# 2.1: Introduction to Classification

+ can be infinite number of "equally good" classification thresholds
+ check: *[visuals]*

# 2.2: Choosing a Classifier

+ costs of misclassifying must be considered (i.e. we may be willing to misclassify at a higher rate to reduce the chance of the opposite condition)
+ check: when the classification plane is horizontal in a 2-d plot, this indicates that the factor plotted on the vertical axis is most important
+ check: soft classifier must be used when it is impossible to create a perfectly separating hyperplane

# 2.3: Data Definitions

+ data types... e.g.. data points; attributes vs. features; responses vs. outcomes; structured data  vs. unstructured data (e.g. text);types of structured data, including quantitative (numbers with meaning), categorical (numbers without meaning) vs. binary (subset of categorical data), unrelated, time-series
+ check: *[examples]*

# 2.4: Support Vector Machines (SVM)

+ margin formulation
    + n number of data points,
    + m attributes,
    + x_i = ith attribute of nth data point,
    + y_j = response for data point j (1 if data point is positive and -1 otherwise)
    + line: (sum from i = 1 to m (a_i * x_i)) + a_0 = 0
    distance between two lines = a / (sqrt(sum over i (a_i) ^ 2))
    + therefore, hard separation objective: minimize across all a_j (sum from i = 1 to m (a_j ^ 2)) such that ((sum from i = 1 to m (a_i * x_ij)) + a_0) * y_j >= 1 for each data point j
    + objective interpretation = maximize the margin between separating lines OR minimize sum of squares of coefficients
+ error
    + error = max(0, 1 - ((sum from i = 1 to m (a_i * x_ij)) + a_0) * y_j)
    + correct = ((sum from i = 1 to m (a_i * x_ij)) + a_0) * y_j >= 1 (i.e. see the hard separation objective formulation)
    + wrong = ((sum from i = 1 to m (a_i * x_ij)) + a_0) * j_j - 1 < 0
    + therefore, total_error = sum from j = 1 to n (error)
+ final formula
    + minimize a combination between margin (which we want to maximize) and error (which we want to minimize) minimize across all a_j (total_error + gamma * sum from i = 1 to m (a_j ^ 2))
+ check: *[formula]*

# 2.5: SVM: What the Name Means

+ support vector = points that "hold up" the shape
+ support vector machine (model) automatically determines support vectors (points supporting shape on  parallel lines)
+ check: *[none]*

# 2.6: Advanced SVM

+ can use a multiplier m_j for error term (in a similar fashion to gamma for the coefficients term) to increase penalty of error
+ important to scale the data so that minimization of coefficients is robust
+ near-zero coefficients are probably not relevant
+ alternatives: kernels for non-linear classifiers; logistic regression for probabilities
+ check: *[formula]*

# 2.7: Scaling and Standardization

+ common to scale data to range of 0 to 1; x_ij_scaled = (x_ij - x_j_min) / (x_j_max - x_j_min)
+ also may standardize (where mean = 0 and sd = 1) -> x_ij_standardized = (x_ij - mu_j) / (sigma_j)
+ which method? shorthand: scale for data in a bounded range, standardize for models (e.g. PCA and clustering)
+ check: *[example of scaling]*

# 2.8: K-Nearest Neighbor Classification

+ determine the class of a new point by picking the k closes points to the new one and choosing the class to be the most common among these k "neighbors"
+ what is nearest? use a p-order distance; may consider adding weighting attributes by importance (which allows for unimportant attributes to be ignored)
+ how to choose k? different training/validation/testing sets
+ check: *[visual example]*

# 3.1: Introduction to Validation

+ real effect = real relationship between attributes and response
+ random effect = random, but looks like a real effect
+ purpose of model validation is to try to distinguish real and random effects
+ can't measure models' effectiveness on data it is trained on because model fit captures real AND random effects, and only the real effects are likely to manifest in other data
+ check: if we use the same data to fit a model as we do to estimate how goot it is, the model will appear to be better than it really is

# 3.2: Validation and Test Data Sets

+ split data into training set (larger) to fit model and validation set (smaller) to estimate effectiveness
+ when comparing two different types of models, need to have a third set (test set), so that training set is used to build and tune models, validation set is used to pick a single model, and test set is used to estimate the performance of the chosen model
+ if not choosing among more than one model, then test set and validation sets are the same (meaning that there are only two sets total)
+ check: *[basically same as previous lesson]*

# 3.3: Splitting Data

+ rule of thumb: 70-90% training, 10-30% test for one model; 50-70% training and 50-50 split for validation and test sets when comparing models
+ methods: random; rotation (e.g. 5 data point rotation sequence)
+ problems: both random and rotation splitting may have issues with time-series data (random may randomly assign an unrepresentative sample to each category, and rotation may introduce bias by allocating too many samples by nature of its sequence)
+ check: most of the data should be in the training set

