CS7641 ML Lecture Summaries
Chapter 2: Unsupervised Learning

Kyle Nakamura

2.1 Randomized Optimization

2.1.1 Optimization

The lecture is about optimization in machine learning and its importance in finding the best solution for a given problem. Optimization involves finding the value of an input (x*) that maximizes the fitness function or objective function (F) in a given input space (X). The goal is to find the value of x* that is closest to the maximum possible value. Randomized optimization algorithms are discussed as a way to improve the effectiveness of optimization. The lecture mentions examples of optimization problems in real-life scenarios such as chemical engineering, process control, route finding, and neural networks. It is noted that optimization is a fundamental component of machine learning, as finding the best parameters or structure for models involves optimization.

2.1.2 Optimize Me Question

In the lecture on randomized optimization, the instructor begins by introducing a quiz called "Optimize Me." The quiz consists of two different optimization problems, each with its own input space and function. The first problem involves an input space of values ranging from one to 100. The function to be optimized is x mod 6 squared mod 7 minus the sin of x. The second problem has an input space of all real numbers and the function to be optimized is negative x to the 4th plus 1000x cubed minus 20x squared plus 4x minus 6. The instructor encourages the students to find the optimal solution or a solution that is close to optimal.

2.1.3 Optimize Me Solution

In this lecture, the instructor discusses how to optimize a function by finding the maximum output value. For the first problem, the instructor suggests writing a program to enumerate all possible values of the function and choose the one with the highest output. The function is described as complex and unfriendly, resembling an electric bow tie. The instructor suggests using the mod function and sine wave analysis to reason through the problem, but it is not effective in this case. For the second problem, where the space of X values is large, the instructor suggests using calculus to find the maximum by taking the derivative and setting it equal to zero. However, the resulting equation is a cubic function, making it difficult to solve directly. The instructor suggests using a graphical approach to zoom in on the function and estimate the maximum value. They also mention Newton's method, which involves using the derivative to take steps towards the maximum value and converge on the peak. The lecture ends with an estimation of the maximum value being around 750.

2.1.4 Optimization Approaches

In terms of optimization approaches, the lecture discusses two main ideas: generate and test, and analytical solutions. Generate and test is useful when there is a small input space, and especially helpful for complex functions. Analytical solutions, such as solving for the optimum analytically, are only feasible when the function has a derivative. However, there are cases where the function may not have a derivative or where the derivative is difficult to find. In such situations, randomized optimization, the topic of the lecture, becomes necessary. Additionally, it is important to consider that even with a derivative, Newton's method can get stuck at local maxima, so the presence of multiple local optima should be taken into account.

2.1.5 Hill Climbing

Hill climbing is an algorithm used to find the maximum value of a function. It starts by guessing an initial value for x and then explores the neighborhood of that point to find a better value. The algorithm moves in the direction of the neighbor with the highest function value, continuously moving to higher values until it reaches a peak. At a peak, it stops and returns the current x value. However, hill climbing can get stuck at local optima, as demonstrated by an example of starting on the other side of a valley. It is important to note that it can also get stuck at the lowest peak.

2.1.6 Guess My Word Question

The lecture discusses a hill climbing approach to the "Guess My Word" problem. The problem involves guessing a 5-bit word, with the lecturer providing the number of correct bits for each guess. The lecture

introduces the concept of a neighbor function, which generates variations of the current guess. The lecture explains that starting at all zeros and considering all possible neighbors leads to a more efficient solution. The lecture mentions that the algorithm pseudocode does not consider this optimization, but it can be implemented if the neighborhood function is symmetric. The lecture concludes by suggesting that with enough information, a smart human could jump to the optimal solution.

2.1.7 Guess My Word Solution

The lecture discusses the solution to a problem of finding the correct sequence of bits. It is mentioned that the fitness function used in the problem allows for incrementing the number of correct bits and has a global optimum. The lecture also explores how the sequence can be determined without extra information, using the common bits and their proximity to the target sequence. The lecturer emphasizes the importance of understanding the structure of the problem space and mentions that in more complex scenarios with larger sequences and unknown fitness functions, additional techniques would be needed to handle local optima.

2.1.8 Random Restart Hill Climbing

Random Restart Hill Climbing is a method used to overcome the problem of getting stuck in local optima when using hill climing algorithms. It involves restarting the algorithm from a different random starting point once a local optimum is reached. The advantage of this approach is that it increases the chances of finding the global optimum by exploring different starting points. The cost of the algorithm is multiplied by the number of random restarts performed. However, if there is only one optimum and no local optima, the algorithm will keep producing the same result. In such cases, it may be necessary to stop the algorithm or ensure that the next random point is far away from the previous starting point to explore the entire search space. The algorithm assumes the ability to make local improvements, as local improvements contribute to global improvements. The distinction between local and global optima is based on the idea that the local optima is a point where one does not want to be, whereas the global optima is the desired point.

2.1.9 Randomized Hill Climbing Quiz Question

The lecturer introduces a randomized hill climbing quiz to explore the advantages and outcomes of using hill climbing to find the maximum. The input space is 1 to 28, and the fitness function is defined. The quiz involves running randomized hill climbing with a simplification of the algorithm. The algorithm randomly chooses a direction to move if both directions improve and checks the neighbors for improvement. If no improvement is found, it declares a local optimum. Once a local optimum is reached, a random restart is triggered. The goal is to determine the average number of function evaluations needed to find the maximum. The lecturer suggests writing code or using math to solve the quiz and emphasizes the importance of understanding basins of attraction.

2.1.10 Randomized Hill Climbing Quiz Solution

In this lecture, the concept of randomized hill climbing as a method for finding the global optimum in a search space is discussed. The speaker evaluates the number of steps required to reach a local optimum from different starting positions and determines the average number of steps to reach the global optimum. It is found that for 22 out of 28 starting positions, the average number of steps to reach a local optimum is 5.39. For 4 out of 28 starting positions, the average number of steps to reach the global optimum is 4. In 2 out of 56 cases, the algorithm gets stuck at a non-global local optimum, requiring 10 steps. In 1 out of 56 cases, the algorithm takes 5 steps to reach the global optimum. The speaker notes that a simple linear algorithm that evaluates all possible inputs could be more efficient than randomized hill climbing. However, it is suggested that faster random restarts and keeping track of visited points could improve the algorithm's performance. It is also mentioned that the effectiveness of randomized hill climbing depends on the size of the attraction basin around the global optimum. If the attraction basin is large, the algorithm can quickly reach the global optimum. Overall, randomized hill climbing may not always be the most efficient method but has the potential for significant improvement in certain cases.

2.1.11 Simulated Annealing

Simulated Annealing is an algorithm that combines exploring and exploiting in order to find optimal solutions. It is related to the concept of overfitting, where only exploiting local information can lead to getting stuck in local optima. The algorithm is similar to an optimization process used in metallurgy called "heating and cooling", where the molecules in a sword align themselves in the same way to make it hard and sharp. In Simulated Annealing, the temperature is simulated to allow for exploration and to find even better solutions.

