Machine Learning A
Home Assignment 5

Jakob Schauser, pwn274

October 2021

1 Principal Component Analysis

1.1 PCA and preprocessing

1.1.1 By the book

b) In question part b), we were asked to qualitatively wonder what an unruly scaling of a single axis would be doing to a PCA. It is clear, that any completely uncorrelated cloud of data can be made to look to have a linearly correlation given a scaling in an axis. Also, post-PCA, a scaling in the axis of least correlation can transform its fluctuations into the axis with the most variance.

c) The exact purpose of whitening is to minimize the covariance. By moving all your data into an uncorrelated 'cloud', the PCA will of course have no obvious axes to project onto. It is best explained by this direct quote from the book: "There is no use doing PCA after doing whitening, since every direction will be on an equal footing after whitening."

1.1.2 In the center

Let $\vec{S}$ be a $d \times d$ matrix. Centering the data contained herein, corresponds to subtracting the mean of every row from every element. We are loosing a degree of freedom, effectively loosing a dimension in the matrix-rank. As an $d \times d$ matrix at most can have a rank of d, this amounts to $\vec{S}$ having a rank of $d - 1$ or less. This can also be seen, as, per definition, the sum of all columns will be 0, meaning that reduced echelon form undoubtedly will have a zero-column.

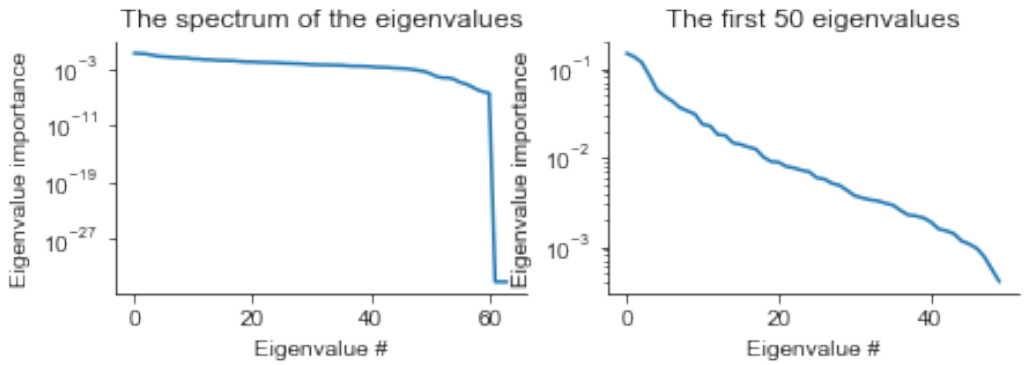

Figure 1: Eigenspectrum of the handwritten digits data. On the right, only the first 50 are shown, as the final eigenvalues are so off-the-charts low, that any dynamics in the first eigenvalues are next to impossible to see.

1.2 PCA in practice

By the virtue of the sklearn python module, the PCA is simply done by the following snippet using the imported data X:

```
pca = PCA()
pca.fit(X)
```

As the package already has a function where the explained variances are returned sorted and as a percentage of the total variance. Finding the cumulative therefore only required this one-liner:

```
cumvar = np.cumsum(pca.explained_variance_ratio_)
```

Finding the 10'th element in this list thereby gives the variance explained by the first 10 elements. This is about 73.8%.

Plotting the eigendigits is also relatively easy, and is done as so:

```
axs[i].imshow(pca.components_[i].reshape(imshape))
```

The results can be seen in figure 3.

1.2.1 Explained variance

The explained variance is shown in Fig. 1.

Ten components are not enough to explain 80% of the variance. This can be seen from figure 2.

Lastly, the eigendigits can be seen in figure 3.

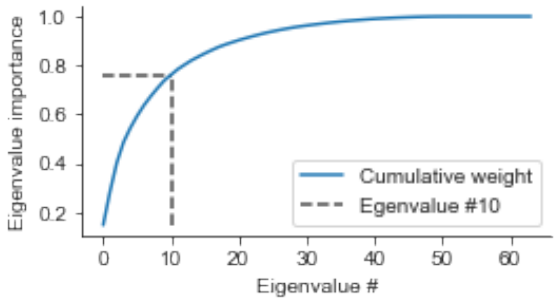

Figure 2: It is clear, that more than 10 eigenvalues are needed to account for 80% of the variance

Figure 3: The first five eigendigits.

