Inlamning 4

Umut Arslan

2023-03-13

Fashion data

We will start by loading the fashion data onto r and convert it to a data frame. Converting the lable column to a factor, this step is important because the SVM model requiers the labels to be categorical variabels, this is my y variable.

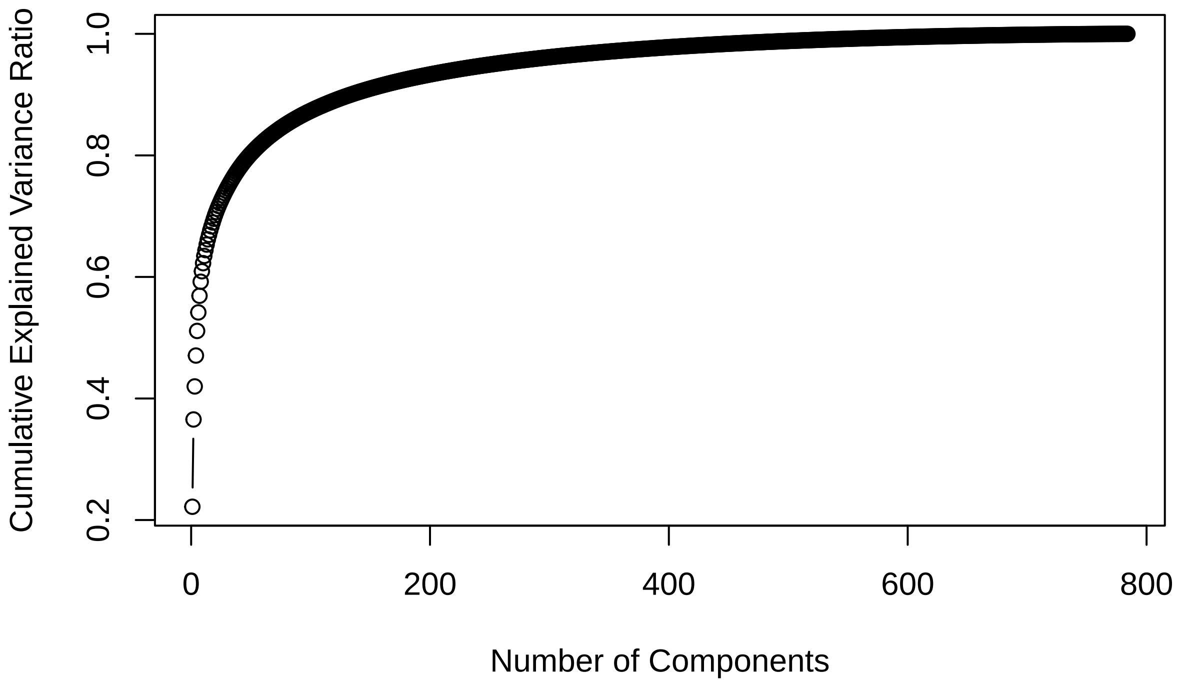
We will then standardize the features, x variabels, we standardize to ensure tha all of the features are on the same scale.

We will train the fashion data with 3 diffrent methods, PCA + KSVM, NN and CNN.

PCA

We will use PCA to reduce the dimensionality from 784 dimensions to n_components. The n_components is determined by the cumulative explained variance ratio against the numbers of components. I chose to use the number of components needed to explain ATLEAST 85% om the total variance. After we have calculated the n_components, we will apply the n_components from the PCA model into my KSVM model, kernal support vector machine.

I have plotted the cumulative explained variance ratio against the numbers of components down below, you could argue that after the first 8 components, the difference in the explained variance is minuscule, so you could probably chose 65-70% as the threshold to reduce alot of dimensions. The risk increases of overfitting the training-model when you are choosing more PCA-components. In other words, i have taken a higher risk of overfitting the model when i chose 85%, because my n_components = 79 PCA-components.



SVM

There are two ways to compile an SVM model, u can use the package e1071, svm(x, y, type, kernel, c, gamma) or u can use another package, kernlab, ksvm(x, y, type, kernel, kpar, and c)

- x = the X variabel, förklarinsgvariabel
- Y = the y variabel, Responsvariabel
- type = what type of calculation you want to do, classification or regression. You can see more if u type, i am using C-svc ?ksvm
- kernel computes the inner product in the feature space between tow vector arguments. rbfdot, polydot, vanilladot, tanhdot etcetra is some of the kernal functions inside of the ksvm model

*kpar is the list of hyperparameters, kernel parameters. See in ?ksvm for more info, we only used sigma in my test. Sigma controls the width of the distribution, smaller value equals to sharper peak and larger sigma gives a broader peak.

- C is cost of constraints violationviolation,controls the trade-off between achieving a low training error and a low testing error.

