Exploring Dimensionality Reduction Techniques on MNIST Handwritten Digit Dataset

Rhichard Koh (100842848)

Faculty of Science, Engineering & Information Technology, Durham College

Machine Learning: Clustering and Dimensionality Reduction (COSC 32000)

Dr. Uzair Ahmad

April 14, 2024

Objective

Through a meticulous examination and juxtaposition of several prevalent techniques for dimensionality reduction and feature extraction, including Principal Component Analysis, Autoencoders, and Restricted Boltzmann Machines, I aim to recognize their comparative strengths and weakness for these crucial tasks. I'll apply each method to the MNIST dataset, which consists of 70,000 small grayscale images of handwritten digits from zero to nine. The dataset was split with 60,000 images allocated for training various models, while 10,000 other images ire reserved for subsequently evaluating how effectively those same models could then generalize to novel data instances. Each image is 28 by 28 pixels.

Algorithms Used:

Principal Component Analysis (PCA): PCA is a technique that simplifies complex data. It works by transforming a set of possibly related variables into a new set of uncorrelated variables called principal components. This transformation helps in simplifying the data, reducing irrelevant information (noise), and revealing hidden patterns.

Autoencoders: An autoencoder is a type of neural network used to encode unlabeled data into a smaller dimension while trying to maintain the essence of the original data. The goal is to compress the data and then reconstruct it as accurately as possible. This process helps the network learn what features of the data are the most important.

Restricted Boltzmann Machines (RBMs): RBMs are a type of neural network that learns to understand the probability patterns of its input data without being directly told the correct answer (unsupervised). They are great for recognizing underlying patterns and features in data, which is useful for tasks like reducing data dimensions and feature extraction.

Deep Belief Networks (DBNs): DBNs are advanced models that consist of multiple layers of RBMs, where each layer builds upon the previous one. By training these layers sequentially, DBNs can model complex data hierarchies, making them powerful for tasks that involve learning from data structures at multiple levels, such as recognizing features in images or reducing data dimensions while preserving important information.

Methodology

Data Preparation:

For this experiment, I transformed the MNIST dataset images into 784-dimensional vectors. Each image, originally 28x28 pixels, was flattened and standardized, meaning I divided each pixel by 255. After standardizing the pixels their values would become between 0 and 1.

PCA Configuration:

I experimented with different numbers of principal components to see how they affected the ability to rebuild the original images (reconstruction loss) and the accuracy of a classifier. The configurations I tested included 50, 100, 150, and 200 principal components.

Tools Used:

I used the PCA class from Scikit-learn to perform the principal component analysis. As well as the LogisticRegression class from Scikit-learn for classification. NumPy was used to sample the test set.

Calculation of Metrics:

Reconstruction Loss: I measured reconstruction loss using the mean squared error between the original flattened images and the images reconstructed from the PCA-transformed space.

Classifier Training: I trained a logistic regression classifier on the PCA-reduced feature space to see how ill the reduced data could still classify images. I tested the classifier's accuracy on a random selection of 1000 images from the test dataset.

Results

- `Reconstruction loss for PCA with 50 components: 0.011795084610364161 Classification Accuracy: 0.924`
- `Reconstruction loss for PCA with 100 components: 0.005756633405031867 Classification Accuracy: 0.93`
- `Reconstruction loss for PCA with 150 components: 0.0034891502018314425 Classification Accuracy: 0.924`
- `Reconstruction loss for PCA with 200 components: 0.0022767954728689765 Classification Accuracy: 0.926`

Analysis

Reconstruction Loss: As we increased the number of principal components, the ability to reconstruct the original data improved, as indicated by a consistent decrease in reconstruction loss.

Classification Accuracy: The best minimal classification accuracy was achieved at 100 components. This suggests that using more than 100 components is not ideal and will waste more resources as 100 provides the best balance between compressing the data and maintaining enough information for accurate classification.

These findings highlight a common trade-off in dimensionality reduction, adding more components tends to improve the reconstruction of the original data. However, beyond a certain point, the increase in classification accuracy starts to level off or even decline. This suggests that there are diminishing returns on adding too many components, as they may introduce unnecessary complexity without much benefit.

