CS 251 Data Analysis and Visualization, Spring 2017

Dr. Bruce A. Maxwell
Department of Computer Science
Colby College

Course Description

This course prepares students to apply computational approaches to data analysis and visualization to real information from a variety of disciplines and applications. Data visualization is the interactive visual exploration of information using 2-D and 3-D graphics using techniques that highlight patterns and relationships. Data analysis incorporates data management, data transformations, statistical analysis, data mining, and machine learning. Through programming projects, students will gain hands-on experience with the fundamentals of data analysis and visualization using data from active research projects at Colby and other institutions.

Prerequisites: CS 231

1 Introduction: Data

Scope of the course

Syllabus

Data: what is it, what forms does it take, what does it mean

- What can you measure? (don't give them the answers)
- For our purposes, data are digital values
- meta-data gives data meaning

Data Visualization: what is it?

- connecting data with our brains
- emphasizing existing connections in data
- enabling discovery of new patterns in data

Data Analysis: what is it?

- extracting knowledge from data
- using computers to identify patterns
- using computers to discover relationships

Data terminology

Data types

Data space

First assignment: find a data set and describe it in terms of the data terminology we discussed in class.

- dimension
- types
- max/min/range
- meta-data
- precision
- accuracy

2 Visualization and Coordinate Systems

Whey do we undertake data visualization

- To answer concrete questions about a given problem and enable rapid perceptual understanding
- To discover previously unknown facts about a problem

T-rex visualization

- What is putting the figure of the person in the drawing doing?
- It's providing a yardstick for the coordinate system

Coordinate systems

- coordinate systems are at the heart of visualization
- what coordinate systems exist when we try to visualize data?
  - Data coordinates
  - View Reference coordinates
  - Normalized view coordinates
  - Screen/Device coordinates

Geometric Tools

- vectors
- matrices
- homogeneous coordinates
- dot-product (scalar product): geometric interpretation as $\cos \theta$
- cross-product (vector product): geometric interpretation as $\sin \theta$

Coordinate Transformations

- rigid geometric transformations maintain the linear relationships between points
- scales increase or decrease the distance between points
- rotations rotate the world around an axis
- translations move the world

We can implement rigid geometric transformations using simple matrices...

3 Defining the View Pipeline

Defining a 2-D view

- Origin of the window in data space
- Dimensions of the window in the data space
- optional: orientation of the window
- dimensions of the screen
- origin of the screen (offsets)

The pipeline

- Subtract the origin of the window
- Scale by the inverse of the window size to get normalized view space
- Scale to screen coordinates, possibly inverting the Y-axis
- Translate by the offset, possibly by the number of rows (if inverting the Y-axis)

Defining an orthographic axis-oriented 3-D view

- Need to know where the view is: VRP
- Need to know where you are looking: VPN
- Need to know which direction is up: VUP
- Need to know the dimensions of the view volume
- Need to know the screen size and offsets

4 3D Viewing Pipelines

Defining an orthographic axis-oriented 3-D view

- Define the view reference point VRP on the edge of one face of the view volume
- Define the view plan normal VPN along one data axis
- Define the view up VUP vector perpendicular to the VPN
- Generate the coordinate system
- Subtract VRP
- Orient so the VPN is aligned with the Z axis: use axis alignment
- Translate to put the origin at the corner
- Scale to the unit cube
- Drop the z-coordinate
- Transform to view space (negate and scale the two axes, then translate)

Do a couple of examples with actual data

Note: quiz on Friday, HW this evening

5 Viewing and User Input I

Homework Review

- How to setup a 2D view
- How to setup a 3D view
- Review of axis alignment matrix
- Normalizing a vector
- Process of building an orthonormal coordinate system

Quiz

6 Data / Numpy Basics

What do we want to be able to do with a Data class?

- read data from a file
- write data to a file
- access one or more columns of data
- access one or more rows of data
- access individual data items
- add new data (columns or rows) to the Data object

Data is stored in files

- ASCII representation
- CSV files (show example)
- XML files (show example)
- Excel files
- Raw/binary files

Reading data: need to know something about the format

Storing data: need to have an internal data structure to hold the data

- The original data is a set of bits, probably want to keep those bits around, if possible. Why?
- The data will have an internal representation, probably want to make that easy to access. Why?
- Two bit representations, two purposes

7 Numpy-Fu

Class in the computer lab so people can explore numpy

Numpy matrix type

- Numpy has a base type for storing sequences of values: ndarray
- Numpy has a subclass of an ndarray for doing matrix math: matrix
- Create a matrix using Matlab style strings, lists of lists, numpy.random.rand(rows, cols)
- Can use mat or matrix as the class name
- A matrix is a 2-dimensional type, and any typical operation on a matrix returns a 2-D matrix
- Index into a matrix using 2 comma-separated values inside brackets: a[0, 1]
- Using a single index gives you a row, as a 2-D matrix
- Using slices lets you grab columns or subsets of a matrix, as a 2-D matrix
- A matrix will always be the most general type necessary to hold any element
- Multiplication and power are matrix multiplication and matrix power
- The optional dtype argument to the matrix constructor can take a data type as argument

transpose:	matrix.T	complex conjugate transpose:	matrix.H
inverse:	matrix.I	return as an ndarray object:	matrix.A
interpret input as a matrix:	asmatrix(a)	stack matrices a and b horizontally:	numpy.hstack((a, b))
convert to list:	a.tolist()		

Show lots of examples in Python, especially hstack

Show concatenation of operations

Show the flexibility of indexes, including conditional indexes (test out a bunch of examples)

```
a[ np.array(a[:,0].T )[0] > 3, :  ]
```

What is expensive in Python?