We will use my 79 PCA-components combined with my training labels to train the KSVM model. I also used

```
##         X1000 X5000 X10000 X30000
## elapsed  0.82 18.19  68.66  68.66
```

Above we can see the time it takes to compile the model based on how many vectors we have, the time is in seconds. All of the models above got an training error = 0, when using mini-batches, there are some backdraw of using mini-batch instead of using the whole dataset, the stochastic nature of mini-batching may result in noisy gradient estimates and the model can fluctuate quite alot. You will save alot of time tho. We could also have gotten training error = 0 becaus of overfitting, its quite hard to know if we dont test the trainingmodel with a testdata.

Neural Network

We have different layers in the model.

- layer_flatten: flattens the image data, in my case i have flattend it to 28x28, 1-d vector.

*layer_dense: this is an hidden layer, we have two hidden layers, a "relu" and a "softmax" activation layer.

After determining the layers, we compile the model by sdpecifiying the loss, optimizer and metrics function

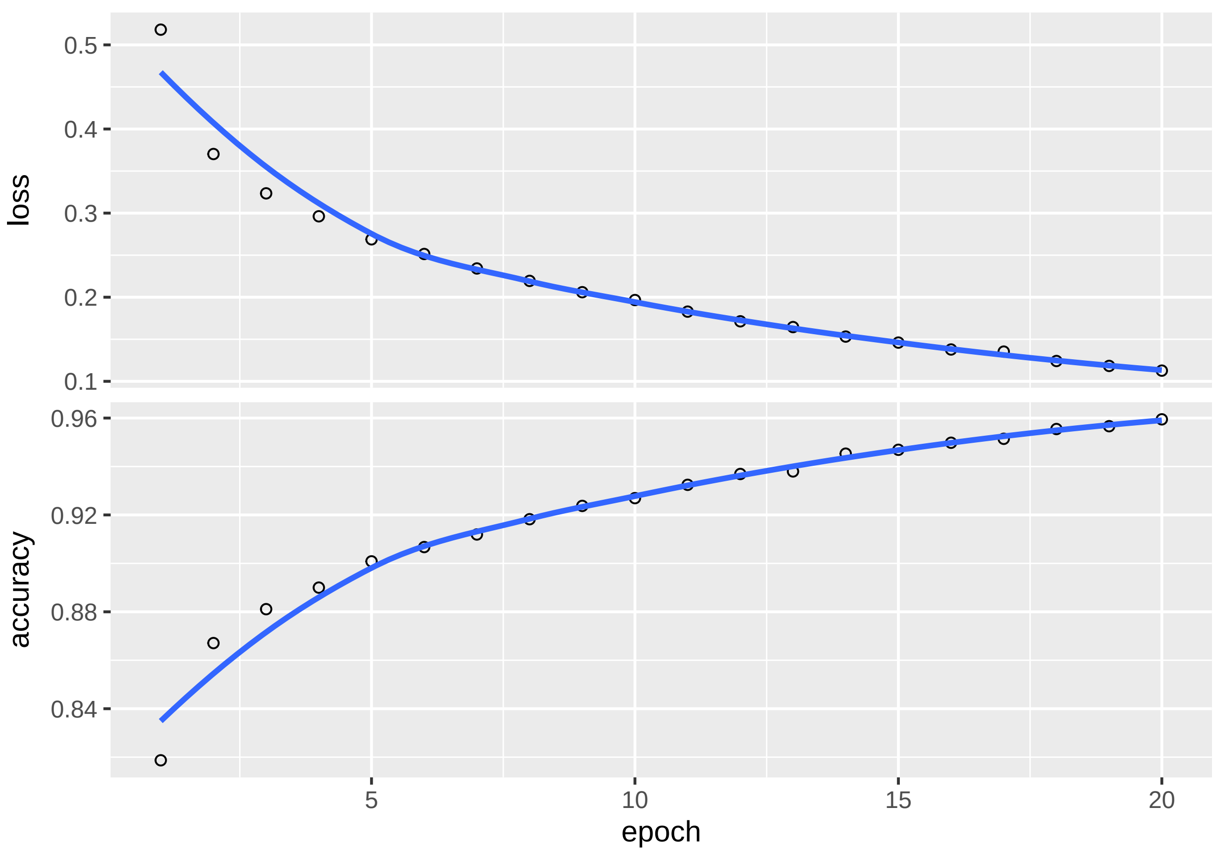
- loss: how well the neural network is able to predict the correct output
- optimizer: updating the weights of the neural network
- metrics: how you want to evaluate the model, i chose accuracy.

lastly we fit the model with the training features and the training lables. we need to transform the lables using to_categorical, the lables were factorial before because of the SVM model.

- epochs: How many iterations
- batch_size: The number of sample per iteration

I will use relu activation = 64, i tried 16, 32, 64 and 128. For me the plot "fitted" the best whitout looking too overfitted. epochs = 20, because it had diminishing return after that batch_size = 32, had the most "regression" like slope, while 64 and 128 had more curvature.

This model is giving us an 4% training error, pretty ok.



