CSE/STAT 416 - Topics for Final Exam

Instructor: Hunter Schafer *Date :* 05/22/2023

1 Regression

1.1 Understanding the Task

1. Understand difference between input variables and response variable
2. Understand input variables vs. features
3. Understand how to include categorical features using one-hot encoding (this was discussed in the context of classification, but applies to regression as well)
4. Understand the form of a linear regression equation (interpret the weights)
5. You need to specify a model form (example is choosing which features to include)

1.2 Fitting a regression model

1. Need to find the best weights given a chosen model form
2. Pose the problem as minimizing an objective
   (a) We often use MSE or RSS. $\sum_{i=1}^{n} \left(y_i - w^T x_i\right)^2$
   (b) A "Robust" loss function is less sensitive to outliers.
3. Understand benefit of general method such as gradient descent.

1.3 Assessing the model

1. Importance of generalization (perform well on data we haven't seen yet)
2. Training set, testing set, validation set
   (a) What happens to training vs. testing error with more training data
   (b) What happens to training error vs. testing as the model gets more complex (higher order features, more features, larger weights)
3. Be able to explain the bias-variance tradeoff
   (a) Relationship between variance and stability
   (b) Why is a model that is too simple bad? Why is a model that is too complex bad?
4. Understand error comes from bias, variance, and noise
5. **Major theme**: understand overfitting and why we want to avoid it
6. Understand methods for avoiding overfitting: cross validation, regularization
7. Use these concepts to choose the best model

1.4 Regularization more specifically

1. More automatic way of avoiding overly complex models
2. General technique: instead of minimizing $Loss(w, x, y)$, we minimize $Loss(w, x, y)+\lambda r(w)$, where $Loss(w, x, y)$ assesses model error and $r(w)$ penalizes very large weights.
3. Understand effect of big or small value of λ
   (a) Regularization path graph
4. Special examples: Ridge Regression and Lasso Regression. Understand how these methods are similar and how they are different.
   (a) Ridge is L2, Lasso is L1. The consequence of this is that Lasso prefers sparse solutions, which are more interpretable

2 Classification

2.1 Understand the task

1. In this class we are usually doing binary classification: predict if a datapoint belongs to one of two categories. For convenience, we label these as -1 and +1 (negative and positive, false and true, etc)
2. Examples of models: Nearest neighbors, linear classifier (one is logistic regression), Decision Tree

2.2 Logistic Regression

1. What we really want to minimize is our error (our false positives and false negative). But these loss functions are not continuous
2. Continuous functions are much easier to optimize since we can take the derivative. Let's use a continuous function to model the probability of being in the positive class; use a transformation of a linear function (**sigmoid function**) to make sure output is between 0 and 1.
3. Find the probability function that maximizes the likelihood of the data we actually saw: gradient ascent. Understand how we use **maximum likelihood estimation** for this task, and understand the intuition for maximum likelihood estimation in more general settings.
4. Just as in linear regression, we want to avoid overfitting and we can add regularization. Bias variance tradeoff still applies
5. Overfitting / weights too large leads to overconfident predictions

2.3 Assessing the model

1. Know how to read a confusion matrix: in some settings we might think false negatives are more "severe" than false positives
2. Understand precision and recall
3. ROC curve: we have a tradeoff between precision and recall. From these curves, can you tell if one model is "better" than another?

3 Societal Impacts of ML

3.1 Understand sources of bias

1. Describe how a particular problem in an ML system might stem from a type of bias
2. Assess whether a potential approach to solving a problem in an ML system will be effective

3.2 Definitions of fairness

1. Select an appropriate definition of fairness for a desired result
2. Analyze if a definition of fairness is met under a particular system

3.3 Worldviews of Fairness

1. Describe potential ways in which fairness and accuracy can contradict each other
2. Explain how a particular notion of fairness fits into a worldview of fairness

4 Decision Trees and Ensembles

4.1 Understand the basic decision tree algorithm

1. Understand that it breaks input space into rectangles, and predictions are constant within these regions
2. Be able to read a tree plot and figure out a prediction for a given input
3. Cons of the greedy approach to decision trees: functions such as XOR might be missed

4.2 Bias variance tradeoff

1. STILL applies (as trees grow in depth, they become very complex and they overfit. But shallow trees have bias)
2. To avoid overfitting, we could stop growing tree early. Or we could grow a big tree and then prune.