# 3.4: Cross-Valdiation

+ overcomes the issue of "important" data being in only one set
+ common practice: k = 10
+ model choice? NONE; do not average coefficients across pslit; should instead train model again using all data for final estimate of coefficents
+ check: in k-fold cross-validation each part of the data is used k-1 times for training and 1 time for validation

# 4.1: Introduction to Clustering

+ clustering = grouping
+ check: *[visual clustering]*

# 4.2: Distance Norms

+ p-norm distance (or Minkowski distance) = (sum from i to n (abs(x_i - y_i) ^ p)) ^ (1 / p)

+ check: *[2-norm]*

# 4.3: K-means Clustering

+ formulation
    + x_ij = attribute j of data point i
    + y_ik = 1 if data point i is in cluster k and 0 if not
    + z_jk = coordinate j of cluster center k
    + minimize y and z over (sum of i (sum of k ( ( sum of j over (x_ij - z_jk) ^ 2)) ^ (1 / 2))) subject to sum for all k y_ik = 1
+ steps (expectation-maximization (EM))
    0) pick k cluster centers
    1) assign each data point to nearest cluster center (centroid)
    2) recalculate centroids
    3) repeat steps 1 and 2 until no changes
+ advantages: fast
+ check: k-means is a "heuristic" because it isn't guaranteed to get the "best" answer

# 4.4: Practical Details for K-Means

+ how to improve quality of output?
    1) run several times, choosing different initial cluster centers
    2) try different values of k and pick the number that fits the context
    3) compare total distance vs k relationship ("Elbow" diagram) and identify the point at which the marginal benefit of adding another cluster is not worth increasing k
+ check: *[elbow diagram]*

# 4.5: Clustering for Prediction

+ assign new point to nearest cluster
+ use Voronoi diagram to identify area associated with cluster and infer to which cluster a new point would be assigned
+ check: *[visual check]*

# 4.6: Supervised vs. Unsupervised Clustering

+ supervised learning (for clustering) = response is known for each data point; otherwise, it is unsupervised

+ check: *[clustering vs. classification]*

# 5.1: Data Preparation

+ scale of the data; outliers...

# 5.2: Outlier Detection

+ types
    + point = values are far from the rest (e.g. a point far from others in a scatter plot)
    + contextual = values isn't far from the rest overall, but is far from points nearby in time (e.g. a large deviation in a time-series, sinusoidal-like curve)
    + collective = value is missing in a range of points, but it is difficult to identify exactly where (e.g. a "missing spike" in a time-series)
+ how to detect? box-and-whisker plot for 1-D; fit a model and identify points where model predictions are most inaccurate
+ check: *[outlier graph]*

# 5.3: Dealing with Outliers

+ when to remove? confirmed that outlier is part of "bad data"
+ with bad data, may consider replacing with imputation
+ can create a logsitic regression model to estimate probability of outliers given certain conditions, and a second model to estimate the value of the replacment data point for the outlier
+ check: incorrectly-recorded data is a justifiable reason to remove an outlier

# 6.1: Introduction to Change Detection

+ check: hypotheses tests are often not sufficient for change detection because they have high threshold levels, which makes them slow to detect changes

# 6.2: CUSUM for Change Detection

+ check if S_t > T, where S_t = max(0, S_t-1 + (x_t - mu - C)), x_t = observed value at time t, mu = mean of x; when detecting a decrease instead of an increase, then the positions of x_t and mu switch in the formula
+ how to choose C and T? depends on trade off of costs of the resolution of the change to be detected and of signaling for action based on cumulative changes
+ check: with CUSUM, a higher T detects changes slower, and is less likely to falsely detect changes

# 7.1: Introduction to Exponential Smoothing

+ S_t = alpha * x_t + (1 - alpha) * S_t-1
+ alpha -> 0: lots of randomness, more weight placed on previous baseline S_t-1
+ initial condition: S_1 = x_1
+ check: alpha -> 1: less randomness, more "weight" placed on current observation x_t

# 7.2: Trends and Cyclic Effects

+ T_t: trend at time period t
+ S_t = alpha * x_t + (1 - alpha) * (S_t-1 + T_t-1)
+ T_t = beta * (S_t - S_t-1) + (1 - beta) * T_t-1
+ initial condition T_1 = 0
+ cycles: can be additive like trends, or multiplicative
+ multiplicative seasonality:
    + L = length of cyle
    + C_t = multiplicative seasonality factor for time t (inflate/deflate x_t value)
    + S_t = alpha * x_t / C_t-L + (1 - alpha) * (S_t-1 + T_t-1)
    + C_t = gamma * (x_t / S_t) + (1 - gamma) * C_t-L
    + C_1, ..., C_L = 1, no initial cyclic effect
+ check: multiplicative seasonality = seasonal effect is proportional to the baseline