2.1.12 Annealing Algorithm

The Annealing Algorithm is a simple and effective method for optimization. The algorithm repeats for a finite number of iterations. At each iteration, a new point is sampled from the neighborhood of the current point x, and the algorithm probabilistically decides whether to move to the new point based on a probability function. If the fitness of the new point is better than or equal to the fitness of the current point, the move is made. Otherwise, the algorithm calculates a fitness difference between the two points, divides it by the temperature, and interprets it as a probability. The move is then made with this probability. The temperature parameter affects the decision-making process. When the fitnesses of two points are very close, the probability of making the move is high. If the difference is negative (a big step down), the probability is low. For moderate differences, the probability depends on the temperature. A high temperature allows for more downward steps, while a low temperature limits moves to uphill directions.

2.1.13 Properties of Simulated Annealing

Simulated annealing has two important properties: as the temperature approaches zero, it acts like hill climbing, only making moves that improve fitness, while at infinite temperature, it behaves like a random walk. To transition between these two extremes, the temperature must be decreased slowly, allowing the system to explore at each temperature level. As the temperature decreases, large valleys become boundaries and smaller gullies become traversable, until eventually, even the smaller gullies become barriers. The goal is to give the algorithm a chance to explore high-value points before cooling it too much. In terms of the probability distribution of ending at any given point, it follows the Boltzmann distribution, with higher fitness values having a higher probability of being selected. As the temperature decreases, the distribution becomes more concentrated on the global optimum. However, if the temperature decreases too rapidly, the algorithm may get stuck at suboptimal points. Simulated annealing is considered a "simulated" Boltzmann distribution, using concepts from physics but not exactly following the same conventions.

2.1.14 Genetic Algorithms

Genetic algorithms are a class of randomized optimization algorithms that are useful for finding optimal solutions in spaces that can be represented combinatorially. In genetic algorithms, input points are considered individuals and groups of them form a population. The process involves mutation and crossover. Mutation is like local search, where small changes are made to an individual, and crossover combines attributes from different individuals to create new offspring. These processes are analogous to genetic mutations and sexual reproduction. The population as a whole represents a distribution over individuals, and this distribution can guide the search for higher scoring individuals.

2.1.15 GA Skeleton

Summary: This lecture provides a skeleton of an algorithm for implementing a Genetic Algorithm (GA). The algorithm starts with an initial population of randomly generated individuals. The algorithm then proceeds to compute the fitness of each individual and selects the most fit individuals. Two selection methods are discussed: truncation selection which chooses the top half of the population based on fitness scores, and roulette wheel selection which randomly selects individuals with higher probabilities for those with higher fitness scores. The concept of exploitation versus exploration is mentioned, and the use of temperature parameters for selection is discussed. The algorithm then pairs the most fit individuals and performs crossover and mutation to produce offspring. Crossover is a process of combining genetic information from two parents, and mutation involves making small local changes to the new offspring. The offspring then replace the least fit individuals in the population. Concrete examples of crossover are mentioned but not elaborated upon.

2.1.16 Crossover Example

Crossover is a key operation in genetic algorithms and depends on how the input space is represented. In this example, the input space consists of eight-bit strings. Two parent strings are combined by aligning the corresponding bits. One way to generate offspring is through one-point crossover, where a random number is chosen along the sequence and the bits before and after that point are swapped. This assumes that the locality of the bits and the independent optimization of subparts of the space are important. Another way is through uniform crossover, where each bit position is randomized independently, disregarding the ordering of bits. This is similar to how genes are combined in biology, with each gene independently chosen.

2.1.17 What Have We Learned

Genetic algorithms are a useful tool in randomized optimization, often considered the second best solution to a given problem. They involve representing the input and selecting and fitting a fitness function. Randomized optimization involves taking random steps and starting in random places when a natural gradient step is not possible. Other algorithms discussed include randomized hill climbing, simulated annealing, and taboo search. While these algorithms are amnesic and do not learn about the optimization space, there are other methods that explicitly model the probability distribution over good solutions. These methods may provide better results and cover the space more effectively.

2.1.18 MIMIC

The lecturer begins by summarizing two main problems with existing randomized optimization algorithms: 1) these algorithms only communicate the final point of optimization, disregarding the structure of the search space, and 2) it is unclear what probability distribution is being used. The lecturer shares that they have found a class of algorithms, specifically one called MIMIC, which addresses these two issues. The lecturer had actually written a paper about MIMIC almost 20 years ago and decided to revisit it. MIMIC directly models a probability distribution and performs a search through the space to refine the estimate of this distribution over time. This allows for the communication of structure within the search space and identifies more optimal points. While the lecturer acknowledges that there have been advancements in this area since their paper, they feel that MIMIC's basic idea is still useful and understandable. They share a simplified version of the MIMIC algorithm but note that more mathematically precise refinements have been made in recent decades.

2.1.19 A Probability Model Question

The lecture discusses a probability distribution that is parameterized by theta, which represents a threshold. The distribution is defined as 1 over z sub theta for values of x that have a fitness function greater than or equal to theta, and it is 0 otherwise. The normalization factor, z sub theta, accounts for the probability of choosing x in the space of high-scoring individuals above the threshold. This probability distribution is uniform over all values of x above the threshold and ignores values below it. The lecture then presents a quiz with two questions regarding the probability distributions when theta is at its minimum and maximum values.

2.1.20 A Probability Model Solution

The lecture discusses the concept of a probability model solution in the context of randomized optimization. The speaker explains that the probability distribution assigns a probability of one to the single point that represents the unique optimum of a function. In the case of multiple optima, the distribution is uniform over all of them. The speaker further states that the distribution for the minimum achievable value of the function assigns uniform probability to all points in the input space. The goal is to estimate the distribution P(sub theta) of X by starting with a uniform distribution and gradually improving the estimates to converge to a distribution of only the optimal points. The lecture emphasizes the importance of being able to sample from the set of optima, as it facilitates finding an optimum. The speaker concludes by stating the need to develop an algorithm to achieve this goal.

2.1.21 Pseudo Code

The lecture discusses a randomized optimization algorithm that generates samples according to a probability distribution and updates the distribution based on the best samples generated. It is similar to genetic algorithms, but instead of a population moving from one bit to another, samples are generated and the best ones are retained. The lecture emphasizes the importance of maintaining structure in the probability distributions and explains that the theta parameter represents a threshold for fitness values. The algorithm iteratively updates the distribution until convergence. The lecture also mentions the need for accurate estimation of the probability distribution and the importance of generating samples that match the next distribution. The algorithm aims to move from the lowest fitness values to the highest by manipulating the representations of the distributions.

2.1.22 Estimating Distributions

In this lecture on estimating distributions, the speaker discusses the challenge of estimating joint probability distributions due to their exponential size. They propose using dependency trees, a special case of Bayesian networks, to estimate distributions by assuming conditional independence. The speaker explains that a dependency tree is a directed graph where each node (representing a random variable or feature) has exactly one parent. By representing the joint distribution as a product over features dependent on their parents, the speaker highlights the advantage of smaller conditional probability tables. The size of the entire distribution is linear in the number of features, allowing for easier sampling. The speaker acknowledges the need to estimate the distribution and select the best tree in the algorithm. Dependency trees are chosen as the representation because they capture relationships while maintaining simplicity. The speaker relates this approach to crossover in genetic algorithms, emphasizing the ability of dependency trees to capture similar relationships without relying on locality. The lecture concludes by mentioning that the next step is to learn the dependency tree from samples, which involves mathematical techniques.

