Review Questions with Solutions

Signal Data Science

Contents

Overview **4**

Linear regression **5**
Assumptions . . . 6
Residuals . . . 6
Cost functions . . . 7
Sources of concern . . . 8
Regularization . . . 10

Resampling **13**
Overfitting . . . 13
Cross-validation . . . 13
Bootstrapping . . . 14

Logistic regression **16**
Comparison to discriminant analysis . . . 16
Maximum likelihood estimation . . . 17
Visualization . . . 18
Multinomial logistic regression . . . 19

Principal component analysis **20**
Visualization . . . 21
Principal component regression . . . 21

Clustering **23**
K-means clustering . . . 24
Minimizing variance . . . 25
Gaussian mixture models . . . 26

Decision trees **28**
Bagging . . . 29
Random forests . . . 29
Gradient boosted trees . . . 31

Recommender systems **33**
Collaborative filtering . . . 34
Content-based filtering . . . 35
Hybrid methods . . . 36

Advanced topics **38**
k-Nearest Neighbors . . . 38
Support vector machines . . . 39
Natural language processing . . . 41
Parametric vs. nonparametric models . . . 42
Bias–variance tradeoff . . . 43

Numerical optimization 44
Miscellaneous 46

Problematic datasets **47**
Missing values 47
Unbalanced classes 47
Target leakage 48

Classical statistics **50**
Theorems 50
Frequentist inference 51
Bayesian inference 51
A/B testing 52
Distributions 53

Overview

Everything you need to know for entry-level data science job interviews with the following exceptions: algorithms, probability/logic questions, and product/metrics questions.

Questions are ordered within each subsection from less to more complicated.

Linear regression

- Is linear regression a supervised or unsupervised method?
  - Linear regression is a supervised method because there is a response variable which you want to predict.
- What does "least squares" refer to in the phrase "ordinary least squares regression"?
  - In ordinary least squares regression, we find the coefficients which minimize the sum of the squared residuals, hence "least squares".
- What does it mean for the coefficient of a feature to be equal to k in a linear regression model?
  - It means that an increase of 1 unit of that feature will on average lead to an increase of k units in the response variable, all else (including other predictor variables) being held equal.
- How can you use categorical variables as predictors in a linear regression model?
  - Categorical variables can be used as predictors by simply expanding out each categorical variable into a set of indicator variables, each of which takes on the values 1 or 0 based on which category a data point belongs to.
- If you have a categorical variable which can take on one of k values, how many indicator variables should be created from that categorical variable?
  - A categorical variable with k possible values should be expanded out to $k - 1$ indicator variables. The indicator variables can represent membership in categories 1 through $k - 1$, in which case membership in class k is represented by all of the indicator variables being equal to 0.
- What does R^2 represent?

  \$ The R^2 value of a linear model represents the percentage of variation in the response variable explained by the linear model.
- What is the difference between R^2 and adjusted R^2?
  - Adjusted R^2 penalizes the R^2 value for a more complex model (with more predictor variables) and is therefore a better estimate of how well the model will perform on unseen data (because it tries to account for the effect of overfitting).

Assumptions

- What is one assumption made by ordinary least squares regression?
  - See below for a variety of such assumptions.
- What assumption is made by ordinary least squares regression about the predictor variables?
  - The predictor variables are linearly independent, *i.e.*, no predictor variable can be perfectly written as a linear combination of the others. This can also be stated as the data matrix having full column rank.
- What assumption is made by ordinary least squares regression about the mean of the error terms?
  - The mean of the error terms is equal to 0.
- What assumption is made by ordinary least squares regression about the variance of the error terms?
  - All the error terms have equal variance.
- What assumption is made by ordinary least squares regression about the correlations between the error terms?
  - The error terms are uncorrelated with each other.
- What assumption is made by ordinary least squares regression about the correlations between the error terms and the predictor variables?
  - The error terms are uncorrelated with the predictor variables.
- What assumption is sometimes made by ordinary least squares regression about the distribution of the error terms?
  - The error terms follow a Normal distribution. This is not strictly necessary for ordinary least squares regression but allows us to mathematically establish a variety of nice properties about the results.

Residuals

- What is a residual?
  - A residual is the error of a prediction result, equal to the actual value minus the predicted result. Typically written as $y_i - \hat{y}_i$.
- After fitting a linear regression model, how can its residuals be calculated?
  - Use the linear model to make predictions for each data point in the training data and subtract each prediction from the corresponding true value of the response variable.

- How would you plot the residuals of a single-variable linear regression?
  - Plot the independent variable on the x-axis and plot the values of the residuals on the y-axis.
- What is one example of information you can get from plotting the residuals of a linear regression?
  - Looking at the residual plot can indicate that some of the assumptions of an ordinary least squares regression model have been violated.

    First, if the variance of the residuals vary significantly, that contradicts the assumptions of error terms having equal variance. For example, we might imagine that the residuals for $x < 0$ are very large but the residuals for $x \geq 0$ are very small; this would indicate that the $x < 0$ and $x \geq 0$ subsets of the data have errors with different variances. One way of dealing with this might be appropriate to add an indicator variable to your model that indicates whether or not $x < 0$. You might also want to take a log transform of the data or to use a nonlinear method.

    Second, if the residuals are correlated with each other, that contradicts the assumption of uncorrelated errors. For example, if we tried to fit a linear model to a stock price (with time as the independent variable), the errors would oscillate up and down—each residual would be correlated with the residual coming right before it. (A stock that goes up today is more likely to go up tomorrow, and a stock that goes down today is more likely to go down tomorrow.) The specific case where r_i is correlated with r_{i-k} for some $k > 0$ is called *autocorrelation*.

Cost functions

- What is a cost function?
  - A cost function is a function minimized by a machine learning algorithm. It describes how "bad" a set of parameters is.
- What is one example of a cost function?
  - Cost functions include the root mean square error (RMSE), the sume of squared errors (SSE), the mean absolute error (MAE), etc.
- What is the minimum value of these cost functions?
  - The minimum value of the above cost functions is 0.
- What is the maximum value of these cost functions?
  - The above cost functions have no maximum.
- What cost function is used for ordinary least squares regression?

  - Ordinary least squares regression uses the sum of squared errors (SSE) as its cost function.

- Will you get equivalent results using the RMSE and SSE cost functions?
  - Yes, because the RMSE and SSE are minimized at the same point. The RMSE increases when the SSE increases and the RMSE decreases when the SSE decreases.

- Will you get equivalent results using the SSE and MAE cost functions?
  - No. The SSE will penalize larger values more than the MAE—if the error at a point *doubles* from e to $2e$, its square increases by a factor of 4 ($e^2 to 4e^2$).

- Why might you use the MAE instead of the RMSE in a regression model?
  - You might use the MAE when you have some *a priori* reason to penalize errors by a cost proportional to the size of the error—when being off by 10 is just twice as bad as being off by 5.

- How could you modify the cost function to combat overfitting?
  - You can add on a regularization term, like the L^1 regularization term (which is a constant λ multiplied by the sum of absolute values of the model coefficients) or the L^2 regularization term.

- What is an example of a situation when you might want to give different points in your training data different weights (representing their importance to the model)?
  - Data might be collected with different amounts of error, so you might want to weight points by the *inverse* of their error such that more precise measurements are given more significance by the model.

- How would you assign points in your training data non-uniform weights in an ordinary least squares regression?
  - Instead of minimizing the sum of squared errors, minimize the sum of squared errors multiplied by the corresponding weights. That is, instead of minimizing $e_1^2 + e_2^2 + \cdots + e_n^2$, minimize $w_1 e_1^2 + w_2 e_2^2 + \cdots + w_n e_n^2$.

Sources of concern

- What is the difference between collinearity and multicollinearity?
  - Collinearity refers to a linear association between two predictor variables, like $x_1 = a + bx_2$. Two variables are perfectly collinear if such a relationship holds perfectly and imperfectly collinear if such a relationship is approximately true. Multicollinearity refers to the same

phenomenon among multiple predictor variables, where one can be written (either perfectly or approximately) as a linear function of the others.

- What is one example of a real-life situation where you might have perfect multicollinearity?
  - One example of perfect multicollinearity is when you measure the same quantity in different unit systems, like recording average temperature in both Fahrenheit and Celsius or average height in both feet and meters.
- What is one example of a real-life situation where you might have imperfect multicollinearity?
  - One example of imperfect multicollinearity is when multiple variables are proxies for some underlying trait. For example, the SAT is a college admissions examination in the United States with three different subtests, labeled Writing, Critical Reading, and Mathematics; if one records scores on all three subtests, there will likely be a nontrivial amount of multicollinearity between the three. (In such a case, one might want to look at the average subtest score instead or, for more complex analyses, perform a 1-factor factor analysis on the three subtests and use the factor scores as a predictor variable.)
- If there is significant multicollinearity between the predictor variables, what do you expect to happen to the coefficient estimates of a linear model as you add and remove predictor variables?
  - In a situation with significant multicollinearity, the coefficient estimates will be very unstable—they are liable to undergo significant changes upon the addition or removal of a single predictor variable. This is on account of the exact form of the linear relationship between the multicollinear variables changing rapidly as you add and remove more correlated variables.
- How might significant multicollinearity make it more difficult to interpret the results of a linear regression?
  - The standard interpretation of a coefficient estimate is as the expected or average change in the response variable upon a one-unit increase in the predictor variable if all other predictor variables are held equal. However, in the presence of significant multicollinearity, the "held equal" part of that interpretation doesn't hold—if x_1 and x_2 are strongly collinear, we simply don't have data about what happens when x_1 changes independently of x_2!
- If there is significant multicollinearity between the predictor variables, what might the statistical significance of each predictor variable look like?