4.3 Building ensembles of decision trees

1. Bagging: primary goal is variance reduction
   (a) Build many complex models. Each individual has high variance. But have each model built to different random data subset. Averaging over all the models gets reduces the variance
   (b) Build lots of models, all to different subsamples of the data. Final model returns average or majority of predictions from all the individual models.
   (c) Can be built in parallel.
2. Boosting: primary goal is bias reduction
   (a) Build many weak models. Each individually has high bias. But have each model focus on the points that the last one got wrong.

   (b) Must be built sequentially because each model must take into account the points that the previous model got wrong.
   (c) Final prediction is weighted sum of all the individual predictions.
   (d) Extremely successful in practice
   (e) Look through the AdaBoost algorithm and make sure you understand the intuition and key points.
3. General techniques that can be applied on models other than decision trees.

5 Retrieval and Clustering

5.1 Nearest neighbor methods

1. Problem: you're given an object and you want to find similar objects to it.
   (a) For example, for a given news article, find other articles that are similar/related to it.
2. If data are vectors, we can compute distances between points.
   (a) If data are not vectors (documents, images, etc),we can decide on a way to represent data as vectors (tf-idf, vectors of pixels, etc).
   (b) Distance between points could be Euclidean distance, but could also be manhattan, cosine, etc.
3. Nearest neighbor search: pick the points with the smallest distance to the query point.
   (a) This is SLOW: need to compute distance between query points and every single other point.
   (b) Solution: Locality sensitive hashing: smart way to store data in bins such that, with high probability, two points that are actually neighbors will end up in the same bin
   (c) How? Divide space randomly lots of different times. A single line could accidentally divide two neighbors into opposite bins; unlikely that multiple lines will all coincidentally fall between two neighboring points.
   (d) If we give each point a 0 or a 1 corresponding to which side of the line it falls on for every line, we have a binary code for every data point. Neighboring points are very unlikely to be off by more than one digit in these codes (their codes will have Hamming distance of 0 or 1 to one another).
4. Nearest neighbor for regression/classification.
   (a) For example, if I want to predict the price of a house, just find the most similar house (or k houses) in our training dataset, and use its price.
   (b) Vulnerable to overfitting. Need lots of data, and it's best to consider multiple nearest neighbors.
   (c) Kernel regression: all data points contribute to the prediction, with the closer points having more influence/weight

5.2 K-means clustering

1. We want to form clusters of datapoints that are similar to one another.
2. K-means algorithm is the most popular. Understand how it works.
   (a) Start with random cluster centers
   (b) Repeat until convergence:
      i. Assign each data point to its closest cluster center
      ii. Update cluster center to be the mean of the points that were assigned to it
3. K-means only converges to local optimum and is sensitive to how it's initialized.
4. K-means++ tries to improve initialization
   (a) Choose a random data point for the first cluster center
   (b) Repeat k times: Pick a data point to be the next cluster center. The probability of any point being the next cluster center is proportional to its squared distance to the nearest pre-existing cluster center. Thus we're more likely to pick a cluster center that is far away from all other centers.
5. Can only produce cluster boundaries that are lines, and produces "round" clusters (since each point is assigned to its closest cluster center).
6. Uses sum of distances to cluster center as a cluster quality of metric, which is not always good if the cluster isn't round and has a weird shape.

5.3 Document similarity in particular

1. Bag of words vs. tf-idf
   (a) Tf-idf is good because it down-weights importance of very common words. Don't want to say that two documents are similar because they both contain "the" a lot of time.
   (b) Normalize so that document length does not affect similarity. You don't want preference towards long or short documents.
   (c) Review concepts from assignment and concept quiz.

5.4 Other Clustering Methods

1. Hierarchical clustering
   (a) Understand what a dendrogram is
   (b) Divisive: start with all points in one cluster, and recursively split
   (c) Agglomerative: start with every point in its own cluster, an repeatedly merge closest clusters.
      i. In single-linkage, distance between clusters A and B is defined to be the closest distance between any point in cluster A and any point in cluster B.

6 Dimensionality Reduction

6.1 Motivation

1. Embed complex data in a lower dimensional space such that similarity is preserved and redundancy is eliminated.
2. We like lower dimensions for visualization, and for efficient storage / training. Also reduces variance of models; protects against overfitting,
3. If I project d-dimensional data onto d-dimensions, I can reconstruct the data perfectly. I did not lose any information, I just "rotated" my data.
4. If I choose $k < d$, I will definitely lose some information when I project my data. How can I choose the projection lines such that I lose as little information as possible? One method is principal components (see below)
5. You need to pick k. The best choice depends on the intrinsic structure of your data.

