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)

5 Viewing and User Input I

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

The perspective transformation

$$P_{\text{per}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{d} & 0 \end{bmatrix} \tag{1}$$

After multiplying the point by the perspective matrix, re-normalize the point.

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.

6 Viewing and User Input II / Data

User Input

- **Review** Panning, Scaling
- Rotation: modifies the orientation of the VPN and VUP vectors.

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)

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

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 Basics

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

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))

Show lots of examples in Python

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 to the Data object

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 Design and Review

Review homework

- Setting up the VRC in 2D and 3D
- Setting up the remaining parameters
- Work through details of the homework cases

9 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)
- 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),

10 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
- texture: limited palette
- icons/shape: limited palette, good for categories
- aural feedback: small number of bits
- haptic feedback: small number of bits

11 Visualization Parameters

Goals of the visualization process

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

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

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 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.

12 Finish up Visualization

Histograms

- Picking bin sizes
- Aliasing in histograms
- Distributing weight among adjacent bins: noise dependent
- Linear v. Gaussian weight distribution

1D histograms v. 2D histograms v. 3D histograms

- 1D bar charts or something that looks more continuous
- 2D graph using color to represent density, or present in 3D with height as density
- 3D histograms generally use color or size to represent density

Probability distribution function [PDF]: normalized histogram

Cumulative distribution function [CDF]: cumulative sum of the PDF

- Picking random numbers from an arbitrary distribution

Entropy of a histogram: $E = \sum_{i=1}^{N} p_i \log_2(p_i)$

Why are histograms important?

- Look at the mean of a 2-peak distribution
- What does std or variance mean in that context?
- are you seeing one signal, two signals, multiple signals?

Most standard statistical measures assume something close to a normal distribution

Gaussian distribution: 1-D, N-D

13 No class

14 Quiz

15 Linear Regression

Starting Analysis: Regression

- predicts a dependent variable based on one or more independent variables
- minimizes the error in the dependent variable; it is not the best fit to the data

Understanding the linear regression parameters and outputs

Set up a basic x v. y linear regression

$$y = \begin{bmatrix} y_0 \\ y_1 \\ \cdots \\ y_{N-1} \end{bmatrix} = Ab = \begin{bmatrix} x_0 & 1 \\ x_1 & 1 \\ \cdots & \cdots \\ x_{N-1} & 1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} \tag{2}$$

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

- Singular Value Decomposition [SVD] is a common method of computing the inverse

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 $y = p_2x^2 + p_1x + p_0$.

$$y = \begin{bmatrix} y_0 \\ y_1 \\ \cdots \\ y_{N-1} \end{bmatrix} = Ab = \begin{bmatrix} x_0^2 & x_0 & 1 \\ x_1^2 & x_1 & 1 \\ \cdots & \cdots & \cdots \\ x_{N-1}^2 & x_{N-1} & 1 \end{bmatrix} \begin{bmatrix} p_2 \\ p_1 \\ p_0 \end{bmatrix} \tag{3}$$

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?

$$E = y - Ab \tag{4}$$

$$SSE = E^tE/dof_e \tag{5}$$

16 Multiple Linear Regression, the Covariance Matrix

Quick review of linear regression

- What is linear about the process?
- How is the problem set up?
- How do we do multiple linear regression?

Error of the prediction?

$$E = y - Ab \tag{6}$$

Sum squared error per data point.

$$SSE = E^t E / dof_e \tag{7}$$

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.

$$stderr = \sqrt{diag(SSE(A^t A)^{-1})} \tag{8}$$

The quality 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?
- How do we discover relationships in any data set?

17 Covariance Matrix and PCA

Explicit computation equation

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 eigenvalues

Introduction to SVD, alternative method of calculating eigenvectors

- 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.

How many dimensions do we want to keep?

- 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

18 Eigenvalues and Eigenvectors

Review

- How many eigenvectors do we want to keep?
- Make a cumulative plot

What do the eigenvectors tell us?

- 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

Projecting the data onto the eigenvectors

- Subtract the mean from each data point
- Take the dot product with each eigenvector you want to use
- Store the projected points as a new data set with columns labeled as the eigenvectors

Plotting in eigenspace

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

Normalization prior to PCA

- PCA is a description of how the data set varies
- The variation is measured in a least-squared sense
- Absolute magnitude matters
- We really want all of the dimensions to have the same metric meaning

Options for normalization

- max/min normalization: sensitive to the particular data sample
- mean/std normalization: less sensitive to the particular data sample

What can you measure?

- distance v. a reference stick
- count
- voltage

Think about it: those are the only things you can measure.