  - It is possible for each coefficient to not be statistically significant even if, paradoxically, an F-test rejects the null hypothesis of all coefficients being equal to 0.

- You run a linear regression and the response variable is perfectly predicted. What is one possible explanation?
  - It's possible that the response variable really is perfectly captured by your predictor variables, but this is typically not the case in real life. One possible error might be that you have a categorical variable which takes on a unique value for each row (for example, a passenger ID in a dataset of boat passengers) and the regression function you're using automatically expands out the categorical variable into many different indicator variables. With a unique indicator variable for each row of the dataset, the coefficient for each indicator variable can be set to the value of the response variable for its corresponding row, with all the non-indicator variable coefficients set to 0; in that case, you have perfect prediction.
- You run a linear regression and some of the coefficients are absurdly high. What is one possible explanation aside from multicollinearity?
  - The model may be overfit. Overfitting means that your model is learning the noise in the training data, not just the generalizable trends, but it can also be thought of as the case where the model overstates the impact of a given set of predictors on the response variable, which corresponds to having overly high coefficients.

Regularization

- What is the L^1 norm of a vector?
  - The sum of the absolute values of its entries, $|x_1| + |x_2| + \cdots + |x_n|$.
- What is the L^2 norm of a vector?
  - The square root of the sum of the squares of the absolute values of its entries, $\sqrt{|x_1|^2 + |x_2|^2 + \cdots + |x_n|^2}$.
- What is L^1 regularization?
  - L^1 regularization refers to adding a regularization term onto the cost function of a model equal to the L^1 norm of its parameter estimates.
- What is L^2 regularization?
  - L^2 regularization refers to adding a regularization term onto the cost function of a model equal to the L^2 norm of its parameter estimates.

- Will an unregularized or regularized linear model have a lower error on training data?
  - The unregularized model will perform better on training data, since it directly minimizes the sum of squared errors! In contrast, regularized model is minimizing the SSE plus a regularization term.
- What is one advantage of using a regularized linear model?
  - Regularization helps prevent overfitting (since it penalizes large coefficients, and overfit models have overly large coefficients) and therefore improves the performance of a linear model on test data.
- What is one disadvantage of using a regularized linear model?
  - A regularized cost function takes longer to fit. The coefficient estimates for an ordinary least squares regression model can simply be calculated with a couple of fast matrix operations, but a regularized cost function means that some numerical optimization is required (*e.g.*, gradient descent).
- What is the major difference in the outcomes obtained by using L^1 or L^2 regularization?
  - L^1 regularized linear regression tends to produce *sparse* coefficient estimates, with many coefficients set to 0 and only the important ones having positive values, whereas L^2 regularized linear regression tends to shrink all coefficients toward 0 in a more uniform manner such that no coefficient is reduced all the way to 0.

    Intuitively, this can be understood as the cost function decreasing at a constant rate in L^1 regularization as coefficients grow smaller, all the way to 0, whereas with L^2 regularization there are "diminishing returns" on making a coefficient smaller and smaller (since squaring a small number will produce an even smaller number).

    To elaborate further, L^1 regularization will tend to only retain the predictor variables which are the most strongly correlated with the response variable. L^2 regularization can actually be used to find a unique solution to an *ill-posed* problem, where you have more predictor variables than data points!
- Why do we use L^1 regularization instead of $L^{1/2}$ or L^0 regularization?
  - Although the $L^{1/2}$ and L^0 norms enforce more sparsity in the solution—indeed, the L^0 norm of a vector is simply equal to the number of nonzero entries of that vector—they are difficult to use in optimization.

    Consider a situation where we constrain the L^2 norm of (x, y) to be less than some constraint C, that is, $\sqrt{x^2 + y^2} < C$. The set of

(x, y) points which satisfy this condition form a filled-in circle. If we choose any two points in that region and connect them with a straight line, the line itself is also contained completely within the region; this property is called *convexity*, and convex optimization problems are substantially easier to optimize than nonconvex optimization problems. For the L^1 norm, we have a filled-in diamond instead of a filled-in circle, but the region is still convex. However, for the L^p norm as p decreases below 1, the region starts to looks like a diamond with the sides caved in; such a region is *not* convex.

Although the L^0 norm is the norm that enforces the most sparsity, we use the L^1 norm partially because it is the closest *convex* approximation to the L^0 norm.

Tangentially, we don't typically use the L^3 norm or other norms because nice statistical properties haven't been established for such norms (whereas they have been for the L^1 and L^2 norms).

- How can L^1 and L^2 regularization be combined?
  - You can add on a regularization term equal to a linear combination of the L^1 and L^2 regularization terms, like $\alpha + \|\beta\|_1 + (1 - \alpha)\|\beta\|_2$. This is called elastic net regularization.
- What is one interpretation of regularization in terms of Bayesian modeling?
  - Regularization can be interpreted as imposing a prior distribution on the distribution of coefficient estimates, where the L^1 regularization term corresponds to a Laplacian prior peaking at 0 and the L^2 regularization term corresponds to a Gaussian prior peaking at 0.

Resampling

Overfitting

- What is one disadvantage of having more columns than rows in your dataset?
  - If you have too many predictor variables, you might end up overfitting on account of having too many degrees of freedom, which allows your model to learn the noise in your training data instead of just the generalizable trends and patterns.
- What is one concrete example of overfitting?
  - A concrete example of overfitting is a complex polynomial model which curves wildly to go through every data point perfectly.
- What is one way to deal with overfitting?
  - Overfitting can be dealt with by using a regularized model which penalizes large parameter estimates. (An overfit linear model, for example, will have overly large coefficient estimates, *i.e.*, it will overstate the effect of some predictor variables on the response variable.)

Cross-validation

- What is n-fold cross-validation?
  - n-fold cross-validation is the following procedure: First, we split our data into n different "folds" of approximately equal size. Next, we iterate over each of the folds; for each fold, we train a model on the *rest* of the data and make predictions for that specific fold. We aggregate these predictions and compare them to the true values of the response variable by calculating some error metric (like the RMSE).
- What is the purpose of n-fold cross-validation?
  - n-fold cross-validation allows us to provide an estimate of the generalization error of our model. We can use these estimates to compare different models in a way that is less biased by the overfitting of each model to the training data.
- What values can n take on in n-fold cross-validation?
  - The minimum value of n is 2. The maximum value of n is equal to the number of rows in the dataset.
- What is the special name used for the case when n is maximal?

  - Choosing the maximum value of n for n-fold cross-validation is called leave-one-out cross-validation (LOOCV).

- What is an advantage of using a higher value of n in n-fold cross-validation?
  - n-fold cross-validation will typically *overestimate* the generalization error of a model, because each model trained in n-fold cross-validation only sees a subset of the data whereas when your model is actually, you train it on *all* of the training data. This is not an issue for model *comparison*, because we only look at the relative differences between the cross-validated RMSE (or some other cost function) values of different models; shifting them all up or down by approximately the same amount has no effect on which model we choose. However, it is a problem when we want to know how well our chosen model will really do on truly new data. Increasing the value of n will correspondingly increase the amount of data used for each model trained by n-fold cross-validation, resulting in a less upwardly biased estimation of the generalization error of the model.
- What is a disadvantage of using a higher value of n in n-fold cross-validation?
  - Using more folds in n-fold cross-validation requires a substantial increase in the amount of computation time required.

Bootstrapping

- What is a bootstrapped sample?
  - A bootstrapped sample can be constructed by taking a dataset and sampling its rows *with replacement* until we have a new dataset with the same number of rows. Since we sample with replacement, some of the rows of the original dataset will show up multiple times in the bootstrapped sample and some will not show up at all.
- What is the size of a bootstrapped sample?
  - By definition, a bootstrapped sample has the same number of rows as the original dataset.
- Approximately what proportion of the original dataset shows up at least once in a bootstrapped sample?
  - Approximately two-thirds, which is a consequence of $1 - 1/e \approx 0.632\ldots$. (It's important to remember this, less so to remember the derivation, which is easily found online.)
- Suppose that we generate many bootstrapped samples of a dataset. We can estimate the generalization error by training a model on the original

dataset and making predictions on the bootstrapped samples, or by training a model on each bootstrapped sample and making predictions on the original dataset. Which method is preferred?

  - We prefer the latter. In the former, the model has already seen the entire training dataset, and none of the data in each bootstrapped sample is new to the model; as such, doing so does not give an estimate of the generalization error! In the latter, each model only sees the data points which occur in each bootstrapped sample, which is approximately two-thirds of the original dataset.

- If we train a model on each bootstrapped sample and make predictions on the original dataset, approximately two-thirds of the "test" data will already have been seen by the model. How can we account for this?

  - We account for this by only making predictions on the subset of the original dataset which each model did *not* get trained with in its corresponding bootstrapped sample. As such, each model is only used to predict values for approximately one-third of the original dataset.

- How can bootstrapping be used to estimate the distribution of parameters of probability distributions?