Discussion

Using 100 principal components for the PCA analysis resulted in the highest classification accuracy, measured at 93%. This indicates that the PCA model was highly effective at retaining essential information from the original dataset during the compression process, which in turn led to less loss reconstructions of the original images. However, an interesting issue arose when generating new images with these 100 components: all 5 images produced a noisy feint number. Several factors might contribute to this phenomenon. Firstly, the PCA could have overfitting features predominantly because of the large number of components relative to the number of classes. Secondly, there might be a many different features and patterns the components are learning from and they are all getting merged. Additionally, the 100 dimensions might not be sufficient to capture the full variability of other digits, which could limit the model's ability to represent a diverse range of digits accurately. To address this, we could consider increasing the number of PCA dimensions to better capture the full spectrum of digit variations. Alternatively, exploring other dimensionality reduction techniques like t-SNE or UMAP might be beneficial. These methods could potentially preserve a greater diversity of the dataset's features, thus enhancing the ability to generate a more varied set of digit images.

Autoencoder Configuration

An autoencoder neural network was used for dimensionality reduction and feature extraction. This network is divided into two main parts: an encoder and a decoder. The encoder's job is to compress the input images into a smaller, specified encoding dimension. Meanwhile, the decoder works to reconstruct the original images from these compressed, encoded representations. This process allows the autoencoder to learn the most important features of the images while reducing their dimensionality.

I conducted experiments using the autoencoder neural network with various encoding dimensions, specifically 50, 100, 150, and 200, to analyze their effects on reconstruction loss and classification accuracy. The autoencoder was trained on the MNIST dataset for 8 epochs using the Mean Squared Error (MSE) as the loss function and the Adam optimizer, set to a learning rate of 0.001. This setup helped us gauge how different levels of data compression influence the network’s performance in reconstructing

images and classifying digits. The architecture of the autoencoder is defined within the Autoencoder class, which distinctly separates the encoder and decoder components. This separation is critical for the functionality of the autoencoder. Moreover, the forward method in this class efficiently handles the forward pass, ensuring the input data is correctly encoded into a compressed format and then decoded back to its original form, facilitating both learning and performance evaluation.

Results:

- `50: Reconstruction Loss: 0.9358660034152236, Classification Accuracy: 0.8038`
- `100: Reconstruction Loss: 0.9164198400623509, Classification Accuracy: 0.8621`
- `150: Reconstruction Loss: 0.9233703469035468, Classification Accuracy: 0.8531`
- `200: Reconstruction Loss: 0.9157706576623896, Classification Accuracy: 0.8583`

Analysis

The results from the autoencoder experiments demonstrate that the encoding dimension significantly influences both the reconstruction loss and the classification accuracy. Generally, a higher encoding dimension improves classification accuracy because the autoencoder can capture more detailed features of the input images. However, once we pass a certain size we get diminishing returns. This comes with a trade-off: a higher reconstruction loss and efficiency loss. This happens because the autoencoder may overfit the 10 classes into its many components.

Choosing an encoding dimension of 100 has proven to offer a good balance. With this setting, the autoencoder achieved a classification accuracy of 86%, indicating that it retains sufficient detail to accurately classify the images while also being efficient in terms of data compression. This balance suggests that a 100-dimensional encoding is effective for maintaining essential information needed for tasks like image classification without excessive data loss.

Discussion

The results suggest that using an encoding dimension of 100 offers the best classification accuracy for the autoencoder. Although the images produced by the autoencoder were not as sharp as those generated by PCA, they showcased a greater diversity and covered a broader spectrum of digits. This indicates that the autoencoder can extract more significant features from the data, which contributes to its enhanced classification

performance. This ability to capture a wide array of important features suggests that autoencoders, while perhaps sacrificing some image clarity, are better at distinguishing between different types of data, leading to better overall results in tasks requiring categorization and recognition.