2.1.23 Finding Dependency Trees

Finding dependency trees involves representing a probability distribution. The distribution is estimated using a distribution called p-hat that depends on a parent function. The goal is to find the best dependency tree that closely represents the underlying distribution, and the measure of similarity used is the KL Divergence. The KL Divergence measures the divergence between the true distribution and the estimated distribution. The closer they are, the smaller the divergence. The Kullback-Leibler divergence is not a distance but a measure of divergence. The objective is to minimize the KL Divergence, which leads to a simple function that includes the entropy of the underlying distribution and the conditional entropies. The goal is to find the optimal dependency tree that minimizes the entropy for each feature, given its parents. Choosing parents that provide valuable information about the feature values is important.

2.1.24 Finding Dependency Trees Two

The lecture discusses an optimization problem related to finding dependency trees. The goal is to minimize a cost function, which is defined as the sum of conditional entropies of each feature given its parents. The lecturer introduces a new cost function by adding the sum of unconditional entropies to the original cost function. This new cost function is equivalent to maximizing mutual information between each feature and its parents. The lecturer explains that mutual information is a symmetric measure compared to conditional entropy, making it easier to work with. The lecture concludes by stating that finding the best dependency tree is accomplished by maximizing the sum of mutual information between each feature and its parent. The optimization process is described as being relatively easy.

2.1.25 Finding Dependency Trees Three

In this lecture, the professor discusses the concept of finding a dependency tree that maximizes the sum of mutual information between every feature and its parents. The professor explains that this can be achieved by finding a maximum spanning tree in a fully connected graph, where each node represents a feature and the edges represent the mutual information between them. The professor clarifies that a maximum spanning tree is the same as a dependency tree, and it can be found by applying the Prim's algorithm to the fully connected graph. Additionally, the professor mentions that the directed structure of the tree can

be achieved by selecting a root node and performing a depth-first reversal. The use of Prim's algorithm is particularly effective for this task as it efficiently handles densely connected graphs.

2.1.26 Back to Pseudo Code

This lecture focuses on the pseudo code for the MIMIC algorithm and its application in generating samples from a probability distribution. The algorithm aims to find the best dependency tree (Pi function) by generating samples from it. Samples are generated by starting from a node and generating according to the unconditional distribution, then generating samples from each node according to its conditional distribution given its parents. Conditional probability tables are obtained from the mutual information graph, which provides estimates of unconditional and conditional probability distributions for each feature. The process of generating samples and building the mutual information graph yields all the necessary probability tables. Generating samples can be done in linear time with the number of features. The lecture also mentions that any single node can be chosen as the root in the generated undirected graph to induce a specific directed tree. The process of generating samples and estimating the next one using the maximum spanning tree over all the mutual information is repeated to improve the algorithm. The lecture states that dependency trees are powerful due to their ability to capture relationships within the probability distribution while avoiding exponential cost in estimation.

2.1.27 Probability Distribution Question

The lecture begins with a quiz about probability distributions in the context of the "mimic" algorithm. Three problems are presented: maximizing the number of 1s in a binary string, maximizing the number of alternations between bits in a binary string, and minimizing the number of two-color errors in a graph. The three probability distributions are a chain, a dependency tree, and complete independence. The chain represents a specific ordering of bits, the dependency tree represents an unknown dependency structure, and complete independence assumes no relationship between variables. The lecture emphasizes the importance of the number of parameters to estimate and the potential for overfitting with different probability distributions. The goal is to match the appropriate probability distribution with each problem.

2.1.28 Probability Distribution Solution

In this lecture, the professor discusses the probability distribution solution for maximizing the number of 1s and alternations in a binary string. The professor explains that for the problem of maximizing the number of 1s, the fitness values of the bits are all independent, so capturing other dependencies is not necessary. The probability distribution is uniform for all values greater than or equal to a certain threshold, and the distribution can be represented by sampling each bit independently and uniformly. The professor explains that this distribution can represent the minimum and maximum values, but it may not accurately capture values in between. The professor uses an example with four bits to demonstrate different ways of obtaining fitness values of 2, 3, and 4. The distribution is an approximation, but it still has a good chance of generating values greater than the threshold. For the problem of maximizing alternations, the professor explains that knowing the value of the neighboring bits is important, and using a chain representation provides this information. The professor concludes by mentioning that mimic, a type of randomized optimization algorithm, performs well on problems with complex graph structures because it captures the underlying structure. The professor emphasizes that even though there may be multiple solutions, they all share a common relationship to the underlying structure.

2.1.29 Practical Matters

Mimic is a randomized optimization algorithm that performs well when optimal values depend on structure rather than specific values. It can get confused by different values that are both optima but look different. Representing everything in between in probability space is crucial for a successful search. Mimic can get stuck in local optima but can be restarted. Mimic tends to run orders of magnitude fewer iterations compared to other algorithms like Simulated Annealing. However, each iteration of Mimic can be more time-consuming than Simulated Annealing. The decision to use Mimic depends on the trade-off between fewer iterations and the computational cost of each iteration.

2.1.30 Practical Matters Two

MIMIC (Mutual Information Maximizing Input Clustering) provides structure and more information for each iteration at the expense of building dependency trees and estimating probability distributions. MIMIC is effective when the cost of evaluating the fitness function is high, making it beneficial to minimize the number of iterations. Fitnes functions that involve complex simulations or human evaluation can benefit from MIMIC. When choosing algorithms, trade-offs include overfitting and the space and time complexity of different models. Overall, understanding sample complexity, time complexity, and space complexity is important in selecting the appropriate model.

2.2 Clustering

2.1.31 What Have We Learned

The lecture on clustering discussed the concept of structure fitting and its relevance to the overall structure. The speaker expressed gratitude to the listener, Michael, for the opportunity to share their knowledge and address a concern. The conversation ended with a farewell.

2.2.1 Unsupervised Learning

This lecture introduces the concept of unsupervised learning in machine learning. Unsupervised learning involves making sense of unlabeled data, whereas supervised learning uses labeled training data to generalize labels to new instances. Unsupervised learning is not as unified or well-defined as supervised learning. It focuses on data description, finding a more compact way to describe the data.

2.2.2 Basic Clustering Problem

The basic clustering problem in unsupervised learning involves taking a set of objects and grouping them based on their relationships, represented by inter-object distances. The objects don't have to be in a metric space, and the distances don't need to obey the triangle inequality. Clustering algorithms aim to create a partition, where objects assigned to the same cluster have the same output of the partition function.Possible trivial clustering algorithms include putting all objects in the same partition or assigning each object to its own partition. The preference for a specific partitioning is not specified in the problem definition.

2.2.3 Single Linkage Clustering Question

Clustering is algorithm-driven and there are different algorithms for performing clustering. Single linkage clustering is a simple and natural algorithm that has been widely used. In single linkage clustering, each object is initially considered as a cluster and the intercluster distance is defined as the distance between the closest two points in the two clusters. The algorithm iteratively merges the two closest clusters until the desired number of clusters is reached. In this example, clusters A and B are merged first, followed by clusters C and D. Then, clusters C, D, and E are merged together. The next step is to determine the closest pair of clusters, which could be either E to C and D or G to A and B. Finally, the next merge would depend on the chosen closest pair.