  - Generate many bootstrapped samples of a dataset, calculate the best parameter estimate for each of those bootstrapped samples, and look at the distribution of those parameter estimates. For example, if you wanted to know how precisely you can estimate the mean of a set of values, you can generate many bootstrapped samples of those values, calculate the mean of each of those bootstrapped samples, and look at the variance of the distribution of those means.

    In practice, simple parameters like means and standard deviations are better treated with tools of classical statistics, but bootstrapping is useful when parameters estimates are more complex functions of a dataset which have not been theoretically studied in great detail (or are not easily theoretically analyzed).

Logistic regression

- What does the "binomial" in "binomial logistic regression" mean?
  - The "binomial" in "binomial logistic regression" refers to the fact that the response variable is binary, *i.e.*, it can take on one of two values. It is not related to the binomial distribution.
- What assumption is made about the distribution of the response variable in a logistic regression model?
  - We assume that the response variable follows a Bernoulli distribution. That is, we assume that each data point x_i with a class label $y_i \in \{0, 1\}$ takes on $y_i = 1$ with some probability p_i, and that all the different p_i are independent of one another.
- What is being linearly modeled in logistic regression?
  - Instead of modeling the probabilities p_i directly, we model the corresponding log odds value, because probabilities are constrained to be within 0 to 1 but a linear model can predict values well outside that range.
- What is the minimum value of a log odds?
  - Negative infinity.
- What is the maximum value of a log odds?
  - Positive infinity.
- What assumption is made about the log odds associated with the probabilities of the Bernoulli distributions in a logistic regression model?
  - We assume that the log odds of the model are a linear function of our predictor variables.

Comparison to discriminant analysis

- What generative assumption is made by linear discriminant analysis?
  - Linear discriminant analysis (LDA) assumes that both classes (in a two-class classification problem) are generated by sampling from Normal distributions with equal covariances.
- Which classification method makes stronger assumptions: logistic regression or linear discriminant analysis?
  - LDA makes stronger assumptions than logistic regression.

Maximum likelihood estimation

- What is a likelihood function?
  - A likelihood function is calculated with respect to the parameters of some model and a set of observed data. It assumes that the model is true and measures the corresponding probability of observing the observed data given those model parameters.
- What is the minimum value of a likelihood function?
  - The minimum value of a likelihood function is 0.
- What is the maximum value of a likelihood function?
  - The minimum value of a likelihood function is 1.
- What does the area under a probability density function need to be equal to?
  - The area under a probability density function must be equal to 1.
- What is the function for the probability density of a uniform distribution from a to b?
  - A uniform distribution in the interval $[a, b]$ has a probability density equal to $1/(b-a)$ in $[a, b]$ and 0 elsewhere.
- What is the likelihood function for a uniform distribution from a to b and a single observation y?
  - The corresponding likelihood function is equal to $1/(b-a)$ if y falls within $[a, b]$ and 0 otherwise.
- A random variable is 1 with probability p and 0 otherwise. What is the likelihood function for a single observation y?
  - The corresponding likelihood function is p if $y = 1$ and $1 - p$ if $y = 0$. This can also be written as $p^y(1-p)^{1-y}$.
- A set of independent random variables are 1 with probability p_i and 0 otherwise. What is the likelihood function for a set of observations y_i?
  - The corresponding likelihood function is the product of all of their individual likelihood functions, *i.e.*, $p_1^{y_1}(1-p_1)^{1-y_1}p_2^{y_2}(1-p_2)^{1-y_2}\cdots p_n^{y_n}(1-p_n)^{1-y_n}$.
- What numerical method is typically used to calculate the probabilities which maximize the likelihood function for logistic regression?
  - Newton's method (or a more complex variation thereof) is typically used to optimize a logistic regression model.

- How can the maximum likelihood estimation of a logistic regression model be interpreted in terms of minimizing a cost function?
  - Denote the likelihood function by L. Maximizing the likelihood is equivalent to minimizing the additive inverse of the likelihood, $-L$, or the multiplicative inverse of the likelihood, $1/L$. Maximizing L is equivalent to maximizing $\log L$ which is equivalent to minimizing $-\log L$, and $-\log L$ is known as the "log loss".

Visualization

- What is sensitivity?
  - Sensitivity is the true positive rate (the number of true positives divided by the total number of positive classifications).
- What is specificity?
  - Sensitivity is the true negative rate (the number of true negatives divided by the total number of negative classifications).
- What variable is plotted on the y-axis of an ROC curve?
  - Sensitivity is plotted on the y-axis of an ROC curve.
- What variable is plotted on the x-axis of an ROC curve?
  - 1 minus the specificity, or the false positive rate, is plotted on the x-axis of an ROC curve.
- What ROC curve would correspond to a completely random classifier?
  - A random classifier that chooses one of two classes with equal probability corresponds to a straight line of the form $y = x$, connecting $(0, 0)$ and $(1, 1)$.
- What is the AUC?
  - The AUC is the area under the ROC curve.
- What is one usage of the AUC?
  - The AUC can be used as a metric of model performance (especially if you calculate a cross-validated estimate of the AUC). Indeed, if the ROC curve for model A lies completely above the ROC curve for model B, model A is strictly superior than model B *and* has a higher AUC.
- What is one way to visualize and interpret log odds values corresponding to predictions made by a logistic regression model?

  - For a classification model which distinguishes between classes A and B, you can plot two histograms on the same graph where one histogram is for the log odds values predicted for points belonging to class A and the other histogram is for the log odds values predicted for points belonging to class B. If there is substantial overlap in the two histograms, the model does a poor job of distinguishing between the two classes; if there is significant separation, the model distinguishes between the two classes well.

Multinomial logistic regression

- What does the "multinomial" in "multinomial logistic regression" mean?
  - The "multinomial" in "multinomial logistic regression" refers to the fact that the response variable can take on one of *multiple* values. It is not related to the multinomial distribution.
- Briefly describe how multinomial logistic regression differs from standard binomial logistic regression.
  - Multinomial logistic regression calculates a probability of membership for each class, normalized so that the probabilities all add up to 1.

Principal component analysis

- Is principal component analysis supervised or unsupervised learning?
  - Principal component analysis does not take into account any response variable and is therefore an unsupervised learning technique.
- What is the first principal component of a dataset?
  - The first principal component of a dataset is the single direction which captures the most variation in the data.
- What is the second principal component of a dataset?
  - The second principal component of a dataset is the direction which captures as much variation in the data as possible while being perpendicular (orthogonal, at right angles) to the first principal component.
- How many principal components does a dataset have?
  - The number of principal components of a dataset is equal to the number of variables (columns) in the dataset.
- What is a linear combination?
  - A linear combination refers to an operation where you take a set of values x_i, multiply each one by a constant (which is not necessarily the same for each x_i), and take their sum. For example, $x_1 + 2x_2 - 9x_3$ is a linear combination of $\{x_1, x_2, x_3\}$.
- What are the "loadings" of a principal component?
  - Each principal component is represented as a linear combination of the original variables. The loadings are the coefficients of this linear combination. For example, a principal component might be equal to $5x_1 + 3x_2 - 4x_3$, in which case the loadings would be $(5, 3, -4)$.
- What are the "scores" of a principal component?
  - The scores of a principal component are precisely the results of evaluating the corresponding linear combination of the original variables.
- What is a geometric interpretation of principal component analysis?
  - Suppose we have a dataset with p variables, which we can think of as a set of points in p-dimensional space. Principal component analysis can be thought of as a rotation of the coordinate axes in this p-dimensional space such that the coordinate axes all remain orthogonal to each other (with the constraint that the first principal component explains as much variation as possible, the second explains as much as possible after the first, etc.).

- What technique from linear algebra is typically used to calculate the principal components of a dataset?
  - Singular value decomposition, a matrix factorization technique, is typically used to perform principal component analysis. It represents the matrix of data as the *product* of several other matrices.
- Principal component analysis produces an "eigenvalue" associated with each principal component. What is one interpretation of these eigenvalues?
  - Of the variation in the original dataset, each principal component explains a certain proportion of that variation. The eigenvalue of each principal component is proportional to the amount of variation explained by that principal component, and the exact percentages can be calculated by taking each eigenvalue and dividing them by the sum of all eigenvalues.

Visualization

- What is one way to visualize the results of principal component analysis?
  - Methods to visualize the results of principal component analysis include plotting the scores of the first and second principal component against each other, plotting the correlations between the original predictor variables and the scores of the principal components, and plotting the loadings of the principal components.
- What information can principal component analysis provide you?
  - Principal component analysis can indicate if your predictor variables can be well-represented in a lower-dimensional space. Moreover, principal component analysis is very interpretable; by examining the loadings of the first principal component, one can assign some meaning to the direction of greatest variation, and so on and so forth. In general, it helps to reveal structure in unlabeled data.
- What is one way to visualize the eigenvalues of principal component analysis?
  - The kth principal component's eigenvalue can be plotted on the y-axis and k can be plotted on the x-axis. The eigenvalues should decrease with increasing k.

Principal component regression

- What is principal component regression?

  - Principal component regression refers to the technique of performing principal component analysis on a set of predictor variables, taking the top k principal components' scores, and running a linear regression to predict the response variable in terms of those k scores.