6.2 Principal Components

1. Project onto the direction of maximum variance; if we keep as much variance in the data as possible, we lose as little information as possible.
   (a) Second principal component = direction of maximum remaining variance after accounting for the first principal component, and subject to the constraint that the second principal component must be orthogonal to the first. Third, fourth, fifth, follow same pattern. We fix ourselves in the space that is perpendicular to first principal component, and then search for direction of maximum variance.
2. Comes from eigenvalues and eigenvectors- don't worry if you have not taken a course in Linear Algebra, you just need to understand the intuition.
3. Understand limitations; you must pick K = number of principal components. If the data does not have intrinsic lower dimensional LINEAR structure, you may not get meaningful results.
4. Understand how you would draw the principal components in a 2D scatter plot (see pictures from lecture or section slides)
5. We can also think of principal components as eliminating all redundancy in our data, because it makes new features that are perfectly uncorrelated with one another.

7 Recommender Systems

7.1 Co-occurrence based recommendation systems

1. Understand the input: a ratings matrix. Rows are users and columns are products. There are many missing entries because not every user has tried every product
   (a) Matrix is (Num Users)x(Num Products)
2. Understand the output: A user matrix and a product matrix.
   (a) L is the user matrix. (Num Users) x (K)
   (b) R is the product matrix. (Num Products) x (K)

   (c) K is the number of hidden topics; we must choose this (avoid overfitting!!!)
   (d) We want to find L and R such that L*R' comes as close as possible to reconstructing the original ratings matrix
   (e) If we take the i^{th} row of L and the j^{th} column of R', the dot product tells us predicted rating for user i on product j
3. Understand the matrix completion algorithm.
   (a) If we treat the user matrix (L) as fixed, we do ordinary least squares to learn the product vectors (R). If we treat the R as fixed, we do ordinary least squares to find L.
   (b) This is a form of **coordinate descent (optimizing one parameter at a time while holding other parameter fixed)**: understand how this iterative approach is different from gradient descent
4. Understand the cons of this approach
   (a) Be able to describe the cold start problem
   (b) Understand how course themes such as local optima and overfitting fit into this context.

7.2 Broad level: Compare Co-occurrence approach to popularity approach or feature-based approach

1. Can combine these approaches
2. Understand how we might evaluate a model in the context of precision and recall
3. Connect to broader course themes; how should we pick K? How should we pick regularization? Evaluate final model?

8 Deep Learning

8.1 Relationship between Neural Networks and Linear Classifiers

1. We are just building a linear classifier with very complex non-linear features
2. We use the first few layers of a network to cleverly form linear combinations of inputs (which are then squashed with an activation function to form non-linear functions of inputs). This step is feature extraction, and it is really the key contribution of deep networks. The last layer just performs simple classification on the transformed input.
3. Features are learned from data (the neural network learns features that help it perform its task). They aren't manually designed.

8.2 Training:

1. Gradient descent; be careful of local optima; tuning and initial conditions matter.
2. Backpropagation: Using the calculus chain rule, we can compute the gradient of the loss function with respect to every weight in the network. This tells us how to adjust each weight to reduce the loss. You won't be asked to compute the derivatives by hand. Nowadays, Python libraries do it for you anyways.
3. Time intensive in a deep network

8.3 Transfer Learning:

1. The bulk of the work in training a deep network comes with the feature extraction task (first many layers). If a bunch of different people want to build image classifiers with different goals (i.e. animal classification vs. facial recognition vs. handwriting recognition), they cannot use the same network. But, they can probably share the same feature extraction layers. Only the last few layers are specialized to the task at hand.
2. To do image classification, I can use the already learned weights from someone else's deep network to extract features from my images. Then, I can use a simple linear classifier on these learned features to get my results.
3. The "transfer" part of transfer learning means sharing the feature extraction layers across many tasks.

8.4 Convolutions for processing images

1. Understand how convolutions work.
   (a) For images, we slide small 2-D filters (for example, 3 x 3) across the image. At every position, compute the dot product (multiply element-by-element and add them up) of the filter and the image at that position
   (b) Max-pooling layers are often inserted between convolution layers
2. Understand why a convolutional neural network might be superior to a plain old network in some cases.

8.5 Adversarial examples

1. Since neural networks often overfit, adversaries can exploit this to generate examples that will be misclassified by the neural network
   (a) Use gradient ascent to figure out how to modify the input image to minimize the probability of it being classified correctly
2. One approach to prevent this: Add random noise to your data so that the model is robust to adversarial image attacks

9 Misc Topics

9.1 Generative AI

1. TBD