2.2.4 Single Linkage Clustering Solution

In this lecture, the speaker discusses single linkage clustering. They start by discussing how to measure distances on a flat screen and consider various points that are closest to each other. They conclude that points e and f, as well as points d and f, are the closest. However, they later realize that d and f are actually farther apart, and points e and f are closer together. They mention that any of these options would be acceptable for clustering. The speaker also mentions that this realization will not change the final answer. They indicate that they will continue with the clustering process now that the quiz explanation is complete.

2.2.5 Single Linkage Clustering Two

In this lecture on single linkage clustering, the speaker discusses the process of merging clusters based on their intercluster distance. Starting with clusters D and F, it is noted that since they are already in the same

cluster, their intercluster distance is zero. The speaker then suggests merging clusters B and G, resulting in the merging of all the clusters. The lecture goes on to explain the representation of these merges using a hierarchical structure, providing a tree-like visualization. This hierarchical structure can provide additional information about the connections between clusters. The speaker mentions that by cutting the tree at different points, different cluster structures can be obtained. The concept of further versus farther is briefly discussed, and the speaker answers a question about the possibility of using different measures for intercluster distance, such as average or farthest distance. It is mentioned that single linkage clustering specifies the use of the closest distance, but other methods like average and max linkage clustering exist. The lecture concludes by noting that the definition of intercluster distance is a parameter that can be influenced by domain knowledge.

2.2.6 Running Time of SLC Question

Single-link clustering has several advantageous properties. Firstly, it is deterministic, meaning that running the algorithm will always produce the same result, assuming there are no ties in distances. This eliminates the need for randomized optimization. Additionally, when performing single-link clustering in a space where distances represent edge lengths of a graph, it is equivalent to a minimum spanning tree algorithm. Furthermore, the running time of single-link clustering can be characterized in a relatively friendly manner. Consider a scenario where there are n points or objects to be clustered and K clusters are desired. In this case, the most suitable running time characterization for single-link clustering can be determined.

2.2.7 Running Time of SLC Solution

The lecture discusses the running time of the SLC (Single Link Clustering) solution. The speaker mentions two different approaches to finding the running time, one of which is considered "lame." It is noted that very small and very large values of k are easy to compute, but the middle range is challenging. The speaker suggests that any function directly dependent on k, which becomes harder as k gets bigger and easier as k gets smaller, is likely incorrect. Therefore, it is suggested that the worst-case running time is on the order of n cubed. The algorithm works by finding the two closest points at each step and merging clusters that have different labels. The finding of the closest points takes n squared time because all possible combinations need to be considered. It is possible to optimize this by cleverly storing distances, but the running time is still characterized as n squared. The lecture also mentions that it is not necessary to merge clusters explicitly, but rather to find the closest two points with different labels. The lecture then discusses the possibility of using fancier data structures, such as Fibonacci heaps or hash tables, to optimize the finding of the closest points without considering all points repeatedly. The speaker mentions that there have been PhD-level studies on clever methods to avoid considering all points, such as dividing points into smaller boxes. Overall, the lecture concludes that the running time of the SLC solution is likely n cubed, although it is possible to improve upon this slightly. The other suggested answers, such as linear time or quadratic time, are deemed incorrect based on the computation of the closest distances and the structure of the problem.

2.2.8 Issues With SLC Question

The speaker presents a quiz to practice the concept of single-link clustering and to challenge assumptions about clustering. A set of points is shown in the upper left corner and the question is which pattern will be obtained when running single-link clustering with K equals two. The audience is given time to think and answer.

2.2.9 Issues With SLC Solution

Single Link Clustering (SLC) is a method of clustering where clusters are formed based on the distance between the two closest points in each cluster. In this method, points that are closer to each other than to any other points are merged together. The clusters formed through SLC tend to be stringy because proximity is the main criterion for linking points. In the given example, a set of points on a circle is used for illustration. It is observed that each point on the outer circle is closer to its neighboring points on the circle than to the four points located in the middle of the circle. Starting with these observations, SLC would gradually merge the points together, connecting the outer shell and eventually forming one big cluster. This outcome can be considered unusual as it results in stringy clusters. The lecturer mentions that their preferred outcome would be the bottom right cluster, but the SLC algorithm would not select this option

due to the given value of k (number of clusters). Instead, SLC would walk around the circle and connect clusters, resulting in a different cluster formation, such as the one observed in the lower left.

2.2.10 K Means Clustering

K-means clustering is a clustering algorithm where K represents the number of clusters. The algorithm starts by randomly selecting K points as the centers and then assigns each point to the closest center to form initial clusters. The algorithm then iteratively recalculates the centers as the average of the points in each cluster and reassigns each point to the closest center. This process continues until convergence, where the centers and clusters remain unchanged. In an example, the algorithm is applied to a set of points, with two random centers initially selected. Points are then assigned to the closest center to form two clusters. The centers are recalculated, and the points are reassigned based on proximity. This process repeats, resulting in a final clustering where the centers and clusters stabilize. K-means clustering generally produces more compact clusters than other clustering methods, and it converges to a stable solution.

2.2.11 K Means in Euclidean Space Part 1

The text discusses the notation and algorithm for performing K-means clustering in a Euclidean space. The notation includes a partition of points, denoted as $p^t(x)$, and the set of points in each cluster, denoted as C_i^t. The algorithm starts by selecting initial centers and then assigns each point to its closest center. The centers are then updated by computing the mean or centroid of the points in each cluster. The process iterates, continuously updating the partition and centers. The text suggests that this algorithm can be seen as an optimization algorithm, continually improving the clustering solution.

2.2.12 K Means as Optimization

K-means clustering can be framed as an optimization problem. The goal is to find configurations, or clusters, with high scores. The scores are based on the error introduced by representing points as partitions or centers. One way to measure the error is by computing the sum of the squared distances between the objects and their respective centers. This uses Euclidean distance, as the objects and centers are in the same space. The neighborhood of a configuration can be defined as the set of pairs where the centers stay the same but the partitions change, or vice versa.

2.2.13 K Means as Optimization Quiz Question

In this lecture on clustering, the k-means procedure is examined from the perspective of optimization algorithms. The lecturer mentions three types of randomized optimization algorithms: hill climbing, genetic algorithms, and simulated annealing. The question posed is which of these optimization algorithms is most similar to k-means.

2.2.14 K Means as Optimization Quiz Solution

The speaker argues that the K-means algorithm can be understood as a form of hill climbing, rather than simulated annealing or genetic algorithms. They explain that K-means takes steps towards configurations that improve the overall score, and demonstrate that it indeed maximizes the score and minimizes error.

2.2.15 K Means in Euclidean Space Part 2

- The E function is minimized in the K-Means algorithm when points are moved to different clusters only if it reduces the squared distance to the cluster center. - The error can stay the same if a point remains in the same cluster. - Moving cluster centers will not increase the error, as the average is the best representation of a set of points. - The error equation is reduced by moving points and centers. - The algorithm is guaranteed to be monotonically non-increasing in error, meaning the error never increases as the process iterates. - The finite number of configurations and deterministic definition of cluster centers ensures convergence in finite time. - Breaking ties consistently and always selecting the lower numbered cluster when distances are equal helps prevent spinning in the algorithm. - The number of possible configurations is exponential, but in practice, the algorithm tends to converge quickly due to the nature of distance metrics and close points.