RBN Configuration

The RBM (Restricted Boltzmann Machine) class incorporates several key functions essential for its operation. These include functions for sampling the hidden (`sample_h`) and visible (`sample_v`) layers, calculating the free energy crucial for the RBM's training process, and performing a forward pass that generates the reconstructed visible layer. This forward pass plays a critical role in determining the reconstruction loss during training, which helps in assessing the RBM's performance. In terms of initial setup, the weights (`W`) of the RBM are initialized to values very close to zero. This initialization strategy is standard practice for RBMs and is crucial for avoiding overly large activations at the start of training. Additionally, the biases for both the visible and hidden layers are set to zero. This setup ensures that the RBM begins learning without any initial bias, allowing the training process to adaptively develop its understanding of the data structure based on the input it receives.

For training Restricted Boltzmann Machines (RBMs), data preparation is a crucial step. The input data is first flattened, meaning it is transformed from a multi-dimensional array into a single-dimensional vector. It is then clamped to values between 0 and 1 and subsequently binarized. This conversion to binary data is typical and essential for RBMs, as they are generative models that are particularly effective at handling binary input. The training process for an RBM utilizes Stochastic Gradient Descent (SGD) with a learning rate of 0.01. During training, the RBM computes the free energy for both the original data and the reconstructed data from the model. This computation is used to determine the loss, which in turn guides the updates made to the model's weights. This method ensures that the RBM learns to reconstruct the input data accurately while discovering significant underlying patterns in the dataset. Post-training evaluation involves extracting the hidden representations from the RBM and using them to train a SoftMax classifier. This step is critical as it assesses how effectively the RBM's extracted features can be used for classification tasks. It reflects the quality of the representations learned by the RBM, indicating how well the model can differentiate between various classes based on the features it has identified and extracted during training.

Results

- `50: Reconstruction Loss: -19.923210117608498, Classification Accuracy: 0.8439`

- `100: Reconstruction Loss: -17.291857076860442, Classification Accuracy: 0.8758`
- `150: Reconstruction Loss: -15.531131760652132, Classification Accuracy: 0.8845`
- `200: Reconstruction Loss: -14.382853977715792, Classification Accuracy: 0.8889`

Analysis

Reconstruction Loss: In general, adding more hidden units to a Restricted Boltzmann Machine (RBM) enhances its ability to model complex patterns within the data, typically leading to a reduction in reconstruction loss and an increase classification accuracy. However, when the model is expanded to 200 hidden units, there is an observed increase in reconstruction loss. This could indicate potential issues such as overfitting, where the model becomes too tailored to the training data, or it might suggest that there are insufficient training epochs to adequately train larger models.

Classification Accuracy: As the number of hidden dimensions in the RBM increases, there is a noticeable improvement in classification accuracy, which peaks at 200 hidden units. From 150 to 200 hidden units, there is only a marginal increase in accuracy. This pattern of diminishing returns suggests that increasing the number of hidden units beyond 150 offers little benefit in this specific setup, likely because the model has reached a threshold beyond which additional complexity does not translate into significant performance gains.

Discussion

At 150 hidden dimensions, the images generated by the RBM were reported to be superior to those produced by PCA and an autoencoder. This outcome highlights the RBM's proficiency in capturing crucial features that are beneficial not only for accurate reconstruction but also for generating new, diverse, and interpretable images of digits. The advantage of the RBM in this context might stem from its binary training approach and its stochastic nature, which could allow it to capture more subtle and nuanced features of the digits. In contrast, methods like PCA and autoencoders might lose some detail by averaging out features during their processes. The enhanced interpretability and diversity of the images at 150 hidden dimensions indicate an optimal balance within the RBM's configuration. This "sweet spot" suggests that the model is complex enough to capture significant and meaningful features without being overly complex, which could lead to poor generalization, or too simplistic, which might cause it to overlook finer details in the data. This experiment underscores the effectiveness of RBMs in tasks related to feature extraction and image generation. It also illustrates how architectural decisions, such as the

choice in the number of hidden units, can have a profound impact on the model's performance in practical applications like digit classification and image generation.

