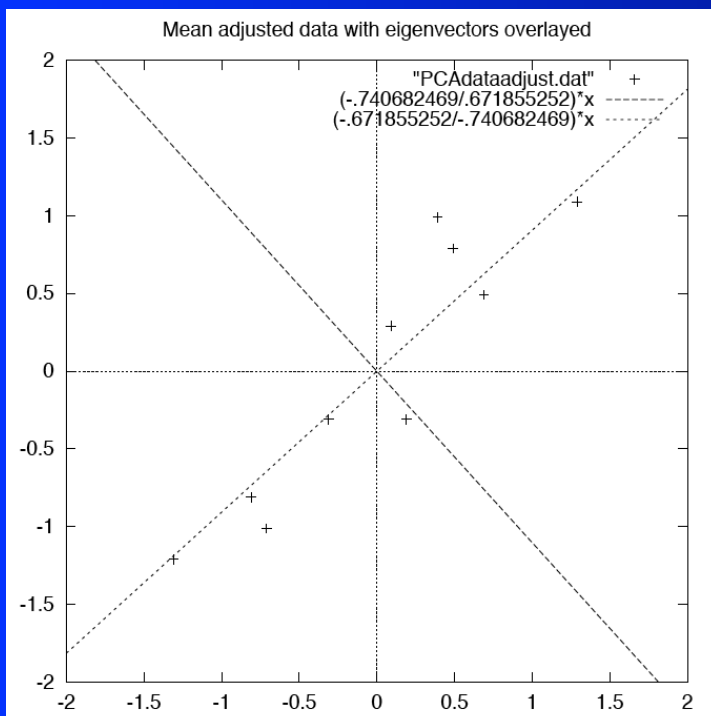
PCA Example

- Third: Calculate the unit eigenvectors and eigenvalues of the covariance matrix (remember 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/dimensions) are orthogonal to each other and will form the new set of bases/dimensions for the data
  - The magnitude of each eigenvalue corresponds to the variance along that new dimension – Just what we wanted!
  - We can sort the principal components according to their eigenvalues
  - Just keep those dimensions with the largest eigenvalues

$$eigenvalues = \begin{bmatrix} 0.490833989 \\ 1.28402771 \end{bmatrix}$$

$$eigenvectors = \begin{bmatrix} -0.735178656 & -0.677873399 \\ 0.677873399 & -0.735178656 \end{bmatrix}$$
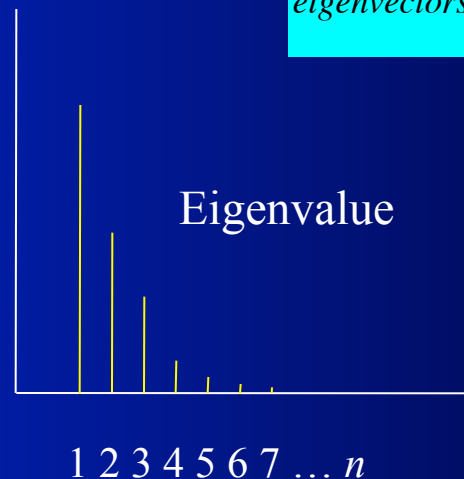
PCA Example

- 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?

$$eigenvalues = \begin{bmatrix} 0.490833989 \\ 1.28402771 \end{bmatrix}$$

$$eigenvectors = \begin{bmatrix} -0.735178656 & -0.677873399 \\ 0.677873399 & -0.735178656 \end{bmatrix}$$