# 7.3: Exponential Smoothing: What The Name Means

+ check: all past observations are considered when calculating S_t

# 7.4: Forecasting

+ best guess for prediciton: x_t+1 = S_t, for simple exponential smoothing
+ F_t = alpha * S_t + (1 - alpha) * S_t, so F_t+1 = S_t for simple exponential smoothing
+ F_t is calculated similarly for double and triple exponential smoothing (i.e. F_t = S_t + T_t and F_t = (S_t + T_t) * C_(t+1)-L for multiplicative seasonality where best estimate of C_(t+1)-L = C_t+1)
+ choose alpha, beta, and gamma with optimization (i.e. min((F_t - x_t) ^ 2)
+ check: exponential smoothing is best for short-term forecasting because its forecast is based primarily on the most recent data points

# 7.5: ARIMA

+ 3 parts:
    + Differences: epxonential smoothing basic equations; useful if data is stationary (i.e. mean,  variance, etc. are constant over time), but if not stationary, then need to apply differencing; D_1 = diff of consecutive x_t, D_2 = diff of diffs, etc.
    + Autoregressive: predicting current value based on previous time periods' values
        + order-p autoregressive model = go back p time periods; autoregression on differences = use p time periods of previous x_t to predict dth order diffs
    + Moving average: using previous errors epsilon_t as predictors, where epsilon_t = (x_t_hat - x_t)
        + order-q moving average = go pack q time periods
+ ARIMA(p, d, q) model: dth order D, pth order AR, qth order MA
+ shorthand: ARIMA(0,0,0) = white noise; ARIMA(0,1,0) = random walk; ARIMA(p,0,0) =

AR(p); ARIMA(0,0,q) = MA(q); ARIMA(0,1,1) = exponential smoothing
+ forecasting: ARIMA is better than exponential smoothing when the data is more stable and/or has less peaks/valleys/outliers
+ common practice: need 40 past data points
+ check: *[definition of autoregression]*

# 7.6: Generalized Auto Regressive Conditional Heteroscedasticity (GARCH)

+ formula is similar to that of ARIMA, but (1) uses variances and squared errors instead of observations and linear error returns, and (2) does not use differences of variances
+ check: GARCH estimates or forecasts variance

# 8.1: Introduction to Regression

+ best fit regression line minimizes sum of squared errors, defined by coefficient estimates a_0, a_1, etc.
+ check: regression would be used instead of a time series model when there are other predictors that affect the response (not just previous values of the response variable)

# 8.2: Maximum Likelihood and Information Criteria

+ likelihood = measure the probability density for any parameter set
+ Example
error ~ N(0, sigma ^ s), iid
observations: z_1, ..., z_n
model estimates: y_1, ..., y_n
MLE: set of parameters that minimze sum of squared errors (i.e. min(sum over i = 1 to n (z_i - y_i) ^ 2))
For regression, where m = number of predictors j, n = number of observations i
MLE: min(sum over i = 1 to n (z_i - (a_0 + sum over j = 1 to m (a_j *x_ij)) ^ 2))*
+ AIC = 2 *(m + 1) - 2 ln(L*), where L* = ML value; smaller AIC -> better fit; more useful if there are lots of points*
+ e ^ ((AIC_1 - AIC_2) / 2) -> % difference in model 1 and 2
+ BIC = m *ln(n) - 2* ln(L*); encourages models with fewer predictors; only applicable when n > m
+ shorthand: BIC difference > 10 -> smaller BIC model very likely better; 10 > BIC diff > 6 -> smaller BIC model likely better; 6 > BIC > 2 -> smaller BIC model smoewhat likely to be better
+ check: simpler models are often better because (1) they are easer to explain, (2) to undestand, and (3) are less likely to be overfit

# 8.3: Using Regression

+ regression is most useful for descriptive analytics (i.e. understanding relationships betweeen predictors and the response variable) and predictive analytics, but not so much for prescriptive analystics (i.e. determining the best course of action)
+ check: regression is not commonly used for prescriptive analytics

# 8.4: Causation vs. Correlation

+ check: regression should not be used to determine causation, only to model relationships

# 8.5: Transformation and Interactions

+ either response or predictor variables may be transformed in order to create a more linear relationship
+ check: a negative sign for an interaction term indicates a decrease in the overall estimate

# 8.6: Regression Output

+ p-value
    + estimates the probability that a coefficient = 0; use 0.05 as a basis for possible variable elimination, but understand that one might use different threshholds in order to be more/less conservative

+ gotchas: get small when data set is larger even if strength of relationship between predictor and response is not actually better; only represents a probability, which can always be wrong
+ other output that are related to p-values: confidence intervals, t-statistic (i.e. coefficient / standard error), coefficient value (which, if low in magnitude when multiplied by the attribute value, sort of "nullifies" the significance indicated by a low p-value), r-squared (i.e. estimate of how much variability model accounts for)
+ check: the r-squared value must be compared to that of other models to gain insight (i.e. a 0.2 r-squared can be very good, depending on the situation)

