**Part2**

**1) Introduction**

In this second part of the project, we will train a classifier that is able to classify images by determining if an image contain a human or not.

We will use two models to classify our set of images, Neural Networks and SVM.

**2) Data Description**

The training Data for classification consists in N= 8545 input and output samples. Each input sample has a dimension D= 8545, that represents the HOG (Histogram Of Oriented Gradients) of the corresponding image. So, each image is represented as a vector of dimension D.

The output samples represents labels to each feature vector. The value of 1 means that the corresponding image contains a human, -1 otherwise.

Our goal is to train a model ,using the training Data, that is able to predict the label of a given vector of features (That represents an image).

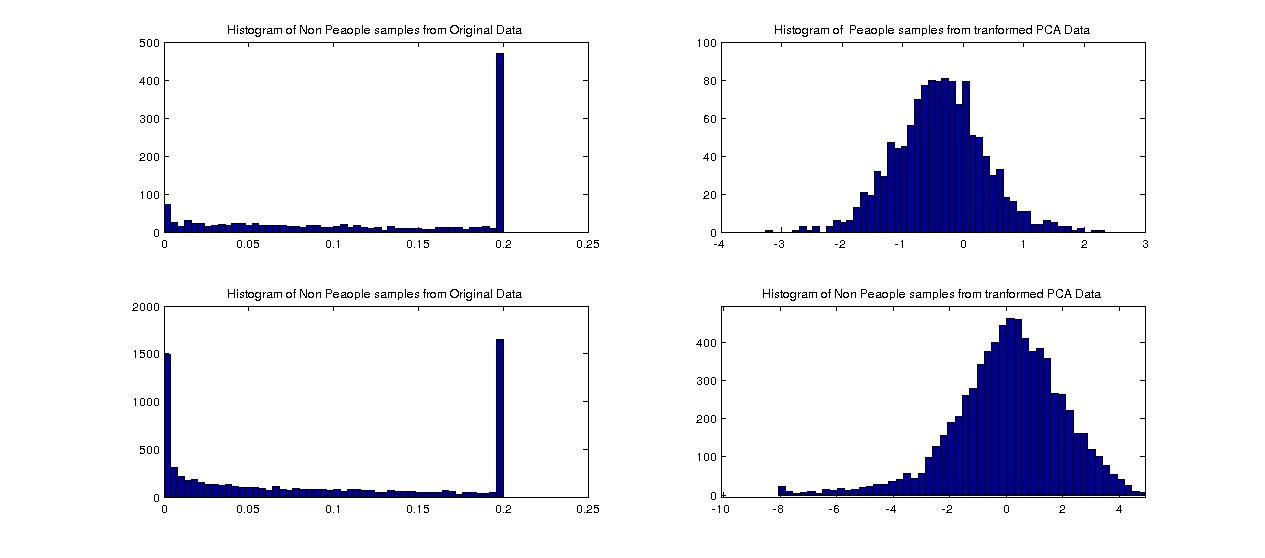
**3) Data Visualization**

An interesting visualization of the data would be to plot the pair of histograms of the dimensions belonging to each class(Label). For example, for dimension1, take all the inputs labeled as 1, and plot the histogram of all the inputs according to dimension 1.

This will help us seeing if each class has a certain distribution along the different dimensions.

Another fact that appears while exploring the data is that the input samples are high dimensional. A dimensionality reduction of the data would probably help in training the models.

Our first intuition was to apply PCA to the Data and see the effect on the histogram of dimensions of each class.



Here we can see the histograms of People and Non People samples over the second dimension taken from the original Data(Left) and from the transformed(with PCA) data. We can clearly see the gaussian distribution on the histograms of each of the class's samples over the second Dimension of the transformed Data.One can use these informations to determine the distribution of each class over each dimension and use them to make predictions.

We will first begin with the recommended methods: Neural Networks and SVM.