2 Logistic Regression in PyTorch

2.1 Logistic regression in PyTorch

First, I completed the definition of the logistic regression model:

```
class LogisticRegressionPytorch(nn.Module):
    def __init__(self, d, m):
        super(LogisticRegressionPytorch, self).__init__()
        # LAYER DEFINITION ADDED HERE
        self.linear = nn.Linear(d,m)
    def forward(self, x):
        # RETURN VALUE ADDED HERE
        return self.linear(x)
```

I then defined the loss function:

```
# DEFINITION OF LOSS FUNCTION ADDED
loss_fn = nn.CrossEntropyLoss() # computes softmax and then
   the cross entropy
```

The optimization and gradient descend loop was then written:

```
for epoch in range(no_epochs):  # Loop over the dataset
   multiple times
```

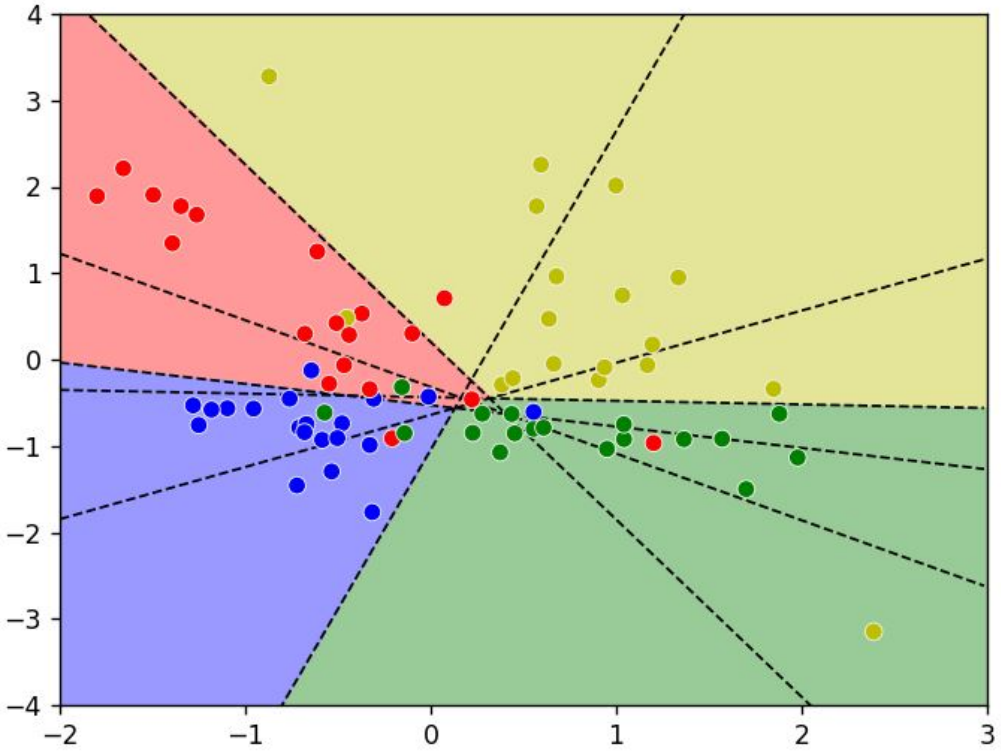

Figure 4: The final plot of the classifications

```
    # Zero the parameter gradients
    optimizer.zero_grad()

    # Forward + backward + optimize
    outputs = logreg_pytorch(X_train_tensor)
    loss = loss_fn(outputs, y_train_tensor)
    loss.backward()
    optimizer.step()
```

The parameters of the linear cuts were extracted as follows:

```
# SOMETHING ADDED HERE
params = []
for param in logreg_pytorch.parameters():
    params.append(param.detach().numpy())

ws_torch = params[0]
bs_torch = params[1]
```

The final classification can be seen in figure 4.