# 9.1: Box-Cox Transformation

+ Used to address assumption of normally distributed data
+ logarithmic transformation t(y) = ((y ^ gamma) - 1) / gamma
+ check: heterosecdasticity = variance is different in different ranges of the data

# 9.2: De-Trending

+ trend = increase/decrease of data over time
+ how? factor-by-factor one-dimensional regression
+ why? trend could mess up factor-based analysis
check: maybe de-trend before using time-series data in a regression model

# 9.3: Introduction to Principal Component Analysis (PCA)

+ useful for models with lots of dimensions; reduce the amount of data needed (useful if data isn't "complete"); eliminates collinearity; concentrating on only top handful of components can help reduce randomness (bicause first several components have higher signal-to-noise ratio)
+ identifies order of dimensions according to amount of "spread" in a given dimensional
+ check: PCA can eliminate correlation between dimensions and rank dimensions in likely order of importance

# 9.4: Using PCA

+ how?
    1) Scale the data to get matrix X: scale such that (1 / m) * sum_i (x_ij) = mu_j = 0 where x_ij is the jth factor of data point i
    2) Find all eigenvectors of X^T * X, where V = [V_1 V_2 ...] is the matrix of eigenvectors
    3) Find the principal componenets: PC1 = X * V_1, etc.; kth new factor value for ith data point = t_ik = sum_j_over_m x_ij * v_jk
+ can have non-linear kernels
+ interpretation in original factors: more math...
+ check: in a regression model using PCA the original attributes' implied regression coefficient is a linear combination of the PC's regression coefficients (which is equivalent to the inverse transformation)

# 9.5: Eigenvectors

+ v is a vector such that A * v = gamma * v, where v is an eigenvector of A and gamma is an eigenvalue of A such that det(A - gamma * I) = 0
+ given gamma, solve A * v = gamma * v to find v

# 10.1: Introduction to Classification and Regression Trees (CART)

+ trees are useful for regression, classification, AND decisions (not necessarily identical to classification)
+ decision tree: for scenario where separate, inter-dependent decsisions need to be made in a process
+ good if effects of factors are different in different combinations
+ regression for each leaf in tree
+ tree approach is useful for pinpointing exactly where a model can get better (via low r-squared values for certain leafs)
+ check: every leaf has a different model (with different predictors, coefficients, etc., or even different "forms" (i.e. knn, svm, etc.))

+ check: each leaf's model is tailored to its subset of data

# 10.2: Branching

+ method for branching: most common approach is consider each factor one at a time; for example, create a regression model at a given leaf, split on the predictor which would lead to the lowest total variance in the splits; stop splitting if the decrease in variance is greater than some minimum thresshold; then start going backwards and use the other split at each pair of splits to identify if error is actually improved by the branch; if branch does not improve error, then it is removed (i.e. "pruned")
+ common practice: stop splitting if leaf would have less than 5% of data
+ check: don't branch past 5% in order to avoid overfitting

# 10.3: Random Forests

+ bootsrapped version of trees where many different trees are made (randomly), meaning that "weaknesses" are given more weight than they might be given otherwise; nonetheless, this can improve the "average" error overall and avoid overfitting; on the other hand, explaining the prediction is much more difficult
+ randomness? one: each tree has different data points, becasue points are resampled WITH replacement; two: when branching, randomly choose subset of all predictors, and make choice of predictor from that subset
+ final tree: average for regression, most common response for classification
+ check: random forest is not good for interpretation purposes

# 10.4: Logisitic Regression (logistic vs. linear)

+ similarities: transformations of data; interaction terms; variable selection; can be used for forests
+ differences (logisitic): longer to calculate; no closed-form solution
+ model quality: r-squared (fraction of variance explained) for linear; "pseudo" r-squared for logistic
+ thresholding: logistic based on probabilities
+ ROC curve: 1 - specificity (x-axis) vs. sensitivity (y-axis)
+ AUC (a.k.a. concordance index) = probabilitiy that model give's data point for option A a higher response value than for option B; not perfect measure (major limitation is that it does not differentiate between cost_fn and cost_fp)
+ check: logisitic can be used when the response is a probability or is binary

# 10.5: Confusion Matrices

```
actual     | model
-----------------------
           | real | spam
-----------------------
     real | tp   | fn
     spam | fp   | fn
```
+ tp, fp, tn, fn definitions...
+ Guidelines: positive = model says yes; negative = model says no; true: model is right; false: model is wrong
+ Example
```
     actual     | model
-----------------------
           | real | spam
-----------------------
     real | 490  | 10
     spam | 100  | 400
```

```
% of email is spam = (100 + 400) / sum(...)
% of spam in inbox = 100 / (490 + 100)
% of real email lost = 10 / (490 + 10)
```

check: tn = email is not spam, model is correct

# 10.6: Situationally-Driven Comparison

+ Evaluating a model's quality: use confusion matrix.
+ Additionally, need costs
total_cost = sum((cnt_tp * cost_tp) + (cnt_fn * cost_fn) + ...)