- What is one advantage of principal component regression?
  - Principal component analysis will represent groups of highly multicollinear predictor variables as a single principal component, so principal component regression may combat overfitting or other problems associated with multicollinearity. In addition, throwing away data means that your regression model will take less time to run.

- What is one disadvantage of principal component regression?
  - Principal component regression typically (but not always) gives results worse than those of regularized linear regression. This is because discarding the less important principal components still amounts to throwing away data and information, whereas a regularized model can work well with an overabundance of information.

- How can you quickly determine a good number of principal components to "keep" for principal component regression?
  - You can plot the eigenvalues of the principal components and look for where an "elbow" occurs in the curve.

- What is one reason why principal component regression might have much less predictive power than ordinary least squares regression even if the first few principal components explain most of the variation in the data?
  - The first principal component is the dimension which explains as much variance as possible *among the predictor variables*. This is *not* necessarily the same as being the dimension which would explain as much variance as possible in the *response* variable. It might be the case, for instance, that some of the less-important principal components are in fact much more predictive of your response variable; if those are discarded in favor of the first couple of (less-predictive) principal components, the predictive power of your principal component regression model will be severely impaired.

Clustering

- Is clustering supervised or unsupervised learning?
  - Clustering methods do not take into account any response variable and are therefore unsupervised learning techniques.
- What is one way to measure how "real" a cluster is? *I.e.*, whether or not it's a random fluke.
  - In general, you can run a clustering algorithm on your original dataset, generate many bootstrapped samples of your dataset, run the clustering algorithm on each bootstrapped sample, and try to determine how often the clusters found in the original dataset continue to show up in the clusters of the bootstrapped samples.

    Precisely, one can use the Jaccard index to determine if a cluster found in the original dataset is approximately equivalent to a cluster found in a bootstrapped sample. To calculate the Jaccard index, you divide the number of points found in both clusters by the number of points found in either cluster; the more similar the two clusters, the closer the Jaccard index is to 1, and the more dissimilar the two clusters, the closer the Jaccard index is to 0. We can say, for example, that two clusters are "approximately identical" if their Jaccard index is greater than 0.9; with such a criterion, we can then proceed to calculate the probability that a cluster found in the original dataset "survives" among the clusters found in a bootstrapped sample.
- What is one reason why clustering is challenging with high-dimensional data?
  - The primary reason is the curse of dimensionality, which typically refers to the fact that distance functions in higher-dimensional spaces become effectively constant. That is, distances between pairs of points all converge to the same value as the dimensionality of a dataset increases. Since clustering depends on looking at the distances between points, this will stop clustering algorithms from working properly.

    Another reason is that it is difficult to interpret the meaning of clusters in high-dimensional space on account of such spaces being difficult to visualize.
- What is one way to overcome the curse of dimensionality?
  - You can perform dimensionality reduction on the dataset before trying to cluster the data.

K-means clustering

- Is *K*-means clustering a generative or discriminative model?
  - *K*-means clustering is a discriminative model. It does not propose any assumption about how the data are generated; rather, it just tries to find the best possible cluster assignments based on some criterion.
- What is the minimum value of *K*?
  - The minimum value of *K* is 1. A single-cluster model would be centered at the centroid of all the data.
- What is the maximum value of *K*?
  - The maximum value of *K* is equal to the number of points in the dataset, corresponding to a model where each cluster is centered precisely on a single point.
- What are the steps of the *K*-means algorithm?
  - (1) First, we initialize the centers of the *K* clusters to random points. (2) Second, we assign each point to the cluster with the center closest to the point. (3) Third, we calculate the center of each cluster (based on the points assigned to that cluster). (4) We repeat steps 2 and 3 until the locations of the centroids converge.
- What is one advantage of using *K*-means clustering?
  - *K*-means clustering is very fast.
- What is one disadvantage of using *K*-means clustering?
  - The results of *K*-means clustering are unstable and can change based on the whim of the random number generator.
- Where does the instability of the results of *K*-means clustering come from?
  - The randomness in *K*-means clustering, and therefore the instability in the results, comes from the fact that the starting position is randomly chosen. The algorithm may therefore converge to a "local minimum" which is not the ideal result.
- What is one way to account for the instability of the results of *K*-means clustering?
  - There is a tradeoff between using *K*-means clustering, which is fast but unstable, and a more complex clustering algorithm (there are many such algorithms, like simulated annealing), which is slow but stable. One can therefore use a more complex clustering algorithm and stop it before it converges just to get a good "starting position" for *K*-means clustering.

Minimizing variance

- What quantity is minimized by the K-means algorithm?
  - The K-means algorithm chooses cluster centroid locations such that the *within-cluster variation* is minimized. More precisely, it minimizes the sum of the squared distance from each point to the closest cluster centroid.
- How is the distance between points calculated in the K-means algorithm?
  - The K-means algorithm uses Euclidean distance to calculate the distance between points. Two points $x = (x_1, x_2, \ldots, x_n)$ and $y = (y_1, y_2, \ldots, y_n)$ are separated by a distance $d(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2}$.
- Can you use other distance metrics with the K-means algorithm?
  - The answer is nuanced. The K-means algorithm is *implicitly* based on the Euclidean distance because of how it minimizes the within-cluster variance; if you use a different distance metric with K-means, the algorithm may not converge.

    However, if you have a matrix of cosine distances, correlations, or covariances between data points, it is possible to easily *transform* those into Euclidean distances and then use that matrix for K-means clustering.
- As K increases, what happens to the within-cluster variance?
  - Increasing K necessarily decreases the within-cluster variance. The more clusters, the lower the within-cluster variance.
- What is one way to choose a value of K?
  - A value of K can be chosen due to *a priori* (background) knowledge of the data you are trying to cluster. (For example, you might be trying to cluster data coming from what you know are three different plant species.) Alternatively, you can plot the within-cluster variation for different values of K and try to look for an "elbow" in the graph.
- What is the general algorithm used to for K-means clustering called?
  - The K-means algorithm is a specific instance of the expectation–maximization (EM) algorithm.
- What is the "expectation" step of the K-means algorithm?
  - The expectation step corresponds to the step where each data point is assigned to a specific cluster based on the centroid locations. In that step, we look at the current estimates for the centroid locations

and ask: based on those centroid locations, what cluster assignments do we *expect*?

- What is the "maximization" step of the K-means algorithm?
  - The maximization step corresponds to the step where we recalculate the centroid locations based on the cluster assignments. In that step, we look at the current estimates of cluster assignments and ask: what centroid locations *maximize* the likelihood of those assignments?

Gaussian mixture models

- What is a Gaussian function?
  - A Gaussian function is a function of the form $ae^{-b(x-c)^2}$ for arbitrary constants a, b, and c. The simplest Gaussian function is therefore just e^{-x^2} which, when graphed, looks like a bell centered at $x = 0$.
- What probability distribution is represented by a Gaussian function?
  - The Normal distribution has a probability density function which is a Gaussian function. All Normal probability density functions are Gaussian, but not all Gaussian functions are Normal probability density functions (the area underneath them might not be equal to 1).
- Why is the Normal distribution so commonly used?
  - The central limit says that if you repeatedly sample a set of independent random variables, all of which come from distributions with finite mean and variance (but possibly very different distributions), the means of those samples will follow a Normal distribution. Many different naturally-occurring quantities can be construed as being equal to the mean of several random variables, so in practice assuming that data are generated via sampling from Normal distributions will tend to work fairly well.

    One concrete example of the central limit theorem being applied is in the case where we have a random walk over the natural numbers—we begin at 0 and repeatedly move +1 or -1 with equal probability. Over an infinitely long span of time, the probability that we are at any particular location will follow a Normal distribution peaked at 0. Again, one can imagine such a process being approximately representative of many naturally-occurring phenomena, for which the Normal distribution serves as an appropriate generative assumption.
- Is a Gaussian mixture model a generative or discriminative model?
  - A Gaussian mixture model is a generative model.

- What generative assumption is made by Gaussian mixture models?
  - A Gaussian mixture model proposes that (1) our data are generated from k Normal distributions (where k is specified in advance, like with K-means clustering) and (2) a new data point is first generated by choosing one of the k Normal distributions from a discrete probability distribution over the k possibilities (not necessarily uniform!) and then sampled from the chosen Nomal distribution.
- What is the general algorithm used to fit a Gaussian mixture model called?
  - Like K-means clustering, the algorithm used to fit Gaussian mixture models is a specific instance of the expectation–maximization (EM) algorithm.
- Why do people sometimes call K-means clustering "hard" and Gaussian mixture models "soft"?
  - In K-means clustering, a data point is conclusively labeled as absolutely belonging to a specific cluster. With a Gaussian mixture model, we cannot say anything about which of the Normal distributions each data point is sampled from; we can only estimate the probability of a specific data point coming from a particular Normal distribution.