19 Measuring Stuff and 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

20 Homework Review and Quiz

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

21 K-means Clustering

Quiz Review: projecting data into eigenspace

Identifying natural clusters in data

K-means: N-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

22 Clustering Part Deux

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

ISODATA: another name for the 2-cluster K-means algorithm

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

23 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

$$DL = -\log_2 P(x^n|\Theta) + \frac{k}{2}\log_2 n = SSE + \frac{k}{2}\log_2 n \tag{9}$$

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

$$\text{DIFF}(k) = (k-1)^{\frac{2}{p}} SSE(k-1) - k^{\frac{2}{p}} SSE(k) \tag{10}$$

$$\text{R\&L}(k) = \left| \frac{\text{DIFF}(k)}{\text{DIFF}(k+1)} \right| \tag{11}$$

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.

$$\text{intra} = \frac{1}{N}\sum_{i=0}^{K-1}\sum_{x\in C_i} ||\vec{x} - \vec{z}_i||^2 \tag{12}$$

$$\text{inter} = \min_{\substack{i=0,\ldots,K-2 \\ j=i+1,\ldots,K-1}} (||z_i - x_j||^2) \tag{13}$$

$$RL = \frac{\text{intra}}{\text{inter}} \tag{14}$$

24 Distance Metrics

Triangle Inequality for a metric space.

$$D(\vec{x}, \vec{z}) \leq d(\vec{x}, \vec{y}) + d(\vec{y}, \vec{z}) \tag{15}$$

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.

$$D_{\text{L1}}(\vec{x}, \vec{y}) = \sum_{i=1}^{N} |x_i - y_i| \tag{16}$$

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

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

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

$$D_{\text{L2}}(\vec{x}, \vec{y}) = \sum_{i=1}^{N} (x_i - y_i)^2 \tag{18}$$

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

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

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

$$D_{\text{Maha}}(\vec{x}, \vec{y}) = (\vec{x} - \vec{y})^T \Sigma^{-1} (\vec{x} - \vec{y}) \tag{20}$$

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.

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

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.

25 More Distances and Fuzzy C-Means

More Distance Metrics

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

$$d_r(\vec{x},\vec{y}) = 1 - \frac{\sum_{i=1}^{N}(x_i-\hat{x})(y_i-\hat{y})}{\sqrt{\sum_{i=1}^{N}(x_i-\hat{x})^2}\sqrt{\sum_{i=1}^{N}(y_i-\hat{y})^2}} = 1 - \frac{\sum x_i y_i - n\hat{x}\hat{y}}{\sqrt{\sum x_i^2 - n\hat{x}^2}\sqrt{\sum y_i^2 - n\hat{y}^2}} \quad (21)$$

- Cosine distance/angular distance

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

Converting Enums to something you can compare

- Could have a feature that is either 0 or 1 for each enumerated value (common)
- Could use a binary distance metric for that column of data (not as common)

Example: enumerated type $f_i = \{Red, Green, Blue\}$

Instead of one enumerated column, make three columns whose value is either 0 or 1.

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.

26 DTW / Quiz

Dynamic Time Warping [DTW]: matching time series with squish and stretch

There are often situations where we have two vectors of features that represent time series. For example, consider two samples of someone saying a word. The overall shape of the signal is the same, but the lengths of the individual elements of the pattern will generally be different, even if their ordering is identical. DTW lets us compare these two signals, allowed for time stretch.

- Initial concept: the optimal path matches up the first and last elements of the feature vector.
- Inductive concept: the cost of the path of length $N-1$ plus the cost of adding step N must be optimal if the cost of a path of length N is optimal.
- Think of finding the optimal match as a recursive problem.

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

$\gamma(x_i, y_i)$ 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').

The recursive way to think about DTW is that the optimal path ending in the upper right corner of the diagram is the minimum sum of the optimal path to any possible prior node plus the cost of getting to the last node. The recursion ends when you get to the first node, whose cost is simply the matching cost of the first elements of each vector.

27 Pattern Recognition, NN, KNN

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 predicates

Nearest Neighbor

- Training data is the classifier
- O(N) matching unless you have a really good data structure
- 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

28 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:

$$P(A|\vec{B}) = \frac{P(\vec{B}|A)P(A)}{P(\vec{B})} = \frac{\text{Learned Probabilities} * \text{Priors}}{\text{Normalizing Factor}} \tag{24}$$

- 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.

29 Decision Trees

Review Bayes w/Continuous Numbers represented using Gaussians

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

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

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

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

Building a Tree

- Greedy process: pick the feature/decision with the greatest information gain
- Any leaf with only one output doesn't need to be divided
- Generally pick a maximum depth to the tree
- When done, prune the tree

30 Boston Marathon

31 Decision Trees and Regression 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.

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

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
- K is the number of neighbors required to fit a model
- Fit the model to the K nearest neighbors and use the model to calculate the output value.

32 HW Review and Quiz

Homework/Decision Tree Review

Quiz

Stumps, Random Trees, and Forests
(Covered Monday)

The computational cost of determining optimal thresholds and selecting optimal features at each node make building large decision trees costly to build. Furthermore, given that the process is a greedy process, there is no guarantee of optimality in the resulting tree. A suboptimal branching may be necessary in order to obtain a more optimal global tree.