+ Example
same numbers as before;
counts imply 50% are spam;
counts imply cnt_fn / (cnt_fp + cnt_fn) = (100 / (490 + 100)) = 17% are incorrectly classified as real;
costs: cost_tp = 0, cost_fn = 1, cost_fp = 0.04, cost_fn = 0;
so, total_cost = sum(490 * 0 ...) = 14;
now, if 40% fp or fn, then
total cost = sum((cnt_tp * (6/5) * cost_tp) + (cnt_fn * (4/5) * cost_fn) + ...);
so total_cost = sum((490 * (6/5) * 0) ...) = 15.2;

now, a new model has cnt_fn = 50 and cnt_fp = 50;
check: although it's more accurate (i.e. only (50 / (450 + 50)) = 10% are incorrectly classified as real), it "costs" more

# 10.7: Advanced Topics in Regression

+ Poisson regression: useful when response follows a Poisson distribution (e.g. arrivals to airport)
+ Regression splines (spline = function of polynomials that connect to each other);: fiti different functions to different parts of the data; multi-adaptive regression splines (MARS) is one implementation
+ Bayesian regression: uses Bayes' theorem to update initial estimates; most useful when there is not much data
+ k-nearest-neighbor (KNN) regression: useful when there is not estimate of prediction function; predict response as average response of k closes data points
+ check: [none]

# 11.1: Introduction to Variable Selection

+ factor-based models: classification, clustering, regression
+ why limit number of factors?
   1. avoid overfitting: especially problematic when # of factors is => # of data points; cauases model to fit to random effects
   2. simplicity:
       + also implies less data collection is required
       + reduces chances of identifying insignificant factors
       + easier to interpret and communicate results (i.e. for prescriptive analysis)
       + sometimes factors can be illegal to use (e.g. race, gender, religion, etc.), so must be careful not to use corresponding highly correlated factors; also, it can be difficult to prove that an overly complex model avoids using these factors
+ check: simpler models with fewer factors avoids (1) overfitting and (2) difficulty of interpretation
(not necessarily low prediction quality nor bias in the most important factors)

# 11.2: Models for Variable Selection

+ forward selection, backward elimination, and stepwise regression
   + "greedy": takes action without considering future options
   + criteria other than p-value can be used to determine factor importance (e.g. R^2, AIC, BIC)
+ lasso:
   + adds constraint to standard regression optimization objective of minimizing sum of scquared errors by adding a "budget" tau to use for sum of coefficients (i.e. adds constraint to minimize size of coefficients, which is similar to SVM)
   + data must be scaled beforehand
   + some coefficients may be calculated to be 0
   + how to choose tau? try different values and
       + consider number of variables
       + consider quality of model
+ ridge regression:
   + similar to lasso, but constrains on a combination of the absolute value of the coefficients and their squares
   + same issues regarding scaling and choosing parameters (in this case, tau and lambda)
+ ridge regression:

+ elastic net without the absolute value term
   + no variable selection, but can still lead to better predictive models
+ check: key difference between stepwise and lasso regression? lasso requires data to be scaled first

# 11.3: Choosing a Variable Selection Model

+ comparison:
   + good for initial analysis: forward selection, backward elimination, stepwise regression; stepwise regression is most common
   + slower but better: lasso, elastic net
+ regularized regression comparison:
   + lasso: some coefficients forced to 0 to simplify model (because penalty term is linear)
   + ridge: coefficients shrink toward 0 (because penalty term is quadratic) to reduce variance in estimate (but adds some bias, which has a tradeoff in prediction error)
+ elastic net:
   + advantages:
      + variable selection benefits of lasso
      + predictive benefits of ridge
   + disadvantages:
      + arbitrarily rules out some correlated variables like lasso (e.g. two highly correlated variables might have different "costs", and lasso might choose the one with the higher cost)
      + underertimates coefficients of very predictive variables like ridge
+ check: when two predictors are highly correlated, ridge (not lasso) regression will usually have non-zero coefficients for both

# 12.1: Introudction to Design of Experiments (DOE)

+ DOE is useful when we don't hvae the data and getting a full set of data is eithe rimpossible or would take too long
+ examples: different colors in banner ad; retailer's display of "related" products to consumer; "representative" sample of survey respondents
+ comparison and control: e.g. when comparing sales price of cars, need to control for color, age, type of car, etc.
+ blocking factor: a factor that could create variation (e.g. sports car instead of a family car)
+ check: [definition of control]