Convolution Neural Network

Here we use the same data as the NN-model, but we also reshape our features to a 4D tensor, where the first dimension is the samples, second is the height of the pixels, third is the width of the pixels and last one is t he channels. We also use a 2D convolutional layer instead of a 1D layer, like NN does.

- layer_conv_2d : 2D convolutional layer with filter, kernel size, activation output reLU.
- layer_max_pooling_2d: includes a pool_size, which reduces spatial size of the output from the convolutional layer
- layer_flatten: flattens the output of the max pooling layer to a 1D vector so it can be passed to the layer_dense
- layer_dense: same as on NN, hidden layers, reLU and softmax.

after the layers CNN has the same compilecode as NN

- loss: how well the neural network is able to predict the correct output
- optimizer: updating the weights of the neural network
- metrics: how you want to evaluate the model, i chose accuracy.

lastly, like NN we have the fit of the model which consits of the same features.

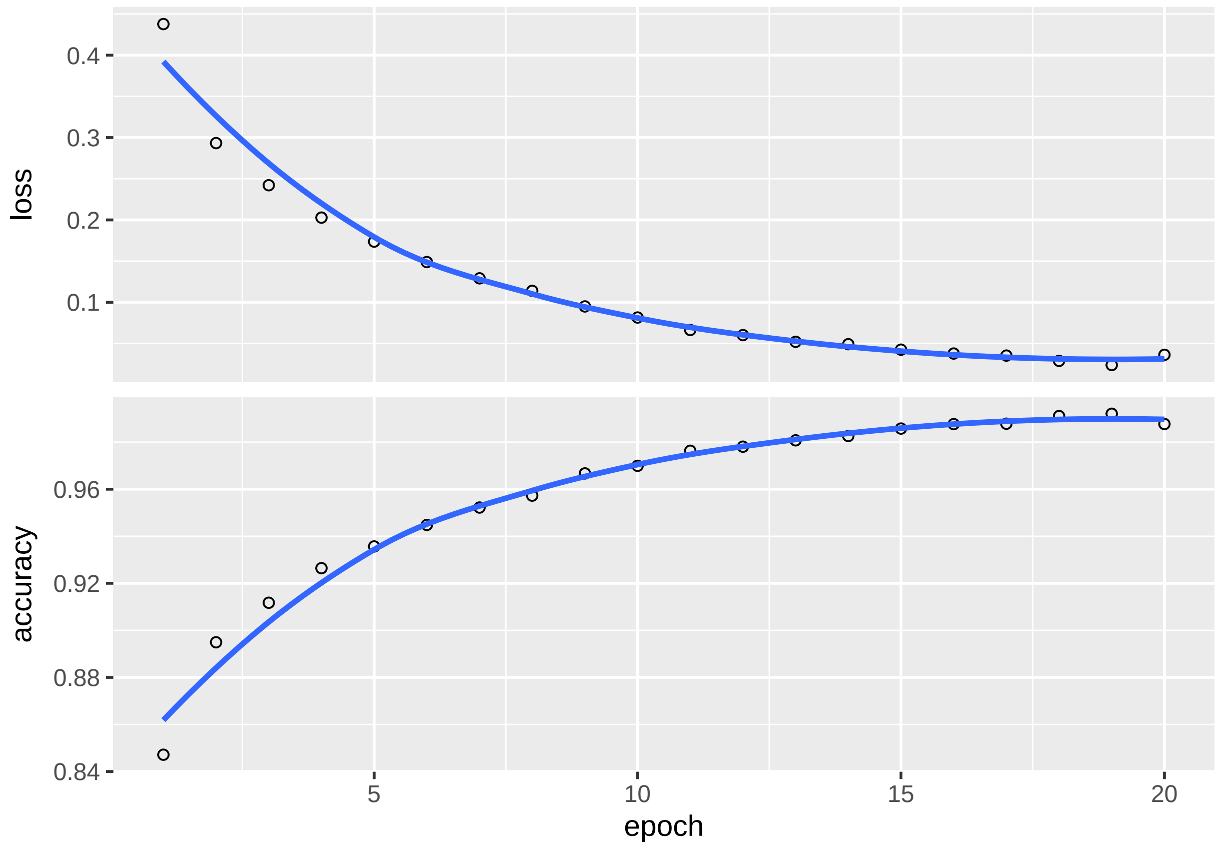
- epochs: How many iterations

- batch_size: The number of sample per iteration

I will mirror the NN-model, reLU = 64, epochs = 20 and batch_size = 32

ps, CNN has all the hyperparameters as NN plus more, like in the conv_2d layer and max_pooling layer.

We got an much lower training error with this model compared to NN, 0.3%. I also plotted the epoches down



below.

```
## [1] 0.007933319
```

Summary

SVM + PCA took the longest time if i used all the datapoints, CNN came second and the fastet one was NN (if you ignore the mini-batches of SVM + PCA)

But if we look at the training error, SVM + PCA got an perfect score, CNN came second and NN came last.

Depending on what you want to do and how you tune the hyperparameters, your mileage may vary.