2.2.16 Properties of K Means Clustering Quiz Question

The properties of k-means clustering can be summarized as follows: - Each iteration of the k-means process is polynomial and efficient, involving the calculation of distances between k centers and n points to reassign them, and then running through all points to redefine the clusters. - The number of iterations is finite and exponential, with a total of k to the nth ways of assigning points to k clusters. However, in practice, the number of iterations tends to be low and the clusters converge quickly. - The error on each iteration decreases if ties are broken consistently. In some cases, the clusters may not improve or change, but this can only happen once before convergence is reached. - A specific example is given where six points are divided optimally into three clusters. The task is to provide a way to assign the three cluster centers to the points in such a way that the clustering does not progress or change. The question asks for a comma-separated list of three cluster centers out of the six points (a, b, c, d, e, f) that will result in no progress in clustering.

2.2.17 Properties of K Means Clustering Quiz Solution

In the lecture, the instructor discusses the properties of k-means clustering and strategies to avoid local optima. They go over an example where certain initial points result in three clusters instead of the expected four. To avoid this, the instructor suggests using random restarts or finding initial cluster centers that are spread out by choosing points that are far away from each other. The instructor also mentions that randomly assigning clusters can help spread things out. Overall, the lecture focuses on understanding the behavior of k-means clustering and techniques to improve its performance.

2.2.18 Soft Clustering Quiz Question

The lecture discusses soft clustering as a method for clustering data. It presents a dataset with two groups of points, A, B, and C on the left, and E, F, and G on the right, with point D in the middle. The lecture raises the question of what will happen to point D in a k-means clustering setting with two clusters. The possibilities discussed include point D ending up with the left group, the right group, being randomly assigned, or being shared between both clusters. The correct answer is not explicitly mentioned in the text.

2.2.19 Soft Clustering Quiz Solution

Soft clustering is a type of clustering algorithm that allows for the possibility of points belonging to multiple clusters. In contrast to hard clustering, where a point belongs to only one cluster, soft clustering assigns membership probabilities to each cluster. This means that a point can be partially assigned to multiple clusters based on its characteristics. In the given example, the lecturer discusses a scenario where points are randomly assigned to clusters. Depending on the initial random assignment of cluster centers, the point 'd' can end up in either the left or right clusters. If the random centers are on points 'a' and 'b', point 'd' will be dragged towards the right cluster. Conversely, if the random centers are on points 'f' and 'g', point 'd' will be dragged towards the left cluster as it moves to the left. If the random centers are on points 'a' and 'g', the final cluster assignment of 'd' will depend on how ties are broken. The lecturer mentions breaking ties lexiographically, which would result in 'd' being assigned to the left cluster. However, the lecturer mentions a desire for the answer to be the fourth choice. This introduces the idea of another clustering algorithm that performs soft clustering. With soft clustering, a point can have partial membership in multiple clusters, allowing for a more nuanced representation of data points and their relationships to different clusters.

2.2.20 Soft Clustering

Soft clustering is a technique that allows points to be probabilistically assigned to multiple possible clusters instead of definitively belonging to one cluster. In this approach, data is assumed to be generated by selecting one of K possible Gaussian distributions and selecting an actual data point from that Gaussian. The goal is to find a hypothesis, which is a collection of K means, that maximizes the probability of the data given that hypothesis. This is achieved through maximum likelihood estimation. The specific algorithm for soft clustering is not provided, but it likely involves probability theory and optimization. Soft clustering is a softer, gentler version of K-means clustering and is sometimes referred to as K Gaussians.

2.2.21 Maximum Likelihood Gaussian

In the lecture on Maximum Likelihood Gaussian, the instructor explains the process of setting the mean to capture a set of points in a maximum likelihood Gaussian with a known variant. It is mentioned that the mean of the data is the maximum likelihood mean of the Gaussian. The computation of the mean is easy when all the data points come from the same Gaussian distribution, as the mean can be calculated by taking the sample mean. However, when there are multiple clusters, the means need to be set differently. To address this, hidden variables are introduced, which are indicator variables indicating which cluster a data point came from. Although the values of these variables are unknown and require inference, adding them helps break down the problem.

2.2.22 Expectation Maximization

Expectation Maximization (EM) is an algorithm that is similar to K-means in terms of its approach. It involves alternating between two probabilistic calculations: expectation and maximization. In the expectation phase, a probabilistic indicator variable, Z, is used to represent the likelihood that a data element belongs to a specific cluster. This is determined using Bayes' rule and the probability that the data element was produced by the cluster. In the maximization phase, the means of the clusters are computed based on the soft clustering information obtained from the expectation phase. This is done by taking the average value of the variables within each cluster. The EM algorithm is analogous to K-means, as it determines cluster assignments based on likelihood probabilities, rather than strict assignments. Overall, EM aims to improve the probabilistic likelihood of the data over time.

2.2.23 EM Examples

The lecture covers the implementation and visualization of the expectation-maximization (EM) algorithm for clustering. The lecturer demonstrates the algorithm's workings using a dataset with two Gaussian clusters. Initial cluster centers are randomly selected from the dataset. After running an iteration of EM, some points are assigned to one cluster while others remain in the middle with intermediate probabilities. Subsequent iterations shift points to their appropriate clusters. The lecturer explains that EM allows for soft clustering, where points can have a non-zero probability of belonging to multiple clusters. This behavior is seen as a positive characteristic of the algorithm. The lecturer concludes by mentioning that EM can be applied beyond Gaussian distributions and proceeds to discuss other properties of the algorithm.

2.2.24 Properties of EM

The EM algorithm, used for clustering, has several properties. Firstly, the likelihood of the data is non-decreasing with each iteration of EM. However, the algorithm may not necessarily converge. While it cannot diverge since it works with probabilities, it can get stuck in local optima. In such cases, the algorithm can be restarted randomly. Another property is that EM can be applied to any scenario where probability distributions are used, not just Gaussian distributions. Different algorithms can be defined by different probability distributions and by determining the E step and M step. The estimation step is usually harder than the maximization step as it involves probabilistic inference. However, there are cases where the maximization step is harder. Overall, the EM algorithm is a useful and versatile tool.

2.2.25 Clustering Properties

In this lecture, the instructor discusses three desirable properties of clustering algorithms: richness, scale-invariance, and consistency. Richness refers to the idea that for any desired clustering, there exists a distance matrix that will produce that clustering. In other words, all inputs (distance matrices) are valid and any way of clustering could be an output. Scale-invariance means that the clustering algorithm should be invariant to changes in the scale of the distances between points. If the distances are doubled or halved, the clustering should remain the same. This property is analogous to changing units of measurement (e.g., miles to kilometers) without affecting the clustering outcome. Consistency states that if a clustering algorithm produces a certain clustering, shrinking the distances within clusters and expanding the distances between clusters should not change the clustering. In other words, it should not introduce new clusters or merge existing ones based on changes in the distances. The instructor provides examples and explanations to illustrate these properties and emphasizes the importance of considering domain knowledge and the concept of similarity in clustering algorithms.

2.2.26 Clustering Properties Quiz Question

This lecture discusses three variations of the single-link clustering algorithm and explores their properties. The first variation stops clustering when there are n/2 clusters. The second variation stops clustering when the distance between two clusters exceeds a threshold parameter theta. The final variation stops clustering when the distance between two clusters exceeds theta/omega, where omega is the largest pairwise distance in the dataset. Students are asked to determine which variations have the properties of richness, scale-invariance, and consistency.