**4)Neural Netwroks**

**4.1 Introduction**

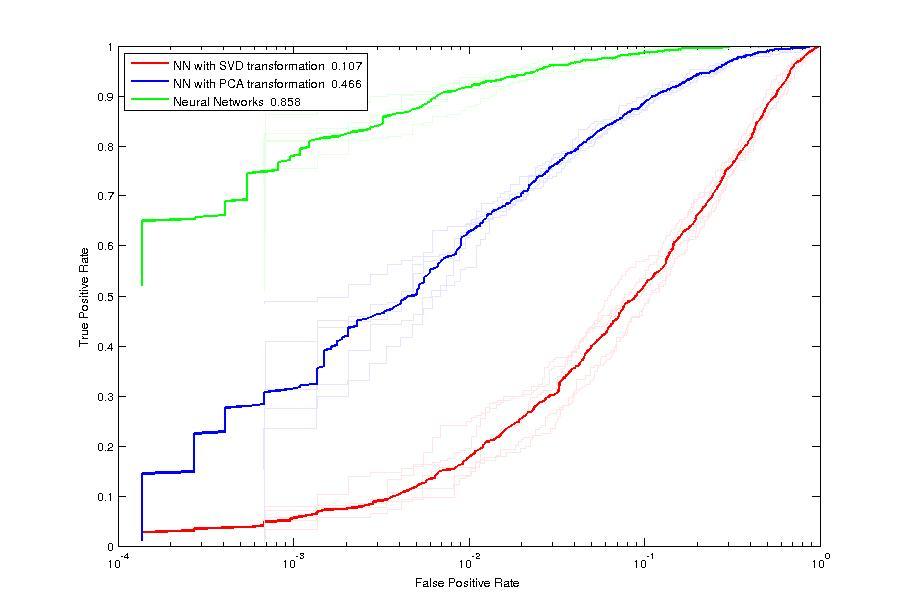
We previously see in the course, models for classification that uses linear combinations of fixed basis functions. Neural networks allow the basis functions to be adaptive by making them depending on parameters.In fact, each basis function is itself a nonlinear function of a linear combination.

This introduces the notions of layers, hidden variables and activation function.

**4.2 Neural Networks and Dimensionality reduction**

As discussed in the previous section, since we have high dimensional data, it’s maybe better if we could train our model on smaller Dimension.

Let’s first, gain some intuition, and see if a reduced Data can help having a better model for Neural Networks. We fix the number of layers, dimension of hidden variables and the activation function. Then, we train our model on all the Data, reduced Data using SVM,and PCA respectively.



We can clearly see that a dimensionality reduction on the data do not improve the performance of our model. So, in the next sections, we will use all the data to train our model.

**4.3 Parameters**

Before training our model, we have first to determine the best parameters for NN, which are:

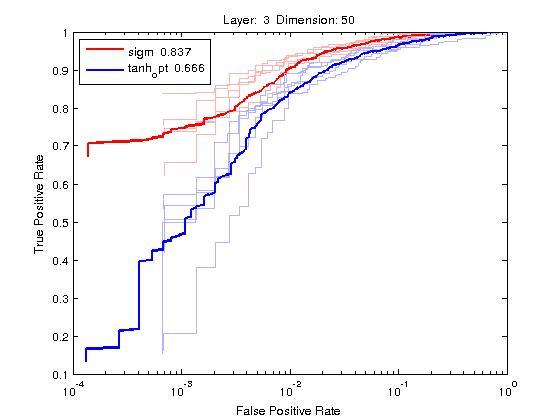
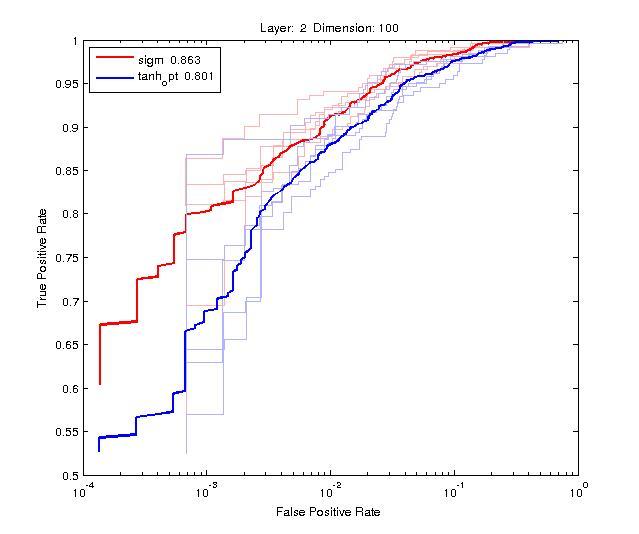
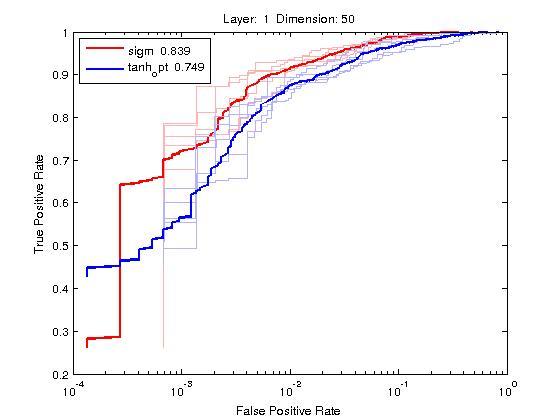
1. The number of layers
2. The dimension of the hidden Variables
3. The activation function (sigm or tanh)