- for loops
- list accesses
- conversions of large amounts of data from one form to another

What is fast in Python?

- direct numpy operations

8 View Pipeline and User Input

Writing a summary

- Purpose: CS purpose and application, give it context
- Your solution, your results, this may be all someone reads of your work
- Balance concision and detail, be aware of special terms, can assume a practitioner of the art
- Strategies for writing a summary: write it first, then re-write it last, read it aloud

Review 3D View pipeline

- View reference coordinates: VRP, VPN, VUP, U
- Extent of the view volume in data space E_x, E_y, E_z
- Screen size and screen offset: (S_x, S_y) and (O_x, O_y)

$$vtm = T(S_x + O_x, S_y + O_y, 0)S(\frac{-S_x}{E_x}, \frac{-S_y}{E_y}, \frac{1.0}{E_z})T(0.5E_x, 0.5E_y, 0)R_{xyz}(\hat{U}, V\hat{U}P', V\hat{P}N)T(-VRP) \quad (1)$$

User Input

- Panning: modifies the VRP by moving it within the view plane
- Panning in 3D: could use a modifier like shift to move in and out
- Scaling: modifies the size of the view window
- Rotation: modifies the orientation of the VPN and VUP vectors.

9 Rotation / Number representations

Rotation Process:

- Move the center of rotation to the origin
- Align the axes
- Apply the rotation
- Reverse the axis alignment
- Move the center of rotation back to its original location

You want to apply this transformation to the view coordinate system: VRP, U, VUP, VPN
(**Review**: transforming vectors v. points)

Transforming Vectors v. Points

- A vector is a direction, it has no location associated with it
- A point is a location
- Use a 0 for the homogeneous coordinate of a vector: no translations
- Use a 1 for the homogeneous coordinate of a point

Transposing the R_{XYZ} matrix reverses the alignment rotation.

$$T(VRP+E_zVPN)R_{xyz}^{-1}(\hat{U},V\hat{U}P',V\hat{P}N)R_y(\sigma)R_x(\theta)R_{xyz}(\hat{U},V\hat{U}P',V\hat{P}N)T(-(VRP+E_zVPN)) \quad (2)$$

User input as a control law

- Controlling the strength of the response
- Adding inertia

Number Representations

- int: 4 bytes
- float: 4 bytes
- double: 8 bytes
- Can't represent more numbers than you have bytes, just distribute them differently
- Spacing on the number line is different, can cause issues with simple things like summation

10 Numerical Issues in Computation

Computing the sum of a set of numbers

- What are the problems that can arise?
- What is the number to which you can add 1 and get back the same number?

  In Python, an answer is no larger than 1.0e+16. (or 9.0075e+15)

- How do we avoid computations that add really big and really small numbers?

Numerical accuracy

- Wikipedia: Kahan summation algorithm (go through this in detail)

```
def KahanSum(data):
  sum = 0.0
  c = 0.0
  for value in data:
    y = value - c
    t = sum + y
    c = (t - sum) - y
    sum = t

  return sum
```

  Work an example for 1000.0, 3.142, and 2.611 keeping only one decimal of accuracy.

  Round 1
  sum = 1000.0, c = 0.0

  Round 2
  t = 1003.1, c = -.42, sum = t

  Round 3
  y = 2.653, t = 1005.8, c = 0.047

- Pairwise computation using a binary tree (error grows as square root of logN for random errors)

  Only costs twice as much in computation as doing the computation straight up, no extra memory cost necessary

- Radford Neal: exact computation of sums and means (uses a really big accumulator and calculates exact sums)
- John D. Cook: Accurately computing online variance (online v. static computations),

Quiz

11 Normalization and Higher Dimensions

Normalization

- Why do we need to normalize data?
  - Think about trying to do a rotating visualization if unit distances are different on different axes.
  - Normalization just means we have to be able to put the same yardstick on all axes: this is the coordinate system in which the view needs to be defined.
  - Sometimes we want to use the same normalization constant for all axes, sometimes we want independent normalization values.

    Example 1: income v. average education levels: different units, vastly different ranges, normalize individually.

    Example 2: height in inches for adults and height in inches for children: same units, different ranges, normalize together to comparative differences.

It is best to do whatever normalization of the data should occur prior to defining the view. If you can position the data set in the unit cube, it is easier to define the viewing parameters.

Displaying Higher Dimensions

How do we display more dimensions than three?