Decision trees

- What does the acronym CART stand for?
  - CART stands for Classification And Regression Tree.
- Intuitively, how does a decision tree work?
  - A decision tree partitions the space of predictor variables by recursively splitting the space into smaller and smaller volumes. This can be represented as a downward-growing tree, where at each node the tree splits into two different branches based on the value of some particular predictor variable. For example, if $x_i < 5$, we might go down the left split, whereas if $x_i \geq 5$, we might go down the right split.
- What is a "leaf" of a decision tree?
  - A leaf of a decision tree is the same thing as a terminal node—one of the places you end up if you start at the top of the tree and keep going down the splits until you reach the bottom.
- In a regression tree, how is a prediction made at a leaf of the tree?
  - At the leaf of a regression tree, a prediction is made by averaging the values of all the points of the training set which reside in that leaf.
- In a classification tree, how is a prediction made at a leaf of the tree?
  - At the leaf of a classification tree, a prediction is made by examining all the points of the training set which reside in that leaf and choosing the most common class (choosing arbitrarily but consistently in the case of a tie).
- What is one advantage of using a decision tree?
  - Single decision trees are fast, work well on large datasets, and easily visualized and interpreted.
- What is one disadvantage of using a decision tree?
  - The main disadvantage of a single decision tree is that it will tend to overfit on the data. Indeed, in the extreme situation where the tree is allowed to grow all the way until each leaf contains only a single data point from the training set, the degree of overfitting will be tremendous.
- What is one way to combat overfitting in a decision tree?
  - The most common way to combat overfitting in a decision tree is to first grow out the entire tree and then to *prune* the tree back via some criterion.

Bagging

- What does "bagging" stand for?
  - Bagging stands for "bootstrap aggregating".
- How can bootstrapped samples of a dataset be used to improve decision tree performance?
  - You can create many bootstrapped samples, train a decision tree on each one, and average the predictions together. This will help correct for overfitting to some extent.
- How can you estimate the generalization error using models trained on bootstrapped samples?
  - The generalization error can be estimated by using each of the models trained on a bootstrapped sample to make predictions for the subset of the original dataset containing points upon which it was *not* trained, *i.e.*, the out-of-bag samples.

Random forests

- How do random forests relate to bagging?
  - A random forest is a more complex version of using bagging with decision trees. It consists of bagging plus an element of randomness (discussed below).
- What part of a random forest is random?
  - For each split in each decision tree trained in a random forest, the predictor variable used for the split is chosen from a random subset of the predictor variables.
- What purpose does the randomness of a random forest serve?
  - The randomness of a random forest *decorrelates* the different trees, further preventing overfitting in the final, averaged model.
- What is the single hyperparameter of a random forest?
  - The single hyperparameter of a random forest is the size of the random subset of predictors chosen at each split. In R, it is typically referred to as `mtry`; if `mtry` is equal to, say, 10, then at each split of each tree we choose the best predictor to split on out of 10 randomly chosen predictor variables.

    Note that sometimes the number of trees to train is considered a hyperparameter of random forests as well.

- What is the minimum value of the single hyperparameter of a random forest?
  - The minimum value of the single hyperparameter of a random forest is 1.
- What is the maximum value of the single hyperparameter of a random forest?
  - The minimum value of the single hyperparameter of a random forest is the number of predictor variables in the dataset.
- What is typically considered to be a good default value for the single hyperparameter of a random forest?
  - A commonly accepted default value for the single hyperparameter of a random forest is the square root of the number of predictor variables in the dataset. Note that you will typically have to round the square root to a whole number (unless the number of predictor variables happens to be a perfect square).
- What is one advantage of using a random forest?
  - Random forests are easy to parallelize because the trees of a random forest are independent of each other—one can train a large number of trees in parallel and simply average their predictions. They are also quickly trained on account of only having one hyperparameter to tune. In general, they perform quite well as an out-of-the-box machine learning technique.
- What is one disadvantage of using a random forest?
  - Each prediction made by a random forest model entails making a prediction from each of its constituent decision trees and averaging those predictions together. If there are many trees in the model, making many predictions can take a long time.
- How do the trees of a random forest depend upon each other?
  - The trees in a random forest do *not* depend upon each other—each one can be trained independently of the others. You can even train a random forest model, decide that its performance is insufficient, and try adding in more trees to see if it improves.
- How can the training of a random forest be parallelized across multiple processors?
  - Since the trees of a random forest are not dependent upon one another, a random forest can easily be parallelized by allocating an equal proportion of the trees to be trained to each processor and then combining their results afterward.

- How is the standard measure of variable importance for a random forest calculated?

Gradient boosted trees

- What is an intuitive explanation of gradient boosting?
  - Gradient boosting is a method where a single model is iteratively improved by looking at its residuals, training a new model on those residuals, incorporating the new model into the boosted model, training a new model on the residuals of the boosted model, incorporating the new model into the boosted model, and so on and so forth. It can be thought of as incremental, iterative improvement of a single boosted model, where each new, to-be-incorporated model is trained specifically to account for the error of the boosted model.
- What are the three hyperparameters of a gradient boosted tree?
  - Gradient boosted trees typically have three hyperparameters: the regularization or "shrinkage" parameter (also known as the learning rate), the depth of the trees, and the number of trees to train.
- What is one advantage of using a gradient boosted tree?
  - Gradient boosted trees are one of the best "out-of-the-box" machine learning models in terms of final performance on a regression or classification task (after tuning). They typically perform better than random forests, regularized linear regression models, etc.
- What is one disadvantage of using a gradient boosted tree?
  - The main disadvantage of using a gradient boosted tree is that training the model involves a very long hyperparameter search—with three hyperparameters to search over, the corresponding grid search will take a long time.
- How do the trees of a gradient boosted tree model depend upon each other?
  - There is a sequential dependence, that is, we can only train one tree at a time, since each additional tree is trained to predict the residuals of the results of previous trees combined.
- How can the training of a gradient boosted tree be parallelized across multiple processors?
  - Parallelization can be accomplished at the level of individual decision trees. The decision trees are still trained sequentially, but each one can be spread out over multiple processors. (Unlike random forests, parallelization cannot be accomplished via spreading out entire trees

over multiple processors because of the sequential dependence inherent in gradient boosting.)

- What is the name of the most popular software package for parallelized training of gradient boosted trees?
  - The most popular gradient boosted tree software package is called XGBoost. It can be used in a variety of programming languages, including R, Python, and Scala.

Recommender systems

- What is one example of a problem for which you would use a recommender system?
  - Recommender systems can be used for many different tasks—music recommendation, movie recommendation, book recommendation, e-commerce product recommendation, online article recommendation, etc.
- What is one example of explicit data collection?
  - Examples of explicit data collection include asking users to rate movies, asking users to indicate whether they like or dislike a song, and asking a user which genres they prefer.
- What is one example of implicit data collection?
  - Examples of implicit data collection include tracking users' browsing habits (both the sequences of items viewed and the time spent on each item) or trying to access, for instance, the Music folder of a users's computer to get information on song preferences.
- What is one example of a situation where you might want to recommend an already-seen item to a user?
  - There are many purchases which should be made on a recurring basis, like toilet paper or detergent. An e-commerce website like Amazon might want to re-recommend such items to a user after a certain minimum amount of time has elapsed from the last time the item was purchased.
- What is one example of a situation where you might not want to recommend already-seen items to a user?
  - People typically prefer watching new movies rather than old movies, so Netflix might not want to recommend to users movies which they have already seen.
- Should recommender systems take into account a user's entire history?
  - In general, no, recommender systems should not always take into account a user's entire history (in an equally weighted fashion). Users' preferences can change over time, so it is typically useful for an algorithm to give higher weight to more recent user activity.

    There may also be potential privacy concerns; for example, an avid purchaser of pornography on Amazon might not want more pornography to be visibly recommended, which might lead to embarrassing situations if they opened up Amazon in public.

- Aside from optimizing solely for recommendation accuracy, what are some other factors a recommender system should take into account?
  - A recommender system might want to take into account diversity of recommendations (a user might prefer a more diverse assortment of recommendations instead of optimizing purely for enjoyment of each particular item), the degree to which already-seen items should be re-recommended, and the surprisingness of the recommendations (some recommendations might be so obvious as to be banal and annoying).

Collaborative filtering

- What is one concrete, real-world example of collaborative filtering?
  - Netflix primarily uses collaborative filtering based on users' movie ratings to recommend new movies to its users.
- What is the fundamental assumption made in collaborative filtering?
  - The fundamental assumption of collaborative filtering is that two people who like some item X will also be more likely than average to agree on item Y.
- What is one advantage of using collaborative filtering?
  - Collaborative filtering only uses ratings and therefore does not need any knowledge about the *content* of each recommended item. By simply making predictions for a user based on similarly-minded users' preferences, collaborative filtering can easily make, *e.g.*, cross-genre recommendations which seem quite serendipitous to the end user.
- What is the scalability problem?
  - Collaborative filtering methods often require the re-computation of an enormously large matrix for every purchase, which is computationally expensive and may not be sufficiently responsive to users' actions and choices on the timescale of minutes (*e.g.* the system might not be able to easily react to a user's purchase and recommend related items without adapting the system slightly).
- What is one way to overcome the scalability problem?
  - The scalability problem can be to some extent overcome by spending a lot of money on computing power and storing your data in RAM to speed up I/O time during computation.
- What is one disadvantage of collaborative filtering which has not already been mentioned?

  - Collaborative filtering suffers from the *cold start* problem, where good recommendations for a user can only be made once robust ratings data for that user have been obtained. As such, this makes it difficult to generate high-quality recommendations for a new user.

- What is the name of one algorithm for performing collaborative filtering?
  - Reduced rank singular value decomposition via the method of alternating least squares.