Let’s first gain some intuitions and see the influence of some parameters on the ROC curves.

The first experiment consists in fixing the number of layers and dimension of hidden variables and see the effect of both activation functions on the ROC curves.

Multiple ROC curves are generated using different layers and dimension of hidden variables.

Each curve in the figure represents ROC curve obtained in a specific fold (We use K fold here with K=5)



We can observe that for different layers and dimension of hidden variables, the red curves are alway above the blue ones. Add to that the average TPR for ‘sigm’ is always higher,which means that the sigmoid activation function performs better that the tanh.

We can conclude that we don’t need to iterate over both activation functions when determining the hyper parameters, and fix the activation function to ‘Sigm’

Let’s now do a grid search over the number of layers and the dimension of the hidden variables to determine which configuration is the best.

In order to determine the best score, we average the TPR of each K fold for each configuration and compare them.

The parameters that obtains the top 2 best scores are:

1st Best Score achieved with:

*Layer* = 2 *Dimension* = 100 *activation function* = ‘sigm’

*average TPR over K Fold* = 0.8643 *Variance =* 0.0014

2nd Best Score achieved with:

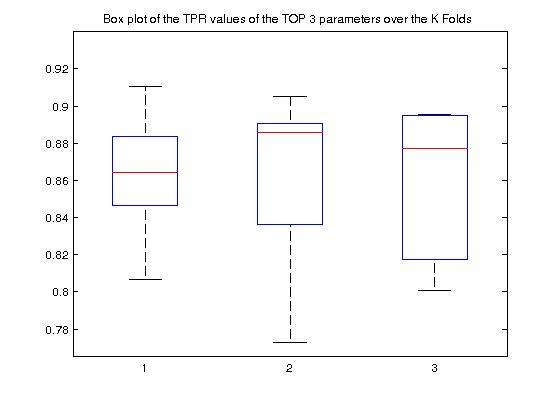
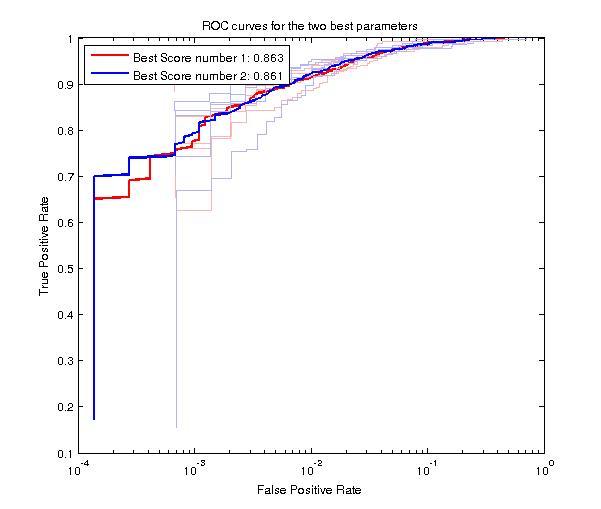
*Layer* = 3 *Dimension* = 120 *activation function* = ‘sigm’

*average TPR over K Fold* = 0.8642 *Variance* = 0.0027

The average TPR of the different parameters are very close to each other. In the other hand the variance over the different Folds differ. A parameter with less variance is better since the average do not vary a lot with respect to the change of the Folds.