# 12.2: A/B Testing

+ examples: collect data regarding click per appearance for banner ads and perform hypothesis test to determine if one is better than the other (can do hypothesis testing "on the fly", (i.e. when significant difference is detected, stop testing alternatives and use better option(s)))
+ a/b testing: choosing between two alternatives
+ a/b testing requirements:
   + collect data quickly
   + data must be representative
   + amount of data is small compared to whole population
+ check: a/b testing is not a good model when collected data is not representative of the population for which we seek insight

# 12.3 Factorial Designs

+ like a/b testing, but with multiple factors
+ full factorial design example:
   + test every combination when ther are not too many (e.g. 2 fonts x 2 wording x 2 bagkrounds)
   + use ANOVA to determine importance of each factor
+ fractional factorial design:
   + test subset of combinations when there are many possible combinations (e.g. 7 factors with 3 choices each = 3^7 combinations)
   + create a balanced design where each choice and each pair of choices is tested the same number of times
   + use regression (possibly includeing interaction terms, i.e. so as to avoid similar choices for different factors that prove to be the best for their factor, but negatively influence one another when paired) to estimate the effect of each choice
+ check: [an example where factorial design is more appropriate than a/b testing because there are multiple factors]; note that a/b testing is still applicable when there is only one factor with more than 2 choices

# 12.4: Multi-Armed Bandits

+ useful when you want to continuously test at a rapid pace and want to "maximize" value by also promoting tests that prove to be most "successful"
+ tradeoff of more information vs. immediate value -> "exploration vs. exploitation"

+ exploration: focusing on getting more information (to determine results with more certainty)
   + explitation: focusing on getting more immediate value
+ theory: from k alternatives...
   + start with no information, assume equal probability of selecting each alternatives
   + repeatedly
        1) choose an alternative to test based on probability of each being best
        2) update probabilities
   + stop when best alternative is clear
+ parameters
   + number of tests between recalculating probabilities
   + method of updating probabilities -> do we assume there is an underlying distribution? if so, which?)
+ there are no real "shorthands", but it is better than running a fixed, large number of tests
+ check: [multi-armed bandit definition]

# 13.1: Introduction to Advanced Probability Distributions

+ why study these? simple approaches work better sometimes, and probability distribution can form backbone of simple models
+ examples...
+ check: [none]

# 13.2: Bernoulli, Binomial and Geometric Distributions

+ binomial examples: probability (constant) of people sending donations when charity asks donations from 1/12 of mailing list each month
+ geometric examples: number of interview until first job; number of hits until baseball bat breaks; number of good manufactured units before a defective one
+ geometric (and binomial) assumption: each Bernoulli trial is iid
+ can infer whether data is iid by comparing it to geometric distribution
+ check: binomial distribution is not a good model when estimating the number of days n in each month in which temperature is above a threshold (wiht probability p) because the results are not independent (e.g. days above threshold are likely to be "clumped" in the summer)
+ check: [geometric formula application]

# 13.3: Poisson, Exponential and Weibull Distributions

+ weibull extra info:
   + k < 1 -> when failure rate decreases (i.e. "worst things fail fast", like parts with defects)
   + k > 1 -> when failure rate increases (i.e. "things that wear out, like tires)
   + if k = 1 -> Weibull = exponential, where lambda = 1 / lambda
+ using software to determine if data fits a probability distribution:
   + input: set of data
   + output: fit of varying distributions and parameters
   + should be used cautionsly (e.g. if software finds that distribution is Weibull with k = 1.002, then it might be better to use exponential with k = 1
+ check: if the number of arrivals follows the exponential distribution, then the number of arrivals per unit time follows the Poisson distribution (and visa versa)
+ check: geometric distribution models how many tries it takes for something to happen, while the Weibull distribution models how long it takes

# 13.4: Q-Q Plots

+ usefulness?
   + for visualizing whether two distributions are about the same
   + for visualizing whether a data set is distributed similarly to a probability distribution
   + note that statistical tests can hide details, so visualization can be better (even if it is not really more quantitative)
+ when comparing a single data set to a probability distribution...
   + horizontal axis = data, vertical axis = theoretical values of percentiles
+ check: [visual example]

# 13.5: Queuing

+ example description:
   + autodialer automatically calls phone numbers
   + if the call is answered it is put into a queue
   + how many employees should we have?
      + based on how many people will answer the autodialer