Content-based filtering

- How does content-based filtering differ from collaborative filtering?
  - In collaborative filtering, we calculate recommendations based on how different users have rated the various items under consideration. In contrast, with content-based filtering we look at the *categories* that the items fall into (*e.g.*, the genres of different movies or the national origin of books' authors) and make recommendations based on users' preferences for those categories.
- What is one concrete, real-world example of content-based filtering?
  - The classic example of real-world content-based filtering is Pandora, an Internet music radio, where users can indicate that they like or dislike particular songs. Genre preferences are indicated from a users' likes and dislikes and those preferences are used to determine which songs are suggested for the user.
- What is one advantage of using content-based filtering?
  - The advantages of content-based filtering include: the system needs less information than collaborative filtering to work well, as it only takes into account users' categorical preferences and the categorical assignments of each item; the method of recommendation is more transparent to the end users, because it's based on observable item categories rather than other users' unobservable preferences; there's less of a cold start problem for new users, who only need to express rough categorical preferences before good recommendations can be made (as opposed to needing many ratings of different individual items).
- What is one disadvantage of using content-based filtering?
  - The disadvantage of content-based filtering include: as items' categorical memberships and users' categorical preferences in general contain less information than a matrix of user–item ratings, the recommendations produced may be "rougher" than those produced by a collaborative filtering algorithm with sufficiently many ratings; the

system will naively produce very few "surprising" recommendations or cross-genre recommendations; although the cold start problem is not as severe as in collaborative filtering, one nevertheless still needs information from new users about which categories (genres, etc.) they prefer.

Hybrid methods

- What is one way to combine the results of multiple recommender systems?
  - Multiple recommender systems can be combined in a variety of ways. Predicted ratings can be averaged across the different systems, either uniformly or in a weighted fashion with the weights determined individually for each user; the hybrid system can switch between different recommender algorithms entirely, finding the best one for each user; recommendations from different algorithms can be presented separately and simultaneously in different locations (as the best recommender algorithm for two different places on a website need not be the same algorithm); recommender systems can be given a hierarchical ordering, where a lower-order system is used as a "tiebreaker" if a higher-order one fails to clearly distinguish between two recommendations. Finally, recommender systems can be chained together, with the output of one being used as the inputs of the next.
- What is the cold start problem?
  - The cold start problem typically refers to the difficulty of producing recommendations for new users with a collaborative filtering algorithm; it is difficult to produce good recommendations for someone about whom almost nothing is known! There is an analogous version of the problem for *items*; as new items are added to the database, they won't show up in recommendations until they've been rated by a sufficiently large number of users (which works against them getting any ratings at all). Cold start problems arise from the sparsity of the ratings matrix.
- What is one way to overcome the cold start problem for users?
  - It's possible to hybridize a collaborative filtering system along with a content-based system, preferring the former if users have a sufficient amount of ratings data but preferring the latter for new users.
- What is one way to overcome the cold start problem for items?
  - New items could be automatically assigned "default" ratings based on similarities between the new item and preexisting items (*i.e.*, through a content-based system), with user-generated ratings slowly taking precedence as they come in.

- If the vast majority of a website's users don't have accounts and you can't use cookies to track individual users, how would you design a recommender system?
  - For each item or webpage, you could track which pages the user was most likely to have *come* from and which pages the user is most likely to go to *next* using some sort of URL-based tracking. The recommender system can then recommend the items or pages which users are most likely to want to see next (*i.e.*, the clickthrough rate to other items), perhaps also weighted by some other quantity like the amount of profit made on each item.

Advanced topics

k-Nearest Neighbors

- What is *k*-Nearest Neighbors?
  - *k*-Nearest Neighbors is a model which makes predictions on a new data point by looking at the *k* points in the training set closest to the new point. In a regression problem, the values of the response variable corresponding to those *k* points are averaged, whereas in a classification problem, the most common class among those *k* points is taken as the predicted class (breaking any ties in an arbitrary fashion).
- What is one advantage of *k*-Nearest Neighbors?
  - *k*-Nearest Neighbors has no model training process—the model just consists of the training set and the rule used for making predictions. There are therefore no distributional or generative assumptions made about the data, and in principle a *k*-NN model can learn arbitrarily complex patterns. Finally, *k*-NN is easy to visualize and straightforward to explain to others.
- What is one disadvantage of *k*-Nearest Neighbors?
  - A *k*-Nearest Neighbors model is difficult to interpret, as it doesn't generate any parameters or other descriptions of the trends learned. It is also computationally quite expensive to find the *k* points closest to a given point; the time necessary to make a prediction may be quite large. Finally, *k*-NN relies on distance functions and therefore suffers from the curse of dimensionality—it will perform poorly in a high-dimensional setting, where distances all become very similar to each other.
- When would *k*-Nearest Neighbors work well?
  - *k*-Nearest Neighbors works well for prediction when the training set is sufficiently large and there are enough points in the region of interest that all of the small-scale detail is well-represented. Unfortunately, *k*-NN may also take a long time to make predictions in such situations. *k*-NN can also work well as *part* of a larger machine learning algorithm; one example of this is Cubist, which incorporates a more complex version of *k*-NN as the final step of a decision tree-based methodology.
- How would you extend *k*-Nearest Neighbors so that closer points affect the model's predictions more than points farther away?

  - One way to do this in a regression setting would be take the *weighted mean* of the response variable's values at the k points, where each value is weighted by the the inverse distance $1/d$ between the corresponding point in the training set and the point for which you want to make a prediction. An analogous weighting scheme could be used for classification.

Support vector machines

- What does it mean for two classes to be linearly separable?
  - In a binary classification problem, two classes are linearly separable if they can be perfectly separated by a hyperplane in the space of predictor variables. *E.g.*, if there were 3 predictor variables, two classes of data would be linearly separable if they could be perfectly separated by a plane in the 3-dimensional space of the 3 predictor variables.
- In a linearly separable binary classification problem, what is the margin?
  - The margin is, intuitively, the "distance" between the two classes. It can be formalized as follows: Consider all *pairs* of hyperplanes such that (1) the hyperplanes are parallel and (2) both hyperplanes separate the two classes of data. Next, consider the *distance* between these pairs of hyperplanes. The margin refers to the space between the pair of hyperplanes with the greatest distance in between them or, alternatively, the distance itself between that pair of hyperplanes.
- What is the maximum-margin hyperplane and why is it important?
  - Using the previously described construction of the margin, consider the hyperplane which is perfectly *in between* the two hyperplanes which border the margin. This is the maximum-margin hyperplane. Its importance arises from the fact that the margin is the most "uncertain" region of classification—for data which resides very far away from the margin, the prediction of class is not difficult, but it is precisely the data close to the margin for which we want the most predictive confidence. We can choose many different separating hyperplanes, but which one is the best? The construction of the maximum-margin hyperplane is motivated precisely by our desire to discriminate as well as possible for data at the very edge of the margin.
- How do we fit a linear SVM if the two classes of data are not linearly separable?
- What is a support vector?
- What is a dot product?

  - A dot product is a function of two vectors of equal length. If $x = (x_1, x_2, \ldots, x_n)$ and $y = (y_1, y_2, \ldots, y_n)$, then the dot product between x and y is given by $x \cdot y = x_1 y_2 + x_2 y_2 + \cdots + x_n y_n$. The dot product between two vectors is related to the *angle* between them and therefore can be interpreted as a metric of similarity; dot products show up very often in a variety of different machine learning algorithms.

- What is a kernel function?
- What is one example of a kernel function?
  - Kernel functions include the linear kernel (which is equivalent to the regular dot product), the radial kernel (which is a Gaussian function of the Euclidean distance between two points), and the polynomial kernel (which is rougly equivalent to the linear kernel raised to a finite integer power).
- What is another name for a radial kernel function?
- What is one example of a situation where you would want to use a linear kernel?
  - A linear kernel is a good choice for problems where you believe the decision boundary is roughly linear in the original feature space. You might be able to determine this from *e.g.* plotting the dataset (if it isn't too large).
- What is one example of a situation where you would want to use a radial kernel?
  - A radial kernel is a good default choice for problems where the decision boundary is at least somewhat nonlinear. It's typically worth comparing the performance of a (tuned) radial kernel SVM against that of a linear SVM just to see how much improvement you can get by modeling nonlinearity.
- What is one example of a situation where you would want to use a polynomial kernel?
  - A polynomial kernel is very popular in natural language processing. Polynomial kernels can be thought as implicit maps onto higher-dimensional feature spaces containing all possible polynomial interaction terms up to a specified degree. Their applicability in NLP can be illustrated by considering a simple example: if two predictor variables x_1 and x_2 respectively indicate the presence or absence of two different individual words, say "birthday" and "party", then the quadratic interaction term $x_1 x_2$ models the presence of *both* "birthday" and "party" in the same document.

Natural language processing

- What is an n-gram?
  - An n-gram is a continuous sequence of n elements. For example, "abcde" is a 5-gram if we consider individual letters to be different elements.
- How are n-grams used?
  - Typically, an n-gram denotes either n consecutive words in a sentence or n consecutive characters in a word. The idea of n-grams is valid in domains aside from natural language processing; for instance, you could consider consecutive sequences of amino acids in a protein to constitute different n-grams.