2.2.27 Clustering Properties Quiz Solution

The lecture discusses three clustering algorithms and their properties. 1. Fixed number of clusters: This algorithm lacks the richness property, as it only allows a fixed number of clusters. It is scale-invariant, meaning that multiplying distances by a constant factor does not affect the resulting clusters. It is consistent, as points clustered together will still be picked up even if they get closer or farther apart. 2. Clusters that are theta units apart: This algorithm is rich, as the number of clusters can vary based on the value of theta. However, it is not scale-invariant, as multiplying distances by theta can change the number of clusters. It is consistent because points clustered together will still be within theta distance even if they get closer or farther apart. 3. Clusters that are theta omega units apart: This algorithm is rich, as the clusters can be adjusted by changing theta. It is scale-invariant because changing the scale of distances does not affect the clusters. However, it lacks consistency, as changing theta and omega can lead to clusters being too small to form connections, disrupting the previous clustering. The lecture concludes by mentioning a final algorithm that achieves a trifecta of having richness, scale-invariance, and consistency, but does not provide further details.

2.2.28 Impossibility Theorem

A clustering algorithm called the trifecta, which consists of three properties - richness, scale invariance, and consistency - cannot exist simultaneously. This impossibility theorem was proven by Kleinberg. It is surprising because these properties individually make sense, but they are mutually contradictory. Although it is disappointing, in practice, people are willing to adapt their notion of clustering to achieve their goals. While clustering cannot be solely relied upon, it is still a powerful tool for understanding data.

2.3 Feature Transformation

2.2.29 What Have We Learned

In this lecture on feature transformation, the speakers summarize the key concepts they have learned. They discuss the definition of feature selection, which involves selecting a subset of features for use in machine learning algorithms. They differentiate between filtering methods, which are simpler and potentially faster, but may overlook important information, and wrapping methods, which are slower but more effective. They also discuss the distinction between relevant and useful features. Relevant features provide information about the classification or regression problem, while useful features help facilitate the learning process. The concept of a Bayes optimal classifier is introduced as the gold standard in machine learning. The speakers also mention the concept of strong and weak relevance, where relevant features can be made weak by duplicating them. They conclude by considering their own relevance and usefulness in the context of the course. The lecture ends by highlighting the importance of feature transformation and hinting at future discussions on this topic.

2.3.1 Introduction

Feature transformation is the process of pre-processing a set of features in order to create a new set of features that is more compact while retaining as much relevant and useful information as possible. It is similar to feature selection, but feature transformation is more powerful and can involve arbitrary pre-processing. Linear feature transformation is a specific type of feature transformation that uses a linear transformation operator to project points in the instance space into a new subspace of smaller dimensionality. The goal is to find a matrix that can transform the original feature space into the new subspace by creating linear combinations of the original features. Feature selection is a subset of feature transformation, where the

pre-processing involves taking a subset of features. In comparison, feature transformation can create new linear combinations of the original features. The lecture will focus on linear transformations that reduce the number of dimensions, with the assumption that a smaller subset of features can capture all the relevant information, helping to overcome the curse of dimensionality.

2.3.2 What Are Our Features

The lecture discusses the use of words as features in machine learning and the problems associated with their use. The instructor explains that although words are a reasonable choice as features, there are issues such as the curse of dimensionality, polysemy (words with multiple meanings), and synonymy (words with the same meaning). The example of the word "car" is used to illustrate these problems. Polysemy can lead to false positives, where documents are wrongly considered relevant, while synonymy can result in false negatives, where relevant documents are overlooked. The lecturer emphasizes that feature selection alone cannot solve these problems. The issues discussed are relevant not only in information retrieval but in any problem involving a large set of features that generate false positives and false negatives.

2.3.3 Words Like Tesla

Feature transformation is a technique used to combine words or features in order to improve the accuracy of identifying false positives and false negatives. By combining words that are highly correlated or related to each other, such as car and automobile, higher scoring documents can be identified. This technique is particularly effective in addressing synonymy, where different words have similar meanings. However, it can also address polysemy, where a word has multiple meanings, by combining features to eliminate ambiguity. This process can be thought of as an unsupervised learning problem, where sets of features are combined to give correct labels and minimize the impact of both polysemy and synonymy. Three specific algorithms will be discussed in the following lectures.

2.3.4 Principal Components Analysis

Principal Components Analysis (PCA) is a linear transformation algorithm that finds directions of maximal variance in a dataset. It is an example of an eigenproblem. PCA finds the direction that maximizes the variance of the data when projected onto that direction. In a simple two-dimensional example, if the data points are projected onto the x-axis or y-axis, the variance spans the space between the minimum and maximum values on each axis. However, if the data points are projected onto a direction that is about 45 degrees, the variance will be higher. PCA also finds a second component that is orthogonal to the first. In this two-dimensional example, there is only one choice for the second component.

2.3.5 Principal Components Analysis Two

Principal Component Analysis (PCA) is a technique used to transform data into a new set of orthogonal dimensions. It has properties that make it useful in various ways. PCA finds directions that maximize variance and ensure they are mutually orthogonal. It also provides the best reconstruction of the original data by finding directions that minimize the L2 error. When projecting data onto a single axis, PCA ensures the distance between points in the original space is minimized. PCA is an eigenproblem, meaning each new dimension obtained through PCA has an associated eigenvalue. These eigenvalues are non-negative and tend to decrease monotonically as more dimensions are added. Thus, the dimensions with the smallest eigenvalues can be discarded, indicating features with the least variance.

2.3.6 Principal Components Analysis Three

Principal Components Analysis (PCA) is a feature transformation algorithm that allows for effective feature selection. By transforming data into a new space, PCA aligns the variance of the data with orthogonal axes. This enables the identification and removal of dimensions with low eigenvalues, as they provide no information in the original space. PCA is a global algorithm that gives the best reconstruction error and identifies important features through their corresponding eigenvalues. Additionally, PCA is a well-studied method with fast algorithms that can handle large and sparse datasets. However, while PCA is useful for reconstruction, it is not clear if it is effective for classification. Removing dimensions with low eigenvalues

may result in the loss of relevant information for labeling. PCA can be likened to a filter method, where irrelevant data is discarded.

2.3.7 Independent Components Analysis

Independent Components Analysis (ICA) is an algorithm similar to Principal Components Analysis (PCA) but focuses on maximizing independence rather than correlation. It aims to find a linear transformation of the feature space where each new feature is statistically independent of each other. The transformation allows the original features (represented as X) to be converted into new features (represented as Y) with zero mutual information. Additionally, ICA strives to maximize the mutual information between the new features and the original feature space, ensuring data reconstruction is possible.

2.3.8 Independent Components Analysis Two

Independent component analysis (ICA) is a technique used in unsupervised learning to find hidden variables by analyzing observable data. The fundamental assumption of ICA is that the hidden variables are mutually independent of each other. A concrete example of this is the blind source separation problem, also known as the Cocktail Party Problem. In this problem, multiple people are speaking simultaneously, and the goal is to isolate and listen to a specific source of interest while separating it from the other sources. This can be modeled using microphones recording the conversations, where each microphone receives a slightly different volume and delay due to their random placement in the room. Despite the non-linear nature of voices and sound propagation, the physics of a small room allow the microphones to be modeled as receiving unknown linear combinations of the sources. ICA aims to recover each source independently from the recorded signals, without losing any information. An example found on the web demonstrates the application of ICA to solve this problem.