DBN Configuration

I implemented a Deep Belief Network (DBN) using TensorFlow and Keras to perform dimensionality reduction and feature extraction on the MNIST dataset. The network is structured as a sequence of layers including dense (fully connected) layers and dropout layers to manage overfitting. The MNIST dataset was preprocessed by reshaping the images into 784-dimensional vectors (corresponding to the 28x28 pixels of each image), normalizing the pixel values to a [0, 1] scale, and encoding the labels into one-hot vectors. This standard preparation ensures that the model inputs are uniform and suitable for training in a neural network. The model is dynamically built with tunable hyperparameters including the number of units in each dense layer and the dropout rate. I utilized a RandomSearch tuner from Keras Tuner to explore different configurations:

- **Units**: Number of neurons in each layer tested at [50, 100, 150, 200].
- **Dropout Rate**: Proportion of neurons to drop out during training tested from 0.0 to 0.5 to prevent overfitting.

Each dense layer uses ReLU activation to introduce non-linearity, ensuring the model can learn complex patterns. The final layer uses softmax activation to output probabilities across the 10 classes of digits. The model was compiled with the Adam optimizer and categorical crossentropy loss, which is typical for multi-class classification tasks. It was trained on the training data with validation on a separate set to monitor performance and prevent overtraining. The RandomSearch tuner ran for a maximum of 10 trials to identify the best combination of units and dropout rate, based on validation accuracy. This process ensures that the model configuration is optimized for the given data.

Results

```
Top 1 Trial:
Hyperparameters: {'units': 50, 'dropout_rate': 0.4}
Objective (e.g., Loss): 0.9639000296592712
Best Validation Loss: 0.12775953114032745
Best Validation Accuracy: 0.9639000296592712


Top 2 Trial:
Hyperparameters: {'units': 50, 'dropout_rate': 0.30000000000000004}
Objective (e.g., Loss): 0.9678999781608582
Best Validation Loss: 0.10606943070888519
Best Validation Accuracy: 0.9678999781608582
```

```
Top 3 Trial:
Hyperparameters: {'units': 50, 'dropout_rate': 0.2}
Objective (e.g., Loss): 0.970799982547776
Best Validation Loss: 0.09899251908063889
Best Validation Accuracy: 0.970799982547776


Top 4 Trial:
Hyperparameters: {'units': 100, 'dropout_rate': 0.4}
Objective (e.g., Loss): 0.9760000109672546
Best Validation Loss: 0.08561421930789948
Best Validation Accuracy: 0.9760000109672546


Top 5 Trial:
Hyperparameters: {'units': 100, 'dropout_rate': 0.30000000000000004}
Objective (e.g., Loss): 0.9763000011444092
Best Validation Loss: 0.08270733803510666
Best Validation Accuracy: 0.9763000011444092


Top 6 Trial:
Hyperparameters: {'units': 100, 'dropout_rate': 0.2}
Objective (e.g., Loss): 0.9778000116348267
Best Validation Loss: 0.07927803695201874
Best Validation Accuracy: 0.9778000116348267


Top 7 Trial:
Hyperparameters: {'units': 150, 'dropout_rate': 0.4}
Objective (e.g., Loss): 0.9781000018119812
Best Validation Loss: 0.07377926260232925
Best Validation Accuracy: 0.9781000018119812


Top 8 Trial:
Hyperparameters: {'units': 200, 'dropout_rate': 0.4}
Objective (e.g., Loss): 0.9786999821662903
Best Validation Loss: 0.07289712131023407
Best Validation Accuracy: 0.9786999821662903


Top 9 Trial:
Hyperparameters: {'units': 150, 'dropout_rate': 0.30000000000000004}
Objective (e.g., Loss): 0.9787999987602234
Best Validation Loss: 0.07296411693096161
Best Validation Accuracy: 0.9787999987602234


Top 10 Trial:
Hyperparameters: {'units': 200, 'dropout_rate': 0.30000000000000004}
Objective (e.g., Loss): 0.9815000295639038
Best Validation Loss: 0.0684111937880516
```

```
Best Validation Accuracy: 0.9815000295639038
```