- color: good for both enumerated and ranges, but watch out for certain palettes
- size: have to be careful here, because increasing radius linearly is a squared increase in area

  People tend to be sensitive to area, though a single linear dimension is easier to comprehend
- texture: limited palette, but people are very good at identifying when textures are different
- icons/shape: limited palette, good for categories
- aural feedback: small number of bits
- haptic feedback: small number of bits

Colors: characterization and discrimination

- RGB: not great for representing color distances (there are other color spaces for that)

  Pretty reasonable for just displaying differences, with a limited color palette.
- HSV: cylinder representation

  Lots of options in this space for displaying patterns of variation
- Magnitude axes: R-G, Y-B, intensity, HUE, saturation axes
- CIE-LUV: designed so small distances are perceptually meaningful

No quiz this week, b/c gone next Wed and Friday

12 Visualization Parameters

Goals of the visualization process

- Correct characterization of the data
- Discriminability of relevant characteristics
- Stability and comparability across data sets

Range Selection: characterization, discriminability, and stability

- Setting range by max/min is not stable
- Characterization: a really big outlier makes the distribution look strange
- Discriminability: a big outlier means the bulk of the data gets compressed
- Stability: two data sets measuring the same process will have different ranges

Range selection/normalization alternatives

- (x - min) / (max - min) : guarantees complete visualization of the data
- range $= \mu \pm R\sigma$: let R control the range of the visualization
- range $= \mu \pm MAD$, $MAD = \frac{1}{N} \sum |x_i - \mu|$, MAD is the mean absolute distance, more stable than standard deviation
- Pick the range based on potential data values, or reasonable min/max values given what is being measured.
- Use one of the above methods, but also include bookend categories that include data beyond the visible range
- Use a sigmoid to map values to the range [0, 1]: useful for things like color

  $S(x) = \frac{1}{1+e^{-B(x-x_0)}}$
- Pick ranges based on desired visualization outcome: red is hot
- Use log scales to present information with high dynamic range.

Give the user power over the range selection.

Histograms: what are they? How do we design them?

- A histogram shows the frequency of each value in the data set
- A histogram where the sum of the buckets is 1.0 is a probability distribution function [PDF]
- Picking bin sizes is not trivial
- Aliasing in histograms creates problems
- Distributing weight among adjacent bins: noise dependent

13 Linear Regression

Thinking about models

- Models are compact representations of data
- In predictive models the dependent variable is a function of the independent variables: $y = f(\vec{x})$
- Representative models compactly approximate all of the data in a few parameters: $0 = f(\vec{x})$
- We usually have many more data points than are required to estimate the model
- Not all of the data points fit the model perfectly, so how do we judge different models?
- Most model estimation procedures assume Gaussian/normal noise distributions
  - Let the error be the sum of the squared error between each data point and the model
  - It is possible to show, under Gaussian noise, that the result will be the best estimate

Starting Analysis: Regression

- predicts a dependent variable based on one or more independent variables
- minimizes the error in the dependent variable: the prediction error

Set up a basic x v. y linear regression

$$y = \begin{bmatrix} y_0 \\ y_1 \\ \cdots \\ y_{N-1} \end{bmatrix} = Ab = \begin{bmatrix} x_0 & 1 \\ x_1 & 1 \\ \cdots & \cdots \\ x_{N-1} & 1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} \tag{3}$$

In general, the system is over-constrained, so you have to find the least-squares solution: e.g. SVD

Linear regression does not mean you have to fit a line. It works for any situation where you can write y as a linear function of x. The following would fit a 2nd order polynomial of x to predict y. The polynomial would be $y = p_2x^2 + p_1x + p_0$.

$$y = \begin{bmatrix} y_0 \\ y_1 \\ \cdots \\ y_{N-1} \end{bmatrix} = Ab = \begin{bmatrix} x_0^2 & x_0 & 1 \\ x_1^2 & x_1 & 1 \\ \cdots & \cdots & \cdots \\ x_{N-1}^2 & x_{N-1} & 1 \end{bmatrix} \begin{bmatrix} p_2 \\ p_1 \\ p_0 \end{bmatrix} \tag{4}$$

- There are C-1 degrees of freedom in the model, where C is the number of coefficients. N is the number of data points. There are N-C degrees of freedom in the error
- What is the error of the prediction? Sum-squared error per data point?
- $r^2 = 1 - \sigma_{error}/\sigma_y$ tells you the error of the output relative to the variation in the output

$$E = y - Ab \tag{5}$$

$$SSE = E^tE/dof_e \tag{6}$$

14 Multiple Linear Regression, the Covariance Matrix

Quick review of linear regression

- What is linear about the process?
- How do we do multiple linear regression, which means multiple independent variables?

Error of the prediction?

$$E = y - Ab \tag{7}$$

Sum squared error per data point.

$$SSE = E^t E / dof_e \tag{8}$$

The standard error makes use of the covariance matrix $A^t A$ and the SSE. It is the square root of the diagonals of $SSE(A^t A)^{-1}$. Note that this is essentially dividing the error of the fit by the standard deviation in each dependent dimension. More variation with the same error means a lower standard error.

$$stderr = \sqrt{diag(SSE(A^t A)^{-1})} \tag{9}$$

The utility/importance of the fit, $r^2 = 1 - \sigma_{error}/\sigma_y$.

The t-statistic is $t = b^t/stderr$. Use the student-t distribution CDF to compute the probability of the coefficient being non-zero, which implies a non-random influence on the dependent variable.

Covariance Matrix

Multiple linear regression does not really tell us about relationships between independent variables. If there are two highly related independent variables, then there are numerous possible functions that will provide a similar model fit. Random noise in the dependent data set will cause the process to lock on to one particular relationship.

- How do we discover relationships in the independent variables, or in any data set?

Guest lecture Wed, quiz on Friday

15 Guest Lecture

Kara Kugelmeyer: visualization of data

16 Quiz

Quix

17 Covariance and PCA

Covariance Matrix

Matrix computation process, given A, an N x M matrix with $N \geq M$

- Compute means of the columns $\vec{\mu}$, a 1 x M matrix
- Compute the differential $D = A - \vec{\mu}$
- Compute $D^T D$ to get an M x M matrix
- $D^T D$ is the covariance matrix except for the scaling by $\frac{1}{N-1}$

The covariance matrix tells us how features correlate. In particular, the off-diagonals give us information about the relationships between variables.