        + based on duration of call once the employee is on the phone
    + calls are answered and added by a probability distribution
    + we have c number of employees
    + calls leave the system based on another probability distribution
+ example, simple:
    + call start is Poisson (lambda)
    + 1 employee
    + call end is Exponential (u) time
    + we can calculate:
        + expected fraction of time employee is busy
        + expected waiting time before talking to employee
        + expected number of calls waiting in queue
    + resulting equations:
        + arrival Rate (calls) = lambda
        + service Rate (calls) = u > lambda
        + transition Equations (>= 1 calls in queue)
            + P(next event is an arrival) = lambda / (lambda + u)
            + P(next event is a finished call) = u / (lambda + u)
        + can calculate:
            + Expected fraction of time employee is busy = lambda / u
            + Expected waiting time before talking to employee = lambda / u(u + lambda)
            + Expected number of calls waiting in queue = lambda^2 / (u(u + lambda))
+ example, more complex: more exmployees
    + all can be solved with closed form answers due to memoryless property of the Poisson and Exponential probability distributions
    + memoryless exponential: distribution of remaining call time = initial distribution of call time
    + memoryless Poisson: distribution of time to next arrival = initial distribution of time to next arrival
    + if data fits exponential/Poisson distribution, then it is memoryless, and visa versa

+ memoryless property example :
    + setting: should tire manufacturer pay damage for accident that happened at 10K miles?
    + probability (tire fails at 10K) = ?
        + tires are more likely to fail the more worn out they are, so this is not memoryless
        + cannot be modeled with the exponential distribution (possibly try the Weibull with k > 1)

+ potential queuing model parameters:
   + general arrival distribution (A)
   + general service distribution (S)
   + Number of servers (C)
   + size of the queue (K)
   + population size (N)
   + queuing discipline (D)
+ kendall notation
+ model extensions: potential "hang-ups", balking (i.e. leaving after seeing wait time), etc.
+ simulation is good for modeling complex scenerios
+ check: queuing is not approrpiate for estimating something not having to do with waiting in line

# 13.6: Simulation Basics

+ simulation? build a model and watch its behavior

+ types of simulation:
   + deterministic = same inputs give the same outputs (no randomness)
   + stochastic = use when system has randomness
+ continuous time simulations: changes happen continuously
+ discrete-event simulations: changes happen at discrete time points only
   + valuable when systems have high variability
   + using average values is not good enough
+ simulations software includes:
   + modeling elements:
      + entities: things that move through a simulation
      + modules: parts of process (e.g. queues)
      + actions
      + resources (e.g. workers)
      + decision points
      + statistical tracking
   + GUI
   + complexities "under the hood" (e.g. pseudo random number generation)

+ replications: number of runs of simulation
   + one replication = one data point (may be unrepresentative)
   + run multiple times to get distribution of outcomes
+ simulation validation with real data
   + real and simulated averages don't match -> problem
   + averages match, variances don't match -> problem

+ check: a stochastic simulations should be run many times because one random outcome might not be representative of system performance in the range of different situations that could arise
+ check: it is important to validate a simulations by comparing it to real data as much as possible because if the simulation isn'g a good reflection of reality, then any insights we gain from studying the simulation might not be applicable in reality

# 13.7: Prescriptive Simulation

+ prescriptive analytics describes how we use simulation for analytics
+ use automated heuristic optimization offered by simulation software
+ when making comparisings, need to be careful
   + a simple comparison of means may not be sufficient
   + if possible, compare performance (of different parameter values) on exactly same data point (i.e. use the same random numbers before each trial)
+ simulation can be a powerful tool:
   + model is only as good as quality of input
   + missing or incorrect information may lead to incorrect answers
+ example:
   + in a call center simulation, assuming that workers answer calls equally quickly (which is likely incorrect) can lead to costly bad decisions

+ check: in simulation, both (1) use automated optimization functions in simulation software to find good parameter values, and (2) vary parameters manually

# 13.8: Markov Chains (MCs)

+ markov chains = probability based models based on states of a system
+ formulation:
   + for each state i in the model:
      + pij = transition probability form state i to state j
      + p = [pij] is the transition matrix
+ example: p(sunny - rainy) = probability sunny today rainy tomorrow
   + what is long run of probability of rainy days?
   + given pi = [0.5, 0.25, 0.25] = probabilities of xyz today

   + pi * P = probabilities of xyz tomorrow
   + pi * P * P = probabilities of xyz the day after tomorrow
+ pi_star = "steady state" solution
   + doesn't always exist
   + can't have cyclic behavior
   + every state must be reachable from all others
+ key assumption: memoryless
   + state transitions only depend on the most recent state
   + most systems do not exhibit this property though, but we still study them because they are usful in trying to cnnect smaller amounts of information to find larger ones
+ example: Google's page ranks sytstem
   + more complex than a simple MC system, but MC is useful for understanding (i.e. we pages = states, chain = jumping from one page to another, steady state probability = rank of web pages)
+ other examples:
   + ubran sprawl, population dynamics, disease propagation
        + only memoryless in the short term, but that may be all that is needed
+ summary:
   + no so common due limiting assumption of memoryless system
   + nonetheless, powerful in some cases
+ check: [memoryless definition] (i.e. memoryless = next state of a process doesnt know much about the form of the underlying distribution the data comes from, or it doesns an edge between them then remove it; and if theres previous choice beats"
+ information levels:
   + perfect: know all information (e.g. chess)
   + imperfect: some mau have more information than others (i.e. not symmetric) (e.g. example in next video)
+ zero-sum games:
   + zero-sum: whatever one side gets, the other side loses (e.g. rock-paper-scissors)
   + non-zero-sum: total benefit might be higher or lower (e.g. economics)
+ summary:
   + how to determine best strategy? optimization
+ check: [example] a game theoretic model is appropriate when ... and it must model somethng as a function of the number of units its competitor produces