2.3.9 Cocktail Party Problem

The lecture discusses the Cocktail Party Problem, which involves using Independent Components Analysis (ICA) to separate mixed sounds into their original sources. The speaker demonstrates this by playing mixed sounds and then playing the separated sources using gramaphone icons. The speaker highlights that machines historically struggled with this task, but ICA is able to solve it. The fundamental assumption of ICA is that the sources of the sounds are statistically independent from each other and are combined in a linear way. The speaker explains that ICA uses mutual information to find independent components while preserving information from the observables. Overall, ICA is a fast and effective method for separating mixed sounds into their original sources.

2.3.10 Matrix

To perform Independent Component Analysis (ICA), a matrix is created to represent samples of sound waves. Each row in the matrix represents a feature in the original feature space, and each column represents a sample at a specific time. The goal of ICA is to find a projection that allows for the recovery of underlying structure, such as individual speakers, that is statistically independent of one another. Mutual information is a measure of how much one variable tells us about another variable, and in this case, the assumption is that the sounds produced by the speakers are independent. Therefore, ICA aims to recover features, in this case, individual speakers, whose sound waves are statistically independent. However, it is important that the new transformation retains a relationship with the original values to avoid losing information. By ensuring high mutual information between the microphones and the candidates for the speakers' voices, the original voices can be reconstructed if the model is true.

2.3.11 PCA vs ICA Question

In this lecture from the course '2.3 Feature Transformation', the lecturer quizzes Michael about the differences and similarities between Principal Component Analysis (PCA) and Independent Component Analysis (ICA). Michael is asked to check off which phrases apply to PCA and ICA from a list provided by the lecturer.

2.3.12 PCA vs ICA Solution

In this lecture, the instructor discusses the differences between Principal Component Analysis (PCA) and Independent Component Analysis (ICA) in terms of their defining properties. It is noted that PCA focuses on mutually orthogonal dimensions whereas ICA does not prioritize orthogonality. PCA aims to find dimensions that maximize variance, while ICA is concerned with finding mutually independent features. It is mentioned that PCA can sometimes find independent projections, but this is not a common occurrence. The lecture also highlights that PCA tends to find uncorrelated dimensions, whereas ICA focuses on statistically independent dimensions. It is mentioned that the distribution that maximizes variance is the normal distribution, explaining the connection between PCA and Gaussian distributions. The lecture then discusses how the goals of PCA and ICA differ in terms of maximizing mutual information and achieving maximal reconstruction. Lastly, the instructor explains that PCA orders the features according to their variance, while ICA does not have a notion of ordering features. It is mentioned that ICA produces a "bag of features" without a specific order. The overall conclusion is that PCA and ICA have different underlying assumptions and optimization functions, despite their common goal of capturing the original data in a new transformed space.

2.3.13 PCA vs ICA Continued

The lecture discusses the differences between Principal Component Analysis (PCA) and Independent Component Analysis (ICA) in the context of feature transformation. It explains that ICA is designed to solve the blind source separation problem and performs well in that regard, while PCA does not. PCA is shown to be less directional than ICA, as it produces the same results regardless of the orientation of the data matrix, whereas ICA produces different results. The lecture provides examples of how PCA and ICA handle different types of data. For images, PCA tends to find the direction of maximum variance, which is often brightness or luminosity, while ICA identifies specific features like noses, eyes, and hair. In the case of natural scenes, PCA focuses on brightness and the average image, while ICA identifies edges as the fundamental underlying components. The lecture emphasizes the importance of understanding the underlying structure of the data for analysis purposes and highlights that once these fundamental features are discovered through techniques like ICA, efficient algorithms can be developed to extract and analyze them further. The lecture also mentions the similar result obtained from applying ICA to information retrieval problems, where it identifies topics in documents. In conclusion, using feature transformation techniques like ICA can facilitate a better understanding of data and allow for more effective analysis and interpretation.

2.3.14 Alternatives

Summary: The lecture discusses two alternatives to PCA and ICA for feature transformation. The first alternative is called Random Components Analysis (RCA) or random projection. RCA generates random directions and projects the data onto these directions. It works well for classification tasks, as it captures correlations between features despite projecting into a lower-dimensional space. The number of lower dimensions used in RCA tends to be larger than in PCA. RCA can even project into higher-dimensional spaces. The second alternative is not discussed in the text. The advantages of RCA over PCA and ICA are not mentioned, but the lecturer poses a question about it.

2.3.15 Alternatives Quiz Question

The lecture posed a quiz question about the big advantage of RCA (presumably "Regularized Component Analysis") and requested a single word to describe it. The lecturer suggested that the word should be obvious and easily noticeable. The question was directed towards Michael.

2.3.16 Alternatives Quiz Solution

The lecture discusses the advantages and simplicity of using ICR, RCA for feature transformation in machine learning. The speaker mentions that ICR, RCA is cheap, simple, and easy to implement. They emphasize that it is also fast, which is the word the speaker was looking for. The speaker finds it irritating that simple algorithms like ICR, RCA can be effective, while they want to come up with more complex approaches. However, the speaker acknowledges that complexity must be earned. They explain that RCA

randomly throws things together and picks up correlations, making it a fast and effective method. In contrast, PCA and ICA can be time-consuming. Generating random numbers is a task that computers excel at, contributing to the speed of ICR, RCA.

2.3.17 Alternatives Two

Linear Discriminant Analysis (LDA) is an alternative approach in feature transformation that aims at finding a linear projection to separate data based on its label. LDA is similar to supervised learning as it utilizes the labels to transform and cluster data. It can be seen as finding linear separators or lines between different clusters of points. In the binary case, LDA can be compared to SVM, where points are projected onto a line to create clusters. Unlike other approaches, LDA explicitly considers the labels and aims to find features that facilitate discrimination. Although LDA does not consider the subsequent learner, it is effective in simplifying classification tasks for easily linearly separable data. The lecture concludes with a mention of Latent Dirichlet Allocation (LDA), a different approach unrelated to linear discriminant analysis discussed in the course.

2.3.18 Wrap Up

In this lecture on feature transformation in machine learning, the instructor reflects on what has been learned about unsupervised learning algorithms. The focus is on several techniques, including Principal Component Analysis (PCA), Independent Component Analysis (ICA), Linear Discriminant Analysis (LDA), and Robust Component Analysis (RCA). It is noted that these techniques are examples of feature transformation. The instructor emphasizes that the 'A' in the algorithms stands for the analysis of data, as unsupervised learning involves finding the underlying structure in the data. The relationship between ICA and the structure of data, such as the independent components being edges in natural scenes, is discussed. The instructor also mentions their personal experience and research with ICA. The differences between PCA and ICA are highlighted, with PCA being more focused on linear algebra and ICA being more probabilistic and rooted in information theory. It is acknowledged that linear algebra approaches are often easier to implement and interpret as probability, but may not always produce the desired answer. In contrast, ICA can be more expensive, may not always find independent components, but when it does, the results are satisfying. The lecture concludes with the instructor mentioning that the mini-course on unsupervised learning is finished and the next topic will be decision problems and reinforcement learning, with a reminder to review the material and complete any associated homework or exams.

