Advanced Algorithms First Report

Hudson Arney & Carson Willms

11/10/2023

Top Three Algorithms:

1. Gradient Descent
2. Adagrad
3. K Means Clustering

Gradient Descent:

Computational Problem:

- When training a machine learning model, you need an algorithm that is able to efficiently calculate the loss, calculate how to possibly tweak the parameters, for efficient model training. As well, you need an algorithm that is fast, and able to withstand very large models.

Example: Employ gradient descent to solve linear regression. This could be done by adjusting parameters to get the lowest difference between expected and actual values.

Applications for Algorithm:

1. Gradient descent helps to minimize the loss function in a machine learning model. When the model makes one epoch, it uses gradient descent to determine which parameters to tweak and by how much

2. Gradient descent also helps the model converge on the minima for loss. It does this by iteratively updating the parameters with each epoch until the local minima has been reached.

Worst Case:

- O(knd), where k is the number of iterations, n is the number of datapoints, and d is the number of features

Average Case:

- Also O(knd), this doesn't change because complexity of Gradient Descent is proportional to the factors involved (k + n + d).

Other Algorithms for the same problem:

- Momentum
- Adagrad

Comparing & Contrast:

- Momentum is a bit more complex than gradient descent, as it requires you to keep track of momentum, which is essentially a history of the past gradients.
- Adagrad is more complex than Momentum, as it deal with an adaptive learning rate based on the momentum of that variables history.

Difficulty Implementing the Algorithm:

- The Algorithm is fairly easy to implement, however creating a use case will be the difficult part. Understanding how to interpret the results and coming up with good hyperparameters may be challenging.
- We could maybe use a dataset relating to price of something (perhaps housing) to predict the price based on certain attributes (house size).

Entrepreneurial Component

- Intending to use

Adagrad:

Computational Problem:

- Although gradient descent is a good optimization algorithm for the model training process, but it's time complexity suffers when dealing with complex state spaces. Adagrad hopes to solve that by adaptive the speed of learning based on "distance from the optima"
- Example: Training a machine learning algorithm for grabbing hundreds of complex relationships in the financial/economics world, and having a higher time constraint that requires a faster algorithm that gradient descent

Applications for Algorithm:

1. Assess Adagrad's performance when dealing with features of significantly different scales. Measure how Adagard's adaptive learning rates influence convergence in comparison to algorithms with fixed learning rates.
2. Evaluate how well Adagrad performs when dealing with sparse data. Measure its ability to efficiently update parameters for features with limited occurrences

Worst Case:

- In situations where the optimization landscape is highly irregular which may include having deep and narrow valleys or plateaus. The algorithm might face challenges navigating the complex surface of the cost function, leading to a larger number of iterations needed for convergence.
- O(d), where d is the number of dimensions in the dataset

Average Case:

- When dealing with machine learning tasks where features have similar scales, the optimization problem is not plagued by extreme variations. Adagrad's adaptive learning rates can efficiently guide the algorithm towards convergence with fewer iterations.
- Still, O(d), the underlying operations that contribute to time complexity remain linear to the number of parameters

Other algorithms for the same problem:

- Gradient Descent
- Momentum

Compare & Contrast:

- Gradient descent may be preferred for simpler scenarios where implementation simplicity is crucial and manual tuning is less of a concern. The choice between Adagrad and traditional gradient descent typically depends on whether you value a more precise adaptability or a simpler approach for machine learning optimization.
- Momentum introduces a memory-like term to accelerate convergence by smoothing out oscillations, making it effective in scenarios with challenging optimization landscapes. While Adagrad excels in adaptability, Momentum focuses on overcoming oscillations for faster convergence.

Difficulty Implementing Adagrad:

- Implementing Adagrad could be challenging due to the additional complexities associated with managing gradient values for each parameter. Unlike simpler optimization algorithms, Adagrad requires careful handling of these accumulated squared gradients, adding computational and storage overhead. Fine-tuning hyperparameters, such as the initial learning rate and epsilon, adds to the implementation difficulty.

K Means Clustering:

Computational Problem:

Finding groupings of datapoints without a specified classification. An example could be trying to segment out groups of people for certain marketing strategies.

Applications:

1. As stated earlier, using K-Means to segment out different customers for different marketing strategies. This could give companies the ability to customize their ability to advertise. One example may be offering targeted advertisements to specific customers.
2. Identify clusters of performance of certain stats in sports. This could provide valuable insights for coaches and analysts to understand patterns and trends in player performance.

Worst Case:

- O(n^2)

Best Case:

- Theta(n^2)

Other Algorithms:

- Hierarchical Clustering
- DBSCAN

Comparing & Contrast:

- Hierarchical Clustering does not require specifying the number of clusters beforehand. K-Means forms exclusive, non-overlapping clusters, whereas Hierarchical Clustering produces a structured arrangement of clusters that reflects the inherent relationships and similarities within a dataset.
- DBSCAN is well-suited for identifying clusters of arbitrary shapes and can handle noise and outliers effectively. DBSCAN doesn't require predefining the number of clusters, and it can find clusters of varying shapes and sizes based on the density of data points.

Difficulty in Implementation:

- Programming the algorithm steps (initializing centroids, assigning data points, updating centroids, and iterating until convergence) may be quite an initial challenge, but something I think we're capable of. Convergence criteria and optimizing for performance make could be more additional challenges. Testing with a small dataset that is well defined and established would be ideal.

Entrepreneurial Component

- Not intending to use