Mean adjusted data with eigenvectors overlayed

"PCAdataadjust.dat"   +
(-.740682469/.671855252)*x
(-.671855252/-.740682469)*x

Eigenvalue

$$1\ 2\ 3\ 4\ 5\ 6\ 7\ \ldots\ n$$

Proportion of Variance

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

PCA Example

- Last Step: Transform the n features to the p ($< n$) chosen bases (Eigenvectors)
- Transformed data (m instances) is 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 dimensionality p
- Keep matrix A to transform future 0-centered data instances
- Below shows transform of both dimensions, would if we just kept the 1st component
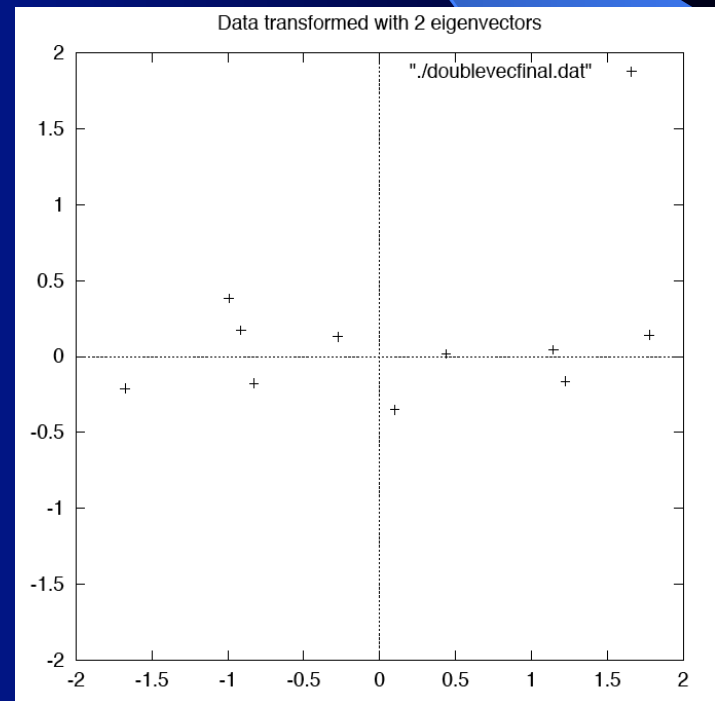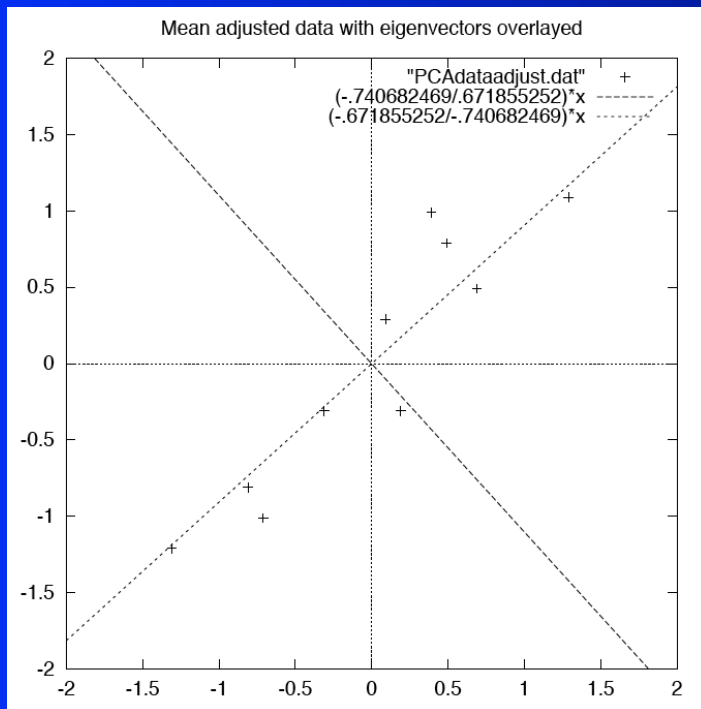
PCA Algorithm Summary

1. Center the m TS features around 0 (subtract m means)
2. Calculate the covariance matrix of the centered TS
3. Calculate the unit eigenvectors and eigenvalues of the covariance matrix
4. Keep the p ($< m$) eigenvectors with the largest eigenvalues
5. Matrix multiply the p eigenvectors with the centered TS to get a new TS with only p features

● Given a novel instance during execution
  1. Center instance around 0
  2. Do the matrix multiply (step 5 above) to change the new instance from m to p features

PCA Homework

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

Original Data		
	x	y
p1	.2	-.3
p2	-1.1	2
p3	1	-2.2
p4	.5	-1
p5	-.6	1
mean	0	-.1

- Use PCA on the given data set to get a transformed data set with just one feature (the first principal component (PC)). Show your work along the way.
- Show what % of the total information is contained in the 1st PC.
- 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, Matlab, etc. to do the arithmetic for you.
- You may use any web tool or Matlab to calculate the eigenvectors from the covariance matrix.

PCA Summary

- PCA is a linear transformation, so if the data is highly non-linear then the transformed data will be less informative
  - Non linear dimensionality reduction techniques can handle these situations better (e.g. LLE, Isomap, Manifold-Sculpting)
  - PCA is good at removing redundant 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 discover
- Caution: Not a "cure all" and can lose important info in some cases
  - How would you know if it is effective?
  - Pros and Cons of PCA vs Wrapper approach, example

Group Projects