- The eigenvectors and eigenvalues of the covariance matrix give us a new coordinate system
- Transforming to the eigenvector coordinate system minimizes the dependence of the data dimensions
- The eigenvalues tell us the relative importance of each eigenvector in explaining variation in the data
- Variation is not necessarily signal, it just tells us how best to represent the information in the data

PCA: Principal Components Analysis

- Compute the covariance matrix of N variables
- Compute the N eigenvectors and eigenvalues
- Project the data onto the N eigenvectors
- To reduce dimensionality, keep the projection onto the $M \leq N$ eigenvectors with the largest values

Optional: whiten the data prior to computing the covariance matrix by subtracting the mean and dividing by the standard deviation

Motivation: for data sets with many features, features are often closely related. PCA tells us how many independent dimensions actually exist in the data.

Example: images have lots of redundancy that can be captured in just a few dimensions.

18 Eigenvalues and Eigenvectors

PCA Process

- If the data is heterogenous, whiten the data $x_i' = \frac{x_i - \mu}{\sigma}$
- Calculate the covariance matrix
- Calculate the eigenvectors and eigenvalues of the covariance matrix
- Optionally reduce the number of eigenvectors
- Project the data onto the eigenvector coordinate system

How many dimensions do we want to keep?

- The relative size of the eigenvalues indicates the relative variation along each eigenvector
- Look at a plot of the cumulative fraction of the eigenvalue values
- Select a criterion like explaining 90% of the variation in the data set
- Select dimensions until the cumulative fraction goes above the criterion

Introduction to SVD, alternative method of calculating eigenvectors

- SVD is a method of decomposing a matrix
- SVD works no matter the shape of the matrix or if its singular
- SVD is numerically stable
- SVD breaks down a matrix as $A = UWV^t$
- Columns of U are the eigenvectors of the columns of A
- Rows of V^t are the eigenvectors of the rows of A
- The diagonal values of W (diagonal matrix) are the singular values of A
- The singular values are related to the eigenvalues $w_i = \sqrt{\lambda_i}$

What do the eigenvectors tell us?

- Each eigenvector represents an independent dimension of the data
- Look at the coefficients of the eigenvectors.
- The large positive coefficients show a strong positive correlation
- The large negative coefficients show a strong inverse correlation
- Coefficients close to zero indicate low correlation
- Can name an eigenvector based on the largest magnitude coefficients

Plotting in eigenspace

- No different than selecting any other three axes
- The information should show independence between the eigenvectors

19 Review and Quiz

HW Review

- Analyzing a covariance matrix
- What eigenvectors tell you
- Projecting data into eigenspace

Quiz

20 Noise

Sensing: standard model of sensing

- plant
- sensor system
- noise

Decibel system of describing noise

Natural noise sources

- Johnson noise: thermodynamic, things that aren't absolute zero have motion, uniform spectrum
- Shot noise: quantization of charge, more relevant with small numbers, uniform spectrum
- Flicker or 1/f noise: uncertainty principle, things happen, worse things happen less often

The general noise spectrum

How to measure noise: hold the plant constant, if possible

How to reduce noise

- better designed measurement apparatus
- take multiple measurements: works well for Gaussian noise
- use a carrier signal

Inherent noise is important, because it tells us when we can reliably differentiate two measurements

Variation versus noise: variation is signal we might want to measure.

Next topic: clustering

Start with ISODATA: 2 clusters

- Pick two data values as the initial means
- Expectation: classify each point according to its closest mean
- Maximization: calculate new cluster means given the tags
- Terminate either after a fixed number of iterations or when the means don't change much

21 K-means Clustering

Identifying K natural clusters in data

K-means: K-clusters

- Go through algorithm
- Initialize using randomly chosen existing points
- Initialize using the mean value + offsets
- Run the algorithm multiple times with different starting points
- Representation error of the cluster means

K-means Key Design Decisions

- Picking initial clusters
- Handling clusters with no members
- Maximum number of iterations: why will it not always go to zero?
- How many times to re-run K-means?
- How many clusters K?
- Distance metric

Vector Quantization [VQ] and Codebook generation

- binary splitting method of generating large numbers of clusters with good distributions

Online Clustering

- Key parameter is the maximum distance to an existing cluster
- Useful when you have exceedingly large amounts of data

Hierarchical Clustering

- Average distance: recalculate distances after each merge
- Min/Max distance: don't need to recalculate distances, just keep careful records

Clustering after PCA: in high dimensional covarying spaces, distances have less meaning

22 How Many Clusters?

A final note on clustering: there are many other clustering methods

Preprocessing before clustering: clustering in high dimensions tends not to work well, as all points seem the same distance away from each other.

What is the right number of clusters?

- Balance representation error and model complexity
- Look for the point where the increase in model complexity is greater than the reduction in error

Rissanen's Minimum Description Length

- Term 1: $-\log_2 P(x^n|\Theta)$ the log probability of the data given the model
- Term 2: $\frac{Dk}{2}\log_2 n$ the number of clusters times the dimension of the data times the log of the number of points (D ends up as a multiplier in both terms, so we leave it out)

$$DL = -\log_2 P(x^n|\Theta) + \frac{k}{2}\log_2 n = SSE + \frac{k}{2}\log_2 n \tag{10}$$

Krzanowski and Lai, "A Criterion for Determining the Number of Groups in a Data Set Using Sum-of Squares Clustering", Biometrics 44, 23-34, March 1988.