2.3.19 Introduction

The lecture introduces information theory and its relevance to machine learning algorithms. It explains that information theory allows us to compare probability density functions of input and output vectors in order to ask interesting questions about their relationships. Two key measures in information theory are mutual information, which measures the similarity between input vectors, and entropy, which measures if a feature has any information. The lecture aims to explore the meaning of these terms, their connection to machine learning, and provide an overview of the field's history.

2.3.20 History

Claude Shannon, a mathematician working at Bell Labs, developed information theory which sought to understand and quantify information. This was motivated by the need to determine how to charge for long distance communications. Shannon was the first person to tackle this problem and define what information is. Information theory has roots in physics, specifically in thermodynamics, where physicists first understood the concept of information. Maxwell's Demon, a famous thought experiment by Maxwell, demonstrated that energy can be converted into information and vice versa. Shannon's task was to send messages and determine their information content.

2.3.21 Sending a Message Question

Assuming we want to send a message from Atlanta to San Francisco, we can consider two types of coins: a fair coin and a biased coin. The fair coin has an equal chance of landing on heads or tails, while the biased coin always lands on heads. After flipping both coins and recording their states, we obtain a sequence

of coin flips for each. The fair coin sequence consists of five heads and five tails, while the biased coin sequence consists of ten heads. To transmit these sequences, we can represent them using binary digits. Assuming we can represent each sequence using ten binary digits, with a zero representing heads and a one representing tails, the fair coin sequence can be represented as a combination of zeros and ones, while the biased coin sequence would consist of all zeros. Now, considering the size of each message, we can determine how many bits are required to transmit the sequences from Atlanta to San Francisco for both the fair coin and biased coin.

2.3.22 Sending a Message Solution

The lecture discusses the concept of communication and information in the context of coin flips. It explains that if the outcome of a flip is predictable, there is no need to communicate it, but if it is random, the result of each flip must be communicated. This communication requires the translation of information, which can be measured by entropy. Entropy is described as the minimum number of yes or no questions needed to predict the next symbol in a sequence. In an example with ten coin flips, one question is required for each flip, resulting in an entropy of one. However, if the coin is unfair and the outcome is always the same, no questions are needed and the entropy is zero.

2.3.23 Sending a New Message Question

The text discusses the transmission of a message composed of four words, A, B, C, and D, using a binary representation. Each letter is represented by two bits. The text then introduces the idea of using a different bit representation to achieve less than two bits per symbol. The question is posed to the reader to think of a new representation that might be more efficient.

2.3.24 Sending a New Message Solution

This lecture discusses the concept of sending a new message in feature transformation. It starts by explaining that having two bits per symbol is necessary to represent bit patterns in a tree. The lecture then introduces a new language and explains that the symbol ”A” occurs 50% of the time. It demonstrates how to represent ”A” with a 0 and then proceeds to differentiate between the symbols ”B” and ”C” using another symbol. Finally, the lecture raises the question of how many questions need to be asked for each symbol in this new language.

2.3.25 Expected Size of the Message Question

The expected message size in the given language can be calculated by adding up the number of bits required to send each letter. In this case, A requires 1 bit, D requires 2 bits, and B and C both require 3 bits each. To calculate the expected message size, add up these values. The answer should be expressed in bits.

2.3.26 Expected Size of the Message Solution

In this lecture on feature transformation, the concept of expected size of the message is discussed. The expected number of bits required to transmit each symbol is calculated by multiplying the probability of seeing that symbol by the size required to transmit it, and then adding up these values for all symbols in the language. This calculation gives an average of 1.75 bits. The concept of variable length encoding is introduced, which explains why some symbols in Morse code are smaller than others. In Morse code, the symbols representing the letters ’e’ and ’t’ are smaller because these letters occur most frequently in the English alphabet. The measure of the number of bits per symbol is called entropy and is mathematically represented by a formula involving the size of the symbol, which is given as the logarithm of the inverse of the symbol’s probability.

2.3.27 Information Between Two Variables

In this lecture on feature transformation, the concept of information between two variables is discussed. The lecturer uses the example of predicting thunder based on the presence of rain to illustrate this concept. The information between two variables can be measured using joint entropy, which quantifies the randomness in both variables together. Joint entropy is calculated using the joint probability distribution between the

variables. The other measure is conditional entropy, which measures the randomness of one variable given the other variable. When two variables are independent, the conditional probability of one variable given the other is equal to the conditional probability of the variable itself. In this case, the joint entropy is the sum of the entropies of both variables.

2.3.28 Mutual Information

Mutual information is a measure of dependence between two variables, denoted as I. It is calculated by subtracting the entropy of x given y from the entropy of y. Mutual information quantifies the reduction of randomness in a variable when knowledge of another variable is known. It is a better measure of dependence than conditional entropy. The lecture suggests referring to Charles's notes for derivations and then provides an example to calculate and understand the values of mutual information.

2.3.29 Two Independent Coins Question

In this lecture, the quiz involves two fair coins, A and B, with no information exchange between them. The goal is to determine the joint probability of AB and the conditional probability of A given B. Additionally, the lecture covers the concepts of entropy for A and B, as well as joint entropy and conditional entropy. Students are encouraged to refer to previous videos and formulas to answer the quiz questions.

2.3.30 Two Independent Coins Solution

In this lecture on 2.3 Feature Transformation titled "2.3.30 Two Independent Coins Solution," the speaker discusses the concept of independence between two events, A and B. They use various formulas to calculate different probabilities and entropies. They determine that the probability of A and B co-occurring is 0.25, the probability of A given B is equal to the probability of A (0.5), and the entropies of A and B are both 1. The joint entropy of A and B is calculated to be 2, and the conditional entropy of A given B is 1. Finally, they calculate the mutual information between A and B and find it to be zero, indicating that the two coins are independent and have no mutual information.

2.3.31 Two Dependent Coins Question

In this quiz, two coins, A and B, are dependent on each other due to a gravitational or external force. If coin A flips heads, coin B will also turn up heads, and vice versa for tails. This means that complete information is transferred between the two coins and they are completely dependent on each other. The goal is to find the joint probability between A and B, the conditional probability, their entropies, the conditional entropies, and their mutual information.

2.3.32 Two Dependent Coins Solution

The lecture discusses the concept of joint probability and conditional probability in the context of two dependent coins. The joint probability is 0.5, as both coins can be either heads or tails. The conditional probability, which is the probability of A and B given B, is 1. The entropy of A and B is both 1. The joint entropy, calculated using a specific formula, is also 1. The conditional entropy, again calculated using a specific formula, is 0. The mutual information between A and B is calculated by subtracting the entropy of A and the conditional entropy of A given 0, resulting in a mutual information of 1. This indicates that the random variable A provides information about the random variable B due to their dependency.

2.3.33 Kullback-Leibler Divergence

Kullback-Leibler (KL) divergence is a distance measure that measures the difference between two distributions. It is non-negative and zero when the two distributions are equal. KL divergence is used in supervised learning to model data to a particular distribution. It can be used as a substitute for the least square formula for fitting data to a model.

2.3.34 Summary

In this lecture on feature transformation, we learned about information theory and its applications in machine learning. We discussed how information can be measured using entropy and explored concepts such as joint entropy, conditional entropy, and mutual information. Lastly, we were introduced to the KL divergence, a distance measure between distributions. This primer on information theory provides a foundation for understanding the rest of the machine learning course. Further resources on information theory can be found in the Comments section.