    One concrete example of n-grams might be if you wanted to train a classifier to distinguish between French and German surnames; for predictor variables, you could consider the frequency of each possible 1-gram, *i.e.*, the frequency of each letter in a particular surname ("a", "e", "h", ...), as well as the frequency of each possible 2-gram, *i.e.*, the frequency of every possible *pair* of letters in a particular surname ("ae", "no", ...).
- What is the fundamental assumption of a naive Bayes classifier?
  - A naive Bayes classifier works with indicator variables denoting the presence or the absence of various features in each item to classify. The fundamental assumption of a naive Bayes classifier is that these indicator variables are all conditionally independent, that is, the presence or absence of one gives no statistical information about the presence or absence of another. This allows a series of complex conditional probabilities to be calculated as the product of simple probabilities, making the model computationally tractable.
- What is the classic example of a naive Bayes classifier?
  - The classic example of a naive Bayes classifier is spam filtration. Although the conditional independence assumption of naive Bayes is quite far from the actual state of reality, a naive Bayes classifier nevertheless seems to do fairly well for separating spam emails from real emails.
- What is sentiment analysis?
  - Sentiment analysis refers to the usage of natural language processing
- What is topic modeling?
- What is the generative model proposed by latent Dirichlet allocation?

- Given a large dataset of books, how would you organize them into different genres?
- What is one way to represent a document as a vector?
- What is one appropriate distance metric to use for comparing document similarity?
- Is there any special reason to use cosine similarity instead of Euclidean distance for comparing document similarity?

Parametric vs. nonparametric models

- What is a parametric model?
  - A parametric model is one where the number of parameters to estimate is fixed in advance—that is, the form of the model is specified *a priori*.
- What is one example of a parametric model?
  - Linear regression models are all parametric models. For a dataset with p predictor variables, there are $p + 1$ parameters to estimate (the p coefficients as well as the intercept term); the number of points in the training dataset does not affect the functional form of the model or the number of parameters to be estimated.
- What is a nonparametric model?
  - A nonparametric model is one where the number of parameters to estimate in the model is determined by the amount of data (as in the number of rows), typically roughly increasing with the size of the dataset. Nonparametric models do not completely lack structure—rather, the number and nature of the parameters is simply not completely fixed in advance.
- What is one example of a nonparametric model?
  - k-Nearest Neighbors is the most straightforward example of a nonparametric model. The model generated by k-NN is dependent upon every point of data, and the size of the model grows with the number of rows in a dataset. Decision tree methods are also nonparamtric because they make no assumptions about the distribution of the training data.
- How do parametric and nonparametric models differ in their assumptions?
  - Parametric models make certain assumptions about the distribution of the training data; nonparametric models do not.

- Is a linear SVM a parametric or nonparametric model?
  - The result of a linear SVM is fully specified by a hyperplane in p-dimensional space (where p is the number of predictor variables), so the number of parameters does not change as a result having model.
- Is a radial SVM a parametric or nonparametric model?
  - Perhaps surprisingly, radial SVMs are actually nonparametric models—the radial basis function is calculated on the

Bias–variance tradeoff

- What is the bias of a model?
  - The bias of a model refers to error coming from incorrect assumptions made about the form of the data's generalizable trends. High bias will cause underfitting.
- What is the variance of a model?
  - The variance of a model refers to error coming from the model's sensitivity to small fluctuations in the training set. High bias will cause overfitting, where the model learns the noise in the training set instead of just the generalizable patterns.
- What is the irreducible error?
  - The irreducible error in a dataset refers to inherent noise in the data. Even with a perfect model that has zero bias and zero variance, we would not be able to eliminate the irreducible error in a test set; being noise in the problem itself, it forms a lower bound for the performance of predictive models.
- What is the sum of the bias, variance, and irreducible error, where the first two are associated with some particular error?
  - Their sum gives the generalization error with respect to some particular machine learning algorithm used for some particular predictive modeling problem.
- What is the bias–variance tradeoff?
  - The bias–variance tradeoff represents the fact that changing a model typically reduces either the bias or the variance at the cost of increasing the other. For example, the bias of a linear model can be reduced by adding on polynomial interaction terms between the predictor variables, allowing it to accurately represent a wide variety of different trends, but doing so will also increase the variance of the model, making it highly sensitive to random fluctuations in the training set. Depending on the problem at hand, the decrease in bias may or may

not be enough to counteract the corresponding increase in variance, and finding the correct tradeoff to make is important for optimizing the performance of a model.

- What is an example of a model with low bias and high variance?
  - An example of a low-bias, high-variance model is one which predicts y as a linear combination of $x, x^2, x^3, \ldots, x^n$ for some very high n.
- What is an example of a model with high bias and low variance?
  - An example of a high-bias, low-variance model is a completely constant model which simply models a response variable as being equal to some constant number (*e.g.*, 7.98) plus noise.
- How does regularizing a linear model affect its bias and variance?
  - Adding on a regularization term to the cost function of a linear model increases bias, because it makes the model simpler by imposing a Bayesian prior on the model's parameters being closer to 0 (that is, making a prior assumption of predictor variables being less important than we might naively believe), and correspondingly decreases variance.
- How can you distinguish models which underfit (high bias) from models which overfit (high variance)?
  - As the sample size used to train a model increases, a model suffering from overfitting will asymptotically perform much worse on a test set than on a training set, with a very low error for the latter and a very high error for the former, whereas a model suffering from underfitting may perform similarly on both a training and a test set but will have very high error for both.

Numerical optimization

- What is the gradient of a function?
  - The gradient of a function is, roughly speaking, the multidimensional analogue of the derivative of a function of a single variable. Instead of being a scalar quantity, it is a *vector* denoting the infinitesimal rate of change in each dimension at any particular point.
- What is gradient descent?
  - Gradient descent minimizes a function by iteratively calculating the gradient of the function at any particular point and stepping in the direction of the negative of the gradient. (The negative of the gradient points in the direction of maximal decrease. When *maximizing* a function, gradient descent is instead called gradient *ascent* and steps

in the direction of the gradient itself.) It can be thought of as trying to descend to the bottom of a valley by repeatedly stepping in the direction which takes you down the furthest in that single step.

- What is stochastic gradient descent?
  - In gradient descent, the function to be optimized typically takes the form of the sum of many functions, like $Q = \sum_i Q_i$. (For example, in ordinary least squares regression, we can interpret Q as being the sum of squared errors and each Q_i as being the squared error corresponding to a single data point.) The gradient is then typically of the form $\nabla Q = \sum_i \nabla Q_i$, where i ranges over the entire training set; each iteration of regular gradient descent recalculates the gradient with that full sum. In contrast, stochastic gradient descent uses only a *single* data point for the calculation of the gradient at each point, iterating repeatedly through the possible values of i in a randomly chosen but consistent order.

    Each iteration of stochastic gradient descent takes much less time than the iterations of regular gradient descent, although a much larger number of iterations may be required for convergence. Stochastic gradient descent typically converges faster than gradient descent (in terms of total time required), although the minimum it finds will change each time (because of randomness) and may be slightly less optimal than the result of regular gradient descent.
- What is mini-batch gradient descent?
  - Mini-batch gradient descent can be viewed as a compromise between regular gradient descent and stochastic gradient descent. In mini-batch gradient descent, the gradient is computed at each step with a random subset of the training set (instead of the entire training set or just a single point). The convergence of mini-batch gradient descent is typically smoother than that of stochastic gradient descent and may actually converge faster as well if the requisite numerical operations are properly vectorized.
- What is one advantage of Newton's method compared to gradient descent?
  - Newton's method takes advantage of more information than gradient descent because it looks at the second derivatives of the function it tries to optimize (whereas gradient descent only looks at the first derivatives). As such, when it works, it typically converges faster than gradient descent.
- What is one disadvantage of Newton's method compared to gradient descent?
  - Newton's method is not as widely applicable as gradient descent

because it gets "stuck" easier on poorly behaved functions on which gradient descent would slowly succeed at finding an optimal point instead.

Miscellaneous

- What is one way to combine multiple models into a single, better model?
  - Multiple models can be *stacked* into a single, typically superior model by fitting a (perhaps regularized) linear model for the response variable in terms of the *predictions* of those models, essentially creating a hierarchical structure where the predictions of different models are aggregated together via yet another model.
- What is one real-world example of stacking?
  - The winner of the Netflix Prize, a competition for finding the best collaborative filtering algorithm for predicting user ratings of movies which awarded $1,000,000 to anyone who could beat Netflix's own algorithms by 10%, used a stacked model.
- What is one way to perform a hyperparameter search aside from grid search?
- Why might you want to perform a random hyperparameter search instead of a grid search?
- What is an example of a situation when you would not want to parallelize your code with MapReduce?
  - MapReduce may be inappropriate in situations where the different threads of your algorithm have to communicate with each other (the point of the Map part is to run many jobs independently in a parallel fashion), when you have to handle a large amount of continuously incoming data (MapReduce is best suited for situations where you want to batch-process large amounts of preexisting data), when your algorithm is too complex to be easily separated out into a Map part and a Reduce part (like the training of a SVM), when you need the results of your processing to be available in real-time, or even simply when the complexity of the job does not actually warrant the overhead costs of setting up a distributed system (both time and money).