The measure is based on how the representation error (SSE) changes as the number of clusters increases. It defines the DIFF(k) function as

$$\text{DIFF}(k) = (k-1)^{\frac{2}{p}} SSE(k-1) - k^{\frac{2}{p}} SSE(k) \tag{11}$$

$$\text{K\&L}(k) = \left| \frac{\text{DIFF}(k)}{\text{DIFF}(k+1)} \right| \tag{12}$$

where k is the number of clusters, p is the number of variables, and SSE(k) is the SSE with k clusters. The goal is to maximize this value.

Ray and Turi, "Determination of Number of Clusters in K-Means Clustering and Applications in Colour Image Segmentation", 2000

The logic of Ray and Turi is to find the ratio between the intra-cluster distances–the average distance of points to their associated cluster mean–and the inter-cluster distances.

$$\text{intra} = \frac{1}{N}\sum_{i=0}^{K-1}\sum_{x\in C_i} ||\vec{x} - \vec{z}_i||^2 \tag{13}$$

$$\text{inter} = \min_{\substack{i=0,\ldots,K-2 \\ j=i+1,\ldots,K-1}} (||z_i - x_j||^2) \tag{14}$$

$$RL = \frac{\text{intra}}{\text{inter}} \tag{15}$$

23 Distance Metrics

Triangle Inequality for a metric space.

$$D(\vec{x}, \vec{z}) \leq d(\vec{x}, \vec{y}) + d(\vec{y}, \vec{z}) \tag{16}$$

We also like distance metrics to be commutative so that $d(a, b) = d(b, a)$.

Mean absolute distance (also called L1 distance): not as sensitive to outliers as higher order distances.

$$D_{\text{L1}}(\vec{x}, \vec{y}) = \sum_{i=1}^{N} |x_i - y_i| \tag{17}$$

Normalized mean absolute distance: takes into account the variances in each dimension

$$D_{\hat{\text{L1}}}(\vec{x}, \vec{y}) = \sum_{i=1}^{N} \frac{|x_i - y_i|}{\sigma_i} \tag{18}$$

Euclidean distance (also called L2 distance): optimal distance measure for Gaussian noise.

$$D_{\text{L2}}(\vec{x}, \vec{y}) = \sum_{i=1}^{N} (x_i - y_i)^2 \tag{19}$$

Normalized Euclidean distance: takes into account the variances of the different dimensions.

$$D_{\hat{\text{L2}}}(\vec{x}, \vec{y}) = \sum_{i=1}^{N} \frac{(x_i - y_i)^2}{\sigma_i^2} \tag{20}$$

Mahalanobis distance: takes into account the covariances of the different dimensions.

$$D_{\text{Maha}}(\vec{x}, \vec{y}) = (\vec{x} - \vec{y})^T \Sigma^{-1} (\vec{x} - \vec{y}) \tag{21}$$

Hausdorff distance: defined as the maximum distance between corresponding elements of a set.

The family of distance metrics like mean absolute distance and Euclidean distance are called LN metrics. Mean absolute distance is L1, Euclidean is L2. Hausdorff distance is an $L\infty$ metric.

Cosine distance/angular distance

$$d_{cos}(\vec{x}, \vec{y}) = \frac{\vec{x} \cdot \vec{y}}{||\vec{x}||||\vec{y}||} \tag{22}$$

Binary distance: returns the number of similar features in two vectors. Useful for heterogeneous types.

24 More Distances and Fuzzy C-Means

More Distance Metrics

Intersection distance: defined as the sum of the minimum value of corresponding elements. Useful for matching histograms.

Earth Mover's Distance: defined as the minimum amount of change required to convert one set into another. EMD is often used as a robust distance measure for histograms.

Edit Distance: defined on strings as the number of operations required to transform one string into another. It is similar to EMD. There are a number of different specific edit distance metrics that depend on the types of editing operations allowed: replace, delete, insert, transpose, skip, etc.. The most commonly used edit distance is Levenshtein distance, which allows insertion, deletion, or substitution of a single character and counts the number of operations.

Dynamic Time Warping: allow for substitutions, deletions, additions between two time sequences

$$D(x_i, y_i) = \gamma(x_i, y_i) + \min_{x',y'} \left(D(x', y') + \zeta((x', y'), (x_i, y_i)) \right) \tag{23}$$

$\gamma(x_i, y_i)$ is the cost of matching the the ith element of x and y.

$\zeta((x', y'), (x_i, y_i))$ is the cost of moving from a prior element (x', y') to element (x_i, y_i).

$D(x', y')$ is the cost of the path from the start to element (x', y').

Correlation distance: a normalized Pearson's Correlation Coefficient applied to corresponding elements of the two vectors.

$$d_r(\vec{x}, \vec{y}) = 1 - \frac{\sum_{i=1}^{N}(x_i - \hat{x})(y_i - \hat{y})}{\sqrt{\sum_{i=1}^{N}(x_i - \hat{x})^2}\sqrt{\sum_{i=1}^{N}(y_i - \hat{y})^2}} = 1 - \frac{\sum x_i y_i - n\hat{x}\hat{y}}{\sqrt{\sum x_i^2 - n\hat{x}^2}\sqrt{\sum y_i^2 - n\hat{y}^2}} \tag{24}$$

Fuzzy C-means

Fuzzy C-means is a non discrete version of K-means clustering.