# 16.5a: Competitive Models Demo

+ setup:
   + two gas stations: bp vs. shell

+ can set price at either $2.50 or $2
    + if both set price equally, then demand is 50/50
    + otherwise, all demand goes to the lower-priced product
+ scenario 1: cost = $1/gallon
    + from shell's POV, best strategy is to sell at $2 no matter what bp does
    + lower prices is better for bp as well
+ stable equilibrium:
    + neither station has incentive to change
        + "prisoner's dilemma"
            + even if both sides agree to higher prices, both have incentive to cheat and lower price, so they do that
+ scenario 2: cost = #1.75/gallon
    + both are better off charging higher price
+ extension: consider arbitrary prices pshell and pbp
    + both keep lowering prices until price is about equal to the cost
+ check: [none]

# 18.1: Introduction to Power Company Case

+ context: power company want to shutoff powr fo customers who don't pay their bills
+ turn power off:
    + turn off for those not ever going to pay
    + not people who forgot or got behind
+ logisitical problems:
    + manually shut off
    + go to location
    + more work than the company can handle
+ considerations:
    + which shutoff should be done?
    + some worker's time is taken up by travel
    + how to identify "good" customers whose power should not be shutoff
    + prioritizing shutoffs

# 18.2 Models for Customer Identification

+ power customer identification:
    + customers who can pay, but aren't going to:
        + credit score
        + income
        + past history of defaults on paymebts to any company
        + past power-bill paymenbt history
        + sip code
        + value of home
        + rent or own
        + length of residency
        + marital status
        + number of residents
    + some factors may be illegal to use
        + race, sex, age, other demographic factors, or other factors highly correlated with these
+ types of models:
    + yes/no answer:
        + classification (svm or knn):
            + pay, can pay but do not pay, not able to pay
        + clustering
    + probability of payment: e.g. logisitic regression
    + single-model apporach or treee-based approaches
    + hybrid approaches: e.g. first cluster, then analyze each cluster separately
+ pros and cons of models:
    + unsupervisied approach:
        + clustering
            + quick
            + not exact cluster you might expect
    + supervisied approach:
        + classification:
            + clear decision
        + logisitic regression:
            + requires threshold
+ summary:
    + modeling is an art
    + not always a "right" approach, but there are wrong approaches

# 18.3 Models for Cost Estimation

+ power cost estimation: cost of leaving power on or off
    + for customers with long history...
        + given customer credit, financial, andpayment history data; and

possibly some demographci information
        + use exponential smoothing or ARIMA OR
        + to esimate the amount of power a customer will use in the next month
    + when considering variablity in usage
        + given [same]
        + use GARCH
        + to estimate the amount of variability in a customer's power usage next month
    + OR... for customers with shorter or longer history
        + given [same]
        + regression-based model (simple regression, tree-based, or clustering followed by regression)
        + to [same]
+ pros and cons of models:
    + time series:
        + good if enough past customer usage data exists
        + effective only for short-term forecasts
    + factor-based regression:
        + can be effective even when there's not much specific customer data
        + normalize to account for seasonality
+ hybrid approach: model payment and cost together
    + model the amount of money owed (zero if bill is paid)
    + however, this is not usually effective if there are many zero values, plua a range of others, so usually better to analyze separately

# 18.4 Models for Shutoff Selection

+ expected cost of leaving power on/off
    + E[cost of keeping power on] = p(no pay) * E[cost of power used next month) + (1 - p(no pay)) * 0
    + E[cost of shutting power off] = p(no pay) * cost_shutoff + (1 - p(no pay) * (cost_shutoff + cost_turnon)
+ data
    + past data on travel time/speed
        + other details of driving may be too complex/expensive to collect
    + time to shutoff power
    + estimate of future usage
    + probability of non-payment
+ optimization nmodels
        + question: highest-value set of customers for power shutoffs?
        + binary variable for each customer
            + objective function: cost difference between shutting off power or not, times the vinary variable
            + constraints are hard to write
        + clustering
            + cluster the physical locations
            + modify optimization model
        + simulation
            + variability in drive times, shut offtimes, power usage
            + distribution fitting
    + then use model to determine how many new workers to hire
+ acutal models used
    + customer identification: logitic regression
    + cost estimation: linear regression with Box-Cox transformation
    + shutoff selection: vehicle routing (optimization)