Analysis

Reconstruction Loss: In the context of tuning neural networks for classification, particularly when employing techniques like dropout and varying the number of hidden units, the objective is often to minimize validation loss, which corresponds to the model's generalization capability on unseen data. The results show a general trend where increasing the number of hidden units tends to decrease the validation loss, indicating improved model capacity to capture complex patterns in the data. Notably, the best validation loss is observed with 200 hidden units and a dropout rate of 0.3, achieving a loss of 0.0684111937880516. However, the minimal differences in loss between models with 150 and 200 units suggest a potential onset of diminishing returns, where further increases in units do not yield proportional decreases in loss. This could be an indication of overfitting, particularly if the model is excessively tailored to the training data, or it might also suggest the network architecture has reached its practical limit of complexity for the given task.

Classification Accuracy: The classification accuracy improves as the number of hidden units increases, peaking at 200 hidden units with a classification accuracy of 98.15%. This improvement aligns with the increased capacity of the network to model more complex relationships in the data as more computational units (neurons) are available. However, the slight increment in accuracy moving from 150 to 200 units indicates diminishing returns, where additional gains in accuracy are marginal despite increased model complexity. This pattern suggests that while the network benefits from more hidden units up to a point, further increases beyond 200 units might not provide significant advantages and could lead to more complex issues such as overfitting, where the network might perform exceptionally well on training data but less so on new, unseen data.

Discussion

At 200 hidden dimensions, the images generated by the Deep Belief Network (DBN) demonstrated superior quality compared to those produced by Principal Component Analysis (PCA), an autoencoder, and a Restricted Boltzmann Machine (RBM). This advantage underscores the DBM's efficacy in capturing essential features that not only contribute to accurate reconstructions but also enhance the generation of new, diverse, and interpretable images of digits. The DBM's edge in this scenario likely derives from its binary training approach and stochastic nature, enabling the capture of more subtle and nuanced features of the digits. In contrast, methods like PCA and autoencoders may diminish some details by averaging out features during their computational processes. The improved interpretability and diversity of images at 200 hidden dimensions suggest an

optimal balance within the DBM's configuration, hitting a "sweet spot" that allows the model to capture significant and meaningful features without being overly complex which could impair generalization or too simplistic, potentially overlooking finer details in the data. This experiment highlights the effectiveness of DBMs in tasks related to feature extraction and image generation, showcasing how architectural choices, such as the number of hidden units, critically influence the model's performance in practical applications like digit classification and image generation.

Conclusion

In this study, we conducted a comprehensive evaluation of different dimensionality reduction and feature extraction techniques, applying them to the MNIST dataset to ascertain their effectiveness in terms of reconstruction loss and classification accuracy. The techniques assessed included Principal Component Analysis (PCA), Autoencoders, Restricted Boltzmann Machines (RBMs), and Deep Belief Networks (DBNs), each providing unique insights into data handling and feature representation.

The findings reveal that PCA, with an optimal number of 100 principal components, provides a good balance between data compression and the preservation of essential information, achieving a high classification accuracy of 93%. Autoencoders showed potential in capturing a broad spectrum of features, evidenced by their best performance at a 100-dimensional encoding which offered an 86% classification accuracy. Meanwhile, RBMs demonstrated their capability to capture intricate data patterns, with 150 hidden units optimizing both reconstruction quality and interpretability of digit images. DBNs, especially configured with 200 hidden units, excelled in capturing detailed and nuanced features, which translated into the highest classification accuracy of 98.15% among the models tested.

These results underscore the importance of choosing the appropriate dimensionality reduction and feature extraction techniques based on the specific requirements of the task at hand. While PCA is efficient for applications demanding high accuracy with moderate computational resources, Autoencoders and RBMs are better suited for tasks requiring detailed feature extraction and data representation. DBNs, with their complex architectures, offer superior performance in tasks that benefit from deep, nuanced understanding of the data.

Overall, this comparative analysis not only highlights the strengths and weaknesses of each method but also suggests that no single approach fits all scenarios. Instead, the choice should depend on the specific characteristics of the data and the performance requirements of the application. This study provides a foundational reference that can aid

practitioners in selecting the most suitable technique for their data-intensive tasks, enhancing both the efficiency and effectiveness of their models.