- Each data point has a membership weight in each cluster, based on some distance metric.
- 1/distance or else a Gaussian distribution of a given standard deviation could be used
- The new cluster means after each iteration are a weight sum of the points, weighted by membership

Fuzzy C-means is a good example of an E-M algorithm (Expectation-Maximization)

You can use fuzzy C-means to identify points that do not fit the cluster model very well.

25 Quiz

26 Pattern Recognition, NN, K-NN

Fundamentals of Pattern Recognition

- Pattern recognition is one method of converting data into knowledge.
- The basic question in pattern recognition is to identify when a new object is sufficiently similar to a previously seen pattern.
- The computer is subdividing the world of data into a set of classes.
- In most cases, the classes have some semantic meaning, such as an object or a particular person's face.
- Another interpretation is that pattern recognition is a method of prediction
- If the computer finds a certain set of features, then it predicts, or hypothesizes the existence of a semantic entity.

The most common process for building a pattern recognition system is as follows:

1. Extract features from the raw data.
2. Build a training set of example features from each class we wish to recognize.
3. Build a classifier using the training data.
4. Evaluate the classifier on a test set of labeled data not included in the training set.

Pattern recognition and machine learning algorithms require data. Data Issues:

- Not enough data given a complex classifier: classifier overtrains and learns just the training data
- Classifier learns the wrong function because of bias in the training data
- Training data does not cover all aspects of the possible input space

No Free Lunch Theorem: all classifiers are equally useful over all problems

Ugly Duckling Theorem: given any two patterns, there are an equal number of similarity predicates

Nearest Neighbor

- Training data is the classifier
- O(N) matching unless you have a really good data structure (then O(NlogN))
- Boundaries are planes and arcs
- Not all points are equally useful

K-Nearest Neighbor [KNN]

- Distance to a class is the sum of the distances to the K nearest neighbors in that class
- Allows complex boundaries

27 K-D Trees, Naïve Bayes

K-D Tree Data Structure

- A binary tree structure where each node is a data point
- Each node splits the space in half along one dimension

Nearest Neighbor Search

```
Given: a legal KD tree (kd), a target, a hyperrectangle, max distance

def nearestNeighbor( kd, target, hr, max-dist-sq )
  if kd is empty
      return NULL, inf
  Use the the kd.point split field (s) to cut hr into left-hr and right-hr
  if target(s) <= node(s)  // target is in the left
      nearer-kd := kd.left; nearer-hr := kd.left-hr
      further-kd := kd.right; further-hr := kd.right-hr
  else // target is in the right
      nearer-kd := kd.right; nearer-hr := kd.right-hr
      further-kd := kd.left; further-hr := kd.left-hr
  nearest, dist-sq := nearestNeighbor( nearer-kd, target, nearer-hr, max-dist-sq )
  max-dist-sq := dist-sq < max-dist-sq ? dist-sq : max-dist-sq
  if some part of further-hr is within sqrt(max-dist-sq) of the target
      if (kd.point-target)^2 < dist-sqd // test root node
          nearest := kd.point
          dist-sq := (pivot - target)^2
          max-dist-sq := dist-sq
      tmp-nearest, tmp-dist-sq := nearestNeighbor(further-kd, target,
                                                  further-hr, max-dist-sq)
      if tmp-dist-sq < dist-sq
          nearest := tmp-nearest; dist-sq := tmp-dist-sq
  return nearest, tmp-dist-sq
```

When building a tree, choose dimensions with high range and points close to the middle of the range.

Naïve Bayes

Bayes rule:

$$P(A|\vec{B}) = \frac{P(\vec{B}|A)P(A)}{P(\vec{B})} = \frac{\text{Learned Probabilities} * \text{Priors}}{\text{Normalizing Factor}} \tag{25}$$

- For multi-dimensional data, assume independent features
- Calculate a PDF for each feature
  - For discrete/enumerated features, use a table based on training data
  - For continuous features use a histogram or a parametric model (e.g. Gaussian)
- Use the product of the PDFs for each feature to calculate $P(\vec{B}|A) = \prod P(B_i|A)$
- The dimensions after PCA are more independent

Example with discrete values / example with continuous numbers.

28 Decision Trees

Review Bayes w/Continuous Numbers represented using Gaussians

- Basic assumption is that the features of the input vector are independent

Decision Trees

- 20 questions, but more than yes/no is allowed
- Each leaf is an answer
- Each non-leaf node is a question
- Enumerated types can be a node with many branches
- Numeric types can be any number of predicates: thresholds, intervals, comparisons of features
- Decisions trees can handle missing explicitly or implicitly by splitting the weight of the instance

How do we pick questions?

- What is a property of the best possible question we could ask at each node? Half the world goes away
- What property do we want from the data heading into each branch? More specificity.

Entropy $E(x) = -p(x)log_2(p(x))$

29 Boston Marathon

Quiz

30 Entropy and Growing Decision Trees

Information Gain:

Data Set $X = \{(\vec{x}_i, c_i)\}$, Entropy $H(X) = \sum_{\text{Categories(X)}} -p(c_i)log_2(p(c_i))$

Information Gain $= H(X) - \sum_{\text{Branches}} f_i H(X_i)$, Fraction of the data set going to branch i is f_i.