All the characteristics of the 3 best parameters are presented in the boxplot below.

The ROC curves of the two models with the best two parameters are displayed below

**2) SVM**

In this section , we will discuss the methods we adopted to choose the best kernels and data transformation on the data in order to have the best Results with SVM.

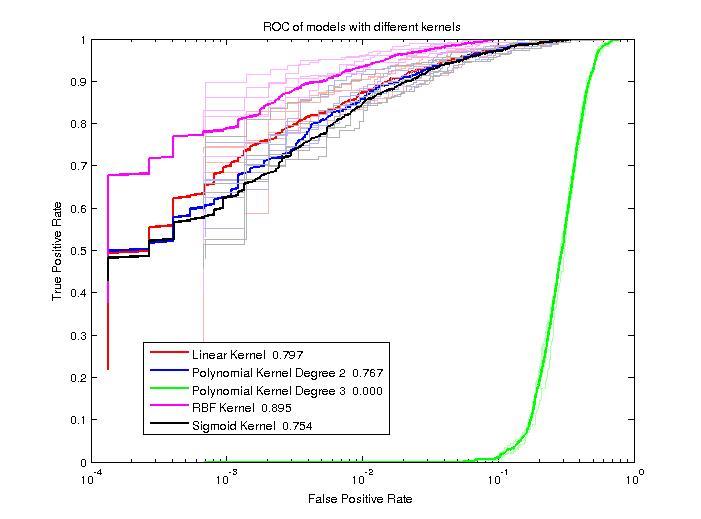
Before searching the best parameters for the SVM model to obtain the best scores. Let’s first gain some intuition about the different models that we can use and the effect of the Data transformation on training our model and the results it gets.

**2.1 Different Kernels**

In this Section, we will train our model with different Kernels over K folds split and plot the corresponding ROC curves.

All the models are trained with a fixed value of C=1 and with K=5.

For the RBF Kernel: we first opted for a small value of gamma(0.02) to have a simple Model



We clearly see that the RBF kernel performs better that the other Kernels since its mean ROC curve (over K Folds) is higher than all the other ones. Add to that, we can see that the average TPR when training our model with the RBF kernel is by far higher that the other values.

We can, then conclude that the RBF kernel is the best of all the kernels. In all the remaining sections, RBF kernel will be used

**2.2 Dimensionality Reduction**

The Dimension of the Data are extremely high, if we would launch a grid search to determine the best parameters of the RBF kernel(gamma and C) on all the Data Set, it would take several hours!

So, In this section, we explore the effect of, reducing the dimension of data by projecting them into new spaces, on the models and the corresponding results **.**