Problematic datasets

Missing values

- What is a problem which might arise if you remove all rows containing missing values from your dataset?
  - Removing all rows with missing values can skew your analysis. For example, suppose that one wants to predict spending habits based on a number of demographic variables, and that low-income citizens disproportionately tend to leave entries blank in your survey. If you simply remove every row with even a single missing value, your resulting model will capture the relationship between the demographics of high-income citizens and spending habits but *not* the relationship between the demographics of low-income citizens and spending habits. If this goes unnoticed, you might go on to make overly strong conclusions from your data.

    In addition, removing data will reduce the effectiveness of machine learning algorithms in general.
- What is one way to fill in missing entries in a dataset?
  - The simplest way to fill in missing entries in a dataset is to replace each missing value in a column with the column mean. In the case of a categorical variable, you might either select the most common entry or randomly choose a category based on the distribution of the existing categories.
- What is one way to use a recommender system to fill in missing entries in a dataset?
  - There are certain collaborative filtering algorithms which simply fill in every entry of the users–items matrix, such as alternating least squares; such an algorithm can also be used for imputation purposes.

Unbalanced classes

- What is an unbalanced dataset?
  - An unbalanced dataset is one where the response variable is categorical, but the proportions of data in each class vary wildly. An example would be a dataset where 97% of the data are labeled as class 1 and 3% of the data are labeled as class 2.
- If the classes in a binary classification problem are very unbalanced, how could you correct for this?

  - The standard response is to say that you can "enrich" the less common class by repeatedly sampling points from that class's data with replacement and adding that data to the dataset. As such, a class which comprises 0.3% of the data could be enriched up to 10%, 20%, etc. The classes are called "balanced" when the less-prevalent class is sufficiently enriched that each one comprises around 50% of the entire dataset.

    There is *one* classification metric which balancing the classes can substantially improve—the *sensitivity*, or true positive rate, of the classifier. It tends to have very slight or negative effects upon other metrics of classification quality (like the specificity). In a case where you care a *lot* about identifying all of the positive cases, don't mind if you misclassify some points as being positive when they aren't, *and* the positive class makes up a very small proportion of your entire dataset, it may make sense to enrich or even balance the dataset.

- What is the best classification algorithm to use for an enriched classification problem?
  - Of the standard classification methods, it is typically best to use a support vector machine because SVMs make no assumption about the distribution from which the data are drawn. (The caveat is that this is only strictly true for the hard-margin SVM, although the extent that the distribution of data matters for a soft-margin SVM is still small compared to, say, for logistic regression.) That is, SVMs are "distribution-free".
- What classification approach would you use to detect credit card fraud?
  - Fraud cases make up a small proportion of all credit card usage, detecting fraud cases is very important, and the cost of misclassifying a real transaction as a fraud case is fairly low; as such, you could enrich the fraud cases until the fraud/not-fraud classes are balanced and then train a radial basis SVM to classify between the two.

Target leakage

- What is target leakage?
  - Target leakage refers to a situation where the training data has more than it's "supposed" to, leading to unnaturally high model performance on the test set but making the model useless when used in an actual production environment. Target leakage includes situations where the test data is somehow "leaked" into the training data (so the model has seen some of the test data before), when data from the future is mistakenly included with data from the past, when data

not present in the model's real-world operational environment is included in the training and test sets, etc.

- What is one real-world example of target leakage?
  - Many great examples of target leakage come from predictive modeling competitions. Here's an example from Claudia Perlich:

    > Amazon case study: big spenders. The target of this competition was to predict customers who spend a lot of money among customers using past purchases. The data consisted of transaction data in different categories. But a winning model identified that "Free Shipping = True" was an excellent predictor.
    >
    > What happened here? The point is that free shipping is an *effect* of big spending [because sufficiently large orders get free shipping]. But it's not a good way to model big spending, because in particular it doesn't work for new customers or for the future. Note: timestamps are weak here. The data that included "Free Shipping = True" was simultaneous with the sale, which is a no-no. We need to only use data from beforehand to predict the future.

- How can you deal with target leakage?
  - There is no general method for dealing with target leakage; it boils down to being careful with your analysis and ensuring that you understand every part of the modeling process as well as possible.

Classical statistics

- What is the mode of a set of values?
- What is the median of a set of values?
- What is the mean or expected value of a random variable?
- What is the variance of a random variable?
- What is the variance of a random expressed variable expressed as the difference of two quantities?
- What is the probability of sampling any particular specific value from a continuous probability distribution?

Theorems

- What is the law of large numbers?
  - The law of large numbers says that if you repeatedly sample a random variable, the mean of your samples will converge to the expected value of the random variable as your sample size becomes arbitrarily large.
- What is one example of the real-world significance of the law of large numbers?
  - The profits of casinos are guaranteed via the law of large numbers. Each game which casino-goers can play has positive expected value (for the casino), so even if some people get very lucky and win a lot of money from the casino every now and then, the casino still makes a positive amount of money per player given that sufficiently many players go to the casino and play its games.
- What is one common misconception about the law of large numbers?
  - The "gambler's fallacy" is a common misconception about the law of large numbers. The fallacy refers to the *mistaken* belief that the mean of a *small* number of samples from a random variable must coincide with the expected value of the variable or that a streak of unnaturally high results must be "balanced out" by a streak of unnaturally low results.
- What is the central limit theorem?
  - The central limit theorem says the following: Suppose we have a large number of independent random variables $X_1, X_2, \ldots, X_n$. We can take a sample from each random variable and consider the *mean* of those samples. Suppose that we repeat this process many different times; the central limit theorem says that the generated means will follow a Normal distribution.

- What is one example of the real-world significance of the central limit theorem?
  - The central limit theorem guarantees that the number of heads in n flips of a fair coin follows a normal distribution. To make this clear, we can think of each random variable X_i as denoting the outcome of the ith coin flip (in a series of n flips), taking on a value of 1 for heads and 0 for tails. For any given series of n flips, taking the mean of the samples from $X_1, X_2, \ldots, X_n$ gives the proportion of heads in those n flips. The central limit theorem then guarantees that the proportion of heads and therefore the *number* of heads in n flips is Normally distributed.

    Another example is a random walk on a 1-dimensional number line, where we start at 0 and are equally as likely to step in the +1 direction as we are in the -1 direction. Iterating infinitely many times, the probability distribution of our position is guaranteed by the central limit theorem to follow a Normal distribution. Such a process serves as the basis of a variety of naturally-occurring processes, which

Frequentist inference

- What is a null hypothesis?
- What is a p-value?
- What is the typical threshold for statistical significance?
- What is a t-test?
- How do a one-sample and two-sample t-test differ?
- How do a one-sided and two-sided t-test differ?
- If the p-value for a two-sided t-test is equal to C, what is the p-value for the corresponding one-sided t-test?
- If you obtain a p-value for a two-sided t-test slightly greater than 0.05, would it be appropriate to use a one-sided t-test instead to attain statistical significance?
- What is one possible explanation for a high p-value?
- What is the multiple comparisons problem?
- What is one example of a real-life situation where you might encounter the multiple comparisons problem?
- How can you correct for the multiple comparisons problem?
- What is a confidence interval?

Bayesian inference

- What is Bayes' rule?

- What is one difference between the Bayesian and frequentist interpretations of probabilities?
- What is one difference between Bayesian inference and frequentist inference?
  - Frequentist inference often returns a specific point estimate for the value of a parameter of interest (*e.g.*, the distribution of the mean) along with a confidence interval. In contrast, a Bayesian method explicitly models the prior distribution and therefore will give a full posterior *distribution* for the parameter of interest.

    Many frequentist methods can be interpreted in terms of being roughly equivalent to a Bayesian method with an implicit prior, that is, the method actually *does* assume some sort of prior distribution. (This is analogous to how regularizing a linear model is the same as imposing a Bayesian prior for the coefficient estimates to be equal to 0, where the prior is of the form of a Laplacian or Gaussian function peaked at 0.)
- Is Bayesian or frequentist inference typically more computationally expensive?
  - Because Bayesian methods might have to handle a complex prior distribution and then calculate an entire posterior distribution, they are generally more computationally expensive than frequentist methods for statistical inference.
- What numerical methods are used to handle Bayesian inference?
  - Monte Carlo Markov Chain methods (like the Metropolis-Hastings algorithm) are used to perform the numerical optimizations necessary for training Bayesian models.

A/B testing

- What do the "A" and "B" stand for in an A/B test?
- What variables need to be taken into account to determine the length of an A/B test?
- Suppose that you run an A/B test to test an improvement to your signup button and that you run a separate A/B test for each of the 50 states. What p-value should be your threshold for statistical significance?
- What is an A/B/n test?
- Will an A/B/n test take more or less time than an A/B test?
- What is one disadvantage of using an A/B/n test compared to a series of multiple A/B tests?

Distributions

- What is a uniform distribution?
- What does the Bernoulli distribution model?
- What is one real-world example of a quantity modeled by a Bernoulli distribution?
- What is the probability density function of a Bernoulli distribution with success probability p?
- What is the expected value of a Bernoulli distribution with success probability p?
- What is the variance of a Bernoulli distribution with success probability p?
- What does the binomial distribution model?
- What is one real-world example of a quantity modeled by a binomial distribution?
- How does the binomial distribution relate to the Bernoulli distribution?
- What is the probability density function of a binomial distribution with n trials and success probability p?
- What is linearity of expectation?
- What is the expected value of a binomial distribution with n trials and success probability p?
- What is the variance of a binomial distribution with n trials and success probability p?