The entropy of a single branch is the sum of the entropies of each output class.

- Two class example (uneven): $E([0.6, 0.4]) = -0.6\log_2(0.6) - 0.4\log_2(0.4) = 0.971$
- Two class example (even): $E([10, 10]) = 0.5\log_2(0.5) - 0.5\log_2(0.5) = 1.0$ (max possible for two categories)

The information content of the outgoing branches is their entropies multiplied by the probability of being taken, given the training data.

- Two class example, three outgoing branches:

$$\begin{aligned} \text{IC} &= \frac{6}{20}E([4,2]) + \frac{7}{20}E([5,2]) + \frac{7}{20}E([1,6]) \\ &= (6 * 0.918 + 7 * 0.863 + 7 * 0.591)/20 \\ &= 0.785 \end{aligned} \tag{26}$$

The information gain is the difference between the incoming and outgoing branches.

$$Gain = E(10,10) - \left[\frac{6}{20}E([4,2]) + \frac{7}{20}E([5,2]) + \frac{7}{20}E([1,6])\right] = 1.0 - 0.785 = 0.215 \tag{27}$$

J.45 Algorithm: Grow the tree in a greedy fashion, then prune

Build(Node, Training Data)

- Return if a base case: no data, all one output category, maximum depth node
- For all available features
  - identify the best threshold/test using entropy as a metric
  - calculate information gain
- Pick the feature with the largest information gain to add a node to the tree
- Recurse on each sub-branch, passing in only the training data associated with it

Prune(Node, Training Data)

- Return if a base case: leaf node
- For each child
  - Evaluate the maximum bound error rate for the child
- Calculate a weighted sum of the maximum bound error rates
- If the error rate of the node is less than the weighted maximum bound, replace the node with a leaf
- Recurse on each child

Can run pruning with tuning data instead of calculating confidence bounds

31 Regression Trees and DT Variations

Finish pruning

Regression Tree Concept

- Subdivide the space into pieces.
- The goal is to maximally reduce the standard deviation of the data instead of information gain
- Must be balanced with even data splitting
- Create a leaf node when the standard deviation is small compared to the original data
- Internal splitting nodes also maintain a regression equation for their data
- The regression at a node uses attributes not used on the path to that node.
- When calculating a value, blend with the value predicted by parents all the way back to the root.

Variation: Locally-weighted Linear Regression

- Same concept as a regression tree, but use a KNN approach.
- Given a new point, find the K nearest neighbors, where K is the number required to fit a model
- Fit the model to the K nearest neighbors and use the model to calculate the output value.

Stumps, Random Trees, and Forests

Stumps: depth 1 trees; evaluate all features and decisions points to find the best one

Stumps can be useful in **ensembles**, such as those generated by Adaboost

- Give each training sample a uniform weight
- Select the best simple classifier given weighted performance on the data set
- Add the classifier to the ensemble if it is better than random
- Re-weight the training samples so that incorrectly classified samples get more weight
- Repeat until the desired performance parameters are met or the ensemble has reached its max

Ensembles make a decision using weighted majority vote, weighted by performance on the training set.

Random Trees

- Given a large number of potential features $F = \{f_i\}$
- Handle termination conditions: not enough data, max depth, pure enough data
  - Select a random subset $G << F$.
  - Evaluate each element $g_i \in G$ to find the g_o with maximum information gain.
  - If the information gain is large enough, use it as the decision rule for the node
  - Design decision: select another subset if the information gain is not enough

Random Trees can also be used in ensembles (Forests) Kinect body part classifier works this way

32 ANNs

Neural networks are function approximators built from directed graphs.

- Each node in the graph computes a weighted regression of its inputs.
- If all of the node functions are linear, then the network can learn to compute linear functions.
- If the node function is non-linear, the network can learn to compute non-linear and disjoint functions.
- The functionality and complexity of a network is defined by its structure and its weights.
- Learning in neural networks means learning the best set of weights given the data and structure.

Typical nonlinear activation equation: $O_j = \Phi\left(\sum_{i=0}^{N} w_{ji} O_i\right)$, includes a bias node $O_0 = 1$

Sigmoid squashing function: $\Phi(x) = \frac{1}{1+e^{-x}}$

Rectified Linear Unit [RELU]: $\Phi(x) = \begin{cases} x < 0 & 0 \\ x \geq 0 & x \end{cases}$

Feed-forward networks: output is a function of the inputs; the network has no memory

Fully-connected network: all nodes in one layer are connected to all nodes in the next layer

Layers

- Input layer: structure determined by the nature of the problem
- Output layer: structure determined by the nature of the solution
- Hidden layer(s): one or more layers of nodes, determined by the complexity of the problem

Network design is an important factor in using ANNs

- How complex to make the network? Can it learn the task?
- Does the input capture all of the necessary components?
- How do we structure the output to make it easy to learn and easy to interpret?

How do we adjust the weights to train the network?

33 Writing / Quiz Review

Writing ability we want students to have:

Be able to integrate text, code, and figures into a comprehensive and smoothly flowing document so that the reader can understand the content better or faster than they would with text alone.

Questions:

- why would we use a code-like presentation of an algorithm?
- what is the best way to present an algorithm? Should we use actual code?
- is there ever a need to include actual code? yes, if you are talking about the language itself
- why do we include images or diagrams in a document?
- what is important about the image/diagram? how do we make sure the reader knows that?
- how do we tie the information in the image/diagram with the text?
- what is the right size for an image? what options do we have on a wiki/blog?