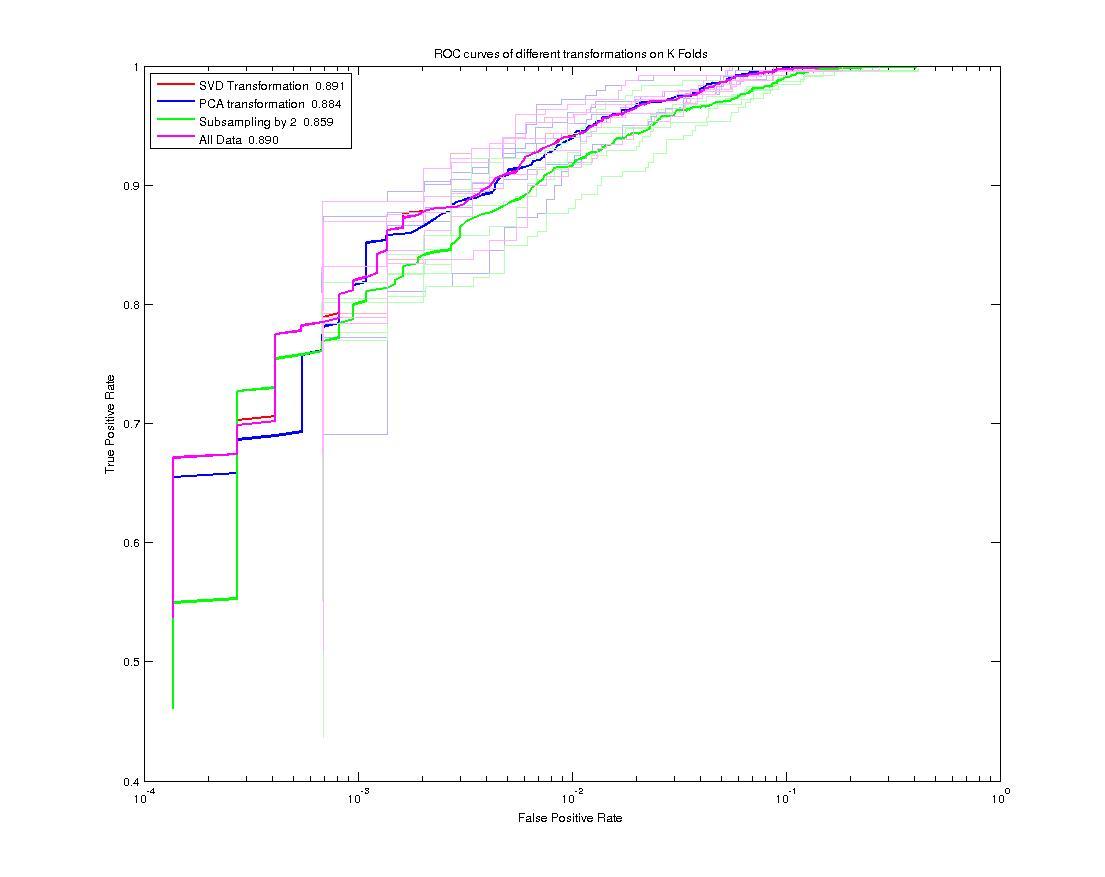
We plot the ROC curves of the RBF model trained over different transformation of the Data.

We first apply SVD on the Data, and take the first 3500 columns of the matrix U\*S as the input matrix ([U,S,V]= SVM(data)).We actually extract the 3500 main concepts represented by the matrix U\*S.

Second, we apply PCA and take the transformation of the Data on the space of the main components(first 1100 main components), as input matrix.

Third, we just subsampled the data by a factor of 2.

The plots show the ROC curves of an RBF model trained over different Transformation of the Data.



We can clearly see that SVD and PCA performs very well. In fact, the ROC curves of the SVD and ‘All the data’ are very close to each other.Also, PCA performs as well as SVD.

We can conclude that we can apply a dimensionality reduction on the data and train our model on smaller dimensions.

**2.2 Determining the parameters**

The next and final step is to determine the best parameters of our model that uses the RBF Kernel. Mainly, the cost C and the bandwidth of the kernel gamma.

We performed a grid search over different values of gamma and C, using the reduced Data (with PCA and SVD). Here are the parameters that obtained the best scores overs K Folds.

For PCA (we take k= 2000 main components) and SVD(we take 3500 main concepts) :

1st Best Score achieved with:

*PCA: Gamma = 0.0162 C=7 average TPR over K Fold* = 0.9077 *Variance =*  0.4384e-03