Stumps: depth 1 trees

- Evaluate all features and all decision points, pick the best one

Stumps are 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 through weighted majority vote. Votes are weighted by performance on the training set.

Random Trees

- Given a large number of potential features $F = \{f_i\}$
- At each node
- 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)

33 ANNs

Stumps, Random Trees, and Forests

- A stump is a 1-node tree.
- A random tree follows the greedy growing procedure, but looks at only M randomly selected features at each node, where M is much less than the N available features.
- Forests are ensembles of simpler classifiers.

Ensemble(AdaBoost)

- Start with each training example having equal weight
- After training/selecting the first classifier, modify the weights
- The training examples the first classifier missed get a higher weight
- Repeat the process until nothing improves or you reach some cutoff (performance, number of classifiers)

Neural networks are function approximators built from directed graphs.

- Each node in the graph computes a function 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, then 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.

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

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?

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

$$E = \frac{1}{N}\sum_{i=1}^{N}(O_i - T_i)^2 \tag{25}$$

$$\Delta w_{ji} \propto -\frac{\partial E}{\partial w_{ji}} \tag{26}$$

$$I_j = \sum_i w_{ji} o_i \tag{27}$$

$$o_j = \Phi(I_j) = \frac{1}{1 + e^{-I}} \tag{28}$$

$$\Phi'(x) = \Phi(x)(1 - \Phi(x)) \tag{29}$$

$$\Delta w_{ji} = \eta \delta_j o_i \tag{30}$$

$$\Delta w_{ji} = \eta (t_j - o_j)\Phi'(I_j) o_i \tag{31}$$

$$\Delta w_{ji} = \eta \left(\Phi'(I_j) \sum_{k=1}^{K} \delta_k w_{kj} \right) o_i \tag{32}$$

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

36 BLAST

Sequence analysis requires a distance metric for comparing strings.

- DNA sequences have four possible values: $\{A, G, C, T\}$.
- Unlike DTW, a symbol in one string cannot match multiple symbols in another
- Deletion: one or more bases are missing relative to the other string.
- Insertion: one or more bases have been inserted into one string relative to the other.
- Swaps: one base has been substituted for another.
- Each action has a specified cost, generally based on the chemistry
- The movement cost for each action is zero, all costs are based on the matches

Needleman-Wunsch algorithm

- Similarity matrix determines the cost of matching
- Insertions and deletions cost the same as the worst match
- The **F matrix** is the DTW cost matching matrix

$$F_{ij} = \max(F_{i-1,j-1} + S(A_i, B_j), F_{i,j-1} + d, F_{i-1,j} + d) \tag{33}$$

Smith-Waterman: similar but alignment stops at scores of zero

Basic Local Alignment Search Tool [BLAST]

Given: two strings A and B.

1. Discard low complexity regions of A.
2. Use matches between short, fixed-length sequences to identify which sequences occur commonly in A and B.
3. Identify where sequences in S' match exactly
4. Look for places in F with aligned matching sequences (same diagonal) and do ungapped extensions
5. For each good ungapped extension, do a gapped extension until the fit is bad
6. For each good gapped extension, do another gapped extension with a lower dropoff

The end result is generally a set of (e.g. 500) good matches between segments of A and B.

37 Quiz: Review

HW:

Question 1: 3 layer network

N1: Bias 0
(1, 1)
N2: Bias 0 N3: Bias 0
(-1, 1) (1, -1)
N4: Bias 1 N5 Bias 1

2A: Meaning of $\Delta w_{ij} = \eta \delta_j o_i$

- $\Delta w_i j$ is the amount to change the weight connecting nodes i and j
- η is the learning rate, or what fraction of the proposed weight change to actually apply
- δ_j is the relationship between the weight and the output
- o_i is the output of node i.

2B: Updating the network after each training example is done, but it is like walking down a hill by stepping in many different directions. You tend to go down the hill, but not in a straight line. Updating the network with the average after evaluating many training samples is more like stepping directly down the hill. The average of the suggested weight changes tends to produce a smoother/faster path to training.

2C: Momentum means that you remember the last weight change and apply some fraction of that change to the current update. So if the network was changing in a certain direction after the last weight change, it continues to move in that direction to some degree even if the next round of training suggests moving in a different direction. The goal is to keep the network moving through small local minima in the error surface.

3: A network will train to a local minimum using backpropagation training. Which local minimum it terminates at is determined by the initial starting location. Most of the time the network ends up in a good local minimum, just as K-means tends to end up in good clustering situations. However, this is not always the case. Training the network multiple times and taking the best performing result–as measured on a test set–gives you some confidence that the network did not end up in a bad place.

Quiz