Expanded version of writing ability:

6.1 The figures and code of a document are selected and organized so their positioning and content support the claims or explanations of the text, making the document more clearly understood by the reader than with text alone.

6.2 The text, which may also include captions, provides descriptions of all figures, tables, code, or images (as specified by the assignment).

Guidelines:

- Does the text describe the content of the image?
- Does the text describe how the content was generated? (this will be relevant if the image is of a drawing that was created by a program or of data that was collected or analyzed by your code).
- Does the text describe any implications of the content? (e.g. if the content is results from a simulation, does it analyze the results)
- If the image contains data, are the data properly labeled?
- If the image contains a lot of parts (e.g. the screen dump of a lot of numbers), are the most important parts highlighted in some way? (I.e. is the image or graph too cluttered or does it tell a clear story?)

Homework Review

Quiz

34 Training ANNs

Backpropagation

- Feed forward pass to calculate the output of each node in the graph.
- Calculate the error between the network output and the target values.
- Using the derivative, calculate the weight changes that drive the error towards zero.
- Sum up the weight changes over all training samples
- Adjust the weights, using some learning constant
- Repeat many, many times

Like K-means, ANN training has a random starting point and is a gradient descent algorithm. Therefore, it is useful to train multiple networks.

Key Derivations

- Output error as sum-squared error
- Change in the weights should be proportional to the negative partial derivative
- Activation equation as a two-part function (dot-product and sigmoid)
- Chain rule to calculate the derivative

$$E = \frac{1}{N}\sum_{i=1}^{N}(O_i - T_i)^2 \tag{28}$$

$$\Delta w_{ji} \propto -\frac{\partial E}{\partial w_{ji}} \tag{29}$$

$$I_j = \sum_i w_{ji} o_i \tag{30}$$

$$o_j = \Phi(I_j) = \frac{1}{1 + e^{-I}} \tag{31}$$

$$\Phi'(x) = \Phi(x)(1 - \Phi(x)) \tag{32}$$

$$\Delta w_{ji} = \eta \delta_j o_i \tag{33}$$

$$\Delta w_{ji} = \eta (t_j - o_j)\Phi'(I_j) o_i \tag{34}$$

$$\Delta w_{ji} = \eta \left(\Phi'(I_j) \sum_{k=1}^{K} \delta_k w_{kj} \right) o_i \tag{35}$$

35 ANN Examples

ALVINN: Steering Direction from an Image

Network Design

- 30x32 input image: down-sampled greyscale image
- 4 hidden nodes: fully connected
- 30 output nodes: trained to emulate a Gaussian with a fixed σ
  - To obtain an output, fit a Gaussian to the output node values
  - Provided a measure of confidence in the output
  - Enabled more precise estimation of the desired direction

Training

- Watched a person drive for 30s to 2min.
- Insufficient coverage of the input space
- Created multiple synthetic images from each input image

Analysis

- Looked at the hidden unit activations to determine what was important
- In some cases, the network trained on poor features (e.g. guardrails)
- The four hidden nodes acted like eigenvectors: project/reproject

Face Detection

First real-time face detector with reasonable accuracy ($> 90\%$)

- Not a fully-connected network
- Hidden nodes connected to patches of the input image
- Hidden units are fully connected to one output node: face/not face
- Trained multiple networks (3) and used them as an ensemble
- Ensemble of 3 using voting achieved 89.5% detection with 1/430k false detections

Auto-encoding: force the network to learn a compressed representation of the input space

Time-Delay and Recurrent Neural Networks

- Speech recognition and video labeling are both signals of arbitrary length
- Recursive connections in a network give the network memory of the past
- Training can be done by unfolding the network and linking the weights

Convolutional/Deep Networks

36 Quiz: Review

HW Review

1. Network outputs

input: 0, 0	output: 0
input: 0, 1	output: 1
input: 1, 0	output: 1
input: 1, 1	output: 0
input: 0.75, 0.25	output: 0.5
input: 0.25, 0.75	output: 0.5
input: 0.50, 0.50	output: 0.0

2. Backpropagation training.

A. The Generalized Delta Rule indicating how to modify the weight leading from node i to node j is the following:

$\Delta w_{ij} = \eta \delta_j o_i$

What is the meaning of each term on the right?

> Δw_{ij} is the amount to change the weight of the connection from node i to j.
>
> η is the learning constant, generally a fraction much less than 1.
>
> δ_j is the delta function for node j, which describe the relationship of the input to node j to the output target.
>
> o_i is the output of node i.

B. Why does back-propagation training calculate the average Δw_{ij} over all, or some subset of the training patterns before modifying the network? Why not adjust the weights after each individual pattern?

> Adjusting the weights after each pattern will move the network in many different directions, including potentially in circles. Using small batches or the whole training set to estimate the weight change allows the small differences to be averaged out.

3. Why might it be a good idea to train multiple networks when trying to solve a particular problem?

> ANN training initializes the network with small random weights. As backpropagation is a gradient descent algorithm, it can end up in a local minimum of the weight space. Running the training multiple times creates the opportunity for the training to find a better solution.

4. For any given network, what determines the number of inputs and the number of outputs of the network?

> The particulars of the problem define the input and output form of the network. The input contains all of the independent information being used to predict the output. The output layer is determined by the function the network needs to learn and how you want to use the output of the network.

Quiz