SVD Gamma = 0.0207 *C=7 average TPR over K Fold* = 0.9027 *Variance =*  0.0014

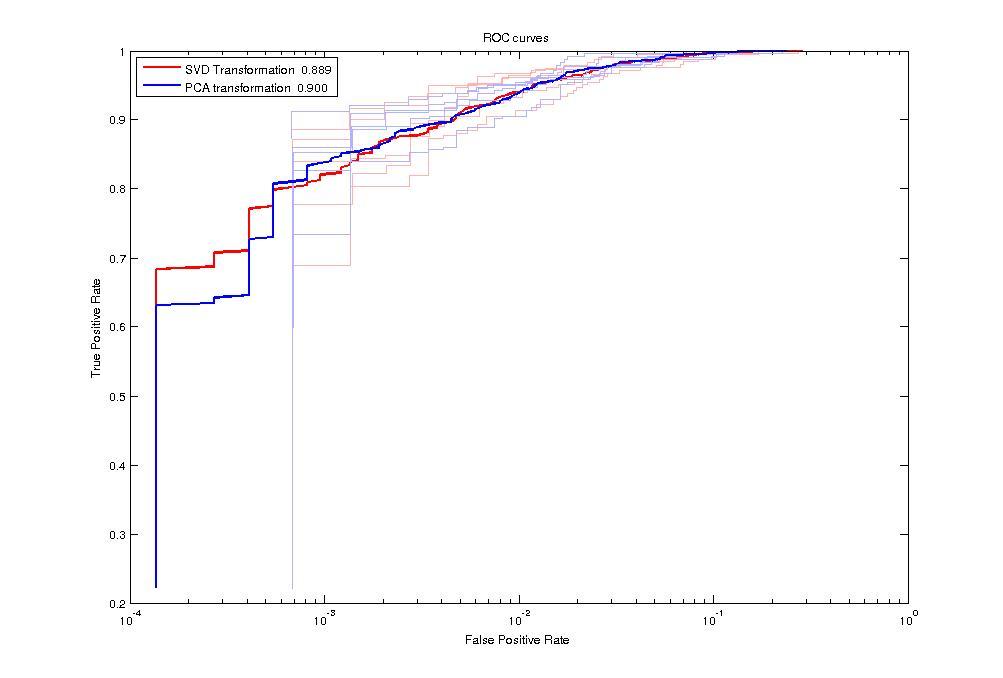
2nd Best Score achieved with:

*PCA: Gamma = 0.0183 C = 7 average TPR over K Fold* = 0.9076 *Variance =*  0.4366e-03

*SVD: Gamma = 0.0234 C=7 average TPR over K Fold* = 0.9025 *Variance =*  0.0014

We can conclude that the PCA performs a little bit better than SVD, after optimizing all the parameters, since we obtain a better average TPR and also a smaller Variance.

The plot below, shows the ROC curves obtained with models trained over different transformations of the data.

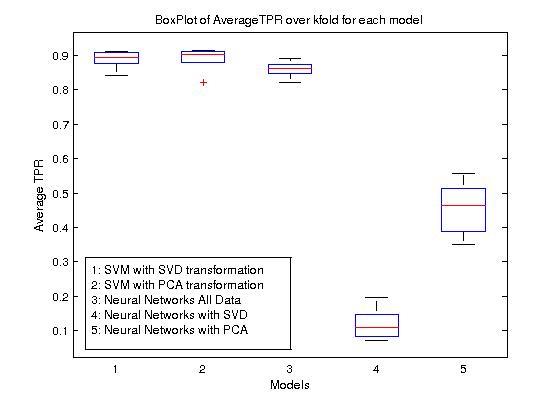
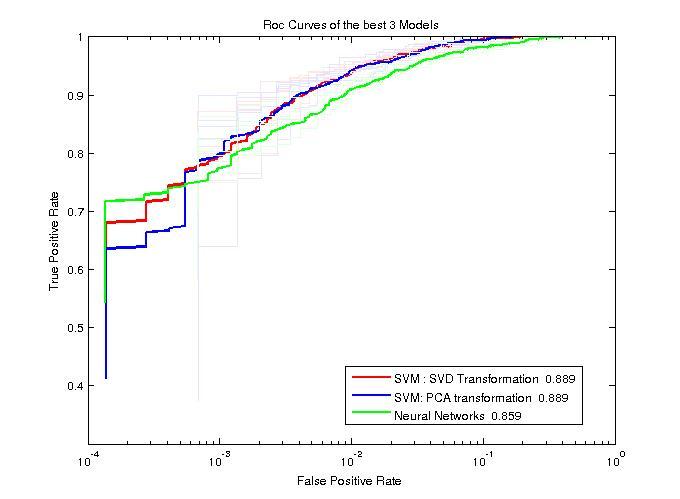


**3) Methods Comparison**

At this point, we have trained two models based on different methods.

Neural Networks uses all the data to find the best model that classify the feature vectors. In the other hand, SVM uses reduced input data to train a model that gives a high score.

Let’s plot the ROC curves and boxplots, obtained from the different Models.



With these two figure, we can finally deduce, that the best model is the one obtained with SVM (with RBF Kernel) on data transformed with PCA. We will use this model for our predictions.